模型检测新技术研究*)

戎 玫[1] 张广泉[2,3]

(暨南大学中旅学院 深圳518053)[1] (苏州大学计算机科学与技术学院 苏州215006)[2]

(中国科学院软件所计算机科学重点实验室 北京100080)[3]

New Approaches for Model Checking

RONG Mei[1] ZHANG Guang-Quan[2,3]

(Tourism College, Jinan University, Shenzhen 518053)[1]

(College of Computer Science and Technology, Suzhou University, Suzhou 215006)[2]

(Key Lab for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)[3]

Abstract Model checking is an algorithmic verification technique that checks automatically whether a given finite state concurrent system satisfies its temporal specification. The main disadvantage of model checking is state space explosion problem. In this paper, several important approaches have been proposed for dealing with the state explosion problem. Such approaches are symbolic, abstraction, partial-order reduction, compositional reasoning, etc. Then, a number of way are proposed for verifying real-time and hybrid systems using model checking. At last, several approaches combining model checking and other verification techniques or mathematical methods are considered.

Keywords Model checking, State explosion, Symbolic model checking, Abstraction, Partial-order reduction, Compositional reasoning, Real-time system, Hybrid system

1 引言

软件是否可信赖已成为一个国家的经济、国防等系统能否正常运转的关键因素之一,尤其在一些诸如核反应堆控制、航空航天以及铁路调度等安全悠关(safety-critical)领域更是如此。这类系统要求绝对安全可靠,不容半点疏漏,否则将导致灾难性后果。如1996年6月4日,欧洲航天局阿丽亚娜(Ariane)501火箭因为其控制软件的规范和设计错误而导致发射37秒后爆炸。类似的报道屡见不鲜,如何确保这些系统的可靠性成为计算机科学与控制论领域共同关注的一个焦点问题[1]。

软件的可靠性主要取决于两方面,一是软件开发的方法与过程,二是软件产品的测试与验证。在目前大多数的工程实践中,软件产品的设计与开发仍缺乏坚实的科学基础和成熟的方法学,软件产品质量主要还是通过测试和模拟等方法来保证的。由于测试用例的覆盖率是有限的,加之系统的运行通常与外部环境有关(如反应式系统),其执行往往具有不确定性,因此测试极为困难,不仅花费很大,而且无法保证发现所有潜在的错误。为了从根本上保证软件系统的可靠安全,包括图灵奖得主 A. Pnueli 在内的许多计算机科学家都认为,采用形式化方法(formal methods)对系统进行形式化验证和分析,是构造可靠安全软件的一个重要途径[4]。

2 形式化方法概述

形式化方法原则上就是采用数学与逻辑的方法描述和验证系统。其描述主要包括两方面:一是系统行为的描述,也称建模(modeling),通过构造系统的模型来描述系统及其行为模式;二是系统性质的描述,也称规范或规约(specification),即表示系统满足的一些性质如安全性、活性等。它们可以用一种或多种(规范)语言来描述。这些语言包括命题逻辑、一阶逻辑、高阶逻辑、时序逻辑、自动机、(并发)状态机、代数、进程代数、π-演算、μ-演算、特殊的程序语言,以及程序语言的子集等。

形式化验证主要包括两类方法,一类是以逻辑推理为基础的演绎验证(deductive verification),另一类是以穷尽搜索为基础的模型检测(model-checking)。

演绎推理用逻辑公式描述系统及其性质,通过一些公理或推理规则来证明系统具有某些性质。其逻辑推理方法主要有自然推演、归结、Hoare 逻辑以及时序演算(如 Manna-Pnueli 命题线性时序逻辑 PLTL 推理系统[2])等,常见演绎推理工具包括机器定理证明器或检验器 ACL2、HOL、TLV、Coq 等[1,3]。演绎验证的优点是既可以验证有穷状态系统,又可以使用归纳的方法来处理无限状态的问题。这类方法不足之处是不能做到完全自动化验证,对于稍微复杂的系统,自动化的推理就难以胜任,人为的推理过程十分繁琐,费时费力,效率较低。因此该方法在实际应用中的影响有限,只适宜较小系统的验证,难以被工业界所接受。

