模型检测:理论、方法与应用

林惠民,张文辉
(中国科学院软件研究所,北京 100080)

摘 要: 随着计算机软硬件系统日益复杂,如何保证其正确性和可靠性成为日益紧迫的问题.在为此提出的诸多理论和方法中,模型检测(model checking)以其简洁明了和自动化程度高而引人注目.模型检测的研究大致涵盖以下内容:模态/时序逻辑、模型检测算法及其时空效率(特别是空间效率)的改进以及支撑工具的研制.这几个方面之间有着密切的内在联系.不同模态/时序逻辑的模型检测算法的复杂性不一样,优化算法往往是针对某些特定类型的逻辑公式.本文将就这几个方面分别加以阐述,最后介绍该领域的新进展.

关键词: 系统可靠性; 模态/时序逻辑; 模型检测

中图分类号: TP311.1 **文献标识码:** A **文章编号:** 0372-2112 (2002) 12A-1907-06

Model Checking: Theories, Techniques and Applications

LIN Hui-min, ZHANG Wen-hui
(*Institute of Software, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract: As computer hardware and software systems become more and more complex, how to assure the correctness and reliability of such system sbecomes an urgent problem. Among theories proposed as solutions to this problem, model checking has become a very attractive and appealing approach, because of its simplicity and high level of automation. Research on model checking covers the following subjects: modal/temporal logics, model checking algorithms, efficiency of model checking with respect to time and space (especially space complexity), and development of model checking tools. These aspects are closely related. Complexities of model checking algorithms vary very much for different modal/temporal logics, and optimizations are often targeted at certain types of logic formulas. Some new achievements and research directions are also discussed.

Key words: system reliability; modal/temporal logics; model checking

1 引言

随着计算机软硬件系统日益复杂,如何保证其正确性和可靠性成为日益紧迫的问题.对于并发系统,由于其内在的非确定性,这个问题难度更大.在过去二十多年间,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(model checking)以其简洁明了和自动化程度高而引人注目.

模型检测的研究始于八十年代初,当时 Clarke, Emerson 等人提出了用于描述并发系统性质的 CTL 逻辑,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.这一工作为对并发系统的性质自动进行验证开辟了一条新的途径,成为近二十年来计算机科学基础研究的一个热点.随后不久出现的符号模型检测技术使这一方法向实际应用性迈出了关键的一步.模型检测已被应用于计算机硬件、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功,并从学术界辐射到了产业界.许多大公司,如 Intel、HP、Phillips 等成立了专门的小组负责将模型检测技术应用于生产过程中.Bryant, Clarke, Emerson 和 McMillon 因模型检测的创始性工作获得了 1998 年 ACM Paris Kanellakis Award for Theory and Practice.

模型检测的基本思想是用状态迁移系统(S)表示系统的行为,用模态/时序逻辑公式(F)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"状态迁移系统 S 是否是公式 F 的一个模型?",用公式表示为 $S \models F$?.对有穷状态系统,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确定.

模型检测的研究大致涵盖以下内容:

模态/时序逻辑、模型检测算法及其时空效率(特别是空间效率)的改进以及支撑工具的研制.这几个方面之间有着密切的内在联系.不同模态/时序逻辑的模型检测算法的复杂性不一样,优化算法往往是针对某些特定类型的逻辑公式.本文将就这几个方面分别加以阐述,最后介绍该领域的新进展.

收稿日期:2002-08-16;修回日期:2002-10-16
基金项目:国家自然科学基金(No.6983320)

2 模态逻辑

模态/时序逻辑是模型检测的基础.这里介绍三种常用的模态逻辑:计算树逻辑[1],命题线性时序逻辑[2]和命题 μ - 演算[3,4].

2.1 计算树逻辑

一个系统的运行可以看成是系统状态的变化.系统状态变化的可能性可以表示成树状结构.比如说一个并发系统从一个初始状态开始运行,由于行为的不确定性,它可以有多个可能的后续状态,每个这样的状态又可以有多个可能的后续状态,依此类推可以产生一棵状态树.

