计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160303
54(7):1405-1425,2017

SMT求解技术的发展及最新应用研究综述

王 翀[1,2,3] 吕荫润[1,2,3] 陈 力[1,2,3] 王秀利[4] 王永吉[1,3]

[1](基础软件国家工程研究中心(中国科学院软件研究所) 北京 100190)

[2](中国科学院大学 北京 100049)

[3](计算机科学国家重点实验室(中国科学院软件研究所) 北京 100190)

[4](中央财经大学信息学院 北京 100081)

(wangchong@nfs.iscas.ac.cn)

Survey on Development of Solving Methods and State-of-the-Art Applications of Satisfiability Modulo Theories

Wang Chong[1,2,3], Lü Yinrun[1,2,3], Chen Li[1,2,3], Wang Xiuli[4], and Wang Yongji[1,3]

[1](*National Engineering Research Center for Fundamental Software* (*Institute of Software*, *Chinese Academy of Sciences*), *Beijing* 100190)

[2](*University of Chinese Academy of Sciences*, *Beijing* 100049)

[3](*State Key Laboratory of Computer Science* (*Institute of Software*, *Chinese Academy of Sciences*), *Beijing* 100190)

[4](*School of Information*, *Central University of Finance and Economics*, *Beijing* 100081)

Abstract The satisfiability modulo theories (SMT) problem is a decision problem for the satisfiability of first-order logical formula with respect to combinations of background theories. SMT supports many background theories, so it can describe a lot of practical problems in industrial fields or academic circles. Also, the expression ability and the efficiency of decision algorithms of SMT are both better than those of SAT (satisfiability). With its efficient satisfiability decision algorithms, SMT has been widely used in many fields, in particular in test-case generation, program defect detection, register transfer level (RTL) verification, program analysis and verification, solving linear optimization over arithmetic constraint formula, etc. In this paper, we firstly summarize fundamental problems and decision procedures of SAT. After that, we give a brief overview of SMT, including its fundamental concepts and decision algorithms. Then we detail different types of decision algorithms, including eager and lazy algorithms which have been studied in the past five years. Moreover, some state-of-the-art SMT solvers, including Z3, Yices2 and CVC4 are analyzed and compared based on the results of the SMT's competition. Additionally, we also focus on the introduction for the application of SMT solving in some typical communities. At last, we give a preliminary prospect on the research focus and the research trends of SMT.

Key words satisfiability modulo theories (SMT); SMT solver; decision algorithms of SMT; test-case generation; program defect detection; cloud computing

收稿日期:2016-04-26;**修回日期**:2016-07-14

基金项目:国家自然科学基金项目(61170072,61303057);中国科学院、国家外国专家局创新团队国际合作伙伴计划

This work was supported by the National Natural Science Foundation of China (61170072, 61303057) and the CAS/SAFEA International Partnership Program for Creative Research Teams.

通信作者:王永吉(ywang@itechs.iscas.ac.cn)

摘 要 可满足性模理论(satisfiability modulo theories, SMT)是判定一阶逻辑公式在组合背景理论下的可满足性问题. SMT 的背景理论使其能很好地描述实际领域中的各种问题,结合高效的可满足性判定算法,SMT 在测试用例自动生成、程序缺陷检测、RTL(register transfer level)验证、程序分析与验证、线性逻辑约束公式优化问题求解等一些最新研究领域中有着突出的优势. 首先阐述 SMT 问题的基础 SAT(satisfiability)问题及判定算法;其次对 SMT 问题、判定算法进行了总结,分析了主流的 SMT 求解器,包括 Z3, Yices2, CVC4 等;然后着重介绍了 SMT 求解技术在典型领域中的实际应用,对目前的研究热点进行了阐述;最后对 SMT 未来的发展前景进行了展望,目的是试图推动 SMT 的发展,为此领域的相关人员提供有益的参考.

关键词 可满足性模理论;SMT 求解器;SMT 求解算法;测试用例自动生成;程序缺陷检测;云计算

中图法分类号 TP301

近年来,随着可满足性模理论(satisfiability modulo theories, SMT)的不断进步以及互联网、云计算等新兴技术的不断发展,SMT 被广泛地应用于各个领域,例如云计算与云存储[1-2]、访问控制[3-4]、多核问题[5-6]、程序缺陷检测验证[7]、有界模型检测[8-9]、RTL 验证[10]、优化问题求解[11-12]、静态分析[13-14]、程序验证[15-16]等等. 这些领域中待解决的实际问题都可以建模为约束可满足问题,SMT 在这类问题的表述和求解上有突出优势.

计算机科学和数理逻辑中的 SAT(satisfiability)问题是命题逻辑公式的可满足性问题. 1971 年, Cook[17]证明了 SAT 问题是非确定性多项式完全(non-deterministic polynomial-time complete, NPC)问题,SAT 问题也是第 1 个被证明了的 NPC 问题.

起初,人们注重研究 SAT 在硬件测试、电路验证等领域中的应用,SAT 求解技术的发展对自动电子设计(electronic design automation, EDA)领域中相关问题的研究起到了重要作用. 随后 SAT 被广泛地应用于各个新兴领域,例如静态程序分析[18-20]、测试用例生成[21-23]等. SAT 只面向命题逻辑公式,表达能力有较高的局限性,因此将实际问题转化为 SAT 问题时会丢失很多高层级(high-level)信息. 例如在 RTL 验证中,SAT 使用位向量描述原问题会导致大量逻辑信息的丢失,降低了结果的准确性.

由于上述缺点,人们将 SAT 扩展为 SMT. SMT 面向一阶逻辑公式,在命题逻辑的基础上补充了量词和项,具有更强的表达能力. SMT 融合了多种背景理论,公式中的命题变量可以是理论公式,能够直接描述问题中的高层级信息. 例如 SMT 的数组理论能直接描述数组定义和相关操作. 实际应用中的理论不仅限于单一理论,通常是多个理论的组合. 比如线性逻辑约束公式优化问题求解[11]需要线性实数理论和未解释函数理论等理论的支持;有界模型检[7]测需要数组理论、未解释函数理论和位向量等理论的支持;验证条件分析和定理证明[24]需要数组理论、未解释函数和线性整数算术理论等理论的支持.

SMT 求解技术的具体实现被称为 SMT 求解器(SMT solvers). 最初的 SMT 求解器[25-27]是研究人员为形式化方法设计的判定算法,直到 1990 年,可以处理大规模实际问题的 SMT 求解器[28-32]成为了新的研究热点. 近年来,SMT 求解技术的发展情况如下:

1) 核心算法、数据结构以及现代微处理器的发展使得 SAT 求解器[33]可以处理含有数万变量的公式,以 SAT 求解器为基础的 SMT 求解技术随之提升;

2) SMT 求解技术的广泛应用前景使得各个科研机构积极开发 SMT 求解器;

3) 年度 SAT 竞赛① 和 SMT 竞赛② 推动了 SMT 求解技术的发展;

4) SMT-LIB (satisfiability modulo theories library)标准的出现使得 SMT 求解器开发变得更加容易. 人们研发了很多 SMT 求解器,比较成熟的有 Z3[34], CVC4[35], Yices2[36], MathSAT4[37], Boolector[38]. SMT 求解器也被集成到一些大型工具中,例如微软开发的 PEX[39]工具,其主要功能是研究和分析托管代码. Henzinger 等人[40]开发的 BLAST 工具是一个 C 语言软件模型检测工具,其实现主要依赖于谓词抽象和插值技术.

① http://www.satcompetition.org

② http://www.smtcomp.org

国内外研究人员对 SMT 进行了相关的研究和总结. 王建新等人[41]给出了 SAT 问题定义和分类的综述,对各类衍生 SAT 问题进行了阐述和举例说明;Sheini 等人[42]阐述了 SAT 和 SMT 的基本概念,介绍了基于 SAT 技术的 SMT 问题及求解方法等内容;Moura 等人[43]对 SMT 求解技术和背景理论进行了详细地介绍;金继伟等人[44]对 SMT 求解技术进行了简述,包括 SMT 的部分背景理论、SMT 判定算法及优化,重点介绍了 SMT 的基础知识及 SMT 求解器的基本情况.

SMT 的实际应用是目前的研究热点,但是缺乏对 SMT 最新应用及最新进展进行综述的文章,也缺乏对 SMT 竞赛进行归纳和分析的文章. 本文试图全面地介绍 SMT 在工业界和学术界中的最新主流应用,并根据目前 SMT 领域中公认度最高的 SMT-COMP (international satisfiability modulo theories competition),比较主流 SMT 求解器的综合性能及支持背景理论的数量. 为了文章的完整性以及易于后续研究人员理解,本文从 SMT 的起源,即 SAT 问题开始,简洁而又力求全面地介绍 SMT 的发展过程,详细阐述 SAT,SMT 判定算法及理论组合判定算法,分析算法的原理及实现过程,希望对程序分析与验证、软件缺陷检测等领域的研究提供支持.

