良基归纳法在时序逻辑程序不变式验证中的应用

杨潇潇 段振华
(西安电子科技大学计算理论与技术研究所 西安 710071)

摘 要 并发程序的不变式验证对理解程序和提高程序的正确性具有重要意义。以一种区间时序逻辑程序设计语言Framed Tempura为研究对象,给出了该语言的等价正则形,定义了该正则形在相邻两个状态上的良基关系,进而利用良基归纳法原理对该语言所描述的系统的不变式进行归纳验证。提出的基于良基归纳法的验证方法在时序逻辑程序中可以方便地验证系统的不变式,尤其是循环结构的不变量性质。

关键词 时序逻辑程序,正则形,良基关系,良基归纳法,不变式证明
中图法分类号 TP301 **文献标识码** A

Application of Well-founded Induction in Verifying Invariance of Temporal Logic Programs

YANG Xiao-xiao DUAN Zhen-hua
(Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China)

Abstract Invariance properties are very central in concurrent software systems. Thinking by virtue of establishing an invariant and preserving it has immediate implications for reasoning about programs and their design. In this paper, we emploied an interval temporal logic programming language, called Framed Tempura, to specify concurrent systems, and gave the logical equivalent normal form for this language. Further, we investigated and defined a well-founded relation based on the normal form. Therefore, well-founded induction can be used to verify the invariance properties of the temporal logic programs.

Keywords Temporal logic programs, Normal forms, Well-founded relation, Well-founded induction, Invariants proofs

框架时序逻辑[1-3]是一种适用于描述带有时间周期的软硬件系统性质的规范语言,可以处理顺序、并发[9]、投影等结构,对推理各类性质如安全性、活性具有强大灵活的表达力。为了便于在统一的形式化规范语言中验证程序的性质,继而开发了该逻辑的一个可执行子集:Framed Tempura语言[1-4]。该语言是一种基于共享程序变量的真并发建模语言。作为区间时序逻辑程序语言,Framed Tempura不仅实现了简单结构的顺序语句、选择语句、循环语句,还可以实现合取操作、并行操作、投影操作、同步通信机制等复杂程序结构。

不变量性质(也称不变式),即程序状态经过变化后仍旧保持不变的性质,在并发和实时软件系统开发和设计中起着重要作用。目前已有的验证不变式的方法[6-8],都是针对一般命令式程序语言开发的,而对于时序逻辑语言如Framed Tempura,由于它的执行过程是基于正则形的逻辑等价推导,不同于一般命令式程序语言的执行方式,因此已有的验证方法不能直接用于该类语言不变式的验证。本文针对Framed Tempura语言的正则形的特点,定义了当前状态执行的程序和下一状态执行的程序之间的良基关系($<$),给出了在同一区间上不同状态上可满足的程序的集合($Chain_p$),在该良基结构($Chain_p$,$<$)基础上,采用良基归纳法原理对程序的不变式进行归纳验证。最后本文给出并验证实例说明该方法的应用。

1 框架时序逻辑的语法和语义

Π是命题的可数集合,V是静态变量和动态变量的可数集合。逻辑的项e和公式p归纳定义如下:

$e::=a|x|Oe|f(e_1,\cdots,e_n)$

$p::=\pi|e_1=e_2|P(e_1,\cdots,e_n)|\neg p|Op|p_1\wedge p_2|\exists x:p$

其中a是静态变量,x是动态变量,π是原子命题,在$f(e_1,\cdots,e_n)$和$P(e_1,\cdots,e_n)$中,f表示函数,P表示谓词。如果一个公式(项)不包含时态操作符如Op(Oe),就称该公式为状态公式(项),否则称它为时态公式(项)。

状态s是一个赋值的序偶(I_v,I_p)。对于任意变量$v\in V$,有$s[v]=I_v[v]$,$I_v[v]$表示变量的值;对于任意命题$\pi\in\Pi$,有$s[\pi]=I_p[\pi]$,$I_p[\pi]\in\{\text{true},\text{false}\}$。区间$\sigma=\langle s_0,s_1,\cdots\rangle$是一个非空的状态序列,有限区间$\sigma$的长度记为$|\sigma|$,为状态个数减1;无限区间$\sigma$的长度记为$\omega$。公式(项)的解释是一个四元组$I=(\sigma,i,k,j)$,其中$\sigma$是一个区间,$i,k,j$是非负整数且$i\leqslant k\leqslant j\leqslant|\sigma|$,$(\sigma,i,k,j)$表示公式(项)在区间$\langle s_i,\cdots,s_j\rangle$上且当前状态是$s_k$上的解释。两个区间$\sigma$和$\sigma'$,如果$|\sigma|=|\sigma'|$且对

