ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.1, January 2008, pp.27−38
DOI: 10.3724/SP.J.1001.2008.00027

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

谓词抽象技术研究*

屈婉霞+, 李 暾, 郭 阳, 杨晓东

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Advances in Predicate Abstraction

QU Wan-Xia+, LI Tun, GUO Yang, YANG Xiao-Dong

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573679, E-mail: wxqu2002@yahoo.com.cn

Qu WX, Li T, Guo Y, Yang XD. Advances in predicate abstraction. *Journal of Software*, 2008,19(1):27−38.
http://www.jos.org.cn/1000-9825/19/27.htm

Abstract: With the growing increase in software/hardware system scale and function, the further development and application of model checking has been greatly limited by state space explosion, which becomes the bottleneck of verifying large industrial designs. Predicate abstraction, as one of the most effective ways to address state explosion, has been fueled over the recent years. This paper presents a survey of the latest developments in predicate abstraction. A basic algorithm for predicate abstraction is introduced first, followed by comparison among several solvers. Emphases are put on counterexample-guided abstraction refinement and interpolation-based abstraction refinement, including the principles and improvements. The qualities of the new predicate generation methods are also analyzed. Finally, the major challenges in making this technology more pervasive in industrial design verification domain are noted.

Key words: model checking; predicate abstraction; abstraction refinement; counterexample; interpolation

摘 要: 随着软、硬件系统规模和功能的不断扩充,状态空间爆炸问题严重影响了模型检验的进一步发展与应用,成为验证大规模系统的瓶颈.谓词抽象是解决状态空间爆炸的最有效方法之一,近年来得到迅速发展.介绍了谓词抽象的基本算法并比较了不同的求解支持工具;重点分析了反例指导的抽象求精和基于插值的抽象求精原理;分析了产生新谓词的各种方法的优、缺点;最后指出了谓词抽象技术进一步发展所面临的挑战和发展方向.

关键词: 模型检验;谓词抽象;抽象求精;反例;插值
中图法分类号: TP301 **文献标识码**: A

随着软硬件系统规模和功能的不断扩充,系统的正确性验证日益困难.目前,功能验证主要采用模拟和形式化验证方法.模拟方法是目前的主流验证方法,但由于它极大地依赖于模拟激励,因此很难保证验证的完备性.形式化验证方法是对模拟方法的重要补充,它使用严格的数学推理来证明系统是否满足全部或部分规范(又称为系统属性),大体上可分为等价性检验、模型检验和定理证明 3 类.

模型检验通过遍历状态空间自动地判断一个有限状态系统是否满足某些属性,在最坏情况下,状态空间的

* Supported by the National Natural Science Foundation of China under Grant Nos.60403048, 60573173 (国家自然科学基金)
Received 2006-09-12; Accepted 2007-05-10

大小与系统规模成指数关系,状态空间爆炸问题严重影响了模型检验的进一步发展与应用,成为验证大规模系统的瓶颈.抽象是解决或者缓解状态空间爆炸最有效的方法之一,通过删除系统(又称为原始模型)中与验证无关的信息得到一个简化模型(又称为抽象模型),属性验证工作在抽象模型中进行,由于状态空间相对较小,验证效率大为提高,可以处理规模较大的设计.

在软件验证领域,最成功的抽象技术是谓词抽象,而硬件领域使用最多的局部抽象是谓词抽象的一个特例.谓词抽象是一类特殊的属性保持的抽象方法,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成一个易于处理的、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,而在抽象模型中不能证明真伪的属性,在原始模型中可能成立、也可能不成立.

本文的主要内容包括谓词抽象基本算法、谓词抽象求解工具、抽象模型的精确度问题和新谓词的产生方法等.

1 谓词抽象基本算法

在谓词抽象中,通常用有限状态机描述一个系统,初始状态集合和状态迁移关系集合唯一地确定一个系统.设原始模型 M 的状态集合为 C,初始状态集合为 I_C,状态迁移关系集合为 R_C.谓词在状态集合 C 上定义了一个等价关系,这个等价关系对状态集合 C 进行划分,每一个状态属于一个等价类,两个状态等价当且仅当它们在同一个等价类中.一个等价类用一个抽象状态来表示.

给定一组谓词 $\varphi_1,\varphi_2,\ldots,\varphi_n$,定义一组布尔变量 $B_1,B_2,\ldots,B_n$(B_i 与谓词 φ_i 对应),原始状态用该状态满足的谓词公式 φ 表示,抽象模型状态集合 A 是变量 $B_1,B_2,\ldots,B_n$ 上的一组正交布尔表达式.抽象状态用表达式 $\exp^A(B_1,\ldots,B_n)$ 表示,抽象模型的初始状态集合用 I_A 表示.原始状态和抽象状态之间的关系由抽象算子 α 和精化算子 γ 确定.

定义 1. 抽象算子 $\alpha:C\to A$ 把原始状态 φ 映射到抽象状态:

$$\alpha(\varphi)=\wedge\{\exp^A(B_1,...,B_n)\mid \varphi\Rightarrow \exp^A[\bar{\varphi}/\bar{B}]\} \tag{1}$$

其中,$\bar{\varphi}$ 表示由谓词 $\varphi_1,\varphi_2,\ldots,\varphi_n$ 构成的向量,$\bar{B}$ 表示由布尔变量 $B_1,B_2,\ldots,B_n$ 构成的向量,$[\bar{\varphi}/\bar{B}]$ 表示一个置换,它将向量 $\bar{B}$ 中的每一个布尔分量 B_i 用对应的谓词分量 φ_i 替换.

对于任意一个原始状态 φ,$\alpha(\varphi)$ 是满足 $\varphi\Rightarrow\exp^A[\bar{\varphi}/\bar{B}]$ 的所有公式 $\exp^A(B_1,\ldots,B_n)$ 的合取范式.因此,抽象模型的初始状态集合 $I_A=\alpha(I_C)$.

定义 2. 精化算子 $\gamma:A\to C$ 是抽象算子的逆运算,把抽象状态映射到原始状态:

$$\gamma(\exp^A(B_1,...,B_n))=\exp^A[\bar{\varphi}/\bar{B}] \tag{2}$$

对于任意一个抽象状态 $\exp^A(B_1,\ldots,B_n)$,把其中的 B_i 用相应的谓词 φ_i 替换即可得到对应于该抽象状态的所有原始状态 φ.

显然,一个原始状态只对应一个抽象状态,而一个抽象状态可能对应一组原始状态.

