随机模型检测理论与应用

周从华　著

科学出版社

随机模型检测理论与应用

周从华　著

科学出版社

北　京

内 容 简 介

本书是作者多年从事随机模型检测相关科研工作的结晶。全书致力于缓解随机模型检测中的状态空间爆炸问题，深入系统地论述克服状态空间爆炸的两种基本技术：限界模型检测技术与抽象技术。首先，介绍离散时间马尔可夫链、马尔可夫决策过程、连续时间马尔可夫链和概率实时解释系统中的限界检测技术。然后，讨论模型检测概率、实时认知时态逻辑中的二值与三值抽象技术。最后，探讨随机模型检测技术在云计算和物联网领域的应用。

本书可作为高等院校计算机专业高年级本科生和研究生的教材，也可供相关领域的科研人员参考。

图书在版编目(CIP)数据

随机模型检测理论与应用/周从华著. —北京：科学出版社，2014.9

ISBN 978-7-03-041892-0

Ⅰ.①随… Ⅱ.①周… Ⅲ.①随机过程-数学模型-检测 Ⅳ.①O211.6

中国版本图书馆 CIP 数据核字(2014)第 211417 号

责任编辑：王 哲/责任校对：宋玲玲

责任印制：肖 兴/封面设计：迷底书装

科学出版社出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

新科印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2014 年 9 月第 一 版 开本：720×1 000 1/16

2014 年 9 月第一次印刷 印张：13 1/2

字数：272 000

定价：62.00 元

(如有印装质量问题，我社负责调换)

前　言

对模型检测的研究始于20世纪80年代初Clarke、Emerson等提出的并发系统自动化验证技术，目标在于保证计算机软硬件系统的正确性、可靠性和安全性。与模拟、仿真、测试三种方法相比，模型检测的主要优势是简洁明了和自动化程度高，因此成为近三十年来计算机科学研究的焦点，我国也非常重视模型检测技术的发展。国家自然科学基金委员会于2007年9月首次发布“可信软件基础研究”重大研究计划项目指南，拟在2008年1月至2015年12月在可信软件基础研究方面投入1.5亿元，旨在解决由相对不可信构件开发可信软件和可信软件运行保障的问题。国家重点基础研究发展计划(973计划)2014年重点支持安全攸关软件系统的共性理论和构造方法：面向安全攸关软件系统开发的重大需求，研究安全攸关软件系统的建模原理、构造方法及其运行与演化机理；研究软件安全性评测的理论与方法。这些项目的设立推进了模型检测技术的发展。

目前学术界在模型检测领域的研究主要集中于时序逻辑、模型检测算法和时空效率。时序逻辑的研究主要关注设计表达力强的逻辑语言，以实现对系统各种性质的形式化描述。模型检测算法主要研究时序逻辑满足性判定算法。时空效率主要研究模型检测算法的优化以及状态空间约简技术。模型检测通过遍历系统的状态空间完成属性的验证。然而，对于并发系统，其状态空间随着并发分量的增加呈指数级增长，这就是状态空间爆炸问题。该问题是模型检测技术从学术界走向工业界的主要瓶颈。为了能够在实际应用中更好地使用模型检测技术，很多学者提出了多种有效的技术来约简状态空间。

模型检测技术最初关注的是系统行为的绝对正确性，如系统不能进入死锁状态。然而，分布式算法、多媒体协议、容错系统等往往关心某种量化属性，如消息传送失败的概率不高于1%等。随机模型检测致力于解决这类属性的自动化验证问题。本书致力于缓解随机模型检测中的状态空间爆炸问题。首先，介绍离散时间马尔可夫链、马尔可夫决策过程、连续时间马尔可夫链和概率实时解释系统中的限界检测技术；然后，讨论模型检测概率、实时认知时态逻辑中的二值与三值抽象技术；最后，从应用出发探讨随机模型检测技术在云计算和物联网领域的应用。

2001年秋，我在南京大学开始攻读硕士学位，师从丁德成教授，从事数理逻辑领域的研究。2003年硕博连读，开始从事模型检测领域的研究。博士期间主要研

究了基于命题公式满足性求解的限界模型检测技术的优化与扩展。2006 年 6 月底，我来到江苏大学计算机科学与通信工程学院工作，并将模型检测技术应用于隐蔽信道的搜索。在应用的过程中，我深刻体会到状态空间爆炸对模型检测进一步走向实际应用的严重影响。因此，2008 年底我将研究重点转移到状态空间约简上，并聚焦于随机模型检测中的空间缓解技术。

