ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.6, June 2009, pp.1414−1424
doi: 10.3724/SP.J.1001.2009.00558
© by *Institute of Software, the Chinese Academy of Sciences*. All rights reserved.

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

一种基于满足性判定的并发软件验证策略[*]

周从华

(江苏大学 计算机科学与通信工程学院,江苏 镇江 212013)

SAT-Based Compositional Verification Strategy for Concurrent Software with States, Events

ZHOU Cong-Hua

(School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013, China)

+ Corresponding author: E-mail: chzhou@ujs.edu.cn

Zhou CH. SAT-Based compositional verification strategy for concurrent software with states, events. *Journal of Software*, 2009,20(6):1414−1424. http://www.jos.org.cn/1000-9825/558.htm

Abstract: For the state/event linear temporal logic SE-LTL, an SAT-based Bounded Model Checking procedure which avoids the space blow up of BDDs is presented. For $SE\text{-}LTL_{-X}$, it is shown how to integrate the procedure and the stuttering equivalent technique. The integration speeds up the verification procedure. Furthermore, a framework for model checking concurrent software systems which integrates three powerful verification techniques is presented: SAT-based Bounded Model Checking, counterexample-guided abstraction refinement and compositional reasoning. In the framework the abstraction and refinement steps are performed over each component separately, and the model checking step is symbolic. Example shows that the framework can reduce verification time and space.

Key words: bounded model checking; abstract; parallel composition

摘 要: 对线性时态逻辑SE-LTL提出了一种基于SAT的有界模型检测过程,该过程避免了基于BDD方法中状态空间快速增长的问题.在SE-LTL的子集$SE\text{-}LTL_{-X}$的有界模型检测过程中,集成了stuttering等价技术,该集成有效地加速了验证过程.进一步提出了一种组合了基于SAT的有界模型检测、基于反例的抽象求精、组合推理 3 种状态空间约简技术的并发软件验证策略.该策略中,抽象和求精在每一个构件上独立进行.同时,模型检测的过程是符号化的.实例表明,该策略降低了验证时间和对内存空间的需求.

关键词: 有界模型检测;抽象;平行组合

中图法分类号: TP301 **文献标识码**: A

1 Introduction

In order to represent both software implementations and specifications directly without any program

---

* Supported by the National Natural Science Foundation of China under Grant No.60773049 (国家自然科学基金); the Advanced Talent Foundation of Jiangsu University of China under Grant No.07JDG014 (江苏大学高级人才科研启动基金); the Fundamental Research Project of the Natural Science in Colleges of Jiangsu Province of China under Grant No.08KJD520015 (江苏省高校自然科学基金)

Received 2007-05-30; Accepted 2008-03-06

annotations or privileged insights into program execution, Chaki *et al*. proposed a specification language SE-LTL[1] which is a state/event derivative of LTL[1]. The state space explosion[2] is the main problem in checking a system satisfying an SE-LTL formula. SAT-based Bounded Model Checking (BMC)[2,3] has recently been introduced as an effective technique to overcome the state space explosion. Therefore it is necessary to develop a BMC procedure for SE-LTL. To the best of our knowledge, at present a BMC procedure does not exist.

For SE-LTL, we first provide its SAT-based BMC procedure. Given an LKS M, an SE-LTL formula ϕ and a natural number k, our BMC procedure decides whether there exists a computation in M of length k or less that violates ϕ, i.e. $M|=_k E\neg\phi$. Our BMC is performed by generating a proposition formula, which is satisfiable if and only if such a computation exists. For $SE\text{-}LTL_{-X}$ properties we further show how to integrate our BMC procedure and stuttering equivalent technique[4]. Experiments show that the integration can reduce verification time significantly.

Then we present an efficient verification strategy which combines SAT-based BMC, counterexample-guided abstraction refinement[5−7] and compositional reasoning[8]: starting with a coarse initial abstraction, our scheme computes increasingly precise abstractions of the target system by analyzing spurious counterexamples until either a real counterexample is obtained or the system is found to be correct. Of the three steps in this abstract-verify-refine process only the verification stage of our technique requires the explicit composition of a system. The other two stages can be performed one component at a time. To the best of our knowledge, our strategy is the first SAT-based, counterexample-guided, compositional abstraction refinement scheme to perform verification of linear time temporal specifications.

2 Preliminaries

2.1 Labeled kripke structure

Definition 1. *A* labeled Kripke structure M is a 6-tuple (S,s_0,AP,Σ,L,R) where S is a finite non-empty set of states, $s_0\in S$ is an initial state, AP is a finite set of atomic state propositions, Σ is a finite set of events, $L:S\rightarrow 2^{AP}$ is a state-labeling function, $R\subseteq S\times\Sigma\times S$ is a transition relation that must be total, that is, for every state $s\in S$ there is an event $a\in\Sigma$ and a state $s'\in S$ such that $R(s,a,s')$ holds.

Given an LKS $M=(S,s_0,AP,\Sigma,L,R)$ we write $S(M)$, $s_0(M)$, $AP(M)$, $L(M)$, $\Sigma(M)$ and $R(M)$ to mean S, s_0, AP, L, Σ and R respectively. A path $\pi=s_0,a_0,s_1,a_1,\ldots$ of M is an alternating infinite sequence of states and events subject to the following: for each $i\geq 0$, $s_i\in S$, $a_i\in\Sigma$ and $R(s_i,a_i,s_{i+1})$ holds. For the path $\pi=s_0,a_0,s_1,a_1,\ldots$, we use $\pi(i)$ to denote the i-th state s_i, use $\pi(i,E)$ to denote the i-th event a_i. We write $Path(M)$ to denote the set of infinite and finite paths whose first state is $s_0(M)$.

Definition 2. A path π is a (k,l)-loop with $l<k$, if $(\pi(k),\pi(k,E),\pi(l))\in R$ and $\pi=u\cdot v^{\omega}$, where $u=\pi(0),\pi(0,E),\ldots,\pi(l-1),\pi(l-1,E)$ and $v=\pi(l),\pi(l,E),\ldots,\pi(k),\pi(k,E)$. We call π simply a k-loop if there is an integer l with $0\leq l\leq k$ such that π is a (k,l)-loop.

