时序逻辑程序的模型检测

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

摘 要 时序逻辑程序的形式化验证对提高程序的正确性具有重要意义。以投影时序逻辑的可执行子集、框架投影时序逻辑语言 Framed Tempura 为研究对象,使用命题投影时序逻辑描述 Framed Tempura 程序的性质,将程序 p 和性质 ϕ 统一表示在投影时序逻辑中,模型检测需要判定 $p \rightarrow \phi$ 是否有效,可转化为判定 $p \wedge \neg\phi$ 是否不可满足,这可以通过构造 $p \wedge \neg\phi$ 的正则图加以解决。最后,给出了 Framed Tempura 程序的模型检测实例。

关键词 时序逻辑程序,形式化验证,正则图,模型检测
中图法分类号 TP311 **文献标识码** A

Model Checking for Temporal Logic Programs

WANG Xiao-bing[1,2] DUAN Zhen-hua[1]
(Institute of Computing Theory & Technology, Xidian University, Xi'an 710071, China)[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract Formal verification of temporal logic programs plays an important role in improving program correctness. A framing projection temporal logic language, called Framed Tempura, is the research object as an executable subset of projection temporal logic. Propositional temporal logic was used to describe properties of a Framed Tempura program, thus the program p and properties ϕ were both formalized in projection temporal logic, then model checking needs to eva-luate whether or not $p \rightarrow \phi$ is valid which can be translated into evaluate whether or not $p \wedge \neg\phi$ is unsatisfiable, and this problem is solved by constructing the normal form graph of $p \wedge \neg\phi$. At last, an example of model checking a Framed Tempura program was given.

Keywords Temporal logic programs, Formal verification, Normal form graphs, Model checking

框架投影时序逻辑语言 Framed Tempura[1]是投影时序逻辑(Projection Temporal Logic, PTL)的一个子集,它与 XYZ/E[2]同类,也是一种可执行的时序逻辑语言。Framed Tempura 包含了数组、列表、指针[3]等数据结构,实现了顺序语句、条件语句、循环语句,还加入了投影语句、并行语句、等待语句等复杂的程序结构[4],具有更强的表达能力。PTL 适用于描述带有时间特性的软硬件系统性质,Framed Tempura 可用于系统建模,这样便于在统一的 PTL 框架内验证系统性质。目前,PTL 和 Framed Tempura 已经用于混合系统的形式化验证[5]、组合 Web 服务的描述和验证[6]。

程序的形式化验证方法主要有定理证明和模型检测。定理证明的验证方法过程复杂,只能判断系统是否满足性质,不能找出违反性质的原因;而模型检测[7]具有较高的自动化程度,当系统不满足性质时还能够给出反例,便于调试程序,因而是当前研究的热点之一,并已被用于验证通讯协议,安全协议和控制系统等。已经证明,命题投影时序逻辑(Propositional Projection Temporal Logic, PPTL)存在一个可判定算法[8],这为研究框架投影时序逻辑程序的模型检测提供了理论基础。本文介绍了投影时序逻辑和 Framed Tempura 语言,针对 Framed Tempura 程序 p 和 PPTL 描述的性质公式 ϕ,定义了模型检测算法来验证 p 是否满足 ϕ,即 $p \rightarrow \phi$ 是否有效。$p \rightarrow \phi$ 有效等价于 $p \wedge \neg\phi$ 不可满足:若模型检测算法构造出 $p \wedge \neg\phi$ 的正则图只包含根结点,则其不可满足,即 $p \rightarrow \phi$ 始终为真,模型检测成功;否则 $p \wedge \neg\phi$ 可满足,则程序 p 就违反了性质 ϕ,正则图显示的是一个反例。最后本文给出了一个模型检测实例来说明该算法的应用。

1 投影时序逻辑

设 Π 是一个可数的命题集合,V 是一个可数的变量集合,项 e 和公式 p 如下所示,其中包含了逻辑操作符 $=$,$\neg$,$\wedge$,$\vee$ 和 $\exists$,时序操作符下一状态 O,上一状态 $\ominus$ 和投影 prj 等。u 是静态变量,x 是动态变量,π 是命题,f 是函数,P 是谓词。

$$e ::= u \mid x \mid f(e_1, \cdots, e_m) \mid Oe \mid \ominus e$$

到稿日期:2008-11-24 返修日期:2009-02-17 本文受国家自然科学基金重点资助项目(60433010),国家自然科学基金资助项目(60873018),武汉大学软件工程国家重点实验室开放基金项目(SKLSE20080713)资助。

