利用投影时序逻辑的多内核进程调度建模与验证

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

摘要：针对软件测试无法满足多内核处理器上进程调度的验证需要这一问题，提出利用投影时序逻辑(PTL)的定理证明方法来验证进程调度.使用PTL公式建立了支持当前主流进程调度算法的多内核处理器进程调度一般模型S，并将系统期望的性质描述为PTL公式P，在PTL公理系统的基础上，通过证明S蕴含P是否为一个定理来验证系统是否具备该性质.以2内核处理器上的多级反馈队列算法的正确性为案例进行检验，结果表明所提方法可验证多内核处理器进程调度的系统性质，保证多内核进程调度的可靠性.由于多内核处理器的进程调度具备了并发系统的主要特点，因此该方法也适用于一般的并发系统验证.

关键词：投影时序逻辑；进程调度；定理证明；多核处理器；调度验证
中图分类号：TP301 **文献标志码**：A **文章编号**：0253-987X(2010)03-0052-06

Modeling and Verification of Processes Scheduling Based on Projection Temporal Logic for Multi-Core CPU

SHU Xinfeng, DUAN Zhenhua
(Institute of Computing Theory and Technology, Xidian University, Xi′an 710071, China)

Abstract: Software testing is unable to meet the verification needs of process scheduling for multi-core CPU, therefore a theorem proving approach with projection temporal logic (PTL) is adapted to verify the process scheduler. A general model supporting the most commonly used scheduling algorithms for multi-core CPU is constructed by a PTL formula S, and the desired property of the system is described by a PTL formula P, then whether the system possesses the property can be identified by proving whether or not S implying P is a theorem based on the axiomatization of PTL. As a case study, the correctness of the processes scheduling with the multi-level feedback queue scheduling algorithm over a two-core CPU is proved, which indicates that the proposed approach can be used to verify system properties of process scheduling over multi-core CPU and hence to ensure the reliability of the process scheduler. Since the process scheduler over multi-core CPU has the typical features of concurrent system, the method can also be applied to verify general concurrent systems.

Keywords: projection temporal logic; process scheduling; theorem proving; multi-core CPU; scheduling verification

投影时序逻辑(PTL)[1]是区间时序逻辑(ITL)[2]的扩展，引入了全新的2个投影操作符Projection (prj)和Projection-plus(⊕)[3-4]，并同时支持有穷和无穷模型，具备完全正则表达式的描述能力[3]，适用于各类软硬件系统的描述和验证[5-8].更重要的是，PTL继承了ITL可以执行的优点[9].

收稿日期：2009-09-28. 作者简介：舒新峰(1975—)，男，博士生；段振华(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助重大国际合作项目(60910004)；国家“973计划”重点资助项目(2010CB328102)；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)；总装备部预研项目(51315050105).

文献[10]中使用 PTL 的一个带有 Frame 技术的子集定义了逻辑程序设计语言 MVSL，可以方便地描述顺序、选择、循环、并发、信号量等程序结构，从而可以在同一 PTL 逻辑框架内完成对待验证系统的建模、性质描述和验证，大大简化了验证过程.

进程调度是现代操作系统的核心组成部分，一旦失效会产生十分严重的后果. 测试是目前保证进程调度可靠性的最重要的手段，但只能证明系统中有错误，而不能证明系统中没有错误. 另外，由于在多内核 CPU 上进程间已经实现了真正意义上的并发执行，而并发进程推进速度的不可预知性导致了系统运行结果的不确定性，所以使得传统测试方法更加难以胜任多核处理机进程调度验证的需要. 与测试不同，形式化验证采用数学的方法，在严密的理论基础支撑下，用推导和定理证明的方法来证明设计结果的正确性，已成为最有希望解决进程调度验证的途径. 本文将 PTL 引入到多核处理器进程调度的验证中，使用 PTL 逻辑公式为多核处理器上的进程调度建立了一个通用模型，可以支持多种主流的进程调度算法，并结合实例展示了基于 PTL 的定理证明方法在验证多内核进程调度上的有效性.

1　投影时序逻辑

1.1　语法

令 P_{prop} 是原子命题的可数集合，V 为静态变量和动态变量的可数集合，D 为论域. PTL 的项 e 和公式 P 递归定义如下

$$e ::= d \mid a \mid x \mid \bigcirc e \mid f(e_1,\cdots,e_n)$$

$$P ::= p \mid e_1 = e_2 \mid \rho(e_1,\cdots,e_n) \mid \neg P \mid \bigcirc P \mid P \wedge Q \mid \exists v : P \mid (P_1,\cdots,P_m)\text{prj } Q \mid (P_1,\cdots,(P_k,\cdots,P_l)^{\oplus},\cdots,P_m)\ \text{prj } Q$$

式中：$d\in D$ 为常量；$a\in D$ 为静态变量；$x\in D$ 为动态变量；$v\in V$ 为任意的静态或动态变量；$p\in P_{\text{prop}}$ 为原子命题；$P_1,\cdots,P_k,\cdots,P_l,\cdots,P_m$ 及 Q 为 PTL 的合式公式；$f(e_1,\cdots,e_n)$ 与 $\rho(e_1,\cdots,e_n)$ 分别表示带有 n 元参数的函数与谓词.

