基于 Petri 网的模型检测研究*

蒋屹新+, 林 闯, 曲 扬, 尹 浩

(清华大学 计算机科学与技术系,北京 100084)

Research on Model-Checking Based on Petri Nets

JIANG Yi-Xin+, LIN Chuang, QU Yang, YIN Hao

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62782587, E-mail: yxjiang@csnet1.cs.tsinghua.edu.cn, http://www.tsinghua.edu.cn

Received 2003-08-27; Accepted 2004-02-03

Jiang YX, Lin C, Qu Y, Yin H. Research on model-checking based on Petri nets. *Journal of Software*, 2004,15(9):1265~1276.

http://www.jos.org.cn/1000-9825/15/1265.htm

Abstract: Model-Checking is a formal verified technique to check on whether a computing model, by searching the model state spaces, satisfies a given property described by an appropriate temporal logic. The main drawback of model checking, the explosion problem of state spaces, is mainly caused by concurrence and the interleaving semantics used to represent any sequences of possible actions. In this paper, the correlative model-checking theory and techniques based on Petri Nets are investigated in detailed, especially about the following problems, i.e. partial order reduction and partial order semantics techniques based on the state reachability graph, Buchi automata method, state cohesion method based on Petri Nets, and symbolic and parametriesed model-checking techniques based on system symmetries. Moreover, the key idea and our main researching work in the future are listed. With the gradual improvement of reducing techniques of state space and optimization of model-checking algorithm, model-checking technique has been successfully applied to verify communication protocols and complex hardware logic circuits, and also takes on a wide application prospect in other fields.

Key words: temporal logic; Petri net; state space; model checking

摘 要: 模型检测是关于系统属性验证的算法和方法.它通常采用状态空间搜索的方法来检测一个给定的计算模型是否满足某个用时序逻辑公式表示的特定属性.系统模型的状态空间的爆炸问题是模型检测所面临的主要问题,其主要原因是系统自身的并发特性和状态变迁的语义交织.对基于 Petri 网的模型检测理论和验证技术进行了较为详细的研究,着重探讨了基于 Petri 网状态可达图的偏序简化和偏序语义技术、基于自动机的模型

* Supported by the National Natural Science Foundation of China under Grant No.90104002 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2001AA112080 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2003CB314804 (国家重点基础研究发展规划(973))

作者简介: 蒋屹新(1972－),男,博士生,主要研究领域为随机 Petri 网,模型检测,计算机网络及安全;林闯(1948－),男,博士,教授,博士生导师,主要研究领域为系统性能评价,计算机网络,随机 Petri 网,逻辑推理模型;曲扬(1978－),男,硕士生,主要研究领域为随机 Petri 网,工作流,计算机网络;尹浩(1974－),男,博士,主要研究领域为随机 Petri 网,流媒体,计算机网络.

检测算法、基于 Petri 网的状态聚合法以及基于系统对称性的参数化和符号模型检测技术,并给出了研究思路以及未来所要进行的重点研究工作.模型检测技术已在通信协议和硬件系统的验证等领域得到成功应用,并且随着各种状态空间简化技术和模型检测算法的不断优化,其在其他应用领域也展示出广泛的应用前景.

关键词: 时序逻辑;Petri 网;状态空间;模型检测

中图法分类号: TP301 **文献标识码**: A

模型检测[1]是关于系统属性验证的算法和方法.它提供了一个完整的系统属性的验证框架,通常采用状态空间搜索的方法来检测一个给定的计算模型是否满足某个用时序逻辑公式表示的特定属性.由于模型检测的方法在通信协议和硬件系统的验证等方面取得了成功,该方法近年来得到了广泛关注;并且,随着计算机硬件速度的提高和算法的不断优化,模型检测的方法展示出广泛的应用前景.

与模型检测密切相关的基本概念是时序逻辑,在形式化验证技术中,常用时序逻辑公式描述系统需要被验证的属性.Pnueli 首次用线性时序逻辑 LTL 描述系统的并发特性[2];Clarke 则首次将分支时序逻辑 CTL 引入到系统的并发特性分析之中[3].时序逻辑可以方便、准确地描述并发系统的重要性质,如安全性(safety)和活性(liveness).安全性是指系统的部分正确性、互斥性和无死锁性,用于说明"坏事情永远都不会发生";活性是指系统的终止性、无活死锁性、保证服务性和响应性等,用于说明"好事情最终会发生".

Petri 网是一种重要的数学工具,它能有效地对信息系统进行描述和建模,并对系统的并发性、异步性和不确定性具有很强的动态分析能力.Petri 网研究的系统行为特性主要包括状态的可达性、位置的有界性、变迁的活性、初始状态的可逆达、标识之间的可达等.作为数学工具,Petri 网可以通过建立系统的可达图或者状态方程来描述系统的行为.其中,可达图是 Petri 网模型的主要分析方法之一.

模型检测的复杂性主要依赖于系统状态空间大小.系统模型的状态空间爆炸问题是形式化验证系统属性所面临的主要问题,这主要是因为系统自身的并发特性和为表示各种可能变迁序列而引入的交织语义.因此,在具体的应用领域中,为有效地简化系统的状态空间,模型检测问题必须与其他方法和技术相结合,如有效的状态空间搜索算法、变迁交织语义的消除以及系统状态的简化抽象技术等.

1 基本概念

1.1 Petri网的基本概念

定义 1. 一个三元组 $N=(S,T;F)$是一个 Petri 网[4],当且仅当:

① $S\cup T\neq\varnothing$(网非空);$S\cap T=\varnothing$(二元性);

② $F\subseteq(S\times T)\cup(T\times S)$(流关系仅在于 S 与 T 的元素之间);

③ $dom(F)\cup cod(F)=S\cup T$(没有孤立元素),

其中,S 是位置集,T 是变迁集,F 是弧集;$dom(F)=\{x|\exists y,(x,y)\in F\}$;$cod(F)=\{x|\exists y,(y,x)\in F\}$.在图形表示上,$S$ 用圆圈表示,T 用长方形或者黑线表示;元素之间的流关系由带箭头的弧表示.

定义 2. 一个六元组$\Sigma=(S,T;F,K,W,M_0)$是一个位置/变迁(P/T)系统,当且仅当:

① $(S,T;F)$是一个网,S 的元素是位置,T 的元素是变迁;

② $K:S\rightarrow N^+\cup\{\infty\}$是位置容量函数($N^+$为正整数集合);

③ $W:F\rightarrow N^+$是弧权函数;

④ $M_0:S\rightarrow N$(N 为非负整数集合)是初始标识(marking),且满足:$\forall s\in S:M_0(s)\leq K(s)$.

定义 3. 令$\Sigma=(S,T;F,K,W,M_0)$是一个 P/T 系统.

