文章编号:1009-0568(2013)04-0119-06

模型检测技术研究综述

化希耀[1] 苏博妮[2] 陈立平[1] 高贤强[1*]

(1 塔里木大学信息工程学院,新疆 阿拉尔 843300)
(2 上海海事大学信息工程学院,上海 201306)

摘要 从模型检测技术的研究背景入手,首先阐述了模型检测技术的基本原理和过程。然后介绍了制约模型检测技术发展的状态爆炸问题和一些状态约简技术,包括符号模型检测、on-the-fly技术、偏序归约和抽象技术,并对SPIN、NuSMV、UPPAAL和PAT等模型检测工具进行了介绍和比较。最后总结了模型检测技术在新的应用领域、工具研制、算法研究和与其它技术相结合等几个方面的研究进展。可为今后进一步对并发和实时系统进行建模、仿真和验证提供借鉴和参考。

关键词 模型检测;形式化验证;状态爆炸;状态约简;研究进展
中图分类号:TP311.1 **文献标识码**:A DOI:10.3969/j.issn.1009-0568.2013.04.022

A Survey of Model Checking

Hua Xiyao[1] Su Boni[2] Chen Liping[1] Gao Xianqiang[1*]
(1 College of Information Engineering, Tarim University, Alar, Xinjiang 843300)
(2 College of Information Engineering, Shanghai Maritime University, Shanghai 201306)

Abstract Basic principles and processes of model checking are given, and the notorious state explosion problem and some state-of-the-art reduction techniques are introduced, including symbolic model checking, partial order reduction, on-the-fly and abstraction techniques. Further more, a comparison on some distinguished model checking tools, including SPIN, NuSMV, UPPAAL and PAT is given. Finally, a summarization about the progress of model checking is presented and provides beneficial references to further studying on model checking.

Key words model checking; formal verification; state explosion; state reduction; research progress

随着计算机软硬件系统集成度与复杂度的提高,系统的并发和高度交互等因素。给系统的行为带来了不确定性,使得传统的测试和仿真手段已无法确保系统的正确性和可靠性。形式化方法作为一种以严格数学理论为基础的形式化描述方法,不仅可精确地描述系统的需求,而且可以对系统进行形式化验证。形式化方法的这些特性使得它在软硬件系统开发的各个阶段得到了广泛地应用。近年来该方法受到学术界和工业界的重视。

模型检测是二十世纪最为成功的形式化验证技术之一,它是由CMU的Clarke和Emerson、法国的Quielle和Sifakis分别提出的[1,2]。1981年,Clarke

投稿日期:2013-05-13
基金项目:国家自然科学基金(61162018);塔里木大学校长基金项目(TDZKSS201320)。
作者简介:化希耀(1982-),男,硕士,讲师,主要从事计算机应用技术方面的教学与研究。 E-mail: hua-28@163.com
*为通讯作者 E-mail: mrgaotai@126.com

和他的博士生 Emerson 首次提出一个基于 CTL 的模型检测算法,并开发了模型检测工具 SMV。模型检测是通过对有限状态系统的穷举搜索来实现验证的技术,其面临的主要困难是状态空间爆炸问题。1987 年,CMU 的博士生 McMillan[3]采用有序二叉决策图(OBDDs)[4]隐式地表示系统的迁移关系。他们将这种方法命名为符号模型检测,并在 SMV 的基础上开发了一个新的模型检测工具 NuSMV。符号模型检测可以检测的系统状态数达 10^{120} 以上,这极大地推动了模型检测技术的应用。在随后的若干年里,研究者们纷纷提出各种状态约简算法,并开发了许多经典的模型检测工具。

1 模型检测基本原理和过程

1.1 模型检测基本原理

模型检测的基本原理[5]是用状态迁移系统(S)表示待检测系统的逻辑行为,用时态逻辑公式(φ)描述系统所期望的性质。这样该问题就转化为 S 满足公式 φ。从直观上说,S 为待检测系统的抽象模型,常用的描述方法如有穷状态自动机和进程代数等,而属性公式 φ 则常用时态逻辑描述,如线性时态逻辑 LTL。模型检测是对系统模型的状态空间进行完全搜索的:检测系统的每一个状态是否满足期望的性质。

1.2 模型检测的过程

1.2.1 系统建模(Modeling)