1 SAT 问题及判定算法

1.1 SAT 问题

SAT 问题是命题逻辑公式(propositional logical formula)的可满足性判定问题,下面先介绍有关命题逻辑的一些必要概念. 命题逻辑中的逻辑运算符号(又称二元连接符)是指定义在布尔集合上的逻辑运算,包括 $\wedge$(与),$\vee$(或),$\neg$(非),$\rightarrow$(蕴含),$\leftrightarrow$(等价),$\oplus$(异或). 命题变元的取值为真或为假,在取值意义上等价为布尔变量. 命题逻辑公式的形成规则为

1) 命题逻辑公式 f(或 SAT 公式 f)可以是单独的命题变元,也称为原子公式;

2) 如果 f 是命题逻辑公式,那么 $\neg f$ 也是命题逻辑公式;

3) 如果 f_1 和 f_2 是命题逻辑公式,那么 $f_1 \perp f_2$ 也是命题逻辑公式,其中 $\perp \in \{\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \oplus\}$. 原子公式 f 与其否定命题 $\neg f$ 统称为文字,而 1 个子句由若干个文字析取而成.

基于命题逻辑,SAT 问题可以进一步被描述为:给定 1 个命题逻辑公式 f,公式 f 由子句集 S 组成,其中 S 由 1 组布尔变量 V 组成,判定是否存在 1 组对于 f 的赋值使得 f 中所有子句取值为真,如果存在,则称公式 f 可满足;否则,f 不可满足. 判定 f 是否可满足是 SAT 问题的核心.

随着 SAT 求解技术不断的发展,SAT 问题的衍生问题也成为了研究热点,例如带权可满足性问题(weighted satisfiability)[45]、参数化带权可满足性问题(3-CNFSAT, q-CNFSAT)[45]、最大可满足性问题(maximum satisfiability)[46]、带权最大可满足性问题(weighted MAX-SAT)[47]、参数化 MAXSAT 问题[48]、参数化 Almost2-SAT 问题(parameterized2-ASAT)[48]等.

1.2 SAT 问题判定算法

SAT 问题判定算法可以分为 2 类:局部搜索算法和完备算法(回溯搜索算法). 局部搜索算法基于随机搜索策略,对于任意给定的问题,它不一定能判断该问题是否可解;而完备算法基于穷举和回溯的思想,可以判断给定的问题是否可满足,对于无解问题拥可给出无解的证明. 判定 SAT 问题时,需要确定给出的问题是否可满足,因此完备算法是研究的重点.

基于搜索回溯的 SAT 问题判定算法由 Davis 和 Putnam 在 1960 年提出,称为 DP(Davis-Putnam)算法[49]. 该算法不适用于大规模命题公式的可满足性问题判定,因此 Davis, Putnam, Logemann, Loveland 对 DP 算法进行了改进,改进后的算法称为 DPLL (Davis-Putnam-Logemann-Loveland)算法[50].

DPLL 算法包括的主要操作为推理(unit-propagation)、纯文字(pure-literal)、决策(decide)、不可满足(fail)和回溯(backtrack).

unit-propagation 对于子句取值未定义并且只含 1 个取值未定义的变量,可以直接令该变量取值为真并加入到赋值中,并根据当前赋值情况对搜索空间进行裁剪. 若文字 l 在赋值中被判定为真,则公式 f 中所有含有 l 的子句都可以从搜索空间中删除,含有 $\neg l$ 的子句可以将 $\neg l$ 去掉.

pure-literal 规则用于化简公式 f 中的变量,若 f 中仅存在 l 或 $\neg l$ 中的 1 种形式,则 l 的值可以立即被判定并从搜索空间中删除包含 l 的子句.

decide 通过一些策略来选取 1 个尚未赋值的变量,并将它的值指定为真或假.

fail 用于检测公式 f 在当前搜索空间下的可满足性以及是否存在冲突.

backtrack 在 fail 检测到公式 f 在当前搜索空间不可满足时进行回溯,返回到上 1 层搜索空间并通过 decide 重新寻找赋值变量.

DPLL 算法的输入是 SAT 公式 f,输出为真(true)或者假(false),主要步骤如下:

1) 使用 fail 操作判断 f 的可满足性,如果不可满足则返回 false;

2) 使用 decide 操作对 f 中 1 个未赋值的变量进行赋值;

3) 使用 unit-propagation 操作裁剪搜索空间;

4)使用 pure-literal 规则化简 f;

5) 使用 fail 判断 f 的可满足性,若 f 可满足,则返回 true;

6) 使用 fail 判断 f 是否存在冲突,如果 f 不存在冲突,则返回步骤 2,否则使用 backtrack 进行回溯;

7) 如果回溯失败,则返回 false,否则执行回溯并返回步骤 2.

现代 DPLL 搜索算法均是在此基础上进行改进的,典型的基于冲突检测的子句学习求解算法 CDCL(conflict-driven clause learning SAT solver, CDCL)描述如下所示:

算法 1. CDCL 求解算法.

```
CDCL(f,v)
UnitPropagation(f,v)
if (ConflictDetection()=Conflict)
   return unsatisfiable;
end if
DecisionLevel=0;
while (not AllVariableAssigned())
   (x,v)=FindBranchingVariable(f,v);
   DecisionLevel=DecisionLevel+1;
   UnitPropagation(f,v);
   if(ConflictDetection()=Conflict)
      ConflictLevel=ConflictAnalysis(f,v);
      if(ConflictLevel<0)
         return unsatisfiable;
      else
         Backtrack(f,v,ConflictLevel)
            dl=ConflictLevel;
      end if
   end if
end while
return satisfied.
```

CDCL 算法与 DPLL 算法的最大区别在于每当 *UnitPropagation* 执行完成后,*ConflictDetetction* 便会检测是否存在冲突,若存在冲突,则调用 *ConflicAnalysis* 分析冲突产生的原因并进行子句学习,以此确定回溯的层次.

现代 SAT 求解器采用了加速搜索的相关策略,例如启发式变量决策提升了 decide 操作的效率,学习子句删除减少了内存的消耗和性能的损失,搜索重启有利于对变量的决策顺序进行调整. 研究人员也提出了一些改进的求解算法,比如 Prestwich[51] 在 2006 年提出的 RANGER 算法结合了贪心随机算法,求解速度比回溯算法快. Audemard 等人[52] 在 2007 年提出了 GUNSAT 算法,该算法的核心是局部搜索算法,可以求解不可满足问题. 许道云等人[53] 在 2007 年提出了带文字改名策略的 DPLL 算法,使用文字改名规则、消解规则、子公式规则和重复规则完成不可满足公式的反驳证明工作.

2 SMT 问题及判定算法

2.1 SMT 问题及背景理论

SMT 问题的基础是一阶逻辑公式,在命题逻辑的基础上补充了项和量词,公式中的函数和谓词符号需要用对应的背景理论解释. 通常情况下,SMT 公式是无量词(存在、任意)的一阶逻辑公式(quantifier free formula),判定公式可满足性的问题称为 SMT 问题.

使用 SMT 描述实际问题并将问题等价的转化为 SMT 公式时,需要将一些数学理论、数据结构理论与公式相结合,这些理论称为 SMT 背景理论,可以增强 SMT 公式的表达能力. 基于背景理论的 SMT 问题可以进一步被描述为:给定 1 个背景理论 T 和 1 个 SMT 公式 F, F 是 T-可满足的(T-satisfiable)[54] 当且仅当存在 1 个赋值使得公式 F 和背景理论 T 同时满足,即 $F\bigcup\{T\}$ 是可满足的.

SMT-LIB[55] 是公认程度较高的 SMT 研究标准,主要包括:

1) 规定了 SMT 求解器标准输入输出的语言规范;

2) 建立了格式严格统一的背景理论知识测试集,用来评价和比较 SMT 求解器的求解能力;

3) 提出了 SMT 背景理论的规范,主要分为未解释函数理论(quantifier free_uninterpreted function,

QF_UF)、数组理论(quantifier free_arrays, QF_A/QF_AX)、整数集/实数集上的线性算数理论(quantifier free_linear integer arithmetic/quantifier free_linear real arithmetic, QF_LIA/QF_LRA)、整数集/实数集上的非线性算数理论(quantifier free_non_linear integer arithmetic/quantifier free_non_linear real arithmetic, QF_NIA/QF_NRA)、整数差分逻辑理论(quantifier free-difference logic over the integers, QF_IDL)、实数差分逻辑理论(quantifier free_difference logic over the reals, QF_RDL)、位向量理论(QF_BV)等.