到稿日期:2008-08-29 返修日期:2008-11-03 本文受国家自然科学基金资助项目(60873018),国家自然科学基金重大资助项目(60433010)资助。

杨潇潇 博士研究生,主要研究方向为程序设计语言的形式语义、形式化验证等,E-mail: yang_xiao@126.com;段振华 教授,博士生导师,主要研究方向为时序逻辑及时序逻辑程序设计、形式化验证等。

于 $y\in V-\{x\}$，有 $I_v^h[y]=I'^h_v[y]$，$I_p^h[\pi]=I'^h{}_p[\pi]$（$0\leqslant h\leqslant|\sigma|$），则我们称这两个区间为 x-等价，简记为：$\sigma \underline{x} \sigma'$。图 1 对项给出了解释，其中 *nil* 表示未定义；图 2 对公式 p 的满足关系 $\vdash$ 给出了定义。

(1) $I[a]=s_k[a]=I_\sigma^k[a]=I_{\sigma'}[a]$，a 是静态变量

(2) $I[x]=s_k[x]=I_\sigma^k[x]$，x 是动态变量

(3) $I[Oe]=\begin{cases}(\sigma,i,k+1,j)，\text{如果 } k<j\\ nil，\text{其他情况}\end{cases}$

(4) $I[f(e_1,\cdots,e_n)]=$

$\begin{cases}nil，\text{如果存在某个 } h\in\{1,\cdots,n\}\text{使得 } I[e_h]=nil；\\ f(I[e_1],\cdots,I[e_n])，\text{其他情况}\end{cases}$

图 1　框架时序逻辑项的解释

(1) $I\vdash\pi$，如果 $s_k[\pi]=I_\sigma^k[\pi]=true$

(2) $I\vdash e_1=e_2$，如果 $I[e_1]=I[e_2]$

(3) $I\vdash P(e_1,\cdots,e_n)$，如果 $P(I[e_1],\cdots,I[e_n])=true$，而且对于任意 $h\in\{1,\cdots,n\}$使得 $I[e_h]\neq nil$

(4) $I\vdash\neg p$，如果 $I\nvdash p$

(5) $I\vdash Op$，如果$(\sigma,i,k+1,j)\vdash p$ 对于 $k<j$

(6) $I\vdash p_1\wedge p_2$，如果 $I\vdash p_1$ 而且 $I\vdash p_2$

(7) $I\vdash\exists x:p$，如果对于某个 σ'使得 $\sigma' \underline{x} \sigma$，有$(\sigma',i,k,j)\vdash p$

图 2　框架时序逻辑公式的解释

称 p 是可满足的，如果$(\sigma,0,0,|\sigma|)\vdash p$（简记为 $\sigma\vdash p$）成立；称 p 是有效的，如果对于所有的区间 σ，$\sigma\vdash p$ 成立；如果在任意区间上的所有状态上公式 p 和公式 q 都具有相同的真假值，则称公式 p 和 q 是等价的，简记为 $p\equiv q$。

2　框架时序逻辑程序设计语言 Framed Tempura

Framed Tempura 语言的所有语句均能够在框架时序逻辑中形式化定义出来。由于篇幅的限制，下面仅给出 Framed Tempura 语言的基本语句的简单介绍。对于复杂语句如投影，同步通信技术和框架技术以及该语言的极小模型及有关性质可参考文献[1-3]。

2.1　语法和语义

Framed Tempura 算术表达式 e 和布尔表达式 b 定义为：

$e::=n|x|Ox|e_0\ op\ e_1$

$b::=true|false|e_0=e_1|e_0<e_1|\neg b|b_0\wedge b_1$

其中 n 是整数，x 是变量，操作符 $op::=+|-|\times|\div$。

Framed Tempura 语言的基本语句包括：空语句 *empty*；赋值语句 $x=e$；条件语句 if b then s_1 else s_2；串连语句 $s_1;s_2$；选择语句 s_1 or s_2；循环语句 while b do p；合取语句 $s_1\wedge s_2$；并行语句 $s_1||s_2$ 以及块语句 $\exists x:s$。

