Abstraction In Model Checking Real-Time Temporal Logic of Knowledge

CongHua Zhou
Jiangsu University, Zhenjiang, 212013, China
Email: chzhou@ujs.edu.cn

Bo Sun
Jiangsu University, Zhenjiang, 212013, China
Email: sunbo87@126.com

Abstract—Model checking in real-time temporal logic of knowledge TACTLK confronts the same challenge as in traditional model checking, that is the state space explosion problem. In order to alleviate this problem, we present our abstraction techniques. For the real time part of TACTLK, that is TACTL, we adopt the abstract discrete clock valuations, and in this way the infinite state space of a real time interpreted system can be converted into a finite form. For the epistemic operator K in TACTLK, the definition of epistemic equivalent to an agent between abstract states is given, therefore, the corresponding equivalent relations can be deduced and which can be used to combine the abstract states, as a result, the state space of the real time interpreted system can be further simplified. Finally, we adopt a variant of the standard railroad crossing system to illustrate the effectiveness of our abstraction techniques.

Index Terms—model checking, real-time temporal logic of knowledge, TACTLK, state space explosion, abstraction

I. INTRODUCTION

Model checking[1] is a very important automated verification technique for finite state systems, which has been used in the verification of hardware checking, communication protocols and control systems and has attracted wide attention. In model checking, a multi-agent system $S$ is modelled as a suitable model $M_S$, and the specification $P$ to be verified is represented as a logical formula $\phi$, thus the problem of checking whether a specification $P$ is satisfied by a multi-agent system $S$ is converted into whether the model checking problem $M_S \models \phi$ is valid. Reasoning about knowledge[2] has always been a core in artificial intelligence, thus many specification forms based on modal logic have been proposed and refined in the recent years, and the most popular one is the temporal logic of knowledge, which is a specification language used for modelling and reasoning for multi-agent systems.

However, the verification of temporal logic of knowledge using model checking remains lack for several essential functionalities, and one of them is real-time. In paper[3], A. Lomuscio et al have considered the real-time aspect and proposed a logic to reason about real-time and knowledge in multi-agent systems, that is real-time temporal logic of knowledge TACTLK.

Model checking in real-time temporal logic of knowledge TACTLK confronts the state space explosion problem as in traditional model checking, that is the state space grows exponentially with the increase of the number of concurrent components. As the states satisfying the verified property are searched exhaustively by the model checking algorithm, a too large or infinite state space will severely affect the efficiency of model checking. To alleviate the state space explosion problem, many techniques have been proposed by researchers, such as statute of partial order[4], symmetric statute[5], symbolic computation based on OBDD and abstraction[6-7] et al. Abstraction is one of the most effective methods to alleviate the state space explosion problem, which uses an abstraction function to divide the state space of the original model equivalently and as a result a corresponding abstract model can be obtained, in this way, the information of the original model that is irrelevant to the property to be verified is ignored and the model checking procedure is carried out in the obtained abstract model. As the state space of the abstract model is comparably smaller, the efficiency of verification by model checking can be improved significantly.

In model checking temporal logic, abstraction and abstraction refinement techniques have been studied[7-8], and in real time systems, the abstraction has also been studied[9]. However, as far as we know, none of these techniques has ever been studied in model checking real-time temporal logic of knowledge. Therefore, the abstraction techniques in model checking real-time temporal logic of knowledge have been systematically studied in this paper. Our work is as following: (1). For the real time part of real-time temporal logic of knowledge TACTLK, that is TACTL, the abstract discrete clock valuations[9] is applied to implicitly construct the clock regions of the state space of a real...
time interpreted system, and in this way we can obtain the finite form of the state space of the real time interpreted system. For the epistemic operator $K$ in TACTLK, the definition of epistemic equivalent to an agent between two abstract states is given by us, therefore, the abstract states satisfying the constraints of this definition can be combined into one equivalent class and the state space of the real time interpreted system can be further simplified. (2). Using the abstraction techniques presented above, the corresponding abstract model $M^a$ can be deduced from the original model $M$ of a real time interpreted system, the semantics of TACTLK on the abstract model is given, and we also prove that the satisfaction relations of TACTLK formulae are preserved on the abstract model. (3). The effectiveness of our abstraction techniques is illustrated by the simplification of the state space of a variant of the standard railroad crossing system.

II. THE SYNTAX AND SEMANTICS OF REAL-TIME TEMPORAL LOGIC OF KNOWLEDGE TACTLK

Real-time temporal logic of knowledge TACTLK is used for modelling and reasoning for multi-agent systems, which is the fusion of TCTL representing branching real time[10] and the modal logic[11] S5n representing knowledge operators.