模型检测使用状态空间搜索的办法来全自动地检验一个有穷状态系统是否满足其设计规范。这类方法的优点在于它有全自动化的检测过程且验证速度快、效率高,并且如果一个性质不满足,它能给出这个性质不满足的理由,据此可对系统描述进行改进。该方法自提出以来,发展非常迅速,其理论与技术得到了工业界和学术界的广泛关注[4]。目前,许多世界著名大公司如 AT&T、Fujitsu、Inter、IBM、Microsoft、Lucent、Motorola、Siemens 等纷纷在其产品设计和开发过程中使用模型检测技术,并在许多复杂的实例研究中发挥了重要的作用。

由于软件系统的描述要比硬件系统和协议复杂,一个软

*)本研究得到国家863高科技项目(2001AA113200)、重庆市应用基础研究项目、中国科学院计算机科学重点实验室资助。

件描述所包含的状态空间通常来讲可以是无限的，验证难度很大。因此，尽管模型检测在集成电路和通信协议等实际系统的验证方面取得了成功，但是对于软件来讲还有很多没有解决的问题，还需要不断研究、探索新的模型检测技术。

3 模型检测及相关技术

模型检测是一种基于算法的性质验证方法。即对于一类给定的有穷状态并发程序(系统)和表示系统性质(或规范)的某种时序逻辑公式，能否找到一算法，它能够判定系统类中的任一给定系统是否满足公式类中任意给定的一个时序逻辑公式。如图1所示，模型检测算法的输入包括二部分，分别是待验证系统的模型 M 和系统待检测性质的描述 ψ，如模型 M 满足性质 ψ，则算法输出"true"；否则给出反例说明 M 为何不满足 ψ。系统建模、性质描述和算法验证是模型检测技术的三个主要步骤。最初的模型检测算法由 E. M. Clarke、E. A. Emerson、Queille，Sifakis 等人在20世纪80年代初期提出[5]，他们采用分支时序逻辑 CTL 来描述系统的性质，又称为 CTL 模型检测；稍后又出现了线性时序逻辑 LTL 模型检测。一般情况下，CTL 模型检测算法的时间复杂性为线性或多项式的[6]，而 LTL 模型检测算法的时间复杂性为 PSPACE-完全的[1]。

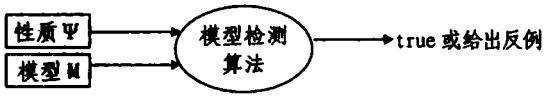

图1 模型检测

由于模型检测基于状态搜索的基本思想，搜索的可穷尽性要求系统模型状态数有穷，故不能直接对无穷状态系统进行验证。因此对于一般软件来说，首先需要有一个从任意状态到有限状态的建模过程。即使对于有穷状态系统，模型检测也会面临"状态空间爆炸(state space explosion)"的严重问题。

CTL 或 LTL 模型检测方法一般采用列表或表格等方式显式表示状态空间，这些状态空间图的大小与系统模型的状态数成正比，而模型的状态数与并发系统的大小成指数关系。因此随着所要检测的系统的规模增大，所要搜索的状态空间呈指数增大，算法验证所需的时间/空间复杂度将超过实际所能承受的程度。如早期的模型检测系统至多能以每秒100个状态的速度检测10^4～10^5个状态的状态转换图，这极大限制了模型检测的系统规模和应用范围。

如何有效缓解"状态爆炸"是模型检测能被广泛使用的一个重要前提，在这方面已有一些重要的方法被相继提出，主要包括符号(模型检测)方法、抽象技术、偏序归约，分解与组合以及对称、归纳、On-the-fly 方法等[5]。

