软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(6):1236−1251 [doi: 10.3724/SP.J.1001.2011.04015]

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

基于自动机理论的分布式实时调度分析工具*

桂盛霖[1+], 罗 蕾[1], 李 允[1], 于 淼[2], 徐建华[2]

[1](电子科技大学 计算机科学与工程学院,四川 成都 610054)

[2](西南交通大学 信息科学与技术学院,四川 成都 610031)

Schedulability Analysis Tool for Distributed Real-Time Systems Based on Automata Theory

GUI Sheng-Lin[1+], LUO Lei[1], LI Yun[1], YU Miao[2], XU Jian-Hua[2]

[1](School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

[2](School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)

+ Corresponding author: E-mail: shenglin_gui@uestc.edu.cn

Gui SL, Luo L, Li Y, Yu M, Xu JH. Schedulability analysis tool for distributed systems based on automata theory. *Journal of Software,* 2011,22(6):1236−1251. http://www.jos.org.cn/1000-9825/4015.htm

Abstract: Distributed systems are complicated real-time systems, which have been used in many safety-critical domains. In order to ensure the real-time constraints over the tasks running on these systems, the traditional schedulability analysis techniques, based on worst-case response time analysis, usually consider the worst case which could never occur in real world applications, and therefore the obtained results are pessimistic. The model checking technique based on automata theory could exhaustively search the whole system state space and return precise results. By using formal methods to analyze the schedulability of tasks on distributed systems, this paper presents the formal task model on distributed systems. It uses action automata and environment automata to model the task execution semantics and the external event arrival patterns respectively. It also translates the schedulability analysis to the reachability analysis of the locations in automata network, and proves the decidability of schedulability under certain scheduling policies with attached conditions and scope. Based on these conclusions, the formal check model implements a schedulability check tool, SCT (schedulability checking tool), and compares it with other techniques and tools on accurateness and performance. The comparisons show that SCT always provides the most accurate results but with the longest execution time.

Key words: distributed system; hard real-time; schedulability analysis; action automata; environment automata

摘 要: 分布式实时系统是广泛应用在众多关键领域的一类复杂实时系统.为保证其上运行任务的实时性,传统基于最坏响应时间的调度分析方法往往包含了实际系统运行过程中无法达到的最坏情况,因此在这些情况下的分析结果过于悲观.基于自动机理论的模型检测方法的好处在于能够穷尽地搜索整个系统状态空间,得到精确的分析结果.为了利用形式化方法的优势来精确分析分布式系统上任务的调度性,建立了分布式系统上的任务形式化模型,提出了行为自动机和环境自动机以分别描述任务的执行语义及其外部到达关系,把任务的调度性分析转换为对自

* 基金项目: 国家自然科学基金(90718019); 国家高技术研究发展计划(863)(2007AA010304); 国家“十一五”重点科技攻关项目(51315040106)

收稿时间: 2010-07-09; 定稿时间: 2011-03-29

动机网络位置的可达性进行分析,证明了在某些调度策略下的调度性的可判定性,并给出了满足调度可判定性调度策略的条件和范围.基于上述结论,实现了一个支持分布式系统任务实时调度分析工具 SCT(schedulability checking tool),并与其他工具进行了分析精确度和性能的比较.比较结果显示,SCT 可以提供最为精确的分析结果,但同时也具有最长的分析时间.

关键词: 分布式系统;硬实时;调度性分析;行为自动机;环境自动机

中图法分类号: TP311 **文献标识码**: A

调度分析是实时领域一个非常重要的研究方向,自从 20 世纪 70 年代 Liu 和 Layland 提出调度分析模型[1]以来,研究人员针对不同的系统计算模型和不同的调度策略提出了大量的调度分析技术.调度分析寻找系统的最坏情况并且检查在此最坏情况下系统资源的分配情况,如果在最坏情况下所有任务在自己的截止期限内都能得到足够的系统资源并且执行完毕,则在其他更有利的情况下任务也能正常执行完毕.调度分析技术通常可分为两大类:

一类是基于资源利用率,通过计算资源的使用率来确定任务集是否可调度[1],并且通常只是充分条件,但判断简单,执行效率高,适合大规模系统的分析.例如,在满足:1) 任务相互之间独立;2) 任务是周期任务;3) 任务不能被信号量等机制阻塞;4) 任务的截止时间等于周期;5) 任务的到达时立即被释放这 5 个条件时,如果处理器利用率低于 $\sum_{i=1}^{n} C_i/T_i \leqslant n(2^{1/n}-1)$,其中,$C_i$ 和 T_i 分别是任务 i 的最坏执行时间和周期,任务数为 n,则该任务集一定能被 RMS(rate-monotonic scheduling)调度算法成功调度.

另一类调度分析技术是基于任务响应时间分析,通过计算任务的最坏响应时间来与任务的截止时间进行比较.如果任务集中每个任务的最坏响应时间均不超过它的截止时间,则该任务集可调度[2].虽然基于任务响应时间分析较前一种技术分析精确度更高,但其效率却更低.

尽管两种技术被认为是等价的[3],但是人们可以根据自己的需求灵活选择更合适的方法.由于基于自动机理论的模型检测方法在状态空间搜索中所体现出来的精确性优势[4],自 20 世纪 90 年代中期开始,时间自动机(timed automata)[5]和 Stopwatch 自动机被分别用来解决 Job-shop 调度问题中的不可抢占调度策略和抢占调度策略的检查[6,7].这些技术将自动机的某些位置预定义为目标位置,这样将对调度性的分析转换为对位置的可达性分析.如果可达性是可判定的,则对调度性的分析也是可判定的.而对 Job-shop 问题中可抢占调度策略进行建模的 Stopwatch 自动机的可达性是不可判定的[8],因此,寻找和证明一种具有足够语义描述能力且可达性是可判定的自动机成为解决此类问题的关键.

在实时系统领域,分布式系统是一类结构复杂的实时系统,不同处理器上的任务可能存在优先或资源互斥等复杂关系.虽然目前已有一些支持分布式系统的调度分析方法和工具,但是这些方法和工具在某些条件下往往考虑了一些系统无法真正到达的最坏情况,从而造成分析结果过于悲观.考虑到自动机理论的优势,本文针对分布式系统任务到达的特点改进了时间自动机,使之能够对任务的到达模式进行建模,还能够建模任务之间通信媒介的传输延迟.而对任务的执行语义,则提出了一种新的层次化自动机模型:行为自动机,形成自动机网络来对分布式系统上的任务执行进行建模,从而利用自动机的可达性分析来精确确定任务集的可调度性.因此,行为自动机状态的可达性判断成为本文研究的关键.

本文第 1 节介绍相关工作.第 2 节定义所使用的任务模型.第 3 节和第 4 节分别定义环境自动机和行为自动机.第 5 节介绍使用环境自动机和行为自动机对分布式系统上的任务进行建模的过程,并给出行为自动机位置可达性的可判定条件及理论证明和相关推论.第 6 节给出一个具体实例以及基于本文所开发的调度分析工具 SCT 与其他相关技术和工具的详细比较.最后,在第 7 节对全文的研究工作进行总结和展望.

1 相关工作

自动机技术对调度性进行分析的本质是属于上节所述的第 2 类调度方法,是利用其穷举性对任务的响应时间进行分析,找到任务的最坏响应时间.尽管 UPPAAL[9]可以对抢占调度策略下的任务进行调度性分析[10],但

用户自己不得不显式地对系统进行建模,不但对用户的专业化程序要求高,而且在系统复杂程度较高时,用户进行建模行为的工作量较大且容易出错.UPPAAL 的一个扩展是在 4.1 版本中支持 Stopwatch 自动机来对系统建模[11],然后进行近似分析.

为使更多的用户能够更方便地利用自动机的优势对调度性进行精确分析,文献[12]提出了任务自动机建模任务的到达行为,并且实现了一个单处理器调度分析工具 TIMES[13].为了避免分析抢占调度策略而使用 Stopwatch 自动机时所带来的可达性的不可判定问题,文献[12]中在普通时间自动机中扩展了有界减法(bounded subtraction)操作,使能够建模一些常见的抢占性调度策略,如单调速率调度算法(RMS)和最短截止时间优先调度算法(earliest deadline first,简称 EDF),并且证明了其可达性仍然是可判定的.尽管在随后的工作中将单处理器模型扩展到多处理器中[14],但并没有考虑分布在不同处理器上的任务之间的交互关系,即每个处理器是一个独立的子系统,单独使用文献[12]中的方法进行分析.文献[15]针对单处理器上的固定优先级调度策略对任务自动机的位置数和时钟数进行了优化,在没有数据依赖的条件下,对任务执行语义进行建模的自动机中只需两个时钟,大大约减了系统状态空间.

在分布式实时系统领域,目前已有一些使用较为广泛的理论和方法:SymTA/S[16]是基于 Holistic 调度分析方法[17]的扩展,使用了标准事件模型[18]来耦合分布式系统各个组件之间的联系,从而支持在系统级进行调度分析;实时演算(real-time calculus)[19]是基于网络演算(network calculus)[20]的扩展,通过建立合适的任务到达曲线和系统服务曲线来对任意模式的事件流进行建模,从而不仅仅局限于将非周期事件流近似估计为带最小到达间隔时间的事件流.以上两种方法均是通过估计系统各个模块中任务的最坏可能情况下的响应时间.文献[21]通过建立一系列的测试用例对以上的几种调度分析方法在建模能力、建模复杂程度、工具支持和精确度方面做了详细对比.结果显示,基于自动机理论的调度分析方法可以得到比其他几种方法更精确的分析结果,但是分析时间远远长于其他几种方法.同时,在某些特定情况下,其他方法可以得到非常接近于基于自动机理论的分析结果.因此,基于自动机理论的调度分析方法是以牺牲分析时间来换取更加精确的分析结果的方法.

