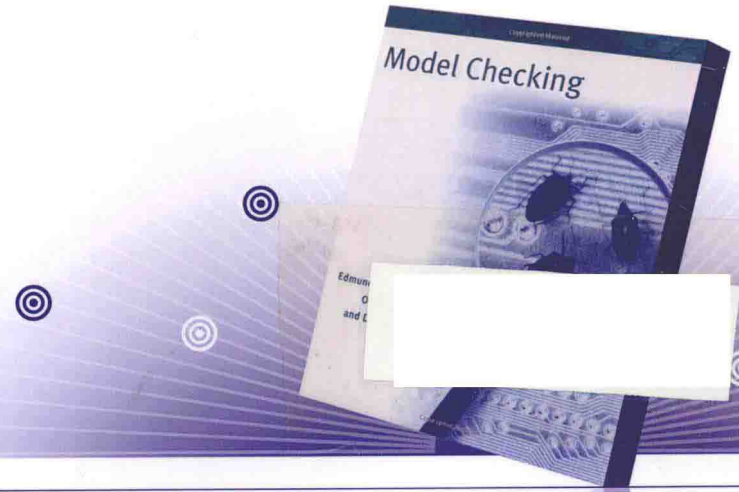
# 模型检测

# Model Checking


▶[美] Edmund M. Clarke Jr.
Orna Grumberg 著
Doron A. Peled

▶李 刚 宋 雨 译



中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

# 模型检测

## Model Checking

Edmund M. Clarke Jr.
[美] Orna Grumberg 著
Doron A. Peled

李 刚 宋 雨 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

## 内容简介

模型检测是一种用于自动验证有限状态并发系统的技术，与基于模拟、测试和演绎推理的传统技术相比，具有许多方面的优势。本书共分18章，涵盖的主要内容包括模型检测的基本知识、模态逻辑、符号化技术、SAT Solver、限界模型检测、自动机上的模型检测、抽象解释、程序分析、实时系统验证，同时介绍NuSMV和UPPAAL两个流行的模型检测器。

本书既适合从事计算机科学、电子科学、电气工程、工业制造等复杂系统研究的高等院校科研人员和研究生阅读，也适合系统管理、测试部门的企事业单位人员作为参考用书。



版权贸易合同登记号　图字：01-2015-0309

**图书在版编目(CIP)数据**

模型检测/(美)克拉克(Clarke Jr., E. M.)，(美)格伦贝格(Grumberg, O.)，(美)佩莱德(Peled, D. A.)著；李刚，宋雨译. —北京：电子工业出版社，2015.8
书名原文：Model Checking
ISBN 978-7-121-27295-0

Ⅰ. ①模…　Ⅱ. ①克…②格…③佩…④李…⑤宋…　Ⅲ. ①自动检测系统　Ⅳ. ①TP274

中国版本图书馆CIP数据核字(2015)第227864号

策划编辑：冯小贝
责任编辑：周宏敏
印　　刷：涿州市京南印刷厂
装　　订：涿州市京南印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编　100036
开　　本：787×980　1/16　印张：14.75　字数：330千字
版　　次：2015年8月第1版
印　　次：2015年8月第1次印刷
定　　价：59.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010)88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：(010)88258888。

# 译 者 序

本书译自美国卡内基梅隆大学 Edmund M. Clarke 教授所著 *Model Checking* 一书，该书是一部关于形式化验证的经典著作，作者 Edmund M. Clarke 是世界著名的计算机科学家，2007 年曾获得图灵奖。

模型检测是一种用于自动验证有限状态并发系统的技术，与基于模拟、测试和演绎推理的传统技术相比，具有许多方面的优势。本书涵盖的内容包括模型检测的基本知识、模态逻辑、符号化技术、SAT Solver、限界模型检测、自动机上的模型检测、抽象解释、程序分析、实时系统验证，同时介绍 NuSMV 和 UPPAAL 两个流行的模型检测器。

