软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(2):211−221 [doi: 10.3724/SP.J.1001.2011.03729]

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

基于命题投影时序逻辑的单调速率调度算法模型检测*

田 聪[1,2], 段振华[1,2+]

[1](西安电子科技大学 计算理论与技术研究所,陕西 西安 710071)

[2](西安电子科技大学 综合业务网理论及关键技术国家重点实验室,陕西 西安 710071)

Model Checking Rate Monotonic Scheduling Algorithm Based on Propositional Projection Temporal Logic

TIAN Cong[1,2], DUAN Zhen-Hua[1,2+]

[1](Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China)

[2](State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China)

+ Corresponding author: E-mail: zhhduan@mail.xidian.edu.cn

Tian C, Duan ZH. Model checking rate monotonic scheduling algorithm based on propositional projection temporal logic. *Journal of Software*, 2011,22(2):211−221. http://www.jos.org.cn/1000-9825/3729.htm

Abstract: A propositional projection temporal logic (PPTL) based model checking approach for rate monotonic scheduling (RMS) is presented. With this approach, RMS controlled systems are modeled by PROMELA, which is the system modeling language in model checker SPIN. The desired property is specified by a PPTL formula. Next, whether or not the system satisfies the property can be verified with SPIN. Accordingly, the schedulability of a group of tasks can be obtained; meanwhile, other properties of the tasks scheduling system under RMS algorithm can also be verified.

Key words: temporal logic; model checking; rate monotonic scheduling algorithm; verification; real time system

摘 要: 提出了基于命题投影时序逻辑(propositional projection temporal logic,简称 PPTL)的单调速率调度(rate monotonic scheduling,简称 RMS)模型检测方法.该方法使用 SPIN 模型检测器的系统建模语言 PROMELA 为任务调度系统建模,使用 PPTL 描述系统期望的性质,通过 SPIN 验证系统模型是否满足性质,从而得知一个任务组在 RMS 下是否可调度.同时,RMS 算法控制下的任务调度系统的其他性质也可以得到验证.

关键词: 时序逻辑;模型检测;单调速率调度算法;验证;实时系统

中图法分类号: TP311 **文献标识码**: A

单调速率调度(rate monotonic scheduling,简称 RMS)算法是一种经典的周期性任务调度算法,它是 Liu 和 Layland 于 1973 年提出来的[1].RMS 是静态优先级调度算法.在所有的静态优先级调度算法中,RMS 是最优的,即如果一组任务在 RMS 算法下是不可调度的,那么不存在其他任何静态调度算法使得该组任务可调度.RMS 算法在实时系统调度的实现中应用非常广泛,因此,在实时系统中,在使用 RMS 算法之前,对 RMS 的性质,特别是

* 基金项目: 国家自然科学基金(61003078, 91018010, 60873018, 60910004); 国家重点基础研究发展计划(973)(2010CB328102); 教育部博士点基金(200807010012); 中央高校基本科研业务费专项资金(JY10000903004)

收稿时间: 2008-12-24; 修改时间: 2009-06-09; 定稿时间: 2009-08-28

可调度性进行验证和分析,可以预测到哪些任务将延误时限,从而调整任务的优先级,甚至改变任务的调度策略,使得所有的任务都能满足时限要求[2,3].另外,通过对 RMS 的其他性质的验证和分析,可以保证 RMS 的正确性.特别地,RMS 算法一般针对并发性实时系统,并发使得系统易错,而实时性又对系统执行有着更高的要求.因此,对 RMS 进行的验证分析是重要的.

模型检测[4]是基于时序逻辑的验证方法.与定理证明相比,该技术可以被完全自动化.模型检测是将要验证的系统表示成有限状态机,如 Kripke 结构[5]、迁移系统或者自动机;将系统期望的性质用时序逻辑公式描述;然后,遍历有限状态机以检验性质是否满足.当断定某性质不满足时,模型检验能提供反例,以便于定位设计错误.模型检测方法的优点是,它是全自动的,故能用于比较大的系统的形式验证.模型检测技术的成功在很大程度上归功于优秀的模型检测工具的开发.其中,SPIN[4]是由贝尔实验室开发的基于命题线性时序逻辑(propositional linear temporal logic,简称 PLTL)[6]的模型检测工具.SPIN 良好的算法设计和非凡的检测能力得到了 ACM 的认可,并于 2001 年被评为 ACM 优秀软件系统奖.然而,PLTL 描述能力的有限性使得一些复杂的性质不能够被验证[7].特别地,PLTL 描述能力的有限性(PLTL 的描述能力等价于 star-free 正则语言)使得组合模型检测无法用 PLTL 来实现[8].而组合模型检测是当前克服模型检测技术中状态空间爆炸问题的重要技术之一.另外,PLTL 不方便对一些实时性相关性质进行描述.如"p 在第 6 个时钟到来时成立",用 PLTL 公式描述该性质需要使用 6 个 *next* 操作,$OOOOOOp$;若要描述"p 在第 15 个和第 20 个时钟之间成立"就更加复杂了[8,9].

基于 SPIN 的命题投影时序逻辑(propositional projection temporal logic,简称 PPTL)[10−13]模型检测是在 SPIN 的基础上扩充它的功能,使得 SPIN 能够支持 PPTL 的验证[14].PPTL 是一种基于区间的时序逻辑,它的表达能力等价于 full 正则语言[15].另外,PPTL 能够方便地描述实时性相关的性质,如"p 在第 6 个时钟到来时成立"可以简单地描述为 $len(6);p;true$,"p 在第 15 个和第 20 个时钟之间成立"可以表达为 $len(15);\Diamond p\wedge len(5);true$.命题投影时序逻辑强大的描述能力以及自身的语法结构特点,使得该逻辑适合被用作模型检测中的性质描述语言.

