ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.7, July 2010, pp.1491−1502
doi: 10.3724/SP.J.1001.2010.00585

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

改进的以 SMT 为基础的实时系统限界模型检测*

徐 亮[1,2+]

[1](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[2](中国科学院 软件研究所,北京 100190)

Improved SMT-Based Bounded Model Checking for Real-Time Systems

XU Liang[1,2+]

[1](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: xul1981@126.com

Xu L. Improved SMT-based bounded model checking for real-time systems. *Journal of Software*, 2010,21(7): 1491−1502. http://www.jos.org.cn/1000-9825/585.htm

Abstract: SAT-Based bounded model checking (BMC) has high complexity in dealing with real-time systems. Satisfiability modulo theories (SMT) solvers can generalize SAT solving by adding the ability to handle arithmetic and other decidable theories. This paper uses SMT in BMC for real-time systems instead of SAT. The clocks can be represented as integer or real variables directly and clock constraints can be represented as linear arithmetic expressions. These make the checking procedure more efficient. TCTL (timed computation tree logic) is used to specify the properties of real-time systems and improvement of the encodings has been done.

Key words: bounded model checking; satisfiability modulo theories; real-time system; timed automata; timed Kripke structure; TCTL (timed computation tree logic)

摘 要: 基于 SAT 的限界模型检测在处理实时系统时具有很高的复杂度.SMT 求解器在计算可满足性的同时,还能处理算术和其他可判定性理论.在对实时系统进行检测时,用 SMT 求解器代替 SAT 求解器,系统里的时钟就可以用整型或实型变量表示,时钟约束则可以直接表示成线性算术表达式,从而使整个检测过程更加高效.带时间参数的计算树逻辑(timed computation tree logic,简称 TCTL)被用来描述实时系统里的性质.同时,还对检测方法作了相应的改进.

关键词: 限界模型检测;可满足性模块理论;实时系统;时间自动机;时间 Kripke 结构;带时间参数的计算树逻辑

中图法分类号: TP301 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60721061, 60833001 (国家自然科学基金); the CAS Innovation Program of China (中国科学院知识创新工程领域前沿项目); the Knowledge Innovation Key Directional Program of the Chinese Academy of Sciences under Grant No.KGCX2-YW-125 (中国科学院知识创新工程重要方向项目)

Received 2008-12-12; Accepted 2010-01-25

1 Introduction

In symbolic model checking, *Binary Decision Diagrams* (BDD's)[1] are used to represent system states. The technique is used in many model checking tools, such as NuSMV[2] and TSMV[3]. But the size of BDD may grow significantly as the number of variables increase. Checking a system with a large number of variables remains a difficult problem for BDD-based model checking tools. On the other hand, *Bounded Model Checking* (BMC) based on *Boolean Satisfiability* (SAT) has been introduced as a complementary technique to BDD's model checking[4–6] for combating the state explosion problem. The basic idea of BMC presented in Refs.[4–6] is to restrict the general model checking problem to a bounded one. Instead of finding out whether the system M violates the property ψ, we only need to know whether the system M has some counterexamples of length k to ψ. The problem is encoded into a propositional formula, and then a SAT solver is used to check the formula, in order to see whether it is satisfiable. The method's efficiency is based upon the fact that if there is a counterexample, then it may be found only in a small portion of its state space[4,5].

In recent years, people's interest in automated verification has shifted to real-time systems. The verification of real-time systems becomes a very important and challenging problem. References [7,8] transform the *Timed Automaton* (TA) to a *Region Graph* (RG)[9], which is based on *dense-time* approach[10–12], then encode a Timed CTL (TCTL)[9] formula to a propositional formula and use SAT solvers to check it. TSMV[3], improved from NuSMV for verifying timed systems, uses *Timed Kripke Structure* (TKS) as its model, which is based on *discrete-time* approach[13], and solves problems with BDD-based method. Either SAT-based or BDD-based method for real-time systems, needs to encode clocks and clock constraints into boolean formulae. After this encoding, the clocks' characteristics disappear which means the whole checking process has no time information to use. In order to overcome this disadvantage, we use *Satisfiability Modulo Theories* (SMT) instead of SAT to do the check.

The SMT problem is a generalization of the SAT problem where Boolean variables are replaced by predicates from various background theories, such as linear real and integer arithmetic. So, we can use real or integer variables to represent clocks and linear arithmetic expressions to represent clock constraints instead of Boolean formulae, which preserve the time characteristics in the checking process. There are some related works for SMT-based BMC, such as CBMC[14] which is dealing with programming languages and SAL[15] which can check finite state systems based on SAT solvers and check infinite state systems based on SMT solvers. By doing so, the encoding of clock variables and clock constraints has less effect on BMC's efficiency. A preliminary version of SMT-based BMC for real-time systems is mentioned in Ref.[16] and the improvement to simplify the encodings in order to improve its efficiency is given in this paper.

The rest of this paper is organized as follows. Some preliminaries are introduced in the next section. The SMT-based BMC approach and the improvement are given in Section 3. The experimental results are summarized in Section 4. Concluding remarks are given in Section 5.

2 Preliminaries

Real-Time systems do not only contain discrete variables but also have dense-time clocks which have a real domain and continuously increase at a uniform rate. Clocks are usually set to zero at the beginning and can be reset at any time. Real-time systems can often be modeled as a TA, which is a popular approach and several model checkers exist (e.g. Refs.[17,18]), or simply modeled as a TKS, which combines simplicity and efficiency (e.g. Ref.[3]).

2.1 Timed automata

A TA is a finite-state machine equipped with a set of *clocks*. Hereafter, the set $\mathcal{AP}=\{p_1,p_2,\ldots\}$ denotes *atomic propositions*, $\mathbb{N}$ denotes the set of natural numbers, $\mathbb{N}^+$ denotes the set of $\{1,2,\ldots\}$, $\mathbb{Z}$ denotes the set of integer numbers, $\mathbb{R}$ denotes the set of real numbers and $\mathbb{R}^+$ denotes the set of non-negative real numbers.