王小兵(1979—),男,博士研究生,讲师,主要研究方向为时序逻辑语言、形式化验证等,E-mail:xbwang@mail.xidian.edu.cn;段振华(1948—),男,教授,博士生导师,主要研究方向为高可信软件理论与技术、互联网计算和互联网软件技术等。

$p::=\pi|e_1=e_2|P(e_1,\cdots,e_m)|\neg p|p_1\wedge p_2|p_1\vee p_2|\exists x:p|Op|\Theta p|(p_1,\cdots,p_m)$ prj q

定义 1 状态 $s=(I_v,I_p)$是一个二元组，$I_v:V\to D$，$I_p:\Pi\to B$。D 是变量的值域，包含 nil 表示未定义值，B 是布尔值集合。

定义 2 区间 $\sigma=\langle s_0,s_1,\cdots\rangle$是一个非空的状态序列。区间长度记为$|\sigma|$，当 σ 无限时，$|\sigma|$为 ω，否则为其状态数减 1。操作·用来将两个区间连接成一个大区间。

定义 3 解释 $I=(\sigma,i,k,j)$是一个四元组，σ 是一个区间，i,k 为整数，j 为整数或ω，并且 $0\leqslant i\leqslant j\leqslant|\sigma|$，其中$\leqslant$定义为$\leqslant-\{\omega,\omega\}$。

对项的解释归纳定义为：

$I[u]=s_k[u]=I_v^k[u]=I_v^i[u]$，若 u 是静态变量。

$I[x]=s_k[x]=I_v^k[x]$，若 x 是动态变量。

$I[f(e_1,\cdots,e_m)]=f(I[e_1],\cdots,I[e_m])$(若 $I[e_h]\neq$nil，对$1\leqslant h\leqslant m$)，否则=nil。

$I[Oe]=(\sigma,i,k+1,j)[e]$(若 $k<j$)，否则=nil。

$I[\Theta e]=(\sigma,i,k-1,j)[e]$(若 $i<k$)，否则=nil。

为了定义投影操作的语义，需要一个辅助操作符↓。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 $0\leqslant r_0\leqslant\cdots\leqslant r_h\leq|\sigma|$，则$\sigma\downarrow\langle r_0,\cdots,r_h\rangle$是 σ 在 $r_0,\cdots,r_h$ 上的投影，为$\langle s_{t_0},\cdots,s_{t_l}\rangle$，其中 $t_0,\cdots,t_l$ 是通过删除 $r_0,\cdots,r_h$ 中的冗余元素得到的，即 $t_0,\cdots,t_l$ 是 $r_0,\cdots,r_h$ 的最长严格递增子序列。例如，$\langle s_0,s_1,s_2,s_3\rangle\downarrow\langle 0,0,2,2,3\rangle=\langle s_0,s_2,s_3\rangle$。

定义 4 若 $I\vdash p$，则称解释 I 满足公式 p，即 p 在 I 的区间σ 的子区间$\sigma_{(i,\cdots,j)}=\langle s_i,\cdots,s_j\rangle$的当前状态 s_k 是满足的。

解释与公式间的满足关系$\vdash$归纳定义为：

$I\vdash\pi$ 当且仅当 $s_k[\pi]=$true。

$I\vdash e_1=e_2$ 当且仅当 $I[e_1]=I[e_2]$。

$I\vdash P(e_1,\cdots,e_m)$当且仅当 P 是除=以外的原始谓词，并且对 $1\leqslant h\leqslant m$，有 $I[e_h]\neq$nil 和 $P(I[e_1],\cdots,I[e_m])=$true。

$I\vdash\neg p$ 当且仅当 $I\nvdash p$。

$I\vdash p_1\wedge p_2$ 当且仅当 $I\vdash p_1$ 并且 $I\vdash p_2$。

$I\vdash p_1\vee p_2$ 当且仅当 $I\vdash p_1$ 或者 $I\vdash p_2$。

$I\vdash\exists x:p$ 当且仅当存在σ'，满足$|\sigma'|=|\sigma|$，$(\sigma',i,k,j)\vdash p$，并且 σ 和σ'除了对 x 的解释可以不同外，对其他变量的解释都相同。