基于 SPIN 和 PPTL 的特点,本文研究基于 PPTL 的 RMS 模型检测方法.该方法使用 SPIN 模型检测器的系统建模语言 PROMELA 为任务调度系统建模,使用命题投影时序逻辑描述系统期望满足的性质,通过 SPIN 检测系统模型是否满足性质.当系统不满足期望的性质时,SPIN 给出反例.通过模型检测方法,可以验证 RMS 算法下的任务调度系统的性质,特别是可调度性和自治并发性.关于 RMS 算法下任务的可调度性研究已有很多研究[2],其中最著名的是 Liu 和 Layland 的定理[1].但是,该定理只是 RMS 可调度性的一个充分不必要条件.在 Liu 和 Layland 工作的基础上,Lehoczky 等人对静态优先级算法的特征进行了更精确的研究,并提出了 RMS 算法可调度行判定的充要条件[16],但是算法的复杂度较高.本文提出的基于 PPTL 的 RMS 算法模型检测方法可以用来判定 RMS 的可调度性.与传统的 RMS 可调度性理论研究不同:(1) 模型检测方法可以完全自动化;(2) 在任务不可调度的情况下,模型检测方法还可以给出不可调度的实例模拟,便于查找问题、调整方案;(3) 除了可调度性以外,模型检测方法还可以验证其他的性质.

本文第 1 节简单地介绍命题投影时序逻辑的语法和语义.第 2 节介绍基于命题投影时序逻辑的模型检测.第 3 节给出基于 RMS 算法的任务调度系统的建模和验证方法.第 4 节通过具体的验证实例来展示基于 PPTL 的 RMS 算法的模型检测.第 5 节是对本文的总结和未来工作的展望.

1 命题投影时序逻辑

命题投影时序逻辑是命题区间时序逻辑(propositional interval temporal logic,简称 PILT)[17]的一种扩展,它将 PITL 扩展到无穷区间,并引入了新的时序操作符、投影[10,13]和投影星[12].

用 *Prop* 表示一个可数的原子命题的集合,N_0 表示大于 0 的正整数的集合,N_ω 表示 $N_0\cup\{\omega\}$.一般地,我们用小写字母(或者带下标)p,q 或 r 等表示原子命题,用大写字母(或者带下标)P,Q 或 R 表示其他一般的 PPTL 公式.命题投影时序逻辑公式归纳定义如下:

$$P::=p|\neg P|P\vee P|OP|(P_1,...,P_m)\ prj\ Q|(P_1,...,(P_i,...,P_s)^{\otimes},...,P_m)\ prj\ Q,$$

其中,O,*prj* 和 $prj^{\otimes}$是时序操作符,$\neg$和$\vee$与经典逻辑中的定义相同.

状态 s 定义为从 $Prop$ 到 $B=\{true,false\}$的映射,$s:Prop\to B$.p 在状态 s 的值用 $s[p]$来表示,$s[p]=true$ 表示 p 在状态 s 为真;相反,$s[p]=false$ 表示 p 在状态 s 为假.区间定义为状态的序列$\sigma=\langle s_0,s_1,...,s_i\rangle$.区间可以是有穷的,也可以是无穷的.对于有穷区间,区间长度$|\sigma|$等于状态数减 1;对于无穷区间,$|\sigma|=\omega$,其中,状态 s_ω没有定义.操作•用来将两个区间连接成一个大区间.$\sigma_{(i..j)}$表示σ的子区间$\langle s_i,...,s_j\rangle$.

为了定义投影操作的语义,需要一个辅助操作符↓.设$\sigma=\langle s_0,s_1,...\rangle$是一个区间,$r_1,...,r_h(h\geqslant 1)$是整数且有 $0\leqslant r_1\leqslant\ldots\leqslant r_h<<|\sigma|$.其中,$<<$定义为$\leqslant-\{\omega,\omega\}$,表示≤中去除$\omega=\omega$.$\sigma$在 $r_1,\ldots,r_h$ 上的投影定义为$\sigma\downarrow(r_1,\ldots,r_h)=\langle s_{t_1},s_{t_2},...s_{t_l}\rangle$,其中,$t_1,...,t_l$是通过删除 $r_1,...,r_h$ 中的冗余元素得到的,即 $t_1,...,t_l$ 是 $r_1,...,r_h$ 的严格递增子序列.例如,

$$\langle s_0,s_1,s_2,s_3,s_4\rangle\downarrow(0,0,2,2,2,3)=\langle s_0,s_2,s_3\rangle.$$

另外,为了定义投影星操作,我们扩展$\sigma\downarrow(r_1,...,r_h)$,使得 r_h 可以为ω.对于任意一个区间$\sigma=\langle s_0,s_1,...,s_{|\sigma|}\rangle$,

- $\sigma\downarrow()=empty$;
- $\sigma\downarrow(r_1,...,r_{h-1},r_h)=\sigma\downarrow(r_1,...,r_{h-1},r_h)$,如果 r_h 不是ω;
- $\sigma\downarrow(r_1,...,r_{h-1},r_h)=\sigma\downarrow(r_1,...,r_{h-1})$,如果 r_h 是ω.