当今的工业系统规模越来越大，如集成电路、电子系统、软件系统等，如果设计过程中的缺陷不被发现，将对系统运行带来隐患，甚至是灾难，这一点已经被很多大型的事故所证明。传统的模拟和测试技术已经不能满足对系统设计有效性的保证，亟需一种新的方法来有效地保障复杂系统开发过程中的正确性。而形式化验证技术是一个非常有吸引力的验证方法，因其可以用自动搜索代替手动证明来解决验证的问题，并能在系统不满足性质时提供反例路径，目前工业界普遍承认模型检测在验证系统正确性方面具有巨大的实力和潜力。

2013 年，德国联邦教研部与联邦经济技术部提出，继蒸汽机的应用、规模化生产和电子信息技术等三次工业革命后，人类将迎来以信息物理融合系统(CPS)为基础，以生产高度数字化、网络化、机器自组织为标志的第四次工业革命，即“工业 4.0”。因此，未来的工业领域将呈现出以复杂系统研究为主要对象的新常态，而复杂系统的可靠性、完备性需要严格的形式化验证方法来支持，目前国内这方面的著作还不多见。本书对于从事计算机科学、电子科学、电气工程、工业制造等复杂系统及其形式化验证工作的研究者来说，是一本非常适用的参考书。

全书由李刚翻译第 1 ~ 12 章及前言，宋雨教授翻译第 13 ~ 18 章及附录，最后由李刚统稿和审校。限于时间和水平，错误和不妥之处在所难免，敬请读者批评指正。

译 者

2015 年 7 月于华北电力大学

# 序　言

目前，人们广泛认为“帮助计算机使之更好地服务于人类”的理念出现了一些偏差，这使得我们更易于将更复杂更敏感的系统丢给其他人去作，这种现象并非由现有机器的计算速度或者计算能力造成，而是我们缺乏对设计实现出的复杂系统在所有环境中都会正确运行的自信。

这种设计有效性问题——尽可能早地保证设计的正确性——是所有负责的系统开发遇到的挑战，而且对这个问题的解决方案在开发周期的成本和时间预算中所占的比例不断增加。

目前保证设计有效性的实际方法依旧是持续了多年的、具有丰富经验的模拟和测试两种技术。虽然在早期的调试阶段这两种方法是有效的，但是经过这两种检测手段后，设计仍然包含大量错误。随着系统设计越来越清楚，它们的效率急剧降低，每发现一个小错误都需要花费大量时间。这些方法还会导致一系列的问题：没有人能知道何时才到达了这种技术的查错极限，更没有人能预测出经检测后设计中还有多少错误。随着设计复杂度的急剧增加，比如从大约每片有 50 万个门提高到每片有 500 万个门，一些有远见的管理者预见这些传统方法将要崩溃，并且它们将无力再次提高。

形式化验证技术是一个非常有吸引力验证方法，人们不断呼吁用它来替代模拟与测试方法，它也是这本书的主旨。当模拟与测试检测了系统的一些可能行为与情况，却不能确定系统是否还含有致命错误这样的问题时，形式化验证通过对所有系统行为进行彻底检测来清楚地回答这一问题。因此当系统设计被形式化验证方法证明为正确时，就暗示了所有的行为已被检测，并且再也不用考虑是否达到足够的覆盖率或是否还有遗失行为这样的问题。

这些年已经提出了多种形式化验证技术。本书集中介绍模型检测方法，模型检测通过对给定系统（模型）中所有可达状态与其上的行为进行彻底（明显的或暗含的）检测，来验证反应系统期望的行为特性。

与其他方法比较，模型检测方法有两个显著的优势：

- 它是全自动进行的，并且这种方法不要求使用者具有专业的数学（如定理证明）知识和经验。任何可以对设计进行模拟验证的人完全能够胜任使用模型检测验证相同的设计。同当前实际的验证方法相比，模型检测可以被看作一个更出准的模拟工具。
- 若检测出设计未能满足一个期望的性质，模型检测总是产生一个反例来表明如何违背了这种系统行为。这个缺陷轨迹有助于理解真正的失败原因，同时也提供了修复此问题的重要线索。