$I\vdash Op$ 当且仅当$k<j$ 并且$(\sigma,i,k+1,j)\vdash p$。

$I\vdash\Theta p$ 当且仅当$i<k$ 并且$(\sigma,i,k-1,j)\vdash p$。

$I\vdash(p_1,\cdots,p_m)$ prj q，若存在整数 $k=r_0\leqslant\cdots\leqslant r_m\leq j$ 使$(\sigma,i,r_0,r_1)\vdash p_1$，$(\sigma,r_{l-1},r_{l-1},r_l)\vdash p_l$(对 $1<l\leqslant m$)并且对下面的某种情况有$(\sigma',0,0,|\sigma'|)\vdash q$：

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

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

投影操作允许用不同的时间粒度刻画计算进程，是 PTL 的核心时序操作符，由它可定义顺序和循环等语句。为了解释$(p_1,\cdots,p_m)$ prj q，需要两个不同时间粒度的状态序列：一个是执行 $p_1,\cdots,p_m$ 的局部序列，另一个是并行执行 q 的全局序列。直观地，q 与 $p_1,\cdots,p_m$ 在一个区间上并行执行，q 的区间状态仅仅是 $p_1,\cdots,p_m$ 各自区间的初始状态和终止状态。投影操作允许 $p_1,\cdots,p_m,q$ 各自独立，都有权利来定义本身的执行区间。进程独立运行，通信依赖于共享变量，且只发生在全局序列的状态上。特别地，进程序列 $p_1,\cdots,p_m$ 与进程 q 可能结束于不同的状态点。(p_1,p_2) prj q 的三种可能执行情况如图 1 所示。

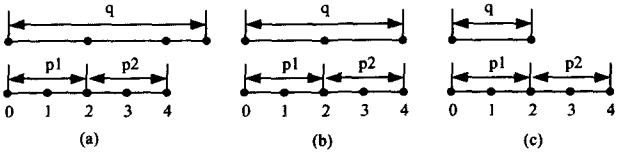

图 1 (p_1,p_2) prj q 的三种可能执行

2 框架投影时序逻辑语言

框架投影时序逻辑语言 Framed Tempura[1] 是 PTL 的一个可执行子集，具有完全的形式化语义。Framed Tempura 程序中一些常用的导出公式定义如下，其中区间时序逻辑[9]的核心操作符 chop(;)能够被 prj 表示。

empty≡¬Otrue　more≡Otrue　skip≡Oempty

len(n)≡O^{n}empty　$\Diamond p\equiv$true;p　$\Box p\equiv\neg\Diamond\neg p$

$p_1;\cdots;p_m\equiv(p_1,\cdots,p_m)$ prj empty

$p^*\equiv$empty$\vee(p;p^*)\vee p\wedge\Box$more

时序逻辑程序在一个状态序列上执行，变量的值不是自动遗传的。为了使变量具有惰性，可以显式或隐式地重复赋值[2]，但是程序会变得比较复杂。定义谓词 $af(x)\equiv p_x$，当变量 x 被赋值时它就为真，否则为假。状态框架操作符 $lbf(x)$和区间框架操作符 $frame(x)$定义如下，它们分别用来声明 x 的值在单个状态和整个区间上是自动遗传的。

定义 5 状态框架操作符 $lbf(x)\equiv\neg af(x)\to\exists b:(\Theta x=b\wedge x=b)$，其中 b为静态变量。$lbf(x)$表示 x 是状态框架变量，即若 x 在当前状态没有赋值，则它继承上一状态的值。

定义 6 区间框架操作符 $frame(x)\equiv\Box(more\to Olbf(x))$，表示 x 是区间框架变量，即 x 在区间的每个状态上都是状态框架变量。$frame(x_1,\cdots,x_n)$用于声明多个区间框架变量。

Framed Tempura 中的表达式对应于 PTL 的项，语句对应于 PTL 的公式，执行程序即是为 PTL 公式寻找一个区间，使公式在此区间的相应状态上能被满足，能成功执行的程序即是可满足的 PTL 公式。在如下基本语句中，x 是变量，e 是表达式，b 是布尔表达式，s(可带下标)是语句。