下面给出这些语句的非形式化解释。

空语句 *empty* 是一个简单的不做任何操作的语句；赋值语句 $x=e$ 中的 x 和 e 的类型必须一致，使用该语句我们可以定义一种新的常用的赋值语句，称为单位赋值语句：$x:=e\equiv skip\wedge Ox=e$，含义是变量 x 在下一状态被赋值 e，其中 *skip* 表示区间长度为 1；条件语句 if b then s_1 else s_2 定义为$(b\rightarrow s_1)\wedge(\neg b\rightarrow s_2)$；串联语句（顺序成分）$s_1;s_2$ 可以用投影操作符直接定义[3]，其含义是执行语句 s_1，在其结束状态上执行语句 s_2；选择语句 s_1 or s_2 定义为 $s_1\vee s_2$；循环语句 while b do p 定义为$(b\wedge p)^*\wedge fin(\neg b)$，其中操作$(b\wedge p)^*$ 表示 $b\wedge p$ 执行零次或一次或多次；合取语句 $s_1\wedge s_2$ 的定义如图 2 所示；并行语句 $s_1||s_2$ 定义为 $s_1\wedge(s_2;true)\vee s_2\wedge(s_1;true)$，该并行语句可以模拟真并发；块语句 $\exists x:s$ 的定义如图 2 所示。

2.2　框架时序逻辑程序语言的正则形

定义 1　一个程序 q 是正则形，如果

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

这里 q_{ai} 和 q_{cj} 是状态公式，$k,h\geqslant 0$，$k+h\geqslant 1$。

定理 1　对任意可满足的 Framed Tempura 程序 p，总存在一个正则形 q，使得 $p\equiv q$。

证明：见参考文献[3]。

从定义 1 和定理 1 可知，Framed Tempura 程序 p 的执行过程是把程序 p 逻辑等价地转换到它的正则形的过程。如果 p 在当前状态上的正则形为 $\bigvee_{j=1}^{h}(q_{cj}\wedge Oq_{fj})$，则状态公式（即赋值语句）$q_{cj}$ 在当前状态执行，而程序 q_{fj} 在下一状态继续化简到正则形；如果 p 的正则形为 $\bigvee_{i=1}^{k}(q_{ai}\wedge empty)$，则状态公式 q_{ai} 在当前状态执行，同时在该状态程序 p 终止。下面举例进行说明。

例 1　化简 $p\equiv x=0\wedge$ while $(x<3)$ do $x:=x+1$。

定义 1 的正则形可被简化为$(q_e\wedge empty)\vee(q_c\wedge Oq_f)$。我们省略程序在每个状态上等价转换的步骤，仅给出每个状态上化简得到的正则形。注意上标 i 在 $q_f^i(i=0,1,2,3)$中表示状态数。p（记为 q_f^0）在初始状态 0 上的正则形为：

$q_f^0\equiv x=0\wedge O(x=1\wedge empty;\text{while}(x<3)\text{ do }x:=x+1)$

可知，赋值语句 $x=0$ 在当前状态 0 上被执行，程序 $x=1\wedge empty;\text{while}(x<3)\text{ do }x:=x+1$（记为 q_f^1），将在状态 1 上继续化简，最终得到它的正则形为：

$q_f^1\equiv x=1\wedge O(x=2\wedge empty;\text{while}(x<3)\text{ do }x:=x+1)$

同样地，赋值语句 $x=1$ 在当前状态 1 上被执行，下一状态程序$(x=2\wedge empty;\text{while}(x<3)\text{ do }x:=x+1)$（记为 q_f^2）将在状态 2 上继续化简为它的正则形为：

$q_f^2\equiv x=2\wedge O(x=3\wedge empty;\text{while}(x<3)\text{ do }x:=x+1)$

当前状态 2 上执行赋值语句 $x=2$，下一状态程序$(x=3\wedge empty;\text{while}(x<3)\text{ do }x:=x+1)$（记为 q_f^3）在状态 3 上的正则形为：

$q_f^3\equiv x=3\wedge empty$

$x=3$ 在状态 3 上被执行，同时程序 p 终止。

3　Framed Tempura 程序不变式的归纳验证

3.1　Framed Tempura 程序的良基结构

从例 1 的化简过程中，我们可以得到一个程序序列 q_f^0，$q_f^1,\cdots,q_f^n$，它们分别在状态 $s_0,s_1,\cdots,s_n$ 上执行。这组状态序列 $s_0,\cdots,s_n$ 构成一个区间 σ，使得$(\sigma,0,0,|\sigma|)|=q_f^0$。一般地，有