这两个重要的优势与符号模型检测（允许对天文数字般的状态进行彻底的暗含列举）的出现引起了形式化验证领域的根本改变，符号模型检测把模型检测技术从一个纯理论的学科转

变为实际可行的技术，这个技术可以融入许多工业开发流程中，它已经成为一种确保设计有效性的极有价值的方法。

工业界普遍承认模型检测具有巨大实力和潜力，大量的研究者和开发者致力于模型检测的开发，他们开发的产品已经应用于大型公司对先进的半导体电路和处理器的验证。

我们非常幸运拥有这本关于模型检测的原理与方法的权威性著作，这本书的作者首先帮助构思出了模型检测思想，并坚持进行模型检测研究，直到它变成一个令人惊奇的成功技术。

我对这本极好的参考书非常自信，这本书将非常有助于读者(包括学生与从业者)理解形式化验证特别是模型检测技术的原理和实现。

Amir Pnueli

# 前　言

有限状态并发系统出现在计算机科学的几个领域中，特别在数字电路与通信协议的设计中大量出现。在这些系统的设计后期发现的逻辑错误对于电路设计者与程序员而言都是一个非常棘手的问题。这些错误或许会推迟新产品的上市时间，而且可能导致一些已经使用的重要设备发生故障。目前最广泛使用的验证技术是基于测试或仿真的，当电路或协议的状态规模巨大时这些技术无疑会遗漏重要的错误。虽然已经对定理证明器、项重写系统和证明检测器进行了长期和大量的研究，但是这些技术不但耗时，还常常需要许多人工干预。在20世纪80年代，一个被称作时序逻辑模型检测的验证技术由美国的Clarke与Emerson[61]和法国的Quielle与Sifakis[219]独立提出。这种方法使用命题时序逻辑来表示性质规范，电路设计与协议建模为状态变迁系统。换句话说，检测变迁系统看它是否是性质规范的模型。

同机械的定理证明或证明检测相比，模型检测在验证电路与通信协议方面有几个重要的优势。一个优势是它的检测过程是全自动的，使用者只需提供被检测的模型与性质规范的高阶描述。模型检测算法要么以结果为真终止，这表示模型满足规范；要么给出一个反例指出为什么性质没有被满足。这个反例对在复杂变迁系统中发现和修正细微错误非常重要。检测过程相当快，通常大约几秒钟产生一个结果。由于可以检测部分性质规范，所以验证时在获得有用信息之前不必构建完整的系统模型。当一些性质规范没有被满足时，可以通过检测另一些公式——不是未满足的那些性质——来定位错误的源头。模型检测使用的描述性质规范的逻辑能直接表示许多并发系统推理中的性质。

模型检测主要的缺点是状态爆炸，如果被检测的系统有许多使变迁可以并发进行的组件，则可能会产生状态爆炸问题。在这种情况下，整个系统状态的数目将按组件数呈指数增长。由于这个问题，许多形式验证的研究者预言模型检测对于大型系统绝对是不实用的。但是，在20世纪80年代后期应用符号模型检测技术后，所能检测的变迁系统的规模显著地增大了。

这种增长归因于二叉判定图(一种表示布尔函数的数据结构)的使用。这种新的数据结构使变迁系统不但获得了简洁的表示，并使快速运算成为可能。符号模型检测方法对同步电路特别有用。在验证异步协议时，可通过使用偏序约简来减少状态空间的大小。偏序约简基于下面的观察：不同顺序的独立执行事件对应的计算是等价的，但却被性质规范严格区分。因此只需检测一个简化的状态空间，这个状态空间只为每个等价类包含至少一个典型的计算方法。

因为这些技术和稍后将在本书中介绍的另一些技术的出现，现在模型检测已经作为一种实用的验证技术在工业界中广泛使用。实际上，几个公司正开始把模型检测工具推向市场。

我们认为这本书既可作为模型检测的介绍，也可作为研究者的参考。我们试图包含尽可能多的内容使其完整，但是这个领域的研究进展如此之快以至于勉强跟上许多令人兴奋的新

的研究成果变得不可能。这本书的若干部分与其他部分相比更加专业，在第一次阅读时可以跳过它们。我们会指出这些部分，它们主要针对从业者与研究者。我们真诚地希望这本书能激励您在模型检测领域做出进一步的研究。