对系统建模首先选用一种形式化描述方法,将待验证的系统设计转化为工具所能接受的模型,比如状态迁移图。通常建模中会采用抽象的方法去除不重要或不相关的细节,以避免引入过多的细节而引起状态爆炸。

1.2.2 性质描述(Specification)

系统所要验证的性质通常是采用逻辑公式来描述,比如时态逻辑。它能够描述系统随着时间变化而引起的行为变化。模型检测提供了许多验证模型是否满足性质的方法,但这并不能保证这些性质包含所有系统所要满足的性质。因此这就要求设计人员在性质描述时要保证性质的完整性。

1.2.3 系统验证(Verification)

系统验证是通过模型检测算法对系统的状态空间进行穷尽搜索。验证结束后,如果未发现违反性质描述的状态,则表明该模型满足期望的性质;否则给出一个反例路径,供设计人员参考。错误发生的原因可能是由于不正确的系统建模或是错误的性质描述,设计人员可根据反例路径分析其原因,修改后重新验证。模型检测的过程如图 1 所示。

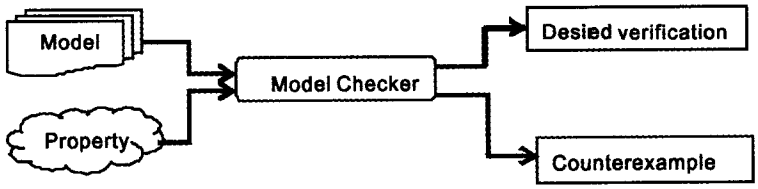

图 1 模型检测的过程

2 状态爆炸问题

在起初的模型检测算法实现时,系统的状态和状态间的迁移都是采用显式的状态迁移图来表示。这种方法对那些进程数量较少的并发系统非常实用。而当并发进程分量较多时,系统的全局状态空间会随着并发量的增加,呈指数增长,即产生状态爆炸问题[5]。状态爆炸问题是阻碍模型检测技术应用的瓶颈。目前,研究人员已经提出许多缓解状态爆炸问题的有效方法,包括符号模型检测、On - the - fly 技术、偏序归约和抽象技术等。

2.1 符号模型检测

符号模型检测[3]是采用布尔公式来隐式表示系统的状态和迁移的方法。该方法基于 Bryant[4]提出

的有序二叉决策图 OBDD。由于 OBDD 提供了比合取范式和析取范式更为压缩的布尔公式的范式,以及许多有效的操作算法,所以符号模型检测可用来验证状态数更多的系统。目前该方法可验证的状态数超过了 10^{120}。

2.2 On - the - fly 技术

On - the - fly[5]技术也称为局部模型检测,是指验证之前不必展开系统所蕴含的所有状态,而是按待验证模型性质的需要,只产生部分和必要的系统状态。该方法目前被绝大部分模型检测工具所采用。

2.3 偏序归约

在一个系统的并发执行的事件中,如果两个事件以任意的次序执行得到相同的全局状态,则称这两个事件是独立的。通常一个系统是由多个进程并发执行,而大部分的用来描述并发系统性质的逻辑无法区分交替序列中按不同次序执行的两个独立事件,所以通常要考虑所有可能的交替序列,从而导致状态爆炸问题。偏序归约[5]的基本思想就是减少验证本质上相同的事件交替序列,只考虑其中一种执行路径。

2.4 抽象技术

抽象[6]是指去除系统中不必要的细节而产生较小的系统模型。常见的抽象技术包括状态合并、数据抽象和谓词抽象。状态合并是通过去除不影响系统规范中变量状态,得到一个简化的自动机模型。数据抽象是一种通用性很强的抽象技术,其基本思想是通过去掉系统的部分信息来构造状态数较小的系统模型。当系统变量的定义域为无穷域时,将导致系统拥有无穷状态,而通常所需验证的系统性质与系统变量的具体值无关,因此可以在系统的精确数据值和一个小的抽象数据值之间建立一个映射关系,通过这种映射关系可以构造一个比实际系统较小的抽象系统。谓词抽象实现了自动将无穷状态系统映射到有穷状态系统的方法,该技术被广泛应用在对软件系统抽象中。

3 模型检测工具

模型检测的优点是可以完全自动地进行验证,这主要归功于许多成熟的模型检测工具的支持。开发新的模型检测工具也是模型检测技术研究的主要内容。目前国际上有许多成熟的模型检测工具,包括:

3.1 SPIN

SPIN[7](Simple Promela Interpreter)是一款运行在 Linux/Unix 环境下的开源软件验证工具,由美国贝尔实验室于 1980 年开发,可用来对多线程软件应用进行验证。SPIN 使用 PROMELA 语言和 LTL 对系统和其性质进行描述,并集成了 on - the - fly、偏序归约和多核/并行等多种模型检测技术。目前该工具被广泛地应用在操作系统、数据通信协议、并发算法、铁路信号协议、航天器控制软件和核电站等领域逻辑设计验证中。SPIN 于 2002 年 4 月荣获 ACM 软件系统奖。

3.2 NuSMV

NuSMV[8]是由 CMU 和 FBK - IRST 联合研制的对有限状态系统进行形式化验证的工具。NuSMV 是在 SMV 的基础上开发的,并对其进行了扩充和升级。其同时支持批处理、命令行式和图形界面三种交互方式。NuSMV 采用 SMV 形式语言描述系统,通过符号模型检测和有界模型检测等技术来分析以 CTL 和 LTL 表达系统的性质。1992 年 CMU 的 Clarke 和他的学生使用 SMV 对 IEEE Futurebus + 缓存一致性协议进行验证,发现了一些协议设计中潜在的错误。

3.3 UPPAAL

UPPAAL[9]是由瑞士的 Uppsala 大学和丹麦的 Aalborg 大学联合开发的模型检测工具。它主要采用时间自动机网络对实时系统进行建模、确认和验证。UPPAAL 适用于对具有有限控制结构和实数值时钟的不确定性进程集、通过信道和共享变量通信的系统进行验证。其典型的应用领域包括实时控制器和通信协议尤其是那些对时间方面要求较高的协议。

3.4 PAT

PAT[10](Process Analysis Toolkit)是由新加坡国立大学 PAT 小组开发的对并发和实时系统进行建

模、仿真和验证的模型检测工具。PAT 采用 CSP#语言对系统建模、LTL 描述系统性质,集成了偏序归约、对称归约和并行模型检测等多种优化技术。另外 PAT 以其友好的用户界面和开放的框架受到诸多用户的青睐,目前该工具已被来自 62 个国家和地区的 2500 + 个用户使用。

传统的模型检测工具都只面向一个特定的领域,如 UPPAAL 支持实时系统的验证、SPIN 和 NuSMV 都只支持常规并发系统的模型检测,而 PAT 集成了多种建模语言,面向的领域包括并发、实时和概率等系统的模型检测。据 PAT 小组的最新论文表明,用户使用 PAT 可以在 1 至 2 个月内开发出面向具体应用领域的模型检测工具。表 1[11]是对上述工具的对比。

表 1 模型检测工具比较

名称	模型检测					可用性
	适用领域			建模语言	属性描述语言	适用平台
	常规	概率	实时			
SPIN	√			Promela	LTL	Windows, Unix
NuSMV	√			SMV	CTL, CTL, PSL	Windows, Unix, MacOS
UPPAAL			√	Timed automata, C subset	TCTL subset	MacOS, Windows, Linux
PAT	√	√	√	CSP#, Timed CSP, Probabilistic CSP	LTL, Assertions	Windows, other OS with Mono library

4 模型检测技术的重要研究领域

模型检测技术自从诞生之日起,已经历了三十余年的发展,在学术界和工业界都引起了广泛的关注。在工业界包括 Microsoft、Intel、Google 和 IBM 等巨头公司先后斥巨资研究该技术并成功地将其应用到实际的产品开发中,取得了巨大的经济效益。在学术界许多世界一流的高校和研究所将模型检测作为理论计算机科学研究的重点方向之一,如美国卡内基梅隆大学和贝尔实验室、英国牛津大学和伯明翰大学、国内的中科院软件研究所、南京大学、同济大学和吉林大学等。