为了避免采用自动机理论进行调度分析过程中所带来的状态爆炸问题,尽量缩短分析时间,人们提出了一些比较有效的符号化技术和规约技术来表述状态空间,以避免穷尽搜索带来的状态空间过大.例如,region graph 和 zone graph[5]、固定点迭代技术(fixpoint iteration)[22]、分割细化技术(partition-refinement)[23]、DBM 的压缩数据结构[24]、时钟差分图(clock difference diagrams)[25]、on-the-fly 方式的动态可达符号状态集的构造[25].

尽管已有 UPPAAL 和 TIMES 等基于自动机理论的调度分析工具,通过前面的分析,分布式系统上任务的复杂执行行为不能由前者轻易地建模,并且模型的复用程度不高.后者把建模细节作为后台处理,用户只需输入任务的相关参数即可,极大地方便了用户的使用,但是只支持单处理器上的任务调度分析.因此,结合两个工具的优点,本文建立了支持分布式系统环境下的任务的自动机语义模型,并完成了一个调度分析原型工具 SCT,使得用户只需输入系统和任务参数即可进行复杂环境下的调度性分析,得到精确的分析结果.

2 任务模型

任务 T_i 定义为来自外部环境或者其他所关联任务的无限次实例执行请求(也称为事件).每两个执行请求的到达时间间隔为 P_i,初始请求可以具有一个相位 O_i.如果任务 T_i 是周期任务(periodic task),则其到达时间间隔 P_i 为固定常数;如果任务 T_i 是非周期任务(aperiodic task)或是零星任务(sporadic task),则到达间隔 P_i 可以根据用户需求确定一个最小下界值;如果任务 T_i 是零星性周期任务(sporadically periodic task)[2],则到达间隔 P_i 可由两个参数来代替: P_i^e 和 P_i^o,分别表示任务的固定内部周期值和外部最小时间间隔值;并且 P_i^e 和 P_i^o 满足 $P_i^o \geqslant k_i \cdot P_i^e$, 其中,$k_i$ 表示任务按照固定内部周期值所需要连续到达的次数.每次到达的任务称作任务实例,需要在相对截止时间 D_i 前完成最坏执行时间 C_i.临界区互斥是实时系统中普遍存在的机制,以保证对关键资源的正确顺序访问.本文假设由于使用优先级天花板[26]等技术以保证临界区互斥所带来的高优先级任务被低优先级任务阻塞所产生的运行开销已经被考虑在每个任务的最坏计算负载 C_i 中.任务的相对截止时间 D_i 与到达间隔 P_i(如果任务是零星性周期任务,P_i 替换为 P_i^e)之间可为任意关系.如果任务没有缓冲区来缓冲它的到达请求,则

每次所到达的请求都会触发该任务的一个新实例进行执行;否则,如果当前的任务尚未执行完毕,其所到达的执行请求存储在一个固定大小的缓冲区里.即在任意时刻,该任务最多只有一个实例存在于任务队列中.在当前任务实例执行完毕后,检查其缓冲区是否还有被缓冲的执行请求,如果有,则从缓冲区中取出并开始一次新的实例执行;否则,任务转入挂起等待状态.当缓冲区满时,所到达的执行请求会被丢弃.

图 1 给出了任务模型的一个例子.其中,6 个任务被分配在两个处理器上,任务 T_1,T_2 和 T_4 接收系统外部环境产生的事件流,称为任务简单触发模式.当任务 T_1,T_2 或 T_4 的实例执行完毕之后,会立即产生事件.任务 T_3,T_5,T_6 分别接收任务 T_2,T_3,T_4 的输出事件流,称为任务复杂触发模式.注意,在任务复杂触发模式中,任务当前实例执行完毕时才能输出事件到接收任务.任务 T_i 和 T_j 之间的事件传输延迟表示为 DL_{ij}.传输媒介传输延迟可以通过下节定义的环境自动机建模.本文假设所有任务是静态分配在处理器上,系统运行过程中任务不能在处理器间动态迁移.此外,任务属性相对截止时间是一个硬截止期限,任务必须在截止期限内执行完毕.不考虑软截止期限.

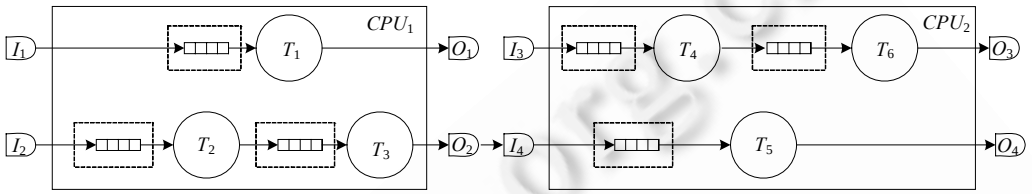


Fig.1 Complex activation pattern

图 1 任务复杂触发模式

3 环境自动机

在任务简单触发模式中,任务的事件流直接来源于外部环境,并且任务之间的事件流传输依赖于外部传输媒介,如总线、网络等.本文改进了时间自动机对分布式系统中由外部环境直接触发的任务的到达模式和传输媒介延迟进行建模,而对任务复杂触发模式的建模在第 4 节讨论.

环境自动机本质上是一个时间自动机[5],本文用来定义直接来源于外部环境的实例执行请求,首先给出符号定义.环境自动机上的有限事件集合表示为 $Sync^o$,有限实值时钟集合 X,形如 $x\sim c$ 的原子限制的合取公式集合 $\Phi_0(X)$,其中,$x\in X$,$\sim\in\{<,\leqslant,\geqslant,>\}$,$c$ 是一个自然数.$\Phi_0(X)$里的元素称为时钟限制,时钟值可以表示为从时钟集合 X 到非负实数 $R_{\geqslant 0}$ 的一个映射 μ.对 $t\in R_{\geqslant 0}$,$\mu+t$ 表示 $\forall x\in X,\mu(x)+t$.对 $X_i\subseteq X$,$\mu[X_i]$表示将 X_i 中的所有时钟的值重置为 0 且保持 X 中剩下的时钟值不变.

定义 1(环境自动机). 环境自动机是一个六元组 $A=(Sync^o,L,L_0,X,E,inv)$,其中:

- $Sync^o$ 是有限事件集合,用来与行为自动机同步;
- L 是位置的有限集合,L_0 是初始位置集合,满足 $L_0\subseteq L$;
- X 是一个有限时钟集合;
- $E\subseteq L\times[Sync^o]\times 2^X\times\Phi_0(X)\times L$ 定义了迁移关系.符号[]表示其内部的元素可以不出现在迁移边上.迁移边 $(l, sync_i^o,\lambda,\delta,l')$表示从位置 l 到 l'的迁移,且和行为自动机在同步事件 $sync_i^o$ 上同步.$\lambda\subseteq X$ 是被重置的时钟且 $\delta\in\Phi_0(X)$.
- $inv:L\to\Phi_0(X)$称为不变量函数,定义在位置上所必须满足的时间区域.

图 2 是使用环境自动机来对周期、零星、零星性周期时间流及传播媒介延迟建模的例子.在图 2(a)中,周期事件流 $sync_i^o$ 的周期值为 10 个时间单位,初始相位为 3 个时间单位;图 2(b)中,零星事件流 $sync_i^o$ 的最小到达间隔为 10 个时间单位;零星性周期任务的建模比前两者更复杂,如图 2(c)所示,$P_i^e=8$ 和 $P_i^o=8k+10$.其中,k 是已知常数.图 2(d)建模了任务 T_3 和 T_5 之间的最大传输延迟为 1 个时间单位值的传输媒介,并且假设一固定时间间隔内所到达的最大传输请求数 m 是已知的,在图 2(d)中,m=3.为了处理并发关系,环境自动机的并行组合操作可以使用与普通时间自动机的相同并行组合操作[27],受篇幅所限,这里不再赘述.

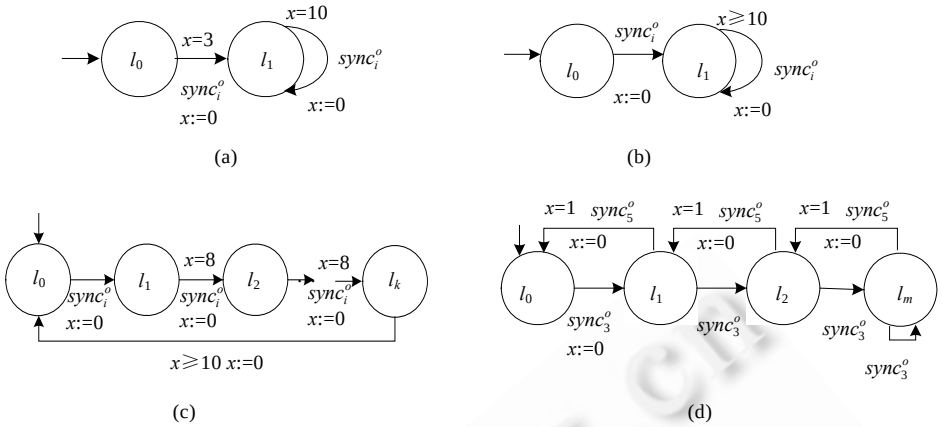


Fig.2 Four cases to illustrate how to use environment automata to model events

图 2 使用环境自动机建模的 4 个例子

4 行为自动机

本节提出 Suspension 自动机[28]的一个特定子类——行为自动机,来建模分布式系统上的任务的执行语义,包括任务实例的执行行为及任务之间的复杂触发模式关系.以前大部分基于自动机理论的调度分析技术均采用将可调度性转换为可达性的分析[6,7,10,13],本文也采用相同的原理.在文献[28,29]中证明了一般性通用的 Suspension 自动机的可达性是不可判定的,除非具有初始化行为.但是,本文在第 5.2 节将证明行为自动机的可达性是可判定的,从而给出了除具有初始化行为外,在满足行为自动机所限制的前提条件下,Suspension 自动机的可达性也是可判定的.先给出 Suspension 自动机所需要使用的符号及定义[28]:C 是一个有限实值计时器集合;$\Phi(C)$称为计时器限制,是形如 $c_i\sim c$ 或者 $c_i-c_j\sim c$ 的原子限制的合取公式集合.其中,$c_i,c_j\in C,\sim\in\{<,\leqslant,\geqslant,>\},c$ 是一个自然数.