最后，作者要感谢那些帮忙创作这本书的人。首先我们想要对 David Long 表示我们的谢意，他的努力使这本书变为现实。我们也想要感谢那些审阅初稿的人们，他们是：Eric Allen、Ilan Beer、Armin Biere、Sergey Berezin、Ser-gio Campos、Ching-Tsun Chou、Allen Emerson、Kousha Etessami、Nissim Francez、Masahiro Fujita、YairHarel、Wolfgang Heinle、Hiromi Hiraishi、N-eil Immerman、Somesh Jha、Irit Ka-triel、Shmuel Katz、Bob Kurshan、Kim G. Larsen、Yuan Lu、Jan Maluszynski、WillMarrero、Marius Minea、Bud M-ishra、Ulf Nilsson、Wojciech Penczek、Amir Pnueli、ToshiioSekiguchi、Sub-ash Shankar、Zeev Shtadler、Prasad Sistla、Frank Stomp、WolfgangThomas、Moshe Vardi、DongWang、PierreWolper、Bwolen Yang、Husnu Yenigün、Y-unshan Zhu。如果我们意外地遗漏了对这本书有所帮助的人们，我们对此道歉。Edmund Clarke 希望感谢 Michael Shostak 在写书的过程中所给予的鼓励。Doron Peled 想要感谢 Marta Habermann 在 CMU 的 1998 年春天的学期中提供了一个舒适安逸的居所。

# 目　录

# 第1章 绪 论

模型检测是一种用于自动验证有限状态并发系统的技术。它和基于模拟、测试和演绎推理的传统技术相比有许多优势。这种方法已经实际应用于验证复杂的时序电路设计和通信协议，并取得了成功。模型检测中最大的困难来自于状态空间爆炸。在验证包含大量交互组件的系统或者大规模数据系统(例如，电路的数据通路)时，通常会遇到这个问题。在这种情况下全局状态的数量极有可能是巨大的。在过去10年中，人们已经找到了相当多减缓状态空间爆炸的方法。在本章，我们从硬件设计和软件设计两个角度来比较模型检测与其他形式化验证方法。同时我们将描述如何使用模型检测来验证复杂的系统设计。我们也将回顾各种模型检测算法的发展历程，并讨论若干现有的缓解状态爆炸的手段。

## 1.1 形式化方法的需求

现今，硬件和软件广泛应用于故障敏感系统：电子商业、电话交换网、高速公路和航线控制系统、医疗器械等等。我们经常听说由硬件或软件系统中的微小缺陷引起故障造成事故的例子。最近一个事故发生在1996年6月4日，Ariane 5火箭在发射升空不到40秒爆炸。事故调查委员会发现事故的原因是计算火箭动作的计算机中的一个软件错误。在发射过程中，当64位长的浮点数转换成16位有符号整数时，发生了一个例外。但例外处理代码没有覆盖到这个转换过程，导致此计算机死机。同样的失误也导致备份计算机死机。最终，错误的飞行姿态数据传送给火箭的主计算机，从而导致了这场灾难。调查出此故障的小组建议了若干避免类似事故的策略，其中就包含Ariane 5的软件验证。

很明显，对硬件与软件系统可靠性的需求是急迫的。随着我们对此种系统的依赖与日俱增，确保其正确运行的重担也越来越重。遗憾的是为了重获安全性而简单关闭故障的系统的方案不再可行。我们更加依赖于连续运转的系统。其实在一些场合中，关闭设备可能导致更不安全。即使故障没有生命威胁时，替换重要代码或电路以保证系统正确运行的后果在经济上也可能行不通。

由于在汽车、飞机与其他安全攸关系统中因特网与嵌入式系统的成功使用，未来我们极有可能更加依赖计算机设备的正确运行。而从目前看来这种依赖的节奏在加快。对应于技术的快速提高，发展出一种方法论来增长我们在保证系统设计正确性方面的信心变得更加重要。

## 1.2 硬件与软件验证

复杂系统主要的验证方法是模拟、测试、演绎验证和模型检测。模拟和测试[202]都是在系统实际使用之前进行实验验证。不同的是模拟对系统的抽象或模型进行验证，而测试是对实