计算树逻辑(CTL-Computation Tree Logic)是一种分叉时序逻辑.CTL 可以描述状态的前后关系和分枝情况.描述一个状态的基本元素是原子命题符号.CTL 公式由原子命题,逻辑连接符和模态算子组成.CTL 的逻辑连接符包括:¬(非),∨(或),∧(与).它的模态算子包括:E(Exists),A(Always),X(Next-time),U(Until),F(FutureI,G(Global).E 表示对于某一个分枝,A 表示对于所有分枝,X 表示下一状态,U 表示直至某一状态,F 表示现在或以后某一状态,G 表示现在和以后所有状态.前两个算子描述分枝情况,后四个算子描述状态的前后关系.CTL 中描述分枝情况和描述状态的前后关系的算子成对出现,即一个描述分枝情况的算子后面必须有一个描述状态的前后关系的算子.

CTL 公式的产生规则如下:原子命题是 CTL 公式;如果 p,q 是 CTL 公式,则 $\neg p, p\vee q, p\wedge q, EXp, E(pUq), EFp, EGp, AXp, A(pUq), AFp, AGp$ 是 CTL 公式.例如公式 EFp 表示:一定会沿某条路径达到一个满足 p 的状态.对 CTL 公式存在线性时间的模型检测算法,即算法的最坏时间复杂度与 $|S|\cdot|F|$ 成正比,这里 $|S|$ 是状态迁移系统的大小,$|F|$ 是逻辑公式的长度[1].

2.2 命题线性时序逻辑

系统状态变化的可能性可以看成是一种树状结构.同时它又可以看成是所有可能的系统初始状态经历各种可能变化的集合,也就是把一颗树看成是有限或无限条路径的集合.一条路径所代表的是系统的一次可能的运行情况.

命题线性时序逻辑(PLTL-Propositional Linear Temporal Logic)关心的是系统的任意一次运行中的状态以及它们之间的关系.PLTL 公式由原子命题,逻辑连接符和模态算子组成.PLTL 的逻辑连接符包括:¬,∨,∧.它的模态算子包括:◇(Eventually),□(Always).◇表示现在或以后某一状态(类似 CTL 的 F),□表示现在和以后所有状态(类似 CTL 的 G).

PLTL 公式的产生规则如下:原子命题是 PLTL 公式;如果 p,q 是 PLTL 公式,则 $\neg p, p\vee q, p\wedge q, \Diamond p, \Box p$ 是 PLTL 公式.例如公式 $\Diamond\Box p$ 表示:某个时刻后所有的状态都满足 p.

PLTL 和 CTL 的表达能力不同,各有各的长处.PLTL 模型检测的常用方法是将所要检测的性质即 PLTL 公式的补转换成 Büchi 自动机,然后求其与表示系统的自动机的交.如果交为空,则说明系统满足所要检测的性质;否则生成一个反例,说明不满足的原因[2].

2.3 命题 μ - 演算

以上介绍的两种逻辑语言关心的是系统的状态以及它们之间的关系.现在我们介绍一种面向动作的逻辑语言,即 μ - 演算.

系统状态的改变总是某种动作引起的.μ - 演算关心的是系统的动作与状态之间的关系.描述动作的基本元素是动作符号.μ - 演算公式由原子命题,命题变量,逻辑连接符,模态算子和不动点算子组成.逻辑连接符包括:∨,∧,对于每个动作符号 a,有两个模态算子[a]和⟨a⟩.[a]表示对所有的 a 动作,⟨a⟩表示对某个 a 动作.不动点算子有最小不动点算子 μ 和最大不动点算子 ν.

μ - 演算公式的产生规则如下:原子命题和命题变量是 μ - 演算公式;如果 p,q 是 μ - 演算公式,则 $p\vee q, p\wedge q, [a]p, \langle a\rangle p, \mu X.p, \nu X.p$ 是 μ - 演算公式,例如公式 $\mu X.(p\vee[a]X)$ 表示:在任何无穷的 a - 路径都存在某个满足 p 的状态.