实际上,由于难以求出所有满足 $\varphi\Rightarrow\exp^A[\bar{\varphi}/\bar{B}]$ 的公式,需要对抽象算子进行化简.Graf 和 Saidi[1] 用 α 的一个上界逼近 $\alpha'(\varphi)=\wedge\{B_i\mid\varphi\Rightarrow\varphi_i,1\le i\le n\}$ 代替 α,并给出了由原始状态求抽象状态的方法:一个抽象状态是 n 个布尔变量的某个赋值,用原子公式 false 或者合取范式公式 $c_1\wedge\ldots\wedge c_n$ 表示,其中,$c_i(1\le i\le n)$ 取值为 B_i,$\neg B_i$ 或 true.若当前原始状态满足谓词 φ_i,则 c_i 取值为 B_i;若当前原始状态满足谓词 $\neg\varphi_i$,则 c_i 取值为 $\neg B_i$;若当前原始状态既不满足谓词 φ_i 也不满足谓词 $\neg\varphi_i$,则 c_i 取值为 true.

抽象迁移关系集合 R_A 与原始迁移关系集合 R_C 的关系如下:设原始状态 φ 的抽象状态为 $\exp^A(B_1,\ldots,B_n)$,从 φ 出发的原始状态迁移关系为 $\tau_j,0\le j\le|R_\varphi|,R_\varphi\subseteq R_C$ 是 R_C 中所有从 φ 出发的状态迁移关系,则从 $\exp^A(B_1,\ldots,B_n)$ 出发的、与 τ_j 对应的抽象状态迁移关系 τ_j^A 由下列表达式确定:

$$\tau_j^A(\exp^A(B_1,...,B_n))=\begin{cases}\text{false}, & \text{if } \exp^A[\bar{\varphi}/\bar{B}]\Rightarrow\neg g_j\\ \wedge_{i=1}^{n}\begin{cases}B_i, & \text{if } post[\tau_j](\exp^A[\bar{\varphi}/\bar{B}])\Rightarrow\varphi_i\\ \neg B_i, & \text{if } post[\tau_j](\exp^A[\bar{\varphi}/\bar{B}])\Rightarrow\neg\varphi_i\\ \text{true}, & \text{otherwise}\end{cases}\end{cases} \tag{3}$$

其中,g_j 是 τ_j 的发生条件,当前状态满足 g_j 时,当前状态的状态迁移才允许发生.因而,若 $\exp^A[\bar{\varphi}/\bar{B}]\Rightarrow\neg g_j$ 可满足时,$\tau_j^A(\exp^A(B_1,...,B_n))=\text{false}$ 表示抽象状态 $\exp^A(B_1,\ldots,B_n)$对于抽象迁移关系 τ_j^A 没有后继状态.

$post[\tau_j](\exp^A[\bar{\varphi}/\bar{B}])$ 表示在迁移关系 τ_j 下当前状态 $\exp^A[\bar{\varphi}/\bar{B}]$ 的后继状态,若 $post[\tau_j](\exp^A[\bar{\varphi}/\bar{B}])\Rightarrow\varphi_i$ 可满足,则 c_i 取值为 B_i;若 $post[\tau_j](\exp^A[\bar{\varphi}/\bar{B}])\Rightarrow\neg\varphi_i$ 可满足,则 c_i 取值为$\neg B_i$,否则 c_i 取值为 true.

至此,对初始状态集合为 I_C、状态迁移关系集合为 R_C 的原始模型,利用抽象算子α'和 τ_j^A 运算可以求出初始状态集合为 I_A、状态迁移关系集合为 R_A 的抽象模型.文献[1]给出了利用定理证明器 PVS 构造完整的抽象模型的算法.

算法 1. 构造完整抽象模型的算法.

输入:初始状态集合为 I_C、状态迁移关系集合为 R_C 的原始模型 M,谓词集合Φ;

输出:原始模型 M 相对于Φ的抽象模型 $M_A(\Phi)$.

步骤 1. 初始化.

根据谓词集合Φ求抽象算子α';

抽象模型初始状态集合 $A_{init}=\alpha'(I_C)$;

全局变量 $A_{reach}=A_{init}$,保存当前求出的所有可达抽象状态;

全局变量 $A_{tou}=NULL$,保存当前求出的所有抽象迁移关系;

全局变量 $To\_explore=A_{init}$,保存当前未遍历的抽象状态.

步骤 2. 如果 *To_explore* 为空,则转步骤 3,否则执行下述操作:

步骤 2.1. 从 *To_explore* 中选择一个抽象状态 m;

步骤 2.2. 求 m 的状态迁移关系 $\tau_j^A(m)$;

步骤 2.3. 利用 $\tau_j^A(m)$ 求 m 的所有后继 *succ*,*succ* 实质上是 m 的所有后继的上界逼近;

步骤 2.4. 修改全局变量 A_{reach},A_{tou} 和 *To_explore*,转步骤 2.

步骤 3. 构造出一个初始状态集合为 A_{init}、状态迁移关系集合为 A_{tou} 的抽象模型 $M_A(\Phi)$,其可达状态集合为 A_{reach}.

例 1:原始模型的状态集合 $C=\{s_i|s_i=i,1\le i\le 15\}$,状态迁移关系集合为 $R_C=\{s_1\to s_2,s_1\to s_7,s_5\to s_6,s_5\to s_{12},s_6\to s_{12},s_8\to s_9,s_{10}\to s_4\}$.若谓词集合为 $\Phi=\{\varphi_1,\varphi_2\}$,$\varphi_1=s_i>10$,$\varphi_2=s_i<5$,设 B_1,B_2 是分别与 φ_1,φ_2 对应的布尔变量.按照上述方法有下列等式成立:

$$\alpha'(s_1)=\alpha'(s_2)=\alpha'(s_3)=\alpha'(s_4)=\alpha'(\neg\varphi_1\wedge\varphi_2)=\neg B_1\wedge B_2,$$
$$\alpha'(s_5)=\alpha'(s_6)=\alpha'(s_7)=\alpha'(s_8)=\alpha'(s_9)=\alpha'(s_{10})=\alpha'(\neg\varphi_1\wedge\neg\varphi_2)=\neg B_1\wedge\neg B_2,$$
$$\alpha'(s_{11})=\alpha'(s_{12})=\alpha'(s_{13})=\alpha'(s_{14})=\alpha'(s_{15})=\alpha'(\varphi_1\wedge\neg\varphi_2)=B_1\wedge\neg B_2.$$

那么,抽象状态集合 $A=\{a_1{:}\neg B_1\wedge B_2,a_2{:}\neg B_1\wedge\neg B_2,a_3{:}B_1\wedge\neg B_2\}$,抽象状态迁移关系集合 $R_A=\{a_1\to a_2,a_2\to a_1,a_1\to a_1,a_2\to a_2,a_2\to a_3\}$.图 1 显示了原始模型与利用谓词抽象得到的抽象模型之间的关系.