2.2 Abstraction

Let $M=(S,s_0,AP,\Sigma,L,R)$ and $A=(S^A,s_0^A,AP^A,\Sigma^A,L^A,R^A)$ be two LKSs. We say that A is an abstraction of M, written $M\subseteq A$ iff: 1) $AP^A\subseteq AP$; 2) $\Sigma^A=\Sigma$; 3) For every path $\pi=s_0,a_0,s_1,a_1,\ldots$ of M there exists a path $\pi'=s'_0,a'_0,s'_1,a'_1,\ldots$ of A such that for each $i\geq 0$, $a'_i=a_i$ and $L^A(s'_i)=L(s_i)\cap AP^A$. The abstraction of M defined as above over-approximates the behaviors of M, that is a behavior of M is also a behavior of A.

2.3 Existential quotients of labeled kripke structures

We use the method introduced by Chaki *et al*. in Ref.[1] to construct abstractions. An abstraction of an LKS M

is obtained by quotienting the states of M by a suitable equivalence relation. More precisely, for $M=(S,s_0,AP,\Sigma,L,R)$, Let $AP^A\subseteq AP$ be a subset of atomic state propositions of M, and let $\approx$ be an equivalence relation on the states S of M that respects AP^A. The existential quotient of M (with respect to AP^A and $\approx$) is the LKS $A=(S^A,s_0^A,AP^A,\Sigma^A,L^A,R^A)$ such that: 1) $S^A=S/\approx$, the collection of equivalence classes of S; 2) $s_0^A=[s_0]$; 3) for all $s\in S$, $L^A([s])=L(s)\cap AP^A$; 4) $\Sigma^A=\Sigma$; 5) for all $s,s'\in S$ and $a\in\Sigma$, $([s_1],a,[s_2])\in R^A$ iff there exists $s_1'\in[s_1]$, $s_2'\in[s_2]$ such that $(s_1',a,s_2')\in R$.
We write $M/\approx$ (when the set AP^A is clearly understood from the context) to denote the abstraction A of M obtained in the above manner.

2.4 Parallel composition

The notion of parallel composition we consider in this paper is adapted from Ref.[1] which allows for communication through shared events only; in particular we forbid the sharing of variables. This restriction facilitates the use of compositional reasoning in verifying specification.

Let $M_1=(S_1,s_0^1,AP_1,\Sigma_1,L_1,R_1)$ and $M_2=(S_2,s_0^2,AP_2,\Sigma_2,L_2,R_2)$ be two LKSs. M_1 and M_2 are said to be compatible, i.e., that they do not share variables: $S_1\cap S_2=AP_1\cap AP_2=\varnothing$. The parallel composition of M_1 and M_2 is given by $M_1\,||\,M_2=(S_1\times S_2,s_0^1\times s_0^1,AP_1\cup AP_2,\Sigma_1\cup\Sigma_2,L_1\cup L_2,R)$, where $(L_1\cup L_2)(s_1,s_2)=L_1(s_1)\cup L_2(s_2)$ and $R((s_1,s_2),a,(s_1',s_2'))$ holds iff one of the following holds: 1) $a\in\Sigma_1\backslash\Sigma_2$ and $R_1(s_1,a,s_1')$ holds and $s_2'=s_2$; 2) $a\in\Sigma_2\backslash\Sigma_1$ and $R_2(s_2,a,s_2')$ holds and $s_1'=s_1$; 3) $a\in\Sigma_1\backslash\Sigma_2$, $R_1(s_1,a,s_1')$ holds and $R_2(s_2,a,s_2')$ holds.

Let M_1 and M_2 be as above, and let $\pi=(s_0^1,s_0^2),a_0,\ldots$ be an alternating infinite sequence of states and events of $M_1||M_2$. The projection $\pi\upharpoonleft M_i$ of π on M_i consists of (possibly finite) the subsequence of $s_0^i,a_0,\ldots$ obtained by simply removing all pairs (a_j,s_{j+1}^i) for which $a_j\notin\Sigma_i$. In other words, we keep from π only those states that belong to M_i, and excise any transition labeled with an event not in Σ_i. We now introduce the following theorems, which is useful for our composition verification strategy.

Theorem 1[1]. Let $M_1,\ldots,M_n$ be compatible LKSs, and let π be an infinite alternating sequence of states and events of the composition $M_1||\ldots||M_n$. Then π is a path of $M_1||\ldots||M_n$ iff, for each i, there exists a path π_i' of M_i such that $\pi\upharpoonleft M_i$ is a prefix of π_i'. In other words, whether a path belongs to the composition of LKSs can be checked by projecting and examining the path on each individual component separately.

Theorem 2[1]. Let $M_1,\ldots,M_n$ be compatible LKSs, and let $A_1,\ldots,A_n$ be respective abstractions of the M_i: for each i, $M_i\subseteq A_i$. Then $M_1,\ldots,M_n\subseteq A_1,\ldots,A_n$. In other words, parallel composition preserves the abstraction relation.

2.5 State/Event linear temporal logic SE-LTL

Given an LKS $M=(S,s_0,AP,\Sigma,L,R)$, we consider linear temporal logic state/event formulas SE-LTL over the sets AP, Σ: $\phi::=p\,|\,a\,|\,\phi\wedge\phi\,|\,\phi\vee\phi\,|\,X\phi\,|\,G\phi\,|\,F\phi\,|\,\phi U\phi$. About the semantics of SE-LTL, readers can refer to Ref.[1]. We introduce the notation $M|=f$ which represents that for all path π of M, $\pi|=f$, and the notation. $M|=Ef$ which represents there is a path π of M such that $\pi|=f$.