Let $\mathcal{X}$ be a finite set of variables called *clocks*. A *clock valuation* is a function $v:\mathcal{X}\to\mathbb{R}^+$, which assigns a non-negative real number $v(x)$ to each clock $x\in\mathcal{X}$. For a subset $\mathcal{Y}$ of $\mathcal{X}$ by $v[\mathcal{Y}:=0]$ we mean the valuation v' such that $\forall x\in\mathcal{X}$, $v'(x)=0$ and $\forall x\in\mathcal{X}\setminus\mathcal{Y}$, $v'(x)=v(x)$. For $\delta\in\mathbb{R}^+$, $v+\delta$ denotes the valuation v'' such that $\forall x\in\mathcal{X}$, $v''(x)=v(x)+\delta$. The set $\Psi_{\mathcal{X}}$ of clock constraints over the set of clocks $\mathcal{X}$ is defined as follows:

$$\psi::=x\prec c|x-x'\prec c|\psi\wedge\psi'|\neg\psi,$$

where $x,x'\in\mathcal{X}$, $\prec\in\{<,\leqslant,=,\geqslant,>\}$ and $c\in\mathbb{N}$.

A clock valuation $v\in\mathbb{V}_{\mathcal{X}}$, in which $\mathbb{V}_{\mathcal{X}}$ denotes the set of all valuations, satisfies the clock constraint $\psi\in\Psi_{\mathcal{X}}$ denotes:

$$\begin{aligned} v\vDash x\prec c \quad &\text{iff} \quad v(x)\prec c,\\ v\vDash x-x'\prec c \quad &\text{iff} \quad v(x)-v'(x)\prec c,\\ v\vDash\psi\wedge\psi' \quad &\text{iff} \quad v\vDash\psi\wedge v\vDash\psi',\\ v\vDash\psi \quad &\text{iff} \quad v\nvDash\psi. \end{aligned}$$

We denote by $[[\psi]]$ the set of valuations that satisfy ψ, that is:

$$[[\psi]]=\{x\in\mathbb{V}_{\mathcal{X}}|v\vDash\psi\}.$$

Definition 1. A TA is a tuple $\langle\mathcal{S},\mathcal{X},\Sigma,s_0,\varepsilon,I\rangle$ where:

- $\mathcal{S}$ is a finite set of locations.
- $\mathcal{X}$ is a finite set of clocks.
- Σ is a finite set of labels.
- $s_0\in\mathcal{S}$ is an initial location.
- ε is a finite set of transition relations, $\varepsilon\subseteq\mathcal{S}\times\Sigma\times\Psi_{\mathcal{X}}\times2^{\mathcal{X}}\times\mathcal{S}$.
- $I:\mathcal{S}\to\Psi_{\mathcal{X}}$ is a state invariant function.

Each element $e\in\varepsilon$ is denoted by $e:=s\xrightarrow{l,\psi,\mathcal{Y}}s'$. This represents a transition from location s to location s' on the input label $l\in\Sigma$. The set $\mathcal{Y}\subseteq\mathcal{X}$ gives the clocks to be reset with this transition. $\psi\in\Psi_{\mathcal{X}}$ is the enabling condition for e. Figure 1 is a simple example of TA. Let $c_{\max}$ be the largest constant appearing in $\Psi_{\mathcal{X}}$. For $x\in\mathcal{X}$, $frac(v(x))$ denotes the fractional part of $v(x)$, and $\lfloor v(x)\rfloor$ denotes its integral part.

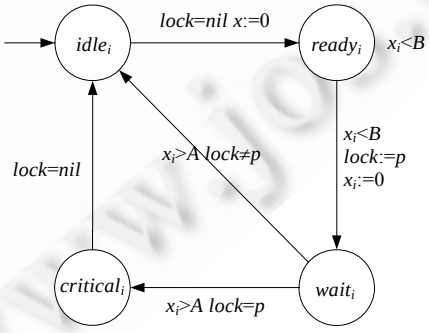

Fig.1 A simple example of TA

Definition 2. For two clock valuations v and v', $v\simeq_{\Psi_{\mathcal{X}}}v'$ iff for all $x,y\in\mathcal{X}$ follows the conditions below:

$$v(x) > c_{\max} \to v'(x) > c_{\max},$$
$$v(x) \leqslant c_{\max} \Rightarrow ((\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor) \wedge (frac(v(x)) = 0 \to frac(v'(x)) = 0)),$$
$$\forall (x - y \prec c) \in \Psi_{\mathcal{X}}.((c \leqslant c_{\max}) \Rightarrow (v \models x - y \prec c \to v' \models x - y \prec c)).$$

We use $[v]$ to denote the equivalence class of the relation $\simeq_{\Psi_{\mathcal{X}}}$ to which v belongs. Such a class is called a *zone*. The set of all the zones is denoted by $\mathcal{Z}(n)$. A zone $[v]$ is final iff $[v]>c_{\max}$ for all $v\in[v]$ and $x\in\mathcal{X}$. A zone $[v]$ satisfies the clock constraint $\psi\in\Psi_{\mathcal{X}}$, if $[v]\models\psi$ iff $\forall v'\in[v], v'\models\psi$,.

Definition 3. The RG of a TA is a finite structure $\langle \mathcal{Z}, q^0, \to, \mathcal{L}\rangle$:

- $\mathcal{Z}=\{(s,[v])|(s,[v])\in\mathcal{S}\times\mathcal{Z}(n)\}$, is the set of states.
- $q^0\in\mathcal{Z}$ is the initial state.
- $\to$ is defined as follows:
  - $(s,[v])\xrightarrow{l}(s',[v'])$ iff $\forall e: s\xrightarrow{l,\psi,\mathcal{Z}} s'\in\varepsilon$ such that $s=source(e)$, $s'=target(e)$, $[v']=[v[\mathcal{Z}:=0]]$ and $v'\models I(s')$.
  - $(s,[v])\xrightarrow{\delta}(s,[v'])$ iff $[v']=I(s)$ and $[v']=[v]+\delta$ or $[v']=[v]$ if $[v]$ is final.
- $\mathcal{L}:\mathcal{Z}\to 2^{\mathcal{AP}}$ is a labeling function that maps each state of $\mathcal{Z}$ to a set of atomic propositions true in that state.

A state q is *deadlock* if there is no delay $\delta\in\mathbb{R}^+$ and an action $l\in\Sigma$ such that $q\xrightarrow{\delta}q'\xrightarrow{l}q''$, for some $q',q''\in\mathcal{Z}$. For simplicity of presentation, we consider only progressive TA[7].

2.2 Timed Kripke structure

TKS is an extension of Kripke Structure (KS) where each transition is labeled by a nonnegative integer.