解释 $I=(\sigma,k,j)$为一个三元组,其中,σ表示一个区间,k 为整数,j 为整数或者ω,$0\leqslant i<<j\leqslant|\sigma|$.解释 I 能够满足公式 P,表示为$(\sigma,k,j)\vDash P$.命题投影时序逻辑的解释归纳定义为:

- $I\vDash p$ 当且仅当 $s[p]=true$;
- $I\vDash\neg P$ 当且仅当 $I\vDash P$ 不成立;
- $I\vDash P_1\vee P_2$ 当且仅当 $I\vDash P_1$ 或者 $I\vDash P_2$;
- $I\vDash OP$ 当且仅当 $k<j$ 并且$(\sigma,k+1,j)\vDash P$;
- $I\vDash(P_1,...,P_m)\ prj\ Q$,如果存在整数 $k\leqslant r_1\leqslant\ldots\leqslant r_m\leqslant j$ 使$(\sigma,r_0,r_1)\vDash P_1,(\sigma,r_{l-1},r_l)\vDash P_l(1<l\leqslant m)$,并且对下面的某种情况有$(\sigma',0,|\sigma'|)\vDash Q$:
  (1) $r_m<j$ 并且 $\sigma'=\sigma\downarrow(r_0,...,r_m)\bullet\sigma_{(r_m+1..j)}$;
  (2) $r_m=j$ 并且对某个 $0\leqslant h\leqslant m$ 有$\sigma'=\sigma\downarrow(r_0,...,r_m)$;
- $I\vDash(P_1,...,(P_i,...,P_s)^{\otimes},...,P_m)\ prj\ Q$ 当且仅当下面的某种情况成立:
  (1) $1\leqslant i\leqslant s\leqslant m$ 并且$\exists n\in N_0,I\vDash(P_1,...,(P_i,...,P_s)^{(n)},...,P_m)\ prj\ Q$;
  (2) $s=m$,并且存在 $k\leqslant r_1\leqslant\ldots\leqslant r_k<<\omega,\lim\limits_{k\to\infty}r_k=\omega$,使得$(\sigma,r_{l-1},r_l)\vDash P_l(0<l<i),(\sigma,r_{l-1},r_l)\vDash P_t(l\geqslant i),t=i+((l-i)\bmod(s-i+1))$,并且$(\sigma',0,|\sigma'|)\vDash Q$.其中,$\sigma'=\sigma\downarrow(r_1,...,r_h),r_h\in N_\omega$.

直观来说,OP 表示 P 在下一状态成立.投影操作允许用不同时间粒度刻画计算进程.为了解释$(P_1,...,P_m)\ prj\ Q$,需要两个不同时间粒度的状态序列:一个是执行 $P_1,...,P_m$ 的局部序列,另一个是并行执行 Q 的全局序列.直观来说,Q 与 $P_1,...,P_m$ 在一个区间上并行执行,且 Q 的区间状态仅仅是 $P_1,...,P_m$ 各自区间的初始状态和终止状态,如图 1 所示.投影操作允许 Q 与 $P_1,...,P_m$ 各自独立,都有权来定义自身的执行区间.投影星中允许 $P_1,...,P_m$ 中的部分或全部任意次的重复.

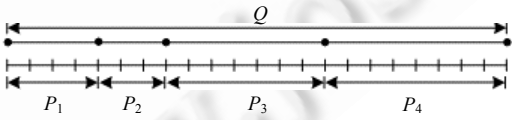

Fig.1 Semantics of $(P_1,P_2,P_3,P_4)\ prj\ Q$

图 1 $(P_1,P_2,P_3,P_4)\ prj\ Q$ 的语义

以下是 PPTL 的一些导出公式:

$$empty \overset{def}{=} \neg O\,true \qquad more \overset{def}{=} \neg empty \qquad O^0 P \overset{def}{=} P \qquad O^n P \overset{def}{=} O(O^{n-1}P),\ n \geqslant 1$$

$$len(0) \overset{def}{=} empty \qquad len(n) \overset{def}{=} O^n\ empty, n \geqslant 1 \qquad skip \overset{def}{=} len(1) \qquad \odot P \overset{def}{=} empty \vee OP$$

$$P;Q \overset{def}{=} (P,Q)\ prj\ empty \qquad \Diamond P \overset{def}{=} true;P \qquad \Box P \overset{def}{=} \neg\Diamond\neg P \qquad P^0 \overset{def}{=} empty$$

$$P^n \overset{def}{=} P;\ P^{n-1} \qquad (P_1,...,P_{i-1},(P_i,...,P_s)^{(0)},P_{s+1},...,P_m)\ prj\ Q \overset{def}{=} (P_1,...,P_{i-1},P_{s+1},...,P_m)\ prj\ Q$$

$$(P_1,...,P_{i-1},(P_i,...,P_s)^{(n)},P_{s+1},...,P_m)\ prj\ Q \overset{def}{=} (P_1,...,P_{i-1},(P_i,...,P_s),(P_i,...,P_s)^{(n-1)},P_{s+1},...,P_m)\ prj\ Q,\ n \geqslant 1$$

$$(P_1,...,P_{i-1},(P_i,...,P_s)^{(\oplus)},P_{s+1},...,P_m)\ prj\ Q \overset{def}{=} (P_1,...,P_{i-1},P_{s+1},...,P_m)\ prj\ Q \vee (P_1,...,P_{i-1},(P_i,...,P_s)^{(\otimes)},P_{s+1},...,P_m)\ prj\ Q$$

$$P^* \overset{def}{=} (P^{\otimes})\ prj\ empty \qquad P^+ \overset{def}{=} (P^{\oplus})\ prj\ empty$$

公式 P 在区间σ上是可满足的,记作$\sigma \vDash P$,如果$(\sigma,k,j) \vDash P$;公式 P 是可满足的,如果存在σ使得$\sigma \vDash P$;公式 P 是有效的,如果对于任意的σ,$\sigma \vDash P$.