任何用时态逻辑公式表示的属性,都可以用模型检验工具在此抽象状态图中进行验证.如果由于丢失过多信息而使抽象模型过于粗糙,导致无法验证某些属性,则需要增加谓词,对原始状态空间进行更细致的划分.抽象模型的状态空间大小与谓词数目成指数级的关系,精化操作必定增加对空间的需求,最坏情况下,抽象模型的状态数为 2^n,可能比原始模型需要的空间还要大.此外,每求一个抽象迁移关系需要调用$(1+2n)$次定理证明器进行有效性判断.时间和空间开销成为影响算法效率的两个重要因素,使得该算法很难用于验证更大规模的系统.

研究人员采用各种手段对该算法进行了改进,期望以尽可能快的速度获得一个能够证明属性的尽可能小的抽象模型.这些改进大致可以分为如下 3 类:

(1) 采用不同的工具求解抽象状态迁移关系,减少因多次调用定理证明器带来的时间开销.

(2) 抽象模型的精化.从技术和方法学等多方面探讨如何生成更精确的抽象模型.

(3) 新谓词的产生及谓词集的优化.采用各种优化技术对谓词集进行优化,以获得尽可能小而精确的抽象模型.

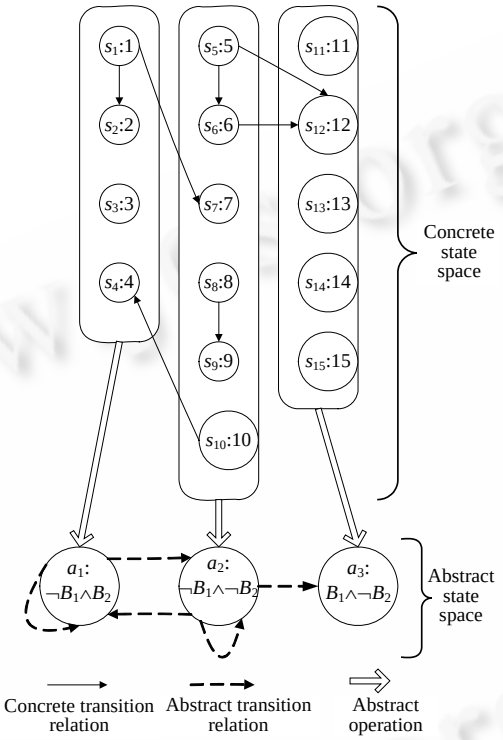


Fig.1 Concrete model and its abstract model in Example 1

图 1 例 1 中原始模型与相应的抽象模型

2 谓词抽象求解支持工具

定理证明器是最早用于谓词抽象的求解器,例如 simplify[2],zapato[3],它用形式逻辑的命题、谓词、定理、推理规则等对系统进行描述,待验证属性被表示成定理的形式.定理证明的过程,是在验证者的引导下,以形式逻辑的公理和推理规则为基础,不断地对公理、已证明的定理施加推理规则,产生新的定理,直至推导出所需要的定理.推导过程需要反映系统假设条件的定义和引理.典型的定理证明系统,如 Cambridge 大学的高阶逻辑定理证明器 HOL(high order logic),Stanford 研究所的原型验证系统 PVS(prototype verification system)等,都内置各种推理规则、推理对策和元对策等,由验证者决定推理方向.用定理证明器计算抽象模型的缺点是需要具有良好数学基础和经验丰富的验证者来引导推理进程.此外,定理证明器只能处理自然数,不能处理硬件系统的实现中使用较多的位向量,而且绝大多数定理证明器仅支持少数几种运算,不支持位运算.

SAT 求解器也可用于计算抽象模型.求抽象状态迁移关系的问题可以转换成一个布尔可满足问题 SAT (Boolean satisfiability problem),其基本思想是,为系统实现代码中每一个基本块中的每一个谓词设置两个符号变量,分别表示基本块执行前和执行后的状态,用谓词作为自由变量产生求解公式,使用位向量逻辑将求解公式转换成合取范式 CNF(conjunctive normal form).一个可满足的赋值对应一个抽象迁移关系,所有可满足赋值构

成抽象模型的迁移关系集合[4].大多数 SAT 求解器基于 DPLL(Davis-Putnam-Logemann-Loveland)后向搜索策略、通过对合取范式 CNF 的消解实现求解[5].DPLL 搜索过程本身对空间需求很大,SAT 问题又是一个 NP 完全问题,最坏情况下的复杂度是 $O(n^2)$,其中,n 是公式中不同变量的个数.因此,SAT 求解器通常集成启发式局部搜索策略或者布尔约束传播策略[6]提高求解效率.目前,应用比较广泛的 SAT 求解器有 GRASP[7],SATO[8]和 Chaff[9].近年来,SAT 求解器的推理能力显著增强,极大地改善了谓词抽象的时间和空间效率,促进了对谓词抽象技术更深入的研究和应用[10−13].但是,对于字级约束,SAT 求解器在求解时需要将其转换成位级约束,这样可能丢失大多数字级信息,造成抽象模型过于粗糙而无法验证给定属性.

约束逻辑编程 CLP(constraint logic programming)集成了逻辑编程强大的表达和约束求解能力,能够在同一个框架中对位、位向量和有界整数进行建模,同时支持多种算术和逻辑操作,解决了 SAT 求解器不能处理字级约束的问题,已用于构造抽象模型[14].由于直接对字级约束进行求解,计算速度大幅提高.实验证明,构造抽象模型的时间与谓词数目呈线性关系.唯一的不足是,当求解公式不可满足时,CLP 求解器不能给出反例(反例是一个从某个初始状态出发不满足给定属性的状态序列).

很多情况下,计算抽象模型的公式不仅包含简单的命题公式,还包含某些组合理论 t(如整数理论、矩阵理论和差分逻辑等理论)上的原子,确定这类公式的可满足性问题称作可满足性模理论 SMT(satisfiablity modulo theories)问题,解决 SMT 问题的系统称为 SMT 求解器.SMT 求解器实质上是一种高效的 SAT 求解器,它首先把求解公式 f 中的每一个原子看作命题原子并调用普通的 SAT 求解器进行求解,如果 f 是命题不可满足的,也一定是理论 t-不可满足的;否则,调用 t 求解器对返回的 f 的模型 m(即 f 的可满足解)进行求解,如果 m 可满足,则 f 是 t-可满足的,否则构造排除 m 的定理并加入到 f 中,重新开始对 f 的命题可满足性进行求解.当这个过程结束时,或者找到 f 的一个 t 模型,或者证明 f 是 t-不可满足的.由于 SMT 求解器能够处理组合理论中的原子,当原始模型中存在无法用函数表达的逻辑时,采用 SMT 是很方便的.此外,一旦实现某个理论 t 上的 SMT 求解器,只要改变 t 就可以得到其他理论上的 SMT 求解器.越来越多的研究人员开始采用 SMT 求解器进行谓词抽象[15−18].