本书是我从攻读博士学位起 11 年时间内科研工作的系统表述，包含本人与合作者共同完成的相关研究，有缓解状态空间爆炸方面的理论探索，也有在云计算与物联网新型系统领域的应用。本书的研究得到了国家自然科学基金项目“信息流安全属性算术验证的研究(61003288)”和江苏大学青年学术带头人培育项目的资助。

限于作者水平，加之时间仓促，书中难免存在不足之处，恳请广大读者批评指正。

周从华

2014 年 5 月 18 日

目　录

第 1 章　随机模型检测概述

1.1　模 型 检 测

计算机软硬件系统随着技术的进步变得日益复杂，如何保证系统的正确性、可靠性与安全性已经成为日益紧迫的问题。对于并发系统，其内在的非确定性决定了解决这个问题的难度。在过去的几十年间，各国研究人员为解决这个问题付出了巨大的努力，取得了重要的进展。在为此提出的诸多理论和方法中，模型检测[1-2]以其自动化程度高、遍历全局空间形成的验证完备性而引人注目。

模型检测的研究始于 20 世纪 80 年代初，Clarke 等提出用于描述并发系统性质的计算树时态逻辑(Computation Tree Logic，CTL)[3]，设计了检测有穷状态系统是否满足给定 CTL 公式的算法，并实现了一个原型系统。这一工作为并发系统性质的自动化验证开辟了一条新的途径，成为近三十年来计算机科学基础研究的热点。随后出现的符号模型检测技术[4]使这一方法向实际应用迈出了关键的一步。模型检测已被广泛应用于计算机硬件、通信协议、控制系统、安全认证协议等方面的分析与验证中，取得了令人瞩目的成果，并从学术界辐射到了产业界。许多公司(如 Microsoft、Intel、HP、Philips 等)成立了专门的小组负责将模型检测技术应用于软硬件系统的生产过程中。2007 年，Clarke、Emerson 和 Sifakis 三名知名学者因为在模型检测领域的开创性工作，获得了 ACM 图灵奖。

模型检测的基本思想是用状态转换系统 M 表示系统的行为，用时态逻辑公式 ϕ 描述系统的性质。这样“系统是否具有所期望的性质”就转化为数学问题“状态转换系统 M 是否为公式 ϕ 的一个模型”，用公式表示为“$M \models \phi$?”。对于有穷状态转换系统，这个问题是可判定的，即可以用计算机程序在有限时间内自动验证。

模型检测的基本流程如图 1.1 所示。模型检测工具首先读入以某种形式化语言描述的系统模型和以时态逻辑公式描述的属性，然后调用模型检测算法判断属性是否成立。当属性不成立时，将提供反例说明属性为何不成立。

在模型检测中，一般使用称为 Kripke 结构的有限状态转换系统来描述系统的动态行为。设 Ap 为一个原子命题的集合，在集合 Ap 上定义的四元组 $M=(S, s_0, R, L)$ 称为 Kripke 结构，其中，S 为有限状态的集合；s_0 为初始状态；$R \subseteq S \times S$ 为系统中全局变迁关系的集合，这里的“全局”是指对任意 $s \in S$ 至少存在一个状态 $s' \in S$，使得 s' 满足 $R(s, s')$；$L: S \to 2^{Ap}$ 为状态标记函数，用来标记在该状态下值为真的原子命题的集合。

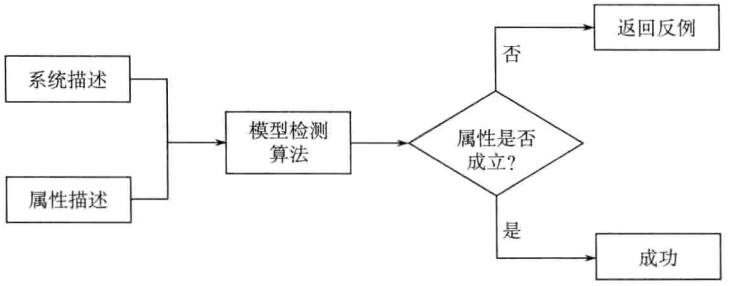

图 1.1 模型检测的基本流程

图 1.2 是一个定义在 Ap=$\{r,q,p\}$上的 Kripke 结构，其中，$S=\{s_0,s_1,s_2,s_3\}$，s_0 为初始状态，$R=\{(s_0,s_1),(s_1,s_2),(s_2,s_3),(s_3,s_2)\}$，$L(s_0)=\{r\}$，$L(s_1)=\{q\}$，$L(s_2)=\{p\}$，$L(s_3)=\{r\}$。

