计算机系统形式化验证中的模型检测方法综述

彭 飞[1]，张 涛[2]，王金双[2]，赵 敏[2]
（1. 解放军理工大学指挥信息系统学院研究生 1 队，江苏 南京 210007；
2. 解放军理工大学指挥信息系统学院）

摘 要：随着计算机系统的功能日益强大，对其进行功能和正确性测试与验证的复杂度也越来越大。形式化方法作为对计算机系统进行描述与验证的重要途径，得到学术界普遍的关注与认可。文章主要介绍形式化方法，详细介绍模型检测的检测原理及其主要技术，介绍了几种典型的模型检测工具，并对它们的性能进行比较，同时研究了模型检测中普遍存在的状态爆炸问题及缩减状态方法，最后介绍模型检测的新进展。文章可为模型检测方法研究提供参考和理论支撑。

关键词：形式化方法；形式化验证；模型检测；状态爆炸
中图分类号：TP391.41 **文献标识码**：A **DOI**：10.16464/j.cnki.cn32-1289.2016.02.008

Overview of Model Checking Methods in Computer System Formal Verification

PENG Fei[1], *ZHANG Tao*[2], *WANG Jin-shuang*[2], *ZHAO Min*[2]
(1. Postgraduate Team 1 CCIS, PLAUST, Nanjing 210007, China;
2. College of Command Information Systems, PLAUST)

Abstract: As the functions of computer system become increasingly stronger, the testing and verifying of its function and correctness has become more complicated. Formal method is an important way, which can describe and verify the computer system, and achieve more attention and recognition. The basic idea of formal method was introduced, and detail information about the detection principle and the main technology of model checking were presented. Several kinds of typical model checking tools were introduced and their performances compared. At the same time, the state explosion existing generally in model checking was studied and some strategies to reduce the amount of states put forward. Finally, some new achievement was discussed. The paper can provide reference and theoretical support for study of model checking method.

Key words: formal method; formal verification; model checking; state explosion

近年来，随着计算机系统的功能和规模日益增加，其问题域也在不断加大，系统变得越来越复杂。由于系统复杂度的增加，出现错误的可能性也就更大，其安全性日益被关注。如何验证计算机系统的安全性，保证系统的可靠性，变得尤为重要。以数理逻辑为基础的形式化方法成为解决该问题的有效途径之一。形式化方法因其严格、准确等优点，一直以来受到普遍的关注与应用。

本文将介绍形式化方法，并重点介绍模型检测技术，列举几种典型的模型检测工具，分析模型检测方法面临的状态爆炸问题及其缩减状态方法，最后分析模型检测方法面临的挑战，以及新的发展成果。

收稿日期：2015-12-18；修回日期：2016-03-25
作者简介：彭 飞（1992－），女，硕士生.

1　形式化方法概述

形式化方法是用数学和逻辑的方法来描述和验证系统设计是否满足需求。它将系统属性和系统行为定义在抽象层次上，以形式化的规范语言去描述系统。形式化的描述语言有多种，如一阶逻辑，Z 语言，时序逻辑等。图 1 是形式化方法的示意图，采用形式化方法可以有效提高系统的安全性、一致性和正确性，帮助分析复杂系统并且及早发现错误。

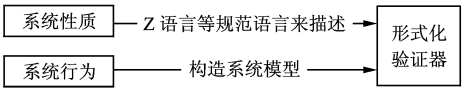


图 1　形式化方法示意图

形式化验证是保证系统正确性的重要方法，主要包括以数学、逻辑推理为基础的演绎验证(deductive verification)和以穷举状态为基础的模型检测(model checking)。

演绎验证是基于人工数学来证明系统模型的正确性。它利用逻辑公式来描述系统，通过定理或证明规则来证明系统的某些性质。演绎验证既可以处理有限状态系统，又可以解决无限状态问题。但是演绎验证的过程一般为定理证明器辅助，人工参与，无法做到完全自动化，推导过程复杂，工作量大，效率低，不能适用于大型的复杂系统，因而适用范围较窄。常见的演绎验证工具有 HOL，ACL2，PVS 和 TLV 等。