定义 2(Suspension 自动机). Suspension 自动机是一个七元组 $A=(Sync,L,L_0,C,flow,E,inv)$,其中:

- $Sync$ 是一个有限事件集合;
- L 是位置的有限集合,L_0 是初始位置集合,满足 $L_0\subseteq L$;
- C 是一个有限计时器集合;
- $flow:L\to\prod_{i=1}^{|C|}k_i$ 是定义在位置上的计时器增长速率赋值函数,其中,$k_i=1$ 或 0;
- $E\subseteq L\times[Sync]\times 2^C\times\Phi(C)\times L$ 定义了迁移关系;
- $inv:L\to\Phi_0(X)$仍然是不变量函数.

在 Suspension 自动机中,如果$\forall c_i\in C,\forall l\in L,flow(l)(c_i)$是唯一的数值而不是一个范围值,称这种 Suspension 自动机是速率确定的.在每个行为自动机里,除了有一个有限计数器集合 C 之外,还有一个有限时钟集合 D.将时钟值和计时器值定义为在位置 l 上从集合 $C\cup D$ 到非负实数 $R_{\geqslant 0}$ 的一个映射 v_l.对 $t\in R_{\geqslant 0}$ 且 $v_l+t\in inv(l),v_l+t$ 表示对所有 $d_i\in D$ 有 $v_l(d_i)+t$,对所有满足 $flow(l)(c_i)=1$ 的计时器 c_i 有 $v_l(c_i)+t$;对所有满足 $flow(l)(c_i)=0$ 的 c_i,其 $v_l(c_i)$不变.对 $X\subseteq C\cup D,v_l[X]$表示在位置 l 中把集合 X 中的所有元素的值置 0.

定义 3(行为自动机). 行为自动机是一个十一元组 $A=(L,Sync,L_0,\rho,\sigma,C,D,V,flow,E,inv)$,其中:

- L 是位置的有限集合,且满足 $L=L^b\cup L^c$.其中:L^b 是一个基本位置集合;L^c 是一个组合位置集合,并且 $\exists!l\in L^c,\forall l'\neq l,l'\in\rho^*(l)$,称其为 $Root$;
- $Sync$ 是一个有限事件集合,满足 $Sync=Sync^o\cup Sync^e$.其中,$Sync^o$ 和 $Sync^e$ 分别称为外部事件集合和内部事件集合,分别用于与外部环境和其他任务同步;
- $L_0\subseteq L^b$ 是初始位置集合;

- $\rho: L^c \to 2^L$ 是标记函数,将一个组合位置 l 映射到所有它所包含的子位置集合.对$\forall l \in L^c$,$\rho(l)$表示位置 l 的直接子孙位置集合;对任意的位置 $l \neq Root$,$\rho^{-1}(l)$表示位置 l 的直接祖先位置;$\rho^*(l)=\rho^0(l) \cup \rho^1(l) \cup \rho^2(l) \cup \ldots \cup \rho^\infty(l)$,表示 l 及其所有子孙位置所组成的集合.因此,ρ^*表示ρ的自反传递闭包;
- $\sigma: L \to \{and, xor, basic, \varnothing\}$将每个位置关联一个类型.在 *xor* 类型的位置中,在任何时刻只有一个直接子孙位置处于活跃状态;在 *and* 类型的位置中,在任何时刻其所有直接子孙都是活跃的;*basic* 类型位置也叫基本位置,其没有子孙位置.*xor* 类型的位置和 *and* 类型的位置统称为组合位置集合,每个位置都必须属于这 3 种类型位置的一种.注意,$\sigma(L')=and(L' \subseteq L)$表示 L'中的所有位置的类型均为 *and*.对任意行为自动机,以下限制必须被满足:(a) $\sigma(Root)=and$;(b) $\sigma(\rho(Root))=xor$;(c) $\sigma(\rho^2(Root))=basic$;
- C 和 D 分别是有限的计时器集合和时钟集合,其中,$|C|=|D|=n$(n 是常数),并且存在 C 到 D 的双射函数 $f_c: C \to D$,使得 C 和 D 中的计时器和时钟存在一一对应关系.此外,对每个计时器 $c_i \in C$,存在一个常数 M_{c_i},满足计时器 c_i 的值不会超过 M_{c_i};对每个时钟 $d_i \in D$,存在一个常数 M_{d_i},满足时钟 d_i 的值不会超过 M_{d_i};
- V 是一个有限离散变量集合,其中,每个变量的值域是有限的;
- $flow: L^b \to 2^{\prod_{i=1}^{n} k_i}$ 是计时器增长速率的确定赋值函数,其中,$k_i=1$ 或 0,$n \leqslant |C|$(引入函数 B,定义为从位置 l 映射到满足$\{c_i | flow(l)(c_i)=1$ 或 $0\}$的 C 的子集),且满足如下限制:
  (a) 在任意时刻,C 中最多只能有一个计时器的增长率为 1;
  (b) 如果某个时刻存在某个计时器 $c_i \in C$ 的值不为 0,则该时刻一定存在一个计时器其增长率为 1;
  (c) 如果某个时刻 C 中所有的计时器的增长率都为 0,则它们的值也全部为 0;
  (d) 如果 $L' \subseteq \rho^2(Root)$是当前同时处于活跃状态的基本位置集合,则$\cup_{l \in L'} B(l)=C$ 且$\forall l, l' \in L', l \neq l'$, $B(l) \cap B(l')=\varnothing$.
- $E \subseteq L^b \times [Sync] \times 2^{C \cup D \cup V} \times \Phi_0(C) \cup \Phi(D) \cup \Phi_0(V) \times L^b$ 定义迁移关系.对任意迁移 $e_{mi}=(l_{mi}, sync_i, \lambda_{mi}, \delta_{mi}, l'_{mi})$满足如下限制:
  (a) $\rho^{-1}(l_{mi})=\rho^{-1}(l'_{mi})=l_m \in \rho(Root)$且 $l_{mi} \in L^b, l'_{mi} \in L^b$;
  (b) $c_i \in \lambda_{mi}$ 当且仅当 $d_i \in \lambda_{mi}$ 且满足 $f_c(c_i)=d_i$;
  (c) 如果 $(c_i = M_{c_i}) \in \delta_{mi}$,则 $flow(l_{mi})(c_i)=1$ 且$(c_i:=0) \in \lambda_{mi}$ 且 $flow(l'_{mi})(c_i) = 0$;
  (d) 对任意 c_i,如果 $flow(l_{mi})(c_i)=0$,则 $\delta_{mi} \cap \Phi_0(c_i)=\varnothing$. 对任意两个内部同步迁移 $e_{mi} = (l_{mi}, [sync_i^e], \lambda_{mi}, \delta_{mi}, l'_{mi})$ 和 $e_{nj} = (l_{nj}, [sync_i^e], \lambda_{nj}, \delta_{nj}, l'_{nj})$,满足如下限制:
    (1) $\rho^{-1}(l_{mi}) = \rho^{-1}(l'_{mi}) \neq \rho^{-1}(l_{nj}) = \rho^{-1}(l'_{nj})$,$m \neq n$,且$\delta_{mi}$和$\delta_{nj}$都必须被满足;
    (2) 如果$\exists c_i, c_j (i \neq j)$,$flow(l_{mi})(c_i)=1$ 且 $flow(l'_{nj})(c_j) = 1$,则必须满足:
      1) $(c_i = M_{c_i}) \in \delta_{mi}$;或者
      2) $(c_i = M_{c_i}) \notin \delta_{mi}$ 且 $v_{l_{nj}}(c_j) = 0$.
- $inv: L^b \to \Phi_0(C) \cup \Phi_0(D)$是定义在基本位置上的不变量函数.

注意,在同一行为自动机内部的两条迁移即使标记了同一同步事件,也还必须满足定义 3 中第 9 条子句所定义的限制,同步迁移才能发生.但是在行为自动机与环境自动机之间、行为自动机与行为自动机之间、环境自动机与环境自动机之间的同步迁移的发生则只能依靠同步事件集合 $Sync^o$.与环境自动机与环境自动机之间所定义的并行组合操作类似,行为自动机与环境自动机之间以及行为自动机与行为自动机之间的同步迁移关系给出了它们之间的并行组合操作关系基础.受篇幅所限,行为自动机与环境自动机之间和行为自动机与行为自动机之间的并行组合操作的详细定义这里不再赘述.

5 调度分析建模

本节使用一个行为自动机来对每个处理器上的任务执行语义进行建模,因此,同一处理器上的任务之间的

复杂触发关系可由行为自动机的两条内部迁移边的同步关系来建模,而不同处理器上任务之间的复杂触发关系在不考虑事件传输延迟的条件下由不同行为自动机之间迁移边的同步关系来建模.否则,建模任务语义的行为自动机的迁移边需要和建模传输延迟的环境自动机的迁移边同步.为了讨论方便及篇幅限制,若不做特殊说明,本节的剩余部分仅讨论如何使用行为自动机 A 来对分布式系统中某一处理器上的任务进行调度建模.