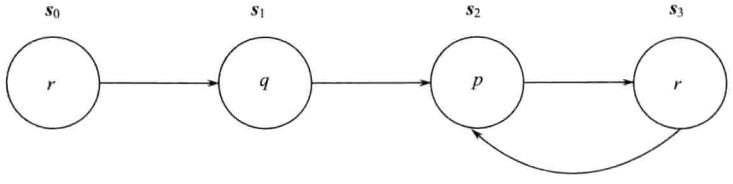

图 1.2 一个简单的 Kripke 结构

在 Kripke 结构 M 中，无穷状态序列 $\pi=s_0s_1s_2\cdots$ 称为路径，当且仅当对于任意 $i\geqslant 0$，有$(s_i,s_{i+1})\in R$。对于路径 π，π^i 表示从路径 π 中的第 i 个状态 s_i 开始的路径，$\pi(i)$表示状态 s_i。

CTL 是由 Clarke 和 Emerson 引入的作为规约有限状态系统的一种分支时态逻辑，其公式能够表述关于现在和未来状态的客观事实，具体语法定义如下。

定义 1.1(CTL) 设 Ap 为一个原子命题的集合，Ap 上的 CTL 递归定义如下。

(1) 如果 $p\in$Ap，则 p、$\neg p$ 为 CTL 公式。

(2) 如果 f、g 为 CTL 公式，则$\neg f$、$f\vee g$、$f\wedge g$、$f\rightarrow g$ 为 CTL 公式。

(3) 如果 f、g 为 CTL 公式，则 EXf、AXf、EFf、AFf、EGf、AGf、EfUg、AfUg、EfRg、AfRg 为 CTL 公式。

下面在 Kripke 结构上定义 CTL 的语义。

定义 1.2(CTL 的语义) 令 M 为一个 Kripke 结构，s 为 M 中的状态，f、g 为 CTL 公式，$M,s\models f$ 表示 f 在状态 s 处为真，这里满足性关系 $\models$ 递归定义如下。

(1) $M,s\models p$，当且仅当 $p\in L(s)$。

(2) $M,s\models\neg p$，当且仅当 $p\notin L(s)$。

(3) $M,s\models f\vee g$，当且仅当 $M,s\models f$ 或者 $M,s\models g$。

(4) $M,s\models f\wedge g$，当且仅当 $M,s\models f$ 且 $M,s\models g$。

(5) $M,s\models \mathrm{EX}f$，当且仅当存在一条从 s 出发的路径 π，使得 $M,\pi(1)\models f$。

(6) $M,s\models \mathrm{EF}f$，当且仅当存在一条从 s 出发的路径 π，使得存在自然数 k 满足 $M,\pi(k)\models f$。

(7) $M,s\models \mathrm{EG}f$，当且仅当存在一条从 s 出发的路径 π，使得对于任意自然数 k 满足 $M,\pi(k)\models f$。

(8) $M,s\models \mathrm{E}f\mathrm{U}g$，当且仅当存在一条从 s 出发的路径 π 满足：①存在自然数 i 使得 $M,\pi(i)\models g$；②对于任意 $j<i$，$M,\pi(j)\models f$。

(9) $M,s\models \mathrm{A}f\mathrm{U}g$，当且仅当对于任意一条 s 出发的路径 π 满足：①存在自然数 i 使得 $M,\pi(i)\models g$；②对于任意 $j<i$，$M,\pi(j)\models f$。

对于其他 CTL 公式存在等价关系：$\mathrm{AX}f\equiv\neg\,\mathrm{EX}\neg f$，$\mathrm{AF}f\equiv\neg\,\mathrm{EG}\neg f$，$\mathrm{AG}f\equiv\neg\,\mathrm{EF}\neg f$，$\mathrm{A}f\mathrm{R}g\equiv\neg\,\mathrm{E}(\neg f\mathrm{U}\neg g)$，$\mathrm{E}f\mathrm{R}g\equiv\neg\,\mathrm{A}(\neg f\mathrm{U}\neg g)$。

1.2　状态空间约简

模型检测基于对系统状态空间的穷举搜索完成系统的验证。对于并发系统，其状态的数目往往随着并发分量的增加呈指数级增长，因此当一个系统的并发分量较多时，直接对其状态空间进行搜索实际上是不可行的，这就是所谓的状态空间爆炸问题，也是模型检测方法从学术界走向工业界的主要瓶颈，为此研究人员提出了多种有效的方法来克服状态空间爆炸问题，下面逐一介绍这些方法。