际产品进行验证。虽然模拟验证的是电路设计，测试验证电路本身，但是这两种方法的大体流程都是在验证时，对系统的某些点注入激励信号并在另外的一些点观察生成信号是否与其一致。对于软件验证，模拟和测试的基本方法是为软件提供一些输入而后观察对应输出。所以对于可能具有大量错误的系统设计而言，这两种方法的费效比可能比较高。但是它们几乎不可能检测所有可能的交互故障和潜在缺陷。

演绎验证指使用公理和证明规则来证明系统正确性。在早期的演绎验证研究中，焦点主要集中在证明重大系统的正确性上。可以设想，这种系统的功能正确性是如此重要，以至于开发人员或验证专家(通常是数学家或是逻辑学家)将不计成本和时间去验证此系统。在初期这些证明过程完全是手工构造的。后来研究者才意识到可以开发软件工具来进行推理证明。使用这些工具也能系统地对从某证明状态开始的不同证明路径进行研究。

计算机科学家广泛认可了演绎验证的重要性。演绎验证也深远地影响了软件发展的各个领域(例如，不变量的概念始于演绎验证的研究)。但演绎验证是一个耗时的过程，即使单个协议或电路的证明过程也可能持续若干天或几个月，而且这种方法的使用者也只局限于一些在逻辑推理方面受过培训并有相当经验的专家。因此演绎验证的使用机会相当少。它主要应用于高敏感的系统(例如安全协议)，因为在这种系统中需要投入足够多的资源来保证其使用时的安全。

但一些数学工作是不能被算法代替的。可计算性[142]理论中描述了算法的局限性。实际上，它指出不可能存在一个算法来判定任意一种计算机程序(如使用 C 或 Pascal 书写的程序)是否终止。这也就限制了可自动验证问题的范围。程序的正确终止一般而言是不能自动验证的，因此大部分证明系统不可能完全自动。

演绎验证的一个优点是它能被用于无限状态系统的推理，此时有一部分推理工作可以自动进行。但即使验证的性质是真的，为找到正确推理路径而需要的内存和时间也极有可能是无限的。

模型检测的范围限定在验证有限状态并发系统上，这种限定的一个好处是验证可以自动进行。模型检测算法通常对系统状态空间进行穷尽搜索来确定性质的真假。如果资源充足，检测过程总以是或非终止。而且这种方法能够用高效的算法实现，从而可以在中度规模的计算机(但不是通常的台式计算机)上运行。

虽然限制于有限状态系统可能是模型检测技术的一个主要缺点，但是它非常适用于若干种重要系统的验证：硬件控制器是有限状态系统，并且许多通信协议也是有限状态系统。在非有限状态系统中，也可以把模型检测与抽象和归纳方法结合起来验证。而且在许多情况下，可以把无界数据结构限制到特殊的有限状态系统上验证，例如可以把包含无界消息队列程序的队列个数限制到 2 或 3 这样小的数来调试。

因为模型检测能自动运行且应用范围很广，所以它比演绎验证更优越。但是使用定理证明来完全验证一个系统的情况也是存在的。一个令人兴奋的新方向[220]研究如何把演绎验证与模型检测结合，因此将来复杂系统的有限状态部分或许能够完全自动验证也不一定。

## 1.3 模型检测的过程

使用模型检测技术来进行系统设计的验证包含以下3个步骤，每一步骤都将在后文详细叙述。

**建模** 第一个步骤是把设计转化为能被模型检测工具(模型检测器)接受的形式。在许多情况下这只是个简单的编译过程，但在一些时候，由于验证时间和计算机内存的限制，设计的模型可能还需要使用抽象技术约简不相关或不重要的细节。

**刻画** 在验证之前，需要声明设计必须满足的性质。性质刻画通常是以某种逻辑的形式表示。对硬件与软件系统验证而言，通常使用时序逻辑，这种逻辑体系能表示系统行为随着时间的变化。

性质刻画过程中最重要的问题是完备性。虽然模型检测提供了检测模型是否满足给定性质的一套方法，但是这套方法并不能保证性质刻画确切地表达了待验证系统所需满足的所有性质。