Definition 4. A TKS is a tuple $\langle \mathcal{S}, s_0, \mathcal{R}, \mathcal{L}\rangle$ where

- $\mathcal{S}$ is a finite set of states.
- $s_0\in\mathcal{S}$ is an initial state.
- $\mathcal{R}\subseteq\mathcal{S}\times\mathbb{N}\times\mathcal{S}$ is a finite set of transitions labeled by a natural number, called the duration of the transition.
- $\mathcal{L}:\mathcal{S}\to 2^{\mathcal{AP}}$ is a labeling function that maps each state of $\mathcal{S}$ to a set of atomic propositions true in that state.

A path π in TKS is a infinite sequence $s_0\xrightarrow{d_1}s_1\xrightarrow{d_2}s_2\xrightarrow{d_3}\ldots$, where $s_0,s_1,s_2,\ldots\in\mathcal{S}$. For each path, and for $n\in\mathbb{N}$, let $\pi(n)$ denotes the nth state s_n, and π^n denotes the nth suffix $s_n\xrightarrow{d_{n+1}}s_{n+1}\xrightarrow{d_{n+2}}\ldots$. Finally, for $n\leq m\in\mathbb{N}$, let $\pi[n\ldots m]$ denotes the finite sequence $s_n\xrightarrow{d_{n+1}}s_{n+1}\xrightarrow{d_{n+2}}\ldots\xrightarrow{d_m}s_m$ with $m-n$ transitions and $m-n+1$ states. The duration $D_\pi[n\ldots m]$ of such a finite sequence is $d_{n+1}+d_{n+2}+\ldots+d_m$ (D_π when $m=n$). Fig.2 is a simple example of TKS.

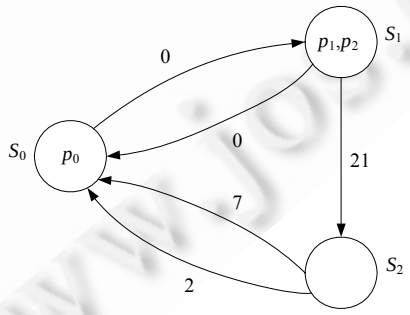

Fig.2 A simple example of TKS

2.3 TECTL

Computation tree logic (CTL) is a propositional branching-time temporal logic introduced by Ref.[19] as a

specification language for finite state systems. TECTL[9] is ECTL with timing constraints. TECTL formulae are built according to the following syntax:

$$\alpha,\beta := p \mid \neg\alpha \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid EF_{\varphi}\alpha \mid E(\alpha U_{\varphi}\beta),$$

where $p \in \mathcal{AP}$ and φ is an interval in $\mathbb{R}^{+}$ with integer bounds of the form $[n,m]$, $[n,m]$, (n,m), (n,m), $[n,\infty]$ and (n,∞), for $n,m \in \mathbb{N}$, or φ called the timing constraint, is a predicate on durations. Typically, for all constraints of the form "$\leqslant n$", "$=n$", "$\geqslant n$", n is in the relevant domain (here $\mathbb{N}$). Let I_{φ} be the predicate that the timing constraint φ is satisfied and I_{φ}^{i} be the predicate that state i satisfies the timing constraint φ where $i \in \mathbb{N}$.

Here, we give a translation of a TECTL formula ψ to an ECTL formula ψ' as follows:

$p \in \mathcal{AP}$ is translated to p

$\alpha \wedge \beta$ is translated to $\alpha' \wedge \beta'$

$\alpha \vee \beta$ is translated to $\alpha' \vee \beta'$

$EF_{\varphi}\alpha$ is translated to $EF(I_{\varphi} \wedge \alpha')$

$EG_{\varphi}\alpha$ is translated to $EG(I_{\varphi} \rightarrow \alpha')$

$E(\alpha U_{\varphi}\beta)$ is translated to $E(\alpha' U(I_{\varphi} \wedge \beta'))$

Definition 5 (**semantics**). Let $s_i \in \mathcal{S}$, $p \in \mathcal{AP}$. The semantics of TECTL formulae are defined as follows:

$s_i \models p$ iff p is satisfied in state s_i

$s_i \models \neg\alpha$ iff $s \not\models \alpha$

$s_i \models \alpha \wedge \beta$ iff $(s_i \models \alpha) \wedge (s_i \models \beta)$

$s_i \models \alpha \vee \beta$ iff $(s_i \models \alpha) \vee (s_i \models \beta)$

$s_i \models EF_{\varphi}\alpha$ iff $\exists\pi.(s_0 = s_i \wedge \exists k \geqslant 0(s_k \models I_{\varphi}^{k} \wedge \alpha))$

$s_i \models EG_{\varphi}\alpha$ iff $\exists\pi.(s_0 = s_i \wedge \forall k \geqslant 0(s_k \models I_{\varphi}^{k} \rightarrow \alpha))$

$s_i \models E(\alpha U_{\varphi}\beta)$ iff $\exists\pi.(s_0 = s_i \wedge \exists k \geqslant 0((s_k \models I_{\varphi}^{k} \wedge \beta) \wedge 0 \leqslant \forall j < k(s_j \models \alpha)))$

As an example, the TECTL formula $EF_{\varphi}\alpha$ is true in state s_i means that there is a path π with its first state equal to s_i and can reach a state which satisfies α and the timing constraint φ.

2.4 Satisfiability modulo theories

Recent SMT solvers[20−22] closely integrate theory-specific solvers with a *Davis-Putnam-Logemann-Loveland* (DPLL) approach to SAT. These types of SMT solvers are often referred to as DPLL(T)[22]. In this type of architecture, the DPLL-based SAT solver passes conjunctions of predicates belonging to theory T, such as linear real and integer arithmetic, uninterpreted functions, and the theories of various data structures[22−24], to a specialized solver. The specialized solver is then responsible for deciding feasibility of those predicates.

A *Satisfiability Modulo Theories Library* (SMT-LIB) benchmark declaration is as follows:

benchmark *b* **begin**
  logic = L
  assumption = φ_1
  ⋮
  assumption = φ_n
  formula = φ
  status = σ
end

where the value of **logic** coincides with the name of a logic L for some logic in SMT-LIB. The formulae in the **assumption** and **formula** are in the language of L and they together constitute the benchmark. The attribute **status**

declaration records whether the benchmark's formula is known to be satisfiable in the associated background theory.