1.2.1　基于有序二叉决策图的符号化模型检测方法

1987 年在卡内基梅隆大学读博士的 McMillan 首次把有序二叉决策图(Ordered Binary Decision Diagram，OBDD)引入模型检测，形成了符号化模型检测技术[4]，其基本思想是通过蕴涵的办法，用有序二叉决策图来表示模型检测中转移关系、可达状态集合，以此来进行状态像、不动点的计算。自此以后，基于有序二叉决策图的优化技术不断被其他研究者提出，使得可以验证状态规模达到 10^{20} 的系统[5]，并且利用符号模型检测的方法对高性能计算机中的总线协议进行了模型检测。这是模型检测在工业界最典型的应用。

尽管符号模型检测极大地拓展了所要验证系统的规模，但是在验证一些异步系统时，例如，异步时序系统以及通信系统的协议，显式的模型检测常优于符号模型检测。其原因在于在进行可达状态遍历时，符号模型检测采用的是广度优先算法，而传统的显式模型检测由于采用深度优先算法而往往更容易发现反例。另外符号模型检测采用的是全局搜索方式，当在进行状态原像计算的时候，不能够避免

实际上并不可达的状态。

此外，在最坏的情况下，有序二叉决策图的大小会随着变量数的增加呈指数级增长，如组合乘法。有序二叉决策图的大小同时依赖于变量的顺序，寻找较优的变量序的时间复杂相当高，甚至有的布尔函数不存在最优的变量序。因此，基于有序二叉决策图的符号化模型检测方法的通用性是比较有限的。

1.2.2 基于命题公式可满足性判定的限界模型检测方法

所谓基于命题公式可满足性判定的限界模型检测方法[6]就是在系统的部分运行空间上检测全称片断属性的失效性，并把属性的失效归约到命题公式的可满足性判定上。基于 DPLL(Davis-Putnam-Logemann-Loveland)算法的命题公式可满足性判定过程中不存在状态空间快速增长的问题，且命题公式可满足性判定工具可以处理具有几千个变量的公式。这两点保证了限界模型检测方法的有效性。

限界模型检测技术在工业界成功运用的范例包括以下几个。

(1) 以 SATO(Satisfiability Testing Optinized)和 GRASP(Generic Search Algorithm for the Satisfiability Problem)作为主要的可满足问题引擎，Motorala 的 PowerPC 微处理器的安全性通过限界模型检验的方法得到验证。

(2) 在对 Compaq 的 Alpha 的微处理器的模型检测中，发现某些“臭虫”的时间从几天缩短至几分钟。在该工具中应用的可满足性问题(Satisfiability Problem，SAT)求解引擎是 GRASP 和 PROVER(Proof and Verifier)。

(3) Thunder 和 Forecast 是 Intel 公司进行限界模型检验的工具。前者基于求解可满足性问题的引擎 SIMO(Satisfiability Internal Module Object)，而后者基于有序二叉决策图。前者应用于 Pentium Ⅳ 的实际设计验证。经过实验对比，前者因使用了求解可满足性问题的引擎，在验证 Pentium Ⅳ 设计时，生产量和生产率两方面明显优越于 Forecast。

2003 年，Penczek 等提出在多智体系统中验证时态认知逻辑(全称认知树逻辑(Universal Fragment of Computation Tree Logic of Knowledge，ACTLK)，认知计算树逻辑(Computation Tree Logic of Knowledge，LTLK)的全称片断)的限界模型检测方法[7]，并开发了相应的限界模型检测工具 BMCIS(Bounded Model Checking Interpreted Systems)。ACTLK 是在 ACTL 中引入知道、全都知道、分布知识和公共知识 4 个认知算子后得到的。2006 年，苏开乐等在时态逻辑 CTL* 的语言中扩展认知算子(知道、全都知道、分布知识和公共知识算子)，从而得到一个新的时态认知逻辑 $ECKL_n$，并给出了 $AECKL_n$($ECKL_n$ 的全称片断)限界模型检测算法及其正确性证明[8]。

对于 CTL，周从华等应用限界模型检测的思想提出了一项新的基于量化布尔公式(Quantified Boolean Formulas，QBF)的符号化模型检测技术。其基本思想是