μ - 演算的主要缺点是公式不易读懂(由于最小、最大不动点的交错嵌套),其优点是它的表示能力非常强.CTL 和 PLTL 都可以嵌入到它的真子集中,并且相应的子集具有与 CTL 和 PLTL 相同的复杂度的模型检测算法.因此很多学者将 μ - 演算作为模型检测的一般框架加以研究.

3 模型检测方法的应用

早期的模型检测侧重于硬件设计的验证,随着研究的进展,模型检测的应用范围逐步扩大,涵盖了通讯协议、安全协议、控制系统和一部分软件.电子线路设计验证的例子包括先进先出存储器的验证,验证的性质包括输入和输出的关系[5].浮点运算部件的验证,验证的性质包括计算过程所需满足的不变式[6].

对于复杂的协议和软件,模型检测的验证基于抽象和简化的模型,从验证角度来讲,这并非十分严谨.更为确切的说法应该是协议和软件的分析.协议验证的例子包括认证协议的验证,验证的性质包括对通话双方的确认[7~9];合同协议的验证,验证的性质包括公平和滥用的可能性[10];缓存协议的验证,验证的性质包括数据的一致性和读写的活性[11].

对于软件,由于软件的复杂和软件运行中可能到达的状态个数通常没有实质上的限制,验证通常局限于软件的某些重要组成部分.软件验证的例子包括飞行系统软件的验证,验证的性质包括系统所处的状态和可执行动作之间的关系[12,13];铁路信号系统软件的验证,验证的性质包括控制信号与控制装置状态的关系[14,15].

模型检测还可以应用于其他方面,其基本思想是将一个过程或系统抽象成一个有穷状态模型,加以分析验证.这方面的例子包括化学过程验证,验证的性质包括阀门、管道和容器的状况[16];公司操作过程分析,分析的内容包括操作过程是否具有所需的功能[17];电站操作程序的验证,验证的性质包括操作程序的动作前后关系和操作是否安全可靠[18].

4 状态爆炸问题

模型检测基于对系统状态空间的穷举搜索.对于并发系

统,其状态的数目往往随并发分量的增加呈指数增长.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在实际上是不可行的.这就是所谓的状态爆炸问题.

为了能够有效地应用模型检测方法,我们需要研究减少和压缩状态空间的方法.文献上关于这方面的文章很多,其中的很多方法已经实现在不同的模型检测工具中.

符号模型检测的基本原理是将系统的状态转换关系用逻辑公式表示[19].在这方法中,二叉图(Binary Decision Diagram)是用以表示逻辑公式的重要手段[20],它能较为紧凑地表示状态转换关系,以降低系统模型所需的内存空间.另外,状态转换的计算可以以集合为单位,以提高搜索的效率.

On-the-fly 模型检测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[21].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,我们可以将某些状态的次序固定,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[22,23].

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,我们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[24,25].

一般来讲能够减少搜索空间的方法能同时节省时间和内存空间的需求.由于内存空间在某些情况下比时间更为重要,有些方法的目标是以时间换空间.例如在 SPIN[21] 中实现的压缩方法和用自动机表示可达状态的方法[26]就是这样的方法.

除了在模型检测的过程应用不同方法以增加效率和减少内存空间的需求外,还有许多研究的目标是减少模型本身或验证性质的复杂性.这方面的方法有不同类型的抽象[27~29],程序切片[30~32],模型分解[33,34],验证性质的分解[35].抽象的基本想法是抽掉系统中的细节、用尽可能少的状态来刻划系统的运作过程.程序切片基本的想法是将程序中不影响所要验证的性质的语句去掉以减少模型复杂性.模型分解的想法是将一个模型分解成若干部分,或者分别验证,或者提供一个较好的组合方法以降低验证的复杂性.同样,一个需要验证的性质也有可能分解成若干部分,然后分别验证.

5 模型检测工具

模型检测的优点在于可以完全自动地进行验证,这一方法的成功在很大程度上应归功于有效的软件工具的支持.以下介绍几个验证不同类型逻辑公式的软件工具.