注意,除零星性周期任务外,每个任务最大可能同时出现的实例数上界为 $W_i=ceil(D_i/P_i)$(函数 $ceil(x)$返回 x 的最大整数下界值),而每个零星性周期任务最大可能同时出现的实例数上界为 $W_i=ceil(D_i/P_i^o)\cdot k_i$.引入状态函数 $state(T_{ij})=\{suspend,released\}$分别表示任务 T_i 的第 j 个实例的当前状态:$suspend$ 状态用于建模当前任务实例处于挂起等待状态,$released$ 状态用于建模在该任务实例已经被触发进入任务队列中.简便起见,引入谓词 $HP(T_{ij})$表示任务实例 T_{ij} 当前在任务就绪队列中具有最高优先级,谓词 $HI(T_{ij})$表示任务实例 T_{ij} 在当前任务 T_i 已经被触发的实例中具有最高优先级.显然,如果 $HP(T_{ij})=True$,则 $HI(T_{ij})=True$.由于任务及任务实例的优先级是由具体的调度策略决定的,因此可以把调度策略用谓词 $HP()$和 $HI()$表达出来.

5.1 任务语义建模

如果任务没有缓冲区存储到达的执行请求,则每个到达的请求会触发该任务的一个新实例进行执行.对任务 T_i 的每个实例 T_{ij},使用一个计时器 c_{ij} 来计时它的累积计算时间,使用一个时钟 d_{ij} 来计时累积截止时间并且满足 $f_c(c_{ij})=d_{ij}$.对每个任务 T_i,其所有实例的执行语义使用一个位置 l_i 及其子位置建模,且满足 $l_i\in\rho(Root_A)$.对每个这种类型的任务 T_i,行为自动机 A 的位置 l_i 有如下直接子孙位置及其限制:

(1) $suspend$ 表示该任务的所有实例都还未被激活,并且 $\forall j,1\leqslant j\leqslant W_i,flow(suspend)(c_{ij})=0$.此外,$suspend\in L_0$.

(2) $ready_{ij}$ 表示任务实例 T_{ij} 就绪等待被执行,其中,$1\leqslant j\leqslant W_i$.其上的计时器增长率和不变量定义为:

(a) $\forall m,1\leqslant m\leqslant W_i,flow(ready_{ij})(c_{im})=0$;

(b) $inv(ready_{ij})=\{\forall m,1\leqslant m\leqslant W_i,d_{im}\leqslant D_i\}$.

(3) $running_{ij}$ 表示当前任务实例具有最高优先级并且处于执行状态,其中,$1\leqslant j\leqslant W_i$.

如果 $m=j$,则 $flow(running_{ij})(c_{im})=1$;否则,$flow(running_{ij})(c_{im})=0$.$running_{ij}$ 上的不变量定义为集合$\{c_{ij}\leqslant C_i,\forall m.1\leqslant m\leqslant W_i,d_{im}\leqslant D_i\}$.

(4) $error_i$ 表示任务 T_i 的某个实例在截止期限内未正常执行完毕.注意,位置 $error_i$ 上不必定义不变量函数及计时器增长率函数.因为一旦进入该位置,说明该任务不可调度,分析立即结束.

位置 l_i 的直接子孙位置之间的部分迁移关系为:

(1) 位置 $suspend$ 到位置 $ready_{ij}(1\leqslant j\leqslant W_i)$:

- 迁移条件:$\exists j,1\leqslant j\leqslant W_i,state(T_{ij})=suspend$;
- 同步事件标记:有;
- 重置:$c_{ij}:=0,d_{ij}:=0,state(T_{ij}):=released$;

(2) 位置 $running_{ij}$ 到位置 $ready_{ik}$:

- 迁移条件:$HP(T_{ij})\neq True,HI(T_{ik})=True,state(T_{ik})=released,c_{ij}=C_i,d_{ij}<D_i$;
- 同步事件标记:有(若任务 T_i 是复杂触发关系的源任务,则有同步事件标记,否则无同步事件标记);
- 重置:$c_{ij}:=0,d_{ij}:=0,state(T_{ij}):=suspend$.

由于篇幅所限,本节不再一一列举所有位置之间迁移边上的时间约束关系,仅举几个典型的迁移条件加以说明,其他一些迁移上的约束,如从位置 $ready_{ij}$ 和位置 $running_{ij}$ 之间的迁移条件取决于具体的调度策略.一个实际的例子将在第 6 节给出.

如果任务有一个有限容量为 K_i 的缓冲区存储到达的执行请求,则在任意时刻该任务最多只有一个任务在任务队列中.当缓冲区被装满但仍有执行请求继续到达时,则新到达的执行请求自动被丢弃.对任务 T_i 使用一个计时器 c_i 来计时它的累积计算时间,使用一个时钟 d_i 来计时累积截止时间并且满足 $f_c(c_i)=d_i$.对每个任务 T_i,其所有实例的执行语义使用一个位置 l_i 及其子位置建模,且满足 $l_i\in\rho(Root_A)$.对每个这种类型的任务 T_i,行为自动机 A

的位置 l_i 有如下直接子孙位置及其限制:

(1) *suspend* 表示该任务还未被激活,并且 $flow(suspend)(c_i)=0$.此外,$suspend\in L_0$.

(2) $ready_{ij}$ 表示任务 T_i 就绪等待被执行,其中,$0\leqslant j\leqslant K_i$,并且满足:$flow(ready_{ij})(c_i)=0$,$inv(ready_{ij})=\{d_i\leqslant D_i\}$.

(3) $running_{ij}$ 表示当前任务实例具有最高优先级并且处于执行状态,其中,$0\leqslant j\leqslant K_i$,且 $flow(running_{ij})(c_i)=1$.$running_{ij}$ 上的不变量定义为集合$\{c_i\leqslant C_i,d_i\leqslant D_i\}$.

(4) $error_i$ 表示任务 T_i 在截止期限内未正常执行完毕.

位置 l_i 的直接子孙位置之间的部分迁移关系为:

(1) 位置 $ready_{iw}$ 到位置 $ready_{i(w+1)}(0\leqslant w\leqslant K_i-1)$:

- 迁移条件:无;
- 同步事件标记:有;
- 重置:无;

(2) 位置 $ready_{iw}$ 到位置 $running_{iw}(0\leqslant w\leqslant K_i)$:

- 迁移条件:$state(T_i):=ready$,$HP(T_i)=True$;
- 同步事件标记:无;
- 重置:$state(T_i):=running$.

注意,针对后一种任务类型,行为自动机 *A* 使用子位置集合(如 $ready_{ij}$)来记忆缓冲区中所保存的执行请求个数.如果某个行为自动机的子位置 $error_i$ 在某种条件下是可达的,则该条件下的任务集不可调度,这样就可以通过分析环境自动机和行为自动机网络的并行组合来分析分布式系统中任务的调度性问题.

定义 4(可调度性). 将绑定在多处理器(或单处理器上)的任务集合的外部环境和传输媒介由环境自动机建模,每个处理器上的任务执行语义及调度策略由一个行为自动机建模,如果在该环境自动机和行为自动机网络的并行组合中的可达位置集合中不包括任何子位置 $error_i$,则该任务集合可调度.

基于可调度性的定义,可以得到定理 1.

定理 1. 如果每个处理器上的调度策略均不存在比较任务(任务实例)之间已经流逝了的执行时间之间的大小关系,也不存在一个任务(任务实例)的已经流逝了的执行时间与另一个任务(任务实例)的已经流逝了的相对截止时间之间的比较,则多处理器上的任务集合的可调度性是可判定的.

证明:由定义 3 和第 5.2 节的推论 2 可直接得出. □

推论 1. 若处理器上的调度策略是下列策略之一:

RMS,EDF,DMS(deadline-monotonic scheduling),FPS(fixed-priority scheduling),FIFO(first-in first-out),

则多处理器上的任务集合的可调度性是可判定的.

证明:可以将 RMS,EDF,DMS,FPS,FIFO 等调度策略不会在行为自动机里从位置 *ready* 到位置 *running* 的迁移条件上编写为形如 $c_i-c_j\sim c$ 或 $c_i-d_j\sim c(c_i,c_j\in C,d_j\in D,c$ 是自然数)的时间约束.例如对 EDF 调度策略,当前时刻离截止时间越近的任务优先级越高,行为自动机里从位置 *ready* 到位置 *running* 的迁移条件上的约束可编写为形如 $d_i-d_j\sim c(d_i,d_j\in D,c$ 是自然数)的原子公式的合取. □

5.2 行为自动机的判定性证明

文献[30]证明了如果两个迁移系统的可达性都是可判定的,则其并行组合也是可判定的.在本文的模型中,环境自动机的本质是时间自动机,可达性是可判定的[5].因此,利用环境自动机和行为自动机网络的并行组合来分析可调度性的关键在于证明行为自动机的可达性是可判定的,这也是本节讨论的主要内容.直接分析行为自动机的可达性十分困难,本节采用的方法是将其规约到一类可达性问题是已知可判定的自动机上.为了研究行为自动机的内部结构,首先需要去掉行为自动机中的位置层次结构.下面首先证明行为自动机的可达性判定问题可以规约到带特殊限制的 Suspension 自动机的可达性判定问题.首先给出带特殊限制的 Suspension 自动机的定义.