模型检测主要应用于验证并发的状态转换系统，通过遍历系统的状态空间，对有限状态系统进行全自动验证，快速高效地验证出系统是否满足其设计期望。下面将主要介绍模型检测方法的发展历史和研究现状，以及当前面临的挑战和未来发展方向等问题。

2　模型检测及相关技术

模型检测方法最初由 Clarke，Emerson 等人[1]于 1981 年提出，因其自动化高效等特点，在过去的几十年里被广泛用于实时系统[2]、概率系统[3]和量子[4]等多个领域。模型检测基本要素有系统模型和系统需满足的属性，其中属性被描述成时态逻辑公式 Φ。检测系统模型是否满足时态逻辑公式 Φ，如果满足则返回“是”，不满足则返回“否”及其错误路径或反例[5]。时态逻辑主要有线性时态逻辑 LTL(Linear Temporal Logic)和计算树逻辑 CTL(Computation Tree Logic)。

2.1　线性时态逻辑

对一个系统进行检测，重要的是对系统状态正确性要求的形式化，其中一个基本维度是时间，同时需要知道检验结果与时间维度的关系。使用线性时态逻辑 (LTL)来描述系统，可以使得系统更容易被理解，证明过程更加直截了当。

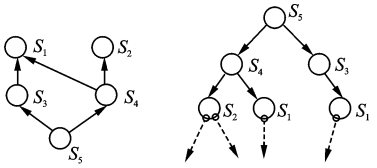
LTL 公式是一种线性时态逻辑。它在表示授权约束时，定义了无限的未来和过去，这样扩展了常用语义，并且保证了证明中判定的结果在各个时间点中都是成立的。LTL 公式用逻辑连接符和时态算子表达系统运行时状态之间的关系。LTL 的逻辑连接符包括：$\wedge$(与)，$\vee$ (或)，$\neg$(非)，$\rightarrow$(逻辑包含)，$\leftrightarrow$(逻辑对等)。时态算子包括：G(Globally)，U(Until)，F(Future)，X(neXt-time)。LTL 模型检测验证系统状态转换模型是否满足属性，使用可满足性判定，即为检测系统模型 M 中是否存在从某个状态出发的并满足 LTL 公式 $\neg\Phi$ 的路径，如果所有路径都满足 LTL 公式 Φ 则不存在有路径满足 $\neg\Phi$。使用 LTL 公式也有一定的局限性，LTL 公式只能包括全称量词[6]，对于混用了全称和存在量词的性质，一般无法用这种方法进行模型检测。

2.2　计算树逻辑

计算树即为通过将迁移系统 M 某一状态作为根，将 M 用树形结构展开表示出来，如图 2 将 S_5 为根，展开 M 状态结构图。

CTL 使用路径量词(包括:A(All),E(Exist))和时态算子(包括 F,G,X,U)对计算树属性进行形式化的描述,表示出系统的状态变化以及状态的分枝情况。

CTL 和 LTL 都有强大的表达能力。LTL 的时间定义是与路径相关的,每个时刻只有唯一的一个后继状态。LTL 可用于有重点的选择感兴趣的路径分析,并且LTL可以表达公平概念而CTL不能。但是对于一些复杂属性,如每个计算总是可能返回到初始状态,LTL 将无法描述,但是 CTL 可以。CTL 的时间定义是与状态相关的,每个状态都有多个可能的后继状态,从一个给定的状态量化分离出路径,能够断言行为的存在。CTL可以用路径量词E,而LTL 不可以;CTL 公式使用路径量词 A 时与LTL 公式表达内容可以相同。LTL和CTL各有优势,Emerson 等人提出扩展的时间逻辑 CTL[7],提供了一种统一的框架,包含了LTL和CTL,但是可满足性判定代价较高。