在目前的实际应用中,主流 SMT 求解器支持并实现的主要理论有 6 个:

1) QF_A/QF_AX. 此背景理论使用 McCarthy 所提出的 axioms[56] 作为理论基础,用于处理高级编程语言中数组. QF_A 的 2 个的核心操作为:①$select(array,k)$,表示选取数组 $array$ 的第 k 个元素;② $store(array,k,v)$,表示先创建 1 个与 $array$ 完全相同的数组 $array'$,然后将数组 $array'$ 的第 k 个元素赋值为 v. 在 QF_A/QF_AX 中,对数组的操作需要符合如下规则[34,38,57]:

规则 1. 如果 $i=j$,则有 $select(store(array,i,v),j)=v$.

规则 2. 如果 $i\neq j$,则有 $select(store(array,i,v),j)=select(a,j)$.

数组背景理论中的相等关系需要通过未解释函数理论进行定义,例如$(array=array'\wedge i=j)$被等价地定义为$(select(array,i)=select(array',j))$. 基于 QF-UF 可以对数组背景理论可以进行如下扩展:

规则 3. 对于 2 个相等的数组 $array$ 和 $array'$,任意 1 个整数(并且是数组的下标)i 使得等式 $select(array,i)=select(array',i)$ 成立.

规则 4. 对于 2 个不相等的数组 $array$ 和 $array'$,存在 1 个整数(并且是数组的下标)i 使得等式 $select(array,i)\neq select(array',i)$ 成立.

2) QF_IDL 和 QF_RDL. 这 2 种理论的公式通常表示为

$$x_1-x_2\odot J, \tag{1}$$

其中,x_1 和 x_2 表示实数或者整数变量,$\odot\in\{=,\geqslant,\leqslant,\neq\}$,$J$ 是 1 个常量.

3) QF_UF. 此背景理论中主要包含没有经过解释的函数符号,比如 $f(x,y)$,$g(h(z))$ 等,每个函数符号都可以被赋予不同的含义.

4) QF_BV. 此理论主要用于处理位向量相关操作,例如左移、右移等.

5) QF_LIA/QF_LRA. 这 2 种理论的公式通常被表示为不等式/等式,比如:

$$c_1x_1+c_2x_2+\cdots+c_ix_i+\cdots+c_nx_n\odot J, \tag{2}$$

其中,$c_1,c_2,\cdots,c_n$ 为常系数,$x_1,x_2,\cdots,x_n$ 为整数变量(在理论 QF_LIA 中)或实数变量(在理论 QF_LRA 中),$\odot\in\{=,\geqslant,\leqslant,\neq\}$.

求解线性不等式集是 QF_LIA 或 QF_LRA 理论的 1 个主要应用,比如 $F=x>3\wedge y<4$,令 $x=6\wedge y=3$ 可使 F 为真.

6) QF_NIA/QF_NRA. 此理论由 QF_LIA 和 QF_LRA 扩展而来,用于求解计算机实际应用中的非线性问题,其表达式可以为任意形式. 比如当 QF_LIA/QF_LRA 中的目标函数为非线性公式时,此问题由线性问题演变为非线性问题.

以上介绍了目前应用较为广泛的主流理论,SMT-LIB 目前支持的理论及相互之间的关系如图 1 所示.

一些主流 SMT 求解器(比如 Z3, CVC3, Boolector)对线性、非线性、位向量等背景理论的描述为[7,55]

$$L::=L\ con\ L\mid\neg L\mid A, \tag{3}$$

$$con::=\wedge\mid\vee\mid\rightarrow\mid\leftrightarrow\mid\oplus, \tag{4}$$

$$A::=\boldsymbol{B}\ rel\ \boldsymbol{B}\mid Id\mid\text{true}\mid\text{false}, \tag{5}$$

$$rel::=<\mid>\mid\leqslant\mid\geqslant\mid=\mid\neq, \tag{6}$$

$$\boldsymbol{B}::=\boldsymbol{B}\ op\ \boldsymbol{B}\mid\sim\boldsymbol{B}\mid ite(cb,t_1,t_2)\mid Const\mid Id\mid Extract(\boldsymbol{B},i,j)\mid SignExt(\boldsymbol{B},k)\mid ZeroExt(\boldsymbol{B},k), \tag{7}$$

$$op::=+\mid-\mid*\mid/\mid\gg\mid\ll\mid\&\mid\mid\mid@, \tag{8}$$

其中,L 代表布尔值表达式,由 A 和 L 组成,¬ 代表逻辑运算符非. con 是逻辑连接符,包括合取(∧)、析取(∨)、蕴含(→)、等价(↔)、异或(⊕). $\boldsymbol{B}$ 代表由整数、实数、位向量构成的项,∼代表补运算,$Const$ 表示常量,Id 表示变量. $ite(cb,t_1,t_2)$ 表示 if-then-else,如果布尔表达式 cb 的值为真,则执行 t_1,否则执行 t_2. $Extract(\boldsymbol{B},i,j)$ 表示抽取位向量 $\boldsymbol{B}$ 的第 i 到第 j 位,产生 1 个新的长度为 $i-j+1$ 的位向量. $SignExt(\boldsymbol{B},k)$ 表示用 k 个 0 扩展位向量 $\boldsymbol{B}$,形成新的长度增加了 k 位的带符号位向量. $ZeroExt(\boldsymbol{B},k)$ 跟 $SignExt(\boldsymbol{B},k)$ 操作类似,区别在于 $ZeroExt(\boldsymbol{B},k)$ 生成不带符号的位向量. op 代表操作符,包括与(&)、或(|)、四则运算符(+,−,*,/)、右移(≫)、左移(≪)、位向量的级联(@).

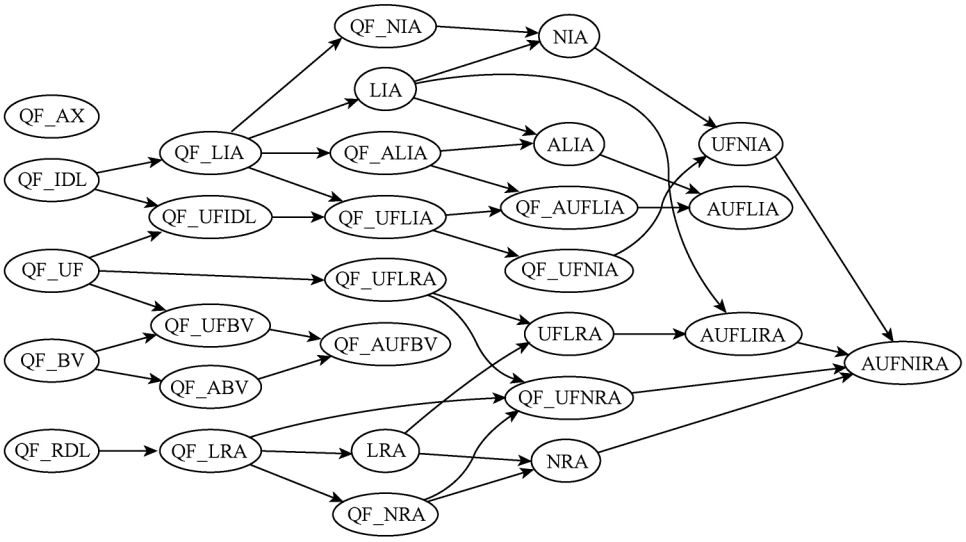

Fig. 1 SMT-LIB background theories and their relations with each other[55]

图 1 SMT-LIB 理论之间的关系[55]

2.2 组合背景理论求解方法

由实际应用问题等价转化而成的 SMT 问题通常包含多种理论，需要结合数组、整数线性算数、实数差分逻辑等多种背景理论才能解释 SMT 公式 F 中每一项的含义，这种情况下单理论判定方法无法满足判定需求，需要组合理论判定方法的支持，主要方法有：Nelson-Oppen(NO)方法[58]、Delayed Theory Combination(DTC)方法[59]和 Ackerman 方法[60].

Nelson 等人[58]于 1979 年提出的 Nelson-Oppen 方法是最为经典的组合理论求解方法，其他方法都是以此为基础改进而成的. Nelson-Oppen 方法的前提是各个背景理论符号集不相交且各理论之间相互独立，每个背景理论 T_i 中的无量词 SMT 公式都需要有 1 个可满足赋值 M_i(基于理论的模型). Nelson-Oppen 方法首先在各个理论间传递含有共享变量的等式，称为接口等式(interface equation)，然后各个理论将此接口等式加入到自己的约束条件中并进行可满足性判定. 如果出现不可满足的理论，则此 SMT 公式是不可满足的，否则重复上述步骤直至没有新的接口等式可以传递. 如果此时没有不可满足的理论，则此 SMT 公式是可满足的. Nelson-Oppen 方法的缺点是过于依赖于共享变量的传递，并且提取共享变量又需要求解器进行纯化操作，需要各个背景理论间相互传递接口等式. 负责的算法框架使得 Nelson-Oppen 方法求解效率低下.