定义 5(带特殊限制的 Suspension 自动机). 带特殊限制的 Suspension 自动机是具有如下形式的八元组(*L*,

$Sync^o$,L_0,C,D,$flow$,E,inv),其中:

- L,$Sync^o$,L_0 的定义同定义 1;
- C,D 的定义同定义 3;
- $flow$:$L\to\prod_{i=1}^{|C|}k_i$ 是计时器增长速率的确定赋值函数,其中,k_i=1 或 0,满足同定义 3 中第 8 个子句的(a),(b),(c)相同的 3 条限制;
- $E\subseteq L\times[Sync^o]\times 2^{C\cup D}\times\Phi_0(C)\cup\Phi(D)\times L$ 定义了迁移关系.对任意迁移 $e_i=(l_i,sync_i^o,\lambda_i,\delta_i,l_i')$,满足如下限制:
  - (a) $c_i\in\lambda_i$ 当且仅当 $d_i\in\lambda_i$ 且满足 $f_c(c_i)=d_i$;
  - (b) 如果 $(c_i=M_{c_i})\in\delta_i$,则 $flow(l_i)(c_i)=1$ 且$(c_i:=0)\in\lambda_i$ 且 $flow(l_i')(c_i)=0$;
  - (c) 如果 $flow(l_i)(c_i)=0$,则$\delta_i\cap\Phi_0(c_i)=\varnothing$;
  - (d) 若$\exists c_i,c_j(i\neq j)$,$flow(l_i)(c_i)=1$ 且 $flow(l_i')(c_j)=1$,则:
    - 1) $(c_i=M_{c_i})\in\delta_i$;或者
    - 2) $(c_i=M_{c_i})\notin\delta_i$ 且 $v_{l_i}(c_j)=0$;
- inv:$L\to\Phi_0(C)\cup\Phi_0(D)$是位置上的不变量函数.

使用带特殊限制的 Suspension 自动机来编码(即模拟)行为自动机,把后者的可达性判定问题规约到前者的可达性判定问题.如果前者是可判定的,则后者一定可判定.在证明编码过程中,先定义同时活跃位置集合及其操作.由于在行为自动机中 *xor* 和 *and* 类型位置的存在,其同时处于活跃状态的位置不唯一,因此,使用位置配置集合 $conf_A$ 表示行为自动机 A 中当前同时处于活跃状态的位置集合,并可根据以下规则递归计算得出($conf_A$ 中的元素表示为 $conf_{Ai}$):

(1) $Root_A\in conf_{Ai}$;

(2) 如果位置 $l\in conf_{Ai}$ 且$\sigma(l)=xor$,则$\exists!l'\in\rho(l),l'\in conf_{Ai}$;

(3) 如果位置 $l\in conf_{Ai}$ 且$\sigma(l)=and$,则$\forall l'\in\rho(l),l'\in conf_{Ai}$.

在行为自动机 A 中,如果 $e\in E$,谓词 $source(e)$和 $target(e)$分别表示迁移 e 的源位置和目的位置,因此,对在 $conf_{Ai}$ 和 $conf_{Aj}$ 之间的任意迁移 $e\in E$,应该满足 $source(e)\in conf_{Ai}$ 和 $target(e)\in conf_{Aj}$.

定理 2. 行为自动机的可达性是可判定的,如果其所对应的带特殊限制的 Suspension 自动机的可达性也是可判定的.

证明:由于行为自动机中的有限个离散变量的值域是有限的,所以该变量集合的存在不影响行为自动机的可达性.对任一离散变量集合为空的 $2n$ 维行为自动机 A=(L,$Sync$,L_0,ρ,σ,C,D,$flow$,E,inv)都能够根据如下规则编码为一个对应的带特殊限制的 $2n$ 维 Suspension 自动机 A'=(L',$Sync^o$, L_0' ,C',D',$flow'$,E',inv'):

(a) 位置和同步事件.A 中的同时活跃位置集合 $conf_A$ 中的每个元素 $conf_{Ai}$ 对应 A'中的一个位置 $l_i'\in L'$.如果 $conf_{Ai}$ 中的所有基本位置都是 A 中的初始位置,则该 $conf_{Ai}$ 所对应的 l_i' 也是 A'中的初始位置.A 和 A'的事件集合 $Sync^o$ 相同;

(b) 计时器集合 C 和时钟集合 D.A 中的计时器集合 C 和时钟集合 D 分别对应 A'中的计时器集合 C'和时钟集合 D',且满足:

- (1) $|C|=|C'|$,$|D|=|D'|$;
- (2) $\forall c_i\in C,d_i\in D,f_c(c_i)=d_i,\exists c_i'\in C',d_i'\in D',f_c'(c_i')=d_i'$;

(c) $flow$ 和 inv 函数.对任意的 $conf_{Ai}\in conf_A$ 和其对应 A'中的位置 $l_i'\in L'$,l_i' 上的 $flow'$和 inv'函数定义为 $flow'(l_i')=\wedge_{l_i\in conf_{Ai}}flow(l_i),inv'(l_i')=\wedge_{l_i\in conf_{Ai}}inv(l_i)$.

(d) 迁移边.考虑下面两种情况:

- (1) 如果在 A 中有一条从位置 l_{ki} 到位置 l_{kj} 的迁移 e_{ki} 与一组从位置集合 L_m 到位置集合 L_n(L_m,$L_n\in L$)的迁移集合 $E_m\subseteq E$ 同步,且 $l_{ki}\in conf_{Ai},L_m\subseteq conf_{Ai},l_{kj}\in conf_{Aj},L_n\subseteq conf_{Aj}$($conf_{Ai}$,$conf_{Aj}\in conf_A$,$l_{ki}$,$l_{kj}\in L^b$,$L_m$,$L_n\subseteq L$),则以上所有同步迁移在 A'中对应一条迁移 e_i',满足 $source(e_i')=conf_{Ai}$ 且 $target(e_i')=$

$conf_{Aj}$ 且 $\lambda_{e_i'}=\wedge\lambda_{e_{ki}}\wedge_{e_k\in E_m}\lambda_{e_k}$ 且 $\delta_{e_i'}=\wedge\delta_{e_{ki}}\wedge_{e_k\in E_m}\delta_{e_k}$,并消去所有内部事件标记 $sync_i^e\in Sync^e$;

(2) 如果 A 中没有迁移与从位置 l_{ki} 到位置 l_{kj} 的迁移 e_{ki} 同步($l_{ki}\in conf_{Ai}$,$l_{kj}\in conf_{Aj}$),则 A'中对应的迁移 e_i' 满足 $source(e_i')=conf_{Ai}$ 且 $target(e_i')=conf_{Aj}$ 且 $\lambda_{e_i'}=\lambda_{e_{ki}}$ 且 $\delta_{e_i'}=\delta_{e_{ki}}$,同时保持相同的外部事件标记 $sync_i^o\in Sync^o$.

通过上面的编码,A 的每个状态 $q_i=(conf_{Ai},v_{conf_{Ai}})$在 A'中存在一个对应的状态 $q_i'=(l_i',v_{l_i'}')$,其中,$l_i'=conf_{Ai}$.如果 A 的一个状态 $q_j=(conf_{Aj},v_{conf_{Aj}})$可由 q_i 到达,则 q_j 在 A'中的对应状态也可由 q_i' 到达.因此,如果带特殊限制的 Suspension 自动机的可达性是可判定的,则行为自动机的可达性是可判定的. □

为了证明带特殊限制的 Suspension 自动机的可达性是可判定的,继续将其规约到可达性是可判定的一类时间自动机:带有界减法的时间自动机[12].由于篇幅限制,带有界减法的时间自动机的详细定义见文献[12].

定理 3. 带特殊限制的 Suspension 自动机的可达性是可判定的.

证明:证明思路是根据如下规则,用 $2n+1$ 维带有界减法的时间自动机 $A'=(L',Sync^o,L_0',X,E',inv')$来编码 $2n$ 维的带特殊限制的 Suspension 自动机 $A=(L,Sync^o,L_0,C,D,flow,E,inv)$:

(a) 位置和同步事件.令$\kappa=\{0,\perp,\perp'\}$.$A'$的位置集合满足 $L'\subseteq(L\times\kappa^{\{1,\ldots,n\}})\cup L^T$,其中,$|L^T|=|L|$.因此,$A'$的位置或者形如 $l_i^T\in L^T$ 或者形如(l,f).其中,$l\in L$,f 是计时器集合中每个计时器到κ的值域的映射.A'和 A 有相同的外部事件集合 $Sync^o$;

(b) 时钟集合 X 和不变量函数.A 中的计时器集合 C 和时钟集合 D 分别对应 X 的两个不相交的子集 X_1 和 X_2,满足$|C|=|X_1|$,$|D|=|X_2|$,$X=X_1\cup X_2\cup x_0$.A'中的位置(l_i,f)和 A 中的位置 l_i 具有相同的不变量函数;

(c) 迁移.对 A 中迁移 $e_i=(l_i,sync_i^o,\lambda_i,\delta_i,l_j)$考虑如下几种情况:

(1) 如果$\forall c_i\in C$,$flow(l_i)(c_i)=0$ 且 $flow(l_j)(c_i)=0$,根据定义 5 的第 3 子句,C 中所有计时器的值都为 0.因此,A'中对应的迁移为 $e_i'=((l_i,f),sync_i^o,\lambda_{i'},\delta_{i'},(l_j,g))$,其中,$\forall c_i\in C$,$f(c_i)=g(c_i)=0$ 且$\lambda_{i'}=\lambda_i$ 且$\delta_{i'}=\delta_i$.

(2) 如果$\forall c_i\in C,flow(l_i)(c_i)=0$ 且$\exists!c_m\in C,flow(l_j)(c_m)=1$,则 A'中对应的迁移为 $e_i'=((l_i,f),sync_i^o,\lambda_{i'},\delta_{i'},(l_j,g))$,其中,$\forall c_i(i\neq m)\in C$,$f(c_i)=f(c_m)=g(c_i)=0$ 且 $g(c_m)=\perp'$且$\lambda_{i'}=\lambda_i$ 且$\delta_{i'}=\delta_i$.