(a) M 状态结构图　　(b) 展开后的计算树

图 2　M 状态结构图到计算树的转换

2.3　模型检测工具

模型检测因其自动化、高效等特点得到广泛应用,各类模型检测工具也层出不穷。以下是几类典型的模型检测工具。

SPIN[8]是 1980 年美国贝尔实验室开发的模型检测工具,主要关心系统进程间的交互问题。它以 promela 为建模语言,以 LTL 为系统属性的逻辑描述语言,支持 on-the-fly 技术,可以根据用户的需要生成系统的部分状态,而无需构建完整的状态迁移图。SPIN 验证器无法验证实时系统。图3是 SPIN 工具验证的工作过程[9]。

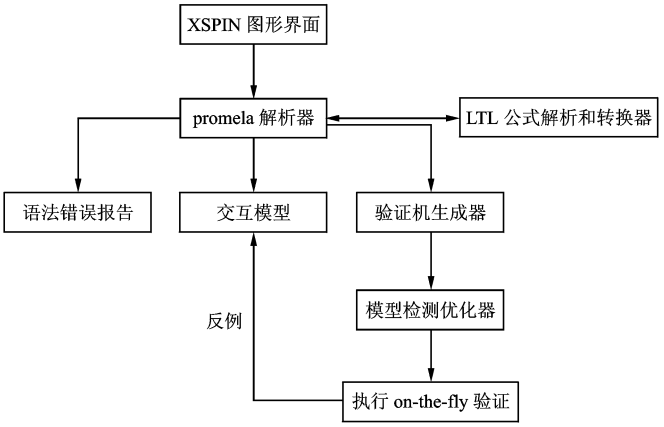


图 3　SPIN 验证原理图

NuSMV 是 1987 年由 McMillan 提出的开源的符号模型检测工具。它可以工作于批处理模式,也可以工作于交互模式。NuSMV 采用扩展的 SMV 语言描述系统,并用CTL和LTL描述需求。NuSMV 结合了以可满足性(SAT)为基础的模型检测和以二叉决策树BDD(Binary Decision Diagram)为基础的模型检测。它具有健壮性,以模块形式构建,不同模块间无依赖关系,代码易修改。提供了同步模型和异步模型的分区方法,可以结合可达性分析,验证不变性质。NuSMV 非常灵活,使用者可以控制并且可以改变其系统模块的执行顺序,并且可以检查和修改系统的内部参数来调整验证过程。

普通的有限状态转换图无法模拟动态变化的物理环境,于是出现了由时间自动机与有着一系列变量的状态转换图结合而成的新模型。UPPAAL 就是基于这种模型的成熟的实时系统验证工具。UPPAAL 是 1995 年由 Aallorg 大学和 Uppsala 大学共同提出,具有可视化图形编辑器。它基于时间自动机并对其进行扩展,引入了坚定位置、初始化程序、紧迫位置和紧迫管道等概念,有助于对真实系统进行建模,并能够有效减少系统内存占用[10]。它被用于检测不变量和可达性属性,尤其是检测时间自动机的控制节点某些组合以及变量约束是否满足初始配置。

CPN-Tool 是由 Aarhut 大学开发的工具,用于有色 petri 网 CPN(Colored Petri Net)的构造和分析。有色 petri 网是用于对系统进行建模并验证其并发、通信和同步的语言,是一种描述离散事件的图形化建模语言。它基于 meta-language,并有强大的可扩展性。它能够通过仿真分析系统的行为,通过模型检测验证系统属性。对于状态爆炸问题,CPN-Tool 有很多方法来减少状态数量,Christensen 等人使用全局时钟和时间戳作为标记,通过状态等价关系化简状态空间,将无限状态转化为有限状态。表1为几种模型检测工具的比较。