(1)空语句：empty。

(2)基本赋值语句：$x=e$，$x:=e\equiv$skip$\wedge Ox=e$。

(3)区间框架语句：frame(x)。

(4)存在语句：$\exists x:s$。

(5)next 语句：Os。

(6)always 语句：$\Box s$。

(7)投影语句：$(s_1,\cdots,s_m)$ prj s。

(8)顺序语句：$s_1;s_2\equiv(s_1,s_2)$ prj empty。

(9)选择语句：$s_1\vee s_2$。

(10)合取语句：$s_1\wedge s_2$。

(11)并行语句：$s_1\|s_2\equiv s_1\wedge(s_2;$true$)\vee s_2\wedge(s_1;$true$)$。

(12)条件语句：

if b then s_1 else $s_2\equiv(b\to s_1)\wedge(\neg b\to s_2)$。

(13)while 语句：

while b do $s\equiv(b\wedge s)^*\wedge\Box($empty$\to\neg b)$。

(14)until 语句：repeat s until $b \equiv s$; while $\neg b$ do s。

(15)等待语句：

await(b)$\equiv$frame($x_1,\cdots,x_h$) $\wedge \Box$(empty$\leftrightarrow b$)，其中 $x_1,\cdots,x_h$ 是 b 中的变量。

表达式和程序的解释与 PTL 中项和公式的解释一样。操作符的优先级从高到低依次为$\neg,\exists,\bigcirc,\Theta,\Box,=,:=,\wedge,\vee,||,\rightarrow,\leftrightarrow$,prj,;。empty 表示程序在当前状态终止。执行基本赋值语句首先要解释 e,然后将 x 解释为 e 值,使得公式 $x=e$ 为真。frame(x)表示 x 是区间框架的。存在语句中的变量 x 只在 s 中有效,即是一个局部变量。next 语句表示语句在下一状态执行,而 always 语句表示语句在每个状态都要执行。基于 prj 的投影语句是可执行的,常见的顺序、条件、while 和 until 语句都被定义成一个可执行的时序逻辑公式。选择语句用于构造非确定性程序。合取与并行语句比较类似,都可用于描述并发程序,但后者允许各进程定义自己的区间。例如,len(2)||len(3)可满足,而 len(2) $\wedge$ len(3)不可满足。一个进程可以包含等待语句 await(b),表达式 b 中包含若干变量,当其它进程改变这些变量使得表达式为真时,该进程才跳过等待语句继续执行,因此等待语句可以实现进程间的同步。

3 模型检测算法

Framed Tempura 程序的性质描述可以使用命题投影时序逻辑 PPTL,它是 PTL 的命题形式,排除了变量、谓词和量词等一阶成分,其语法定义如下。

$p ::= \pi \mid \neg p \mid p_1 \wedge p_2 \mid p_1 \vee p_2 \mid Op \mid \Theta p$

$\mid (p_1,\cdots,p_m)$ prj q

PPTL 的语义与 PTL 的类似,每个状态对 Π 中的命题进行解释,但不包括对变量的解释。特别地,PPTL 存在一个可判定的算法[8],它为 Framed Tempura 程序的模型检测提供了理论基础。

Framed Tempura 程序的模型检测思想如下:对程序 p 使用 PPTL 描述待验证的性质为 ϕ,再使用模型检测算法来判定程序是否满足该性质,即 $p \rightarrow \phi$ 是否有效。判定 $p \rightarrow \phi$ 是否有效,可以转化为 $p \wedge \neg\phi$ 的可满足性问题。如果 $p \wedge \neg\phi$ 不可满足,则 $p \rightarrow \phi \equiv \neg(p \wedge \neg\phi)$始终为真,模型检测成功;如果 $p \wedge \neg\phi$ 可满足,则程序 p 就违反了性质 ϕ。

Framed Tempura 程序及其性质都统一表达在 PTL 的框架内,因此公式的形式很复杂,研究它们的性质较为困难。为了简化研究,可将 Framed Tempura 程序和 PPTL 公式转化为等价的正则形,它们的定义分别如下。

定义 7 Framed Tempura 程序 q 的正则形为

$$\bigvee_{i=1}^{k}(q_{ai} \wedge \text{empty}) \vee \bigvee_{j=1}^{h}(q_{cj} \wedge \bigcirc q_{fj})$$