① 函数 $M:S\rightarrow N$ 叫做Σ的标识,当且仅当$\forall s\in S:M_0(s)\leq K(s)$;

② 一个变迁 $t\in T$ 在标识 M 下是可实施的,当且仅当$\forall s\in S:W(s,t)\leq M(s)\leq K(s)-W(t,s)$;

③ 如果 $t\in T$ 在标识 M 可实施,则实施 t 后产生的新标识 M'为:$\forall s\in S,M'(s)=M(s)-W(s,t)+W(t,s)$.系统标识 M 经过 t 的实施得到的新标识 M'可以表示成 $M[t>M'$.

定义 4. 一个 P/T 系统Σ的可达标识集$[M_0>$表示Σ的最小标识集合,如果$[M_0>$满足:

① $M_0\in[M_0>$;

② 如果 $M_1\in[M_0>$且有 $t\in T$ 使 $M_1[t>M_2$,那么 $M_2\in[M_0>$.

定义 5. 若$\forall M\in[M_0>$,存在 $M'\in[M>$使得 $M'[t>$,则称 $t\in T$ 是活的(liveness).若$\forall t\in T,t$ 都是活的,则称该 P/T 系统Σ是活的.若$\forall M\in[M_0>$,存在 $t\in T$ 使得 $M[t>$,则称 P/T 系统Σ在 M 下不死锁;否则,P/T 系统Σ在 M 死锁(deadlock).因此,一个 P/T 系统Σ是活的的必要条件是:Σ在任何可达标识 M 都不死锁.

一个 P/T 系统Σ的可达图则是以标识为结点的图,其弧线由 T 元素标注.可达图的分析方法存在的主要困难是状态空间的爆炸问题,即模型的状态空间随着实际系统的规模增大而呈指数增长.为解决状态空间爆炸的问题,很多学者提出了高级 Petri 网 HLPN、随机高级 Petri 网 SHLPN 等非常有效的新模型[4].这些模型采用对称的方法对系统模型进行压缩表示,能有效地简化系统的状态空间.

1.2 时序逻辑概述

时序逻辑是一种重要的描述和验证并发系统特性的形式化工具.在形式化验证技术中,常用时序逻辑公式描述系统需要被验证的属性.时序逻辑 CTL*[5]是一种具有极强描述能力的逻辑,它能形式化地描述状态变迁系统或 Kripke 结构中状态之间的变迁关系.通过指定 Kripke 结构中的某一初始状态作为一棵树的根,我们可以将 Kripke 结构展开成具有无限结构的计算树.这种计算树表示了从该初始状态所有可能的执行序列.因此,时序逻辑 CTL*能对计算树的有关属性进行形式化的描述和验证.

CTL*公式由时态操作符和路径限定符组成.路径限定符描述计算树中的分支结构,包括 A(所有计算路径)和 E(某一计算路径)两种限定符.时态操作符主要描述计算树中某一计算路径的属性,包括 X(next time),F(eventually),G(always),U(until)和 R(release).CTL*中具有两种类型的公式:状态式(在某一特定的状态为真)和路径式(在某一特定的路径上为真).CTL*的语义可解释为 Kripke 结构.

定义 6. 设 AP 为原子命题集合,定义在 AP 集上的 Kripke 结构是一个三元组 $M=(S,R,L)$,其中:

① S 是有限状态集合;

② $R\subseteq S\times S$ 是完备的变迁关系集,即 $\forall s\in S,\exists s^*\in S,(s,s^*)\in R$;

③ $L:S\rightarrow Power(AP)$是一个函数,表示在状态 s 下所有为真的原子命题集,用 $L(s)$表示.

在 Kripke 结构 M 中,从状态 s 开始的路径π是一非空的无限状态序列集$\pi=s_0\ s_1\ s_2\ldots$,其中 $s=s_0$,且$\forall i\geq 0,(s_i,s_{i+1})\in R$.若 f 是状态式,则$(M,s)\models f$表示 f 在 Kripke 结构中的状态 s 为真;若 f 是路径式,则$(M,\pi)\models f$表示 f 在 Kripke 结构的计算路径π上为真.

时序逻辑 CTL*包括两类:分支时序逻辑 CTL 和线性时序逻辑 LTL.其主要差别在于如何处理展开 Kripke 结构所对应的计算树分支.在 CTL 中,时态运算符限定于从一个给定的状态开始的所有可能路径上;而在 LTL 中,时态运算符仅限定于描述从一个给定的状态开始的某条路径上的事件.

1.3 模型检测概述

设 Kripke 结构 $M=(S,R,L)$表示有限状态的并发系统,时序逻辑公式 f 表示系统需要验证的属性,而并发系统的初始状态为 S_0,$S_f=\{s|s\in S\wedge(M,s)\models f\}$表示所有满足公式 f 的状态集,则模型检测的问题可描述为检验$S_0\subseteq S_f$是否成立[6].若成立,则系统需要验证的属性为真;否则为假.

基本的 CTL 模型检测算法是一种状态标记算法,该算法的时间复杂度是 $O((|S|+|R|)\cdot|f|)$,其中$|f|$是 CTL 公式 f 的子公式数.基本的 LTL 模型检测算法[7]则是一种基于 Tableau 构造的算法(Tableau 是根据 LTL 公式构造的图).LTL 公式 f 所描述的系统属性是否成立,可以通过检测在 Tableau 中是否存在一条满足相应条件的计算路径来进行验证,该算法的时间复杂度为 $O((|S|+|R|)\cdot 2^{|O(f)|})$.CTL*模型检测算法则综合了 CTL 模型检测算法的状态标记技术和 LTL 的 Tableau 方法,算法的时间复杂度为 $O((|S|+|R|)\cdot 2^{|O(f)|})$[8].

一种较为重要的模型检测方法是符号模型检测.在模型检测算法的最初实现中,变迁关系被表示为显式的邻接表.这种算法具有很大的局限性,特别是当并发系统的状态数爆炸增加时.McMillan 则在 Clarke 的模型检测算法的基础上,通过利用二叉判定树 OBDD[9]表示状态变迁图或 Kripke 结构,可对状态空间数超过 10^{20} 的模型

进行检测[10].随后,各种改进的 OBDD 算法的推出使得现在对状态空间数超过 10^{120} 的模型进行检测成为现实[11,12].另外,基于命题逻辑 SAT(satisfiability)程序的符号模型检测技术是近年来一个较新的研究方向[13,14],该方法能够有效地改善符号模型检测的可伸缩性,具有代表性的方法是有界模型检测[14].

2 模型检测的 Büchi 自动机算法

2.1 基本的Büchi自动机算法

Büchi 自动机是ω有限自动机的一种,它包含若干初始状态和若干接受状态,每条弧上都标记有命题 p.如何将 LTL 公式转化为 Büchi 自动机可参见文献[15].