2.4　状态爆炸问题

模型检测使用状态空间检索来进行系统验证。状态空间检索的主要缺点就是状态空间随着进程数量增

多会呈现指数增长，从而导致状态爆炸问题。这个问题在实时系统中尤为突出，因为无限的时间会造成状态空间无限增大。目前有存储压缩、状态压缩、组合方法、启发式算法和限界模型检测 BMC(Bounded Model Checking)等方法来解决系统无限状态问题。

Anshuman 等人提出一种以时间换空间的存储压缩方法，即使用不同的方法(如哈希表)来存储状态[12]，节省存储空间。Kang，Inhye 等人提出了一种状态简化算法来解决实时系统的无限状态问题[13]。他们使用有限组节点和转换来表示控制流，使用有限组的真值时钟表示时间限制，有效表达离散密集时间模型的可达性图。同时引入等价类的概念，通过一些等价关系生成状态空间图，即有相同不计时的历史状态合并为等价类，有相同未来状态的合并为等价类。这个方法是基于路径的简化。由于等价关系相同的组是有限的，这样部分解决了实时系统状态爆炸问题。相似的，Boucheneb 等人提出通过制定转换路径，避免交错语义来减少状态数量，达到状态压缩的目的[14]。Ostroff 等人采用组合方法部分解决状态爆炸问题，即将系统先分成小的简单模块，验证小模块的正确性，从而推断出整体的正确性[15]。Huang 等人用有色 Petri 网来表示系统[16]，对有色 Petri 网做一些改进，设置 Petri 网只有单一的入口和出口，将一个大的系统分成多个小系统进行表示，对小系统进行分别验证，部分解决状态爆炸问题。NuSMV 模型检测工具利用启发式方法(heuristics)部分减少状态爆炸问题。启发式方法采用最好优先搜索[17]，搜索顺序是寻找符号检测上下文中最可能出现错误的第一个继承状态优先搜索，这样有针对性的搜索，有效避免了状态爆炸问题。Biere 等人提出限界模型检测 BMC[18]，能够有效抑制爆炸问题。BMC 依赖于可满足性(SAT)工具进行求解。限界模型检测的基本思想是限制考虑系统路径的有限前缀 k，并对其进行检查，若有限前缀中不存在反例，则 $k+1$。在检测过程中，将界限模型检测问题转化为 SAT 问题进行求解。这种方法的效率在于如果系统是错的，仅仅一个小的状态空间碎片足以找到错误，部分解决了状态空间爆炸问题。Bryant 等人[19]提出使用二叉决策图 BDD(Binary Decision Diagram)来存储表示状态转换的布尔函数，极大节省了存储布尔函数所用的空间。二叉决策图中非终端节点用布尔变量 A，B，C，…标识，终端节点用 0，1 标识，从初始节点到终端节点，即可确定路径上对变量的赋值是否使得布尔表达式为真。

表 1　模型检测工具比较

名称	适用系统	状态爆炸问题	建模语言	属性描述语言
SPIN	有限状态系统	可通过有选择的模块化检测避免状态爆炸问题	promela	LTL
UPPAAL	实时系统	无状态爆炸问题	时间自动机 C 语言	TCTL 子集
NuSMV	描述同步异步的有限状态系统	可用启发式方法部分减少状态爆炸问题[11]	扩展的 SMV 语言	CTL 和 LTL
CPN-Tool	有色 Petri 网模型	可用状态等价关系简化状态空间	meta-language	CTL/ASK-CTL

3　模型检测的新进展

尽管模型检测验证能力在不断增强，可是对于复杂系统的验证仍然面临许多挑战。验证混成系统时，由于其状态空间庞大，对于一些基础问题的验证，具有不可判定性。验证系统的访问控制策略时，一条策略可能包含大量规则，如何对策略建模成为难点。验证多智能系统 MAS(Multi-Agent System)时，由于 MAS 出现的目的就是用多个模块来解决单一模块无法解决的复杂问题，因此 MAS 的使用环境一般较为复杂，行为多样且具有随机性，验证难度较大。本文针对上述难题，提出一些解决方法如下。