3.1 符号模型检测

符号模型检测是一种采用符号方法表示状态空间的模型检测技术[3,5]。该方法主要基于 BDDs(Binary Decision Diagrams，二元决策图)的状态空间符号表示，BDDs 是 Bryant 在1986年提出的一种比合取范式 CNF 和析取范式 DNF 更便捷的布尔函数的表示形式。布尔函数 $f(x_1, x_2, \cdots, x_n)$ 表示 n 元组(每个值为(0,1))与{0,1}集合之间的对应关系，在数字电路和协议的各种设计和分析中有广泛应用。符号模型检测采用 BDDs 表达状态转换关系和不动点，用不动点算法计算状态的可达性以及这些状态是否满足某些性质。采用这种符号表示使可验证系统的状态数增加若干数量级，例如用最初 Clarke 和 Emerson 提出的 CTL 模型检测算法，基于 BDDs 表示，可以验证状态数超过10^{20}的硬件系统，而采用直接穷举方式表示状态空间只能验证至多10^8个状态。

基于 BDDs 的符号表示已经用于许多模型检测算法和系统中，其中最具影响的是 CMU 的 K. L. McMillan 在其博士论文中提出的一种基于 BDDs 的符号模型检测工具 SMV 系统[7]。它成功地发现了 IEEE Futurebus+ standard(IEEE 标准896.1-1991)中描述的 cache 一致性协议中的错误，这也是使用自动验证工具首次发现 IEEE 标准的错误。目前，符号模型检测技术已取得了突破性进展，可以处理状态数多达10^{120}的系统[5]。但是 BDDs 表示也并非万能，寻求比 BDDs 更简明的符号表示方式是符号模型检测今后的一个研究方向。

3.2 抽象技术

抽象技术是除符号方法外对付"状态爆炸"问题的另一种非常有效的和重要的手段。

传统的模型检测方法主要适用于面向控制的系统，而不太适合与数据路径有关的电路系统或具有复杂数据结构的反应系统。符号方法虽然可以处理一些与数据处理有关的系统，但其验证的复杂性往往较高。对于这类系统的验证，常常需要采用数据抽象技术，即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，通过扩展状态和转换之间的映射，产生一个比实际系统小得多的抽象系统。还有一种重要的抽象技术是状态合并(merging)，为了压缩状态空间，它通过消除一些不影响规范的变量状态，得到简化的自动机模型，通过验证简化模型的性质来降低模型检测的复杂性。

3.3 偏序归约

偏序归约(partial order reduction)是基于并发异步模型交替执行导致的状态爆炸提出的一种重要技术。

并发异步模型是分布式系统、网络通讯/安全协议中的一种常见模型。其行为通常采用交替序列方式，它是通过各个原子转换以不确定顺序交替执行来表示并发行为的，其计算模式是一个线性状态序列 $s_0, s_1, \cdots$，公平转换系统 FTS 就是这样一个典型的并发计算模型[2,10]。如果两个并发事件以任意次序执行都是等效的，则称它们是彼此独立的。一般用来描述并发异步系统性质的(逻辑)规范往往不能区别交替序列中以不同次序执行的两个独立事件。因此，通常要考虑这样事件的所有可能的交替执行，从而可能导致巨大的状态空间。

偏序归约技术基于偏序(满足自反、反对称和传递关系)计算模型。每一偏序计算可以由一个树结构来表示，每一偏序计算对应多个交替序列，如果规范无法区分这样的序列，则可以只分析其中一个交替序列。该方法可减少所考虑的交替序列的数量，因而简化了模型状态空间。偏序归约最初由 Overman 提出，他对并发模型作了限制，不包括循环和不确定选择。目前几种主要的偏序归约技术包括 Valmari 的顽固集、Godefroid 的不变集以及 Peled 的丰富集等[5]。Bell 实验室的 G. Holzmann 等人使用偏序归约等技术研制了模型检测工具 SPIN[8]，NASA/WVU 软件研究实验室利用 SPIN 验证了一个航天容错控制软件，其软件模型约10^5个状态，形式验证发现了3个异常，包括进程优先规则和互斥规则冲突，该问题同导致与 Mars Lander 通讯丢失的 Mars Pathfinder 问题类似[9]。