**验证** 理想上验证应该是完全自动的。但实际上它常常需要人的帮助，其中之一就是分析验证结果。当得到失败结果后，通常可以给用户提供一个错误轨迹，可以把它看作所检测性质的一个反例，从而使设计者能够跟踪错误发生的具体位置。当分析错误轨迹改正系统设计后，再次进行模型检测，重新验证，直到验证通过。

错误轨迹也可能由建模或性质刻画(常常称为假否定)过程的失误导致，因此它也能用于确定和修复这两类错误。另外，由于计算机的内存限制，当验证过程需要大量内存时，验证可能不会在有限时间内正常终止而产生错误轨迹。这种情况下，需要改变模型检测器的若干参数或直接调整模型(比如使用抽象技术)，然后重做验证。

## 1.4 时序逻辑与模型检测

时序逻辑能够在不引入时间细节的情况下描述时间序的事件，所以其对并发系统的刻画非常成功。最初，哲学家把时间作为自然语言参数[145]进行研究，这种研究导致了时序逻辑的最终出现。虽然学术界提出的时序逻辑种类繁多，但是大部分都含有类似 $\mathbf{G}f$ 这样的运算符，它表示如果此公式为真仅当 $f$ 在将来总为真(即，$f$ 全局为真)。比如刻画两个事件 $e_1$ 和 $e_2$ 不能同时发生，可以记为 $\mathbf{G}(\neg e_1 \vee \neg e_2)$。一般可以根据时间假定是线性的还是分支的来对时序逻辑分类。本书中牵涉到的时序逻辑公式的语义将由标记状态变迁图给出，由于历史的原因，这种结构被称为 Kripke 结构[145]。

包括 Burstall[48]、Kröger[158]与 Pnueli[216]的一些研究者提议使用时序逻辑推导计算机程序。而 Pnueli[216]第一个使用时序逻辑推导并发。他的方法是通过描述程序语句的公理集合来证明

我们关心的程序性质。Bochmann[25]、Malachi 与 Owicki[184]将这个方法应用到了对时序电路的推导。因为证明过程是手动构造的，所以这个技术常常很难在实际中使用。

在 20 世纪 80 年代早期，Clarke 与 Emerson[61, 103]提出了时序逻辑模型检测算法，这种算法可以实现了上述方法的自动推导。检验具体模型是否满足公式比证明公式对所有模型合法简单得多，并且这个技术完全可以高效实现。Clarke 与 Emerson 提出的分支时序逻辑 CTL 模型检测算法的复杂度不管对于待验证程序的模型规模，还是时序逻辑刻画的性质公式长度，都是多项式时间的，他们也给出了在不改变算法复杂度的情况下如何考虑和处理公正性[120]问题。在许多以公正性假设为前提的并发程序正确性验证中，处理这一步是非常重要的。例如，在互斥算法中为了避免出现“饥饿”，每个进程要求无限次出现。

几乎在同时，Quielle 与 Sifakis[129]为 CTL 的一个子集给出了模型检测算法，但是他们没有分析算法的复杂度。稍后 Clarke、Emerson 与 Sistla[63]提出了改进的算法，这个算法相对于公式长度与状态变迁图(Kripke)规模的积是线性的。此算法在 EMC 模型检测器中实现，而 EMC 被广泛使用于检验网络协议与时序电路[28~30, 31, 63, 98, 197]。早期的模型检测系统在检测给定时序公式时，能够以每秒钟 100 状态的速度验证拥有 $10^4 \sim 10^5$ 状态的状态变迁图。尽管限制性很大，但模型检测器仍然在若干已经发布的电路设计中成功检测出了隐含的错误。

Sistla 与 Clarke[232, 233]分析了不同时序逻辑系统中的模型检测问题，指出对于线性时序逻辑(LTL)系统而言模型检测问题是 PSPACE 完全的。Pnueli 与 Lichtenstein[173]重新分析了检验线性时序公式的复杂度，发现虽然复杂度似乎相对于公式长度呈指数增长，但对于状态变迁图规模却呈线性增长。基于这个观察，相对于短的线性时序公式而言，模型检测的高复杂度也是可接受的。同一年，Fujita[119]实现了基于 tableau 的 LTL 模型检测系统，并提出此系统用于硬件验证的方法。