3 抽象模型精化

若抽象模型过于粗糙,则可能无法证明给定属性;而若抽象模型过于精细,则将增加对空间的需求.如何控制抽象的精确程度,构造一个尽可能紧凑又能够验证给定属性的抽象模型,在效率和功能之间求得平衡,是谓词抽象算法面临的最大挑战之一.若在当前抽象模型中无法验证给定属性,则需要重新定义抽象算子α,对原始模型进行更细致的划分,验证工作在新的抽象模型中进行,这个过程称为抽象精化(abstraction refinement).抽象精化过程不断迭代,直到证明给定属性或者给出反例.

3.1 反例指导的抽象精化

如果反例在原始模型中存在对应的状态序列(这种反例称为真实反例),则证明给定属性不可满足,否则,反例是虚假的,不能证明属性的真伪.通常,初始抽象模型仅保留与给定属性相关的系统信息,虚假反例提供了一个通过分析已被删除、与属性无关的系统信息产生新谓词的手段,基于反例对抽象模型进行精化是最自然、最直接的方法.

结合反例分析,Clarke[11]于 2000 年提出完全自动化的、反例指导的抽象精化方法 CEGAR(counterexample-guided abstraction refinement),成为自动分析与验证大规模软、硬件系统的一种主要方法.给定原始模型 M 和待验证属性 p(p 是一个时态逻辑公式),CEGAR 集成各种算法求解一个抽象模型 M_A,M_A 必须包含足够的信息,能够验证 $M\Rightarrow p$ 是否成立;同时,M_A 的信息又必须尽可能地少,保证验证过程能够高效执行.CEGAR 过程包括 5 个步骤.

算法 2. CEGAR 算法.

输入:原始模型 M,待验证属性 p;

输出:属性 p 在原始模型 M 中成立时,返回 TRUE;

属性 p 在原始模型 M 中不成立时,返回 FALSE.

步骤 1. 根据原始模型 M 和待验证属性 p,产生最初的抽象模型 M_A;

步骤 2. 检验 p 在 M_A 中是否成立,即 $M_A \Rightarrow p$ 是否成立.如果 $M_A \Rightarrow p$,则 $M \Rightarrow p$,返回 TRUE;否则,转步骤 3;

步骤 3. 分析 $M_A \not\Rightarrow p$ 的原因,在 M_A 中找出 p 的一个反例 C_x;

步骤 4. 在原始模型 M 中检验反例 C_x 的真实性.如果 C_x 是真实反例,即 $M \not\Rightarrow p$,则证明 p 在 M 中不可满足,返回 FALSE;否则,转步骤 5;

步骤 5. 对 M_A 求精,回到步骤 2 继续.

3.1.1 谓词集合保持不变的模型精化

状态迁移关系的上界逼近包含最精确迁移关系中的所有迁移关系,足以证明给定属性,求状态迁移关系的上界逼近的工作量通常比求最精确迁移关系要少,所以在具体实现中,多用抽象状态迁移关系的上界逼近替代最精确抽象状态迁移关系.

抽象状态迁移关系的上界逼近是产生虚假反例的一个主要原因.假设属性 p 在原始模型中成立并且谓词能够证明 p,那么,原始模型中满足 p 的状态集合 $C(p)$是满足 p 的可达状态集合 $C_{reach}(p)$的超集.由于上界逼近的不精确性,抽象模型中可能包含属性 p 的一个反例,它从 $C_{reach}(p)$中的一个初始状态出发,由于某个中间状态不在 $C_{reach}(p)$内,最终导致某个后继状态不在 $C(p)$内,属性 p 的正确性因此得不到证明.反例中至少包含一个从 $C_{reach}(p)$内的某个状态到 $C_{reach}(p)$外的某个状态的迁移关系,这是由上界逼近造成的一个虚假迁移关系.对抽象状态迁移关系进行精化,可以删除这样的虚假迁移关系,此时,划分原始模型状态空间的谓词集合保持不变.

Das[19]基于反例分析,通过对抽象状态迁移关系的连续逼近,实现了抽象模型的精化.抽象状态迁移关系的初值是最精确抽象状态迁移关系 R_A 的一个上界逼近 R_0,如果属性 p 在这个抽象模型中成立,则验证结束;否则,检查反例中每一个状态迁移关系是否包含在 R_A 中.如果反例中的所有状态迁移关系都在 R_A 中,说明当前抽象模型过于粗糙;如果存在一个状态迁移关系 $R_0(s_j,s_{j+1})$不在 R_A 中,即 $R_A(s_j,s_{j+1})$真值为假,则说明上界逼近 R_0 包含虚假状态迁移关系,增加合适的约束条件,使删除 R_0 后得到的抽象状态迁移关系 R_1 仍然是 R_A 的一个上界逼近,属性验证工作继续在由 R_1 表示的抽象模型中进行.这种方法不改变对原始状态空间的划分,只是不断地删除上界逼近造成的虚假状态迁移关系.当抽象模型中不存在虚假状态迁移关系时,仍有可能无法证明属性,此时,需要增加谓词重新对原始状态空间进行划分.

3.1.2 增加新谓词的模型精化

状态的错误合并是虚假反例存在的另一个主要原因.假设 s_a,s_b,s_c,s_d 是原始模型中的 4 个状态,其中,s_a 是可达状态,s_a 到 s_b 存在一个状态迁移,s_c 到 s_d 存在一个状态迁移,而 s_b 和 s_c 之间、s_a 和 s_c 之间、s_b 和 s_d 之间不存在任何状态迁移.如果抽象运算把 s_b 和 s_c 映射到同一个抽象状态 s_{bc},那么,抽象模型中存在一条从 s_a 经过 s_{bc} 到 s_d 的路径,即 s_a 到 s_d 是可达的.显然,这条路径是虚假的,因为在原始模型中,s_a 到 s_d 不可达.因此,事先选择合适的抽象函数、对原始状态进行相对细致的分类,是避免产生虚假反例的关键.另一方面,从虚假反例中找出被错误合并的状态,通过增加新谓词将其分离,是删除虚假反例的有效方法.