2 基于 SPIN 的 PPTL 模型检测

模型检测一般使用 Kripke 结构、状态迁移系统或者自动机为要验证的系统建立模型 M,采用时序逻辑公式φ刻画系统期望的性质,通过证明 $M \vDash \varphi$?来验证系统模型是否满足性质.SPIN 是基于自动机和 PLTL 的模型检测工具.在 SPIN 中,系统模型通过 PROMELA 来描述,系统性质使用命题线性时序逻辑公式来表达[6].SPIN 通过 PROMELA 语言的解释器执行系统的 PROMELA 模型,并将系统的迁移关系以 Büchi 自动机的形式来表达.SPIN 系统中嵌有一个将 PLTL 公式转换成 Büchi 自动机,并以 PROMELA 语法表达为 Never Claim 的工具.通过向 SPIN 的验证器输入 PROMELA 模型和性质的 Never Claim,SPIN 验证器将自动检查系统模型是否满足性质.若不满足,则给出反例.

我们研究了从 PPTL 公式到 Büchi 自动机的转换算法[14].该算法通过正则形将公式展开为一个正则图,然后将正则图等价地转换为 Büchi 自动机.在此基础上,形成了基于 Büchi 自动机的 PPTL 公式模型检测算法[14].算法基本思想为,用一个 Büchi 自动机 A_{sys} 来表示一个系统模型,用 PPTL 公式 P 刻画系统应该满足的性质,将$\neg P$等价地转换成一个 Büchi 自动机 $A_{\neg P}$,再求 A_{sys} 和 $A_{\neg P}$ 的积自动机 $A_{sys\times\neg P}$.然后判断 $A_{sys\times\neg P}$ 接收的语言是否为空:如果为空,则系统满足性质;否则不满足,且 $A_{sys\times\neg P}$ 所接受的字体现了错误的反例,如图 2 所示.

为了基于 SPIN 实现 PPTL 的模型检测,我们开发了从 PPTL 公式到 Büchi 自动机,并以 PROMELA 语法表达为 Never Claim 的工具,使得 SPIN 可以支持 PPTL 公式的验证,如图 3 所示.

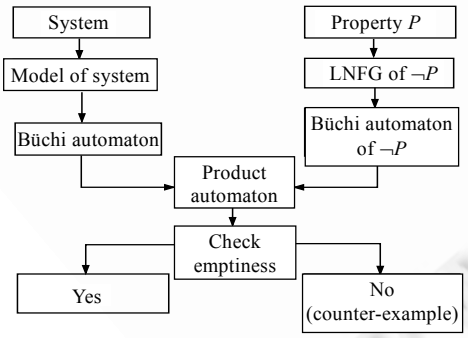

Fig.2 Model checking PPTL

图 2 命题投影时序逻辑模型检测

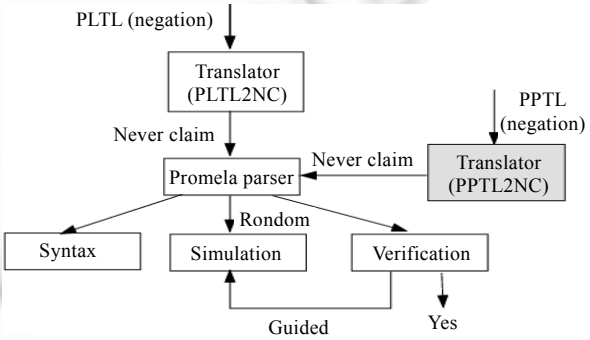

Fig.3 Model checking PPTL based on SPIN

图 3 基于 SPIN 的 PPTL 模型检测

基于 SPIN 的 PPTL 模型检测有以下优点:(1) 提高了 SPIN 的验证能力.原 SPIN 所支持的性质描述语言 PLTL 的描述能力有限,仅等价于 star-free 正则语言,而 PPTL 的描述能力等价于 full 正则语言[15];(2) 使得 SPIN

可以支持实时性相关性质的验证.具有实时特点的性质用 PLTL 描述非常复杂;(3) 投影和投影星操作可以描述一些复杂的并发情况,如自治并发等,这是其他时序逻辑都无法完成的.

3 单调速率调度算法模型检测

3.1 单调速率调度算法

单调速率调度算法是针对一组周期性任务的静态调度算法.以 $T=\{t_1,t_2,...,t_n\}$表示一组含有 n 个周期性任务的集合,每个任务定义为一个四元组 $t_i=(T_i,D_i,E_i,P_i)$,其中,T_i 表示任务 t_i 的周期,D_i 表示 t_i 的截至期限,E_i 表示 t_i 所需的执行时间,P_i 表示任务 t_i 的优先级.一般地,$D_i=T_i$,$P_i>P_j$,如果 $E_i<E_j$.因此,任务的定义可以被简化为一个二元组 $t_i=(T_i,E_i)$.对于任务 t_i,CPU 的利用率定义为 $U_i=E_i/T_i$,任务集 T 的 CPU 利用率定义为 $U=\sum_{i=1}^{n}U_i$.