除了连接符 $\vee$、$\rightarrow$、$\leftrightarrow$，全称量词 $\forall$ 以及逻辑公式 true 和 false 的定义与经典逻辑相同外，使用如下派生的时态公式

$$\varepsilon \triangleq \neg\bigcirc\text{true} \qquad P;Q \triangleq (P,Q)\ \text{prj}\ \varepsilon$$

$$\bar{\varepsilon} \triangleq \neg\varepsilon \qquad \bigcirc^0 P \triangleq P$$

$$\Diamond P \triangleq \text{true};P \qquad \bigcirc^n P \triangleq \bigcirc(\bigcirc^{n-1}P)\ (n>0)$$

$$\Box P \triangleq \neg\Diamond\neg P \qquad \text{fin}(P) \triangleq \Box(\varepsilon\rightarrow P)$$

$$\odot P \triangleq \neg\bigcirc\neg P \qquad \text{halt}(P) \triangleq \Box(\varepsilon\leftrightarrow P)$$

$$\text{skip} \triangleq \bigcirc\varepsilon \qquad \text{keep}(P) \triangleq \Box(\bar{\varepsilon}\rightarrow P)$$

$$\text{len}(n) \triangleq \bigcirc^n\varepsilon \qquad \bigcirc^{i..j}P \triangleq \bigcirc^i P \wedge\cdots\wedge \bigcirc^j P$$

$$P^+ \triangleq (P^{\oplus})\ \text{prj}\ \varepsilon \qquad P^* \triangleq \varepsilon \vee P^+$$

$$P \parallel Q \triangleq ((P;\text{true})\wedge Q)\vee(P\wedge(Q;\text{true}))\vee(P\wedge Q)$$

$$\text{keepx}(P,x_1,\cdots,x_n) = \Box(P\rightarrow\wedge_{i=1}^{n}\bigcirc x_i = x_i)$$

投影时序逻辑操作符的优先级定义如下（数字越小优先级越高）

1：$\neg$　2：$\bigcirc,\odot,\Box,\Diamond,+,*$　3：$\exists,\forall$　4：$=$

5：$\wedge$　6：$\vee,\parallel$　7：$\rightarrow,\leftrightarrow$　8：prj，$\oplus$，；

若一个公式（项）不包含时态操作符，称为状态公式（项），否则称为时态公式（项）；若一个公式（项）不含动态变量，则称其为静态公式（项）.

1.2　语义

由于篇幅所限，此处只介绍 PTL 的语义模型，详细语义解释请参考文献[1,4].

状态 s 是一个赋值的序偶 (I_p,I_v). 对于任意变量 $v\in V$，有 $s[v]=I_v[v]\in D$；对于任意 $p\in P_{\text{prop}}$，有 $s[p]=I_p[p]\in\{\text{true},\text{false}\}$. 区间 $\sigma=\langle s_0,s_1,\cdots\rangle$ 是一个非空的状态序列. 有穷区间 σ 的长度记为 $|\sigma|$，为状态数减 1；无穷区间 σ 的长度记为 ω. 公式（项）的解释是一个三元组 $I=(\sigma,i,j)$，其中 σ 是一个区间，i,j 是非负整数且 $0\leqslant i\leqslant j\leqslant|\sigma|$. 用 $\sigma_{(i..j)}$ 表示区间 σ 中的子区间 $\langle s_i,\cdots,s_j\rangle$，$(\sigma,i,j)$ 表示公式（项）在 $\sigma_{(i..j)}$ 上且当前状态是 s_i 时的解释.

公式 P 的可满足性和有效性定义为：P 是可满足的，当且仅当存在区间 σ，$(\sigma,0,|\sigma|)\models P$ 成立，简记为 $\sigma\models P$；P 是有效的，当且仅当对于所有的区间 σ，$\sigma\models P$ 成立，简记为 $\models P$.

1.3　投影操作符分析

投影公式 $(P_1,\cdots,P_m)\text{prj}\ Q$ 的直观含义是进程 Q 与进程序列 $P_1,\cdots,P_m$ 并行在 2 个不同的时态空间内执行，其中进程 $P_1,\cdots,P_m$ 在一个时态空间顺序执行，进程 Q 执行的时态空间则由每个进程 P_i（$1\leqslant i\leqslant m$）执行的时态空间的端点构成. 子进程 $P_1,\cdots,P_m$ 与 Q 的执行高度自治，每个进程都可以决定自己执行的时态空间，仅在每个进程 P_i 执行开始和结束时才和 Q 发生信息交互，可以方便地描述进程间的并发和同步.