Delayed Theory Combination 方法由 Bozzano 等人[59]提出，避免了各个背景理论间的接口等式传递步骤，将组合理论的可满足性判定请求统一交给顶层求解器处理，简化了求解框架，求解效率好于 Nelson-Oppen 方法. Nelson-Oppen 方法和 Delayed Theory Combination 方法的区别如图 2 所示.

Ackerman 方法是 Nelson 等人[60]基于 Nelson-Oppen 方法提出的改进算法，主要用于求解含有未解释函数理论的组合背景理论问题，通常和 DTC 方法结合使用. 但是此方法具有局限性，仅提升在未解释函数理论下 SMT 的求解效率.

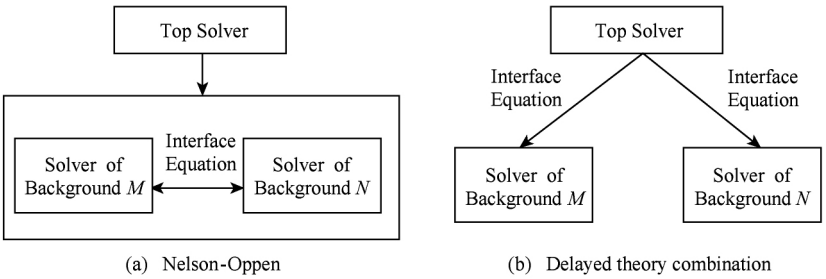

Fig. 2 The difference between NO and DTC

图 2 NO 方法和 DTC 方法的区别

2.3 SMT 问题判定算法

2.3.1 求解 SMT 问题的积极类算法

积极类算法直接将 SMT 公式转化为可满足性意义上等价的 SAT 公式[61],然后利用 SAT 求解器进行求解工作.这个方法的好处在于它可以充分利用高效的 SAT 求解器,不用针对 SMT 的复杂背景理论去开发特定理论求解器,积极类算法是早期求解 SMT 问题的主要方法,主要包括 Per-Constraint Encoding[62-63] 方法和 Small-Domain Instantiation[64-65] 方法.

Per-Constraint Encoding 方法是指:对于 SMT 公式 F,如果 F 含有未解释函数,则通过 Goel 等人[66]提出的方法将 SMT 公式 F 转化为基于可满足意义上等价的 SAT 公式,主要思想是将 F 中的所有形如 $a_i=a_j$ 的项用 1 个新的变元 b_{ij} 来表示,再用 SAT 求解器对公式 F 进行求解.b_{ij} 需要满足传递性,即 $(b_{ij}\wedge b_{jk})\rightarrow b_{ik}$.

Small-Domain Instantiation 方法是指:对于 SMT 公式 F,如果 F 中的项只包含变元,则用布尔变量组成的位向量代替公式中的变元,将 SMT 公式转化为基于可满足性意义上等价的 SAT 公式,利用 SAT 求解器求解公式 F 的可满足性,达到求解原 SMT 公式可满足性的目的.如果公式 F 除了变元外还包含未解释函数,则需要利用 Ackerman 方法将公式 F 转化为只包含等式逻辑的公式,然后判定 F 的可满足性.

积极类算法的求解效率依赖于实际问题建模为 SAT 问题的效率和 SAT 求解器自身的效率.随着实际应用问题规模的不断增大,建模为 SAT 问题后得到的 SAT 公式长度呈现指数级增长的趋势,算法的时间开销难以接受.因此,这类算法并不适用于求解工业界的大规模实际应用问题.

2.3.2 求解 SMT 问题的惰性算法

惰性算法[29]是目前大多数 SMT 求解器采用的算法[67-69],算法主要步骤如下:

1) 对 SMT 公式进行预处理,把公式中的命题变量替换为布尔变量,再将 SMT 公式转化为可满足性意义上等价的 SAT 公式;

2) 检查此 SAT 公式是否可满足,如果不可满足,那么 SMT 公式也不可满足,算法结束;

3) 如果 SAT 公式可满足,则结合 SMT 背景理论去判断 SMT 公式的可满足性,返回判断结果,算法结束.

此算法的优势在于,若算法在步骤 2 中已判定 SAT 公式不可满足,则无需进行后续判定工作,提高了求解效率.惰性算法是 SAT 求解器与对应的背景理论相结合的产物,典型的惰性算法是 DPLL(T)[70-71]算法.

DPLL(T)算法伪代码如下:

算法 2. DPLL(T).

输入:SMT 公式 F;

输出:true(真)或者 false(假).

```
预处理公式 F 得到 F 的 SAT 形式的公式 f
if (f 是不可满足的)
  return false;
else
  for (每个 f 的模型 M)
    检查 M 是否与背景理论一致;
    if (存在一个与背景理论一致的模型 M)
      return true;
    end if
  end for
  if (每一个模型 M 都与背景理论不一致)
    return false;
  end if
end if
```

DPLL(T)算法的基础是通用求解架构 DPLL(X),此架构独立于具体的背景理论,将 X 替换为某个具体的背景理论 T 即可成为针对此背景理论的具体求解算法.但是在求解过程中,背景理论求解器的参与程度较小,通过理论预处理、选择分支、理论冲突分析、理论推导等技术可以提升背景理论求解器的参与程度,从而获得更高的求解效率[67-68,71-72],Sebastiani[73]使用优化技术同样达到了提高求解效率的目的.

一些研究人员注重于如何将 SAT 求解器与背景理论更好地结合以获得更高的求解效率,并做了相关的研究[67,74-75].

3 SMT 求解器及性能比较

随着 SMT 背景理论的逐渐成熟以及 SMT 问题判定算法的不断发展,许多 SMT 求解器能够满足学术界的研究需求,一些成熟的 SMT 求解器也能处理大规模工业化的应用问题.目前主要的求解器有:Z3[34],CVC3[57],CVC4[35],Boolector[38],

Yices2[36], MathSAT[76], Mathsat4[37], Verifun[75], Beaver[77], TSAT[78], Barcelogic[79], VeriT[80]等. 下面重点介绍4个有代表性的SMT求解器.

Z3以其优秀的综合求解能力被业界所认可,是目前为止在扩展性、表达能力以及求解效率等方面都较为出色的SMT求解器. Z3的求解框架集成了基于DPLL的SAT求解器,并支持未解释函数、算术运算、数组等背景理论,核心求解算法是DPLL(T)算法和DTC方法. 许多工业界的项目都用到了Z3,例如Pex[39], HAVOC[81], Yogi[82], SLAM/SDV[83], Vigilante[84]等.

CVC3和CVC4是纽约大学和爱荷华大学联合开发的SMT求解器,使用SAT求解器和理论求解器的Search Engine作为核心求解框架,支持支持多种理论的求解. 与CVC3相比,CVC4能更好地支持SMT-LIB并且优化了求解框架.

Yices2是SRI International计算机科学实验室开发的SMT求解器,前身是Yices,支持数组理论、线性算数运算理论、位向量理论,核心求解算法是优化后的同余闭包算法. 在组合理论的求解方面,Yices使用位移等式改进了传统的Nelson-Oppen方法,并结合动态Ackerman方法进行求解. 在实际应用领域,Yices作为定理证明的核心部件被整合到了SRI International开发的定理证明器PVS中. Yices2对Yices进行了优化和改进,减少了复杂类型的数量,只含有整数、实数、位向量和布尔变量4种原子类型并简化了类型的表示,例如将int(整数)类型视为real(实数)类型的子类型. Yices2提供了符合SMT-LIB标准的接口,增加了对元组和标量等数据类型的支持.

其他SMT求解器也都有各自的特点. 例如,Barcelogic的核心求解算法为Ackerman方法、DTC方法以及位移等式策略. MathSAT4在结合了Ackerman方法和DTC方法的基础上,使用了动态Ackerman方法求解可满足性问题. Boolector可以求解含有位向量理论和可扩展的数组理论的可满足性问题. VeriT是基于改进的Nelson-Oppen方法开发的SMT求解器,仅支持无量词理论和整数差分理论的可满足性问题求解.