Definition 1. (Syntax of real-time temporal logic of knowledge TACTLK)

Suppose $PV$ is a set of propositional variables containing the symbol $T$ representing the constant true, $A_g$ is a set of $m$ agents, and $I$ is a time interval in $\mathbb{R}$ with integer bounds. Let $p \in PV$, $i \in A_g$, and $\Gamma \subseteq A_g$, then the set of TACTLK formulae is defined as follows:

$$\varphi = p \mid \neg \varphi \mid \varphi \land \varphi \mid A_i \varphi \mid A_A \varphi \mid K \varphi \mid D \varphi \mid E \varphi \mid \forall \varphi \mid C \varphi$$

The formula $A_i \varphi$ represents that for each computation we have $\varphi$ holds until, in the interval $I$, $\psi$ holds; $A_A \varphi$ represents that for each computation we have either $\psi$ holds until, in the interval $I$, both $\varphi$ and $\psi$ hold, or $\psi$ always holds in the interval $I$; $K \varphi$ denotes that agent $i$ considers $\varphi$ as possible; $D_i, E_i, C_i$ represent distributed knowledge in the group of agents $\Gamma$, everyone in $\Gamma$ knows, and common knowledge among agents in $\Gamma$, respectively.

The other basic temporal modalities are introduced as usual:

$$\neg \alpha \lor \neg \beta \land \alpha \rightarrow \beta \land \alpha \leftarrow \beta \Rightarrow (\alpha \rightarrow \beta) \land (\beta \rightarrow \alpha).$$

In order to illustrate the semantics of real-time temporal logic of knowledge, let us first review the knowledge about clock constraint[12] and timed automata[13-14].

Definition 2. (Clock constraint) A clock constraint $g$ over the set of clock variables $C$ is formed according to the following grammar:

$$g = x < c \mid x \leq c \mid x > c \mid x \geq c \mid g \land g.$$ 

where $c \in \mathbb{Z}$ and $x \in C$ is a clock variable. We use $CC(C)$ to denote the set of all the clock constraints over the set of clock variables $C$.

Timed automata are used for modelling the behaviours of time-critical systems, and the definition of a timed automaton is as in the following.

Definition 3. (Timed automata) A timed automaton can be represented as a tuple: $TA = (L, Act, C, E, I_o, Inv)$, where $L$ is a finite set of locations, $I_o \in L$ is the initial location, $Act$ is a finite set of actions, $C$ represents a finite set of clock variables, $E$ represents a transition relation: $E \subseteq L \times CC(C) \times Act \times 2^C \times L$, and $Inv$ is a location invariant function $Inv: L \rightarrow CC(C)$, it assigns to each location a clock constraint which defines the condition that must be satisfied for the timed automaton to stay in this location.

As a multi-agent system(MAS) is composed by $n$ agents $(n > 0)$, and $n$ is an integer, if each agent $i (1 \leq i \leq n)$ is modelled as a timed automaton $TA_i = (L_i, Act_i, C_i, E_i, I_o, Inv_i)$, then the multi-agent system can be modelled as the parallel composition of these $n$ timed automata. The definition of the parallel composition of multi-agent automata is as the following.

Definition 4. (Parallel composition of timed automata) The parallel composition of $m$ timed automata $TA_i (1 \leq i \leq m)$ is a global timed automaton $TA = (L, Act, C, E, I_o, Inv)$, where $L = \bigcup_{i=1}^{n} L_i$, that is a global location $l$ of timed automaton $TA_i$ is a tuple $l = (l_1, l_2, ..., l_m)$, and $I_o = \bigcup_{i=1}^{n} Act_i$, $C = \bigcup_{i=1}^{n} C_i$, $l_0 = (l_{01}, l_{02}, ..., l_{0m}) (1 \leq i \leq m)$ is the initial location of the $i$th timed automaton $TA_i$; $Inv(l_1, l_2, ..., l_m)$ is a finite function $Inv(\cdot)$, the global transition $E$ is such that:

$$(\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff (\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff (\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff (\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff (\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff (\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff (\forall i \in Act(a)), (\forall i, j, e, a, D_i(l_i), l_i, l_i) \in E \iff$$

The symbol $D (D \subseteq C)$ above represents the set of clock variables that are reset to 0 in the global transition, $Act(a)$ represents the set of indices of the timed automata whose sets of actions contain the action $a$, that is $Act(a) = \{1 \leq i \leq m | a \in Act_i\}$. And $e, l_i$ represents the location component of timed automaton $TA_i$ in the global location $l$.

The semantic model of TACTLK is real time interpreted systems, and a real time interpreted system is defined as following.

Definition 5. (The semantic model of TACTLK) The real time interpreted system corresponding to the timed automaton $TA = (L, Act, C, E, I_o, Inv)$ is a tuple $M = (Q, q_0, E, V, s, V_s)$, where:

- $Q$ is a subset of $L \times R^C$, that is $Q \subseteq L \times R^C$, and
$R^{C}$ is the set of clock valuations over the set of clock variables $C$, then each state in $Q$ is a tuple $(l,v)$ composed by a location $l$ and a clock valuation $v$. All states in $Q$ are reachable.