3 SAT-Based Bounded Model Checking for SE-LTL

Bounded model checking based on SAT methods has been introduced as a complementary technique to BDD-based symbolic model checking. The main idea of bounded model checking is to search for an execution of the system of some length k, which constitutes a counterexample for a verified property.

3.1 Bounded semantics for SE-LTL

In bounded model checking a crucial observation is that the prefix of a path is finite, it still might represent an infinite path if there is a back loop from the last state of the prefix to any of the previous states. If there is no such back loop, then the prefix does not say anything about the infinite behavior of the path. Thus when we define bounded semantics for SE-LTL, we must consider whether a finite path represents an infinite behavior.

Definition 3 (**bounded semantics for a loop**). Let π be a k-loop. Then an SE-LTL formula f is valid along the path π with bound k (written as $\pi|=_k f$) iff $\pi|=f$.

Definition 4 (**bounded semantics without a loop**). Let π be a path that is not k-loop. Then an SE-LTL formula f is valid along π with bound k (written as $\pi|=_k f$) iff $\pi |=_k^0 f$ where $i\le k$ and

$-\pi |=_k^i p$ iff $p \in L(\pi(i)); \pi |=_k^i \neg p$ iff $p \notin L(\pi(i))$; $\pi |=_k^i a$ iff $a \equiv \pi(i,E)$;

$-\pi |=_k^i f \wedge g$ iff $\pi |=_k^i f$ and $\pi |=_k^i g$; $\pi |=_k^i f \vee g$ iff $\pi |=_k^i f$ or $\pi |=_k^i g$;

$-\pi |=_k^i Gf$ is always false; $\pi |=_k^i Ff$ iff $\exists j, i \le j \le k, \pi |=_k^j f$; $\pi |=_k^i Xf$ iff $i\le k$ and $\pi |=_k^{i+1} f$;

$-\pi |=_k^i fUg$ iff $\exists j, i \le j \le k\ [\pi |=_k^j g$ and $\forall n, i \le n < k, \pi |=_k^n f]$.

We use the notation $M|=_k Ef$ to represent that there exists a path π of M such that $\pi|=_k f$.

Theorem 3. Let AP be a set of propositions, M be an LKS over AP, π be a path of M, f be an SE-LTL formula, and k be a bound. Then $\pi|=_k f$ implies $\pi|=f$.

Definition 5[9]. For every LKS M and an SE-LTL property f, the natural number k called a CT of f if and only if the following condition holds: if there is no counterexample to f in M of length k or less, then $M|=f$.

Theorem 4. Let AP be a set of propositions, M be an LKS over AP, π be a path of M, f be an SE-LTL formula, and k be a natural number. Then $M|=Ef$ implies there exists a bound $k\le|M|\times 2^{|f|}$ such that $M|=_k Ef$. In other words, $|M|\times 2^{|f|}$ is a CT of f.

Proof: In Ref.[1] it has shown that every SE-LTL formula f can be translated into a Büchi automaton $B(f)$ such that $B(f)$ accepts exactly the words (paths) that satisfy f. Therefore existential SE-LTL model-checking can be done as follows: Given an SE-LTL formula f, construct $B(f)$, a Büchi automaton that accepts exactly those paths that satisfy f. Then, check whether $M\times B(f)$ is non-empty. It is straightforward to see that $M|=Ef$ if and only if $M\times B(f)$ is nonempty. Thus, SE-LTL model checking is reduced to the question of Büchi automaton non-emptiness, i.e., proving that there is a word accepted by the product automaton $M\times B(f)$. In order to prove non-emptiness, one has to show that there is a computation of $M\times B(f)$ passing through an accepting state an infinite number of times. That is there exists a path in $M\times B(f)$ that starts with the initial state and ends with a cycle in the strongly connected component including an accepting state. This path can be chosen to be a k-loop with k bounded by $|M|\times 2^{|f|}$ which is the size of $M\times B(f)$. If we project this path onto its first component, the original LKS, then we get a path π that is a k-loop and in addition fulfills $\pi|=f$. By the definition of the bounded semantics this also implies $\pi|=_k f$.

3.2 Translation

Given an LKS M, an SE-LTL formula f and a bound k, we will construct a proposition formula $[M,f]_k$. Let $s_0,a_0,\ldots s_k,a_k$ be a finite sequences of states and events on a path π. Each s_i represents a state at time step i and consists of an assignment of truth values to the set of state variables. Each a_i represents an event at time step i and consists of an assignment of truth values to the set of event variables. The formula $[M,f]_k$ encodes constraints on $s_0,a_0,\ldots s_k,a_k$ such that $[M,f]_k$ is satisfiable iff π is a witness for f. The definition of formula $[M,f]_k$ will be presented as three separate components. We first define a proposition formula $[M]_k$ that constraints $s_0,a_0,\ldots s_k,a_k$ to be a valid path starting from the initial state. We then define the loop condition, which is a proposition formula that is evaluated to true only if the path π contains a loop. Finally, we define a proposition formula that constrains π to satisfy f.

Definition 6 (**unfolding the transition relation**). For the bound k, we define $[M]_k := I(s_0) \overset{k-1}{\underset{i=0}{\wedge}} R(s_i, a_i, s_{i+1})$, where $I(s_0)$ is true if and only if s_0 is the initial state.

The translation of an SE-LTL formula depends on the shape of the path π. We define the proposition formula ${}_lL_k$ to be true if and only if there is a transition from sate s_k to state s_l.

Definition 7 (**loop condition**). For two integers k,l with $k \geq l \geq 0$, let ${}_lL_k := R(s_k, a_k, s_l)$.

Depending on whether a path is a k-loop, we have two different translations of an SE-LTL formula f. First we consider the case where the path is a k-loop. We give a recursive translation of an SE-LTL formula f for a k-loop path π. The translation of f recurses over its subterms and the states in π. The intermediate formula ${}_l[\cdot]_k^i$ depends on three parameters: l,k and i. We use l for the start position of the loop, k for the bound, and i for the current position in π.