其中,$k,h \geqslant 0 (k+h \geqslant 1)$;对 $1 \leqslant j \leqslant h$,$q_{fj}$ 是一般程序(不要求是正则形);q_{ai} 和 q_{cj} 为 true,或形为 $(x_1 = e_1) \wedge \cdots \wedge (x_m = e_m) \wedge \dot{p}_{x1} \wedge \cdots \wedge \dot{p}_{xm}$,其中 $e_1,\cdots,e_m$ 属于 D,$\dot{p}_x$ 代表 p_x 或 $\neg p_x$。

定义 8 PPTL 公式 q 的正则形为

$$\bigvee_{i=1}^{k}(q_{ai} \wedge \text{empty}) \vee \bigvee_{j=1}^{h}(q_{cj} \wedge \bigcirc q_{fj})$$

其中,$k,h \geqslant 0 (k+h \geqslant 1)$;对 $1 \leqslant j \leqslant h$,$q_{fj}$ 是一般 PPTL 公式(不要求是正则形);q_{ai} 和 q_{cj} 为 true,或形为 $\dot{\pi}_1 \wedge \cdots \wedge \dot{\pi}_m$,$\dot{\pi}$ 代表 π 或 $\neg\pi$。特别地,如果 $\bigvee_{j=1}^{h} q_{cj} \equiv$ true,且 $\bigvee_{i \neq j}(q_{ai} \wedge q_{cj}) \equiv$ false,则该正则形称为完全正则形。

完全正则形用于求解一个 PPTL 公式的否定形式[8]。若公式 q 的完全正则形如上所示,那么 $\neg q$ 为 $\bigvee_{i=1}^{k}(\neg q_{ai} \wedge empty) \vee \bigvee_{j=1}^{h}(q_{cj} \wedge \bigcirc \neg q_{fj})$。根据 Framed Tempura 程序的模型检测思想,性质 p 需要转化为 $\neg p$,因此完全正则形具有重要的意义。

已经证明,存在算法可将任意 Framed Tempura 程序转化为正则形[1],将任意 PPTL 公式转化为完全正则形[8],在此不再赘述。根据正则形可以构造出 Framed Tempura 程序或 PPTL 公式的模型,称为正则图,其定义如下。

定义 9 p 是一个 Framed Tempura 程序或 PPTL 公式,则其正则图是有向图 $G(p)=(CL(p),EL(p))$。$CL(p)$是结点集合,每个结点是一个 Framed Tempura 程序或 PPTL 公式。$EL(p)$是有向边集合,包含三元组(q,p_e,r),表示从结点 q 到结点 r 有一条标记为 p_e 的边,其中 p_e 为 Framed Tempura 程序或 PPTL 公式。正则图 $G(p)$归纳定义如下:

(1)p 在 $CL(p)$中;

(2)对于 $CL(p)-\{\text{empty},\text{false}\}$中任意 q,若其正则形为 $\bigvee_{i=1}^{k}(q_{ai} \wedge \text{empty}) \vee \bigvee_{j=1}^{h}(q_{cj} \wedge \bigcirc q_{fj})$,则 empty 在 $CL(p)$中,对于所有 i 有边(q,q_{ai},empty)在 $EL(p)$中,对于所有 j 有 q_{fj} 在 $CL(p)$中,且有边(q,q_{cj},q_{fj})在 $EL(p)$中。

(3)有限次应用(1)和(2)生成的才是正则图。

例 1 Framed Tempura 程序 p 为 frame(x) $\wedge$ ($x=2 \vee x=3$) $\wedge$ if ($x=2$) then len(2) else len(3),其正则图如图 2 所示。

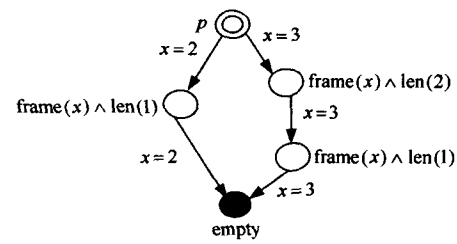

图 2 程序 p 的正则图

从图 2 知,p 是程序正则图的根结点,从它出发到达 empty 终止结点有两条路径,说明程序 p 执行有两种情况。以左边的路径为例进行说明:两条边表明程序有两个状态;边上标注的均是 $x=2$,说明每个状态都要满足该条件,即 x 被置为 2;p 的后继结点是 frame(x) $\wedge$ len(1),表示程序在经过一个状态执行后发生了变化,下一状态要执行的是 frame(x) $\wedge$ len(1),它再经过一个状态执行就变成了 empty,于是程序终止。