为了推动SMT求解技术及SMT求解器的进步,可满足性理论及应用国际学术年会自2005年开始,每年举办SMT-COMP竞赛,到2015年为止共成功举办了11届比赛. 用于评判各个求解器求解能力的标准测试集来源于SMT-LIB,其数量从最初的1 400个增加到如今的10 000多个. 标准测试集是由各个背景理论测试集组成的,例如线性算数测试集(LIA)、位向量(QF_BV)测试集等. 表1包含了自2005年开始到目前为止每一届SMT-COMP的比赛结果. 表1以每个SMT求解器所支持背景理论的数量、在不同背景理论测试集上的排名以及综合

Table 1 Winners of the Each SMT-COMP Editions

表1 历届SMT-COMP的冠军

Competition's Edition	Winner	Background Theories
SMT-COMP2005	Barcelogic	QF_UF, QF_RDL, QF_IDL, QF_UFIDL
SMT-COMP2006	Yices	QF_UF, QF_RDL, QF_IDL, QF_UFIDL, QF_LRA, QF_LIA, QF_UFLIA, QF_UFBV32, QF_AUFLIA, AUFLIA, AUFLIRA
SMT-COMP2007	Yices	QF_RDL, QF_UFIDL, QF_AUFLIA, QF_UFLIA, QF_LRA, QF_LIA
SMT-COMP2008	Z3	QF_RDL, QF_UFIDL, AUFLIA, AUFLIA, AUFLIRA, QF_AUFLIA, QF_UFLRA, QF_UFLIA, QF_LIA
SMT-COMP2009	Yices2	QF_AX, QF_UFLRA, QF_AUFLIA, QF_UFLIA, QF_UF, QF_RDL, QF_LRA
SMT-COMP2010	CVC3	QF_AUFLIA, QF_UFNRA, QF_ABV, QF_AX, AUFLIA, AUFLIA, AUFLIRA, AUFNIRA, UFNIA
SMT-COMP2011	Z3	QF_BV, QF_UFLIA, QF_IDL, AUFLIA, AUFLIA, AUFNIRA, QF_UF, QF_AUFLIA, QF_LRA, QF_LIA
SMT-COMP2012	Z3	QF_IDL, AUFLIA, AUFLIA, AUFNIRA, QF_UF, QF_AUFLIA, QF_LRA, QF_LIA
SMT-COMP2013	Z3	AUFLIA, AUFLIRA, AUFNIRA, LRA, QF_AX, QF_IDL, QF_UFIDL, QF_UFLIA, UFLRA, UFNIA
SMT-COMP2014	Z3	ALIA, AUFLIRA, LIA, NIA, NRA, QF_IDL, QF_NIA, QF_NRA, QF_UFIDL, QF_UFLIA, QF_UFLRA, QF_UFNRA, UFIDL, UFLRA, UFNIA
SMT-COMP2015	Z3	ALIA, AUFLIRA, BV, NIA, QF_ANIA, QF_ANIA, QF_AUFLIA, QF_IDL, QF_LIA, QF_NIA, QF_NRA, QF_UFLIA, QF_UFNRA, UFIDL

评分作为依据,在 Winner 一栏中列出了每一届比赛的冠军,并在 Background Theories 一栏中给出了对应 SMT 求解器在参赛时所用到的背景理论测试集,在这些测试集上该 SMT 求解器综合的求解效率要好于其他 SMT 求解器. 由表 1 可以看出,Barcelogic 获得了第 1 届竞赛的冠军,并在 QF_UF 等 4 个背景理论测试集上有着突出的表现. 在随后的比赛中,Yices 和 Yices2 也获得了冠军,支持背景理论的数量远远大于 Barcelogic. 在 SMT-COMP2010 中,CVC3 也取得了不错的成绩. 最近 5 年中,Z3 始终排在第 1 位,支持背景理论的数量也在逐年增加,在大多数理论测试集上的求解速度快于其他求解器,综合性能较好.

表 2 是 2015 年 SMT-COMP 的比赛结果,Sequential Performances 代表顺序性能的评分,Parallel Performances 代表并行性能的评分,Normal 和 Industrail 分别代表各个 SMT 求解器在标准测试集和工业测试集上的评分,以求解速度、支持的背景理论数量等因素为依据,在 Rank 一栏中给出了每个求解器的综合排名.

Table 2 Ranking of SMT-COMP 2015

表 2 SMT-COMP2015 综合排名

SMT Solver	Sequential Performance		Parallel Performance		Rank
	Normal	Industrial	Normal	Industrial	
Z3	159.36	159.36	139.34	139.34	1
CVC4	144.67	124.59	144.74	124.63	2
Yices	101.91	81.64	101.91	81.64	3
MathSAT	79.77	60.94	79.77	60.94	4
CVC3	24.44	25.19	24.44	25.19	6
Boolector	16.61	16.61	16.61	16.61	7

由表 2 可以看出,无论是在标准集测试上还是在工业测试集上测试,Z3 求解器的综合性能都要比其他 SMT 求解器好. Z3 在标准测试集和工业测试集上的评分相同,说明 Z3 具有较好的稳定性,Z3 的 Sequential Performance 的评分高于 Parallel Performance 的评分. 排名第 2 的 CVC4 的评分与 Z3 相差不大,唯一的区别是在标准测试集和工业测试集上 Parallel Performance 的评分都略高于 Sequential Performance 的评分,是为数不多的在 Parallel Performance 的评分上取得较好成绩的求解器之一. Yices 和 MathSAT 求解器评分低于 Z3 和 CVC4,在 2 个测试集上的 Sequential Performance 的评分和 Parallel Performance 的评分均保持不变,较为稳定. CVC3 和 Boolecter 的评分与其他求解器相差过大,无论是在标准测试集上还是在工业测试集上,求解大规模可满足性问题的能力都要低于其他求解器.

在 2015 年 SMT-COMP 中,举办方提供了 40 种不同的背景理论测试集,基于每个 SMT 求解器在各个测试集上的综合评分,表 3 给出了主流 SMT 求解器在不同背景理论测试集中的排名. 其中 LIA 表示整数线性算数理论,QF_ABV 表示无量词固定位向量数组理论,QF_IDL 表示无量词整数差分逻辑理论,QF_NIA 表示无量词整数非线性算数理论,QF_UFLIA 表示无量词未解释函数的整数线性算数理论,“ ”代表此 SMT 求解器不支持对应理论的求解. 从表 3 中可以看出,Z3 求解器支持主流的背景理论数量最多.

Table 3 Ranking of SMT Solvers Based on Different Background Theory's (SMT-COMP 2015)

表 3 不同背景理论下的 SMT 求解器排名 (SMT-COMP 2015)

SMT Solver	QF_IDL	QF_NIA	QF_UFLIA	LIA	QF_ABV
Z3	1	1	1	2	5
CVC4	3	7	3	1	4
Yices	2		2		2
MathSAT			5		3
CVC3		6		4	
Boolector					1

4 SMT 应用

随着 SMT 判定算法的不断发展以及 SMT 求解器的逐渐成熟,人们开始使用 SMT 解决一些实际问题,例如测试用例自动生成、程序缺陷检测、RTL 验证、程序分析与验证、线性逻辑约束公式优化问题求解等.

4.1 测试用例自动生成

4.1.1 基于 SMT 的测试用例自动生成

测试用例自动生成是设计和编写软件测试用例一种方法,也是软件测试的一种重要手段,常用于检测程序缺陷. 基于 SMT 的测试用例自动生成技术主要分为获取程序执行路径和检查路径可满足性这 2 个部分. 获取程序的执行路径主要依赖于动态符

号执行,即在不执行程序的前提下,使用具体数值代替程序变量作为程序的输入,模拟程序的执行,分析1条指定路径会触发程序中哪些代码的执行,并记录下此路径.检查路径可满足性需要将程序执行路径转化为SMT公式,然后使用SMT求解器判断公式的可满足性.具体思想如下:首先利用基于动态符号执行的代码模拟执行器模拟一条具体的程序执行路径,同时记录路径中的条件语句和赋值语句,再通过SMT背景理论将这些语句转化为SMT公式F,例如可以用未解释函数背景理论将赋值语句表示为等式的合取形式,而数组赋值语句则需要数组理论的支持,利用SMT求解器对判断F的可满足性,此时会出现2种情况:

1) 如果F可满足,F的具体可满足性赋值可作为该条路径的输入(测试用例).通过修改F中的某个条件,例如将分支语句if($a=0$)中的表达式($a=0$)改为(a !=0),可以构建出1条新的执行路径,将新的路径转化为公式F',通过SMT求解器求解F',得到新的输入(测试用例),利用新的输入再进行新一轮的路径构造、约束求解.通过这种迭代的路径生成方法,动态符号执行可以持续遍历程序的可执行路径,直到所得到的测试用例数量达到预期值,从而实现了测试用例的自动生成.

2) 当F不可满足时,说明当前执行路径不正确,修改某一分支语句的分支条件后进行新一轮的路径构造、约束求解.