Definition 8 (**translation of an SE-LTL formula on a (*k*,*l*)-loop**).

$-\,{}_l[a]_k^i := (a \equiv a_i)$; ${}_l[p]_k^i := p \in L(s_i)$; ${}_l[f \wedge g]_k^i := {}_l[f]_k^i \wedge {}_l[g]_k^i$; ${}_l[f \vee g]_k^i := {}_l[f]_k^i \vee {}_l[g]_k^i$

$-\,{}_l[\neg p]_k^i := p \notin L(s_i)$; ${}_l[Xf]_k^i :=$ if $i<k$ then $[f]_k^{i+1}$ else $[f]_k^l$

$-\,{}_l[Ff]_k^i := \overset{k}{\underset{j=\min(i,l)}{\vee}} {}_l[f]_k^i$; ${}_l[Gf]_k^i := \overset{k}{\underset{j=\min(i,l)}{\wedge}} {}_l[f]_k^i$

$-\,{}_l[fUg]_k^i := \overset{k}{\underset{j=i}{\vee}} ({}_l[g]_k^i \wedge \overset{i-1}{\underset{h=i}{\wedge}} [f]_k^h) \vee \overset{i-1}{\underset{j=i}{\vee}} ({}_l[g]_k^i \wedge \overset{k}{\underset{h=i}{\wedge}} {}_l[f]_k^h \wedge \overset{j-1}{\underset{h=i}{\wedge}} {}_l[f]_k^h)$

For the case where π is not a k-loop, the translation can be treated as a special case of the k-loop translation. For LKS with total transition relations, every finite path π can be extended to an infinite one. Since the property of the path beyond event a_k is unknown, we make a conservative approximation and assume all properties beyond a_k are false.

Definition 9 (**translation of an SE-LTL formula without a loop**).

$-[a]_k^i := (a \equiv a_i)$; $[p]_k^i := p \in L(s_i)$; $[f \wedge g]_k^i := [f]_k^i \wedge [g]_k^i$; $[\neg p]_k^i := p \notin L(s_i)$; $[f \vee g]_k^i := [f]_k^i \vee [g]_k^i$

$-[Xf]_k^i :=$ if $i<k$ then $[f]_k^{i+1}$ else false; $[Ff]_k^i := \overset{k}{\underset{j=i}{\vee}} [f]_k^i$;$[Gf]_k^i :=$ false; $[fUg]_k^i := \overset{k}{\underset{j=i}{\vee}} ([g]_k^i \wedge \overset{j-1}{\underset{h=i}{\wedge}} [f]_k^h)$

Combining all components, the encoding of a bounded model checking problem in proposition logic is defined as follows.

Definition 10 (**general translation**). Let M be an LKS, f be an SE-LTL formula, and k be a bound. We define $[f]_k := ([f]_k^0 \vee \overset{k}{\underset{l=0}{\vee}} ({}_lL_k \wedge {}_l[f]_k^0))$ and $[M,f]_k := [M]_k \wedge [f]_k$.

The translation scheme guarantees the following theorem, which we state without proof.

Theorem 5. Let M be an LKS, f be an SE-LTL formula, and k be a bound. Then $[M,f]_k$ is satisfiable if and only if $M \models_k Ef$.

Thus, the reduction of bounded model checking to SAT is sound and complete with respect to the bounded semantics. The following Corollary 1 is straightforward from Theorem 4 and Theorem 5.

Corollary 1. Let M be an LKS, f be an SE-LTL formula, and k be a bound. $M \models Ef$ if and only if there exists an integer $k \leq |M| \times 2^{|f|}$ such that $[M,f]_k$ is satisfiable.

3.3 An example

Let us consider the mutual exclusion example modeled by a Petri net[10] $PN=(P,T,s_0,F)$, where $P=\{p_1,\ldots,p_5\}$ is a set of places, $T=\{t_1,t_2,t_3,t_4\}$ are a set of transitions, $s_0(p_1)=s_0(p_3)=s_0(p_5)=1$ and $s_0(p_2)=s_0(p_4)=0$ is the initial state, $F=\{(p_1,t_1),(p_3,t_1),(t_1,p_2),(p_2,t_2),(t_2,p_1),(t_2,p_3),(p_3,t_3),(p_5,t_3),(t_3,p_4),(p_4,t_4),(t_4,p_3),(t_4,p_5)\}$ is a set of arcs. Each state s of the system PN is represented by five bit variables: p_1,p_2,p_3,p_4,p_5. We use two bit variables to encode events: $00 \leftrightarrow t_1$, 01$\leftrightarrow t_2$, 10$\leftrightarrow t_3$, 11$\leftrightarrow t_4$. We use $a[1]$ for the high bit and $a[0]$ for the low bit.

The initial state is represented as follows: $I(s):=p_1\wedge(\neg p_2)\wedge p_3\wedge(\neg p_4)\wedge p_5$.

The transition relation is represented as follows,

$$\begin{aligned}R(s,a,s'):=&(p_1\wedge\neg p_1'\wedge p_2'\wedge p_3\wedge\neg p_3'\wedge(p_4\leftrightarrow p_4')\wedge(p_5\leftrightarrow p_5')\wedge\neg a[0]\wedge\neg a[1])\vee\\&(p_1'\wedge p_2\wedge\neg p_2'\wedge p_3'\wedge(p_4\leftrightarrow p_4')\wedge(p_5\leftrightarrow p_5')\wedge a[0]\wedge\neg a[1])\vee\\&(p_3\wedge\neg p_3'\wedge p_4'\wedge p_5\wedge\neg p_5'\wedge(p_1\leftrightarrow p_1')\wedge(p_2\leftrightarrow p_2')\wedge\neg a[0]\wedge a[1])\vee\\&(p_3'\wedge\neg p_5'\wedge p_4\wedge\neg p_4'\wedge(p_1\leftrightarrow p_1')\wedge(p_2\leftrightarrow p_2')\wedge a[0]\wedge a[1]).\end{aligned}$$