验证混合自动机。Krishna 等人用混合自动机模型化物联网系统，并且用 LTL 模型检测进行验证。在使用 LTL 模型检测混合自动机时，由于 LTL 具有不可判定性，引入互模拟的概念，并表明一个有限互模拟的存在意味着使得 LTL 模型检测问题的可判定[20]。

验证访问控制安全策略。Maarabani 等人将组织间模型 O2O(Organization to Organization model)与 LTL 模型相结合[21]，来验证互操作访问控制安全策略。将每一个 O2O 策略分别用两个 LTL 公式表示，进行验证。Hwang，Tao 等人定义了一个新的工具 ACPT(Access Control Policy Testing)[22]，将策略制定者的安全需求，转化成可执行的策略，并根据需要对访问控制策略进行动态和静态验证。

验证 MAS。Meski 等人[23]用基于 SAT 的限界模型检测和基于 BDD 的限界模型检测分别验证多智能系统 MAS，并对两种验证方法的时间和内存耗费方面等进行比较。由于目前对 MAS 的验证技术不支持主流验证工具，且输入形式单一，Hunter 等人提出扩展验证框架[24]可以支持多种输入，并且提供翻译器将输入翻译为多个主流验证工具的输入语言，利用现有的验证工具对 MAS 进行验证。由于 MAS 的使用环境相对复杂，其行为具有随机性，Song[25]针对 MAS 行为具有随意性的难题，提出一种概率建模语言对 MAS 的进行描述，并提出相应的模型检测框架。

4 结束语

形式化验证方法已经被广泛的研究，各种描述语言与验证工具层出不穷。相对于演绎验证，模型检测因其全自动并可以提供有数学基础的反例等特点，适用范围更广，可用于验证软件、硬件和协议系统等多个领域。利用模型检测时需控制好状态数量，进行存储压缩或者使用必要的路径压缩、状态缩减算法非常关键。同时前期对于保护目标的选取也非常关键，选出关键资产来保护可以大大提高后期的描述与验证效率。本文工作可为模型检测方法的研究提供借鉴和参考意义。

参考文献：

[1] CLARKE E M. Design and synthesis of synchronization skeletons using branching time temporal logic[J]. Springer Lecture Notes in Computer Science，1981，131(2)：52-71.

[2] ALUR R，COURCOUBETIS D，DILL D. Model-checking in dense real-time[J]. Information and Computation，1993，104(1)：2-34.

[3] BAIER C. Principles of model checking[M]. Cambridge：MIT Press，2008：10-20.

[4] FENG Y，YU N，YING M. Model checking quantum markov chains[J]. Computer System Sciences，2013，79(7)：1181-1198.

[5] 程 亮. 基于模型检测的安全操作系统验证方法研究[D]. 合肥：中国科学技术大学，2009.

[6] CLARKE E M，GRUMBERG O，PELED D A. Model checking[J]. Foundations of Software Technology and Theoretical Computer Science，1999，52(11)：314-324.

[7] EMERSON E A，HALPERN J Y. “Sometimes” and “not never” revisited：on branching versus linear time temporal logic[J]. Symposium on Principles of Programming Languages，1986，33(1)：151-178.

[8] BANG K，CHOI J，YOO C. Comments on the model checker SPIN[J]. IEEE Transactions on Software Engineering，2001，27(6)：573-576.

[9] HOLZMANN G J. The model checker[J]. IEEE Transactions on Software Engineering，1997，23(5)：279-295.

[10] ZHOU Q L，JI L X，WANG Y M. Model checking of real-time systems based on UPPAAL[J]. Computer Applications，2004，24(9)：129-134.

[11] CIMATTI A，CLARKE E M，GIUNCHIGLIA F. NUSMV：a new symbolic model checker[J]. International Journal on Software Tools for Technology Transfer，2000，2(4)：410-425.

[12] MUKHERJEE A，BERTOK P. Memory efficient state-space analysis in software model-checking[C]// 33rd Australasian Computer Science Conference. Brisbane，Australia：ACM Press，2010：23-32.

