PPTL模型检测器实现的一个关键技术

杨琛[1,2,3]，段振华[1,2]
(1.西安电子科技大学计算理论与技术研究所，710071，西安；2.西安电子科技大学ISN国家重点实验室，710071，西安；3.武汉大学软件工程国家重点实验室，430072，武汉)

摘要：针对命题线性时序逻辑表达能力有限的问题，设计并开发了基于SPIN(Simple Promela interpreter)验证系统的命题投影时序逻辑(PPTL)模型检测器.将协议元语言(ProMeLa)描述的系统转换为系统自动机，将PPTL公式表达的性质转换为性质自动机，通过判定系统与性质自动机的积自动机接受的语言是否为空来判断系统是否满足性质.PPTL模型检测器修改了SPIN的匹配机制，从而改进了验证算法，使得PPTL模型检测器支持有穷和无穷模型的验证.实验结果表明，该模型检测器可以减少无效验证产生的无效迹数目，有效地实现PPTL模型检测.

关键词：时序逻辑；自动机；模型检测
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2010)10-0024-06

A Key Technique to Develop the Model Checker for Propositional Projection Temporal Logic

YANG Chen[1,2,3], DUAN Zhenhua[1,2]
(1. Institute of Computing Theory and Technology, Xidian University, Xi′an 710071, China; 2. The State Key Lab of ISN, Xidian University, Xi′an 710071, China; 3. State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)

Abstract: The propositional projection temporal logic (PPTL) has more expressive power than other linear temporal logics, for example, the propositional linear-time temporal logic (PLTL), and thus is more suitable for use as a specification language in model checking. Hence, a key technique for PPTL model checker is presented. The model checker interprets a ProMeLa model S as a Büchi automaton AS, and transforms PPTL property P to an automaton A ¬ P. In order to determine whether S satisfies P or not, the product of automata AS and A ¬ P is computed and it is checked whether the product automaton accepts the empty word or not. The checking algorithm is implemented based on SPIN. However, since PPTL contains both finite and infinite models, SPIN cannot be used off-the-shelf. To cope with the problem, the related algorithm in SPIN is modified to support PPTL. Experimental results show that the PPTL model checker can effectively perform model checking against PPTL properties.

Keywords: temporal logic; automaton; model checking

模型检测是一种形式化的验证方法，能够验证系统的活性、安全性等性质，已广泛用于安全协议[1]、web服务[2-3]以及多核系统[4]的验证.模型检测中性质的描述基于命题投影时序逻辑，而业界普

收稿日期：2010-01-20. 作者简介：杨琛(1981—)，男，博士生；段振华(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金重大国际合作项目(60910004)；国家自然科学基金资助项目(60433010,60873018)；国家重点基础研究发展规划资助项目(2010CB328102)；武汉大学软件工程国家重点实验室开放基金资助项目(SKLSE20080713)；中央高校基本科研业务费专项资金资助项目(JY10000903004).

遍使用的命题线性时序逻辑(PLTL)[5-6]表达能力有限,不能表达所有的正则性质,使得一些性质,如"命题 p 必须在每个偶数状态成立"的验证无法完成[7]. 与 PLTL 相比较,命题投影时序逻辑(PPTL)[7-9]可以表达所有正则性质,用它作为性质描述语言可以描述更多的性质,从而确保系统的需求.

PPTL 模型检测[10]借助 SPIN 验证系统[5]实现. PPTL 模型检测器中输入的系统由协议元语言 ProMeLa[5]描述,性质用 PPTL 公式 P 表达. 因为 PPTL 同时支持有穷和无穷模型验证,所以需要对 SPIN 的验证机制进行修改,使检测器能够判定系统是否满足性质.

1　投影时序逻辑

语法　设 N 为命题集合,命题投影时序逻辑(PPTL)的语法定义为

$$P::=p \mid \neg P \mid P_1 \wedge P_2 \mid \bigcirc P \mid P_1;\ P_2 \mid (P_1,\cdots,\ P_{m-1})\ \text{prj}\ P_m$$

式中:$p\in N$,$P_i(i=1,\cdots,m)$和 P 都是 PPTL 公式;非"$\neg$"和与"$\wedge$"为布尔算子;"$\bigcirc$"(next)、";"(chop)和"prj"(projection)为时序算子. 仅含布尔算子的公式称为状态公式,否则称为时序公式.