We now add a faulty transition: if t_2 is fired, then there is a token in p_3, no token in p_2, and the number of tokens in other places does not change. We denote by R_f the new faulty transition relation.

$$R_f(s,a,s'):=R(s,a,s')\vee(p_2\wedge\neg p_2'\wedge p_3'\wedge(p_1\leftrightarrow p_1')\wedge(p_4\leftrightarrow p_4')\wedge(p_5\leftrightarrow p_5')\wedge a[0]\wedge\neg a[1]).$$

Consider the property that if t_2 is fired then there is a token in p_1. The property can be represented as $G(t_2\rightarrow Xp_1)$. Using bounded model checking, we attempt to find a counterexample of the property, or, in other words, look for a witness for $F(t_2\wedge X\neg p_1)$. Let us consider a case where the bound k=2. Unrolling the transition relation results in the following formula: $[M]_2:=I(s_0)\wedge R_f(s_0,a_0,s_1)\wedge R_f(s_1,a_1,s_2)$. The loop condition is: $L_2:=\bigvee_{i=0}^{2}R_f(s_2,a_2,s_i)$. The transition for paths without loops is ($p_1(s_i)$ denotes $p_1\in L(s_i)$):

$$\begin{aligned}&[F(t_2\wedge X\neg p_1)]_2^0:=a_0[0]\wedge\neg a_0[1]\wedge p_1(s_1)\vee[F(t_2\wedge X\neg p_1)]_2^1,\\&[F(t_2\wedge Xp_1)]_2^1:=a_1[0]\wedge\neg a_1[1]\wedge\neg p_1(s_2)\vee[F(t_2\wedge X\neg p_1)]_2^2,\\&[F(t_2\wedge X\neg p_1)]_2^2:=0.\end{aligned}$$

The transition with loops can be done similarly. Putting everything together we get the following Boolean formula:

$$\begin{aligned}[M,F(t_2\wedge X\neg p_1)]_2:=&I(s_0)\wedge R_f(s_0,a_0,s_1)\wedge R_f(s_1,a_1,s_2)\wedge((a_0[0]\wedge\neg a_0[1]\wedge\neg p_1(s_1))\vee a_1[0]\wedge\neg a_1[1]\wedge\neg p_1(s_2))\vee\\&(R_f(s_2,a_2,s_0)\wedge a_2[0]\wedge\neg a_2[1]\wedge\neg p_1(s_0))\vee(R_f(s_2,a_2,s_1)\wedge a_2[0]\wedge\neg a_2[1]\wedge\neg p_1(s_0))\vee\\&(R_f(s_2,a_2,s_2)\wedge a_2[0]\wedge\neg a_2[1]\wedge\neg p_1(s_2)))\,.\end{aligned}$$

The assignment s_0: 10101, a_0: 00, s_1: 01001, a_1: 01, s_2: 00101 satisfies $[M,F(t_2\wedge X\neg p_1)]_2$. This assignment corresponds to a path from the initial state to the state 00101 that violates the property $G(t_2\rightarrow Xp_1)$.

3.4 Experiments

We have implemented the deadlock detection and SE-LTL model checking translations in a bounded model checker SE-BMC. As benchmarks we use the *DP*(x), *ELEVATOR*(x), *RING*(x) problems which are from Ref.[11]. The experimental results can be found in Table 1. The columns of the table are the following: Problem: The problem name with the size of the instance in parenthesis; $|P|$: Number of places in the original net; $|T|$: Number of transitions in the original net; SE_s: Time needed for the SE-BMC engine to find a deadlock; BDD_s: Time needed for the NuSMV/BDD engine to check the deadlock problem; States: Number of reachable states of the model.

Overall, the results are promising, in particular. However, we need to get a better understanding of the behavior of the bounded model checking approach by doing more experiments.

4 Compositional SAT-Based SE-LTL Verification

We now discuss how to verify SE-LTL specification on parallel compositions of LKSs incrementally and compositionally. When trying to determine whether an SE-LTL specification holds on a given LKS, the following result is the key ingredient needed to exploit abstractions in the verification process.

Table 1 Deadlock checking experiments

Problem	$\|P\|$	$\|T\|$	SE_s	BDD_s	States
DP(6)	36	24	6.7	102.3	728
DP(8)	48	32	107.2	>2500	6554
DP(10)	60	49	1674.1	>2500	48896
DP(12)	72	48	>2500	>2500	>350000
ELEVATOR(1)	63	99	11.3	67.5	158
ELEVATOR(2)	146	299	56.4	143.7	1062
ELEVATOR(3)	327	783	675.2	>2500	7121
ELEVATOR(4)	736	1939	>2500	>2500	43440
RING(3)	39	33	4.5	23.1	87
RING(5)	65	55	68.7	179.8	1290
RING(7)	91	77	835.6	>2500	17000

Theorem 6. Let M and A be LKSs with $M\sqsubseteq A$. Then for any SE-LTL formula ϕ over M which mentions only propositions (and events) of A, if $A|=\phi$ then $M|=\phi$.

We can prove Theorem 6 by induction on the structure of ϕ directly. Here, we omit the proof details. Suppose now that we are given a collection $M_1,\dots,M_n$ of LKSs, as well as an SE-LTL specification ϕ, with the task of determining whether $M_1||\dots||M_n|=\phi$. We first create initial abstractions $M_1\sqsubseteq A_1,\dots,M_n\sqsubseteq A_n$. Then we check whether $A_1||\dots||A_n|=_kE\neg\phi$. In the affirmative, we are given with an abstract counterexample π such that $\pi|=_k\neg\phi$. We must then determine whether this counterexample is real or spurious, i.e., whether it corresponds to a counterexample in $M_1||\dots||M_n$ or not. In the negative, we check whether $k=CT$. If so, then check whether $A_1||\dots||=_{k+1}E\neg\phi$. Otherwise return that $M_1||\dots||M_n|=\phi$.