- $q_0 = (l_0,v_0)$ is the initial state such that $\forall x \in C, v_0(x) = 0$.
- $E$ is the transition relation: $E \subseteq (L \times R^C) \times (L \times \mathbb{R}^C)$, there exist two kinds of transitions:
  1. Time transition: for $\delta \in R$, $(l,v) \xrightarrow{\delta} (l,v+\delta)$, if and only if $v \models \text{Inv}(l)$ and $\delta \models \text{Inv}(l)$.
  2. Action transition: for $a \in A, (l,v) \xrightarrow{a} (l',v')$, if and only if $(\exists \alpha \in CC(C))(\exists D \subseteq C)$ such that $(l_a-\alpha \cdot \delta) \rightarrow (l' \in E)$ and $v[D=0] = \text{Inv}(l')$.

$V : Q \rightarrow 2^V$ is a valuation function, and we have $V((l,v)) = V_{FA}(l)$. $V_{FA}(l)$ denotes the set of propositional variables that are valid in the location $l$ of timed automaton $TA$.

Definition 6. (Semantics of TACTLK: the satisfaction relations)

Let $M = (Q,q_0,\rightarrow_1,\ldots,\rightarrow_n,V)$ be a real time interpreted system, $M,q \models \alpha$ represents that the TACTLK formula $\alpha$ is true at state $q$ in $M$, and in the following satisfaction relations, the symbol $M$ is omitted. Suppose $p, \phi$ and $\varphi$ in the following are all TACTLK formulae, the satisfaction relation “$\models$” is defined inductively in the following:

- $q \models p$ iff $p \in V(q)$;
- $q \models \neg p$ iff $p \notin V(q)$;
- $q \models \phi \lor \varphi$ iff $q \models \phi$ or $q \models \varphi$;
- $q \models \phi \land \varphi$ iff $q \models \phi$ and $q \models \varphi$;
- $q \models A(\delta U) \phi$ iff $(\forall \rho \in f_{\tau A}(q))(\exists r \in I)(\pi_\rho(r) |= \phi)$ and $(\forall \nu < r)(\pi_\nu(r) |= \phi)$;
- $q \models A(\delta R) \phi$ iff $(\forall \rho \in f_{\tau A}(q))(\forall r \in I)(\pi_\rho(r) |= \phi)$ or $(\exists r < \nu)(\pi_\rho(r) |= \phi)$;
- $q \models K \phi$ iff $(\forall q' \in Q)(q \rightarrow q' \text{ implies } q' |= \phi)$;
- $q \models E \phi$ iff $(\forall q' \in Q)(q \nrightarrow q' \text{ implies } q' |= \phi)$;
- $q \models D \phi$ iff $(\forall q' \in Q)(q \nrightarrow q' \text{ implies } q' |= \phi)$.

III. ABSTRACTION

A. The abstraction techniques

Every state in the set of states $Q$ of a real time interpreted system can be represented in the form: $(l,v)$, where $l$ is a global location, and $v$ is a clock valuation for all the clock variables. Due to the continuous nature of time, we have $v \in R^+$, and therefore the state space of real time interpreted systems is infinite. As a result, the existing model checking algorithms for finite state systems can not be applied to model checking in real time interpreted systems. The state space of real time interpreted systems needs to be divided equivalently such that the problem of model checking for real time interpreted systems can be converted into model checking for finite state systems.

Suppose $M = (Q,q_0,E,\rightarrow_1,\ldots,\rightarrow_n,V)$ is the original model of a real time interpreted system, our aim is to present an abstraction technique, and using it a corresponding abstract model $M' = (Q,q_0,E,\rightarrow_1,\ldots,\rightarrow_n,V)$ can be deduced from the original model $M$, which is an over approximation of the original model. Therefore, the model checking procedure can be carried out in the obtained abstract model while preserving the properties of the original model.

For every clock variable $x \in C$ in the timed automaton that models a multi-agent system, we use $Ix$ to represent its integer part variable and $Fx$ its fractional order variable. $Ix$ represents the integer part of clock variable $x$, that is, if $x < c_i$, then $I_x = [x]$; otherwise $I_x = c_i$ (The maximum constant value compared to clock variable $x$). Therefore, $Ix$ is an integer ranging between 0 and $c_i$. For a clock valuation $v$, order the clock variables that are smaller than or equal to the corresponding maximum constant value $c_i$ according to the values of their fractional parts $\text{frac}(x)$, and we use $Fx$ to represent the position of clock variable $x$ in this order. For a clock variable $x$ satisfying $x \leq c_i$, the fractional order variable $Fx = 0$ if and only if the fractional part of $x$ is zero, that is $\text{frac}(x) = 0$. Therefore,
Fx is an integer ranging between 0 to n (n is the number of clock variables).