4.1.2 具体示例

图3(a)中函数*addition*()是1个加法函数,函数输入*choice*的不同取值会产生不同的执行路径,从而影响*sum*的取值.图3(b)是*addition*()的2条执行路径.

基于SMT的测试用例自动生成的过程如下:首先使用代码模拟执行器得到*addition*()的1条执行路径,即图3(b)中的路径1,将路径1转化为SMT公式,记为

$$F_1=(a=0)\wedge(choice>1)\wedge(sum=2),\quad(9)$$

利用SMT求解器判断出F_1是可满足的,且可满足解为($choice=2$),从而得到了第1个测试用例.然后将if语句中的表达式取反,得到图3(b)中的路径2,将路径2转化为SMT公式,记为

$$F_2=(a=0)\wedge(choice\leqslant 1)\wedge(sum=2),\quad(10)$$

利用SMT求解器判断出F_2是可满足的,且可满足解为($choice=1$),达到了测试用例自动生成的目的.

```
void addition (int choice)
{
  int a=1;
  if (choice>1){
    sum=a+1;
  }
  else{
    sum=a-1;
  }
}
```

(a) Source code

```
Path 1:
(choice>1)
Assignment statement (a=1)
Conditional statement (choice>1)
Assignment statement (sum=2)
Path 2:
(choice≤1)
Assignment statement (a=1)
Conditional statement (choice≤1)
Assignment statement (sum=0)
```

(b) Path analysis

Fig. 3 Record the number of requests

图3 记录请求次数

4.1.3 SMT在测试用例自动生成中的应用

许多成熟的工具中都用到了基于SMT的测试用例自动生成方法,例如Pex[39]工具将.Net组件和基于SMT的测试用例自动生成技术相结合,使用Z3求解路径的可满足性,可以自动生成.Net程序的测试用例,例如C#程序.Pex也为含有复杂数据结构的程序提供测试用例自动生成的功能.

SAGE[85]是1个白盒测试工具,使用代码覆盖率检查工具Nirvana和约束产生工具TruScan将程序路径转化为SMT公式,利用Z3求解公式的可满足性并自动生成新的测试用例,结合微软模糊测试的基础框架,SAGE可以找出程序中的大多数错误.

JPF-SE[86]工具利用符号执行技术获取程序的执行路径并转化为SMT公式,使用Yices求解路径的可满足性并生成测试用例并作为输入提供给工具内的其他组件.

Yogi[82]是1个基于SMT的C语言静态分析和测试工具,该工具使用Z3求解程序路径的可满足性并产生测试用例,被集成到微软的静态驱动检测框架中,完成了69个Windows Vista的驱动检测工作.

Arcaini等人[87]提出了一种使用区分配置(distinguishing configuration)检查特征模型错误的方法.该方法利用Yices求解测试谓词(test predicate)

的可满足性,如果可满足,则具体的可满足赋值即为区分配置(distinguishing configuration),使用区分配置作为测试用例进行特征模型的错误检查工作,实现了测试用例的自动生成;如果不可满足,则产生特征模型的突变(mutated)模型.

4.2　程序缺陷检测

4.2.1　基于 SMT 的程序缺陷检测

程序缺陷检测的目的是检查程序是否违反了给定的安全属性,例如是否有死锁、是否存在安全漏洞等.软件测试是检测嵌入式程序缺陷的一种常用方法,所需的测试用例可以由人工编写或者使用动态符号执行等技术自动生成.当程序规模很大时,很难获取覆盖程序所有执行路径的全部测试用例,时间开销也是难以接受的.因此,软件测试只适用于寻找软件的缺陷,无法保证程序不含有某个指定的缺陷.基于模型检测[88]的程序缺陷检测方法是一种自动检测技术,通过对程序进行抽象得到 1 个有限的状态空间,减少了缺陷检测带来的时间开销,在一定程度上减小了缺陷检测的难度.但是程序规模的不断增大导致状态空间也随之增大,状态空间爆炸是模型检测不可避免的问题.

基于 SMT 的有界模型检测(bounded model checking,BMC)方法成为了新的研究热点.主要思想是检测程序在界限 K 内是否满足给定的安全属性(property),给定系统 I,1 个安全属性 P,以及 1 个界限(bound)K,BMC 会将系统 I 展开 K 次得到验证条件 V,V 是可满足的当且仅当 P 在界限 K 内有 1 个反例.这里的界限 K 是指将源程序中的循环结构(比如 for 循环)展开 K 次.V 是源程序所转化成的等价 SMT 公式 F.基于 SMT 的有界模型检测是指将上述 F 与 P 的反($F\wedge\neg P$)送入 SMT 求解器,如果($F\wedge\neg P$)可满足,证明程序的某一条执行路径会违反安全属性,通过 SMT 求解器返回的具体可满足赋值可以得到使得程序出错的具体输入,如果($F\wedge\neg P$)不可满足,则证明程序在界限 K 内不违反安全属性.

4.2.2　具体示例

下面用 1 个例子说明 SMT 求解器在程序缺陷检测中的应用.图 4(a)是 1 个 C 语言程序,其功能是计算斐波那契数列第 n 项的数值,并将结果存在整型数组 $result[0]$中.斐波那契数列是指这样 1 个数列:数列的第 0 项为 0,第 1 项为 1,第 2 项为 1,并且从第 2 项起,每一项都是前 2 项的数字之和,例如,斐波那契数列的前 6 项为:0,1,1,2,3,5.图 4(b)表示函数 $Fib(\text{int } n)$的静态单赋值(static single assignment, SSA)形式,SSA 形式与源程序的区别在于 2 点:1)SSA 形式中的每个变量名仅被赋值 1 次.对于同一变量的多次赋值采用"原变量名+赋值次数"的形式来取代原变量名;2)SSA 形式中消去了原程序中的循环结构(比如 while),利用循环体展开的形式等价表示循环结构.图 4(b)是将源程序中的 while 循环展开 3 次的结果.图 4(c)是将 SSA 形式中的语句进行合取得到的等价表示形式,称为 SMT 公式 F.程序中存在数组 $result$,需要检查是否存在数组越界问题,由于数组只含有 1 个元素,因此安全属性 P 可以被表示为($j=0$).SMT 求解器将逻辑公式($F\wedge\neg P$)作为输入,通过结合相应的背景理论(例如本程序需要用到未解释函数理论)对公式 F 的可满足性进行求解,可知($F\wedge\neg P$)不可满足,程序不存在数组越界问题,没有违反给定的安全属性.

```
int Fib (int n){
  assert(n≥0);
  int a=0,b=1;
  int i=0;
  int j=0;
  int result[1];
  if (n=0)
    result[j]=0;
  if (n=1)
    result[j]=1;
  while (i<(n-1)){
    result[j]=a+b;
    a=b;
    b=result[j];
    i=i+1;
    }
}
```

(a) Source code

```
int Fib (int n0){
  assert(n0≥0);
  int a0=0,b0=1;
  int i0=0,result0=0;
  result1=a0+b0;
  a1=b0;
  b1=result1;
  i1=i0+1;
  result2=a1+b1;
  a2=b1;
  b2=result2;
  i2=i1+1;
  result3=a2+b2;
  a3=b2;
  b3=result3;
  i3=i2+1;
  assert(i=n-1)}
```

(b) SSA form

$$(n_0\geqslant 0)\wedge(a_0=0\wedge b_0=1)\wedge(i_0=0\wedge result_0=0)\wedge(result_1=a_0+b_0)\wedge(a_1=b_0)\wedge(b_1=result_1)\wedge(i_1=i_0+1)\wedge(result_2=a_1+b_1)\wedge(a_2=b_1)\wedge(b_2=result_2)\wedge(i_2=i_1+1)\wedge(result_3=a_2+b_2)\wedge(a_3=b_2)\wedge(b_3=result_3)\wedge(i_3=i_2+1)\wedge\text{assert}(i_3=n_0-1)$$

(c) SMT formula

Fig. 4　Converting the source code into the SMT formula

图 4　源程序转化为 SMT 公式

4.2.3　SMT 在程序缺陷检测中的应用

基于 SMT 的有界模型检测方法中的界限 K 使得此方法在医疗、通信等嵌入式程序的验证中有着很大的优势.原因在于相对于一般软件,嵌入式程序

的代码量小,且嵌入式程序不鼓励使用大量的递归和循环语句,很少见到循环次数超过 K 的循环体,只需要证明程序在 K 步内满足安全属性即可.