For example, in checking real-time systems, the tools need to support linear real or integer arithmetic. The SMT-LIB theory of *Ints* or *Reals* or *Reals_Ints* is needed as its background theory. By using SAT solvers, multiple Boolean variables are used as a bit representation for integers and the necessary integer theories are specified as Boolean operations on those individual bit variables. This can result in extremely large SAT instances. By using SMT solvers, with ***logic*=*QF_LRA*** or ***QF_LIA***, real or integer variables, linear arithmetic and Boolean combinations of inequations between linear polynomials over real or integer variables can be used in **assumption** and **formula**. This preserves the time information in the system and with simplified instances*.

Yices[20], the winner of the 2006 SMT competition, includes an incremental simplex algorithm for the theory of linear arithmetic that is tightly integrated within the DPLL framework. Yices' great ability to work with the theory of linear arithmetic made it particularly well suited for real-time systems model checking. For these reasons, we use Yices as the SMT solver in this paper.

3 Improved SMT-Based Bounded Model Checking

3.1 Bounded semantics of TECTL

In order to define the bounded semantics of TECTL and the encodings of BMC to a SMT problem, we have to give some notions. Let $M=\langle \mathcal{S},T,s_0,\mathcal{L}\rangle$ be a model with different semantics in different models. If TA is used as the model, we have to transform the TA to a corresponding RG. Then the elements in M stand for the corresponding ones in Definition 3 and $dead(w_k)$ means the state w_k is a deadlock. (If the model is represented by TKS, and the elements in M stand for the corresponding ones in Definition 4, then $dead(w_k)$ is set *true* if $D_\pi[0,k]>n$, $k\in\mathbb{N}^+$, and the timing constraint is in the form of "$\leqslant n$" or "$=n$", and else $dead(w_k)$ is set *false*.) A k-path of M is a finite sequence $\pi=w_0,\ldots,w_k$ of states such that $(w_i,w_{i+1})\in T$ for $i=0,\ldots,k-1$. The k-model for M is a structure $M_k=\langle \mathcal{S},path_k,s_0,\mathcal{L}\rangle$, where $path_k$ is the set of all different k-paths of M. We use $loop(\pi)$ to denote $\{l|l\leqslant k\wedge(w_k,w_l)\in T\}$.

Definition 6 (**bounded semantics**). Let M_k be a k-model of M, $s_i\in\mathcal{S}$, $p\in\mathcal{AP}$, α, β TECTL formulae in Negation Normal Form (NNF), $M_k,s\models_k\alpha$ denotes that α is *true* in state s in M_k. $\models_k$ is defined as follows:

$M_k,s\models_k p$ iff $p\in L(s)$

$M_k,s\models_k \neg p$ iff $p\notin L(s)$

$M_k,s\models_k \alpha\wedge\beta$ iff $(M_k,s\models_k\alpha)\wedge(M_k,s\models_k\beta)$

$M_k,s\models_k \alpha\vee\beta$ iff $(M_k,s\models_k\alpha)\vee(M_k,s\models_k\beta)$

$M_k,s\models_k EF_\varphi\alpha$ iff $\exists\pi.(w_0=s\wedge 0\leqslant\exists i\leqslant k(M_k,w_i\models_k I_\varphi^i\wedge\alpha))$

$M_k,s\models_k EG_\varphi\alpha$ iff $\exists\pi.(w_0=s\wedge(loop(\pi)\vee dead(w_k))\wedge 0\leqslant\forall i\leqslant k'(M_k,w_i\models_k I_\varphi^i\rightarrow\alpha))$

$M_k,s\models_k E(\alpha U_k\beta)$ iff $\exists\pi.(w_0=s\wedge 0\leqslant\exists i\leqslant k((M_k,w_i\models_k I_\varphi^i\wedge\beta)\wedge 0\leqslant\forall i\leqslant j(M_k,w_j\models_k\alpha)))$

In TA model, $k'=k$. In TKS model, $k'=k$ if the timing constraint is in the form of "$\leqslant n$" or "$=n$", else $k'=2k-l'$ where $l'=\min(loop(\pi))$. A TECTL formula ψ is true in k-model M_k, denoted $M_k\models_k\psi$, iff ψ is true in the initial state of the model M_k. Similar to Ref.[7], we have the following theorem.

Theorem 1. Let $M_k=\langle \mathcal{S},path_k,s_0,\mathcal{L}\rangle$ be a k-model, and ψ a TECTL formula. $M\models\psi$ iff there is $k\in\{0,\ldots,|M|\}$ such that $M_k\models_k\psi$.

Proof (*sketch*). Follows from the following facts:

- TECTL formula can be translated to ECTL formula which was given in Section 2.3.
- Since we have assumed that we deal with progressive TA and the TKS is an extension of TK, the above

* More details about the currently supported background theories and logics in SMT_LIB, see http://goedel.cs.uiowa.edu/smtlib/.

translation can be translated to the model checking problem for ECTL.

– By Theorem 4.1 of Ref.[6] we can get the conclusion.

With this theorem, we are able to reduce a model checking problem $M\models\psi$ to a bounded model checking problem $M_k\models_k\psi$. We will discuss in the next subsection how to decide whether $M_k\models_k\psi$ or not.

3.2 SMT-Based bounded model checking for TECTL

Given a TECTL formula ψ, a model M and a bound k, SMT-based BMC approach for TECTL needs to generate and solve a semi-propositional formula $[[M,\psi]]_k:=[[M^{\psi,s_0}]]_k\wedge[[\psi]]_{M_k}$ where $[[M^{\psi,s_0}]]_k$ represents the transition relations of the k-paths in the k-model M_k and $[[\psi]]_{M_k}$ specifies which k-paths satisfy ψ. The satisfiability of the semi-propositional formula on some $k\leqslant|M|$ implies that M satisfies ψ. Otherwise, ψ do not hold in M.