Liu 和 Layland[1]证明了,对于任意的任务集 T,如果 $\sum_{i=1}^{n}U_i \leqslant n(2^{1/n}-1)$, 则 T 在 RMS 算法下是可调度的.这一条件是充分不必要的,因此存在任务集,它的可调度性用 Liu 和 Layland 的定理是不可判定的.在 Liu 和 Layland 之后,提出了很多改进算法,其中最具有代表性的是 Lehoczky 在 Liu 和 Layland 工作的基础上提出的 RMS 算法可调度行判定的充要条件.但是,已存在的判定算法都比较复杂.本节我们利用模型检测方法来判定一组任务在 RMS 算法下的可调度性;在不可调度的情况下,该方法给出不可调度的实例模拟,从而方便方案的调整.在检测可调度性的同时,还可以验证 RMS 调度系统的其他一些性质,以保证系统实现的正确性.

3.2 单调速率调度算法的建模

为了使用基于 SPIN 的 PPTL 模型检测方法对 RMS 算法进行验证,我们使用 SPIN 的系统建模语言 PROMELA 为系统建模,对 PROMELA 语言的介绍参见文献[18,19].下面我们定义必要的数据结构.一个任务定义为一个结构体:

$$typedef\ task\ \{byte\ T;\ byte\ E\},$$

因此,一组任务可以定义为

$$task\ t_1,t_2,t_3,\ldots,t_n.$$

我们通过定义一个全局的 *int* 型变量 *clock* 来定义时间序列.*clock*=0 表示第 0 个时刻,*clock*=*clock*+1 表示时钟跳转到下一个时刻,两个不同时刻的时钟差表示经历的时间长度.另外,通过全局变量 e_i 表示任务 t_i 的剩余执行时间.任务的周期性活动用以下方法实现:

do

::$clock\%t_i.T==0\rightarrow e_i=t_i.E$

od

其中,$clock\%t_i.T==0$ 意味着任务 t_i 的周期已到,$e_i=t_i.E$ 将任务 t_i 的剩余执行时间初始化为完成该任务所需的时间.假设任务的优先级为 $P_1>P_2>...>P_n$,按照优先级,对任务的调度和执行可描述为:

do

::$e_1==0$ &&...&& $e_{i-1}==0$ && $e_i!=0$ && $(clock+1)\%t_i.T!=0\rightarrow atomic\{clock=clock+1;e_i=e_i-1\}$

od

其含义是:在任务 t_i 已执行完毕且任务 t_{i+1} 还未执行完的时候,如果下一时刻 t_{i+1} 的周期没有到来,那么在下一时刻 t_{i+1} 的剩余执行时间减 1.当所有任务都完成时,在时钟跳转到下一时刻不做任何操作.

do

::$e_1==0$ &&...&& $e_{n-1}==0$ && $e_n==0\rightarrow atomic\{clock=clock+1\}$

od

因此,任务 t_i 周期性请求执行可以定义为进程 *proctype Request_i*.

proctype Request_i(){
do
::*clock*%t_i.*T*==0→e_i=t_i.*E*
od
}

任务的调度及处理可以定义为进程 *proctype Schedule*.

proctype Schedule(){
do
::e_1!=0 && (*clock*+1)%t_1.*T*!=0→*atomic*{*clock*=*clock*+1;e_1=e_1−1}
::e_1==0 && e_1!=0 && (*clock*+1)%t_1.*T*!=0→*atomic*{*clock*=*clock*+1;e_1=e_1−1}
…
::e_1==0 &&...e_{i-1}==0 && e_i!=0 && (*clock*+1)%t_i.*T*!=0→*atomic*{*clock*=*clock*+1;e_i=e_i−1}
::e_1==0 &&...&& e_{n-1}==0 && e_n==0→*atomic*{*clock*=*clock*+1}
od
}

进程之间的通信通过共享全局变量 $e_1,e_2,\ldots,e_n$ 来实现.

SPIN 的 PROMELA 解释器为每个进程 *proctype Request_i* 和 *proctype Schedule* 生成一个自动机,如图 4 所示.整个系统的状态空间为所有进程自动机的积自动机.

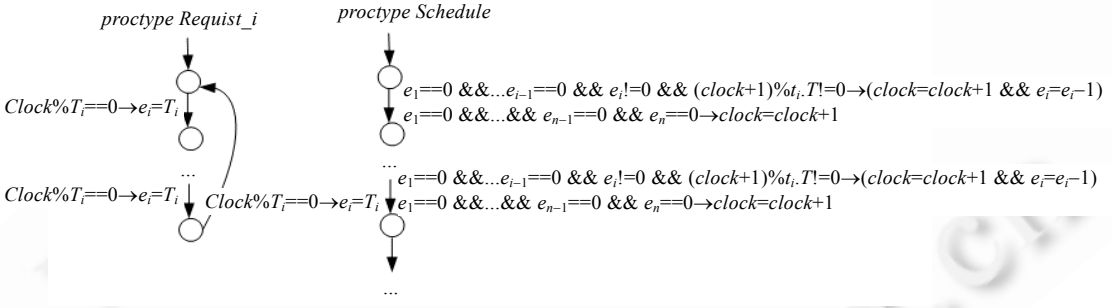

Fig.4 Automata of processes

图 4 进程自动机

3.3 单调速率调度算法的性质

单调速率调度算法是实现实时系统调用的重要方法,一个单调速率调度算法具有以下的特点:

(1) 互斥性:单调速率调度中,任何时刻最多只有 1 个任务被执行.