目前,国内外对模型检测技术的研究热点主要集中在以下几个方面。

4.1 将模型检测技术应用到新的领域:模型检测技术在计算机硬件、安全认证协议等方面的验证已经非常成熟,近年来随着计算机运算速度的大幅提升和大容量存储器的出现,以及各种新的缓解状态爆炸问题的算法的提出,为将模型检测技术应用到新的领域奠定了基础。①软件模型检测[12]是指对软件的需求分析或源代码进行形式化验证的技术。由于软件中存在复杂的数据结构、取值空间无限的数据类型等原因,使得状态爆炸问题显得尤为突出。近年来随着抽象技术的应用,大大缓解了上述问题,并出现了许多软件模型检测工具,如 SLAM、JPF 和 BLAST 等。②实时系统与混成系统[13,14]是工业控制和军事领域应用比较广泛的系统。前者是指在限定的时间内系统对外界的输入产生响应的系统,后者是指既包括连续动态子系统,也包括离散动态子系统的系统。如何保证这两种系统的可靠性是计算机科学和控制理论研究的重要课题。通常采用时间自动机、时间 CSP、混成自动机和混成 Petri 网等形式化方法描述系统模型,著名的工具如 UPPAAL 等。④智能规划是人工智能领域研究的热点问题。规划问题是指在问题域中给定一个初始状态、一个目标状态和一些行为,找到一个从初始状态到目标状态的行为序列。将模型检测搜索算法应用到规划问题

求解中,是当前模型检测技术的应用新方向,并且已开发出一些基于模型检测技术的规划器,如MIPS[15]系统。⑤软件体系结构在系统设计描述中扮演着重要的角色,但缺乏形式化描述和验证方法的支持阻碍了软件体系结构建模的发展。目前已提出了几种体系结构描述语言,如 Wright[16] 和 Darwin[17]等。

4.2 模型检测算法的研究:为解决状态爆炸问题,研究者们先后提出了许多有效的方法。除了本文第三节阐述的四种方法以外,还有对称技术、组合推理和有界模型检测等。①对称技术[13,14]的基本思想是:多并发进程系统执行时可能会产生许多相同或相似路径,可以只搜索对称关系中等价的一种情形,以避免重复搜索对称或相同的系统状态。对称技术通过划分等价类来达到约简状态空间的目的,它只考虑等价类中的一种情形,以此类推同类的其它情形。②组合推理[13,14]采用分而治之的办法,将待验证的系统分解为小的模块,分别对这些小模块进行性质验证,再由这些小模块的性质组合推断整个系统的性质。③有界模型检测[5]是基于 SAT 技术的符号模型检测技术,其基本思想是根据状态转换关系将系统状态转换展开 K 次,得到所有长度为 K 的状态转换路径,然后在其中搜索一个反例。如果没有找到反例,则验证过程将 K 不断加 1,直到找到一个反例或到达预先设好的上界,则搜索终止。

4.3 模型检测工具的研制:模型检测技术的应用要靠工具的支持,同时各类工具的应用也使得模型检测技术得到了极大的推广。除了在上文第四节介绍的模型检测工具 SPIN、NuSMV、UPPAAL 和 PAT 之外,还有 SLAM、JPF(Java Path Finder)和 BLAST 等。①SLAM[18]是 Microsoft 公司开发的一款软件模型检测工具。该工具可以对 C 程序进行静态地分析以确定它是否违反给定 API 的使用规则。SLAM 已被成功地应用到 Windows XP 驱动程序的有效性验证中,并发现了一些调用内核 API 的错误。②JPF[19]是由 NASA 开发的 JAVA 程序验证工具。它集成了模型检测、程序分析和测试功能,采用的状态约简技术有偏序归约、切片技术、抽象和运行时分析技术等。NASA 使用 JPF 验证一个实时航空电子设备操作系统并发现了一个潜在的错误。④BLAST[20]是加州大学伯克利分校开发的 C 程序验证工具。该工具在反例引导谓词抽象求精技术上,提出了懒惰谓词抽象技术,大大提高了可验证程序的规模。

4.4 模型检测技术与其它技术相结合:将模型检测技术与其它验证技术结合起来,发挥各自的优势是当前该领域的研究热点之一。①与概率论方法(如马尔科夫链)相结合称为概率模型检测[13,14],该方法不仅可以检测出系统是否可能有错,而且还可以给出发生错误的概率。常见的概率模型有离散时间马尔科夫链、概率自动机等,比较出名的概率模型检测工具如 PRISM。②与定理证明相结合是目前最有前景的一种方法[13,14],其基本思想是将模型检测作为演绎框架内的一个决策过程,也可以先使用演绎推理获取系统的一个有限状态抽象系统,然后对该抽象系统进行模型检测。