In order to construct $[[M,\psi]]_k$, we first give some useful notions. Let w be a vector state variable, $w=(w[1],\ldots,w[n])$, where $w[i]$ for $i=1,\ldots,n$ are propositional variables and n depends on the size of the model, $n=\lceil\log_2(|M|)\rceil$. A state can be represented by a truth assignment to $(w[1],\ldots,w[n])$. When we talk about a state w, we mean the state represented by w with a given assignment. The equality $w_i\equiv w_j$ is defined by $\bigwedge_{m=1}^{n}w_i[m]\Leftrightarrow w_j[m]$. Let $k\geqslant 0$, $w_{j,i}$ represents the jth state on the ith path and $I_\varphi^{[j,i]}$ means that $w_{j,i}$ satisfies the time constraint φ. So we use $w_{0,i},\ldots,w_{k,i}$ to represent the k+1 states of the ith k-path for each $i\in\{0,\ldots,N_k(\psi)-1\}$, where $N_k(\psi)$ is the number of different k-paths needed for checking the formula φ, and it is defined as follows:

$$
\begin{aligned}
&N_k(p)=N_k(\neg p)=0,\ \text{where } p\in\mathcal{AP}\\
&N_k(\alpha\wedge\beta)=N_k(\alpha)+N_k(\beta)\\
&N_k(\alpha\vee\beta)=\max\{N_k(\alpha),N_k(\beta)\}\\
&N_k(EF_\varphi\alpha)=N_k(\alpha)+1\\
&N_k(EG_\varphi\alpha)=(k+1)\bullet N_k(\alpha)+1\\
&N_k(E(\alpha U_\varphi\beta))=k\bullet N_k(\alpha)+N_k(\beta)+1
\end{aligned}
$$

Definition 7 (**translation of the *k*-model**). Let $M_k=\langle S,path_k,s,\mathcal{L}\rangle$ be a k-model of M and ψ a TECTL formula. The semi-propositional formula $[[M,\psi]]_k$ is defined as follows:

$$[[M,\psi]]_k:=I(s)\wedge\bigwedge_{i=0}^{N_k(\psi)-1}\bigwedge_{j=0}^{k-1}T(w_{j,i},w_{j+1,i})$$

where I(s) is true when s is the initial state, $w_{j,i}$ for $j=0,\ldots,k$ and $i=0,\ldots,N_k(\psi)-1$ are vectors of state variables.

According to the bounded semantics of TECTL, we get the following definition.

Definition 8 (**translation of TECTL formulae**). Let $p\in\mathcal{AP}$, α, β TECTL formulae, m, n represent states' number and paths' number respectively. The rules of a TECTL formula ψ translated to a semi-propositional formula are defined as follows:

$$
\begin{aligned}
&[[p]]_k^{[m,n]}:=p(w_{m,n})\\
&[[\neg p]]_k^{[m,n]}:=\neg p(w_{m,n})\\
&[[\alpha\wedge\beta]]_k^{[m,n]}:=[[\alpha]]_k^{[m,n]}\wedge[[\beta]]_k^{[m,n]}\\
&[[\alpha\vee\beta]]_k^{[m,n]}:=[[\alpha]]_k^{[m,n]}\vee[[\beta]]_k^{[m,n]}\\
&[[EF_\varphi\alpha]]_k^{[m,n]}:=\bigvee_{i=0}^{N_k(EF_\varphi\alpha)-1}((w_{m,n}\equiv w_{0,i})\wedge\bigvee_{j=0}^{k}(I_\varphi^{[j,i]}\wedge[[\alpha]]_k^{[j,i]}))\\
&[[EG_\varphi\alpha]]_k^{[m,n]}:=\bigvee_{i=0}^{N_k(EG_\varphi\alpha)-1}((w_{m,n}\equiv w_{0,i})\wedge(\bigvee_{l=0}^{k}L_{k,i}(l)\vee dead(w_{k,i}))\wedge\bigwedge_{j=0}^{k'}(I_\varphi^{[j,i]}\rightarrow[[\alpha]]_k^{[j,i]}))\\
&[[E(\alpha U_\varphi\beta)]]_k^{[m,n]}:=\bigvee_{i=0}^{N_k(E(\alpha U_\varphi\beta)-1}((w_{m,n}\equiv w_{0,i})\wedge\bigvee_{j=0}^{k}((I_\varphi^{[j,i]}\wedge[[\beta]]_k^{[j,i]})\wedge\bigwedge_{t=0}^{j-1}[[\alpha]]_k^{[t,i]}))
\end{aligned}
$$

Let $[[\psi]]_{M_k}$ denote $[[\psi]]_k^{[0,0]}$ and use the arithmetic expressions to represent $I_\varphi^{[j,i]}$ directly. $L_{k,i}:=T(w_{k,i},w_{l,i})$ encodes a backloop from the kth state to the lth state in the ith k-path. k' in EG case is thesame in Definition 6.

So, from the encodings of $[[M,\psi]]_k$, we have the following theorem.

Theorem 2. Let M be a model, M_k be a k-model, and ψ a TECTL formula. Then $M_k\models_k\psi$ iff $[[M^{\psi,s_0}]]_k\wedge[[\psi]]_{M_k}$ is satisfiable.

Proof (*sketch*). Follows from the following facts:

- TECTL formula can be translated to ECTL formula which was given in Section 2.3.
- Since we have assumed that we deal with progressive TA and the TKS is an extension of TK, the above translation can be translated to the model checking problem for ECTL. Our definition of the translation of TECTL problem is based on this translation.
- Theorem 6.1 of Ref.[6] shows that the translation of the model checking of ECTL problem to the SAT problem is correct. So our translation of TECTL problem to the SMT problem is correct.

3.3 Improved SMT-based encodings

From Definition 8, the translated formula is in the form of $\phi_0\vee\phi_1\vee\ldots\vee\phi_n$, where $n\in\mathbb{N}$ and $\phi_0\vee\ldots\vee\phi_n$ are propositional formulae with the same structure but different path numbers. According to Ref.[25], it's a rotation symmetric formula. We say a formula is rotation symmetric if, for every permutation $\sigma=\{k_0\to0,\ldots,k_n\to n\}$ of its components' indexes, the permutation of a satisfiability assignment is a satisfiability assignment for the permuted formula. The rotating symmetry in the formula enables us to consider only one fixed order of paths such as the improvement in Ref.[26], yet not affecting its satisfiability.