3.4 组合推理

组合推理是一种基于检测局部状态空间的方法。对于大系统的验证，组合推理方法利用"分而治之"策略，根据系统(如复杂电路和协议)的自然的部件或模块结构，先分别验证系统各个部件(模块)的局部性质，再由各个部件的性质推断整个系统的性质。如果系统满足每一局部性质，并且局部性质

的合取蕴涵了整个规范，那么完整的系统也必定满足这个规范。

在验证部件的性质时，有必要对环境（如其它部件的行为）作出假定。这种方法称为假定-保证（assumption-guarantee）推理，最初是由 A. Pnueli 提出的，他用公式 $\langle\varphi\rangle P\langle\psi\rangle$ 表示：如果包含部件 P 的系统满足 φ，那么也满足 ψ。其中 φ、ψ 都是时序逻辑公式，φ 是 P 对环境的假定，ψ 是 P 所作的保证。例如，如果 $\langle\varphi\rangle P_1\langle\psi\rangle$ 并且 $\langle true\rangle P_2\langle\varphi\rangle$，那么 $\langle true\rangle P_1 \| P_2\langle\psi\rangle$。Josko 基于假定-保证，用一种有限制的线性时序逻辑公式来表示对环境的假定，提出了一系统是否在各种环境下都满足给定 CTL 规范的模型检测算法。组合推理方法需要结合符号化方法来实现其自动推理过程。

除上述四种方法外，用来对付状态爆炸的方法还有对称、归纳等，对称技术主要适用于含有许多对称重复部件的有穷状态并发系统（如一些协议和硬件），它采用一种置换图来定义系统状态空间上的一种等价关系，通过划分等价类来简化状态空间。归纳方法可用于无限多个有穷状态系统的验证问题，一般来说该问题是不可判定的，但在许多情况下，可以提供一个不变量进程来表示任意数量家族的行为，通过不变量来检测家族中所有成员的性质。

4 实时与混成系统的模型检测

随着实时（real-time）与混成（hybrid）系统在工业及国防等领域得到愈来愈广泛的应用，采用形式化分析和验证技术（特别是模型检测技术）保证这些系统的可靠性成为近年来计算机科学与控制论领域的一个热门研究课题。

实时系统是一类在限定时间内对外界产生的刺激或输入作出响应的计算系统。许多关键性系统都具有实时性，如核反应堆控制系统、飞行控制系统、铁路调度系统等。这类系统的正确性依赖于事件发生的实际时间。通常采用时间自动机作为系统模型，用来描述实时系统及其与时间有关的行为模式，其它的一些计算模型包括时间转换系统、时钟转换系统，以及时间 CCS、时间 CSP 和时间 Petri 网等[10]；通常采用扩充了时间表达能力的时序逻辑——各种实时逻辑（如 TCTL、MITL、TPTL、RTTL 等）[11]作为实时系统的规范语言，用来描述系统的实时性质。

在实时系统的验证方面，R. Alur 等人给出了时间自动机的非空性的判定算法，并结合抽象技术提出一种将无穷状态空间转换为有穷状态空间的"域等价"方法，奠定了实时系统模型检测的基础。R. Alur 等人给出了有穷状态实时系统关于 TCTL 的一个模型检测算法，并证明了 TCTL 的可满足性是不可判定的。T. A. Henzinger 等人给出了有穷状态实时系统关于 TCTL 的一个符号模型检测算法，R. Alur、T. Feder、T. A. Henzinger 证明了 MITL 的可满足性是可判定的。

混成系统是一类既包含离散量又包含连续量的计算系统。数控系统、机器人等一些与其外部连续变化的物理环境不断交互的嵌入式系统是这类系统的典型例子，实时系统亦可看作是用一组时钟变量表示连续量的特殊混成系统。通常采用混成自动机、相位（phase）转换系统、混成 CSP 和混成 Petri 网等作为混成系统模型；描述混成系统性质的规范语言大多仍然采用各种扩充时序逻辑，包括一些实时逻辑以及 TLA+、HTL、ICTL、时段演算等。