语义　状态 $s:N\to\{\text{true},\text{false}\}$ 定义为一个函数,它为每个包含在 AP 中的命题赋真假值. 用 $s(p)$ 表示 s 赋予 p 的值. 区间 $\sigma=<s_0,\cdots,s_{|\sigma|}>$是一个状态序列,其中 $|\sigma|<\infty$表示区间长度,它等于状态数减 1. 此外,$\sigma_1\cdot\sigma_2$ 表示由区间 σ_1 的末状态和 σ_2 的首状态连接而成的新区间. PPTL 的解释是一个三元组 $I=(\sigma,k,j)$,其中 σ 是区间,$k\leqslant j\leqslant|\sigma|$ 为整数,解释 $I=(\sigma,k,j)$满足 P,记作 $I\vDash P$.

为了定义投影操作的语义,需要辅助算子 $\downarrow$. 设 $\sigma=\langle s_0,s_1,\cdots\rangle$是一个区间,$r_0,\cdots,r_h(h\geqslant 0)$是整数且有 $0\leqslant r_0\leqslant\cdots\leqslant r_h\leqslant|\sigma|$,则 σ 在 $r_0,\cdots,r_h$ 上的投影 $\sigma\downarrow\langle r_0,\cdots,r\rangle=\langle s_{t_0},\cdots,\ s_{t_l}\rangle$,其中 $t_0,\cdots,\ t_l$ 是通过删除 $r_0,\cdots,r_h$ 中重复元素所得. 例如,$\langle s_0,s_1,s_2,\rangle\downarrow\langle 0,\ 0,\ 2,\ 2\rangle=\langle s_0,s_2\rangle$.

PPTL 的可满足关系 $\vDash$ 定义为 $I\vDash p$ 当且仅当 $s_k(p)=\text{true}$;$I\vDash\neg P$ 当且仅当 $I|\neq P$;$I\vDash P_1\wedge P_2$ 当且仅当 $I\vDash P_1$ 并且 $I\vDash P_2$;$I\vDash\bigcirc P$ 当且仅当$(\sigma,k+1,j)\vDash P$;$I\vDash P_1;\ P_2$ 当且仅当存在 $r:k\leqslant r\leqslant j\leqslant|\sigma|$ 使得$(\sigma,k,r)\vDash P_1$ 并且$(\sigma,r,j)\vDash P_2$;$I\vDash(P_1,\cdots,\ P_{m-1})\ \text{prj}\ P_m$,若存在递增整数序列 $k=r_0\leqslant\cdots\leqslant r_m\leqslant j$,使$(\sigma,r_{l-1},r_l)\vDash P_1(1\leqslant l<m)$并且满足$(\sigma',0,|\sigma'|)\vDash P_m$,其中 σ'为

$r_m<j$ 并且$\sigma'=\sigma\downarrow(r_0,\cdots,\ r_m)\cdot\sigma_{(r_m+1,\cdots,j)}$

或

$r_m=j$ 并且$\sigma'=\sigma\downarrow(r_0,\cdots,\ r_h)(0\leqslant h\leqslant m)$.

PPTL 模型检测中,公式 P 用来表达系统的性质,也就是程序的行为. 从这个角度更容易理解 chop 和 projection 算子. 如图 1 所示,P_1;P_2 表示将 P_1 的区间和 P_2 的区间顺序连接. 直观地讲,chop 算子可支持多个子程序的顺序复合. $(P_1,\cdots,\ P_{m-1})\text{prj}P_m$ 从两个时间粒度来刻画程序:①在细粒度(精化)上程序由子程序 $P_1,\cdots,\ P_{m-1}$ 顺序复合而成,即 $P_1;\cdots;\ P_{m-1}$;②在粗粒度(抽象)上程序的行为可由 P_m 表达,其区间由各个子程序 $P_1,\cdots,\ P_{m-1}$ 的端点从细粒度上投影下来构成.