SMV[36]是美国卡耐基梅隆大学开发的模型检测工具.用以检测一个有限状态系统是否满足 CTL 公式.它的建模方式是以模块为单位,模块可以同步或异步(interleaving)组合.模块描述的基本要素包括非确定性选择,状态转换和并行赋值语句.其模型检测的基本方法是以二叉图表示状态转换关系,以计算不动点的方法检测状态的可达性和其所满足的性质.

SPIN[21,37]是美国贝尔实验室开发的模型检测工具.用以检测一个有限状态系统是否满足 PLTL 公式及其他一些性质,包括可达性和循环.它的建模方式是以进程为单位,进程异步组合.进程描述的基本要素包括赋值语句,条件语句,通讯语句,非确定性选择和循环语句.其模型检测的基本方法是以自动机表示各进程和 PLTL 公式,以计算这些自动机的组合可接受的语言是否为空的方法检测进程模型是否满足给定的性质.

CWB[38]的不同版本是英国爱丁堡大学和美国北卡洛莱纳州立大学相继开发的模型检测工具.用以检测系统间的等价关系,PRE-ORDER 关系,以及系统是否满足 μ - 演算公式.它的建模方式是用 CCS 语言或 LOTOS 语言(的子集).由于建模语言的局限性和模型检测方法复杂度较高,这个工具的可应用性比前面所述的工具要差一些.

以上所述是一些开发较早的模型检测工具中的几个例子,有些新的模型检测工具直接面向编程语言,比如贝尔实验室开发的 VeriSoft[39]等.模型检测工具很多,其余不一一列举.另外,需要一提的是模型检测工具与大型软件开发工具的结合.比如,瑞典 Telelogic 公司的软件开发工具(TAU)包含系统建模,系统测试以及模型检测工具.这个软件开发工具主要面向通讯、实时系统,其模型检测工具的基本原理与 SPIN 相同.

6 新近进展

6.1 实时和混成系统的模型检测

作为模型检测基础的传统模态/时序逻辑可以方便地表示离散事件或状态是否发生以及发生的先后顺序,而在许多实际应用中人们不仅关心某一事件是否发生,而且还要求它在确定的时间间隔内发生,例如要求锅炉的水位一旦落到警戒线时进水阀门必须在 3 秒钟内开启.这类对相关事件发生的时间有明确要求的系统称为实时系统.实时系统涉及飞机控制系统、机器人、工业与军事控制系统等安全性至关紧要的应用领域.

九十年代以来,实时系统模型检测的研究取得了重大进展,主要表现在三个方面:

(1)出现了用于表示实时系统的各种数学模型,如时间 Petri 网[40,41],时间自动机[42,43],以及各种进程代数的时间扩充[44,45].其中以时间自动机的影响和应用最为广泛.

(2)提出能描述实时系统的模态/时序逻辑,如 CTL, PLTL 和 μ - 演算的实时扩充[46~48].例如实时 PLTL 公式 $\Box\Diamond_{\leq 3}(p)$ 表示"在任何时候系统总会在 3 个单位时间内达到满足性质 p 的状态".

(3)针对这些实时系统的数学模型和逻辑设计了各种模型检测的算法,并实现了相应的分析与验证工具,如 Uppall[49]、Kronos[50]、STeP[51]、HyTech[52]等.

6.2 模型检测与测试相结合

测试是保证软件产品可靠性和正确性的传统手段.与模型检测不同,测试是对选定的输入数据集运行系统,通过比较所产生的输出与预期值来判断系统是否会有错误.测试的主要局限性在于:对给定数据集通过了测试并不能保证在实际运行中对其它输入不发生错误.

与测试不同,模型检测不是针对某组输入,而是面向某类

性质来检查系统是否合乎规约.在系统不满足所要求的性质时,模型检测算法会产生一个反例(一般是一条执行路径)来说明不满足的原因.这一功能与测试有异曲同工之处.如何将模型检测与测试技术相结合,使两者相辅相成,是一个十分值得注意的研究方向[53].

6.3 软件模型检测

模型检测在硬件和通讯协议的分析与验证中已经取得了很大的成功.如何将这一技术应用于软件,是目前非常活跃的研究领域[54,55].