CMU(Carnegie Mellon University)在基于反例的抽象精化方面进行了大量卓有成效的研究.根据相关性对布尔变量进行分组构成变量群(variable clusters)[11],抽象函数是变量群的域上的一个满射,这个抽象函数保持着布尔变量之间的逻辑关系.一旦出现虚假反例,则首先识别出反例的最短前缀(该前缀在原始模型中没有对应的真实路径),然后把前缀中的最后一个状态分解得到一个精化的抽象函数,重新应用新的抽象函数得到一个不包含当前虚假反例的新的抽象模型.不同的状态分解方法对应不同的抽象函数,能够删除反例的“最粗糙”的分解方法使抽象模型最小,因而是最优解.

基于记分和冲突相关图的分析[13]识别被错误合并的状态,确定需要加入到抽象模型的不可见变量.分值的大小决定了不可见变量的重要程度,也决定了哪些不可见变量在精化过程中能够恢复到可见状态.影响分值的因素包括:变量被回溯的次数、变量出现在冲突语句中的次数.冲突相关图记录了不可满足的原因,出现在冲突相关图中的变量能够有效阻止虚假路径的再次出现.

基于投影的 SAT 枚举算法[10]同样利用冲突图在抽象模型的精化过程中寻找新谓词,该算法进一步对新谓

词所依赖的原始变量进行最小化处理,产生的新谓词结构更加紧凑,求解效率更高.

精细程度不同的抽象模型对应不同的抽象函数,模型精化的过程就是不断寻找能够分离被错误合并的状态的抽象函数的过程,这是一个学习的过程[12]:从抽象函数集合中找出抽象函数 h',它是前一次迭代使用的抽象函数 h 的精化,能够把导致虚假反例的状态重新映射到不同的抽象状态,即 $h(s_b)=h(s_c),h'(s_b)\neq h'(s_c)$.这种方法能够更好地控制抽象模型的规模,保证找到证明属性的最小抽象模型.

对反例指导的抽象精化的研究和优化工作还有很多.在寻找新谓词之前先对虚假反例进行最小化处理[19],分别从反例的开始和结尾删除那些不改变反例性质的状态,然后利用定理证明器 CVC(cooperating validity checker)求新谓词.通过协同分析从一组反例中抽取信息[20],运用可控制性分析和协同分析求精抽象模型.实验表明,算法在运行时间、抽象效率和可证明的属性数 3 个方面优于其他对比算法.利用 Petri 网理论识别并分离因合并导致虚假反例的状态的方法[21],可用于验证拓扑结构动态变化的系统.

3.2 基于插值的抽象精化

给定原始系统 M 和属性 p,是否存在长度(状态序列中的状态数)不超过 k 的 p 反例?这类问题是寻找给定属性的不可满足解,即反驳给定属性.此问题可用命题逻辑形式化描述如下:

$$f^k = I \wedge \left(\bigwedge_{i=0}^{k-1} R_{i,i+1}\right) \wedge \left(\bigvee_{i=0}^{k} \neg p_i\right) \tag{4}$$

其中,I 是 M 的初始状态集合的特征函数,$R_{i,i+1}$ 是状态迁移关系在时间步 i 的特征函数,p_i 表示属性 p 在时间步 i 的真值.

那么,$I \wedge \left(\bigwedge_{i=0}^{k-1} R_{i,i+1}\right)$ 表示 M 中所有从某个初始状态开始、长度不超过 k 的路径.f^k 可满足的当且仅当对于某个 $i(i\leq k)$,M 中存在一个可达状态,该可达状态位于时间步 i 且不满足 p,即 $\neg p_i$ 是可满足的.

基于 SAT 的有界模型检验 BMC(bounded model checking)[22]能够解决此问题,这得益于 SAT 求解器研究领域的发展.BMC 的主要原理是把抽象状态迁移关系在初始状态上迭代 k 次,得到一个有界的状态空间,然后利用 SAT 求解器在此空间上寻找所要验证属性 p 的反例.如果在此空间中没有发现反例,则增加 k 值继续搜索,直到找到 p 的一个反例,或者达到预先定义的 k 值,或者因 k 值过大导致问题不可解.如果找到一个反例,则可用下面的插值定理对抽象模型进行精化(interpolation-based abstraction refinement)[23].

定理 1(Craig 插值定理[24]). 给定一对逻辑公式 A 和 B,如果 $A\wedge B$ 是不可满足的,则 A 和 B 的插值 A' 也是一个逻辑公式,并且满足下列条件:

(1) $A\Rightarrow A'$;

(2) $A'\wedge B$ 是不可满足的;

(3) A' 只包含 A 和 B 中的公共变量.

例 2:设 $A=p\wedge q,B=\neg q\wedge r$.显然,$A\wedge B=(p\wedge q)\wedge(\neg q\wedge r)=$false,那么,$A'=q$.

因为 $A\wedge B$ 不可满足,SAT 求解器的证明序列中包含一个集合 A',它是 $A\wedge B$ 中的全体子句集合的一个不可满足子集,指出了 $A\wedge B$ 不可满足的原因,A' 称为不可满足核(unsatisfiable core).通常,导致 $A\wedge B$ 不可满足的原因有很多,因此插值 A' 并不唯一,但是,A 和 B 的最弱插值和最强插值存在且唯一.令 WI 是 A 和 B 的最弱插值,SI 是 A 和 B 的最强插值,若 A' 是 A 和 B 的插值,则 $A'\Rightarrow WI,SI\Rightarrow A'$.

Craig 插值定理不仅可以求一对公式的插值,还可以求一对公式集合的插值.公式集合 $\{A_1,A_2,\ldots,A_n\}$ 和 $\{B_1,B_2,\ldots,B_m\}$ 的插值是 $(A_1\wedge A_2\wedge\ldots\wedge A_n)$ 和 $(B_1\wedge B_2\wedge\ldots\wedge B_m)$ 的插值.

Craig 插值定理可用于指导抽象模型的精化.用公式 A 表示初始抽象模型,公式 B 表示待验证属性,如果 A 中存在长度不超过 k 的 B 的反例,即 $A\wedge\neg B$ 是 k-可满足的,那么,$A\wedge B$ 是 k-不可满足的.根据 Craig 插值定理,A 和 B 的插值 A' 包含 $A\wedge B$ 是 k-不可满足的原因,从 A 中删除 A' 中包含的信息,得到一个新的抽象模型,它包含的信息量比 A 中的信息量要少,因而减少了对存储空间的需求.如果 A 中不存在长度不超过 k 的 B 的反例,则需要增加 k 的值继续验证.