在有限的运行空间上定义 CTL 的有效性，并将有效性的检测归约到 QBF 的满足性的判定上，使得 QBF 是可满足的，当且仅当 CTL 公式在有限运行空间上是有效的。这种方法是可靠的，即 CTL 在有限运行空间上的有效性是对其在无穷空间上有效性的逼近，同时也是完备的，即 CTL 在无穷运行空间上是有效的，一定存在一个有限运行空间使得 CTL 在该有限空间上是有效的。基于命题公式满足性判定的限界模型检测算法的成功很大程度上归功于基于 DPLL 的满足性判定算法在判定满足性的过程中不会出现空间的快速增长。同理，基于 DPLL 的 QBF 算法在检查满足性的判定过程中也不会出现空间的快速增长。另外，已经存在 QBF 工具能够处理具有上千个变量的 QBF。这两点保证了基于 QBF 方法的有效性。

当前，基于命题公式可满足性判定的限界模型检测方法已经在集成电路、软件的可靠性的验证上获得了成功应用。但是其通用性仍然比较有限，由于该方法是通过寻找使属性失效的反例来达到说明属性不成立的目的，反例太长导致相应的命题公式太大，从而命题公式可满足性判定工具无法处理，所以该方法实际上是一种证伪的方法。

1.2.3　抽象方法

抽象方法[9]的基本思想是剔除系统模型中与待验证属性无关的变量，保留相关变量，在此基础上构造一个状态空间较小的抽象模型，并运行模型检测于抽象模型上。抽象可以分为上近似抽象和下近似抽象。所谓上近似抽象是指原始系统的行为是抽象系统行为的子集，因此对于上近似抽象，当全称属性在抽象系统中成立时，在原始系统中也成立，但是当全称属性不成立时，则需要检测反例的真假。如果反例为假，则需要对抽象模型求精。所谓下近似抽象是指原始系统的行为包含抽象系统行为，因此对于下近似抽象，当片断属性在抽象系统中成立时，在原始系统中也成立。

抽象方法成功与否在很大程度上依赖于对相关变量的选择，变量选取不当会造成虚假的反例。Clarke 等于 2000 年首次提出完全自动化的、反例导引的抽象精化方法[10]，该方法的主要思想如下。

(1) 在原始模型的基础上进行上近似抽象，剔除部分状态变量，得到一个相对简单的抽象模型。在抽象模型上成立的全称属性在原始模型上必然成立。

(2) 如果全称属性在抽象模型上不成立，则必然存在反例，称为抽象反例。将该反例进行反向映射，以测试是否存在一个原始模型上的反例与抽象反例相对应，若是则断定全称属性在原始模型上不成立。

(3) 否则，运行精化算法，选择一部分原先被剔除的状态变量，并将它们插入抽象模型，然后重新运行上近似抽象算法，并回到第(1)步。

Clarke 等的方法是基于有序二叉决策图进行抽象精化操作的，近年来，其他

方式的抽象精化方法也被提出来，包括基于整数线性规划和机器学习的方法、基于SAT的方法，这些方法的目的在于防止基于有序二叉决策图的抽象精化操作遇到的空间爆炸问题发生。实验结果证明，近年来在SAT求解器方面的进展确实极大地改善了抽象精化方法的时间和空间效率。

谓词抽象[11]是最近出现的一种新的抽象方法，该方法仅针对程序中的谓词进行抽象和精化操作，其被广泛应用于处理大值域（如C语言中的32位整数）和无限值域问题（如物理过程中的实数值域），能够有效防止基于有序二叉决策图的抽象方法中遇到的空间爆炸问题。谓词抽象的基本思想是，利用一组谓词在原始模型的状态空间上定义一个等价关系，通过状态集合之间的映射，把一个大规模的或者包含无穷多个状态的原始模型转换为一个易于处理的、包含有限状态的抽象模型，在抽象模型中成立的属性在原始模型中也成立。

在谓词抽象中，使用尽可能少的谓词构造抽象模型，这是提高速度的关键。文献[12]讨论了剔除冗余谓词的策略。文献[13]讨论了使用0-1约束优化的方法，最大限度地剔除冗余谓词。上述方法的精化操作都需要反例的导引，然而McMillan提出一种无须反例的精化方法，该方法通过分析限界模型检测无解的原因来构造抽象模型，并将该抽象模型用于进一步的非限界模型检测。

1.2.4 组合验证

现在的软硬件通常由很多子系统构成，在验证由这些子系统构成的系统时，一个自然的想法就是分而制之，即先验证子系统，再分析集成系统，由此组合验证方法应运而生[14]。组合验证的主要思想可以概括为：将待验证的大规模系统分解为小型子系统，并将待验证的属性分解为各个子系统上对应的子属性，验证每个子属性时，其他子属性被作为天然成立的环境假设。