软件由于涉及无穷数据域上的运算而往往呈现出无穷状态,这是软件模型检测的主要难点.但是很多系统(如控制软件)并不涉及复杂的数值计算,往往只用到线性实数约束或整数加减等简单的运算,因此可以采用诸如线性约束求解等技术实现模型检测.另外也可应用抽象等技术映射到有穷域上进行模型检测.

6.4 一阶模态逻辑

传统的模型检测基于命题模态逻辑,如前面所介绍的CTL,PLTL和 μ - 演算.由于在实际问题中不可避免地要遇到数据与变量,为了用命题模态逻辑来描述性质,在建模时往往需要做很多简化.这不仅增加了工作量,而且性质的描述也不自然.

适用于带数据系统的逻辑是一阶逻辑(也称谓词演算).为了描述系统状态的改变以及并发进程之间的通讯,需要在经典的一阶逻辑中引入模态词;为了表示无穷的行为,需要引入递归(不动点)算子[56~58].对于无穷数据域,一阶模态逻辑的模型检测一般说来不终止.但是对一些特定的问题,往往可以用有限多个抽象数据对象来代表所有的数据对象(例如通讯协议中被传输的数据),因此可以进行模型检测[58].这一方法也可以应用于面向移动计算的形式模型,如 π - 演算和 ambient 演算[59,60].

参考文献:

[1] E M Clarke, E A Emerson, A P Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications[A]. 10th Annual ACM Symposium on Principles of Programming Languages (POPL 83)[C]. New York: ACM Press, 1983. 117 - 126.

[2] M Y Vardi, P Wolper. An automata - theoretic approach to automatic program verification[A]. 1st IEEE Symposium on Logic in Computer Science (LICS 86)[C]. Los Alamitos: IEEE Computer Society, 1986. 322 - 331.

[3] E Allen Emerson, Chin-Laung Lei. Efficient model checking in fragments of the propositional Mu-Calculus[A]. 1st IEEE Symposium on Logic in Computer Science (LICS 86)[C]. Los Alamitos: IEEE Computer Society, 1986. 267 - 278.

[4] Colin Stirling, David Walker. Local model checking in the modal Mu-Calculus[A]. Lecture Notes in Computer Science 351 - 3rd International Joint Conference on Theory and Practice of Software Development (TAPSOFT 89)[C]. Berlin: Springer-Verlag, 1989. 369 - 383.

[5] Jorg Bormann, Jorg Lohse, Michael Payer, Gerd Venzl. Model checking in industrial hardware design[A]. 32nd Conference on Design Automation (DAC 95)[C]. New York: ACM Press, 1995. 298 - 303.

[6] Yirng-An Chen, Edmund M Clarke, Pei-Hsin Ho, Yatin Vasant Hoskote, Timothy Kam, Manpreet Khaira, John W O'Leary, Xudong Zhao. Verification of all circuits in a floating-point unit using word-level model checking[A]. Lecture Notes in Computer Science 1166 - 1st International Conference on Formal Methods in Computer-Aided Design (FMCAD 96)[C]. Berlin: Springer-Verlag, 1996. 19 - 33.

[7] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR[J]. Software-Concepts and Tools, 1996, 17(3): 93 - 102.

[8] J C Mitchell, V Shmatikov, U Stern. Finite - state analysis of SSL 3.0 [A]. 7th USENIX Security Symposium (USENIX 98)[C]. Berkeley: USENIX, 1998. 201 - 215.

[9] Paolo Maggi, Riccardo Sisto. Using SPIN to verify security properties of cryptographic protocols[A]. Lecture Notes in Computer Science 2318 - 9th International SPIN Workshop on Model Checking of Software (SPIN 02)[C]. Berlin: Springer-Verlag, 2002. 187 - 204.

[10] Vitaly Shmatikov, John C Mitchell. Finite-state analysis of two contract signing protocols[J]. Theoretical Computer Science, 2002, 283(2): 419 - 450.

[11] Edmund M Clarke, Orna Grumberg, Hiromi Hiraishi, Somesh Jha, David E Long, Kenneth L McMillan, Linda A Ness. Verification of the futurebus + cache coherence protocol[J]. Formal Methods in System Design, 1995, 6(2): 217 - 232.