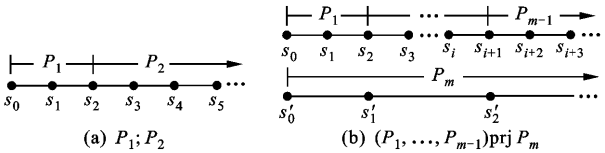

图 1　chop 和 projection 算子的语义

PPTL 的常用派生公式为 $\varepsilon=\neg\bigcirc\text{true}$,$\Diamond P=\text{true};P$,$\Box P=\neg\Diamond\neg\ P$. 其中 ε 表示当前状态是区间的末状态;$\Diamond P$($\Diamond$读作 eventually)表示 P 在将来某个状态可满足;$\Box P$($\Box$读作 always)表示 P 在每个状态都要被满足. 此外,对于布尔运算有 $P_1\to P_2=\neg\ P_1\vee P_2$

第二作者在前期工作[7-8]中提出 PPTL 向 Büchi 自动机的转换方法. 给定一个 PPTL 公式 P,可得到其自动机 $A_P=(V,N_P,v_1,T,F)$,其中 N_P 是公式 P 中包含的命题集合;V 是状态集合,状态 $v\in V$ 可解释为函数 $v:N_P\to\{\text{true},\text{false}\}$;$v_1$ 是起始状态;$(v,v')\in T$ 是迁移集合;F 为可接受状态集合. A_P 的无穷路径 $v_1v_2\cdots$可被接受当且仅当存在状态 $v\in F$ 使得该无穷路径无限次经过状态 v. 文献[8]证明了 A_P 接受的语言可刻画公式 P 的所有可满足区间. 基于该转换方法,开发了相应的 PPTL 解释/转换器负责将 PPTL 公式向 Büchi 自动机转换,并以 never claim 格式输出.

2　基于 SPIN 的 PPTL 模型检测

如图 2 所示,PPTL 模型检测借助通用模型检测器 SPIN 实现,其中待检测的系统由协议元语言 ProMeLa 描述,性质用 PPTL 公式 P 表达. 检测器利用 SPIN 来验证系统是否满足性质. SPIN 的验证

基于自动机理论，系统 S 转为 Büchi 自动机 A_S，其中状态 s 被解释为函数 $s:N_P\rightarrow\{\text{true},\text{false}\}$；将性质 P 的非转为 Büchi 自动机 $A_{\neg P}$（该自动机以 never claim[5] 结构描述），之后对 A_S 和 $A_{\neg P}$ 求交得积自动机 $A_{S\times\neg P}$，若该积自动机为空，说明系统满足性质；否则 A_S 中存在可接受路径 $\vec{s}=s_1\cdots s_k$ 与 $A_{\neg P}$ 中的可接受路径 $\vec{v}=v_1\cdots v_k$ $(k\leqslant\omega)$ 相匹配，即相应状态逐个匹配 $\forall(1\leqslant i\leqslant k)\ [s_i(p)=v_i(p)]$. 此时 $\vec{s}$ 就是系统中违反性质的迹. 然而，SPIN 原有的验证系统并不完全适用于 PPTL 模型检测.