This counterexample validation check can be performed compositionally as follows. By Theorem 1, the counterexample is real iff for each i, the projection $\pi\uparrow A_i$ corresponds to (the prefix of) a valid behavior of M_i. To this end, we 'simulate' $\pi\uparrow A_i$ on M_i. If M_i accepts the path, we go on to the next component. Otherwise, we refine our abstraction A_i, yielding a new abstraction A_i' with $M_i\subseteq A_i'\subseteq A_i$ and such that A_i' also rejects the projection $\pi\uparrow A_i'$ of the spurious counterexample π.

Let us return to our SE-LTL specification ϕ, and let us fix throughout P_ϕ to be the set of all atomic state propositions appearing in ϕ. The initial abstraction $M_i/\approx_i^1$ is the coarsest possible: $s\approx_i^1 s'$ iff $L(M_i)(s)\cap P_\phi=L(M_i)(s')\cap P_\phi$. Suppose now that we are handed $\pi_i\in path(M_i/\approx_i^{k+1})$. We must determine whether π_i is a real or spurious counterexample component, i.e., whether π_i gives rise to a valid path of M_i or not. Moreover, in the latter case, we want to refine our partition $\approx_i^k$ into $\approx_i^{k+1}$ so that π_i is rejected by $M_i/\approx_i^{k+1}$. The validation/refinement step is similar to that originally proposed by Chaki et al. in Ref.[1]. The full algorithm for checking whether $M_1||\dots||M_n|=\phi$ is given in Algorithm 1. Note that the abstraction, counterexample-validation, and refinement steps are all performed one component at a time.

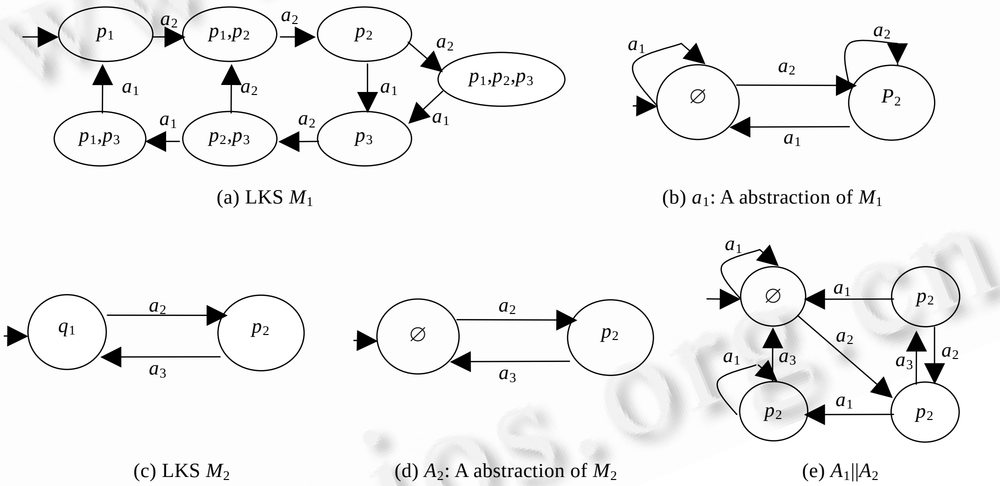
Algorithm 1. **SE-LTL model checking** ($M_1,\dots,M_n;\phi$).

```
for $i$:=1 to $n$: let $A_i$ be the initial abstract of $M_i$;
for $k$:=1 to $CT$ of $M_1||\dots||M_n$
{(1) decide whether $[A_1||\dots||A_n,\neg\phi]_k$ is satisfiable
if $[A_1||\dots||A_n,\neg\phi]_k$ is satisfiable, then suppose
$\pi$ be path in $A_1||\dots||A_n$ violating $\phi$
looking for $i$ such that $\pi\uparrow A_i$ is spurious
if no such $i$ then
    return $M_1||\dots||\neq\phi$ along with the counterexample derived from $\pi$
else refine $A_i$, yielding a new abstraction $A_i'$ with $M_i\subseteq A_i'\subseteq A_i$ and $A_i'$ also rejects $\pi\uparrow A_i'$.
Let $A_i = A_i'$, goto (1)
else if $[A_1||\ldots||A_n,\neg\phi]_k$ is unsatisfiable
if $k=CT$ then return $M_1||\ldots||M_n|=\phi$
else $k:=k+1$.}
```

4.1 An example

In this subsection we will evaluate our compositional SAT-based SE-LTL verification framework by an example shown in Fig.1. In Fig.1 M_1,M_2 are two LKSs. The property that we want to check is if event a_2 occurs then the atomic proposition p_2 in the successor of current state is true. The property expressed by SE-LTL is $\phi=G(a_2\rightarrow Xp_2)$.

We first define the equivalent relation $\approx$. Since p_2 is the only atomic proposition in $G(a_2\rightarrow Xp_2)$, let $P_\phi=\{p_2\}$. We define $s\approx s'$ if and only if $L(s)\cap P_\phi=L(s')\cap P_\phi$. Then we compute the abstractions of M_1 and M_2. In Fig.1 A_1 (Fig.1(b)), A_2 (Fig.1(d)) are abstractions of M_1 and M_2 respectively. Then we compute the parallel composition of A_1 and A_2, i.e. $A_1||A_2$ (Fig.1(e)). After implementing Algorithm 1, we found $A_1||A_2$ has four states, CT=3, and $A_1||A_2|=\phi$. By Theorem 6, $M_1||M_2|=G(a_2\rightarrow Xp_2)$. However, If we check $G(a_2\rightarrow Xp_2)$ by bounded model checking on $M_1||M_2$ directly, then the verified system has fourteen states and CT=9. Therefore, this example shows that our compositional verification framework can reduce verification time and memory space.



(a) LKS M_1 (b) a_1: A abstraction of M_1

(c) LKS M_2 (d) A_2: A abstraction of M_2 (e) $A_1||A_2$

Fig.1 An example of compositional SAT-based SE-LTL verification

5 Stuttering Equivalent in SE-LTL Model Checking

In this section, we show that the stuttering equivalent is also efficient in SAT-based BMC. We will use stuttering equivalent to make SAT solvers used in BMC avoid searching for equivalent bounded paths.