Definition 9 (**improved SMT-based encoding**). Let $p\in\mathcal{AP}$, w a state variable, ψ, α, β TECTL formulae, $k\geqslant0$. The encoding ${}_u[[\psi,w]]_k^{i,N_k(\psi)}$ is defined as follows:

$${}_u[[p,w]]_k^{i,N_k(p)}:=p(w)$$

$${}_u[[\neg p,w]]_k^{i,N_k(\neg p)}:=\neg p(w)$$

$${}_u[[\alpha\wedge\beta,w]]_k^{i,N_k(\alpha\wedge\beta)}:={}_u[[\alpha,w]]_k^{i,g_0}\wedge{}_u[[\beta,w]]_k^{i+g_0,g_1}$$

$${}_u[[\alpha\vee\beta,w]]_k^{i,N_k(\alpha\vee\beta)}:={}_u[[\alpha,w]]_k^{i,g_0}\vee{}_u[[\beta,w]]_k^{i,g_1}$$

$${}_u[[EF_\varphi\alpha,w]]_k^{i,N_k(EF_\varphi\alpha)}:=(w\equiv w_{0,i})\wedge\bigvee_{j=0}^{k}(I_\varphi^{[j,i]}\wedge{}_u[[\alpha,w_{j,i}]]_k^{i+1,g_0})$$

$${}_u[[EG_\varphi\alpha,w]]_k^{i,N_k(EG_\varphi\alpha)}:=(w\equiv w_{0,i})\wedge(\bigvee_{l=0}^{k}L_{k,i}(l)\vee dead(w_{k,i}))\wedge\bigwedge_{j=0}^{k'}(I_\varphi^{[j,i]}\to{}_u[[\alpha,w_{j,i}]]_k^{i+1+j\times g_0,g_0})$$

$${}_u[[E(\alpha U_\varphi\beta),w]]_k^{i,N_k(E(\alpha U_\varphi\beta))}:=(w\equiv w_{0,i})\wedge\bigvee_{j=0}^{k}((I_\varphi^{[j,i]}\wedge{}_u[[\beta,w_{j,i}]]_k^{i+1,g_1})\wedge\bigwedge_{t=0}^{j-1}{}_u[[\alpha,w_{t,i}]]_k^{i+1+g_1+t\times g_0,g_0})$$

i is the start path number, $N_k(\psi)$ is the number of paths needed to check the formula ψ, $g_0=N_k(\alpha)$ and $g_1=N_k(\beta)$. Let ${}_u[[\psi]]_{M_k}$ denote ${}_u[[\psi,s_0]]_k^{0,N_k(\psi)-1}$ and use the arithmetics to represent $I_\varphi^{[j,i]}$ directly. k' in EG case is the same in Definition 6.

Theorem 3. Let M be a model, M_k a k-model, and ψ a TECTL formula. Then $M_k\models_k\psi$ iff $[[M^{\psi,s_0}]]_k\wedge{}_u[[\psi]]_{M_k}$ is satisfiable.

Proof (*sketch*). Follows from the following facts:

- TECTL formula can be translated to ECTL formula which was given in Section 2.3.
- Since we have assumed that we deal with progressive TA and the TKS is an extension of TK, the above translation can be translated to the model checking problem for ECTL. Our definition of the translation of TECTL problem is based on this translation.
- Theorem 3.2 of Ref.[26] shows that the improved translation of ECTL problem to the SAT problem is correct. So our improved translation of TECTL problem to the SMT problem is correct.

4 Experiments

The experiments were run under **RedHat Enterprise Linux AS release 4** (**Nahant update 4**) on a 2-processor **Xeon 3.0GHz** machine with **32GB RAM**. We use "SMT-based BMC" to represent our methods (it is implemented in the tool[27]), "Improved" means we use the improved translation of TECTL formulae in Definition 9, "Unimproved" means we use the translation of TECTL formulae in Definition 8. We also use SAL 3.0[15], UPPAAL 4.0.6[18], TSMV 1.0[3], NuSMV 2.4.3[2] as compariable methods.

4.1 Fischer's protocol

The system consists of $n(n\geqslant 2)$ processes: $Process_1,\ldots,Process_n$, which compete for an access to the critical region. Each process in Fischer's protocol has one local clock x and can access the global pointer $lock$ ($lock$:=p to assign its process ID to the pointer). The TA of one process was shown in Fig.1. $Process_i$'s transitions can be described as follows:

$$i = idle_i \wedge lock = nil \rightarrow (i, x_i) := (ready_i, 0)$$
$$i = ready_i \wedge x_i < B \rightarrow (i, lock, x_i) := (wait_i, p, 0)$$
$$i = wait_i \wedge lock \neq i \wedge x_i > A \rightarrow (i) := (idle_i)$$
$$i = wait_i \wedge lock = i \wedge x_i > A \rightarrow (i) := (critical_i)$$
$$i = critical_i \rightarrow (i, lock) := (idle_i, nil)$$

Experimental Results and Analysis. For n processes, we verify the negation of mutual exclusion property $\psi_1 : EF(\bigwedge_{i=1}^{n} critical_i)$. If $A\geqslant B$, ψ_1 is false, and if $A<B$, ψ_1 is true. We compare our method with SAL and UPPAAL, and TA is used as its model. The experimental results with A=1 and B=4000 are shown in Table 1.

Table 1 Experimental results for mutual exclusion (time in second, size in MB)

n	SMT-Based BMC							SAL		UPPAAL	
	k	Improved			Unimproved						
		Time	Var	Size	Time	Var	Size	Time	Size	Time	Size
2	6	0.006	102	4.08	0.024	292	4.47	0.160	8.84	0.440	1.03
3	8	0.008	179	4.21	0.077	555	4.99	1.021	11.93	0.446	1.54
4	10	0.016	278	4.35	0.192	902	5.79	2.373	15.07	0.766	2.05
5	12	0.032	399	4.47	0.421	1335	6.77	6.241	18.25	0.868	3.08
6	14	0.052	542	4.60	1.055	1848	8.27	11.612	21.47	1.056	4.10
7	16	0.074	707	4.86	2.495	2447	10.46	34.234	24.73	2.110	8.21
8	18	0.112	893	5.05	4.585	3129	12.32	64.865	28.03	14.674	29.75

The 1st column shows the number of processes. The 2nd column gives the bound of SMT-based BMC. The 3rd to 5th columns give the time, variables and memory needed for the improved SMT-based BMC method. The 6th to 8th columns are the same but for unimproved SMT-based BMC; the 9th and 10th columns display the time and memory needed for SAL. The 11th and 12th columns are the same but for UPPAAL.

From Table 1, we can see that, the SMT-based BMC method spend less time and memory than SAL and the time increasing trend of SMT-based BMC is much slower than SAL and UPPAAL with the increase of processes. Otherwise, the improved SMT-based BMC is much better than the unimproved one in the time, variables and memory they needed.