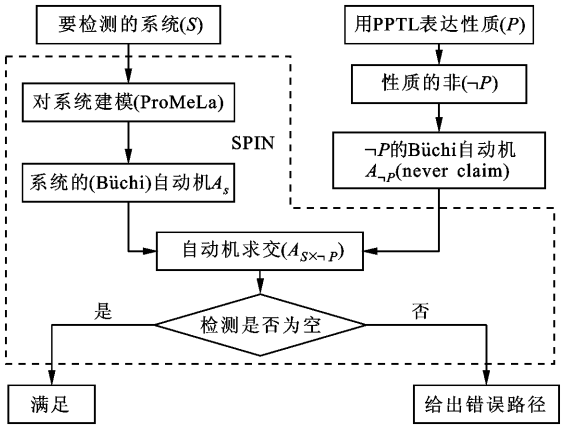

图 2 基于 SPIN 的 PPTL 模型检测器框架

2.1 SPIN 不支持有穷模型验证的问题

系统由计数器和服务器两个实体并发构成. 服务器不断地向计数器发送计数指令，计数器收到指令后先判断变量 count（初始值为 1）是否小于等于 5，如果是再将 count 加 1 并进入下一次循环等待计数指令，否则退出循环. SPIN 将系统模型解释为自动机 A_S，如图 3a 所示，其中 s_4 为可接受状态. 要检测的性质为：存在某一时刻使得 count＝4. 该性质可用 PPTL 公式 $P=\text{true};p$ 表达，式中命题 p 定义为 count＝4. 该性质的非为 $\neg\ P=\Box\neg\ p$，其自动机 $A_{\neg P}$ 如图 3b 所示，其中 v_1 和 v_2 是可接受状态. 注意 s_4 原本没有出度，因为它在 A_S 中仅表示有穷路径的可接受状态（简称有穷接受状态）. 为了与 Büchi 自动机的可接受状态统一，SPIN 会给它们加入 τ 自环，该技术被称为 τ 扩展[5]. 同理，v_2 在 $A_{\neg P}$ 中表示有穷接受状态，因此 v_2 有 τ 自环.

图 3 中 A_S 和 $A_{\neg P}$ 接受的语言没有交集，即系统满足性质. 然而，SPIN 给出了否定结果，并反馈回一条系统不满足性质的迹，如图 3c 所示. 该迹的显示格式类似于 UML 中的顺序图，包括进程 counter 和 server，矩形表示系统（运行）状态，其中的数字代表状态编号，状态编号自上而下递增，表示系统演化过程，图中的箭头表示进程的交互，并且只显示发生交互的状态. 相应的验证结果[10] 显示在状态 7、17 中执行 count 加 1 操作，因此 count 的值在该迹中始终不为 4，从而该迹在 A_S 中对应的路径为 $\vec{s}=s_1s_2s_3s_2s_3$. 验证结果还显示 $A_{\neg P}$ 中的可接受路径 $\vec{v}=v_1v_1v_1v_1v_2$ 与 $\vec{s}$ 匹配. 注意 $\vec{s}$ 不可被 A_S 接受，因为其未到达可接受状态 s_4. 因此，SPIN 对该实例的验证是无效的.

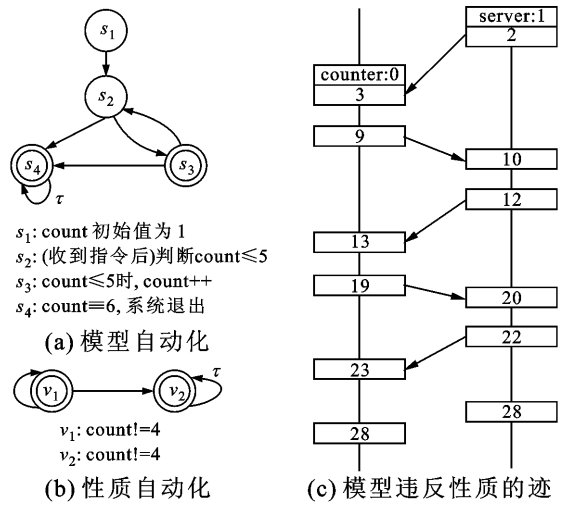

图 3 计数器模型、待验证的性质以及 SPIN 的验证结果

2.2 SPIN 中 $A_{\neg P}$ 的迁移机制的修改

如图 4 所示，上述问题源于 $A_{\neg P}$ 中的非确定性，即处于状态 v_1 时，其后继状态的选择是非确定的：可沿自循环到达 v_1 或者迁移到 v_2. 状态 v_2 是有穷接受状态，因此应与系统有穷接受状态 s_4 匹配，然而 A_S 从 s_2 迁移到 s_3 时（注意 s_3 不是接受状态），SPIN 错误地为 $A_{\neg P}$ 选择 v_2 来匹配 s_3，由此验证失效.