投影公式 $(P_1,\cdots,P_m)\text{prj}\ Q$ 中每一个 P_i 只能按顺序执行 1 次，然而在现实系统中有时期望 P_1，

…,P_m 里有一段 P_k,…,P_l($1\leqslant k\leqslant l\leqslant m$)能重复执行多次,而公式$(P_1,\cdots,(P_k,\cdots,P_l)^{\oplus},\cdots,P_m)\text{prj}\ Q$就具备这种描述能力,其含义根据 l 和 m 的取值分为:当 $l<m$ 时,P_k,…,P_l 段只可以重复有穷次;当 $l=m$时,P_k,…,P_l 段可以重复有穷或无穷多次.

2 多核处理器进程调度建模

图 1 给出了一个典型多内核计算机系统的进程调度示意图:硬件包括 1 个具有 m 个内核的 CPU 以及 n 个各类型的 I/O 设备;进程队列管理所有等待 CPU 或某 I/O 设备的进程;进程调度器在硬件资源空闲时根据一定的策略选择一个等待进程投入运行. 系统允许多个进程并发执行,然而任何时刻一个资源只能被一个进程占用.

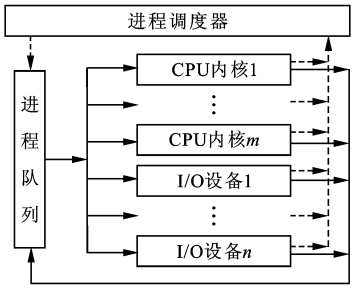

图 1 多核 CPU 进程调度示意图

由于计算机中一类硬件资源可以有多个具体的物理设备,为便于设备分配,引入逻辑设备与物理设备的概念,其中逻辑设备名为物理设备的类别名,并且为每一个逻辑(物理)设备赋予唯一编号. 此外,为每一逻辑设备设置设备类型属性,限定值为 1 表示共享设备,为 0 表示独占设备. 进程在执行时仅对逻辑设备提出请求,而进程调度器则根据一定策略为其分配具体的物理设备.

令逻辑设备编号的集合为 Γ,物理设备编号的集合为 Δ,进程标识符的集合为 Σ. 另外,定义函数 $L_{\text{D}}:\Delta\to\Gamma$ 来根据给定的物理设备编号计算对应的逻辑设备编号,定义函数 $T_{\text{D}}:\Gamma\to\{1,0\}$来求出给定的逻辑设备的设备类型.

为满足各种主流进程调度算法的建模需要,对系统中每个进程 $P_i(i\in\Sigma)$定义了表 1 列出的 8 个动态变量,其中 $\mathbf{Z}$ 为整数集合,$\mathbf{Z}^+$ 为非负整数集合.

2.1 进程描述

从直观意义上讲,一个进程的执行实际上是对不同计算机资源的使用序列,然而由于并发执行时资源竞争的存在,进程必须先提出资源请求,等待进程调度器为其分配好资源后方可继续运行.

表 1 与进程 $P_i(i\in\Sigma)$相关的动态变量

变量名	定义域	含 义
w_i	$\Gamma\cup\{0,-1\}$	P_i 当前等待的逻辑设备编号
u_i	$\Delta\cup\{0\}$	P_i 当前使用的物理设备编号
v	Δ	P_i 分配到的物理设备编号
l_i	$\mathbf{Z}^+$	P_i 当前的优先级(优先数)
t_i	$\mathbf{Z}^+$	P_i 等待分配设备的时间
x_i	$\mathbf{Z}^+$	P_i 所需要的设备使用时间
y_i	$\mathbf{Z}$	系统允许 P_i 使用设备的时间
z_i	$\mathbf{Z}^+$	P_i 申请设备的批次号

进程 $P_i(i\in\Sigma)$对逻辑设备 $\gamma(\gamma\in\Gamma)$申请使用 $k(k\in\mathbf{Z}^+)$个单位时间的函数 $A_{\text{p}}(i,\gamma,k)$定义为

$$A_{\text{p}}(i,\gamma,k)\overset{\text{def}}{=}w_i=\gamma\wedge x_i=k\wedge A_i\wedge\varepsilon$$

进程 P_i 等待第 $n(n\in\mathbf{N})$批次资源的函数定义为

$$W_{\text{t}}(i,n)\overset{\text{def}}{=}z_i=n\wedge t_i=0\wedge\text{keep}(\bigcirc t_i=t_i+1)\wedge\text{keepx}(\bar{\varepsilon},w_i,x_i,l_i,z_i)\wedge\text{halt}(\exists v:(u_i=v\wedge w_i=L_{\text{D}}(v)))$$