Cordeiro 等人[7]在 2012 年提出了一种基于有界模型检测的嵌入式软件缺陷检测方法.此方法先将嵌入式程序的 ANSI-C 语言源文件转化为控制流图(control flow graph, CFG),再把 CFG 对应的 GOTO-Program 转化为 SSA 形式,对 SSA 进行处理并提取出用户断言和安全属性后得到与源程序等价的 SMT 公式,然后使用 Z3 或者 Boolecter 求解公式的可满足性,根据求解结果判定该嵌入式程序是否满给定的安全属性.此方法在软件 ESBMC (efficient SMT-based context-bounded model checker)中得以实现,图 5 是 ESBMC 的结构图,描述了 ESBMC 进行有界模型检测的步骤. ESBMC 支持 ANSI-C 语言中许多数据结构、变量的检查,其中包括标量数据类型(比如整型、长整型等)、固定点算数、算术溢出、数组、结构体、指针、动态内存分配等. ESBMC 会将它们转化为与 SMT 背景理论相符合的公式,再将公式送入 SMT 求解器求解.

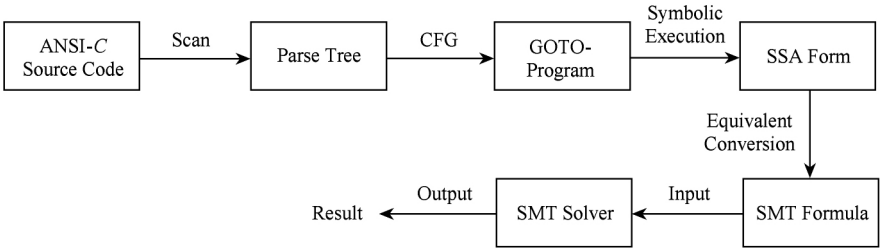

Fig. 5 Overview of the ESBMC architecture

图 5 ESBMC 结构概览

Ramalho 等人[89]在 2013 年提出了基于 SMT 的 C++程序缺陷检测方法,可以检测 C++语言中的标准库函数以及 C++特有的模板、容器、继承、异常的缺陷. C++库函数在有界模型检测中会产生大量复杂而又冗余的验证条件,因此,Ramalho 等人[89]为 C++的库函数建立 C++操作模型(C++ operational model, COM),作为 C++库函数的简单表示,在保证有界模型检测的正确性的同时减少了产生的验证条件的数量,并使用 Z3 等 SMT 求解器判定验证条件的可满足性,提高了验证效率. COM 中包含 ANSI-C 库函数,保证了对 ANSI-C 语言检测的支持.

Cordeiro 等人[90-91]提出了基于 SMT 的多线程软件上下文界限模型检测方法,可以检测多线程软件的缺陷.文中把多线程并发视为以不同顺序激活不同线程的线性序列,将程序的所有可达状态记为状态空间 W,将线性序列、可达状态已经安全属性转化为 SMT 公式,利用 Z3 等 SMT 求解器求解公式的可满足性,判断每一个状态是否会违反安全属性.通过可达树(reachability tree, RT)、惰性方法(lazy approach)、调度算法以及下近似和展开方法(under-approximation and widening approach)完成了多线程软件的模型检测工作.文中还对 Pthread library[92]进行了建模,包括互斥锁操作、条件等待等操作.

Pereira 等人[93]在 2016 年提出了基于 SMT 的统一计算设备架构(computer unified device architecture, CUDA)程序上下文界限模型检测方法,CUDA 是由 NVIDA[94]开发的 1 个并行编程平台和应用编程接口模型,目的是为了充分利用 GPU (graphics processing unit)的能力.他们在 ESBMC 的基础上开发了 ESBMC-GPU,能很好地对 CUDA 函数库进行建模,并利用 SMT 进行模型检测工作.为了缓解多线程环境下状态复杂的问题,他们将单调偏序规约(monotonic partial order reduction, MPOR)应用到 CUDA 程序中来消除冗余的程序执行路径和 RT[95]中的冗余状态.

基于 SMT 的模型检测工具在工业界也得到了应用,Ball 等人[96]使用 SLAM 工具对 Windows NT 驱动程序进行了模型检测工作,检测了其中是否存在死锁等问题.

4.3 RTL 验证

4.3.1 基于 SMT 的 RTL 验证

RTL 验证是检验集成电路功能性错误的方法.随着集成电路规模的不断增大,普通模拟验证已经无法满足大规模集成电路验证的需要,因此,形式化验证受到了高度的关注.目前工业界采用了一些 SAT 求解器处理求解门级电路的问题,比如 Minisat 和 BerkMin 等.研究人员也对 SAT 求解器在电子电路验证中的应用做了相关研究[97],并提出了自动

测试图样产生(automatic test pattern generation, ATPG)、整数线性规划(integer linear programming, ILP)、超图划分等求解方法[98]. 但是这些方法求解门级电路所需要的时间开销让人难以接受,因此,基于 SMT 的 RTL 混合可满足性求解方法是目前的研究热点.

4.3.2　具体示例

下面用 1 个简单的例子来说明 SMT 在 RTL 验证中的应用. 如图 6 所示,电路 E 只含有 1 个与门和 1 个或门,需要验证是否存在 1 组输入使得电路输出结果 Z 为 1(真). 具体步骤如下:首先根据电路图结构将 E 转化为等价的 SMT 公式 $F=(a\wedge b)\vee c$,再利用 SMT 求解器求解 F 的可满足性,可知当 $a=1, b=1, c=0$ 时,F 可满足:即 $Z=1$,达到了验证的目的.

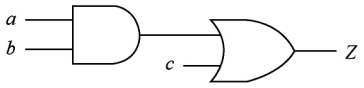

Fig. 6　Circuit diagram E

图 6　电路图 E

4.3.3　SMT 在 RTL 验证中的应用

许多形式化验证工具,包括一些工业界采用的工具都是通过位级模型实现 RTL 验证,主要思想是将高层级的 RTL 设计转化为位级信息,再对位级信息进行展开验证工作. 这种方法不能充分利用高层级所含的结构,在转换中会丢失高层级的信息,可扩展性低.

Kroening 等人[99]提出了一种利用高层级信息模型进行 RTL 验证的方法,该方法利用谓词抽象技术和 SMT 求解器进行位级以及字级的验证(word-level verification). 结合了反例指导的抽象框架[100](counterexample guided abstraction refinement frame)的谓词抽象技术将系统的实际状态空间映射到 1 个抽象的、状态数较少的空间中,并在系统表现出的特性上与原有系统保持一致. 图 7 是 RTL Verilog 谓词抽象技术的大体框架.

Puri 等人[101]提出了基于 SMT 的自动 RTL 测试生成器 SI-SMART(swarm intelligence-SMART),目的是解决传统有界模型检测方法和基于符号执行的检测方法对循环处理能力不足的问题. 在 DUT (design under test)中,循环是很常见的,但是有界模型检测或者基于符号执行的检测方法只能将循环展开至多 K 次,再进行 RTL 验证工作. 此方法不适

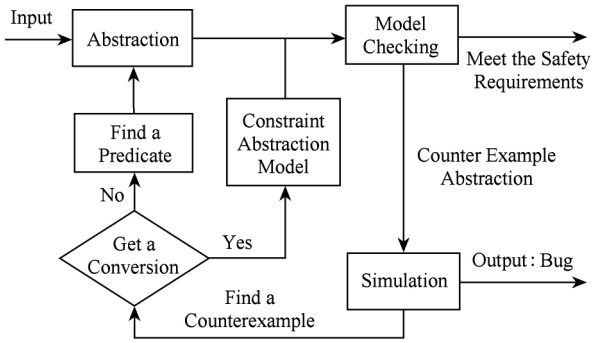

Fig. 7　The predicate abstraction techniques of RTL verilog

图 7　RTL verilog 的谓词抽象技术

用于 DUT. SI-SMART 先对 DUT 中的循环进行抽象,再找出直接或间接影响目标分支条件的变量,分析它们之间的递推关系技术并以此消除控制流图中的循环展开,从而解决了需要按照一定界限展开循环的问题.

Brady 等人[102]提出了硬件设计的自动项级(term-level)抽象技术,是一种基于形式化逻辑的、针对字级设计的抽象技术,所有数据用抽象的项、功能函数和未解释函数等形式表达,解决了目前大多数抽象技术所面临的抽象等价性问题,即是否可以把组件 C 等价的抽象为另一个表示 C'. 此抽象技术先利用随机模拟技术检验功能模块是否可以用未解释函数来抽象,再利用静态分析技术计算抽象条件,最后利用 SMT 求解器验证项级抽象结果的可满足性. 目前这种技术已经成功的应用于处理器设计验证、接口逻辑验证等领域.

赵燕妮等人[103]利用 SMT 求解 RTL 的可满足性问题,其主要思想是将 RTL 电路视为 1 个超图,然后基于超图划分的方法得到 1 个割集最小的等量超图划分,形成有限个超图子问题,再利用 Yices 求解子问题的可满足性.