[12] R J Anderson, P Beame, S Burns, W Chan, F Modugno, D Notkin, J D Reese. Model checking large software specifications[R]. University of Washington, Department of Computer Science and Engineering. Report: TR-96-04-02, April, 1996.

[13] T Sreemani, J M Atlee. Feasibility of model checking software requirements[A]. 11th Annual Conference on Computer Assurance (COMPASS 96)[C]. Gaithersburg, Maryland: National Institute of Standards and Technology, 1996.

[14] E Cleaveland, G Luttegen, V Natarajan, S Sims. Modeling and verifying distributed systems using priorities: A case study[J]. Software Concepts and Tools, 1996, 17(2): 50 - 62.

[15] A Cimatti, F Giunchiglia, G Mongardi, D Romano, F Torielli, P Traverso. Model checking safety critical software with SPIN: an application to a railway interlocking system[A]. Lecture Notes in Computer Science 1516 - 17th International Conference on Computer Safety, Reliability and Security (SAFECOMP 98)[C]. Berlin: Springer-Verlag, 1998. 284 - 295.

[16] Adam L Turk, Scott T Probst, Gary J Powers. Verification of real time chemical processing systems[A]. Lecture Notes in Computer Science 1201 - International Workshop on Hybrid and Real-Time Systems (HART 97)[C]. Berlin: Springer-Verlag, 1997. 259 - 272.

[17] Henk Eertink, Wil Janssen, Paul Oude Luttighuis, Wouter B Teeuw, Chris A Vissers. A business process design language[A]. Lecture Notes in Computer Science 1708 - World Congress on Formal Methods (FM 99)[C]. Berlin: Springer-Verlag, 1999. 76 - 95.

[18] Wenhui Zhang. Model checking operator procedures[A]. Lecture Notes in Computer Science 1680 - Theoretical and Practical Aspects of SPIN Model Checking (SPIN 99a, 99b)[C]. Berlin: Springer-Verlag, 1999. 200 - 215.

[19] J R Burch, E M Clarke, K L McMillan, D L Dill, J Hwang. Symbolic model checking: 10^{20} states and beyond[A]. 5th IEEE Symposium on Logic in Computer Science (LICS 90)[C]. Los Alamitos: IEEE Computer Society, 1990.428 – 439.

[20] Randal E Bryant. Graph-based algorithms for boolean function manipulation[J]. IEEE Transactions on Computers, 1986, 35(8): 677 – 691.

[21] G J Holzmann. Design and Validation of Computer Protocols[M]. New Jersey: Prentice Hall, 1991.

[22] S Katz, D Peled. Verification of distributed programs using representative interleaving sequences[J]. Distributed Compnting, 1992, 6: 107 – 120.

[23] Gerard J Holzmann, Doron Peled. An improvement in formal verification [A]. 7th IFIP WG6.1 International Conference on Formal Description Techniques VII (FORTE 1994)[C]. London: Chapman and Hall, 1995.197 – 211.

[24] E A Emerson, A P Sistla. Symmetry and model checking[A]. Lecture Notes in Computer Science 697 – 5th International Conference on Computer Aided Verification (CAV 93) [C]. Berlin: Springer-Verlag, 1993.463 – 478.

[25] A Prasad Sistla, Viktor Gyuris, E Allen Emerson. SMC: a symmetry-based model checker for verification of safety and liveness properties [J]. ACM Transactions on Software Engineering and Methodology, 2000, 9(2): 133 – 166.

[26] Gerard J Holzmann, Anuj Puri. A minimized automaton representation of reachable states[J]. International Journal on Software Tools for Technology Transfer, 1999, 2(3): 270 – 278.

[27] Patrick Cousot, Radhia Cousot. Systematic design of program analysis frameworks[A]. 6th Annual ACM Symposium on Principles of Programming Languages (POPL 79)[C]. New York: ACM Press, 1979. 269 – 282.