4.2 Bridge-Crossing problem

The bridge-crossing problem is a famous mathematical puzzle with time critical aspects[28]. A group of four people, called P_1, P_2, P_3 and P_4, have to cross a bridge at night. It is dark and they can only cross the bridge if they carry a lamp. Only one lamp is available and at most two of them can cross at the same time. Therefore any solution requires that, after the first two people cross the bridge, one of them returns, bringing back the lamp for the

remaining people. The four people have different maximal speeds: here P_1 crosses in 5 time units (t.u.), P_2 in 10 t.u., P_3 in 20 t.u. and P_4 in 25 t.u. When a pair crosses the bridge, they move at the speed of the slower one. Now, how much time is required before the whole group is on the other side (in the *safe* state)?

The details of this model can be found in Ref.[3]. We use TKS as the model in this problem.

Experimental Results and Analysis. We have tested two properties of this problem:

– ψ_2:$EF_{=60\times T}(safe)$

The formula expresses the property that the whole group will be safe exactly at 60 t.u.. The property is true as the minimum time to cross the bridge is $60\times T$ t.u..

– ψ_3:$EF_{\leqslant 59\times T}(safe)$

The formula expresses the property that the whole group will be safe at some time less than $60\times T$ t.u.. It is obviously false.

Table 2 Experimental results for ψ_2 (time in second, size in MB)

×T	k	SMT-Based BMC						TSMV		NuSMV	
		Improved			Unimproved						
		Time	Var	Size	Time	Var	Size	Time	Size	Time	Size
×1	5	0.012	280	4.21	0.023	442	4.34	0.006	1.82	0.084	6.17
×10	5	0.014	280	4.21	0.026	442	4.34	0.023	1.93	2.618	41.94
×20	5	0.015	280	4.21	0.024	442	4.34	0.044	2.06	8.945	45.48
×50	5	0.012	280	4.21	0.024	442	4.34	0.162	2.34	58.351	58.54
×100	5	0.016	280	4.21	0.022	442	4.34	0.466	2.76	236.662	110.71
×200	5	0.017	280	4.21	0.026	442	4.34	1.274	3.60	2608.600	204.68

Table 3 Experimental results for ψ_3 (time in second, size in MB)

×T	k	SMT-Based BMC						TSMV		NuSMV	
		Improved			Unimproved						
		Time	Var	Size	Time	Var	Size	Time	Size	Time	Size
×1	5	0.015	274	4.21	0.023	436	4.34	0.014	1.82	0.165	8.70
×10	5	0.014	274	4.21	0.026	436	4.34	0.082	2.06	5.305	42.04
×20	5	0.015	274	4.21	0.024	436	4.34	0.191	2.32	18.647	45.48
×50	5	0.014	274	4.21	0.024	436	4.34	0.643	3.03	121.671	60.62
×100	5	0.013	274	4.21	0.022	436	4.34	1.829	4.22	502.045	120.18
×200	5	0.013	274	4.21	0.026	436	4.34	5.918	6.64	6494.262	205.19

The results show in Table 2 and Table 3. The 1st column in Table 2 shows the magnifications of each one's crossing time (e.g. "×10" means replacing 5, 10, 20 and 25 with 50, 100, 200 and 250.). The 2nd to 8th columns are the same in Table 1. The 9th and 10th columns display the time and memory needed for TSMV method. Tthe 11th and 12th columns show the time and memory consumed with NuSMV method. They are the same in Table 3.

For ψ_2, the bound is 5, which means in Step 5 the system is in the *safe* state and the property is true. For ψ_3, the bound is also 5, that means in or after Step 5, the system will never reach the *safe* state within $59\times T$ t.u., and the property is false. This is because the minimal time that the system first in *safe* state is $60\times T$ t.u. at k=5, that means when k<5 the system cannot reach state *safe* and when $k\geqslant 5$ the system reaches *safe* state, as the duration of the system is monotonically increased, $D_\pi[0\ldots k]\geqslant 60\times T$, the timing constraint cannot be satisfied any more. So k=5 is enough for ψ_3.

From Table 2 and Table 3, we can see that SMT-based BMC are much faster than TSMV and NuSMV. The time SMT-based BMC used is almost the same with different magnifications, but TSMV and NuSMV are increased exponentially. Also, the performance of the improved SMT-based BMC is better than the unimproved one.

5 Conclusions and Future Work

We have considered BMC for real-time systems based on SMT solver Yices to verification TECTL properties.

We have chosen TECTL BMC as our main verification technique because of the advantage that the length of the symbolic paths needed in verifying a TECTL property could be much shorter. SMT-Based BMC can deal with two kinds of models: Timed Automata and Timed Kripke Structures. We have compared our method with some other real-time system model checkers by two well-known examples. The experimental results indicate that the efficiency of our method is better than others. We also did improvement to the SMT-based BMC and the improved one is better than the unimproved one.

For future work, stuttering studied in Ref.[29], linear counterexamples studied in Ref.[30] and some reduction methods can be incorporated into our method to simplify the *k*-model and to reduce the number of verification paths. Another direction is to work on efficient techniques for the verification of valid TECTL properties under the TA model without necessary to reach a high completeness threshold, such as Ref.[31] using *k*-induction or similar to that considered for LTL and ACTL properties in Refs.[32,33].

Acknowledgement We thank Prof. Wenhui Zhang for discussing some issues about this paper.

References:

[1] Bryant RE. Graph-Based algorithms for Boolean function manipulation. IEEE Trans. on Computers, 1986,C-35(12):1035−1044.

[2] Cimatti A, Clarke EM, Giunchiglia F, Roveri M. NuSMV: A new symbolic model verifier. In: Halbwachs N, Peled D, eds. Proc. of the 11th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 1999. 495−499.

[3] Markey N, Schnoebelen P. Symbolic model checking for simply-timed systems. In: Lakhnech Y, Yovine S, eds. Proc. of the Formal Techniques in Real-Time and Fault-Tolerant Systems. Berlin: Springer-Verlag, 2004. 102−117.

[4] Biere A, Cimatti A, Clarke E, Fujita M, Zhu Y. Symbolic model checking using SAT procedures instead of BDDs. In: Proc. of the 36th Conf. on Design Automation. ACM Press, 1999. 317−320.

[5] Biere A, Cimatti A, Clarke E, Zhu Y. Symbolic model checking without BDDs. In: Cleaveland R, ed. Proc. of the 5th Int'l Conf. on Tools and Algorithms for Construction and Analysis of Systems. Berlin: Springer-Verlag, 1999. 193−207.