Petri 网状态可达图描述的是模型所有可能的行为(即状态序列),若要验证模型是否满足某条性质(用 LTL 公式 p 描述),通常检测是否存在一个可能的行为满足这条性质的反性质.因此,在进行模型检测时,需生成被检测性质的反性质的 Büchi 自动机,并构造状态可达图和 Büchi 自动机的交集(乘积图).乘积图可用有向图表示,其中每个节点由(e_i,e_j)表示,e_i 是状态可达图中的状态,e_j 是 Büchi 自动机中的状态.

若将乘积图看成一个有向图,则模型检测的问题等价于检测乘积图中是否包含一个从初始状态可达的最大强连通分量,且在该强连通分量中包含了一个接受状态.对于属于安全性一类的性质来说,等价于检测乘积图中没有一条从初始节点到接受节点的路径.对于活性一类的性质来说,等价于检测乘积图中没有由初始节点可达的包含接受节点的环.一种检测强连接分量的有效算法是 Tarjan 的 DFS 算法[16];另一种更为有效的算法是双重 DFS 算法[17],该算法在搜索失败时可以给出反例.

在此基础上改进的动态模型检测算法是 On-the-Fly 算法[16].该算法事先仅需先构造被验证属性的 Büchi 自动机 B.算法在计算状态可达图和自动机 B 的乘积图的同时,利用自动机 B 去引导状态可达图的动态构造.这种算法在找到被验证属性反例之前,可以仅需构造状态可达图的一小部分状态空间,从而避免了对整个状态空间的搜索.因此,这种状态空间搜索策略被形象地称为 on-the-fly.目前,LTL 的 On-the-Fly 算法已在一些主要的模型检测工具中得以实现,如 SPIN[18],PROD[19]和 PEP[20].

2.2 状态聚合法

状态聚合法同样也是一种基于 Büchi 自动机的模型检测算法,其简化工作是在状态可达图构造之前完成的.对状态聚合方法的有关算法和性能我们将在后续的研究工作予以深入展开.

模型的状态可达图随着模型的增长而呈指数增长,因此,根据状态可达图和描述被验证属性的 Büchi 自动机所生成乘积图的空间复杂度将非常大.实际上,许多性质的验证无须考察模型的全部状态空间.状态聚合法的核心思想就是找到与所要验证的性质无关的状态,把它们压缩掉,从而得到简化的状态可达图,进而减小模型检测的空间复杂度.与状态聚合法相关的两个基本概念是无关变迁和相关变迁.

定义 7. 对于某个模型 N 和所要验证的性质 c,模型 N 中的一个变迁称为无关变迁,当且仅当与该变迁相邻的位置不在性质 c 对应的线性时序逻辑公式中出现;反之,称为相关变迁.

在状态可达图中,与无关变迁相连的状态集构成状态聚合子集.状态聚合法进行模型检测的步骤如下:

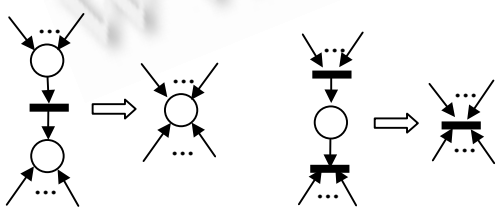

Fig.1 Reduced rules of subnets

图 1 子网化简规则范例

(1) 根据所要验证的性质将模型中的变迁划分为无关变迁和相关变迁.

(2) 对无关变迁及其相邻位置构成的子网进行化简.在找到无关子网之后,可按照简化规则(如图 1 所示)对每个连通子网重复进行简化,直到子网无法再化简为止.

(3) 由简化模型构造压缩的状态可达图.

(4) 由简化的状态可达图和所要验证的性质对应的自动机构造简化的乘积图(空间复杂度更小),可通过检测乘积图的最大连通分量的可达性来得到相关性质的模型检测结果.

状态聚合法与后述的偏序方法都是能简化模型的状态可达图,但二者处于模型检测过程的不同阶段.前者的化简工作是在状态可达图构造之前完成,后者的化简工作是在状态可达图构造时完成,因此是否可以把状态聚合法与偏序方法结合起来,进一步减小模型检测的空间复杂性,是一个非常有意义的问题.

3 状态空间的偏序技术

在异步系统中,系统状态变迁的实施是交织进行的,这是引起 Petri 网状态空间爆炸的主要原因.例如,在一个并发系统中,若某个状态可对 n 个变迁同时进行点火,则需要 $n!$种交织执行顺序才能表示所有的变迁点火次序.通过引入对这种变迁交织执行顺序的表示,即迹(trace)的概念[21],可以消除这种缺陷,进而达到对状态空间简化的目的.迹是一组可实施的变迁顺序集,同一迹中的任意两个不同的变迁顺序之间可以通过相邻的可实施变迁之间的轮换而获得.

偏序技术(partial order technique)是一种基于迹的状态化简技术[22],可实施变迁之间的偏序关系准确地表示了系统变迁之间的相互独立或依赖关系.这种技术能有效地减少模型检测过程中需要被搜索的状态空间数量.偏序技术主要有两类:其一是偏序简化技术(partial order reduction),它通过消除迹中的冗余变迁序列来对状态可达图进行化简,主要有顽固集技术[23~25]、睡眠集技术[24]和覆盖步图技术[26,27].与偏序简化技术密切相关的两个基本概念是变迁的独立性(independence)和不可见性(invisibility).另一类是偏序语义技术(partial order semantics),其主要思想是通过重用位置/变迁网的并发和冲突概念直接表示可实施变迁中的偏序关系,主要是基于网展开(nets unfolding)和进程(process)这两种技术[28,29].

偏序技术能够验证相当大的一类时态属性.图 2 和图 3 分别是一个简单的位置/变迁网及其状态可达图,可利用该模型来简单介绍偏序技术.

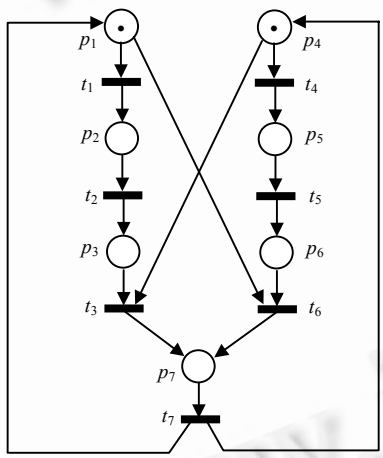

Fig.2 An example of Petri nets

图 2 Petri 网实例

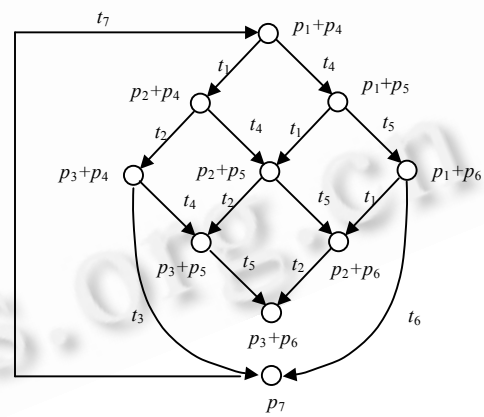

Fig.3 State reachability graph

图 3 状态可达图

3.1 偏序简化技术

3.1.1 稳固集(persistent set)技术

稳固集是定义在每个状态上的可实施变迁集.在每个状态中,应该能够选择与其他可实施变迁子集相互独立的变迁子集予以点火实施.例如,设 T 是状态 s 的稳固集,若变迁序列 $t.w$ 从状态 s 可实施,其中 $t\in T$,w 是 T 中可实施变迁的序列,则 t 和 w 重排后的变迁序列同样在状态 s 是可实施的,并且可在变迁实施后到达同样的目的状态,这是因为稳固集中的可实施变迁之间是相互独立的.