在构造 Framed Tempura 程序的正则图时,从根结点代表的公式出发,不断地求解正则形,生成新的边和结点,如果边代表的公式是不可满足的,则该路径无法继续延展到 empty 结点,它就不是程序的可能执行路径,需要将该路径上的结点删除掉。如果一个程序不能执行,即它代表的公式不可满足,不存在从根结点到 empty 结点的路径,那么除根结点外的所有结点都将删除,最终正则图只包含根结点。如果程序能执行,那么在正则图中存在从根结点到 empty 结点的路径。

PPTL 公式的正则图含义与 Framed Tempura 程序的类似,在此不再赘述。

算法 1 构造 $M \wedge \neg P$ 的正则图。

输入:Framed Tempura 程序 M 和 PPTL 公式 P。

输出:正则图 $G=\{CL,EL\}$。

(1)置结点集 CL 为$\{M\wedge\neg P\}$，且该根结点加未处理标记，置边集 EL 为空；

(2)若 CL 中结点处理完毕，转向(4)；否则任取一未处理结点 r $\wedge\neg\varphi$作已处理标记，计算 r 的正则形为 $\bigvee_{i=1}^{k}(r_{ri}\wedge\text{empty})\vee\bigvee_{j=1}^{h}(r_{rj}\wedge Or_{fj})$，计算$\neg\varphi$的正则形为 $\bigvee_{m=1}^{s}(\neg\varphi_{em}\wedge\text{empty})\vee\bigvee_{n=1}^{t}(\varphi_{en}\wedge O\neg\varphi_{fn})$；

(3)上述两个正则形的合取式记为 Q，根据 Q 的三种不同形式执行如下不同的操作，完成后转向(2)。

①若 Q 形如 $\bigvee_{i=1}^{k}\bigvee_{m=1}^{s}(r_{ri}\wedge\neg\varphi_{em}\wedge\text{empty})$，则将结点 empty 加入 CL，对所有 i 和 m，将边$(r\wedge\neg\varphi,r_{ri}\wedge\neg\varphi_{em},\text{empty})$加入 EL；

②若 Q 形如 $\bigvee_{j=1}^{h}\bigvee_{n=1}^{t}(r_{rj}\wedge\varphi_{en}\wedge O(r_{fj}\wedge\neg\varphi_{fn}))$，对所有 j 和 n，若结点 $r_{fj}\wedge\neg\varphi_{fn}$不在 CL 中，则加入该结点并作未处理标记，将边$(r\wedge\neg\varphi,r_{rj}\wedge\varphi_{en},r_{fj}\wedge\neg\varphi_{fn})$加入 EL；

③若 Q 形如 $\bigvee_{i=1}^{k}\bigvee_{m=1}^{s}(r_{ri}\wedge\neg\varphi_{em}\wedge\text{empty})\vee\bigvee_{j=1}^{h}\bigvee_{n=1}^{t}(r_{rj}\wedge\varphi_{en}\wedge O(r_{fj}\wedge\neg\varphi_{fn}))$，即是①和②形式的析取，则执行①和②的操作。

(4)从根结点出发，将无法到达 empty 结点路径上的结点删除。

如果 $M\wedge\neg P$ 的正则图只有一个根结点，没有任何路径存在，则 $M\wedge\neg P$ 不可满足，$M\rightarrow P$ 始终为真，结论为程序 M 满足待验证的性质 P；如果 $M\wedge\neg P$ 的正则图含有路径，则 $M\wedge\neg P$ 可满足，结论为程序 M 不满足待验证的性质 P，分析正则图中的路径，可以得到程序 M 违反性质 P 的一个反例。