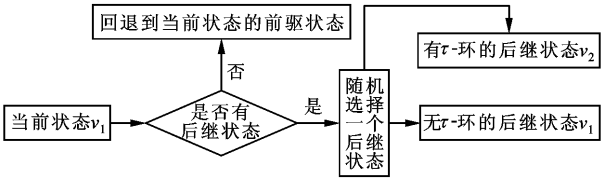

图 4 SPIN 对 $A_{\neg P}$ 后继状态的随机选择

性质自动机 $A_{\neg P}$ 中的非确定性原本对 SPIN 的验证不构成影响. 因为 SPIN 是为命题线性时序逻辑（PLTL）开发的模型检测系统，该逻辑只具有无穷模型[6]，因此从 PLTL 转换得到的 $A_{\neg P}$ 中不包含有穷路径，而 PPTL 解释在无穷区间和有穷区间上，具有无穷模型和有穷模型，所以 PPTL 模型检测器在求 A_S 和 $A_{\neg P}$ 接受语言的交集时，$A_{\neg P}$ 中的有穷接受状态（即带 τ 自环的状态）需要与 A_S 的有

穷接受状态(带 τ-自环的状态)相匹配. 于是,有如下定义.

定义 1　给定一个系统 S 和其性质 P,

(1)A_S 的无穷路径 $s=v_1v_2\cdots$ 匹配 $A_{\neg P}$ 的无穷路径 $s=s_1s_2\cdots$ 当且仅当

$\forall(1\leqslant i\leqslant\omega)\forall(p\in N_P)[v_i(p)=s_i(p)$ 且 v_i 和 s_i 无 τ 自环];

(2)A_S 的有穷路径 $v'=v_1\cdots v_k$ 匹配 $A_{\neg P}$ 的有穷路径 $s'=s_1\cdots s_k$ 当且仅当

$\forall(1\leqslant i\leqslant k)\forall(p\in N_P)[v_i(p)=s_i(p)]$,且只有 v_k 和 s_k 有 τ 自环.

为了使 SPIN 支持 PPTL 模型检测,需要根据定义 1 来修改它的匹配机制,为此在 SPIN 的验证模块(pangen2.c)中增加一个函数 sys_end()来判断系统运行是否结束,即系统是否到达带 τ-自环的状态. sys_end()的实现思想为:先识别出系统中每个包含有穷路径的进程,当所有这些进程到达结束状态时,函数返回 1;否则返回 0. 因此,若 sys_end()返回值为 1,则表示系统已经结束. 修改的匹配机制如图 5 所示,只有当 sys_end()返回值为 1 时,才能选择 $A_{\neg P}$ 中带 τ-自环的后继状态来匹配.

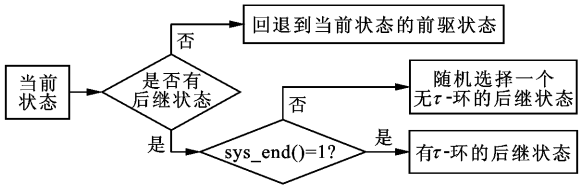

图 5　修改的匹配机制中 $A_{\neg P}$ 对后继状态的选择

基于修改的匹配机制,PPTL 模型检测器重新验证计数器实例的结果如图 6 所示. 图中显示系统满足性质,修改后的 SPIN 可支持系统和性质自动机各自的有穷路径的匹配,从而支持有穷模型验证.

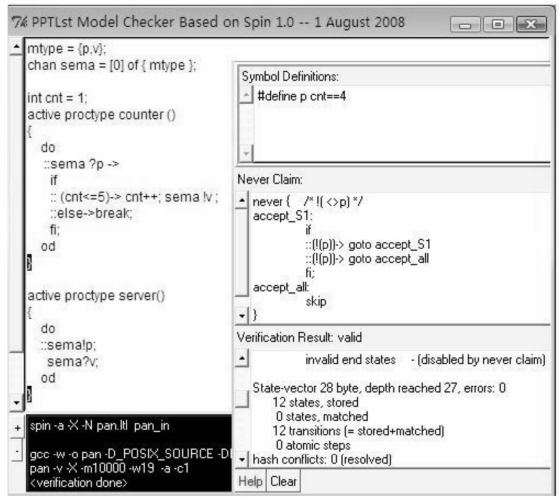

图 6　PPTL 模型检测器验证计数器实例

3　实验结果与分析

首先通过验证 7 个典型实例来分析 SPIN 的匹配机制在修改前后对无效迹的影响,见表 1;然后选取若干基准(benchmark)测试实例,分别用 SPIN 和 PPTL 模型检测器对其进行验证以比较二者的验证效率,见表 2.