(3) 如果$\exists!c_m\in C$,$flow(l_i)(c_m)=1$ 且$\exists!c_n\in C$,$flow(l_j)(c_n)=1(n\neq m)$,有下列两种情况:

1) $(c_m=M_{c_m})\in\delta_i$ 且 $c_m\in\lambda_i$: 则 A' 中对应两条顺序迁移 $e_i'=((l_i,f),sync_i^o,\lambda_{i'},\delta_{i'},l_j^T)$,$e_i''=(l_j^T,\lambda_{m'},\delta_{m'},(l_j,g))$,其中,$\lambda_{i'}=\lambda_i\cup_{f(c_j)\neq 0}\{x_{1j}:=x_{1j}-M_{c_m}\}\cup\{x_0:=0\}$ 且$\delta_{i'}=\delta_i$ 且$\lambda_{m'}=\varnothing$且$\delta_{m'}=\{x=0\}$且 $g(c_m)=0$ 且 $g(c_n)=\perp'$.$\forall c_i\in\lambda_i(i\neq m,i\neq n)$,$g(c_i)=0$,且$\forall c_i\notin\lambda_i(i\neq m,i\neq n)$,$g(c_i)=f(c_i)$;

2) $(c_m=M_{c_m})\notin\delta_i$ 且 $c_m\notin\lambda_i$:A'中对应的迁移为 $e_i'=((l_i,f),sync_i^o,\lambda_{i'},\delta_{i'},(l_j,g))$,其中,$\lambda_{i'}=\lambda_i$ 且$\delta_{i'}=\delta_i$ 且 $g(c_m)=\perp$且 $g(c_n)=\perp'$.$\forall c_i\in\lambda_i(i\neq m,i\neq n)$,$g(c_i)=0$;$\forall c_i\notin\lambda_i(i\neq m,i\neq n)$,$g(c_i)=f(c_i)$;

(4) 如果$\exists!c_m\in C,flow(l_i)(c_m)=1$ 且$\forall c_j\in C,flow(l_j)(c_j)=0$,$A'$中对应的迁移为 $e_i'=((l_i,f),sync_i^o,\lambda_{i'},\delta_{i'},(l_j,g))$,其中,$\forall c_j(j\neq m)\in C$,$f(c_j)=g(c_j)=g(c_m)=0$ 且$\lambda_{i'}=\lambda_i$ 且$\delta_{i'}=\delta_i$.注意,所有在$\lambda_{i'}$和$\delta_{i'}$中出现的计时器必须用 X_1 中的对应时钟进行替换.

在用带有界减法的时间自动机 A'对带特殊限制的 Suspension 自动机 A 编码后,A 中的每个状态 $q_i=(l_i,v_{l_i}(C\cup D))$ 在 A'中都有一个对应的状态 $q_i'=((l_i,f),\mu(X_1\cup X_2))$.其中:如果 $f(c_i)=0$,则 $v_{l_i}(c_i)=0$;如果 $f(c_i)=\perp'$,则 $v_{l_i}(c_i)=\mu(x_{1i})$;如果 $f(c_i)=\perp$,则 $v_{l_i}(c_i)\neq\mu(x_{1i})$.并且在任何情况下,$\forall d_j\in D$,$v_{l_i}(d_j)=\mu(x_{2j})$.注意,$v_{l_i}(c_i)\neq\mu(x_{1i})$ 和 $f(c_i)=\perp$不会影响 A 和 A'之间的状态对应关系,因为迁移上没有形如 $c_i-c_j\sim c$ 的时间限制,根据子句$(c)(3)$,x_{1m} 的值将被减法置回 c_m 的值.因此,如果 A 的一个状态 $q_j=(l_j,v_{l_j}(C\cup D))$ 能够由状态 q_i 到达,则在 A'中存在一个对应的状态 $q_j'=((l_j,f),\mu(X_1\cup X_2))$也能由 q_i 在 A'中对应的状态到达.因此,带特殊限制的 Suspension 自动机能够被带有界减法的时间自动机编码,其可达性也是可判定的. □

推论 2. 行为自动机的可达性是可判定的.

推论 3. 若处理器上某种调度策略的实现需要比较任务(任务实例)之间已经流逝了的执行时间之间的大

小关系,或者需要比较一个任务(任务实例)的已经流逝了的执行时间与另一个任务(任务实例)的已经流逝了的相对截止时间之间的大小关系,则其可调度性是不可判定的.

证明:限于篇幅,简要给出其证明思路:如果假设成立,则必然存在形如 $c_i-c_j\sim c$ 或 $c_i-d_j\sim c(c_i,c_j\in C,d_j\in D,c$ 为自然数)的时间约束.而在其对应的带有界减法的时间自动机中,无法使用时钟来正确编码以上时间约束.一种可能的解决方法是将计时器值编码在位置里[29],但是我们无法使用有限位置数来保存计时器每次可能停止计数的当前值.因此在这种情况下,该类问题的调度性是不可判定的. □

5.3 符号状态操作

为了分析建模分布式系统任务执行语义的环境自动机和行为自动机网络中的 *error* 子位置是否可达,本文采用 on-the-fly 动态构造自动机网络后继符号状态技术[24],以避免完全构造该自动机网络的状态空间所造成的状态爆炸.在该技术中,如何表示和操作符号状态是关键.DBM 矩阵(difference bounded matrix)[31,32]可以有效表示和操作所有时钟增长率都为 1 的时间自动机中的凸时间区域,但是在行为自动机和环境自动机网络中,存在增长率为 0 的计时器.因此,为了使 DBM 能够操作行为自动机和环境自动机网络中的状态时间区域,重新定义 DBM 上的时间流逝、求交和重置这 3 类操作.为了表述方便,本节将时钟和计时器统称为计时器,因为时钟也是一种特殊的计时器,其值一直以速率 1 增长.

假设有 k 个计时器 $x_1,x_2,\ldots,x_k$,其 DBM 是一个 $k+1$ 维矩阵.对任意 $1\leqslant i\leqslant k,D_{i0}=(k,0)$表示计时器 x_i 的严格上界值 k,即 $x_i<k;D_{i0}=(k,1)$表示计时器 x_i 的非严格上界值 k,即 $x_i\leqslant k$.同理,D_{0i} 表示计时器 x_i 的严格或非严格下界值;对任意的 i 和 j,D_{ij} 表示 x_i-x_j 的严格或非严格上界值.为了使一个 DBM 和一个时钟时间域一一对应,因此必须将 DBM 化为标准形式,即 $\forall 0\leqslant i,j,l\leqslant k,D_{ij}\leqslant D_{ij}+D_{jl}$.为求后继状态的时间区域,定义如下 3 个 DBM 操作:

- 求交(∧):对两个标准的 DBM D 和 D',定义交集 D'',其 $D''_{ij}=\min(D_{ij},D'_{ij})$.求交操作完成后,首先检查所得到的 D''所表示的时间区域是否为空,若不为空,则化为标准形式;
- 时间流逝(↑):使用 $D\uparrow$表示 DBM D 的时间流逝操作.在此过程中,任意计时器 x_i 满足 $\dot{x}_i=1$,其 D_{0i} 值保持不变,D_{i0} 值变为∞;任意计时器 x_j 满足 $\dot{x}_j=0$,其 D_{0j} 值保持不变,D_{j0} 值保持不变;任意计时器 x_m 和 x_n 满足 $\dot{x}_m=1$ 且 $\dot{x}_n=1$,其 D_{mn} 和 D_{nm} 值保持不变;任意计时器 x_s 和 x_t 满足 $\dot{x}_s=0$ 且 $\dot{x}_t=0$,其 D_{st} 和 D_{ts} 值保持不变;任意计时器 x_i 和 x_j 满足 $\dot{x}_i=1$ 且 $\dot{x}_j=0$,其 D_{ij} 值变为∞,D_{ji} 值保持不变.然后再对所得到的 D 标准化;
- 重置(λ:=0):假设需要重置的计时器集合为λ.给定 DBM D,对所有计时器 $x_i\in\lambda$,把所有的 D_{0i} 和 D_{i0} 的值置为(0,1);任意计时器 x_m 和 x_n 满足 $x_m\in\lambda$且 $x_n\in\lambda$,把 D_{mn} 和 D_{nm} 值置为(0,1);任意计时器 x_i 和 x_j 满足 $x_i\in\lambda$且 $x_j\notin\lambda$,把 D_{ij} 置为 D_{0j} 的值,把 D_{ji} 置为 D_{j0} 的值.

因此,DBM 数据结构的空间复杂度为 $O((k+1)^2)$,每个操作的时间复杂度为 $O(k^3)$.

6 实例研究及性能评估

本节使用图 1 所示的任务结构模式说明使用行为自动机对分布式多处理器系统上任务的执行语义建模过程.任务 $T_1\sim T_6$ 的参数见表 1.CPU_1 上的 3 个任务对应如图 3 所示的行为自动机,其中,为了方便看图,一些迁移条件被省去,如从位置 $ready_{11}\sim running_{11}$ 上的约束.事实上,这些约束取决于该处理器上具体的调度策略,并且任务实例状态值 *suspend* 和 *released* 被分别略写为 *s* 和 *r*.任务 T_1 和 T_2 由于是接收外部事件流,因此同步事件标记分别为 $sync_1^o$ 和 $sync_2^o$.任务 T_2 和 T_3 之间是复杂触发关系,因此在位置 $running_{20}$ 到 $suspend_2$ 的迁移和 $suspend_3$ 到 $ready_{30}$ 的迁移上标记有相同内部同步事件 $sync_3^e$,用于建模复杂触发关系.任务 T_3 和 T_5 是在不同处理器之间的任务复杂触发关系且具有传输延迟(该传输延迟已由图 2(d)建模),任务 T_3 和该延迟模块在相同外部事件 $sync_3^o$ 上同步以发送事件流,任务 T_5 和该延迟模块在外部事件 $sync_5^o$ 上同步以接收事件流.由于篇幅限制,任务 T_4,T_5 和 T_6 的行为自动机略去.