4 验证实例

根据算法 1 已实现了一个框架投影时序逻辑程序的模型检测器原型。下面通过一个订票系统的例子来说明模型检测的过程。

例 2 订票系统使用 Ticket 保存总票数，其初始值为 100。售票点 A 和 B 分别用 ATicket 和 BTicket 保存本地票数。售票点 A 卖出了一张票，同时 B 卖出了两张票，最后总票数应该为 97，此结论可以用命题 p 表示。程序满足的性质为：当程序终止时，命题 p 为真，即是 PPTL 公式$\Box(\text{empty}\rightarrow p)$为真。将程序性质和程序源码按照模型检测器的语法规范编码如下。

```
</ //使用 PPTL 描述性质
  define p:Ticket = 97;
  always(empty-> p)
/>
frame(Ticket,ATicket,BTicket) and(
  Ticket=100 and skip;//初始化机票
  (//售票点 A 卖出一张票
  ATicket:= Ticket; ATicket:= ATicket-1;
  Ticket:= ATicket)
  ||
  (//售票点 B 卖出两张票
  BTicket:= Ticket; BTicket:= BTicket-2;
  Ticket:= BTicket)
).
```

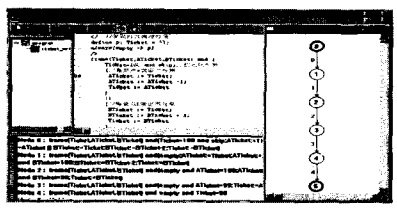

图 3 订票程序的模型检测结果图

模型检测器的运行结果如图 3 所示。正则图含有一条从状态 0 到状态 5 的路径，根据算法 1 得知程序违反了待验证的性质。

根据模型检测器的提示信息对程序进行分析，可知正则图显示的是一个反例：程序执行后总票数 Ticket 为 99。原因在于售票点 A 和 B 的卖票操作是同时进行的，但是程序没有进行并发控制。并发控制需要引入封锁机制，可以通过加入锁变量和等待语句实现，修改后的程序如下。

```
frame(Ticket,Lock,ATicket,BTicket) and(
  //初始化机票和锁
  Ticket = 100 and Lock = 0 and skip;
  (//售票点 A 卖出一张票
  await(Lock=0); Lock:= 1;
  ATicket:= Ticket; ATicket:= ATicket-1;
  Ticket:= ATicket; Lock:= 0)
  ||
  (//售票点 B 卖出两张票
  await(Lock=0); Lock:= 1;
  BTicket:= Ticket; BTicket:= BTicket-2;
  Ticket:= BTicket; Lock:= 0)
).
```

重新运行模型检测器，正则图只有根结点状态 0，因此修改后的程序满足待验证的性质。

结束语 本文针对框架投影时序逻辑语言 Framed Tempura，提出了 Framed Tempura 程序的模型检测算法。使用命题投影时序逻辑描述待验证的性质，将程序和性质统一表示在投影时序逻辑的框架内，模型检测被转化为逻辑公式的可满足性问题，通过构造并分析正则图，可验证程序是否满足待验证的性质。今后的工作包括进一步完善、优化模型检测算法，增强模型检测器的功能，验证更多的 Framed Tempura 程序，扩大 Framed Tempura 语言的应用范围。

参 考 文 献

[1] Duan Zhenhua, Yang Xiaoxiao, Maciej K. Framed temporal logic programming[J]. Science of Computer Programming, 2008, 70(1): 31-61

[2] 唐稚松. 时序逻辑程序设计与软件工程[M]. 北京：科学出版社，1999

[3] 王小兵，段振华. 框架投影时序逻辑程序设计语言中的指针[J]. 西安电子科技大学学报：自然科学版，2008，35(06)：1069-1074

[4] Ma Yongtao, Duan Zhenhua, Wang Xiaobing, et al. An Interpreter for Framed Tempura and Its Application[C]//First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering. Shanghai, 2007

[5] 张海宾，段振华. 混合投影时序逻辑与混合系统的形式化验证[J]. 计算机科学，2007，34(11)：279-282

[6] 王小兵，段振华. 面向对象的时序逻辑语言[J]. 电子科技大学学报：自然科学版，2009，38(01)：97-101

[7] 林惠民，张文辉. 模型检测：理论、方法与应用[J]. 电子学报，2002(S1)：1907-1912

[8] 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

[9] Moszkowski B C. Executing Temporal Logic Programs[M]. Cambridge: Cambridge University Press, 1986