CTL* 是一个极富表达力的逻辑系统，它把分支时间与线性时间的算子结合在一起。这种逻辑的模型检测问题首先在 Clarke、Emerson 与 Sistla[62]的论文中提及，他们指出 CTL* 模型检测是 PSPACE 完全的，并确定其与 LTL 的模型检测有相同的复杂度。这个结果指出 CTL* 与 LTL 模型检测在状态图规模与公式长度两方面有相同的算法复杂度。因此考虑 CTL* 模型检测的复杂度后发现，把模型检测问题限制到线性时序逻辑，[106]实际上并不会使算法复杂度得到有效改观。

另一些研究者提出了检验并发系统的其他方法。大多数方法使用自动机描述系统规范和系统实现，通过检测系统实现的行为是否与规范一致来进行验证。因为实现与规范使用相同的模型来表示，某一层的实现也可以直接作为下一层改进的规范。Kurshan[1]在语言包含方面做了大量工作，开发了一个被称为 COSPAN[132, 133, 162]的高效检验器。Vardi 与 Wolper[245]第一个建议把 $\omega$ 自动机(定义在无限字串上的自动机)用于自动验证，并且给出了如何依据 $\omega$ 自动机的语言包含来实现线性时序逻辑模型检测问题。在自动机理论中一致性的概念目前也被考虑，包括观测的等价性[77, 196, 224]与各种各样的细化关系[77, 195, 223]。

## 1.5 符号算法

在早期的模型检测算法中，变迁关系经常被显式地表示成邻接表。对包含少量进程的并发系统而言，其状态的数目通常也相当小，所以邻接表方式很实用。但是在包含大量并发组件的系统中，全局状态变迁图的状态数目则大到难以处理的程度。1987 年秋天，还是卡纳基·梅隆大学研究生的 McMillan[46,191]认识到采用符号方法表示状态变迁图可以验证更大规模的系统。这种新的符号表示方法基于 Bryant 的有序二叉判定图(OBDD)[34]。OBDD 表示布尔方程比合取或析取范式更简洁，而且已经有了现成的高效 OBDD 算法。由于符号表示附和待验证的电路或协议所确立的状态空间的一些规律，所以符号方法可以验证具有大量状态的系统——超过状态图遍历算法可处理状态数若干数量级。而且将 Clarke 和 Emerson 的早期 CTL 模型检测算法[61]与状态变迁图的 OBDD 表示相结合，甚至可以验证状态数超过 $10^{20}$[46,191]的系统。后来其他研究者又不断推出了各种基于 OBDD 的改进技术，使得可以验证的状态规模超过了 $10^{120}$[43,44]。

对时序电路和协议而言，使用符号方法隐式建模是非常自然的。首先将电路和协议中的状态变量看作一个集合，对此集合的一组布尔赋值表示一个状态编码，变迁关系可以表示成定义在两个变量集合上的布尔公式，这两个集合一个是现态变量集，另一个是次态变量集，现在此方程可以方便地转化为二叉判定图。而基于符号方法的模型检测算法对应着计算从变迁关系中得到的谓词变换的不动点。不动点本质上就是表示并发系统各时序性质的状态集合。在这种新的实现中，谓词变换和不动点都用 OBDD 来表示，这样可以避免显式地构造并发系统的状态图。

McMillan 开发的模型检测系统 SMV[191]是其博士论文的一部分。SMV 使用一种可以分层表示有限状态并发系统的描述语言。这种语言书写的程序也包含了时序逻辑描述的系统性质刻画。SMV 模型检测器从上述语言书写的程序中提取 OBDD 表示的变迁系统，然后使用基于 OBDD 的搜索算法来判断系统是否满足性质刻画。如果变迁系统不满足某性质刻画，检测器会产生出执行路径来表明性质刻画不能满足的原因。目前 SMV 系统使用广泛，已经使用它验证了大量的例子，这些例子为 SMV 可以胜任工业产品的检测提供了令人信服的证据。