表 1　模型检测器对典型实例的验证结果

编号	实例	性质	原匹配机制下所得无效迹个数	修改后匹配机制下所得无效迹个数	有效率 α/%
1	哲学家就餐(5 人)	$\Box\neg q\vee\Diamond q;p$	>100	1	<1.0
2	哲学家就餐(10 人)	$\Box\neg q\vee\Diamond q;p$	7	1	14.3
3	哲学家就餐(20 人)	$\Box\neg q\vee\Diamond q;p$	7	1	14.3
4	尼达姆协议	true;(p prj r)	99	16	16.2
5	青蛙寻路(4 只)	$\Box\neg(p;q)$	30	24	80.0
6	青蛙寻路(6 只)	$\Box\neg(p;q)$	184	160	87.0
7	青蛙寻路(8 只)	$\Box\neg(p;q)$	992	896	93.0

表 2　模型检测器对基准测试实例的验证结果

实例	性质	是否满足性质	SPIN		PPTL 模型检测器	
			状态个数	迁移个数	状态个数	迁移个数
brp	$\Box(p\rightarrow r)$	是	22 881	49 978	22 881	49 978
iprotocol	$((\Box\Diamond q)\wedge(\Box\Diamond r))\rightarrow(\Box\Diamond p)$	否	46	46	55	62
protocols	$\Box(p\vee q)$	是	2 249	7 678	2 249	3 057
尼达姆协议	$\Box((p\wedge r)\rightarrow q)$	否	46 915	74 313	46 916	74 315
leader election	$\Diamond\Box p$	是	181	611	181	349
elevator	$\Box(p\rightarrow\Diamond q)$	是	135 973	459 676	135 973	459 676
train gate	$\Box(p\rightarrow\Diamond q)$	否	35	36	36	40

表1中实例1、2和3针对死锁性质验证，其系统描述如下. 有 $n(n=5,10,20)$ 位哲学家分别坐在圆桌周围的 n 张椅子上，桌上有 n 个碗和 n 根筷子. 哲学家们做的事情为思考或者就餐. 思考时停止就餐，就餐时停止思考. 哲学家能够就餐的前提条件是取到其左右靠近他的筷子. 若每个哲学家都拿着左手的筷子，永远都在等右边的筷子，或者相反，他们都无法就餐，从而产生死锁. 令命题 p 为真表示至少一个哲学家就餐，命题 q 为真表示桌上没筷子. 于是性质 $\square\neg q\vee\Diamond(q;p)$ 表示有筷子在桌面上(此时不存在死锁)，或者没有筷子在桌子上，但至少有一个哲学家可以就餐.

表1中实例4针对活性验证，其问题为基于共享密钥体系的协议(Needham协议)，其系统描述如下. ① $A\rightarrow B:\{A,N_A\}_B$ 表示 A 将身份信息和随机数 N_A 用 B 的公钥加密并发送给 B；②$B\rightarrow A:\{N_A,N_B\}_A$ 表示 B 将收到信息用自己的密钥解密，得到 A 的身份信息，知道发送者是 A，并将随机数 N_A 与自己产生的随机数 N_B 用 A 的公钥加密发送给 A；③ $A\rightarrow B:\{N_B\}_B$ 表示 A 将收到的信息用 A 自己的密钥解密，确认 N_A 为自己刚发送的随机数即可确认自己通信成功，然后将收到的 B 的随机数 N_B 用 B 的公钥加密发送给 B，当 B 收到信息并解密确认 N_B 后则可确认自己通信成功. 当双方均确认通信成功，则通信结束. 协议的目的是在不安全信道上建立可靠的双方通信密钥 N_B，而入侵者试图探知该密钥. 令命题 p 为真表示消息③被 B 接收，r 为真表示入侵者不知道密钥 N_B，则性质 true；(p prj r) 表示消息③总会发生(即通信结束)，并且在通信结束时入侵者也不知道密钥 N_B.