基于对 Petri 网结构的分析,研究人员提出了各种计算可访问状态中稳固集的高效算法[25~27].而顽固集[23](stubborn set)是稳固集算法中一种较为复杂的技术.每个状态的顽固集是一组与其他变迁相独立的可实施

变迁的子集,可以通过分析变迁之间的冲突和因果关系来分析它们之间的依赖关系.顽固集中的任何变迁必须满足:若该变迁是可实施的,则所有与之在结构上存在冲突的变迁也应被选择;若该变迁不可实施,则选择该变迁的输入位置中没有足够 token 数的某一位置,同时该输入位置的所有输入变迁也应全部入选.

显然,可实施变迁和不可实施变迁的输入位置的选取能够显著地影响顽固集的大小,如何有效地选取构造最小顽固集可参见文献[23],Valmari 在该文中指出:顽固集包含关系的验证可以在线性时间内完成,因为其等价于在状态图中搜索一个最小强连接分量.例如,在状态 p_2+p_4 存在两个顽固集:$\{t_2\}$和$\{t_4,t_3,t_2\}$,顽固集$\{t_2\}\subseteq\{t_4,t_3,t_2\}$.因此,状态$p_2+p_4$的顽固集应选取$\{t_2\}$,即选择可实施变迁 t_2,而应推迟选择可实施变迁 t_4.利用顽固集技术,图 3 的状态可达图可简化为如图 4 所示.

3.1.2 睡眠集(sleep set)技术

睡眠集技术[24]充分利用了可实施变迁之间的非冲突性,能够有效地减少变迁实施的语义交织.每个状态均与一个睡眠集相关联,一个状态的睡眠集表示一组可实施(firing)但并不值得实施的变迁集,这是因为这些变迁的目标可达状态可以通过其他变迁序列到达.通过这种方式,睡眠集技术可以消除状态可达图中的部分变迁弧.

睡眠集能在构造状态图的同时以 on-the-fly 的形式构造,与系统的初始状态相关联的睡眠集是空集.设某状态 s 的睡眠集为 SP,则在标记状态 s 的后续状态时,只需对非睡眠集中的可实施变迁进行标记,而忽略睡眠集 SP 中的可实施变迁.例如,图 5 的睡眠集的状态简化图对应于图 3 中的状态可达图,其中状态 p_1+p_5 和 p_1+p_6 对应的睡眠集为$\{t_1\}$,状态 p_2+p_5 和 p_2+p_6 对应的睡眠集则为$\{t_2\}$.

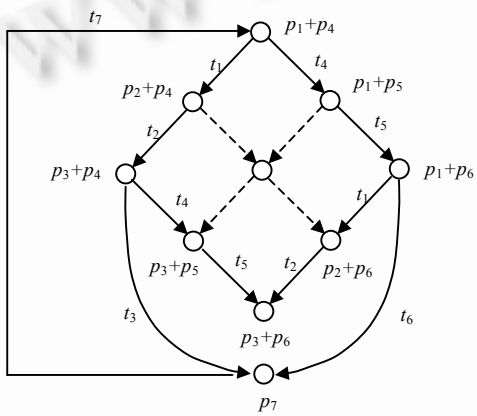

Fig.4 Reduced graph based on stubborn set

图 4 顽固集的状态简化图

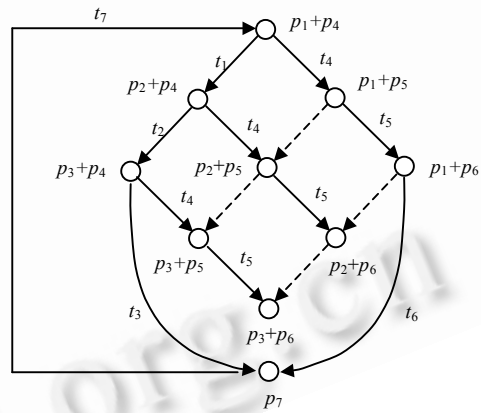

Fig.5 Reduced graph based on sleep set

图 5 睡眠集的状态简化图

在顽固集基础上,综合睡眠集技术可以进一步对状态空间进行简化,消除部分中间状态,如图 6 所示.

3.1.3 覆盖步图(covering step graph)技术

覆盖步图 CSG[26]能有效地减少变迁实施的语义交织.它充分利用变迁之间的独立性关系,使得在状态标识的每一步将所有相互独立的可实施变迁同时进行.因此,CSG 表示的是状态可达图的变迁实施序列,其中弧表示状态标识的每一步能同时实施的所有变迁,CSG 的结点仅表示状态可达图中的部分可达标识,消去了一些中间可达状态.根据变迁之间的独立性关系,CSG 能以 on-the-fly 的形式构造,这样可以进一步简化系统的状态空间.构建 CSG 的算法非常类似于状态可达图的生成算法,只是状态变迁的实施必须遵循如下原则:

① 所有的可实施变迁应被同时实施;

② 一个可实施的变迁可考虑进行合并,若直接与此变迁冲突的变迁也可被实施;

③ 在进行合并的可以实施变迁中,相互存在冲突的变迁可有多步合并步骤.

因此,利用变迁的实施规则,可构造与图 3 的状态可达图对应的 CSG 简化状态图,如图 7 所示,其中的变迁 t_2 和 t_5 可以合并成$\{t_2,t_5\}$.