Definition 7. (Discrete clock valuations) For the set of clock variables C of the timed automaton modelling a multi-agent system, the corresponding discrete clock valuation v is a function: for every clock variable x ∈ C, assigns to it a value from 0, 1, ..., n, and to Fx a value from {0, 1, ..., n}, and we use v(x) to denote the pair (v(x), v(Fx)).

As a clock region is a set of clock valuations satisfying the same clock constraint, it can be inferred that each discrete clock valuation corresponds to a unique clock region, and vice-versa. For example, the discrete clock valuation v(x,y) = ((0,1), (0,2)) for the set of clock variables C = {x, y} with c_x = 2, c_y = 1 corresponds to the clock region shown in the shadow area of Fig. 1.

![Figure 1. The clock region corresponding to the discrete clock valuation ((0,1), (0,2))](image)

One requirement of the equivalence of two clock valuations v and v' is that the ordering relation of the fractional parts of any two clock variables is invariant under the two clock valuations v and v'. For example, for any two clock variables x, y ∈ C, if we have frac(v(x)) ≤ frac(v(y)) in clock valuation v, then it must hold that frac(v'(x)) ≤ frac(v'(y)) in v'. However, if the number of clock variables in the timed automaton modelling a multi-agent system is n, the ordering relations of their fractional parts have 2^n possibilities, which corresponds to 2^n different clock regions and as a result, the state space of the real time interpreted system grows exponentially. Therefore, if we combine the clock regions in which only the ordering relations of the fractional parts of certain clock variables are different, the number of different clock regions can be greatly reduced and the state space of real time interpreted systems is also simplified in this sense.

For the discrete clock valuations in which only the ordering relations of the fractional parts of certain clock variables are different, we can combine them in this way: we replace the fractional order variables Fc with the abstract fractional order variables Fc*. These variables assume values in the abstract domain Fc* = {0, α}, where 0 represents clock variables whose fractional part is equal to zero, and α represents all other possible fractional order values. For example, for the discrete clock valuation v(x, y, z) = ((0,3), (1,1), (2,2)) for clock variables x, y, z, the corresponding abstract discrete clock valuation is v*(x, y, z) = ((0, α), (1, α), (2, α)); for discrete clock valuation v'(x, y, z) = ((1,0), (2,0), (3,0)), its abstract discrete clock valuation is v*(x, y, z) = ((1,0), (2,0), (3,0)). V* is used to represent the set of abstract discrete clock valuations over the set of clock variables C for any clock x ∈ C, we use v*(x) to denote v*(v(x), v(Fx)).

Definition 8. (Abstract discrete clock valuations) Given a set of clock variables C, one abstract discrete clock valuation over C is a function v*: for each clock variable x ∈ C, it assigns a value from {0, 1, ..., n} to Fx, and a value from F* to Fc*.

Definition 9. (Epistemic equivalent to agent i between two concrete states: ~i) For two concrete states (l, v), (l', v') in a real-time interpreted system, the epistemic equivalent to agent i between them can be represented as follows: (l, v) ~i (l', v') if and only if l(i) = l'(i) and v(i) = v'(i). That is, the location component of agent i in the global location l is the same as that in the global location l', and the two clock valuations v and v' are equivalent.

Definition 10. (Epistemic equivalent to agent i between two abstract states: ~i) Suppose (l, v), (l', v') are two abstract states, the epistemic equivalent to agent i between them is defined as following: (l, v) ~i (l', v') if and only if for each concrete state s_i in (l, v), and for each concrete state s_j in (l', v'), it always holds that s_i ~j s_j. That is, the two concrete states are epistemic equivalent to agent i.

For example, suppose there are three agents in a real-time interpreted system: agent 1, 2, 3, that is Ag = {1, 2, 3}; and there are two clock variables x, y, that is C = {x, y}, and C_x = C_y = 2. Suppose in the abstract state s = (l, ((1,0), (1,α))) there are three concrete states: s_1 = ((l_1, l_2, l_3), (1,5,1.1)), s_2 = ((l_1, l_2, l_3), (1,5,1.2)) and s_3 = ((l_1, l_2, l_3), (1,5,1.3)); and in the abstract state s' = (l', ((1,α), (1,α))) there are four concrete states: s_1' = ((l_1', l_2', l_3'), (1,8.14)), s_2' = ((l_1', l_2', l_3'), (1,8.15)) and s_3' = ((l_1', l_2', l_3'), (1,8.16)) and s_4' = ((l_1', l_2', l_3'), (1,8.17)). Then we can learn that the two abstract states are epistemic equivalent to agent 1: s' ~i s', because for each concrete state s_j (1 ≤ j ≤ 3) in s, and for each concrete state s_j' (1 ≤ j ≤ 4) in s', it is always holds that s_j ~1 s_j'. Therefore, the two abstract states s and s' are epistemic equivalent to agent 1: s ~1 s'.