进程 P_i 对当前资源的使用函数定义为

$$U_{\text{s}}(i)\overset{\text{def}}{=}(T_{\text{D}}(L_{\text{D}}(u_i))=1\to y_i=T_{\text{S}}(l_i)\wedge S_{\text{u}}(i))\wedge(T_{\text{D}}(L_{\text{D}}(u_i))=0\to E_{\text{u}}(i))\wedge A_{\text{j}}(i)$$

其中,对共享设备和独占设备需要做不同处理. 对于共享设备,需要为 P_i 预分配 $T_{\text{S}}(l_i)$个单位时间,这里 $T_{\text{S}}:\mathbf{Z}^+\to\mathbf{Z}^+$ 为根据优先级计算分配时间的函数,$A_{\text{j}}(i)$为本次使用完物理设备后修改 P_i 优先级的函数. $T_{\text{S}}(l_i)$和 $A_{\text{j}}(i)$的具体定义取决于相关算法. 共享设备的使用函数 $S_{\text{u}}(i)(i\in\Sigma)$定义为

$$S_{\text{u}}(i)\overset{\text{def}}{=}\square(x_i>1\wedge(y_i>1\vee y_i\leqslant 0)\wedge\bigcirc\bar{\varepsilon}\to\bigcirc(w_i=0)\wedge\bigcirc x_i=x_i-1\wedge\bigcirc y_i=y_i-1)\wedge\text{keepx}(\bigcirc\bar{\varepsilon},u_i,z_i,l_i)\wedge\square(x_i=1\vee y_i=1\vee\bigcirc(u_i=0)\leftrightarrow\bigcirc\varepsilon)\wedge\square(x_i>1\wedge\bigcirc\varepsilon\to\bigcirc w_i=L_{\text{D}}(u_i)\wedge\bigcirc x_i=x_i\wedge\bigcirc z_i=z_i\wedge\bigcirc\neg A_i)\wedge\square(x_i=1\wedge\bigcirc\varepsilon\to\bigcirc z_i=z_i+1)$$

独占设备的使用函数 $E_{\text{u}}(i)$ $(i\in\Sigma)$定义为

$$E_{\text{u}}(i)\overset{\text{def}}{=}\square(x_i>1\to\bigcirc(w_i=0)\wedge\bigcirc x_i=x_i-1)\wedge\text{keepx}(\bigcirc\bar{\varepsilon},u_i,z,l_i)\wedge\square(x_i=1\leftrightarrow\bigcirc\varepsilon\wedge\bigcirc z_i=z_i+1)$$

进程 $P_i(i\in\Sigma)$的执行可以描述为 PTL 公式序列

$$P_i\overset{\text{def}}{=}(l_i=n_i\wedge A_{\text{p}}(i,\gamma_1,k_1),(W_{\text{t}}(i,1),U_{\text{s}}(i))^{\oplus},A_{\text{p}}(i,\gamma_2,k_2),(W_{\text{t}}(i,2),U_{\text{s}}(i))^{\oplus},\cdots,A_i\wedge E_i\wedge\varepsilon)$$

式中:$l_i=n_i(n_i\in\mathbf{Z}^+)$表示 P_i 的初始优先级;$(W_{\text{t}}(i,1),U_{\text{s}}(i))^{\oplus}$表示子序列 $W_{\text{t}}(i,1)$、$U_{\text{s}}(i)$可以重复 1

次到多次；原子命题 E_i 表示进程 P_i 成功结束.

2.2　进程调度条件

无论进程调度器采用何种调度算法，在进程调度时必须满足下面 6 个条件：

(1)进程 P_i 要使用物理设备 $\delta(i\in\Sigma,\delta\in\Delta)$，必须对相应的逻辑设备提出请求或者正在占据着物理设备，即 $C_1\overset{\text{def}}{=}u_i=\delta\rightarrow(w_i=L_{\text{D}}(\delta)\vee w_i=0)$；

(2)任何时候只能有一个进程使用物理设备 δ，即 $C_2\overset{\text{def}}{=}u_i=\delta\rightarrow\bigwedge\limits_{j\neq i}\neg(u_j=\delta)$；

(3)无论任何时候，只要有进程对逻辑设备 $\gamma(\gamma\in\Gamma)$ 提出了请求，则必然有进程能够分配到属于 γ 的物理设备而得以运行，即 $C_3\overset{\text{def}}{=}w_i=\gamma\rightarrow\bigvee\limits_{j\in\Sigma}\exists v:(u_j=v\wedge\gamma=L_{\text{D}}(v))$；

(4)δ 要处于空闲状态，必须确保所有针对 δ 所属的 γ 的请求得到满足，即 $C_4\overset{\text{def}}{=}\bigwedge\limits_{i\in\Sigma}\neg(u_i=\delta)\rightarrow\bigwedge\limits_{j\in\Sigma}(w_j=L_{\text{D}}(\delta)\rightarrow\exists v:(u_j=v\wedge w_j=L_{\text{D}}(v)))$；

(5)P_i 申请设备失败当且仅当属于 P_i 所申请逻辑设备的物理设备已经全部分配给了其他进程，即 $C_5\overset{\text{def}}{=}u_i=0\leftrightarrow\bigwedge\limits_{\delta\in\Delta}(w_i=L_{\text{D}}(\delta)\rightarrow\bigvee\limits_{j\neq i}u_j=\delta)$；

(6)一旦进程 P_i 成功结束，从此时刻开始不再提出任何设备请求，即 $C_6\overset{\text{def}}{=}E_i\rightarrow\Box(w_i=-1)$.