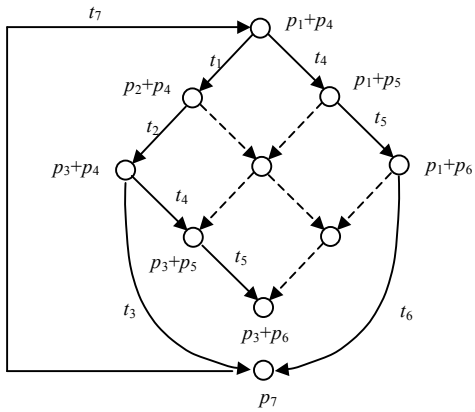

Fig.6 Reduced graph combined stubborn and sleep set

图 6 综合顽固集和睡眠集的简化状态图

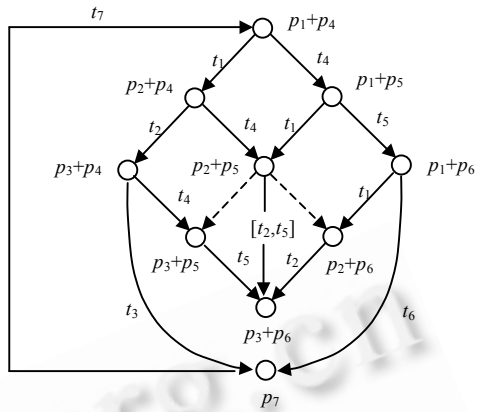

Fig.7 Reduced graph based on CSG

图 7 CSG 的状态简化图

CSG 技术还具有一个微妙的性质,它能隐含检测几种基于路径的属性,如死锁和变迁的活性检测.与稳固集技术相比,它并不需要一些前提条件,这种算法技术特别适合对可见变迁的属性验证.

值得指出的是,尽管如上几种简化后的状态图(如图 4~图 7 所示)均有不同的表示,但均能覆盖图 3 中状态可达图所表示的迹:$[t_1t_2t_4t_5]$,$[[[t_1t_2t_3+t_4t_5t_6]t_7]^*t_1t_2t_4t_5]$,$[[t_1t_2t_3+t_4t_5t_6]t_7]^\infty$.

3.1.4 覆盖步图和顽固集的综合简化技术

P.O. Ribert 在覆盖步图 CSG 和顽固集技术的基础上,提出了一种偏序简化技术,即 PSG(persistent steps graph)技术[27].PSG 技术能有效地模拟和改善基于覆盖步图 CSG 和顽固集的状态简化技术.

基于覆盖步图 CSG 和顽固集技术,可以导出 $P_{min}SG$,$P_{max}SG$ 和 HPSG(hybrid PSG)这 3 类 PSG 图.可以证明:任一 $P_{min}SG$ 图是某类最小顽固集图($P_{min}G$)的压缩表示,并且有任意的 $P_{max}SG$ 图的状态空间均小于相对应的 CSG 图.$P_{min}SG$ 和 $P_{max}SG$ 的生成算法的主要区别在于顽固集计算函数的不同,前者的顽固集计算函数返回最小的顽固集,后者返回可实施变迁的完全集.而 HPSG 技术则是一种基于 $P_{min}SG$ 和 $P_{max}SG$ 的启发式状态简化算法.

表 1 列出了这几种简化技术的对比结果.其中模型 1 是包含 300 个站点的 Milner 调度器模型[30],模型 2 是著名的 8 位哲学家就餐问题,模型 3 是 K.Jenson 提出分布在 10 个站点的分布式数据库算法[31],模型 4 则是 10 个站点的令牌环网模型[32].从表中的结果可以看出,覆盖步图 CSG、顽固集 PG 和 HPSG 技术均能有效地简化 Petri 网模型的状态空间,而 HPSG 的简化效果更优于纯粹的覆盖步图 CSG 和顽固集 PG 简化技术.

Table 1 Comparison of different partial order reduction techniques

表 1 不同状态空间的偏序简化技术对比

No.	Petri nets model	No reduction	$P_{min}G$	CSG	HPSG
1	Scheduler: 300	$2^n \times n \approx 6 \times 10^{92}$	1 394	301	301
2	Philosopher: 8	103 681	233	31 231	227
3	Data base: 10	196 831	191	31	31
4	Token ring: 10	35 840	99	52	51

3.2 偏序语义技术

Petri 网有两种主要偏序语义技术,即进程[28]和网展开[29]技术.网展开是一种偏序分支计算模型,而进程技术描述的是系统偏序行为的线性时间计算模型.分支进程技术(branching process technology)则是基于网展开计算和分支进程图计算的一种偏序语义技术[33],其主要思想是通过重用位置/变迁网的并发和冲突概念直接表示可实施变迁中的偏序关系.值得指出的是,对一个高度并发的系统,展开分支进程可以获得系统可达标识集的简化表示;并且,分支进程的无环特性使得其能有效地对属性进行描述和验证.

进程是一种表示 Mazurkievicz 迹的标记因果网(labelled causal net),进程中的每个位置最多有一个输入变迁,因此,进程中的任一对变迁只存在两种关系:因果依赖关系(变迁的实施存在先后次序关系)和独立性(变迁可

并发执行),而不存在冲突关系.分支进程则允许冲突存在,因此,一个分支进程中不存在环,每个位置最多有一个输入变迁,但是可以有多个输出变迁.

分支进程中的结构(configuration)是一组关于因果依赖关系向下封闭的变迁集,并且在该变迁集中的任意两个变迁之间不存在冲突关系.而分支进程中的割集(cut)则是一组并发可达的位置集,即在该集合的位置之间不存在任何因果依赖关系和冲突关系.进程的割集是稳定的,当且仅当割集中的位置相对应的标记的所有可实施变迁能被该割集的输出变迁所表示.位置变迁网的展开是一个分支进程,并且满足:原始网的每个可达标识至少被一个稳定的割集表示 1 次.

因此,分支进程图可定义为图:$G=(V,E)$,其中 G 的结点集 V 是稳定的分支进程集,而 G 的弧则是相对应的位置变迁网中所标记的变迁.任何结点的分支进程都不包括可观察变迁,分支进程图中的弧表示可实施的可观察变迁.分支进程图特别适合于描述和验证 LTL$\neg_X$ 公式,即无 X 时序算符的 LTL.

3.3 偏序技术的有关结论

通过对如上几种偏序技术的讨论分析,可有如下结论:所有的偏序技术都是基于系统事件之间的依赖关系(包括冲突和因果关系),它在简化状态空间的同时还保证了模型对相当大一类属性的验证能力,从普通的死锁到各种时序逻辑公式,如 LTL$\neg_X$ 公式.

综合利用顽固集、睡眠集和覆盖步图这几种偏序简化技术,能有效地简化系统状态空间.例如,图 6 中顽固集和睡眠集技术的结合以及基于覆盖步图 CSG 和顽固集的 PSG 图.稳固集技术可以结合睡眠集技术对状态可达图中独立变迁的实施进行序列化;覆盖步图则是在变迁实施的每一步将所有相互独立的可实施变迁同时进行,它充分利用状态变迁之间的独立性关系;分支进程技术则为了定义变迁实施的偏序关系,重新定义了 Petri 网,从而消除了直接表示的语义交叉问题.