Having the definition of the epistemic equivalent to agent i (1 ≤ i ≤ n) between two abstract states, the abstract states satisfying the constraints in the definition can be combined into one abstract state, that is an equivalent class. And therefore the state space of real
time interpreted systems can be further simplified.

B. Constructing the abstract model

The abstract model \( M' \) corresponding to the original model \( M \) can be deduced according to the abstraction techniques presented above, that is the abstract discrete clock valuations and the epistemic equivalent to an agent between two abstract states.

Definition 11. (The abstract model of a real time interpreted system)

The abstract model \( M' \) corresponding to the original model \( M \) of a real time interpreted system is a tuple

\[
M' = (Q', q_0, E', c_1, \ldots, c_n, V'),
\]

where

- \( Q' = L \times R_c' \) is the set of states of the abstract model,
- \( L \) represents the set of abstract global locations and \( R_c' \) represents the set of abstract discrete clock valuations over the set of clock variables \( C' \).
- \( q_0 = (l_0, v_0) \) is the abstract initial state, and to each of its concrete initial state \( q_0 = (l_0, v_0) \) it holds that \( l_0 \in L_0 \), and \( \forall x \in C, v_0(x) = 0 \).
- \( E' \subseteq (L \times R_c') \times (Act \cup R_c') \times (L \times R_c') \) is the state transition relations in the abstract model, and two kinds of transitions are available:
  1. Time transition: \((l, v') \leftarrow (l, v') + (v', d)\), if and only if \( v' = \text{inv}(v) \) and \( v' + d = \text{inv}(v) (\forall d > 0) \);
  2. Action transition: \((l, v') \leftarrow (l, v') (a \in Act)\), if and only if \((\exists c \in C(c))((\exists D \subseteq C), (l, v') \rightarrow (l, c) \) and \( v_0(v) = v(D = 0) = \text{inv}(v) \) represents that the abstract clock valuation \( v_0 \) is obtained in this way: for each clock variable \( x \) in the set \( D \subseteq C \), its integer part \( x \) are all reset to 0, and the values of all the other clock variables in \( v_0 \) are the same as in \( v' \).

\( \sqcup \subseteq Q \times Q \) is an epistemic equivalent relation, the epistemic equivalent to agent \( i \) between two abstract states \((l, v') \leftarrow (l, v') \) can be represented in the following: \((l, v') \sqcup (l, v') \), if and only if the two abstract states satisfy the constraints in definition 10.

- \( V' : Q \rightarrow 2^{\nu} \) is a valuation function, for an abstract state \((l, v)\) we have \( V((l, v)) = \bigcap_{\nu} V_{\rho_{\nu}}(l) \) (\( \ell_i \) denotes the global location of the \( i \)th concrete state of this abstract state). That is to say, the set of propositional variables that are valid in an abstract state is the intersection of the sets of the propositional variables that are valid in each of its concrete state.

Definition 12. (The TACTLK semantics on the abstract model)

\( M' = (Q', q_0, E', c_1, \ldots, c_n, V') \) is an abstract model of a real time interpreted system \( M \), \( M', q = \alpha \) denotes that the TACTLK formula \( \alpha \) holds on the abstract state \( q \) of the abstract model \( M' \). And the symbol \( M' \) is omitted in the following satisfaction relations. Suppose \( p, \phi \) and \( \psi \) in the following are all TACTLK formulae, the satisfaction relation “\( | = \)” is defined inductively as in the following:

- \( q \models p \iff p \in V'(q) \). That is, for each concrete state \( s \in q \), it holds that \( p \in V(s) \);
- \( q \models \neg p \iff p \notin V'(q) \). That is, for each concrete state \( s \in q \), it holds that \( p \notin V(s) \);
- \( q \models \phi \land \psi \iff q \models \phi \) or \( q \models \psi \);
- \( q \models \phi \lor \psi \iff q \models \phi \) and \( q \models \psi \);
- \( q \models A(\phi, \psi) \iff (\forall \rho \in f_{\rho}(q))(|(\rho(r) = \phi)) \land (\forall r' < r)(\rho(r') = \psi) \), that is, for each concrete state \( s \in q \), \( (\forall \rho \in f_{\rho}(q))(|(\rho(r) = \phi)) \land (\forall r' < r)(\rho(r') = \psi) \);
- \( q \models F_\phi \iff (\forall q' \in Q)(q \neg \neg' \psi) \), where \( q \neg \neg' \psi \) amounts to \( q \neg \neg' \psi \), that is to say, there is at least one agent \( i \) in the set of agents \( \Gamma \) such that \( q \neg \neg p \) holds;
- \( q \models D_\phi \iff (\exists q' \in Q)(q \neg \neg' \psi) \), where \( q \neg \neg' \psi \) amounts to \( q \neg \neg' \psi \), in other words, for each agent \( i \) in the set of agents \( \Gamma \), it must hold that \( q \neg \neg p \) holds;
- \( q \models C_\phi \iff (\forall q' \in Q)(q \neg \neg' \psi) \), where \( q \neg \neg' \psi \) is the transitive closure of \( q \neg \neg' \psi \).