以上 6 个调度条件对于所有进程、逻辑设备和物理设备均成立，即

$$C_{\text{con}}\overset{\text{def}}{=}\bigwedge_{i\in\Sigma}\bigwedge_{\gamma\in\Gamma}\bigwedge_{\delta\in\Delta}(C_1\wedge C_2\wedge C_3\wedge C_4\wedge C_5\wedge C_6)$$

2.3　进程优先级

进程获取 CPU 及 I/O 资源的优先级与具体的进程调度算法有关. 令 $H_{\text{p}}(i,j)$ 为 P_i 与 P_j 间的优先关系函数. 用表 1 中的进程相关变量，不难定义出各种主要进程调度算法的 $H_{\text{p}}(i,j)$，如多级反馈队列定义为 $H_{\text{p}}(i,j)\overset{\text{def}}{=}(l_i<l_j)\vee(l_i=l_j\wedge t_i>t_j)$.

2.4　进程调度描述

进程 P_i 比 $P_j(i,j\in\Sigma)$ 在获取逻辑设备 $\gamma(\gamma\in\Gamma)$ 时拥有更高优先级的函数为 $P_{\text{r}}(i,j,\gamma)\overset{\text{def}}{=}(w_i=\gamma\wedge w_j=\gamma\wedge H_{\text{p}}(i,j))\vee(w_i=\gamma\wedge\neg(w_j=\gamma))$.

进程调度器在任何时候都必须确保在高优先级进程还没有获得资源的情况下，低优先级进程不能率先获得资源，即

$$P_{\text{pol}}\overset{\text{def}}{=}\bigwedge_{\gamma\in\Gamma}\bigwedge_{\delta\in\Delta}\bigwedge_{j\neq i}\neg(P_{\text{r}}(i,j,\gamma)\wedge\exists v:(L_{\text{D}}(v)=\gamma\wedge u_j=v)\wedge\neg\exists v:(L_{\text{D}}(v)=\gamma\wedge u_i=v))$$

公式 C_{con} 与 P_{pol} 一起确定了进程调度器的调度策略，且在任何时刻均成立，即

$$S_{\text{sch}}\overset{\text{def}}{=}\Box(C_{\text{con}}\wedge P_{\text{pol}}\wedge\bar{\varepsilon})$$

进程调度器与 P_i 各自运行在自己的时态空间，仅在 P_i 对某资源提出请求并且该资源空闲或者可抢占时，两者才发生交互，因此进程调度器对 P_i 的调度可以表达为投影公式 P_i prj S_{sch}. 由于进程调度器在调度时对所有并发进程完全平等，从而系统对所有并发进程的调度可用以下 PTL 公式表达

$$S_{\text{sys}}\overset{\text{def}}{=}(P_1\text{ prj }S_{\text{sch}})\parallel\cdots\parallel(P_{|\Delta|}\text{ prj }S_{\text{sch}})$$

3　进程调度验证实例

3.1　系统描述

不失一般性，令待验证的计算机系统中 CPU 为 2 核，拥有 1 台激光打印机和 1 个以太网卡，逻辑设备和物理设备的对应关系如表 2 所示. 另外，进程调度器选用多级反馈队列(MFQ)算法，且系统分配的时间片与等待进程所属队列编号 l_i 的关系函数为 $T_{\text{S}}(l_i)\overset{\text{def}}{=}l_i+2$，进程优先级修改函数为 $A_{\text{j}}(l_i)\overset{\text{def}}{=}(x_i>1\wedge y_i=1\wedge\bigcirc\varepsilon\rightarrow\bigcirc l_i=l_i+1)$.

表 2　系统逻辑设备与物理设备对照表

逻辑设备			物理设备	
编号	名称	设备类型	编号	名称
1	CPU	1	1	内核 1
			2	内核 2
2	PRT	0	3	激光打印机
3	NET	0	4	以太网卡

假定有 3 个进程 P_1、P_2 和 P_3 正在执行，其对资源的需求情况为：P_1：NET(1)，CPU(6)；P_2：CPU(1)，PRT(2)，CPU(2)；P_3：CPU(5)，NET(1). 其中，括号内的值表示需要使用的单位时间.