以上几种技术,除分支进程技术以外,其他都可运用 on-the-fly 算法来检测模型的有关属性.这是因为,分支进程技术是直接表示变迁实施的偏序关系,因此,系统属性的验证必须在构造完分支进程图之后才能进行.上述技术已在一些系统中实现.如 SPIN[18]中的稳固集和睡眠集技术;PROD[19]中实现的稳固集、OBDD 和对称技术;而 PEP[20]中则实现了分支进程技术的一个原形.

4 符号和参数化方法

Petri 网可达图的分析方法存在一个主要困难:由于可达图都是根据初始标识计算出来的,如果初始标识的位置标记数量是参数化的,那么针对参数的每一个赋值都可能要计算一个完全不同的可达图,这将导致对参数化的系统模型分析变得非常困难,参数化和符号可达图的方法可被用来解决参数化模型检测问题.

4.1 符号可达图(symbolic reachability graph)

符号可达图 SRG 是 WN(well-formed)网的状态可达图的简化表示,其主要思想是利用系统内在的对称性来获得可达状态的压缩表示.WN 网[34]是一种着色网.通过引入颜色函数的语法定义,WN 网提供了一个利用对称性简化状态表示的复杂性和减少状态空间的模型框架,WN 网可以形式化地定义如下:

定义 8. Well-Formed 网是元组 $\Sigma=(P,T,Pre,Post,Inh,pri,C,cd)$,其中:

① P 是一个有限位置集,T 是一个有限变迁集;

② Pre,$Post$ 和 Inh 是弧函数;

③ $pri:T\rightarrow N^+$是与变迁 t 关联的优先级向量.在默许情况下,$\forall\ t\in T,pri[t]=0$;

④ $C=\{C_1,C_2,\ldots,C_n\}$是基本的颜色类型集,颜色类型集由静态子类构成:$C_i=C_i^1\cup C_i^2\cup\ldots\cup C_i^k$,索引 h 被定义为:$\forall\ h<i\leq n$,颜色类型 C_i 是有序的;$\forall\ 0<i\leq h$,颜色类型 C_i 是无序的;

⑤ cd 是将网的位置和变迁映射到颜色域上的函数,变迁的颜色域可以绑定一个限制函数.

与符号标识密切相关的两个概念是颜色排列(color permutation)和标识排列(marking permutation).利用符号标识来表示 WN 网状态空间中的等价类,可以直接构造符号可达图.在符号标识定义的基础上,符号可达图的构造还需对符号表示的唯一性和符号标识的变迁实施规则进行定义.

为保证符号表示的唯一性,首先需要引入对象变量和动态子类的概念.其次,还需引入某些约束以保证符号表示的唯一性,即符号的规范表示[35].符号规范表示的主要思想是必须保证符号标识的有序和最小性.对符号标识的变迁实施规则需要引入动态子类的分割(split)概念,利用动态子类的分割函数,可定义符号标识的三步符号实施(three-step symbolic firing)规则.因此,通过引入符号标识的规范表示定义和符号变迁实施规则后,符号可达图的构造算法基本与普通状态可达图的构造算法相同,只是采用规范的符号标识来表示初始标识,并采用符号实施规则来取代普通的变迁实施规则.

值得指出的是,符号标识图对状态可达图的简化强烈依赖于模型自身的对称性:模型对象之间的等价行为越多,则在同一等价类中的符号标识越多,从而对原始状态可达图的状态的压缩率越高.

4.2 扩展符号可达图

在高级 Petri 网 HLPN 中,通过搜索整个 HLPN 网的对称性,符号可达图 SRG 能有效地简化系统状态空间.在 WM 网理论和 SRG 的基础上,S.Haddad 提出了扩展可达图 ESRG(extended SRG)的概念[36].ESRG 是对符号可达图 SRG 的扩展定义,ESRG 能利用 HLPN 网的部分对称性来简化系统的状态空间.

S.Haddad 首先将 WM 网的变迁划分为对称子网(symmetrical subnet)和非对称子网(asymmetrical subnet),这两种子网的主要差别是变迁的类型不同.在此基础上,相关文献还对符号标记和符号变迁实施规则进行定义扩展定义.其中,共有 3 种符号变迁实施规则,即普通对称变迁实施规则(generic symmetrical firing)、实例化对称变迁实施规则(instantial symmetrical firing)和实例化非对称变迁实施规则(instantial asymmetrical firing).如何构建扩展符号可达图 ESRG 以及扩展符号可达图有关特性,可参见文献[36].

ESRG 的主要思想是:松弛静态子类中对象排列的许可条件,通过扩展对象等价类概念,以方便描述因某些对象而引起的非对称行为.因此,在 ESRG 中,符号标记可以部分展开并以实例化的形式实施,而对称变迁的实施则可以经典形式进行.与经典符号理论相比,ESRG 能更有效地简化系统状态空间.以 ESRG 为基础,S.Haddad 在文献[37]中还提出了一种新的模型检测方法,该方法能够处理系统及其有关属性的部分对称特性.然而,如何综合 ESRG 和基于对称自动机的全局分析方法是值得进一步研究的问题.

4.3 参数化可达图

与采用对称的思想压缩状态空间的方法不同,参数化可达图(parametrised reachability graph)方法的主要思想是利用状态分类来化简状态可达图,并且状态模型的表示是参数化的,例如,对一个分布式的临界资源进行访问的进程数是未知的.另外,参数化方法中的状态分类将依赖于某些特定条件是否成立.

有两类参数化的属性验证方法.一类方法是典型程序(representative program)[35,38~40],该程序是无参数的程序,其描述的属性与实例化参数模型的属性等价.典型程序不带任何参数,是参数化程序行为的一种抽象.系统属性可被视为一种特殊的程序,该程序的执行能够满足该属性.因此,可以采用相同的规范语言同时去描述程序和需要被验证的属性.最终检查属性是否成立的问题可转化为验证语言的包含问题:一个规范满足某一特定属性,当且仅当与该属性关联的程序包含该规范.

另一类方法是行为的符号表示(symbolic representation of behavior),它利用参数符号表示程序的行为,并在符号表示的基础上去验证系统的有关属性.主要有 S.German 提出的利用参数化方法去验证线性时序逻辑 LTL[41]和 I.Vernier 提出的可用于验证分支时序逻辑 CTL 的参数化可达图方法[42].

前者利用某一等价进程的行为来研究参数化程序的行为,而程序中的其他进程与该等价进程通信.算法利用自动机去描述等价进程的行为,因此,可以利用模型检测的自动机算法去验证参数化程序的属性.后者,即参数化可达图是一种很重要的方法,它表示了系统所有的可达状态,并定义了所有实例化程序的执行.该方法是基于 Petri 网的,其状态标识是参数化的.因此,有两种可能:其一,位置的 token 数是已知的;其二,位置的 token 数是未知的,但依赖于参数值.对于参数化的状态标识定义了标识之间的两种偏序关系:包含关系(included)和大于关系(superiority).文献[43]定义了参数化可达图的变迁实施规则,可以分成 3 步:

① 与一般的变迁实施规则一样,根据每个位置的入弧和出弧计算每个位置的标识;

② 当参数化标识并不能表示相同的变迁时,需要对参数化标识进行分裂;

③ 若参数标识大于其某个祖先,应该并免产生无限分支.