(2) 实时性:每个任务必须在下一个周期到来之前完成,即在该任务的周期到来前,该任务的剩余时间为 0,或者为 1 且该任务正在被执行.

(3) 静态性:任务的优先级是固定的,不随时间的变化而变化.

(4) 优先性:在任何时刻,被执行的任务都是当前没有完成的任务中优先级最高的任务.

(5) 自治并发性:任务组中的每个任务的活动周期都是自治的,不受任何其他任务的影响.

从以上分析可以看出,单调速率调度算法是实时性相关算法,用经典的命题线性时序逻辑和计算树逻辑[9]无法完整地描述.而基于区间的时序逻辑,如命题投影时序逻辑,适合描述时间相关的性质.另外,命题投影时序逻辑中的投影和投影星操作具有描述自治并发的能力.为了用 PPTL 描述 RMS 算法下任务调度系统的性质,我们给出以下命题的定义:

define c_i e_i==0 /* 任务 t_i 已完成 */

 # define d_i e_i==t_i.E /* 任务 t_i 的剩余执行时间被重置 */

 # define m P_1<P_2<...<P_n /* 任务的优先级为 P_1<P_2<...<P_n */

另外,为了描述任务 t_j 正在被执行,需另外定义一个布尔型变量 f_j,f_j=1 表示任务 t_j 正在被执行;相反,f_j=0 表示当前时刻任务 t_j 没有被执行.当 e_j!=0 且对于任何其他任务 e_i,i<j,e_i==0 时,f_j=1;否则,f_j=0.因此,单调速率调度算法的性质可以描述为以下命题投影时序逻辑公式:

(1) 互斥性: $\Box\neg(f_i \wedge f_j), i \neq j$.在任何时刻,两个不同的任务 t_i 和 t_j 都不能被同时执行.

(2) 实时性: $(len(T_i-1)^{\otimes})\ prj\ \left(\Box(c_i \vee \neg c_i \wedge \bigwedge_{j=1}^{j<i} c_j)\right)$.在任何时刻,当任务 t_i 的周期在下一时刻到来时,该任务的剩余时间为 0,或者为 1 且该任务正在被执行.

(3) 静态性:$\Box m$.在任何时刻,各个任务的优先级都为 m.

(4) 优先性: $\Box\left(f_j \rightarrow \bigwedge_{i=1}^{i<j} c_i\right)$.在任何时刻,若任务 t_j 正在被执行,比 t_j 优先级高的任何任务都已完成.

(5) 自治并发性: $\bigwedge_{i=1}^{n}((len(T_i)^{\otimes})\ prj\ (\Box d_i)$. 每个任务的活动是周期的、自治的、并发的.

其中,实时性体现了任务的可调度性.如果对于任务组中的任意任务,性质(2)能够满足,则该任务组在 RMS 算法下是可调度的;反之则是不可调度的.在不可调度的情况下,模型检测的结果能够给出错误情况的模拟,便于任务优先级的调整.

4 RMS 模型检测实例

本节我们采用基于 SPIN 的命题投影时序逻辑模型检测来验证给定的一组任务是否是单调速率调度算法可调度的,并验证其他性质是否可满足.给定一组任务 T={t_1,t_2,t_3}如下:

	t_1	t_2	t_3
T_i	8	9	14
E_i	3	3	3

我们假设任务的优先级为 P_1>P_2>P_3,在单调速度调度算法下,该任务组的执行情况如图 5 所示.

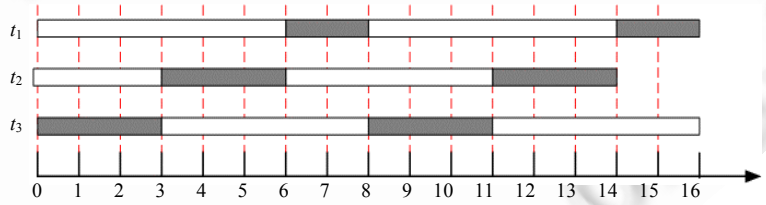

Fig.5 Execution of T under RMS algorithm

图 5 RMS 算法下 T 的执行情况

根据第 3 节给出的建模方法,为 RMS 算法下 T 的调度建立 PROMELA 模型(完整的代码见附录).用命题投影时序逻辑描述性质如下:

(1) $\Box\neg(f_1 \wedge f_2) \wedge \Box\neg(f_1 \wedge f_3) \wedge \Box\neg(f_2 \wedge f_3)$;

(2) $(len(T_1-1)^{\otimes})\ prj\ (\Box c_1)$, $(len(T_2-1)^{\otimes})\ prj\ (\Box(c_2 \vee \neg c_2 \wedge c_1))$, $(len(T_3-1)^{\otimes})\ prj\ (\Box(c_3 \vee \neg c_3 \wedge c_1 \wedge c_2))$;

(3) $\Box m$;

(4) $\Box(f_2 \rightarrow c_1) \wedge \Box(f_3 \rightarrow c_1 \wedge c_2)$;

(5) $((len(T_1)^{\otimes})\ prj\ \Box d_i \wedge ((len(T_2)^{\otimes})\ prj\ \Box d_2 \wedge ((len(T_3)^{\otimes})\ prj\ \Box d_3$.