C. Property Preservation Theorem

The aim of abstraction is to simplify the original model of the system while preserving its properties, and in the following we prove that the satisfaction relations of TACTLK formulae are preserved in the abstract model. That is, if a TACTLK formula \( \phi \) is satisfied by an abstract model \( M' \), it can be inferred that the formula \( \phi \) is also satisfied in the original model \( M \).

Theorem 1. Suppose \( M = (Q, q_0, E, c_1, \ldots, c_n, V) \) is a real time interpreted system, \( M' = (Q', q_0, E', c_1, \ldots, c_n, V') \) is the corresponding abstract model obtained according to the abstraction techniques presented above, and \( \phi \) is a TACTLK formula. If \( M'$, s' \models \phi \), then \( M, s \models \phi \). That is, the abstract model \( M' \) over approximates the original model \( M \).

Proof: The theorem is proved by induction on the structure of the formula \( \phi \). Obviously, the theorem follows directly for the atomic propositions, and the operators \( \land \) and \( \lor \). We mainly consider the following forms:

(1) \( \phi = \neg a \) (\( a \) is an atomic proposition)

By definition 12, if \( M' \models S' \models \neg a \), then \( a \notin V'(s') \). That
is, \(\forall s \in s', a \in V(s)\), therefore, \(\forall s \in s', s = \neg a\). Then, we have \(M, s = \neg a\).

(2). \(\phi = A (\phi \cup \psi)\)

By definition 12, if \(M', s' = A (\phi \cup \psi)\), then \((\forall r' \in f_{s'}(s))((\exists r \in I (\pi_r(r)) = \psi) \land ((\forall r' < r)(\pi_r(r') = \phi))\) holds. That is, for each concrete state \(s \in s'\), we know that on each path \(\rho\) starting from \(s\), there exists a time point \(r\) in time interval \(I\), such that on path \(\rho\) the state \(\pi_r(r)\) corresponding to time \(r\) satisfies the formula \(\psi\) and all the previous states on path \(\rho\) satisfy the formula \(\phi\). That is, \(\forall s \in s', (\forall r \in f_{s'}(s))((\exists r \in I (\pi_r(r)) = \psi) \land ((\forall r' < r)(\pi_r(r') = \phi))\) holds. Therefore, it can be inferred that \(M, s = A (\phi \cup \psi)\).

(3). \(\phi = A (\phi R \psi)\)

By definition 12, if \(M', s' = A (\phi R \psi)\), then \((\forall r' \in f_{s'}(s))((\forall r \in I (\pi_r(r)) = \psi) \land ((\forall r' < r)(\pi_r(r') = \phi))\) holds. It can be inferred that 1) \((\forall r \in I (\pi_r(r)) = \psi)\) holds, or 2) \((\forall r' < r)(\pi_r(r') = \phi) \land ((\forall 0 \leq r' < r)(\pi_r(r') = \phi))\) holds. In the following, we analyze the two cases respectively:

- Suppose case 1) holds. We can learn that for each concrete state \(s \in s'\), on each path \(\rho\) starting from \(s\) and for each time point \(r\) in time interval \(I\), the state \(\pi_r(r)\) corresponding to time \(r\) satisfies the formula \(\psi\), that is, \(\forall s \in s', (\forall r \in f_{s'}(s))((\exists r \in I (\pi_r(r)) = \psi)\) holds. Then it can be known that \(M, s = A (\phi \cup \psi)\).
- Suppose case 2) holds. We can learn that for each concrete state \(s \in s'\), on each path \(\rho\) starting from \(s\) and for each time point \(r\) in time interval \(I\), there exists a time point \(r'\) which is smaller than \(r\), such that the state \(\pi_r(r')\) corresponding to time \(r'\) on path \(\rho\) satisfies the formulae \(\phi\) and \(\psi\), and all the previous states on path \(\rho\) satisfy the formula \(\psi\). That is, \(\forall s \in s', (\forall r \in f_{s'}(s))((\exists r' \in I (\pi_r(r')) = \psi) \land ((\forall 0 \leq i < r')(\pi_r(i)) = \psi))\). Then it can be known that \(M, s = A (\phi \cup R \psi)\).

(4). \(\phi = K \alpha\)

By definition 12, if \(M', s' = K \alpha\), then \((\forall s' \in Q) (s' \sim s\) implies \(s' = \alpha\)). Using the definition of epistemic equivalent to agent \(i\) between two abstract states, it can be inferred that \((\forall s' \in Q) (s' \sim s\) implies \(s' = \alpha\)). As \(s_i = \alpha\) can be inferred from \(s' = \alpha\), we can obtain that \((\forall s' \in Q) (s' \sim s\) implies \(s_i = \alpha\)). That is, \(M, s = K \alpha\).