$(\sigma,0,k,|\sigma|)|=q_f^k(0\leqslant k\leqslant|\sigma|)$

含义为程序 q_f^k 在区间 σ 上的当前状态 k 上被满足。为了方便阐述证明方法，在下文中，用 p_i 表示程序 q_f^i，w_i 表示状态公式 q_c,q_e。当 $i=0$ 时，p_0 表示初始程序 p。

定义 2　集合 $chain_p$ 是程序 p 在化简过程中分解出的下一状态程序 p_k 的集合，定义为：

$chain_p=\{p_k|(\sigma,0,k,|\sigma|)|=p_k,0\leqslant k\leqslant|\sigma|\}$

定义 3（良基关系[5]）　设$\prec$是集合 A 上的二元关系，如果不存在由 A 中元素构成的无穷下降链$\cdots\prec a_i\prec\cdots\prec a_1\prec$

a_0，则称$\prec$为良基关系。

定义 4 对于任意一个 $p_i \in chain_p$，如果 $p_i \equiv w_i \wedge Op_{i+1}$，则我们有 $p_i \prec p_{i+1}$。

由定义 2 和定义 4 不难看出 $(chain_p, \prec)$是一个良基结构。

对任意一个可终止的框架时序逻辑程序 p，存在一个程序 $p_j \in chain_p$ 使得 $p_j \equiv w_j \wedge empty (j \geqslant 1)$，则有 $p \prec^+ p_j$。这里$\prec^+$是$\prec$的传递闭包。

如果程序 p 在执行过程中的任意状态 s_i 上都满足某个性质，则称该性质为程序的不变量性质(不变式)。假设 $p_i \equiv w_i \wedge Op_{i+1}$且 $p_i \in chain_p$，我们约定如下记号：程序 p_i 在状态 i 上满足不变量性质 T 记为 $T_{si}(p_i)$；状态公式 w_i 在状态 i 上满足性质 T 记为 $T_{si}(w_i)$；程序 p 在区间上满足性质 T 记为 $T_\sigma(p)$。

定义 5 给定一个程序 p 和不变式 T，程序 p_i 在状态上满足性质 T 当且仅当状态公式 w_i 在状态上满足 T，即

$$T_{si}(p_i) \Leftrightarrow T_{si}(w_i)$$

程序 p 在区间上满足性质 T，当且仅当任意程序 $p_i \in chain_p$ 在状态上满足性质 T，即

$$T_\sigma(p) \Leftrightarrow T_{si}(p_i) \text{对于所有 } p_i \in chain_p$$

3.2 良基归纳法验证不变式

良基归纳法原理[5]：设$\prec$是集合 A 上的良基关系，T 是某一性质，则 $\forall a \in A.\ T(a)$ 当且仅当 $\forall a \in A.\ (\forall b \prec a.\ T(b) \Rightarrow T(a))$。即，要证明某个对良基集的所有元素性质成立，只要证明如果对任一元素 a 的所有前驱该性质成立，则对 a 性质也成立。定义 4 给出了框架时序逻辑程序在相邻状态上的良基关系，下面我们将良基归纳法原理应用到框架时序逻辑程序的不变量性质的证明中。

定理 2 给定一个程序 p，$(chain_p, \prec)$是良基结构。欲证明对所有元素 $p_i \in chain_p$ 在状态上满足性质 T，也就是程序 p 在区间上满足性质 T，只要证明：对于 $p_{i+1} \in chain_p$，如果使得 $p_i \prec p_{i+1}$的所有 p_i 在状态上满足性质 T，则 p_{i+1}在状态上满足性质 T。该方法形式化描述如下：

$$\frac{\forall p_i.\ (p_i \prec p_{i+1}, T_{si}(p_i) \rightarrow T_{s(i+1)}(p_{i+1}))}{T_\sigma(p)}$$

$$(i \geqslant 0, p_i, p_{i+1} \in chain_p)$$

例 2 验证程序

$p \equiv x = n \wedge y = 1 \wedge \text{while}(x > 0)\ \text{do}(y := y \times x \wedge x := x - 1)$