对 IEEE Futurebus + 标准(IEEE 标准 896.1-1991)中的高速缓存一致性协议的验证使人们深刻地认识到符号模型检测能力的强大。虽然 Futurebus + 高速缓存一致性协议的开发始于 1988 年，但是之前所有对该协议验证的尝试都是基于非形式化的技术。1992 年夏天，卡耐基·梅隆大学的研究人员[66,179]首先用 SMV 语言构造了该协议的精确模型，然后用 SMV 来验证此变迁系统模型是否满足高速缓存一致性的形式化规范。结果他们发现了以前从未发现的错误以及协议设计本身的潜在错误。这是第一次应用自动验证工具检测到 IEEE 标准中错误的成功范例。

符号模型检测方法的强大能力主要体现在验证规模越来越大的电路或协议实例时对 CPU

时间需求的增长率上。将其用于一些已经被各种团队研究过的例子后，观察得知符号检测方法下的 CPU 时间增长率相对于电路组件个数而言只是一个小规模的多项式[18, 43, 44]。

许多学者经过独立研究发现 OBDD 可用于表示大规模的状态变迁系统。Coudert、Berthet 和 Madre[81]提出了深度优先搜索积自动机状态空间时，两个确定性的有限状态自动机的等价算法。在此算法中，他们用 OBDD 表示两个自动机的变迁函数。Pixley[213~215]提出的算法也与此类似。另外，Bose 和 Fisher[26]、Pixley[213]、Coudert、Madre 和 Berthet[82]等人的小组都试验了使用 OBDD 的模型检测算法。

在相关工作方面，Bryant、Seger 和 Beatty[18, 37]开发了基于符号模拟的，并使用受限的线性时序逻辑表示性质刻画的模型检测算法，这种算法中性质刻画由使用受限的时序逻辑表示的前提-结论对表达。其中，前提用来限制电路的输入和初始状态，结论给出用户希望检测的性质。这种逻辑的公式形式如下：

$$p_0 \wedge \mathbf{X}p_1 \wedge \mathbf{X}^2 p_2 \wedge \cdots \wedge \mathbf{X}^{n-1} p_{n-1} \wedge \mathbf{X}^n p_n$$

相对于大多数刻画程序和电路的时序逻辑而言，此公式的语法是高度约束的，它只允许使用合取逻辑运算和下一个时间(**X**)的时序运算。但约束也有其优点，通过限制可处理的公式类型，某些性质就有可能被高效检测。

## 1.6 偏序约简

软件验证给模型检测带来了一些问题，软件的结构化特性不如硬件强，而且并发软件通常是异步的，也就是说，大多数由不同进程表现的行为都独立进行，不存在全局的同步时钟。由于这些原因，对于软件而言，状态爆炸现象特别显著。因此，比起硬件验证，软件验证较少使用模型检测方法。最近，在缓解软件的状态爆炸问题上取得了相当大的进步。缓解此问题的最成功的技术基于偏序约简[126, 209, 244]。这些技术着眼于并发事件间的独立性。当两个事件以任意顺序发生均导致若干相同的全局状态时，这两个事件是相互独立的。

交叉模型是并发软件的常用模型。在交叉模型中，独立发生的所有事件被排列成一个称为交叉序列的线性序列。因此，某独立事件的发生相对于其他独立事件是随意的，但大多数刻画并发系统的逻辑都对以不同顺序发生的独立事件分别考虑，即对所有独立事件的交叉发生都予以考虑，这就导致了一个极大的状态空间。

偏序约简技术致力于减少必须考虑的交叉序列数目，从而可能约简模型的状态空间。如果两个交叉序列除了独立事件发生的顺序不同，其他均一致，而且它们不会引起某性质刻画的变化(即性质刻画不能区分这两个交叉序列)时，那么对其中之一进行分析就足够了。这种方法来自于程序执行的偏序模型理论。依据这个模型，并发事件不需要排序。每一个偏序执行都对应多个多交叉序列。如果性质刻画不能区分这样的序列，那么选择其中之一就足够了。

将状态间的变迁关系独立交叉执行的所有方式看作一个集合，通过只选取一个子集作为减少状态空间策略的方法已经被许多学者所研究。第一批提出这种约简技术的学者包括 Over-