(5). \(\phi = E_i \alpha\)

As we know that \(E_i \alpha = \bigwedge_{s' \sim i} K \alpha\), therefore the result follows from the case (4) above for a specific agent \(i \in \Gamma\) and the inductive hypothesis for operator \(\lor\).

(6). \(\phi = D_\tau \alpha\)

As we know that \(D_\tau \alpha = \bigwedge_{s' \tau} K \alpha\), therefore the result follows from the case (4) above for a specific agent \(i \in \Gamma\) and the inductive hypothesis for operator \(\land\).

(7). \(\phi = C_\tau \alpha\)

The result follows from the case (5) above and the fact that \(C_\tau \alpha\) is the transitive closure of \(E_\tau \alpha\).

VI. A CASE STUDY

In the following, the effectiveness of our abstraction techniques is illustrated by the simplification of the state space of a variant of the standard Railroad crossing system[15].

A. Introduction of the Railroad Crossing System

The standard railroad crossing system has been used to compare different formal methods of real time systems, and the railroad crossing system presented here is a variant of it. The system is the parallel composition of three agents: \(Train\), \(Gate\), and \(Controller\). The timed automata corresponding to them are given in Fig. 2, Fig. 3, and Fig. 4, respectively. The three agents synchronize through the set of actions: approach, lower, exit, and raise. We use \(t (t \geq 0)\) to represent that \(t\) seconds have elapsed since the initial time. When \(0 < t < 100\), agent \(Train\) reaches a place not far from the crossing, and at this moment, \(Train\) sends an \(approach\) signal to \(Controller\); at a time point between 200 and 300 seconds after its sending the \(approach\) signal, \(Train\) sends a \(in\) signal to its environment and enters the crossing; when \(Train\) is to leave the crossing, it sends an \(out\) signal to its environment and at the 400th second exactly after its sending the \(approach\) signal \(Train\) leaves the crossing and sends an \(exit\) signal to \(Controller\) so as to synchronize with it. \(Controller\) sends a \(lower\) signal to \(Gate\) exactly 100 seconds after its receiving the \(approach\) signal, and sends a \(raise\) signal to \(Gate\) within 100 seconds after its receiving the \(exit\) signal. \(Gate\) responds to the \(lower\) signal from \(Controller\) by moving down within 100 seconds, and to the \(raise\) signal by moving up between 100 and 200 seconds.

In our railroad crossing system, a concrete state should be in the form: \(((i, j, k), (v, v', v'')) (0 \leq i, j, k \leq 3)\). Where \((i, j, k)\) is the global location of this state, the three components \(i, j, k\) represent that agent \(Train\), \(Gate\) and \(Controller\) are in locations \(i, g, c\) respectively; \((v, v', v'')\) is a clock valuation over the set of clock variables \(C = \{x, y, z\}\) in the system, which represents that in this state the values of clock variables \(x, y, z\) are \(v, v', v''\) respectively. As the ranges of the values of the
Figure 2. Timed automaton Train

Figure 3. Timed automaton Gate

Figure 4. Timed automaton Controller
clock variables $x, y, z$ in this system are $0 \leq v_x \leq 400$, $0 \leq v_y \leq 200$, $0 \leq v_z \leq 100$, respectively, and $v_x, v_y, v_z$ are all real numbers, there are infinite states in this system, that is, the state space of our system is infinite.

B. Constructing the Abstract Model of the System

The abstract techniques presented above are applied to construct the abstract model of our railroad crossing system such that the infinite state space of our system can be simplified into a finite form.

For a concrete state $((i, j, k), (v_x, v_y, v_z))$ of our system, its clock valuation is represented using the corresponding abstract discrete clock valuation. In this way, the concrete states whose clock valuations belong to the same clock region and global locations are the same are combined into one abstract state, and therefore we can obtain the finite form of the infinite state space of our system. In the obtained finite form of our railroad crossing system, using the definition of the epistemic equivalent to agent Train between two abstract states, any two abstract states satisfying the constraints of this definition can be further combined into one abstract state. At last, the corresponding abstract model of our railroad crossing system can be obtained. In figure 5, all the reachable states of the abstract model obtained using our abstraction techniques are given.

From the abstract model, it can be seen that using our abstraction techniques the infinite state space of the railroad crossing system is simplified into a finite form which only has eight abstract states.