表1中实例5、6和7针对安全性验证，其系统描述为：有 $n(n=4,6,8)$ 个青蛙排在如下的 $n+1$ 个位置上

$$\underbrace{A\cdots A}_{n/2}\_\underbrace{B\cdots B}_{n/2}$$

标志为 A 的青蛙只能向右跳，标志为 B 的青蛙只能向左跳，每一时刻只能有一只青蛙在跳，每个青蛙每次只能跳一格或两格，并且不能跳到其他青蛙的上面，最后标志为 A 的青蛙和标志为 B 的青蛙能否互换位置得到排列

$$\underbrace{B\cdots B}_{n/2}\_\underbrace{A\cdots A}_{n/2}$$

令命题 p 为真表示所有青蛙都不能再向前跳，命题 q 为真表示互换位置成功，则性质 $\square\neg(p;q)$ 表示当所有青蛙都不能向前跳跃时，青蛙的位置也无法成功互换.

显然7个实例均不满足其各自的性质. 实例4的系统只包含有穷路径，其余实例同时具备有穷和无穷路径. 表中第4列表示用原匹配机制验证得到的迹个数；第5列表示基于修改的机制验证得出的迹数[10].

实验采用5.2.4版本SPIN(2009年11月)进行原匹配机制下的验证，而采用改进后的SPIN在修改的机制下进行验证. 假设原匹配机制下检测所得的迹的集合为 A，修改的机制下检测出的迹集合为 B，表中数据显示 $|A|>|B|$. 此外，对这些迹逐个分析可知：①$B\subset A$；②A-B 中都为无效迹；③这些无效迹都是因 $A_{\neg P}$ 中有穷接受状态匹配了 A_S 中非有穷接受状态而造成. 设计人员若得到的是 A，他需要再判断哪些迹有效，从而增加了工作负担，因此减少无效迹的数目可以提高软件设计效率. 表中最后一列表示有效迹占所有迹的比率(有效率)，其计算公式如下

$$\alpha=\frac{|B|}{|A|}$$

表1中实例5、6和7的有效率较高，原因有：①A_S 仅包含少数有穷路径，从而造成因有穷路径 $A_{\neg P}$ 与 A_S 匹配错误的概率较低；②验证的性质是安全性，即可用 $P=\square P'$ 类型公式表达. 由 $\neg P=\Diamond\neg P'$ 以及◇的语义可知，$A_{\neg P}$ 中的路径只要包含满足 $\neg P'$ 的状态(记为 v)就可被接受，且从 v 之后的首个接受状态起每个状态都是接受状态. 因此在匹配 A_S 中的路径时，$A_{\neg P}$ 只要经过了 v 后的首个接受状态，就可在任何时刻停止并完成匹配，在这种情况下所得的迹多数是有效的. 相比之下表1中实例1、2、3和4的效率低，这是因为其系统包含了大量的有穷路径(实例1～3中到达死锁状态的路径，实例4本身是可执行结束的)，而性质属于活性(实例1-3的性质中的析取项 $\Diamond q;p$，实例4的性质等价于 $\Diamond(p$ prj $r)$). 活性一般可用 $P=\Diamond P'$ 类型公式表达. 由 $\neg P=\square\neg P'$ 以及□的语义可知，$A_{\neg P}$ 中有穷可接受路径 $\vec{v}$ 的每个状态都满足 $\neg P'$，其中只有末状态是可接受状态. 此时若 A_S 的有穷路径 $\vec{s}$ 中存在满足 $\neg P'$ 的状态 s，然而 s 不是有穷接受状态，会造成 s 错误地匹配 $\vec{v}$ 的有穷接受状态使得匹配失效. 图3中的计数器验证就属于这种情况. 由此可知，有效率取决于系统和性质二者的结构.

此外，表1中对7个实例的验证表明，PPTL模型检测器可实现对安全性(死锁属于安全性)和活性

性质的验证. 由于任何性质都由安全性和活性组合而成[11]，所以 PPTL 模型检测器可对 PPTL 能够表达的所有性质进行验证.