Definition 11 (**bounded stuttering equivalent**). Let π,π' be two paths, m,n be two integers. We call π,π' are (m,n) bounded stuttering equivalent if and only if one of the following two conditions holds.

(1) For each $0\le l\le m$, if $(\pi(m),\pi(m,E),\pi(l))\in R$ then there exists an integer h with $0\le h\le n$ such that $(\pi'(n),\pi'(n,E),\pi'(h))\in R$ and two paths $(\pi(0),\pi(0,E),\ldots,\pi(l-1),\pi(l-1,E),(\pi(l),\pi(l,E),\ldots,\pi(m),\pi(m,E))^\omega)$, $(\pi'(0),\pi'(0,E),\ldots,\pi'(h-1),\pi'(h-1,E),(\pi'(h),\pi'(h,E),\ldots,\pi'(n),\pi'(n,E))^\omega)$ are stuttering equivalent. For each $0\le h\le n$, if $(\pi'(n),\pi'(n,E),\pi'(h))\in R$ then there exists an integer l with $0\le l\le m$ such that $(\pi(m),\pi(m,E),\pi(l))\in R$ and two paths $\pi(0),\pi(0,E),\ldots,\pi(l-1),\pi(l-1,E),(\pi(l),\pi(l,E),\ldots,\pi(m),\pi(m,E))^\omega$, $\pi'(0),\pi'(0,E),\ldots,\pi'(h-1),\pi'(h-1,E),(\pi'(h),\pi'(h,E),\ldots,\pi'(n),\pi'(n,E))^\omega$ are stuttering

equivalent.

(2) π is not a m loop path, π' is not a n loop path. And there are two finite sequences of integers $0=i_0<i_1<i_2<\ldots<i_l$ and $0=j_0<j_1<j_2<\ldots<j_l$ such that for every $0\le x<l$,

$$L(\pi(i_x))=L(\pi(i_x+1))=\ldots=L(\pi(i_{x+1}-1))=L(\pi'(j_x))=L(\pi'(j_x+1))=\ldots=L(\pi'(j_{x+1}-1))\,\pi(i_x,e)=\pi(i_x+1,E)=\ldots$$
$$=\pi(i_{x+1}-1,E)=\pi'(j_x,E)=\pi'(j_x+1,E)=\ldots=\pi'(j_{x+1}-1,E)$$
$$L(\pi(i_l))=L(\pi(i_l+1))=\ldots=L(\pi(m))=L(\pi'(j_l))=L(\pi'(j_l+1))=\ldots=L(\pi'(n))$$

and $\pi(i_l,E)=\pi(i_l+1,E)=\ldots=\pi(m,E)=\pi'(j_l,E)=\pi'(j_l+1,E)=\ldots=\pi'(n,E)$.

Essentially bounded stuttering equivalent divides some bounded path into many finite sequences of identically labeled states and events. This optimization is based on the idea that if different execution sequences of asynchronous processes cannot be distinguished by the property we want to check, we only to consider one representative sequence. Using this approach, the state space can be greatly reduced because we no longer need to consider all possible execution sequences. The following theorem is very important for improving the efficiency of SAT-based bounded model checking of SE-LTL. It shows that any *SE-LTL*$_{-X}$ (resulting by removing X operator from SE-LTL) property is invariant under bounded stuttering equivalent.

Theorem 7. If two paths π,π' are (m,n) bounded stuttering equivalent then for any *SE-LTL*$_{-X}$ formula ϕ, $\pi|=_m\phi$ if and only if $\pi'|=_n\phi$.

Now we redefine the unfolding of the transition relation such that bounded stuttering equivalent can be integrated in our BMC procedure.

Definition 12. Given an LKS M, and a bound k with $k\ge 0$, we define

$$[M]_k^{Mod}:=I(s_0)\wedge\bigwedge_{i=0}^{k-1}(R(s_i,a_i,s_{i+1})\wedge(s_i\ne s_{i+1}\vee a_i\ne a_{i+1}))\,.$$

Informally in our new encoding, we require that two consecutive states and events can not be same. The following theorem guarantees the correctness of our new encoding.

Theorem 8. For some integer k with $k\ge 0$, and an *SE-LTL*$_{-X}$ property ϕ, if $[M]_k^{Mod}\wedge[\phi]_k$ is satisfiable then $M|=E\phi$.

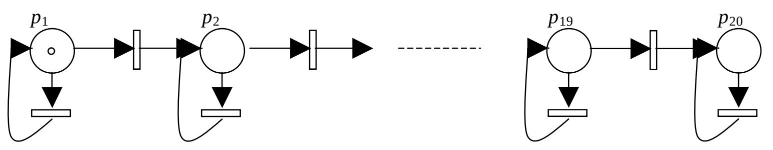
Since $[M]_k^{Mod}\to[M]_k$, the proof of Theorem 8 is easy. In the original bounded model checking equivalent bounded paths may be searched repeatedly. Therefore some searches are redundant. With respect to *SE-LTL*$_{-X}$ properties in our new encoding bounded model checking can avoid searching redundant bounded paths. However, $[M]_k^{Mod}$ is larger than $[M]_k$. Therefore sometimes the new encoding may be inefficient.

Given an SE-LTL formula, we use the notation $\Sigma(\phi)$ to represent the set of events appearing in ϕ. Without loss of generality we assume that the transition relation R for an asynchronous system is expressed as a disjunction of transitions $R_1,\ldots,R_n$, and for each $1\le i\le n$, R_i has only one true assignment. For the specification ϕ, for each $1\le i\le n$, we look for the transition R_i which represents a transition of a state to its self and the event making R_i does not belong to $\Sigma(\phi)$. Without loss of generality assume that $\{R_{j+1},\ldots,R_n\}$ are these transitions. Then let $R':=R_1\vee\ldots\vee R_j$.