5 小结

目前,模型检测技术已被广泛地应用在计算机硬件设计、通信协议、控制系统和安全认证协议等领域,取得了巨大的成功。例如 Microsoft 和 Intel 等许多公司均已采用该技术验证产品的正确性。为表彰 Clarke、Emerson 和 Quielle、Sifakis 等人在模型检测领域所做出的突出贡献,2007 年美国计算机协会(ACM)授予他们 ACM 图灵奖。

保障日益复杂的软硬件系统的可靠性和安全性,是研究者们正在努力研究的难题。建立在严格数学理论基础之上的形式化方法必将在这之中占据主导地位,且已从实验室成功走向工业应用,并产生了许多丰硕的成果。本文从模型检测技术的背景入手,详细阐述了该技术的基本原理和发展现状,并对几款成熟的模型检测工具做了对比分析,总结了近年来模型检测技术在国内外的研究进展,为今后进一步研究相关技术提供参考和借鉴。

模型检测是一个发展非常快的研究方向,在最近的一些研究论文中有人将其应用到了生物系统和智能电网的分析和验证中,取得了新颖的效果。深入研究模型检测技术将为今后可信软件、更高效硬

件系统及其它复杂系统的设计研究提供更佳的手段与方法。

参考文献

[1] Clarke E M, Emerson E A, Sistla A P. Automatic verification of finite - state concurrent systems using temporal logic specifications[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1986, 8(2): 244 - 263.

[2] Clarke Edmund M. The Birth of Model Checking[A]. Symposium 25 Years of Model Checking[C]. Berlin: Springer Heidelberg, 2008: 1 - 26.

[3] Burch J R, Clarke E M, McMillan K L, et al. Symbolic model checking: 1020 states and beyond[J]. Information and computation, 1992, 98(2): 142 - 170.

[4] Bryant R E. Graph - based algorithms for Boolean function manipulation[J]. IEEE Transactions on Computers, 1986, 100(8): 677 - 691.

[5] Clarke E M, Grumberg O, Peled D. Model Checking[M]. Cambridge: The MIT Press, 1999: 1 - 201.

[6] Clarke E M, Grumberg O, Long D E. Model checking and abstraction[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1994, 16(5): 1512 - 1542.

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

[8] Cimatti A, Clarke E, Giunchiglia F, et al. NuSMV: a new symbolic model checker[J]. International Journal on Software Tools for Technology Transfer, 2000, 2(4): 410 - 425.

[9] Bengtsson J, Larsen K, Larsson F, et al. UPPAAL—a tool suite for automatic verification of real - time systems [M]. Springer Berlin Heidelberg, 1996: 1 - 204.

[10] Sun J, Liu Y, Dong J S, et al. PAT: Towards flexible verification under fairness[A]. Computer Aided Verification Springer Berlin Heidelberg, 2009: 709 - 714.

[11] Liu Y, Sun J, Dong J S. Developing modle chedkers using PAT [M] Automated Tehnology for verificatom and Analysis. Sringer Berdin Heidelberg, 2010: 371 - 377.

[12] Visser W, Havelund K, Brat G, et al. Model checking programs[A]. The Fifteenth IEEE International Conference on Automated Software Engineering[C]. IEEE, 2000: 3 - 11.

[13] 林惠民,张文辉. 模型检测:理论、方法与应用[J]. 电子学报, 2002(S1): 1907 - 1921.

[14] 戎玫,张广泉. 模型检测新技术研究[J]. 计算机科学, 2003(5): 102 - 104.

[15] Edelkamp S, Helmert M. MIPS: The model - checking integrated planning system [J]. AI magazine 22. 3 (2001): 67.

[16] Allen R, Garlan D. A formal basis for architectural connection[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 1997, 6(3): 213 - 249.

[17] Magee J, Kramer J. Dynamic structure in software architectures [A]. ACM SIGSOFT Software Engineering Notes[C]. ACM, 1996, 21(6): 3 - 14.

[18] Ball T, Rajamani S K. The SLAM toolkit [A]. Computer aided verification[C]. Springer Berlin Heidelberg, 2001: 260 - 264.

[19] Havelund K. Java PathFinder, a translator from Java to Promela[J]. Lecture notes in computer science, 1999: 152 - 152.

[20] Beyer D, Henzinger T A, Jhala R, et al. The software model checker Blast[J]. International Journal on Software Tools for Technology Transfer, 2007, 9(5 - 6): 505 - 525.