PPTL 因为包含了 chop 和 projection 算子，它不仅将 PLTL 扩充至有穷模型，而且可以表达更多的性质[8]. 事实上 PLTL 的 until 算子（$\bigcup$），可以用 chop 算子表达，即 $P_1 \bigcup P_2 \equiv (\Box P_1 ; \bigcirc P_2) \vee P_2$，其余算子在 PPTL 中都有相同的算子对应. 因此，对于 PLTL 能够验证的系统，PPTL 也可以验证. 换句话说，SPIN 可验证的系统 PPTL 模型检测器同样可以验证. 表 2 中的所有实例都是用来衡量 SPIN 性能的基准实例[12]，第二列给出了每个实例需要验证的性质，第三列是验证结果（即系统是否满足性质）. 我们分别用 SPIN 和 PPTL 模型检测器来验证这些实例，因此第二列中的性质公式在 SPIN 验证时看作是 PLTL 公式，而在用 PPTL 模型检测器验证时就是 PPTL 公式. 表中列出了两种检测器的验证效率（即遍历的系统状态数和迁移数）[12]. 通过比较可知，PPTL 模型检测器的验证效率接近于 SPIN.

由于 PPTL 模型检测器可验证 PPTL 表达的所有性质，并且验证效率等同于 SPIN，所以 PPTL 模型检测器可有效实现 PPTL 的模型检测.

4　结　论

笔者参与设计并实现了支持 PPTL 的模型检测器. 该检测器是基于 SPIN 的. 为了支持 PPTL 无穷和有穷模型的验证，改造了 SPIN 中相应的验证机制. 实验表明，该检测器可减少无效迹产生的数量，并且能有效支持对 PPTL 性质进行模型检测.

参考文献：

[1] MIAO Huaikou, ZENG Hongwei. Model checking-based verification of web application [C] // ICECCS 2007. Piscataway, NJ, USA: IEEE Computer Society Press, 2007: 47-55.

[2] 王小兵，段振华. 面向投影时序逻辑的 Web 服务模型检测 [J]，西安交通大学学报，2009，43(4)：39-44.
WANG Xiaobing, DUAN Zhenhua. Projection temporal logic oriented model checking for web services [J]. Journal of Xi′an Jiaotong University, 2009, 43(4): 39-44.

[3] MEI Jia, MIAO Huaikou, LIU Pan. Applying SMV for security protocol verification [J]. Information Technology Journal, 2009, 8(7): 1065-1070.

[4] 舒新峰，段振华. 利用投影时序逻辑的多内核进程调度建模与验证 [J]，西安交通大学学报，2010，44(3)：52-57.
SHU Xinfeng; DUAN Zhenhua. Modeling and verification of processes scheduling based on projection temporal logic for multi core CPU [J]. Journal of Xi′an Jiaotong University, 2010, 44(3): 52-57.

[5] HOLZMANN G J. The SPIN model checker: primer and reference manual [M]. Reading, Massachusetts, USA: Addison-Wesley, 2003.

[6] LICHTENSTEIN O, PNUELI A. Checking that finite state concurrent programs satisfy their linear specification [C]// Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages. New York, USA: ACM Press, 1985: 97-107.

[7] DUAN Zhenhua, TIAN Cong, ZHANG Li. A decision procedure for propositional projection temporal logic with infinite models [J]. Acta Informatica, 2008, 45(1): 43-78.

[8] TIAN Cong, DUAN Zhenhua. Propositional projection temporal logic, Büchi automata and omega-regular expressions [C]//Proceedings of TAMC'08. Berlin, Germany: Springer, 2008: 47-58.

[9] DUAN Zhenhua, YANG Xiaoxiao, KOUTNY M. Framed temporal logic programming [J]. Science of Computer Programming, 2008, 70(1): 31-61.

[10] 段振华，杨琛，张笑星，等. 命题投影时序逻辑模型检测器 [EB/OL]. [2010-01-01]. http://www.keepandshare.com/doc/view.php?id=1676421&da=y.

[11] SCHNEIDER F B. Decomposing properties into safety and liveness using predicate logic, TR 87-874 [R]. Ithaca, NY, USA: Cornell University. Computer Science Department, 1987.

[12] 杨琛，张笑星. 衡量 SPIN 性能的基准实例 [EB/OL]. [2010-01-01]. http://www.keepandshare.com/doc/view.php?id=1.

（编辑　武红江）