在基于反例的抽象精化方法中,求精过程很简单,但是每次求精操作只能删除一个反例;而在基于插值的抽象求精方法中,一次求精操作可以删除给定长度内的所有反例,但是求精过程很耗时.基于反例的方法和基于插值的方法本质上是同一种求精方法的两个特例[25].大量的实际验证问题表明,求精过程的开销和求精过程的迭代次数关系密切,基于插值的方法比基于反例的方法效率更高.

BMC 的有界性导致了基于插值的抽象求精方法的不完备性,抽象状态迁移关系的迭代次数 k 是人为确定的(通常从 1 开始逐渐增加),如果不存在长度不超过 k 的 p 反例,并不能证明整个模型空间上都不存在 p 反例.k 值越大,抽象模型迭代的次数越多,占用的空间就越大,问题求解的时间就越长,有可能在给定的时间内无法求解问题.完备性问题成为 BMC 研究的热点之一.

最近的研究从多方面对 BMC 加以改进.利用剪枝技术生成更紧凑的合取范式是最常用的方法[26],其目的是简化 SAT 问题,使其便于求解.SAT 求解器中,决策变量的顺序是影响 BMC 性能的重要因素,冲突驱动的启发式变量排序策略比完全静态的变量排序策略更能提高求解效率.约束共享[27]和约束复制[28]分别通过静态和动态学习技术不断增加冲突子句加速求解过程.给定迭代策略 t 和属性 p,任何有限状态系统 M 都存在一个整数 CT,使得最多在 CT 个迭代周期内能够证明 $M\Rightarrow p$ 成立,CT 称为 M 关于 t 和 p 的完备性阈值[22].安全属性公式的 CT 是从初始状态开始到所有可达状态的最短路径中的最大值,称为可达直径 rd(reachability diameter).如何计算其他属性的 CT 的研究刚刚开始.研究人员通过分析网表结构,找出最常用单元的强连通组件 SCC(strongly connected component),并在每个 SCC 上递归定义系统的 rd(大多数网表的 rd 都在 20 左右)[29],已设计出一个用于验证硬件设计的完备的 BMC 算法.BMC 完备性研究虽然取得了一定成就,但是,高效地计算现实世界的时序电路的 rd 仍需深入.

4 新谓词的产生及谓词集合的优化

谓词在原始状态空间上引入一个等价关系,一个等价类用一个抽象状态表示.谓词的数目决定着等价类的数目(即抽象状态空间的规模),较多的谓词能够反映更多的原始模型的行为,抽象模型的精细程度因此更高.目前,新谓词的产生方法大致有以下 4 类:

(1) 从系统实现的高层描述中提取

系统高层描述中包含许多与验证有关的重要信息,例如,分支条件成立与否决定着系统运行的方向,这些信息往往在转换成模型检验工具要求的描述的过程中丢失.Wang 研究了从硬件系统的高层描述中提取新谓词的方法[10]:首先将 RTL(register transfer level)级的 Verilog 描述转换成等价的门级电路描述,其中的分支条件用不同的信号表示.经过重命名的门级电路进一步被转换成状态迁移系统,信号名的唯一性决定了分支的唯一性,与验证有关的分支条件可以很容易地通过信号名识别并作为新谓词.

只要能够对布尔谓词重命名,就可以将此方法应用于其他高层描述语言.但是,高层描述中的一个分支通常对应着转换后的系统描述中的多个分支,并不是每一个这样的分支都适合作为新谓词,在出现虚假反例的情况下,需要采用各种启发式方法剔除不合适的谓词.

(2) 不完备的启发式方法

大多数启发式方法建立在计算最弱前件的基础上[30−33].最弱前件 $wp(S,Q)$表示一个谓词 W,如果系统 S 运行到满足谓词 P 的状态时一定会终止并且终止状态满足谓词 Q,那么 $W\Rightarrow P$.最弱前件运算是谓词集合变换的关键,给定系统运行终止时的一组状态,通过最弱前件运算得到的一组初始状态与给定终止状态之间存在 1 条或多条路径.

在惰性抽象(lazy abstraction)[34]中,抽象模型用控制流自动机 CFA(control flow automaton)表示,其中,顶点表示程序的控制点,边或者用从源控制点到目的控制点所执行的基本程序块标注,或者用对应该分支条件的谓词标注.属性验证通过遍历 CFA 完成.前向搜索过程生成一棵搜索树,结点的 reachable region 标注表示程序从根结点执行到当前结点时的状态,bad region 标注表示程序从当前结点执行到错误结点时的状态,是当前结点指向子结点的边上所标注的程序块和子结点的最弱前件.如果在搜索树中遇到一个错误状态,则通过后向分析确认

反例的真实性.bad region 与 reachable region 的交集为空的第 1 个结点指出增加哪些谓词可以删除虚假反例,这样的结点称为支点(pivot node).新谓词加入到以支点为根的子树中,然后从支点开始继续前向搜索过程.惰性抽象仅对局部状态空间进行求精,不再遍历已经证明正确的状态空间,避免了大量重复工作.惰性抽象是不完备算法,但是在一定的约束下能够终止.

基于最弱前件的方法是不完备的,即使存在能够证明给定属性的谓词集合,对反例的分析也可能产生一个发散的谓词序列,导致最后无法证明属性,计算出的谓词集合还有可能不是最优解.

(3) 完备的启发式方法

基于插值的方法在寻找相关谓词时效率更高,把插值限制在受限语言 L 中,谓词选择的问题转化为从给定的公式序列中寻找 L 受限插值,出现在插值中的谓词就是新谓词[32].通过逐渐扩充语言 L,保证谓词抽象过程收敛.受限语言的选择是启发式的,必须保证快速收敛.这种方法的不足是不能发现量化谓词.

(4) 量化谓词的产生

多处理机系统、网络协议等允许包含任意多个功能相同的子系统,或者使用无上界的数据类型、堆栈或队列,子系统的数目、堆栈的深度和队列的长度通常随着应用需求的不同而不同,这样的系统称为参数化系统.在验证这种系统时,往往需要证明某个属性对参数的所有合法取值是否都成立,此时必须使用量化谓词.但是,量化谓词本身并不是原始系统的组成部分,无法从系统的高层描述中提取,其产生更多地依赖于各种启发式方法.通过对待验证属性的分析,可以手工产生存在量化的新谓词[19].通过增加 Skolen 约束,可以产生全称量化的新谓词[35],这种谓词是推理矩阵和其他复杂数据结构的关键,但却很难自动发现这种谓词.

谓词集合是否最小直接影响到抽象模型的规模.使用 0-1 约束优化方法能够最大限度地剔除冗余谓词[31]:给定有限谓词集合 Φ,若某个属性能够被证明成立或者不成立,则可以找到 Φ 的最小子集,该子集对此证明是充分的.用带标记的 Kripkpe 结构 LKSs(labeled Kripke structures)表示并行 C 程序,分别用谓词抽象和行为指导的抽象(action-guided abstraction)描述系统的静态和动态行为,使用伪布尔约束是减少抽象模型中谓词数目的另一种途径[36].