Suppose one time unit denotes 100 seconds, as the maximum clock constants compared to the clock variables $x$, $y$, and $z$ are 400, 200, and 100, respectively, therefore each state and its location invariant of the state space $Q$ of the abstract model $M^a$ is as the following:

$S_1: ((0,0,0), ((0,0),(0,0),(0,0)))$, $Inv(0,0,0) = true$;
$S_2: ((1,0,1), ((0,0),(0,0),(0,0)))$, $Inv(1,0,1) = x \leq 400 \land y \leq 100$;
$S_3: ((1,1,2), ((0,0),(0,0),(0,0)))$, $Inv(1,1,2) = x \leq 400 \land y \leq 100$
$S_4: ((1,2,2), ((1,0),(0,0),(1,1)))$, $Inv(1,2,2) = x \leq 400$
$S_5: ((2,2,2), ((2,0),(1,0),(2,1)))$, $Inv(2,2,2) = x \leq 400$
$S_6: ((3,2,2), ((3,0),(2,0),(3,1)))$, $Inv(3,2,2) = x \leq 400$
$S_7: ((0,2,3), ((4,0),(2,0),(4,1)))$, $Inv(0,2,3) = z \leq 100$
$S_8: ((0,3,0), ((4,0),(0,0),(0,0)))$, $Inv(0,3,0) = y \leq 200$.

That is, in the obtained abstract model $M^a = (Q, q_0, E, \prec_M, \sim_M, \alpha_M, V)$ of our railroad crossing system, we have:
The abstract state space $Q = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8\}$;
The abstract initial state $q_0 = S_1$;
The set of state transitions $E = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}$;
Each abstract state and itself is epistemic equivalent to agent Train that is, for each state $S_i \in Q (1 \leq i \leq 8)$, $S_i \sim_M S_i$ holds. Similarly, it is also hold to agent Gate and Controller.

C. Model Checking Railroad Crossing System

The safety property to be verified is: in our railroad crossing system, agent Train considers such a behaviour holds, in which after Train sends an approach signal to Controller, Gate will move down at a time point between 100 and 200 seconds. As Train enters the crossing at a time point between 200 and 300 seconds after its sending the approach signal, this property guarantees that the Gate is down when the Train enters the crossing. And therefore, the pedestrians and the vehicles near the crossing can be fully protected.

The safety property can be expressed by the TACTLK
formula $\phi = K_{\text{Train}}(\text{approach} \land AF_{(100,200),\text{down}})$, that is, in the state space of the abstract model such states are not allowed to occur: in these states the location components of $\text{Train}$ and $\text{Gate}$ are 2, 0 or 2, 1. From the abstract model $M'$ obtained above, it can be easily seen that there is no such state in the state space $Q'$, that is, our abstract model satisfies the safety property: $M' \models \phi$.

As in section 3.3 we have proved that the satisfaction relations of TACTLK formulae are preserved on the abstract model, it can be concluded that our railroad crossing system satisfies the safety property: $M \models \phi$.

Using our abstraction techniques, the infinite state space of the original model has been reduced to eight states of the abstract model, which illustrates the effectiveness of our abstraction techniques.

V. CONCLUSIONS AND FUTURE WORK

In order to alleviate the state space explosion problem in model checking real-time temporal logic of knowledge, we present the abstraction techniques: the infinite state space of a real time interpreted system can be simplified into a finite form using abstract discrete clock valuations; using the definition of the epistemic equivalent to an agent between two abstract states, any two abstract states satisfying the constraints of this definition can be combined into one abstract state, that is an equivalent class. Therefore, the state space of the real time interpreted system can be further simplified. Our abstraction techniques implemented the conservative computation of the satisfaction relations of TACTLK formulae, that is, if a TACTLK formula is satisfied by an abstract model, then it is also satisfied by the corresponding original model. The effectiveness of our abstraction techniques is illustrated by the simplification of the state space of a variant of the standard railroad crossing system.

There are many interesting avenues for future research. As the abstract model obtained using our abstraction techniques is only an over approximation of the original model, if a property is not satisfied by the abstract model, it can not be inferred that this property is not satisfied by the original model either. Introducing the third value into our abstraction techniques that represents uncertainty besides true and false is a valuable direction for future research.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant Nos.60773137, 61003288), the Ph.D. Programs Foundation of Ministry of Education of China(Grant No. 20093227110005), the Natural Science Foundation of Jiangsu Province(Grant No. BK2010192)

REFERENCES

Computer Science in China, Volume 5, Number 1, 14-25, June 2010.


**CongHua Zhou** Jiangsu Province, China. Birthdate: June, 1978. He received his Ph.D degree in mathematics from Nanjing University in 2006. His research interests include model checking, information security, modal logics, trusted embedded software, and multi-agent systems.

He is an associate professor of school of computer science and telecommunication engineering, Jiangsu University.

**Bo Sun** Shandong Province, China. Birthdate: July, 1987. He obtained his bachelor’s degree in computer science from Liaocheng University in 2009, and he is now pursuing his master’s degree in school of computer science and telecommunication engineering, Jiangsu University. His research interests include model checking, temporal logic of knowledge.

© 2012 ACADEMY PUBLISHER