[13] KANG I，LEE I，MEMBER S. An efficient state space generation for the analysis of real-time systems[J]. IEEE Transactions on Software Engineering，2000，26(5)：453-477.

[14] BOUCHENEB H，BARKAOUI K. Reducing interleaving semantics redundancy in reachability analysis of time Petri nets[J]. ACM Transactions on Embedded Computing Systems，2013，12(1)：1-9.

[15] OSTROFF J S. Composition and refinement of discrete real-time systems[J]. ACM Transactions on Software Engineering and Methodology，1999，8(1)：1-48.

[16] HUANG H. Formal specification and verification of modular security policy based on colored Petri nets[J]. IEEE Computer Society，2011，8(6)：852-865.

（下转第 76 页）

参考文献:

[1] 熊永坤,王瑞革,赵丹辉. 关于雷达信号指纹特征识别的研究分析[J]. 火控雷达技术,2012,41(2):39-41.

[2] 田 慧. 指纹特征匹配算法的研究与实现[J]. 计算机工程与设计,2008,29(12):3258-3260.

[3] 陈 宏,田 捷. 检验配准模式的指纹匹配算法[J]. 软件学报,2005,16(6):1046-1053.

[4] 林 葵. 2015 年全国研究生数学建模竞赛题目(加密文件)[EB/OL]. (2015-09-18)[2015-12-20]. http://www.shumo.com.

[5] 柯志龙. 基于幅频特性的无线信道指纹研究[D]. 厦门:厦门大学,2011.

[6] 林文彬. 基于频率响应的无线信道指纹研究[D]. 厦门:厦门大学,2012.

[7] XIAO L,REZNIK A,TRAPPE W,et al. PHT-authentication protocol for spoofing detection in wireless network[C]// IEEE Global Communications Conference. Miami,USA:IEEE Press,2010:1-6.

[8] SUNG I P,HEUNG M K,WANGROK O. Receprion power estimation using transmitter identification signal for single frequency network[J]. IEEE Transactions on Broadcasting,2009,55(3):652-655.

[9] HOMAYOUN H. Impulse response modeling of indoor radio propagation channels[J]. IEEE Journal on Selected Areas in Communications,1993,11(7):967-978.

(上接第 42 页)

[17] YOUSEFIAN R,RAFE V. A heuristic solution for model checking graph transformation systems[J]. Applied Soft Computing,2014,24(1):169-180.

[18] BIERE A,CIMATTI A,CLARKE E M. Symbolic model checking without BDDs[C]// The 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems. Amsterdam,Netherlands:Springer Press,1999:193-207.

[19] BRYANT,RANDAL E. Symbolic boolean manipulation with ordered binary-decision diagrams[J]. ACM Computing Surveys,1992,24(3),293-318.

[20] KRISHNA S N. Hybrid automata for formal modeling and verification of cyber-physical systems[J]. Journal of Indian Institute of Science,2013,93(3):1-18.

[21] MAARABANI M. Verification of interoperability security policies by model checking[C]// IEEE 13th International Symposium on High-Assurance Systems Engineering. Paris,France:IEEE Press,2011:376-381.

[22] HWANG J H. ACPT:a tool for modeling and verifying access control policies[C]// IEEE International Symposium Policies Distribution System Networks. Raleigh,USA:IEEE Press,2010:40-43.

[23] ME A. Bounded model checking for knowledge and linear time[C]// International Foundation for Autonomous Agents and Multiagent Systems. Valencia,Spain:IEEE Press,2012:1447-1448.

[24] HUNTER J. A synergistic and extensible framework for multi-agent system verification[C]// Proceedings of the 13th International Conference on Autonomous Agents and Multi-agent Systems. Minnesota,USA:ACM Press,2013:869-876.

[25] SONG S. An extensive model checking framework for multi-agent systems[C]// Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems. Paris,France:ACM Press,2014:1645-1646.