满足不变式 $T = (y \times x! = n!)$。

分析：该程序的不变式为 $T = (y \times x! = n!)$。一般地，假定状态 s_i 上 $p_i \equiv w_i \wedge Op_{i+1}$，程序 p 分解出的下一状态程序的集合为 $chain_p = \{p_0, p_1, p_2, \cdots\}$。由定义 4，对于任意元素 $p_i, p_{i+1} \in chain_p$，存在良基关系 $p_i \prec p_{i+1}$。假设 p_i 在状态 s_i 上满足性质 T，需要证明 p_{i+1}在状态 s_{i+1} 满足性质 T，即证明 $T_{si}(p_i) \rightarrow T_{s(i+1)}(p_{i+1})$成立。由定义 5 可知，程序 p_i 在状态上满足性质等价于状态公式 w_i 在状态上满足性质，因此等价于证明 $T_{si}(w_i) \rightarrow T_{s(i+1)}(w_{i+1})$。

证明：令 y_i, x_i 分别表示变量 y, x 在状态 s_i 上的值。

1)基础：在状态 s_0 上，$y_0 = 1, x_0 = n$，显然，$y_0 \times x_0! = n!$ 满足性质 T。

2)归纳：假设 p_i 在状态 s_i 上满足性质 T，即 $T_{si}(p_i)$成立，由定义 5 可知，也就是 $T_{si}(w_i)$成立，即 $T_{si}(w_i) = (y_i \times x_i! = n!)$成立。需要证明在下一状态 s_{i+1}上该性质成立，也就是需要证明 $T_{s(i+1)}(w_{i+1}) = y_{i+1} \times x_{i+1}! = n!$ 成立。由给定的程序易知，状态 s_{i+1}上的状态公式 w_{i+1}为 $(y_{i+1} = y_i \times x_i) \wedge (x_{i+1} = x_i - 1)$ (1)

把式(1)代入到 $T_{s(i+1)}(w_{i+1})$中，得到

$$T_{s(i+1)}(w_{i+1}) = y_{i+1} \times x_{i+1}! = (y_i \times x_i) \wedge (x_i - 1)!$$
$$= y_i \times x_i! = n! \quad (2)$$

故 $T_{si}(w_i) \rightarrow T_{s(i+1)}(w_{i+1})$成立。因此，由定义 5 可知 $T_{si}(p_i) \rightarrow T_{s(i+1)}(p_{i+1})$成立。由定理 2 可知，$T_\sigma(p)$成立。因此程序 p 在区间上满足不变式 T。

结束语 本文针对区间时序逻辑程序语言 Framed Tempura 的执行特性，分析并构造了该语言的一种良基结构，进而利用良基归纳法原理对该语言所描述的系统的不变式进行归纳验证。本文提出的良基归纳法为区间时序逻辑程序语言的不变式验证问题提供了新的解决途径。作为对该类问题的初次探索，本文仅针对 Framed Tempura 语言的简单语句进行了实例验证。作为今后的一项重要工作，我们将为该语言建立一套完整的推演系统，并开发基于 PVS 的静态验证工具，进行更多的较为复杂的软硬件系统的实例验证。

参 考 文 献

[1] Duan Z, Yang X, Koutny M. Framed Temporal Logic Programming[J]. Science of Computer Programming, 2008, 70(1): 31-61

[2] Duan Z. An Extended interval Temporal Logic and A Framing Technique for Temporal Logic Programming [D]. University of Newcastle upon Tyne, 1996

[3] Duan Z, Koutny M. A Framed Temporal Logic Programming Language[J]. Journal of Computer Science and Technology, 2004, 19: 341-351

[4] Ma Y, Duan Z, Wang X, et al. An Interpreter for Framed Tempura and Its Application[C]// Proceedings of the 1st IEEE and IFIP International Symposium on TASE. 2007: 251-260

[5] Winskel G. The Formal Semantics of Programming Languages [M]. An Introduction. MIT Press, 1993

[6] Fujita M, Kono S, Tanaka H, et al. Tokio: Logicprogramming language based on temporal logic and its compilation to PROLOG[C]// Third International Conference on Logic Programming, Volume 225 of LNCS. Springer-Verlag, 1986: 695-709

[7] Saidi H. A Tool for Proving Invariance Properties of Concurrent Systems Automatically[C]// TACAS'96, Germany, LNCS 1056. Springer-Verlag, 1996

[8] Saidi H. The Invariant Checker: Automated Deductive Verification of Reactive System[C]// Proceedings of the 9th Conference on Computer-Aided Verification, Israel, LNCS 1254. Springer-Verlag, 1997

[9] 雷丽晖，段振华. 使用扩展区间时序逻辑为并发工作流建模[J]. 西安电子科技大学学报, 2007 34(4): 673-680