我们仅通过验证性质(2)来验证任务的可调度性,其他性质的验证方法类似.对于任务 t_1 和 t_2,在验证 $(len(T_1-1)^{\otimes})\ prj\ (\Box c_1)$ 和 $(len(T_2-1)^{\otimes})\ prj\ (\Box(c_2 \vee \neg c_2 \wedge c_1))$ 时,SPIN 提示没有错误,如图 6 所示.因此,任务 t_1 和 t_2

在 RMS 算法下可以满足时限要求.对于任务 t_3,在验证性质 $(len(T_3-1)^{\otimes})\ prj\ (\square(c_3\vee\neg c_3\wedge c_1\wedge c_2))$ 时,系统提示有错误,并给出了错误的信息,如图 7 所示.根据提示的错误,在第 *clock*=13 时,e_3!=0,且 t_2 正在执行.因此在 *clock*=14 时,即 t_3 的周期到达时,t_3 无法完成.

Fig.6 Verification result 1

图 6 验证结果 1

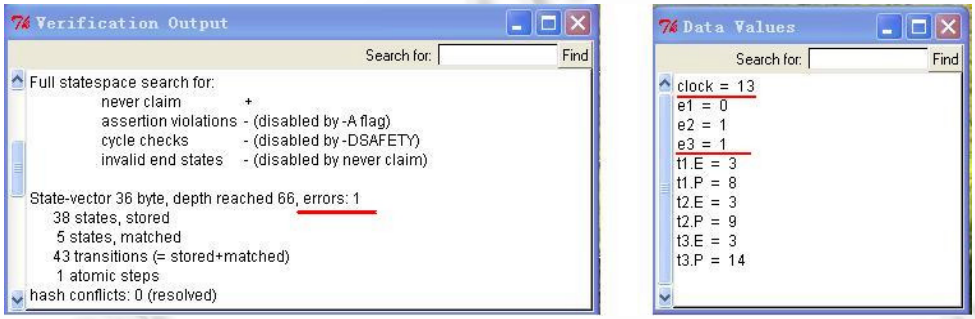

Fig.7 Verification result 2

图 7 验证结果 2

SPIN 系统给出错误路径的模拟如图 8 所示.

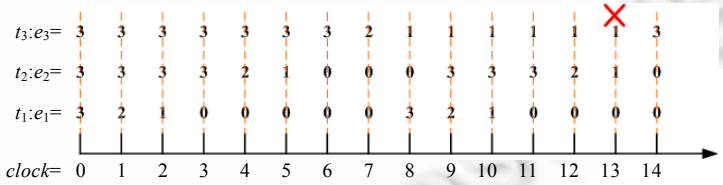

Fig.8 Simulation result

图 8 模拟结果

因此,在任务组 T 中,虽然任务 t_1 和 t_2 可以被正常调度,但是任务 t_3 不能在规定的时限内完成.因此,T 在 RMS 算法下是不可调度的.根据 SPIN 检测出的错误路径,可以调整任务的优先级,甚至寻找其他调度算法,以确保 T 中所有任务都能够在规定的时限内完成.

5 与 UPPAAL 的比较

RMS 是典型的实时系统,传统的方法是通过实时系统模型检测工具 UPPAAL 来验证.与 UPPAAL 相比,基

于 PPTL 的模型检测具有以下优点:

(1) PPTL 模型检测采用的性质规范语言 PPTL 具有 full 正则表达能力[15],并且能够描述顺序、循环等性质;而 UPPAAL 采用的性质描述语言是带时间的 CTL.CTL 的表达能力是有限的,无法描述典型的性质“*Even*(*p*):*p* 在任意的偶数状态为真”,也无法描述顺序和循环等性质.Clark 曾提出[9],当前的模型检测工具对 RMS 的验证是有限的,这些验证只能局限在传统的任务执行时间限制上.因此,本文提出的基于 PPTL 的 RMS 模型检测弥补了采用传统的模型检测工具 UPPPAL 验证的不足之处.

(2) 采用 PPTL 的模型检测是基于命题时序逻辑和自动机的验证,而基于 UPPAAL 的验证是基于带时间的时序逻辑(TCTL)和时间自动机的验证.从理论上来说,实时系统模型检测的过程要复杂一些.

(3) PPTL 中的投影和投影星操作可用来描述一些特殊的并发情形,如自治并发.这是其他时序逻辑及其各种扩展所无法表述的.因此,本文第 4 节中的性质(5)是无法用 UPPAAL 来验证的.

6 结 论

本文提出了基于命题投影时序逻辑的单调速率调度算法模型检测方法.该方法可以用来验证 RMS 算法下任务的可调度性以及其他性质.与传统的 RMS 可调度性判定算法相比较,该方法的优点在于,检测过程是自动化的;在不可调度的情况下,可以得到不可调度的具体信息,便于方案的调整;另外,与 PLTL 相比,采用 PPTL 对系统性质的描述更完整.与采用实时模型检测方法相比,使用命题逻辑模型检测方法更简洁.在今后的工作中,我们将继续完善基于 SPIN 的 PPTL 模型检测工具,使其成为一个实践中有用的验证工具.另外,我们将继续研究基于 PPTL 的符号化、定界以及组合模型检测,从而降低验证过程中状态空间的大小.