利用如上的变迁实施规则,可以极大地简化系统的状态空间.例如,文献[43]中的参数化互斥访问模型的状态可达图,经过简化后只有 13 个结点和 21 条边.而原始的 n 个进程的状态可达图则共有 $3n$ 条边和结点.

在参数化状态可达图的基础上,可以方便地验证系统的有关属性.如,通过分析可达状态的标识集,能验证系统的死锁、状态的不变性等属性.而有关 CTL 公式的属性验证可以参考文献[43].

4.4 可达图的完全参数化

参数化的方法在解决模型检测问题方面很具有吸引力,特别是在分析一个由许多相似部分构成的状态规模较大的系统时,它能有效地简化系统的状态空间.然而,上述的参数化方法仍具有一定的局限性.典型程序方法要受到很多条件的制约,并且在很多情况下都是不可行的;而行为的符号表示法只能处理单个参数的情况,并不能用于处理多个参数的情况,主要的困难是某些属性的验证依赖于各参数值之间的关系.

一种新的构造参数化可达图的方法是构造一种完全参数化的可达图.所谓完全参数化是指可达图中任何一个标识内的每一个位置的标记数目都是用变量参数来表示的.完全参数化标识的形式化定义如下:

定义 9. 一个有界的 P/T 系统$\Sigma=(S,T;F,K,W,M_0)$,假设位置集合 S 有 n 个位置,即 $S=\{s_1,s_2,\ldots,s_n\}$,$|S|=n$,对于$\forall M\in[M_0>$,$M=\{M(s_1),M(s_2),\ldots,M(s_n)\}$,如果令 $M(s_i)=x_i(i=1,2,\ldots,n)$,$x_i$ 是一个变量参数,那么称 $M=\{x_1,x_2,\ldots,x_n\}$为一个完全参数化标识.

此外,对完全参数化可达图的构造算法、状态空间表示的完备性、有效性证明以及它在模型检测和随机 Petri 网中的应用(系统马尔可夫链的稳定状态概率求解),我们将在后续的研究工作中予以深入展开.

5 结束语

本文较为详细地对基于 Petri 网的模型检测理论和验证技术进行了研究,着重探讨了基于 Petri 网状态可达图的偏序简化和偏序语义技术、基于 Büchi 自动机的模型检测算法、基于 Petri 网的状态聚合法以及基于系统对称性的参数化和符号模型检测技术,并给出了我们的研究思路以及未来所要进行的重点研究工作.

通过对基于 Petri 网的模型检测理论及相关技术的讨论,可有如下结论:基于 Petri 网的状态聚合法能有效地在状态可达图的构造之前简化模型的复杂性;偏序技术(包括稳固集、睡眠集和覆盖图)和偏序语义技术(进程和网展开技术)则提供了一种基于系统语义、且极其有效的状态空间的简化算法;基于对称和参数化的模型检测技术能处理包含大量重复对象的大型应用系统,非常适合于对系统的容错性和协议规范等问题进行验证;各种有效的实现技术,如 OBDD 和 on-the-fly 则提供了模型检测的一般框架体系.

在目前,随着各种状态空间简化技术的逐步改善,模型检测技术在工程上的应用已逐步变为现实,并展示了非常广泛的研究前景和实际运用前景.事实上,在通信协议和硬件电路系统的验证等应用领域,模型检测的理论和技术已取得了很大成功.另外,该方法在一些新的领域也逐步开始得到研究人员的广泛关注.例如,Ritchey 和 Ammann 提出的基于模型检测的网络脆弱性分析的方法[44];另外,Ramakrishnan 等人也提出了另一种使用模型检测器进行网络脆弱性分析的模型方法[45].与 Ritchey 和 Ammann 的模型相比,该模型的对象更低一层,而且这个模型的状态是无限的,所以不能够使用 SMV,SPIN 等模型检测工具,但 Ramakrishnan 等人自行开发了用于这种模型的无限状态系统的模型检测技术.

因此,如何深入展开基于 Petri 网的模型检测的相关基础理论研究和如何利用基于 Petri 网的模型检测技术来分析主机或网络系统的脆弱性,特别是对系统日志进行静态或动态分析则是我们下一步的研究重点.

References:

[1] Clarke EM, Grumberg O, Peled D. Model Checking. Cambridge: MIT Press, 2001. 35~49.

[2] Sistla AP, Clarke EM. The complexity of propositional linear temporal logics. Journal of the ACM, 1985,32(3):733~749.

[3] Clarke EM, Emerson EA, Sistla AP. Automatic verification of finite state concurrent system using temporal logical specification. ACM Trans. on Programming Language and Systems, 1986,8(2):244~263.

[4] Lin C. Computer Network and Computer System Performance Evaluation. Beijing: Tsinghua University Press, 2001 (in Chinese).

[5] Emerson EA, Halpern JY. Sometimes and Not Never revisited: On branching versus linear time. Journal of the ACM, 1986, 33(1):151~178.

[6] Girault C, Valk R. Petri Nets for System Engineering: A Guide to Modeling, Verification and Application. Springer-Verlag, 2003.

[7] Vardi MY. Linear vs. Banching tme—A complexity-theoretic perspective. In: Proc. of the 13th Annual IEEE Symp. on Logic in Computer Science. IEEE Computer Society Press, 1998. 94~405.

[8] Bhat G, Cleaveland R, Grumberg O. Efficient on-the-fly model checking for CTL. In: Proc. of the 10th Annual IEEE Symp. on Logic in Computer Science. IEEE Computer Society Press, 1995. 388~397.

[9] Bryant RE. Graph-Based algorithms for boolean function manipulation. IEEE Trans. on Computers, 1986,35(8):667~691.

[10] Burch JR, Clarke EM, McMillan KL, Dill DL, Hwang LJ. Symbolic model checking: 10^{20} states and beyond. Information and Computation, 1998,2:141~170.

[11] Burch JR, Clarke EM, Long DE, McMillan KI, Dill DL. Symbolic model checking for sequential circuit verification. IEEE Trans. on Computer-aided Design of Integrated Circuits, 1994,13(4):401~424.

[12] Burch JR, Clarke EM, Long DE. Symbolic model checking with partitioned transition relations. In: Proc. of the Int'l Conf. on Very Large Scale Integration. 1991. 49~58.