5 谓词抽象的应用

谓词抽象自 1997 年提出后备受关注,研究机构不断对其完善和改进,成为对大规模软、硬件系统进行抽象的一种有效方法,已成功地用于软、硬件系统的验证.

SLAM 是微软亚洲研究院 Ball 等人开发的程序自动验证工具包[37],该工具包由 C2BP,BEBOP,NEWTON 等 3 个工具组成.C2BP 是一个自动的谓词抽象工具,给定一组谓词,它将 C 程序抽象为布尔程序.BEBOP 是一个布尔程序的模型检验工具,BEBOP 通过分析过程间的数据流处理过程调用.NEWTON 通过分析程序中的可行路径寻找用于求精布尔程序的其他谓词.SLAM 已用于对微软驱动开发包中的驱动程序进行安全性验证.

BLAST 由加州大学伯克利分校的 Henzinger 等人开发,是基于谓词抽象的反例指导的程序自动验证工具[34].既可以验证顺序 C 程序的安全属性,也可以验证并发 C 程序的安全属性.抽象状态迁移关系用定理证明器 Simplify 求解,完全自动化的惰性抽象提高了程序验证效率.目前,BLAST2.0 源码已经公开.

MAGIC 是 Canegie Mellon 大学开发的基于谓词抽象的反例指导的 C 程序自动验证工具[36].MAGIC 最大的特点是其组合验证功能,它把一个大规模的设计分解成一组小规模设计,然后独立验证小规模设计,因此,MAGIC 的可扩展性非常好,可用于验证现实世界中的工业设计.MAGIC 项目小组当前的工作重点是提高共享内存的并发程序的验证效率.

VCGEAR 是 Canegie Mellon 大学开发的利用字级谓词抽象技术验证硬件系统安全属性的工具[38],其输入是 RTL 级 Verilog 描述和待验证安全属性,运行结果是该安全属性成立或者是一个真实的反例.在抽象精化的迭代过程中,VCGEAR 通过计算 Verilog 语句的最弱前件产生新谓词.

Wang[10]把谓词抽象和局部抽象相结合,通过冲突分析,有效地避免了验证现实世界中大规模硬件系统时抽象模型过于庞大的问题,成功地实现了对包含 6 000 个寄存器和 250 000 门的硬件系统的验证.在验证大规模电

路系统时,Wang[39]通过引入中间变量把大的组合逻辑分割成多个小逻辑,中间变量和状态变量作为抽象求精过程中的“原子”,实现细粒度抽象的同时增强了处理大规模设计的能力.

6 未来挑战

谓词抽象技术的最大优势在于能够处理无限状态系统.过去 10 年里,谓词抽象技术的研究取得了长足发展,在工程设计领域的应用也取得了显著的效益.然而,日益增加的系统复杂性、验证属性的多样性等多种因素,使谓词抽象的进一步研究面临更多的挑战.

初始谓词的选择对基于谓词抽象的验证方法提高效率、减少虚假反例、提高分析正确性具有十分重要的影响.当前,在基于谓词抽象的验证工具中,初始谓词或者由用户提供,或者通过自动抽取程序控制流中的布尔条件式获得.用户提供初始谓词具有随意性,自动抽取程序控制流中的布尔条件式导致对大规模系统进行验证时初始谓词的数量急剧增加,而启发式谓词选择方法不能保证一定能够找到一组谓词来证明给定属性.因而,现有的初始谓词选择和求精方法需要进行改进.

在验证参数化系统时,对量化公式的可满足性判断,是一个不断对量化变量寻找合适赋值的过程,在证明给定属性或者找到一个反例之前,可能需要尝试很多次,而且,这种判断是不确定的.与非量化公式的可满足性判断相比,前者需要更高效的启发式方法提高验证效率.目前,参数化系统验证方面的研究还为数不多,需要进一步加以深入.

当前的谓词抽象技术多用于验证系统的行为属性,如安全性和活性,支持非行为属性(如设计约束、可诊断性、可靠性等)验证的抽象技术研究已悄然兴起,利用模型检验技术设计、实现高可靠的计算机系统是一个崭新的研究方向.

求解器的效率是影响抽象性能的重要因素.提高求解器的效率、增加求解器支持的不同运算和数据类型、运用多种求解支持工具联合求解将是今后的发展趋势.

References:

[1] Susanne G, Hassen S. Construction of abstract state graphs with PVS. In: Grumberg O, ed. Proc. of the 9th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 1997. 72−83.

[2] Detlefs D, Nelson G, Saxe JB. Simplify: A theorem prover for program checking. Journal of the ACM, 2005,52(3):365−473.

[3] Ball T, Cook B, Lahiri SK, Zhang L. Zapato: Automatic theorem proving for predicate abstraction refinement. In: Harald G, ed. Proc. of the 16th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2004. 457−461.

[4] Clarke E, Kroening D, Sharygina N, Yorav K. Predicate abstraction of ANSI-C programs using SAT. Formal Methods in System Design, 2004,25(2):105−127.

[5] Davis M, Putnam H. A computing procedure for quantification theory. Journal of the ACM, 1960,7(3):201−215.

[6] Krzysztof R. Some remarks on Boolean constraint propagation. In: Krzysztof R, ed. Proc. of the Joint ERCIM/Compulog Net Workshop on New Trends in Constraints. London: Springer-Verlag, 1999. 91−107.

[7] Marques-Silva JP, Sakallah KA. GRASP: A search algorithm for propositional satisfiability. IEEE Trans. on Computers, 1999, 48(5):506−521.

[8] Zhang HT. SATO: An efficient propositional prover. In: Niehren J, ed. Proc. of the 14th Int'l Conf. on Automated Deduction. Berlin: Springer-Verlag, 1997. 272−275.

[9] Moskewicz MW, Madigan CF, Zhao Y, Zhang L, Malik S. Chaff: Engineering an efficient SAT solver. In: Rabaey J, ed. Proc. of the 38th Design Automation Conf. Las Vegas: ACM Press, 2001. 530−535.

[10] Wang D. SAT based abstraction refinement for hardware verification [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 2003.

[11] Clarke E, Grumberg O, Jha S, Lu Y, Weith H. Counterexample-Guided abstraction refinement. In: Emerson EA, ed. Proc. of the 12th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2000. 154−169.

[12] Gupta A. Learning abstraction for model checking [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 2006.