假设这 3 个进程当前所在的队列号分别为 2、1 和 3，逻辑设备编号的集合 $\Gamma=\{1,2,3\}$，物理设备编号的集合 $\Delta=\{1,2,3,4\}$，进程标识符的集合 $\Sigma=\{1,2,3\}$. 将进程 P_1、P_2 和 P_3 的执行描述成 PTL 公式序列如下

$$P_1\overset{\text{def}}{=}(l_1=2\wedge A_{\text{p}}(1,3,1),(W_{\text{t}}(1,1),U_{\text{s}}(1))^{\oplus},A_{\text{p}}(1,1,6),(W_{\text{t}}(1,2),U_{\text{s}}(1))^{\oplus},A_1\wedge E_1\wedge\varepsilon)$$

$$P_2\overset{\text{def}}{=}(l_2=1\wedge A_{\text{p}}(2,1,1),(W_{\text{t}}(2,1),U_{\text{s}}(2))^{\oplus},A_{\text{p}}(2,2,2),(W_{\text{t}}(2,2),U_{\text{s}}(2))^{\oplus},A_{\text{p}}(2,1,2),(W_{\text{t}}(2,3),U_{\text{s}}(2))^{\oplus},A_2\wedge E_2\wedge\varepsilon)$$

$$P_3\overset{\text{def}}{=}(l_3=3\wedge A_{\text{p}}(3,1,5),(W_{\text{t}}(3,1),U_{\text{s}}(3))^{\oplus},A_{\text{p}}(3,3,1),(W_{\text{t}}(3,2),U_{\text{s}}(3))^{\oplus},A_3\wedge E_3\wedge\varepsilon)$$

根据多级反馈队列(MFQ)算法，P_1、P_2 和 P_3

的进程调度图如图2所示，其中进程名 P_i 后括号内的第一个数字为所在的队列号，第二个数字为需要使用设备的时间. 尽管这3个进程在同一时刻开始执行，然而在MFQ算法的调度下，经过一段时间并发执行后，按照 P_2、P_3 和 P_1 的次序先后结束.

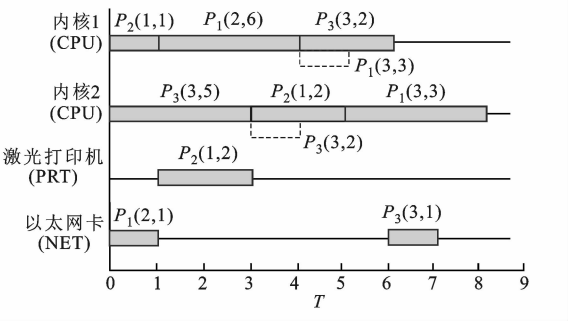

图2 进程 P_1、P_2 和 P_3 的调度图

3.2 基于定理证明的系统验证

下面基于PTL的公理系统[4,11]证明上述2内核系统对进程 P_1、P_2 和 P_3 调度的正确性. 令记号 $\vdash$ 表示公理系统中的形式可推演关系. 为阐述方便，将 $\vdash(P\leftrightarrow Q)$ 简记为 $P\cong Q$，将 $\vdash(P\rightarrow Q)$ 简记为 $P\sqsupset Q$，p_s 表示任意的状态公式，符号 $\Rightarrow$ 表示“推导出”.

定理 上述2内核计算机系统对进程 P_1、P_2 和 P_3 的调度完全正确，即 $S_{sys}\sqsupset\Diamond(E_2;\bigcirc E_3;\bigcirc E_1)$.

为了证明的需要，下面给出PTL公理系统中的一些主要公理、推导规则和定理.

公理AT：$\vdash P$，其中 P 为经典一阶逻辑永真式的替代实例.

公理AXA：$\bigcirc(P\wedge Q)\cong\bigcirc P\wedge\bigcirc Q$.

公理APS：$(p_s\wedge P_1,\cdots,P_m)$ prj $Q\cong p_s\wedge(P_1,\cdots,P_m)$ prj $Q\cong(P_1,\cdots,P_m)$ prj $(p_s\wedge Q)$.

公理APSEF：$(P_1,\cdots,p_s\wedge\varepsilon,P_i,\cdots,P_m)$prj $Q\cong(P_1,\cdots,p_s\wedge P_i,\cdots,P_m)$ prj Q.

公理APX2：$(P_1\wedge\bar{\varepsilon},\cdots,P_m)$ prj $\bigcirc Q\cong P_1\wedge\bar{\varepsilon}$; $(P_2,\cdots,P_m)$ prj Q.

公理APPR：$(P_1,\cdots,(P_k,\cdots,P_l)^{\oplus},\cdots,P_m)$ prj $P\cong((P_1,\cdots,P_k,\cdots,P_l,\cdots,P_m)$ prj $P)\vee((P_1,\cdots,P_k,\cdots,P_l,(P_k,\cdots,P_l)^{\oplus},\cdots,P_m)$ prj $P)$.