Table 1 Task parameters (ms)

表 1 任务参数设置 (毫秒)

Task	Type	CPU	C	D	P	Offset	Buffer size	Event source	DL
T_1	Periodic	CPU_1	4	12	10	0	0	Environment	0
T_2	Periodic	CPU_1	2	7	10	1	1	Environment	0
T_3	Aperiodic	CPU_1	1	9	10	0	0	T_2	0
T_4	Sporadic	CPU_2	1	3	3	0	0	Environment	0
T_5	Sporadic	CPU_2	1	4	10	0	0	T_3	1
T_6	Aperiodic	CPU_2	2	4	5	1	1	T_4	1

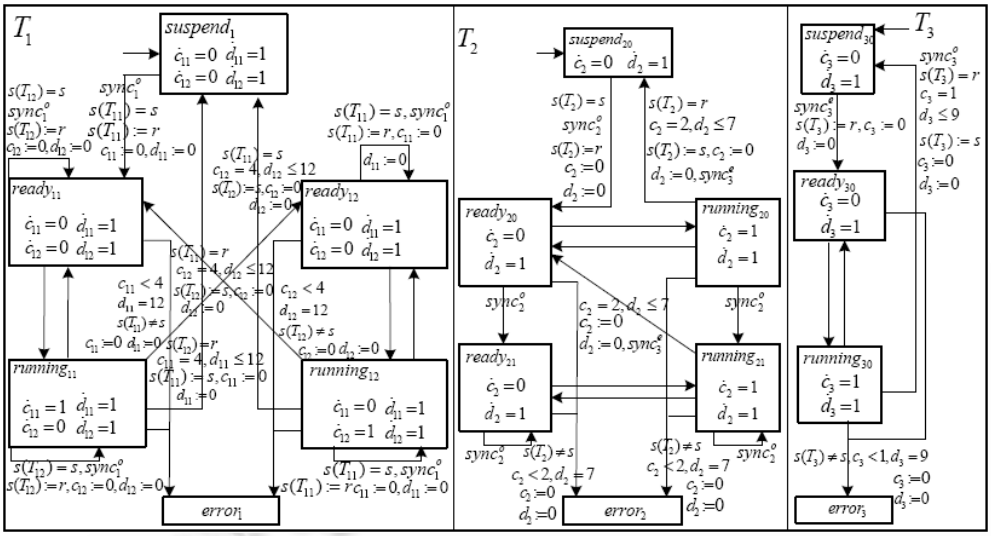
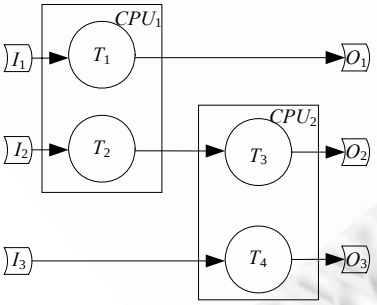


Fig.3 Action automata for CPU_1

图 3 对 CPU_1 上任务建模的行为自动机

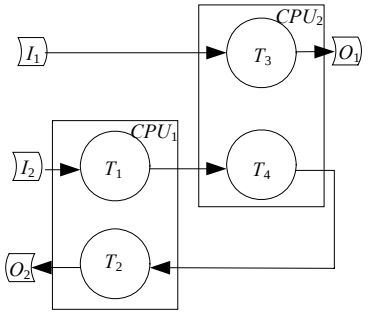
根据本文提出的方法,基于 Eclipse 平台已经开发实现了一个分布式系统调度分析的原型工具 SCT.用户只需通过输入任务参数及选择系统结构模型,把自动机模型的建立交给软件后台处理,从而工具能够返回给用户调度性结论及任务的最坏响应时间,并且支持通过步进或连续仿真任务执行过程.此外,SCT 还提供了画图功能,能够支持用户自定义的任务外部事件流到达方式,从而扩大了适用范围.为了验证 SCT 工具分析的精确性,首先实现了文献[21]中的 3 个带缓冲区的比较用例,所得到的任务最坏响应时间与用 UPPAAL 所得到的最坏响应时间完全一致.另外,为了进一步比较分析精确,本节建立如图 4 和图 5 所示的两个比较用例(时间单位均为 ms).



Input event streams	I_1: periodic (P=30) I_2: periodic (P=15) I_3: periodic (P=[25,…,70])
Scheduling policy	CPU_1: FP preemptive, CPU_2: FP preemptive
Execution times	T_1: 12, T_2: 2, T_3: 3, T_4: 8
Priorities	Priority T_1: high, priority T_2: low Priority T_3: low, priority T_4: high

Fig.4 Specification of benchmark 1

图 4 对比用例 1



Input event streams	I_1: periodic (P=80) I_2: periodic (P=20)
Scheduling policy	CPU_1: FP preemptive, CPU_2: FP preemptive
Execution times	C_1: 3, C_2: 2, C_3: [2,…,18], C_4: 2
Priorities	Priority T_1: high, priority T_2: low Priority T_3: high, priority T_4: low

Fig.5 Specification of benchmark 2

图 5 对比用例 2

与 UPPAAL 及 Holistic 技术[17]对比分析的结果如图 6(a)和图 6(b)所示.

制约自动机理论在调度领域广泛使用的原因在于其较弱的扩展能力.由于其本质是穷尽搜索,当任务数增多时,状态空间呈指数倍增加.本文着重研究使用自动机理论对复杂分布式实时系统进行调度分析的可行性分析,希望在将来的工作中能够使用效率更高的一些数据结构来缩短分析过程的时间.另外,本文发现,在任务集合中存在由外部环境直接触发的周期任务时,把它们外部环境自动机合并成一个自动机能够有效减少动态构造的节点数,缩短分析时间.表 2 给出了 4 个任务时的一个例子.当该 4 个任务都是周期任务时,将其 4 个外部环境自动机合并成一个自动机后,分析时间变为约原来的 1/6,动态生成的节点数变为约原来的 1/5(本文所得所有性能数据均是在 Intel core2 CPU 1.8GHz,内存容量为 1GB 的 Windows 环境中测得).因此,SCT 在当周期任务在任务集中所占比例越大时,分析时间越短.

Table 2 Test duration comparison by merging environment automata

表 2 合并环境自动机前后分析时间对比

Task	C	D	P	Merging environmen automata?	Sched ing policy	Node generated	Testing duration
T_1	2	8	8	No	RMS	5 560	4.063s
T_2	1	9	9	No	EDF	5 547	3.266s
T_3	2	10	10	Yes	RMS	1 179	0.625s
T_4	2	12	12	Yes	EDF	1 192	0.5s

为了进一步测试 SCT 性能,在 TIMES[13]的建模能力范围内与其对比:单处理器上调度策略为 EDF,任务集中所有任务为周期任务且互相独立,个数为 n($5\leqslant n\leqslant 9$).在区间[1,30]之间取随机整数作为每个任务 T_i 的周期值 P_i.任务 T_i 的初始相位 O_i 和最坏执行时间 C_i 分别在区间$[0,0.2P_i]$和$[0.5P_i/n,1.5P_i/n]$中取随机整数.两者的时间性能如图 6(c)所示.TIMES 1.3 BETA 版在 EDF 下所支持的任务个数最大为 8 个,SCT 在 $n\in[5,8]$内性能接近于 TIMES.如果任务集合中存在 $n/2$ 个任务复杂触发关系,分析时间显著缩短.因为当存在任务复杂触发关系时,被触发任务不存在对应的外部环境自动机,因此减少了 on-the-fly 分析过程中将要生成的节点数.因此,SCT 的另一个特点是:若任务集中复杂触发关系越多,分析时间反而更短.

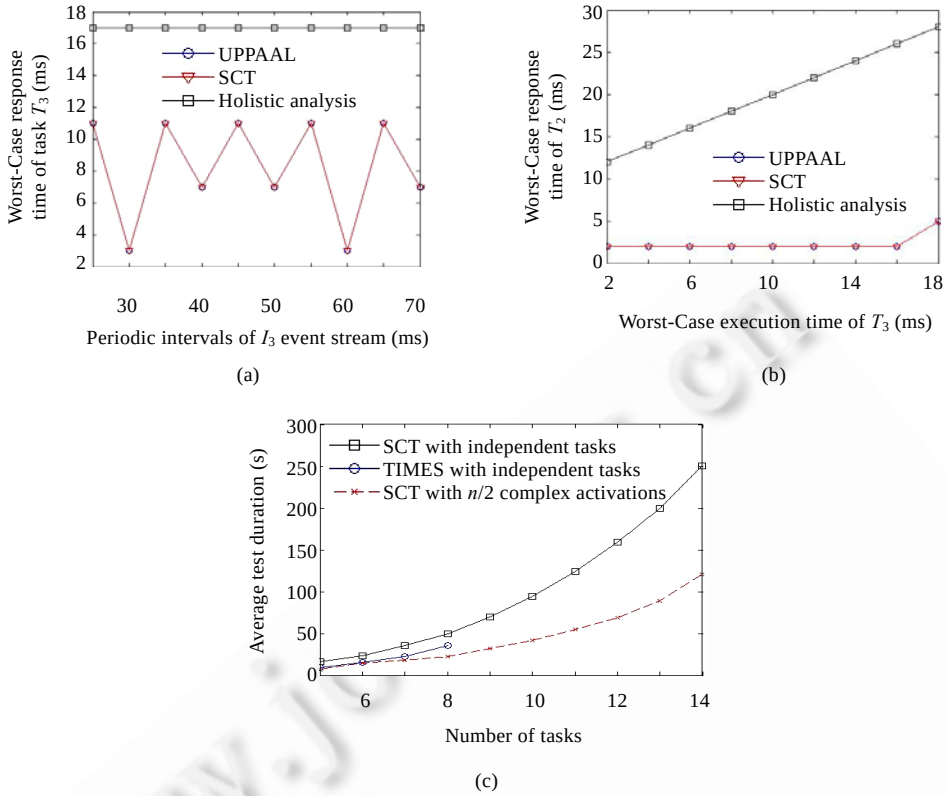


Fig.6 Comparison with other techniques

图 6 与其他工具及技术的效果对比

7 结 论

本文针对分布式实时系统的复杂结构,提出了使用自动机理论精确分析系统中任务集的调度性问题.通过把任务的外部事件到达流和传输媒质延迟建模为环境自动机、把任务执行语义建模为行为自动机,从而形成了自动机网络来对复杂系统建模.然后证明了行为自动机可达性的判定性问题,丰富了 Suspension 自动机可达性可判定的条件.通过该判定条件,明确了本文所提方法的适用范围.进一步研究效率更高的时间区域的符号数据结构及操作,以减少分析时间和提高工具的可扩展能力,是今后我们需要进一步研究的重点问题.