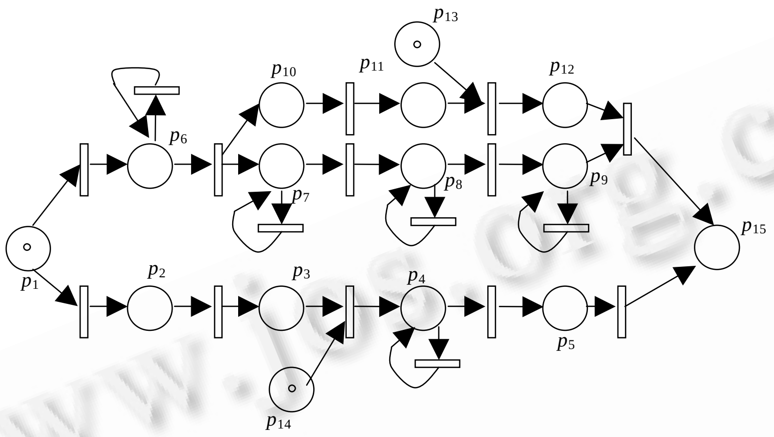
Definition 13. Given an LKS M, and a bound k with $k\ge 0$, we define $[M]_k^{Mod_1}:=I(s_0)\wedge\bigwedge_{i=0}^{k-1}R'(s_i,a_i,s_{i+1})$.

Theorem 9. For some integer k with $k\ge 0$, and an *SE-LTL*$_{-X}$ property ϕ, if $[M]_k^{Mod_1}\wedge[\phi]_k$ is satisfiable then $M|=E\phi$.

Since $[M]_k^{Mod_1}$ is smaller than $[M]_k$ and can avoid searching for bounded stuttering equivalent paths, the new encoding is efficient. We use the Petri nets in Fig.2 as our benchmarks to compare the BMC procedure and the BMC procedure integrating stuttering equivalence. Table 2 is the experiment which shows that introducing stuttering equivalence in our BMC procedure can reduce verification time very much.



Petri net PN_1



Petri net PN_2

Fig.2 Two simple Petri nets

6 Conclusion and Future Work

In this paper, we presented an SAT-based Bounded Model Checking procedure for the state/event linear time temporal logic SE-LTL. We also presented a new method integrating our BMC procedure and stuttering equivalent.

Table 2 Comparing the BMC procedure and the BMC procedure integrating stuttering equivalent

Example	Property	Stuttering equivalent	Bound	Error	Time (s)
PN_1	$G\neg p_{20}$	No	19	Yes	15.594
PN_1	$G\neg p_{20}$	Yes	19	Yes	3.141
PN_1	$G\neg(\neg p_{20}Up_{14})$	No	13	Yes	5.813
PN_1	$G\neg(\neg p_{20}Up_{14})$	Yes	13	Yes	2.156
PN_1	$G(p_{15}\rightarrow FGp_{16})$	No	15	Yes	8.812
PN_1	$G(p_{15}\rightarrow FGp_{16})$	Yes	15	Yes	2.953
PN_2	$G\neg p_{15}$	No	5	Yes	27.136
PN_2	$G\neg p_{15}$	Yes	5	Yes	19.243
PN_2	$G\neg(\neg p_5Up_4)$	No	3	Yes	16.854
PN_2	$G\neg(\neg p_5Up_4)$	Yes	3	Yes	11.346
PN_2	$G\neg(\neg p_{12}Up_{11})$	No	3	Yes	15.278
PN_2	$G\neg(\neg p_{12}Up_{11})$	Yes	3	Yes	10.972

The integration reduces the run time of SAT solvers. Our method is also fit for SAT-based BMC of LTL. We further provided a new verification strategy for concurrent software system which integrates SAT-based BMC, counterexample-guided abstraction refinement and compositional reasoning. We are developing the tool for implementing the new strategy. For future work we would like to discuss other equivalent relations.

References:

[1] Chaki S, Clarke EM, Quaknine J, Sharygina N, Sinha N. Concurrent software verification with states, events, and deadlock. Formal Aspects of Computing, 2004,17(4):461−483.

[2] Biere A, Cimatti A, Clarke EM, Zhu Y. Symbolic model checking without BDDs. Lecture Notes in Computer Science, 1999,1579: 193−207.

[3] Clarke EM, Biere A, Raimi R, Zhu Y. Bounded model checking using satisfiability solving. Formal Methods in System Design, 2001,19(1):7−34.

[4] Wehrheim H. Preserving properties under change. Lecture Notes in Computer Science, 2004,3188:330−343.

[5] Clarke EM, Grumberg O, Jha S, Lu Y, Veith H. Counterexample-Guided abstraction refinement. Lecture Notes in Computer Science, 2000,1855:154−169.

[6] Clarke EM, Gupta A, Strichman O. SAT-Based counterexample-guided abstraction refinement. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2004,23(7):1113−1123.

[7] Clarke EM, Grumberg O, Jha S, Lu Y, Veith H. Counterexample-Guided abstraction refinement for symbolic model checking. Journal of the ACM, 2003,50(5):752−794.

[8] Cobleigh JM, Giannakopoulou D, Pasareanu CS. Learning assumptions for compositional verification. Lecture Notes in Computer Science, 2003,2619:331−346.

[9] Clarke EM, Kroening D, Ouaknine J, Strichman O. Completeness and complexity of bounded model checking. Lecture Notes in Computer Science, 2003,2937:85−96.

[10] Murata T. Petri nets: Properties, analysis and application. Proc. of the IEEE, 1989,77(4):541−574.

[11] Melzer S, Romer S. Deadlock checking using net unfoldings. Lecture Notes in Computer Science, 1997,1254:352−363.

ZHOU Cong-Hua was born in 1978. He is a lecture at School of Computer Science and Telecommunication Engineering, Jiangsu University. His current research areas are model checking and information flow security, etc.