[13] Chauhan P, Clarke E, Kukula J, Sapra S, Veith H, Wang D. Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis. In: Aagaard MD, ed. Proc. of the 4th Int'l Conf. on Formal Methods in Computer Aided Design. Berlin: Springer-Verlag, 2002. 33−51.

[14] Li T, Guo Y, Li SK, Liu GJ. Predicate abstraction of RTL verilog descriptions using constraint logic programming. In: Peled DA, ed. Proc. of the 3rd Int'l Symp. on Automated Technology for Verification and Analysis. Berlin: Springer-Verlag, 2005. 174−186.

[15] Seshia SA, Lahiri SK, Bryant RE. A hybrid sat-based decision procedure for separation logic with uninterpreted functions. In: Fix L, ed. Proc. of the 40th Design Automation Conf. Anaheim: ACM Press, 2003. 425−430.

[16] Marco B, Roberto B, Alessandro C, Tommi J, Van Rossum P, Stephan S, Roberto S. An incremental and layered procedure for the satisfiability of linear arithmetic logic. In: Halbwachs N, ed. Proc. of the 11th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2005. 317−333.

[17] Marco B, Roberto B, Alessandro C, Tommi J, Silvio R, Van Rossum P, Roberto S. Efficient satisfiability modulo theories via delayed theory combination. In: Etessami K, ed. Proc. of the 17th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2005. 335−349.

[18] Shuvendu KL, Robert N, Albert O. SMT techniques for fast predicate abstraction. In: Manolios P, ed. Proc. of the 18th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2006. 424−437.

[19] Satyaki D. Predicate abstraction [Ph.D. Thesis]. Stanford University, 2003.

[20] Mang FYC, Ho P-H. Abstraction refinement by controllability and cooperativeness analysis. In: Martin GE, ed. Proc. of the 41st Design Automation Conf. San Diego: ACM Press, 2004. 224−229.

[21] Barbara K, Vitali K. Counterexample-Guided abstraction refinement for the analysis of graph transformation systems. In: Hermanns B, ed. Proc. of the 12th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2006. 197−211.

[22] Biere A, Cimatti A, Clarke EM, Strichman O, Zhu YS. Bounded model checking. Advances in Computers, 2003. 58. http://citeseer.ist.psu.edu/biere03bounded.html

[23] McMillan KL, Amla N. Automatic abstraction without counterexamples. In: Garavel H, ed. Proc. of the 9th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2003. 2−17.

[24] Esparza J, Kiefer S, Schwoon S. Abstraction refinement with craig interpolation and symbolic pushdown systems. In: Hermanns H, Palsberg J, eds. Proc. of the 12th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2006. 489−503.

[25] Amla N, McMillan KL. A hybrid of counterexample-based and proof-based abstraction. In: Alan JH, ed. Proc. of the 5th Int'l Conf. on Formal Methods in Computer-Aided Design. Berlin: Springer-Verlag, 2004. 14−17.

[26] Biere A, Clarke E, Raimi R, Zhu YS. Verifying safety properties of a PowerPC microprocessor using symbolic model checking without BDDs. In: Halbwachs N, ed. Proc. of the 11th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 1999. 60−71.

[27] Shtrichman O. Pruning techniques for the SAT-based bounded model checking problem. In: Margaria T, ed. Proc. of the 11th Advanced Research Working Conf. on Correct Hardware Design and Verification Methods. Berlin: Springer-Verlag, 2001. 58−70.

[28] Shtrichman O. Tuning SAT checkers for bounded model checking. In: Emerson EA, ed. Proc. of the 12th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2000. 480−494.

[29] Baumgartner J, Kuehlmann A, Abraham J. Property checking via structural analysis. In: Barrett CW, ed. Proc. of the 14th Int'l Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2002. 151−165.

[30] Ball T, Rajamani SK. Generating abstract explanations of spurious counterexamples in C programs. Technical Report, MSR-TR-2002-09, Redmond: Microsoft Research, Microsoft Corporation, 2002. http://research.microsoft.com/research/pubs/view.aspx-msr_id=MSR-TR-2002-09

[31] Chaki S, Clarke E, Groce A, Strichman O. Predicate abstraction with minimum predicates. In: Geist D, ed. Proc. of the 12th Advanced Research Working Conf. on Correct Hardware Design and Verification Methods. Berlin: Springer-Verlag, 2003. 19−34.

[32] Lakhnech Y, Bensalem S, Berezin S, Owre S. Incremental verification by abstraction. In: Margaria T, ed. Proc. of the 7th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer-Verlag, 2001. 98−112.

[33] Ball T. Formalizing counterexample-driven refinement with weakest preconditions. Technical Report, MSR-TR-2004-134, Redmond: Microsoft Research, 2004. 1−19.

[34] Henzinger TA, Jhala R, Majumdar R. Lazy abstraction. In: Bakel SV, ed. Proc. of the ACM Symp. on Principles of Programming Languages. Oregon: ACM Press, 2002. 58−70.

[35] Flanagan C, Qadeer S. Predicate abstraction for software verification. In: Bakel SV, ed. Proc. of the ACM Symp. on Principles of Programming Languages. Oregon: ACM Press, 2002. 191−202.

[36] Chaki SJ. A counterexample guided abstraction refinement framework for verifying concurrent C programs [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 2005.

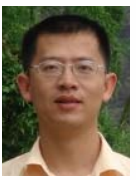
[37] Ball T, Majumdar R, Millstein T, Rajamani SK. Automatic predicate abstraction of c programs. In: Soffa ML, ed. Proc. of the 2001 ACM SIGPLAN Conf. on Programming Language Design and Implementation. Snowbird: ACM Press, 2001.

[38] Clarke E, Jain H, Kroening D. Predicate abstraction and refinement techniques for verifying verilog. Technical Report, CMU-CS-04-139, Pittsburgh: Camegie Mellon University, 2004.

[39] Wang C, Hachtel GD, Somenzi F. Fine-Grain abstraction and sequential don't cares for large scale model checking. In: Greenberg H, ed. Proc. of the IEEE Int'l Conf. on Computer Design: VLSI in Computers & Processors. Los Alamitos: IEEE Computer Society, 2004. 112−118.

屈婉霞(1972－),女,河南南阳人,博士生,助理研究员,CCF 会员,主要研究领域为计算机体系结构,可靠性设计,集成电路功能验证.

李暾(1974－),男,博士,副教授,主要研究领域为集成电路设计验证技术.



郭阳(1971－),男,博士,副研究员,CCF 高级会员,CCF 高级会员,主要研究领域为微处理器设计与验证技术.

杨晓东(1936－),男,教授,博士生导师,主要研究领域为计算机体系结构,性能评价,可靠性理论.