由于混成自动机的验证问题一般是不可判定的，故目前混成系统的模型检测主要集中于一类特殊混成系统——线性混成系统的验证[3]，R. Alur 等人讨论了线性混成系统的符号模型检测问题，但其算法对一些线性混成系统是不可判定的，为此，T. A. Henzinger 线性混成系统一些重要子类（包括多速率混成系统、矩形混成系统等）的可判定性问题[13]。

目前已有一些针对实时与混成系统的模型检测工具，如 UPPAAL、KRONOS、HyTECH 等[12,13]。

5 模型检测与其它方法的结合

鉴于现有的每一验证方法或工具均存在各自的不足或缺陷，如何将这些风格迥然不同的方法（工具）有机平滑地结合到统一的框架中，充分发挥各自的长处和优势，是形式验证的一个重要研究方向[14]。

（1）模型检测与定理证明方法相结合　这是目前最有希望的一条途径，已出现这方面的一些验证工具，它们将模型检测作为演绎框架内的一个决策过程，如原型验证系统 PVS、Stanford 时序证明器 STeP[15]等；另一种结合方法是先使用演绎推理获取系统的一个有穷状态抽象系统，然后再对该抽象系统进行模型检测。

（2）模型检测与测试方法相结合　目前形式化方法可用于测试用例的自动生成，这可以节约许多时间，并在一定程度上保证测试用例的覆盖率，但测试不能穷举所有可能情况，模型检测只能穷举被测系统的模型而不是系统本身，如何（或能否）将模型检测与测试方法相结合，使模型检测也能象测试那样直接针对实际系统进行，也是提高系统可靠性的一条值得探索的途径[1]。

（3）模型检测与概率论方法相结合　将模型检测与概率论方法（如概率推理、Markov 链等）相结合称为概率模型检测，这样不仅可以知道一个系统是否可能有错，而且知道错误发生的概率，可以忽略发生概率非常小的错误。通过对系统行为的概率推理和分析，还可以对系统的性能（如调度、响应时间等）和可靠性指标进行分析和评价[3]。

参考文献

1 Peled D A. Software Reliability Methods. Springer, 2001
2 Manna Z, Pnueli A. Temporal Verification Reactive Systems: Safety. New York, Springer-Verlag, 1995
3 Inan M K, Kurshan R P. Verification of Digital and Hybrid Systems. Springer, 2000
4 Pnueli A. Verification Engineering: A Future Profession. (A. M. Turing Award Lecture) Sixteenth Annual ACM Symposium on Principles of Distributed Computing, San Diego, Aug. 1997
5 Clarke E M, Grumberg J O, Peled D A. Model Checking. MIT. 1999
6 Clarke E M, Emerson E A, Sistla A P. Automatic Verification of Finite-State Concurrent System Using Temporal Logic Specifition. ACM Trans. on Prog. Lang, 1986, 8(2): 244～263
7 McMillan K L. Symbolic Model Checking: An Approach to the State Explosion Problem. Boston Kluwer Academic, 1993
8 Holzmann G J. The Model Checker SPIN. IEEE Trans. on Soft. Eng, 1997, 23(5): 279～295
9 Schneider F, Eastbrook S M, Callahan J R, Holzmann G J. Validating requirements for fault tolerant systems using model checking. In: Int. Conf. on Requirements Engineering, April 1998
10 张广泉. 基于转换系统的广义反应系统形式模型. 计算机科学, 2000, 27(1): 28～30
11 Alur R, Henzinger T A. Real-time logics: Complexity and expressiveness. In: Proc. 5th IEEE Symp. Logic in Comput. Sci. 1990. 390～401
12 Berard B, Bidoit M, Finkel A, et al. Systems and Software Verification: Model-Checking Techniques and Tools. Springer, 2001
13 Henzinger T A, P H Ho. HyTech: a model checker for hybrid systems. CAV'97, LNCS1254, 1997. 460～463
14 Clarke E M, Wing J M. Formal Methods: State of the Art and Future Directions. ACM Computing Surveys, 1996, 28(4)
15 Manna Z, STeP group. STeP: The Steandford Temporal Prover. STAN-CS-TR-95-1562, 1995