公理ATXE：$e_1=e_2(t_1,\cdots,\bigcirc t_i,\cdots,t_m)\cong\exists a:(e_1=e_2(t_1,\cdots,a,\cdots,t_m)\wedge\bigcirc(a=t_i))$.

公理ATSR：$e_s=e_s'\wedge P(e_s)\cong e_s=e_s'\wedge P(e_s')$，其中 e_s 及 e_s' 为状态项.

公理ADF：$f(d_1,\cdots,d_m)=d$，其中 $d_1,\cdots,d_m$ 与 d 在 D 上有关系 f.

公理ADPT：$\rho(d_1,\cdots,d_m)\cong$true，其中 $d_1,\cdots,d_m$ 在 D 上具有 m 元关系 ρ.

推导规则IR：$P_1\cong P_2\Rightarrow Q\cong Q[P_1/P_2]$，其中 $Q[P_1/P_2]$ 表示将 Q 中出现的一些子公式 P_2 替换为 P_1 所得到的公式.

推导规则IMP：$\vdash(P\rightarrow Q),\vdash P\Rightarrow\vdash Q$.

定理TRXA：$(p_s\wedge\bigcirc P)\parallel(q_s\wedge\bigcirc Q)\cong p_s\wedge q_s\wedge\bigcirc(P\parallel Q)$.

定理TAR：$\Box P\cong(P\wedge\varepsilon)\vee(P\wedge\bigcirc\Box P)$.

定理TRSA：$(p_s\wedge P)\parallel Q\cong p_s\wedge(P\parallel Q)\cong(P\parallel p_s\wedge Q)$.

定理DIRI：$P\sqsupset P',Q\sqsupset Q'\Rightarrow(P\parallel Q)\sqsupset(P'\parallel Q')$.

证明 为方便证明，令 $s\cong C_{con}\wedge P_{pol}$. 根据公理AXA及定理TAR，有 $S_{sch}\cong s\wedge\bigcirc S_{sch}$. 从而，根据公理AT、APS、APSEF和APPR，推导规则IR，以及定理TAR和TRSA，有

$$S_{sys}\cong l_1=2\wedge w_1=3\wedge l_2=1\wedge w_2=1\wedge l_3=3\wedge w_3=1\wedge\bigwedge_{i\in\Sigma}t_i=0\wedge s\wedge S_{sys}\tag{1}$$

根据经典谓词演算，不难证明

$$l_1=2\wedge w_1=3\wedge l_2=1\wedge w_2=1\wedge l_3=3\wedge w_3=1\wedge\bigwedge_{i\in\Sigma}t_i=0\wedge s\sqsupset(u_1=4\wedge u_2=1\wedge u_3=2)\vee(u_1=4\wedge u_2=2\wedge u_3=1)\tag{2}$$

由式(1)、式(2)及公理AT、ADF和ADPT，有

$$S_{sys}\cong(u_1=4\wedge u_2=1\wedge u_3=2\wedge S_{sys})\vee(u_1=4\wedge u_2=2\wedge u_3=1\wedge S_{sys})\tag{3}$$

由公理AT、APPR、ATXE、ATSR及定理TRXA，有

$$u_1=4\wedge u_2=1\wedge u_3=2\wedge S_{sys}\sqsupset\bigcirc(l_1=2\wedge l_2=1\wedge l_3=3\wedge w_3=0\wedge u_3=2\wedge(((A_p(1,1,6),\cdots,A_1\wedge E_1\wedge\varepsilon)\text{prj }S_{sch})\parallel((A_p(2,2,2),\cdots,A_2\wedge E_2\wedge\varepsilon)\text{ prj }S_{sch})\parallel(x_3=4\wedge y_3=3\wedge z_3=1\wedge U_s(3);((A_p(3,3,1),\cdots,A_3\wedge E_3\wedge\varepsilon)\text{ prj }S_{sch})\vee(((W_t(3,1),U_s(3))^{\oplus},\cdots)\text{ prj }S_{sch}))))\sqsupset$$
$$\bigcirc^3(\bigcirc(l_1=3\wedge w_1=1\wedge x_1=3\wedge z_1=2\wedge t_1=0\wedge((W_t(1,2),U_s(1))^{\oplus},A_1\wedge E_1\wedge\varepsilon)\text{prj }S_{sch})\parallel\bigcirc(w_2=0\wedge\bigcirc(E_2\wedge\Box(w_2=-1\wedge\bar{\varepsilon})))\parallel\bigcirc(l_3=3\wedge w_3=1\wedge x_3=2\wedge z_3=1\wedge t_3=1\wedge\bigcirc(w_3=0\wedge u_3=1)\wedge\bigcirc^2(w_3=3\wedge l_3=3\wedge(A_p(3,3,1),\cdots,A_3\wedge E_3\wedge\varepsilon)\text{ prj }S_{sch})))\sqsupset$$
$$\bigcirc^5(\bigcirc(w_1=0\wedge\bigcirc^2E_1)\parallel(E_2\wedge\bigcirc(w_2=-1))\parallel\bigcirc(w_3=3\wedge l_3=3\wedge(A_p(3,3,1),\cdots)\text{ prj }S_{sch}))\sqsupset$$

$$\bigcirc^5(\bigcirc^3 E_1 \parallel E_2 \parallel \bigcirc^2 E_3) \sqsupset \Diamond(E_2;\bigcirc E_3;\bigcirc E_1) \tag{4}$$