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

[29] C Loiseaux, S Graf, J Sifakis, A Bouajjani, S Bensalem. Property preserving abstractions for the verification of concurrent systems[J]. Journal of Formal methods in System Design, 1995, 6: 1 – 35.

[30] Mark Weiser. Program slicing[J]. IEEE Transactions on Software Engineering, 1984, 10(4): 352 – 357.

[31] Jingde Cheng. Slicing concurrent programs - a graph-theoretical approach[A]. Lecture Notes in Computer Science 749 – 1st International Workshop on Automated and Algorithmic Debugging (AADEBUG 93)[C]. Berlin: Springer-Verlag, 1993.223 – 240.

[32] L I Millett, T Teitelbaum. Issues in slicing PROMELA and its applications to model checking, protocol understanding, and simulation[J]. International Journal on Software Tools for Technology Transfer, 2000, 2 (4): 343 – 349.

[33] Eugene W Stark. A proof technique for rely/guarantee properties [A]. Lecture Notes in Computer Science 206 – 5th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 85)[C]. Berlin: Springer-Verlag, 1985.369 – 391.

[34] E M Clarke, D E Long, K L McMillan. Compositional model checking [A]. 4th IEEE Symposium on Logic in Computer Science (LICS 89) [C]. Los Alamitos: IEEE Computer Society, 1989, 353 – 362.

[35] K Stahl, K Baukus, Y Lakhnech, M Steffen. Divide, abstract, and model-check [A]. Lecture Notes in Computer Science 1680 -Theoretical and Practical Aspects of SPIN Model Checking(SPIN 99a, 99b)[C]. Berlin: Springer-Verlag, 1999.57 – 76.

[36] K L McMillan. Symbolic model checking: an approach to the state explosion problem [R]. Carnegie-Mellon University, Department of Computer Science, Report CMU-CS-92-131.1992.

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

[38] R Cleavel, J Parrow, B Steffen. The concurrency workbench: a semantics-based verification tool for the verification of concurrent systems [J]. ACM Transactions on Programming Languages and Systems, 1993, 5(1): 36 – 72.

[39] Patrice Godefroid. VeriSoft: A tool for the automatic analysis of concurrent reactive software [A]. Lecture Notes in Computer Science 1254 – 9th International Conference on Computer Aided Verification (CAV 97)[C]. Berlin: Springer-Verlag, 1997.476 – 479.

[40] Bernd Walter. Timed Petri-nets for modelling and alalyzing protocols with real-time characteristics [A]. 3rd IFIP WG 6.1 International Workshop on Protocol Specification, Testing, and Verification (PSTV 83)[C]. Amsterdam: North-Holland, 1983.149 – 159.

[41] James E Coolahan Jr, Nick Roussopoulos. A timed Petri net methodolgoy for specifying real-time system timing requirements [A]. International Workshop on Timed Petri Nets (PNPM 85) [C]. Los Alamitos: IEEE Computer Society, 1985.24 – 31.

[42] David L. Dill. Timing assumptions and verification of finite-state concurrent systems [A]. Lecture Notes in Computer Science 407 -International Workshop on Automatic Verification Methods for Finite State Systems (89) [C]. Berlin: Springer-Verlag, 1990.197 – 212.

[43] Rajeev Alur, Costas Courcoubetis, David L Dill. Model-checking for Real-time systems[A]. 5ht IEEE Symposium on Logic in Computer Science (LICS 90)[C]. Los Alamitos: IEEE Computer Society, 1990. 414 – 425.

[44] Wang Yi. CCS + Time = An interleaving model for real time systems [A]. Lecture Notes in Computer Science 510 - 18th International Colloquium on Automata, Languages and Programming(ICALP 91)[C]. Berlin: Springer-Verlag, 1991.217 – 228.

[45] Steve Schneider, Jim Davies, D M Jackson, George M Reed, Joy N Reed, A W Roscoe. Timed CSP: theory and practice[A]. Lecture Notes in Computer Science 600 - Real-Time: Theory in Practice (REX Workshop 91) [C]. Berlin: Springer-Verlag, 1991.640 – 675.