References:

[1] Liu CL, Layland JW. Scheduling algorithm for multiprogramming in a hard real-time environment. Journal of the ACM, 1973, 20(1):46−61. [doi: 10.1145/321738.321743]

[2] Wang YJ, Chen QP. On scheduability test of rate monotonic and its extendible algorithms. Journal of Software, 2004,15(6): 799−814 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/799.htm

[3] Zou Y, Li MS, Wang Q. Analysis for scheduling theory and approach of open real-time system. Journal of Software, 2003,14(1): 83−90 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/83.htm

[4] Holzmann J. The model checker spin. IEEE Tran. on Software Engineering, 1997,23(5):279−295. [doi: 10.1109/32.588521]

[5] Kripke SA. Semantical analysis of modal logic I: Normal propositional calculi. Mathematische Logik und Grundlagen der Mathematik, 1963,9:67−96. [doi: 10.1002/malq.19630090502]

[6] Pnueli A. The temporal logic of programs. In: Proc. of the 18th Annal IEEE Symp. on Foundations of Computer Science. IEEE Computer Society: IEEE Computer Society, 1977. 46−57. [doi: 10.1109/SFCS.1977.32]

[7] Wolper PL. Temporal logic can be more expressive. Information and Control, 1983,56:72−99. [doi: 10.1016/S0019-9958(83)80051-5]

[8] Armoni R, Fix L, Flaisher A, Gerth R, Ginsburg B, Landver A, Mador-Haim S, Singerman E, Tiemeyer A, Vardi M. The ForSpec temporal logic: A new temporal property-specification language. In: Katoen JP, Stevens P, eds. Proc. of the TACAS 2002. LNCS 2280, Berlin, Heidelberg: Springer-Verlag, 2002. 296−311. [doi: 10.1007/3-540-46002-0_21]

[9] Clark M, Gremberg O, Peled A. Model Checking. Cambridge: The MIT Press, 2000.

[10] Duan ZH. An extended interval temporal logic and a framing technique for temporal logic programming [Ph.D. Thesis]. Newcastle: University of Newcastle Upon Tyne, 1996.

[11] Duan ZH, Tian C, Zhang L. A decision procedure for propositional projection temporal logic with infinite models. Acta Informatica, 2008,45(1):43−78. [doi: 10.1007/s00236-007-0062-z]

[12] Tian C, Duan ZH. Complexity of propositional projection temporal logic with star. Mathematical Structures in Computer Science, 2009,19(1):73−100. [doi: 10.1126/science.19.471.73]

[13] Duan ZH. Temporal Logic and Temporal Logic Programming. Beijing: Science Press, 2006.

[14] Tian C, Duan ZH. Model checking propositional projection temporal logic based on SPIN. In: Butler M, Hinchey M, Larrondo-Petrie MM, eds. Proc. of the ICFEM 2007. LNCS 4789, Berlin, Heidelberg: Springer-Verlag, 2007. 246−265.

[15] Tian C, Duan ZH. Propositional projection temporal logic, Büchi automata and omega-expressions. In: Agrawal M, *et al*., eds. Proc. of the 5th Annual Conf. on Theory and Applications of Models of Computation. LNCS 4978, Berlin: Springer-Verlag, 2008. 7−58.

[16] Lehoczky JP, Sha L, Ding Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In: Proc. of the 10th IEEE Real-Time Symp. IEEE Computer Society Press, 1989. 166−171. [doi: 10.1109/REAL.1989. 63567]

[17] Moszkowski B. Reasoning about digital circuits [Ph.D. Thesis]. Stanford: Stanford University, 1983.

[18] Mordechai B. Principles of the SPIN Model Checker. London: Springer-Verlag, 2008.

[19] Holzmann J. The Spin Model Checker: Primer and Reference Manual. Addison-Wesley, 2004.

附中文参考文献:

[2] 王永吉,陈秋萍.单调速率及其扩展算法的可调度性判定.软件学报,2004,15(6):799−814. http://www.jos.org.cn/1000-9825/15/799.htm

[3] 邹勇,李明树,王青.开放式实时系统的调度理论与方法分析.软件学报,2003,14(1):83−90. http://www.jos.org.cn/1000-9825/14/83.htm

附录:RMS 调度实例的 PROMELA 模型

```
typedef task {byte P; byte E}
task t1, t2, t3;
int clock;
byte e1, e2, e3;
init {
    atomic{t1.P=8;t1.E=3;t2.P=9;t2.E=3;t3.P=14;t3.E=3;
        e1=t1.E; e2=t2.E; e3=t3.E;
        run schedule(); run Request_1(); run Request_2(); run Request_3();
        clock=0
        }
}
proctype Request_1() {
do
::clock%t1.P==0→e1=t1.E
od
}
proctype Request_() {
do
::clock%t2.P==0→e2=t2.E
od
}
proctype Request_3(){
do
:: clock%t3.P==0→e3=t3.E
od
}
```

proctype Schedule() {
do
::e_1!=0 && (*clock*+1)%t_1.*P*!=0→*atomic*{*clock*=*clock*+1;e_1=e_1−1}
::e_1==0 && e_2!=0 && (*clock*+1)%t_2.*P*!=0→*atomic*{*clock*=*clock*+1;e_2=e_2−1}
::e_1==0 && e_2==0 && e_3!=0 && (*clock*+1)%t_3.*P*!=0→*atomic*{*clock*=*clock*+1;e_3=e_3−1}
::e_1==0 && e_2==0 && e_3==0→*atomic*{*clock*=*clock*+1}
od
}

田聪(1981－),女,陕西合阳人,博士,讲师,主要研究领域为形式化方法,时序逻辑,模型检测.

段振华(1948－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为高可信软件,网络计算.