同理可证

$$(u_1 = 4 \wedge u_2 = 2 \wedge u_3 = 1 \wedge S_{sys}) \sqsupset \Diamond(E_2;\bigcirc E_3;\bigcirc E_1) \tag{5}$$

由式(2)～(4)，公理 AT、AXA，以及定理 DICI，有

$$S_{sys} \sqsupset \Diamond(E_2;\bigcirc E_3;\bigcirc E_1)$$

3.3　本文方法与模型检测的比较

在多内核进程调度的验证上，亦可选用模型检测方法. 模型检测的最大优点是验证过程完全自动完成，但缺点是受状态空间爆炸问题所限，不能适用于无穷状态系统. 另外，验证人员需要同时熟练掌握 2 种不同的形式化工具，即描述系统模型的有穷状态系统的建模语言和描述系统性质的时序逻辑，这一点对验证人员要求颇高，影响了模型检测技术的推广和应用.

本文提出的方法则充分利用了 PTL 表达能力强的优点，在同一 PTL 逻辑框架内完成了多内核进程调度的建模和性质的描述，并在 PTL 公理系统的基础上完成了最终性质的证明. 该方法的最大优点是有穷及无穷状态系统均可适用，虽然验证过程不能完全由计算机自动完成，但一些定理证明器(如 PVS)可对验证过程提供半自动化的支持，大大提高了验证的效率.

4　结　论

投影时序逻辑的投影操作符可以方便地描述并发系统. 本文使用 PTL 公式对多核处理器的进程调度进行了建模和性质描述，并基于 PTL 公理系统完成了系统的性质验证. 由于多核处理器的进程调度实质上是并发进程在进程调度器控制下的协调一致运行，具备了并发系统的主要特点，因此本文提出的方法也适用于一般并发系统的验证.

参考文献：

[1] DUAN Zhenhua. Temporal logic and temporal logic programming [M]. Beijing: Science Press, 2006: 35-105.

[2] MOSZKOWSKI B. Executing temporal logic programs [D]. Cambridge, UK: Cambridge University Press, 1986.

[3] TIAN Cong, DUAN Zhenhua. Complexity of propositional projection temporal logic with star [J]. Mathematical Structures in Computer Science, 2009, 19(1): 73-100.

[4] DUAN Zhenhua, ZHANG Nan. A complete axiomatization of propositional projection temporal logic [C]// Proceedings of 2nd IEEE International Symposium on Theoretical Aspects of Software Engineering. Los Alamitos, CA, USA: IEEE Computer Society, 2008: 271-278.

[5] 雷丽晖，段振华. 基于扩展投影时序逻辑的组合 Web 服务描述与验证[J]. 西安交通大学学报，2007，41(10)：1155-1159.
LEI Lihui, DUAN Zhenhua. Specification and verification of composite Web services based on extended projection temporal logic [J]. Journal of Xi′an Jiaotong University, 2007, 41(10): 1155-1159.

[6] 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.
LEI Lihui, DUAN Zhenhua. Modelling concurrent workflow with the extended interval temporal logic [J]. Journal of Xidian University, 2007, 34(4): 673-680.

[7] 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)：41-43.
WANG Xiaobing, DUAN Zhenhua. Projection temporal logic oriented model checking for Web services [J]. Journal of Xi′an Jiaotong University, 2009, 43(4): 41-43.

[8] 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.
ZHANG Haibin, DUAN Zhenhua. Hybrid projection temporal logic and formal verifications of hybrid systems [J]. Computer Science, 2007, 34(11): 279-282.

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

[10] DUAN Zhenhua, TIAN Cong. A unified model checking approach with projection temporal logic [C]// LNCS 5256: Proceedings of 10th International Conference on Formal Engineering Methods. Berlin, Germany: Springer, 2008: 167-186.

[11] 舒新峰，段振华. 投影时序逻辑的公理系统与形式验证[J]. 西安电子科技大学学报，2009，36(4)：680-685.
SHU Xinfeng, DUAN Zhenhua. Axiomatization for the first-order projection temporal logic and formal verifications [J]. Journal of Xidian University, 2009, 36(4): 680-685.

(编辑　葛赵青　苗凌)