[46] E Allen Emerson, Aloysius K Mok, A Prasad Sistla, Jai Srinivasan. Quantitative temporal reasoning[A]. Lecture Notes in Computer Science 531 -2nd International Workshop on Computer Aided Verification (CAV 90)[C]. Berlin: Springer-Verlag, 1991.136 – 145.

[47] Jonathan S Ostroff. Composition and refinement of discrte real-time systems [J]. ACM Transactions on Software Engineering and Methodology, 1999, 8(1): 1 – 48.

[48] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, Sergio Yovine. Symbolic model checking for real-time systems [A]. 7th IEEE Symposium on Logic in Computer Science (LICS 92)[C]. Los Alamitos: IEEE Computer Society, 1992.394 – 406.

[49] Kim G Larsen, Paul Pettersson, Wang Yi. UPPAAL in a nutshell [J].

International Journal on Software Tools for Technology Transfer, 1997, 1 (1 – 2): 134 – 152.

[50] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, Sergio Yovine. KRONOS: A model-checking tool for real-time systems[A]. Lecture Notes in Computer Science 1486 – 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems(FTRTFT 98)[C]. Berlin: Springer-Verlag, 1998. 298 – 302.

[51] Nikolaj Bjorner, Anca Browne, Eddie Chang, Michael Colon, Arjun Kapur, Zohar Manna, Henny Sipma, Tomas Uribe. STeP – The Stanford Temporal Prover, User's Manual [R]. Stanford University, Department of Computer Science, Report STAN-CS-TR-95-1562. 1995.

[52] Thomas A Henzinger, Pei-Hsin Ho, Howard Wong-Toi. HYTECH: A model checker for hybrid systems[A]. Lecture Notes in Computer Science 1254 – 5th International Conference on Computer Aided Verification (CAV 97)[C]. Berlin: Springer-Verlag, 1997. 460 – 463.

[53] Richard H. Carver, Kuo-Chung Tai. Use of sequencing constraints for specification-based testing of concurrent programs[J]. IEEE Transactions on Software Engineering, 1998, 24(6): 471 – 490.

[54] Klaus Havelund, John Penix, Willem Visser (eds). Lecture Notes in Computer Science 1885 – 7th International SPIN Workshop on SPIN Model Checking and Software Verification (SPIN 00)[C]. Berlin: Springer-Verlag, 2000.

[55] Jamieson M Cobleigh, Lori A Clarke, Leon J Osterweil. FLAVERS – A finite state verification technique for software systems[J]. IBM Systems Journal, 2002, 41(1): 140 – 161.

[56] M Hennessy, X Liu. A modal logic for message passing processes[J]. Acta Informatica, 1995, 32(4): 375 – 393.

[57] J Rathke, M Hennessy. Local model checking for value-passing processes[A]. Lecture Notes in Computer Science 1281-3th International Symposium on Theoretical Aspects of Computer Science (TACS 97)[C]. Berlin: Springer-Verlag, 1997. 250 – 266.

[58] Huimin Lin. Model checking value-passing processes[A]. 8th Asian Pacific Software Engineering Conference (APSEC 01)[C]. Los Alamitos: IEEE Computer Society, 2001. 3 – 12.

[59] M Dam. Model checking mobile processes[J]. Information and Computation, 1996, 129: 35 – 51.

[60] Luis Caires, Luca Cardeli. A spatial logic for concurrency (Part I)[A]. Lecture Notes in Computer Science 2215 – 4th International Symposium on Theoretical Aspects of Computer Science (TACS 01)[C]. Berlin: Springer-Verlag, 2001. 1 – 37.

作者简介:

林惠民 男,1947 年 11 月出生于福建福州,1986 年在中国科学院软件所获博士学位,现任中国科学院软件研究所计算机科学重点实验室研究员、主任,研究兴趣包括:并发理论及应用、模型检测、进程代数、移动计算的形式模型与分析、形式化方法.

张文辉 男,1963 年 6 月出生于福建福安,1988 年在挪威奥斯陆大学数学自然科学学院获博士学位,现任中国科学院软件研究所计算机科学重点实验室研究员,研究兴趣包括:程序正确性、模型检测、逻辑推理、形式化方法.