[6] Penczek W, Wozna B, Zbrzezny A. Bounded model checking for the universal fragment of CTL. Fundamenta Informaticae, 2002, 51(1-2):135−156.

[7] Penczek W, Wozna B, Zbrzezny A. Towards bounded model checking for the universal fragment of TCTL. In: Damm W, Olderog ER, eds. Proc. of the Formal Techniques in Real-Time and Fault-Tolerant Systems. Berlin: Springer-Verlag, 2002. 265−290.

[8] Yu F, Wang BY, Huang YW. Bounded model checking for region automata. In: Lakhnech Y, Yovine S, eds. Proc. of the Formal Techniques in Real-Time and Fault-Tolerant Systems. Berlin: Springer-Verlag, 2004. 246−262.

[9] Alur R, Courcoubetis C, Dill DL. Model-Checking in dense real-time. Information and Computation, 1993,104(1):2−34.

[10] Koymans R. Specifying real-time properties with metric temporal logic. Real-Time Systems, 1990,2(4):255−299.

[11] Lynch NA, Attiya H. Using mappings to prove timing properties. In: Proc. of the 9th Annual ACM Symp. on Principles of Distributed Computing. ACM Press, 1990. 265−280.

[12] Dill DL. Timing assumptions and verification of finite-state concurrent systems. In: Sifakis J, ed. Proc. of the Int'l Workshop on Automatic Verification Methods for Finite State Systems. Berlin: Springer-Verlag, 1989. 197−212.

[13] Emerson EA, Mok A, Sistla AP, Srinivasan J. Quantitative temporal reasoning. In: Clarke EM, Kurshan RP, eds. Proc. of the 2nd Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 1990. 136−145.

[14] Armando A, Mantovani J, Platania L. Bounded model checking of software using SMT solvers instead of SAT solvers. In: Valmari A, ed. Proc. of the 13th Int'l SPIN Workshop. Berlin: Springer-Verlag, 2006. 146−162.

[15] http://sal.csl.sri.com/index.shtml

[16] Xu L. SMT-Based bounded model checking for real-time systems. In: Zhu H, ed. Proc. of the 8th Int'l Conf. on Quality Software. IEEE Computer Society, 2008. 120−125.

[17] Yovine S. Kronos: A verification tool for real-time systems. Journal on Software Tools for Technology Transfer, 1997,1(1-2): 123−133.

[18] Larsen KG, Pettersson P, Wang Y. UPPAAL in a nutshell. Journal on Software Tools for Technology Transfer, 1997,1(1-2): 134−152.

[19] Emerson EA, Clarke EM. Using branching-time temporal logics to synthesize synchronization skeletons. Science of Computer Programming, 1982,2(3):241−266.

[20] Dutertre B, de Moura L. A fast linear-arithmetic solver for DPLL(T). In: Ball T, Jones RB, eds. Proc. of the 18th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2006. 81−94.

[21] Bozzano M, Bruttomesso R, Cimatti A, Junttila T, Ranise S, Rossum P, Sebastiani R. Efficient satisfiability modulo theories via delayed theory combination. In: Etessami K, Rajamani SK, eds. Proc. of the 18th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2005. 335−349.

[22] Ganzinger H, Hagen G, Nieuwenhuis R, Tinelli C. DPLL(T): Fast decision procedures. In: Alur R, Peled D, eds. Proc. of the 16th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2004. 175−188.

[23] Armando A, Castellini C, Giunchiglia E, Maratea M. A SAT-based decision procedure for the Boolean combination of difference constraints. In: Hoos HH, Mitchell DG, eds. Proc. of the 7th Int'l Conf. on Theory and Applications of Satisfiability Testing. Berlin: Springer-Verlag, 2004. 16−29.

[24] Filliâtre JC, Owre S, Rueβ H, Shankar N. ICS: Integrated canonizer and solver. In: Berry G, Comon H, Finkel A, eds. Proc. of the 13th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2001. 246−249.

[25] Pieprzyk J, Qu CX. Rotation-Symmetric functions and fast Hashing. In: Boyd C, Dawson E, eds. Proc. of the 3rd Australasian Conf. on Information Security and Privacy. Berlin: Springer-Verlag, 1998. 169−180.

[26] Xu L, Chen W, Xu YY, Zhang WH. Improved bounded model checking for universal fragment of CTL. Journal of Computer Science and Technology, 2009,24(1):96−109.

[27] Wang Xiaoliang. Yices-Based bounded model checking for timed automata [MS. Thesis]. Beijing: Institute of Software, the Chinese Academy of Sciences, 2009 (in Chinese with English abstract).

[28] Rote G. Crossing the bridge at night. Bulletin of the EATCS, 2002,78:241−246.

[29] Zhou C, Ding DC. Improved SAT based bounded model checking. In: Cai JY, Cooper SB, Li A, eds. Proc. of the 3rd Int'l Conf. on Theory and Applications of Models of Computation. Berlin: Springer-Verlag, 2006. 611−620.

[30] Buccafurri F, Eiter T, Gottlob G, Leone N. On ACTL formulae having linear counterexamples. Journal of Computer and System Sciences, 2001,62(3):463−515.

[31] De Moura L, Rue H, Sorea M. Bounded model checking and induction: From refutation to verification. In: Hunt Jr. WA, Somenzi F, eds. Proc. of the 15th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2003. 14−26.

[32] Zhang W. SAT-Based verification of LTL formulas. In: Brim L, Haverkort B, Leucker M, van de Pol J, eds. Proc. of the 11th Int'l Workshop on Formal Methods: Applications and Technology. Berlin: Springer-Verlag, 2006. 277−292.

[33] Zhang W. Model checking with SAT-based characterization of ACTL formulas. In: Butler M, Hinchey MG, Larrondo-Petrie MM, eds. Proc. of the 9th Int'l Conf. on Formal Engineering Methods. Berlin: Springer-Verlag, 2007. 191−211.

附中文参考文献:

[27] 王晓亮.基于 Yices 对时间自动机的有界模型检测[硕士学位论文].北京:中国科学院软件研究所,2009.

XU Liang (Ph.D.) was born in 1981. He is a R&D Engineer at the National Engineering Research Center of Fundamental Software. His current research areas are bounded model checking, formal verification, formal security policy and so on.