[13] Abdulla PA, Bjesse P, Eén N. Symbolic reachability analysis based on SAT-solver. In: Proc. of the 6th Int'l Conf. on Tools and Algorithm for Construction and Analysis of Systems (TACAS 2000). LNCS 1785, Springer-Verlag, 2000. 411~425.

[14] Biere A, Cimatti A, Clarke E. Symbolic model checking without BDDs. In: Proc. of the 5th Int'l Conf. on Tools and Algorithm for Construction and Analysis of Systems (TACAS'99). LNCS 1579, Springer-Verlag, 1999. 193~207.

[15] Gerth R, Peled D, Vardi M, Wolper P. Simple on-the-fly automatic verification of linear temporal logic. In: Proc. of the 15th Int'l Conf. on Protocol Specification, Testing and Verification. Warsaw, 1993.

[16] Courcoubets C, Vardi M, Wolper P, Yannakakis M. Memory-Efficient algorithms for the verification of temporal properties. Formal Methods in System Design, 1992,1:275~288.

[17] Holzmann GJ, Peled D, Yannakakis M. On nested depth first search. In: Proc. of the 2nd SPIN Workshop. AMS, 1996. 23~32.

[18] Holzmann GJ. The spin model checker. IEEE Trans. on Software Engineering, 1997,23(5):279~295.

[19] Varpaaniemi K, Hiekkanen K, Lilius J. PROD 3.2—An advanced tool for efficient reachability analysis. In: Proc. of the 9th Int'l Conf. on Computer Aided Verification (CAV'97). LNCS 1254, Springer-Verlag, 1997. 472~475.

[20] Best B, Grahlmann B. PEP—more than a Petri nets. In: Proc. of the 2nd Int'l Workshop, TACAS'96. LNCS 1055, Passau: Springre-Verlag, 1996. 397~401.

[21] Bollig B, Leucker M. Deciding LTL over Mazurkiewicz traces. In: Proc. of the IEEE 8th Symp. on Temporal Representation and Reasoning. 2001. 189~198.

[22] Alur R, Brayton RK, Henzinger TA, Qadeer S, Rajamani SK. Partial-Order reduction in symbolic state space exploration. In: Proc. of the 9th Int'l Conf. on Computer Aided Verification. LNCS 1254, Grumberg: Springer-Verlag, 1997. 340~351.

[23] Varpaanieme K. On the stubborn set method in reduced state space generation [Ph.D. Thesis]. Helsinki University of Technology, 1998.

[24] Godefroid P. Partial-Order methods for the verification of concurrent systems, an approach to the state-explosion problem. LNCS 1032, Springer-Verlag, 1996.

[25] Varpaaniemi K. Stable models for stubborn sets. In: Proc. of the CS&P'99 Workshop. Warsaw, 1999. 263~274.

[26] Vernadat F, Azéma P, Michel F. Covering step graph. In: Proc. of the 17th Int'l Conf. on Application and Theory of Petri Nets 96. LNCS 1091, Osaka: Springer-Verlag, 1996. 516~535.

[27] Ribet PO, Vernadat F, Berthomieu B. On combining the persistent sets method with the covering steps graph method. In: Proc. of the FORTE 02. LAAS Report 02388, 2002.

[28] Esparza J, Schröter C. Net reductions for LTL model-checking. In: Proc. of the 11th CHARME. LNCS 2144, Springer-Verlag, 2001. 310~324.

[29] Esparza J, Heljanko K. Implementing LTL model checking with net unfoldings. In: Proc. of the 8th Int'l SPIN Workshop on Model Checking of Software (SPIN 2001). LNCS 2057, Springer-Verlag, 2001. 37~56.

[30] Korver H, Springintveld J. A computer-checked verification of Milner's scheduler. In: Proc. of the Int'l Symp. on the Theoritical Aspect of Computer Software (TACS'94). LNCS 789, Springer-Verlag, 1994. 161~178.

[31] Jensen K. An introduction to the theoretical aspects of coloured Petri nets. In: de Bakker JW, de Roever WP, Rozenberg G, eds. A Decade of Concurrency. LNCS 803, Springer-Verlag, 1994. 230~272.

[32] Corbert JC. Evaluating deadlock detection methods for concurrent software. IEEE Trans. on Software Engineering, 1996,22(3): 161~180.

[33] Heljanko K. Combining symbolic and partial order methods for model checking 1-safe Petri nets [Ph.D. Thesis]. Helsinki University of Technology, 2002.

[34] Chiola G, Dutheillet C, Franceschinis G, Haddad S. Stochastic well-formed colored nets and symmetric modeling applications. IEEE Trans. on Computers, 1993,42(11):1343~1360.

[35] Kurshan RP, McMillan K. A structural induction theorem for processes. In: Proc. of the 8th Annual ACM Symp. on Principles of Distributed Computing. Alberta, 1989. 239~247.

[36] Haddad S, Ilié JM, Taghelit M, Zouari B. Symbolic reachability graph and partial symmetries. In: Proc. of the 16th Int'l Conf. on Application and Theory of Petri Nets. LNCS 935, Springer-Verlag, 1995. 238~251.

[37] Haddad S, Ilié JM, Ajami K. A model checking method for partially symmetric systems. In: FORTE 2000. 2000. 121~136.

[38] Miller A, Calder M. An application of abstraction and induction techniques to degenerating systems of processes. In: Proc. of the Int'l Workshop on Model-Checking for Dependable Software Intensive Systems (MCDSIS 2003). IEEE Computer Society Press, 2003.

[39] Li J, Suzuki I, Yamashita M. A new structural induction theorem for rings of temporal Petri nets. IEEE Trans. on Software Engineering, 1994,20(2):115~126.

[40] Clarke EM, Grumberg O, Jha S. Verifying parameterized networks using abstraction and regular languages. In: Proc. of the 6th Int'l Conf. on Concurrency Theory. LNCS 962, Springer-Verlag, 1995. 365~407.

[41] German S, Sistla AP. Reasoning about systems with many processes. Journal of the ACM, 1992,39(3):675~735.

[42] Vernier I. Symbolic executions of symmetrical parallel programs. In: Proc. of the 4th Euromicro Workshop on Parallel and Distributed Processing. Portugal, 1996. 327~334.

[43] Cousot P, Cousot R. Refining model checking by abstract interpretation. Journal of Automated Software Engineering, 1999,6(1): 69~95.

[44] Ritchey RW, Ammann P. Using model checking to analyze network vulnerabilities. In: Proc. of the 2000 IEEE Symp. on Security and Privacy. Oakland, 2000. 156~165.

[45] Ramakrishnan CR, Sekar R. Model-Based analysis of configuration vulnerabilities. Journal of Computer Security (JCS), 2002,10(1): 189~209.

附中文参考文献:

[4] 林闯.计算机网络和系统性能评价.北京:清华大学出版社,2001.