References:

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

[2] Audsley N, Burns A, Richardson M, Tindell K, Wellings AJ. Applying new scheduling theory to static priority preemptive scheduling. Software Engineering Journal, 1993,8(5):284−292. [doi: 10.1049/sej.1993.0034]

[3] Lehoczky JP. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proc. of the 11th RTSS. IEEE Press, 1990. 201−209. [doi: 10.1109/REAL.1990.128748]

[4] Clarke EM, Grumberg O, Peled DA. Model Checking. Cambridge: MIT Press, 1999.

[5] Alur R, Dill DL. A theory of timed automata. Theoretical Computer Science, 1994,126(2):183−235. [doi: 10.1016/0304-3975(94)90010-8]

[6] Fehnker A. Scheduling a steel plant with timed automata. In: Proc. of the 6th RTCSA. Los Alamitos: IEEE Press, 1999. 280−286. [doi: 10.1109/RTCSA.1999.811256]

[7] Cassez F, Laroussinie F. Model checking for hybrid systems by quotienting and constraints solving. In: Emerson EA, Sistla AP, eds. Proc. of the 12th CAV. LNCS 1855, Heidelberg: Springer-Verlag, 2000. 373−388. [doi: 10.1007/10722167_29]

[8] Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, Olivero A, Sifakis J, Yovine S. The algorithmic analysis of hybrid systems. Theoretical Computer Science, 1995,138(1):3−34. [doi: 10.1016/0304-3975(94)00202-T]

[9] Behrmann G, David A, Larsen KG. A tutorial on UPPAAL. In: Proc. of the 4th Int'l School on Formal Methods for the Design of Computer, Communication, and Software Systems. LNCS 3185, Heidelberg: Springer-Verlag, 2004. 200−236.

[10] Hendriks M, Verhoef M. Timed automata based analysis of embedded system architectures. In: Proc. of the 20th IPDPS. IEEE Press, 2006. 268−275. [doi: 10.1109/IPDPS.2006.1639422]

[11] Cassez F, Larsen KG. The impressive power of stopwatches. In: Palamidessi C, ed. Proc. of the 11th CONCUR. LNCS 1877, Heidelberg: Springer-Verlag, 2000. 138−152.

[12] Fersman E, Krcal P, Pettersson P, Wang Y. Task automata: Schedulability, decidability and undecidability. Information and Computation, 2007,205(8):1149−1172. [doi: 10.1016/j.ic.2007.01.009]

[13] Amnell T, Fersman E, Mokrushin L, Pettersson P, Wang Y. TIMES: A tool for schedulability analysis and code generation of real-time systems. In: Larsen KG, Niebert P, eds. Proc. of the 1st FORMATS. LNCS 2791, Heidelberg: Springer-Verlag, 2004. 60−72. [doi: 10.1007/978-3-540-40903-8_6]

[14] Krcal P, Stigge M, Wang Y. Multi-Processor schedulability analysis of preemptive real-time tasks with variable execution times. In: Raskin J, Thiagarajan PS, eds. Proc. of the 5th FORMATS. LNCS 4763, Heidelberg: Springer-Verlag, 2007. 274–289. [doi: 10.1007/978-3-540-75454-1_20]

[15] Fersman E, Mokrushin L, Pettersson P, Wang Y. Schedulability analysis of fixed-priority systems using timed automata. Theoretical Computer Science, 2006,354(2):301−317. [doi: 10.1016/j.tcs.2005.11.019]

[16] Hamann A, Henia R, Racu R, Jersak M, Richter K, Ernst R. SymTA/S-Symbolic timing analysis for systems. In: WIP Proc. of the 16th ECRTS. Catania: IEEE Press, 2004. 17−20.

[17] Tindell K, Clark J. Holistic schedulability analysis for distributed hard real-time systems. Microprocessing and Microprogramming, 1994,40(2-3):117−134. [doi: 10.1016/0165-6074(94)90080-9]

[18] Richter K, Ernst R. Event model interfaces for heterogeneous system analysis. In: Proc. of the DATE. IEEE Press, 2002. 506−513. [doi: 10.1109/DATE.2002.998348]

[19] Thiele L, Chakraborty S, Naedele M. Real-Time calculus for scheduling hard real-time systems. In: Proc. of the IEEE Int'l Symp. on Circuits and Systems. Washington: IEEE Press, 2000. 101−104. [doi: 10.1109/ISCAS.2000.858698]

[20] Le Boudec JL. Application of network calculus to guaranteed service networks. IEEE Trans. on Information Theory, 1998,44(3): 1087−1096. [doi: 10.1109/18.669170]

[21] Perathoner S, Wandeler E, Thiele L, Hamann A, Schliecker S, Henia R, Racu R, Ernst R, Harbour MG. Influence of different system abstraction on the performance analysis of distributed real-time systems. In: Proc. of the 7th EMSOFT. New York: ACM Press, 2007. 193−202. [doi: 10.1145/1289927.1289959]

[22] Henzinger TA, Nicollin X, Sifakis J, Yovine S. Symbolic model checking for real-time systems. Information and Computation, 1994, 111(2):193−244. [doi: 10.1006/inco.1994.1045]

[23] Tripakis S, Yovine S. Analysis of timed systems using time-abstracting bisimulations. Formal Methods in System Design, 2001, 18(1):25−68. [doi: 10.1023/A:1008734703554]

[24] Larsen KG, Larsen F, Pettersson P, Wang Y. Compact data structures and state-space reduction for model-checking real-time systems. Int'l Journal of Time-Critical Computing Systems, 2003,25(2):255−275. [doi: 10.1023/A:1025132427497]

[25] Behrmann G, Larsen KG, Pearson J, Weise C, Wang Y. Efficient timed reachability analysis using clock difference diagrams. In: Halbwachs N, Peled D, eds. Proc. of the 11th CAV. LNCS 1633, Heidelberg: Springer-Verlag, 1999. 341−353. [doi: 10.1007/3-540-48683-6_30]

[26] Sha L, Rajkumar R, Lehoczky JP. Priority inheritance protocols: An approach to real-time synchronization. IEEE Trans. on Computers, 1990,39(9):1175−1185. [doi: 10.1109/12.57058]

[27] Larsen KG, Pettersson P, Wang Y. Compositional and symbolic model-checking of real-time systems. In: Proc. of the 16th RTSS. Pisa: IEEE Press, 1995. 76−89. [doi: 10.1109/REAL.1995.495198]

[28] McManis J, Varaiya P. Suspension automata: A decidable class of hybrid automata. In: Dill DL, ed. Proc. of the 6th CAV. LNCS 818, Heidelberg: Springer-Verlag, 1994. 105−117. [doi: 10.1007/3-540-58179-0_47]

[29] Henzinger TA, Kopke PW, Puri A, Varaiya P. What's decidable about hybrid automata? In: Proc. of the 27th ACM Symp. on Theory of Computing. New York: ACM Press, 1995. 373−382. [doi: 10.1145/225058.225162]

[30] Henzinger TA. Hybrid automata with finite bisimulations. In: Fülöp Z, Gécseg F, eds. Proc. of the 22nd ICALP. LNCS 944, Heidelberg: Springer-Verlag, 1995. 324−335. [doi: 10.1007/3-540-60084-1_85]

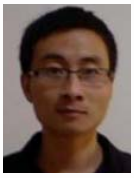
[31] Dill DL. Timing assumptions and verification of finite-state concurrent systems. In: Sifakis J, ed. Proc. of the Int'l Workshop on Automatic Verification Methods for Finite State Systems. LNCS 407, Heidelberg: Springer-Verlag, 1989. 197−212. [doi: 10.1007/3-540-52148-8_17]

[32] Alur R. Timed automata. In: Halbwachs N, Peled D, eds. Proc. of the 11th Int'l Conf. on Computer Aided Verification. LNCS 1633, Heidelberg: Springer-Verlag, 1999. 8−22. [doi: 10.1007/3-540-52148-8_17]

桂盛霖(1983－),男,重庆人,博士生,CCF学生会员,主要研究领域为实时系统调度性分析,自动机理论.

于淼(1985－),男,硕士生,主要研究领域为形式化技术,实时系统.

罗蕾(1968－),女,教授,博士生导师,主要研究领域为实时系统,操作系统技术,形式化建模技术.



徐建华(1985－),男,硕士生,主要研究领域为实时系统建模和分析.

李允(1971－),男,博士,副教授,主要研究领域为实时系统,形式化建模技术,实时自适应技术.