McMillan成功地将组合验证方法应用于验证许多大型设计，包括微处理器的乱序执行单元[15-16]和多处理机的高速缓存一致性协议[17]。该方法的主要缺点在于，用户需要手工定义子系统的划分和属性划分，这在一定程度上降低了模型检测方法的自动化优势。Cobleigh等于2003年首次提出自动化的组合验证方法，该方法自动分析失效的环境假设并加以改进，重新开始组合验证[18]。

1.2.5 其他约简方法

除了上述几种方法，还有on-the-fly模型检测技术[19-21]、偏序归约[22-27]以及对称模型检测技术[28-30]。on-the-fly模型检测技术的基本原理是根据需要展开系统路径所包含的状态，避免预先生成系统中所包含的所有状态。一个系统可以由多个进程组成，并发执行使得不同进程的动作可以有许多不同的次序，基于对这一问题的认识，某些状态的次序可以被固定，以减少重复验证本质上相同的路径，这种

方法称为偏序归约。由多进程组成的系统中，某些进程可能完全类似，并发执行的结果可能产生许多相同或相似的路径，基于对这一问题的认识，可以只搜索在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，这种方法称为对称模型检测。

1.3　线性时态逻辑的限界模型检测

1.3.1　示例

考虑一个由拥有 3 个位的移位寄存器 x 构成的状态机 M。x 的 3 个位分别表示为 $x[0]$、$x[1]$、$x[2]$。谓词 $R(x,x')=((x'[0]=x[1])\wedge(x'[1]=x[2])\wedge(x'[2]=1))$表示当前状态 x 和下一个状态 x'之间的转换关系。初始配置下寄存器 x 中每一个位的值是随机的，谓词 $I(x)$表示初始状态集合，且始终为真。

在连续的 3 次移位之后寄存器应该为空，即所有位的值应该为 0。但是在状态转换中对 $x[2]$人为地引入一个错误：用数值 1 代替数值 0。因此，属性“在充分的移位之后寄存器为空，即 $x=0$”不成立。该属性可利用线性时态逻辑（Linear Temporal Logic，LTL）公式形式化地表示为 $\mathrm{F}(x=0)$。全称模型检测问题 $\mathrm{AF}(x=0)$ 可转换为片断模型检测问题 $\mathrm{EG}(x\neq 0)$。因此，可以检查是否存在满足 $\mathrm{G}(x\neq 0)$的执行序列。与常规方法搜索任意长度的路径不同的是，搜索的范围可被限制在有穷长度的路径上，例如，由 3 个状态组成的路径。令 x_0、x_1、x_2 为路径上的 3 个起始状态，其中，x_0 为初始状态。因为 x 的初始值可以是任意随机值，所以对初始状态 x_0 没有限制。将转换关系展开两次可演绎出对应的命题公式 $f_m=I(x_0)\wedge R(x_0,x_1)\wedge R(x_1,x_2)$。将 R 和 I 扩展为

$$(x_1[0]=x_0[1])\wedge(x_1[1]=x_0[2])\wedge(x_1[2]=1)\wedge$$
$$(x_2[0]=x_1[1])\wedge(x_2[1]=x_1[2])\wedge(x_2[2]=1)$$

任何拥有 3 个状态的路径，如果其是 $\mathrm{G}(x\neq 0)$的证据，则必须包含循环。这样，需要一个从状态 x_2 到任意一个状态的转换（可以是初始状态 x_0，或者 x_1，或者 x_2 自身）。引入记号 L_i 表示 $R(x_2,x_i)$，即 $L_i=((x_i[0]=x_2[1])\wedge(x_i[1]=x_2[2])\wedge(x_2[2]=1))$。现在如果保证该路径满足 $\mathrm{G}(x\neq 0)$，则属性 $S_i=(x_i\neq 0)$必须在每个状态下成立，这里 $x_i\neq 0$ 等价于 $(x_i[0]=1)\vee(x_i[1]=1)\vee(x_i[2]=1)$。

将上述命题公式聚合在一起，可得

$$f_m\wedge\bigvee_{i=0}^{2}L_i\wedge\bigwedge_{i=0}^{2}S_i$$

该公式是可满足的，当且仅当公式 $\mathrm{F}(x=0)$存在长度为 2 的反例。在本例中，指派