Gent 等人[104]提出了基于有界模型检测和群体智能的 RTL 功能测试方法,该方法使用 HYBRO[105](hybrid analysis and branch coverage optimizations)抽取特定的程序执行路径并将其转化 SMT 公式,利用 Z3 求解这些 SMT 公式的可满足性,再结合有界模型检测的方法进行 RTL 验证.

Kunapareddy 等人[10]对 LPSAT 和 SMT 在 RTL 验证中的应用进行了比较,结论表明:基于 Z3 的 RTL 验证方法在代码复杂度、执行时间和迭代次数上的表现均优于 LPSAT 方法.

4.4 程序分析与验证

4.4.1 基于 SMT 的程序分析与验证

基于 SMT 的程序分析与验证是一种形式化方法,基础思想来源于 Floyd 和 Hoare 提出的弗洛伊德-霍尔逻辑(Floyd-Hoare logic). 该方法将前置条件(pre-condition)、循环不变量(loop invariant)和后置条件(post-condition)以断言(assert)的形式引入程序验证中,三者分别判断程序在运行前、运行中以及运行结束时的正确性,通过判定断言得成立情况检验程序的正确性.

4.4.2 具体示例

用 1 个简单的程序来说明 SMT 在程序分析与验证中的应用. 图 8(a)是 1 个求最大公约数的程序 GCD,x 和 y 是程序的输入,m 是程序中的辅助变量. 为该程序加上 pre-condition,loop invariant 以及 post-condition 后得到的程序如图 8(b)所示. 具体过程如下:1)程序 GCD 的 2 个输入为正数是求解最大公约数的基本条件,即 $F_1=(x\geqslant 0)\wedge(y>0)$. 运算过程中的被除数要大于除数,即 $F_2=(x\geqslant y)$. 将 F_1 和 F_2 加入到 pre-condition 中. 2)在 while 循环中,x,y 和 m 需要恒大于 0 且 x 要始终大于等于 y,即 $F_3=(x\geqslant 0)\wedge(y>0)\wedge(x\geqslant y)\wedge(m\geqslant 0)$,将 F_3 加入到 loop invariant 中. 3)最大公约数是 1 个整数,即 $F_4=(y\geqslant 0)$,将 F_4 加入到 post-condition 中. 然后通过动态符号执行技术遍历程序 GCD 的所有执行路径,再将这些路径转化为 SMT 公式 F,然后将 pre-condition,loop invariant 以及 post-condition 转化为 SMT 公式并与 F 合并,记为 F',通过 SMT 求解器求解 F'的可满足性,若 F'不可满足,说明程序中含有错误,若 F'可满足,说明在所有断言得到满足性的情况下,程序是正确的,验证了程序的部分正确性. 本例中的程序 GCD 是正确的.

```
int GCD (int x,int y){
  while (true) {
    int m=x % y;
    if (m=0)
      return y;
    x=y;
    y=m;
  }
}
```

(a) Source code

```
int GCD (int x,int y){
/*pre-condition*/
  assert(x≥0)∧(y>0)
    ∧(x≥y);
  while (true) {
    int m=x%y;
/*loop invariant*/
  assert(x≥0)∧(y>0)∧
    (x≥y)∧(m≥0);
    if (m=0)
      return y;
    x=y;
    y=m;
/*post-condition*/
  assert(y≥0);}}
```

(b) Source code with assertions

Fig. 8 Greatest common divisor program

图 8 求最大公约数程序

4.4.3 SMT 在程序分析与验证中的应用

2005 年,Leroy[15] 研发了 1 个 C 语言编译器 CompCert,该编译器是经过形式化验证的优化编译器,主要功能是将 C 语言编译为 PowerPC 汇编代码,Leroy 对编译的每一步操作给出了严格的数学证明,大大提高了 CompCert 的可信度. 但是这种基于数学证明的程序分析与验证方法难度大、耗时长,因此,基于 SMT 的程序分析与验证成为了新的研究热点.

何炎祥等人[106] 提出了使用 SMT 求解器进行路径敏感程序验证的方法,主要思想是通过最大强连通分量压缩循环路径并使用基于布尔表达式的方法对路径空间进行抽象,减少了待验证程序路径的规模,再结合动态符号执行技术和抽象解释技术将压缩后的程序路径转化为 SMT 公式,使用 Z3 求解公式的可满足性,如果可满足,证明路径是正确的,达到了程序验证的目的,否则说明路径会触发程序的某个错误.

在计算机编程中,需要在编译前确定变量的类型(例如 float 或 double),变量精度会受到类型的影响,在程序输入不变的情况下,改变变量类型在某些情况下会使程序输出不同的结果. 但是,在程序运行之前很难判断所选变量类型是否符合计算的精度. Paganelli 等人[107] 提出了一种通过增加精确度验证浮点数程序稳定性的方法,证明了增加变量精度会使得结果产生微小的变化. 该方法首先分析程序中变量的类型以及用户的断言,计算出程序的最弱前置条件(weakest precondition),然后将最弱前置条件转化为 SMT 公式 F 并使用 Z3 求解 $\neg F$ 的可满足性,如果 $\neg F$ 可满足,则具体的可满足赋值可以作为程序的 1 个测试用例,作为后续证明过程的输入. 如果不可满足,便需要用户对程序进行调整.

克雷格插值(craig interpolation)方法在程序验证领域取得了不错的成绩,例如 Kroening 等人[108] 开发了基于克雷格插值的软件验证工具 Wolverine. 但当插值规模很大时,程序验证的效率可能会受到影响,因此 Pigorsch 等人[109] 提出了一种基于 SMT 的减小 interpolation 规模的算法,可以提升克雷格插值方法的效率,MathSAT 负责算法中公式的不可满足性的证明工作.

工业界的验证工具也用到了 SMT 求解器,例如,VCC[110](verifying C compiler)是 1 个低层级并发系统(low-level concurrent system)代码验证工具,集成了 SMT 求解器 Z3,可以根据程序中的注释(比如函数功能、状态断言等)验证程序的正确性. 微软使用 VCC 验证了虚拟化产品 Hyper-V 的正确性.

4.5 线性逻辑约束公式优化问题求解

4.5.1 SMT 与线性逻辑约束公式优化问题求解

SMT 在线性逻辑约束公式优化问题求解中的主要应用是:结合背景理论(例如线性实数理论),利用 SMT 求解器找到满足目标公式及约束条件的最优解. 具体来说,首先根据实际问题的约束条件和优化目标抽象出约束表达式 φ 和目标函数 H,再利用 SMT 求解器求出满足 φ 的可行解集 G,然后找出 G 中使 H 达到最大值(或最小值)的可行解,即为最优解.

4.5.2 具体示例

结合 Li 等人[111]提出的基于 SMT 的线性逻辑约束公式优化问题求解方法,举 1 个简单的例子,给定基于线性实数算术理论的公式:

$$\varphi\equiv(1\leqslant y\leqslant 3)\wedge(1\leqslant x\leqslant 3\vee x\geqslant 4), \tag{11}$$

其中,φ 的可行域表示二维坐标系上的一片区域,x 和 y 均为实数变量,分别位于坐标系的横轴和纵轴上,(x,y) 代表坐标系中的 1 个点记为 $p=(x,y)$. 目标函数为 $H=\{y,x+y\}$. 需要在满足约束条件 φ 的情况下求出使目标函数 H 达到最大值的 x 和 y,即 $(x_{\text{opt}},y_{\text{opt}})$,基于 SMT 的线性逻辑约束公式优化问题求解步骤如下:

1) 假设最优解为 $opt_{\text{T}}(\varphi)$,根据约束条件 φ 可将最优解的最小上界(under-approximation, UA)初始化为

$$UA\equiv y\leqslant-\infty\wedge x+y\leqslant-\infty, \tag{12}$$

最大上界(over-approximation, OA)初始化为

$$OA\equiv y\leqslant\infty\wedge x+y\leqslant\infty. \tag{13}$$

2) 将 $\varphi\wedge\neg UA$ 送入 SMT 求解器,求得可行解为 $(x=2)\wedge(y=2)$,记为 $p_1=(2,2)$,更新最小上界为

$$UA\equiv y\leqslant 2\wedge x+y\leqslant 4,$$

更新最大上界为

$$OA\equiv y\leqslant 3.$$

3) 将 φ 中每个原子公式中的不等号替换为等号,得到的集合记为

$$\varepsilon(\varphi)=\{l=k\mid l\leqslant k\in\varphi\},$$

例如,此时 $\varepsilon(\varphi)=\{x=1,x=3,x=4,y=1,y=3\}$. 点 p 的边界类定义为 $[p]=e\in\varepsilon(\varphi)\mid p\mid=e$. $[p_1]=\varnothing$,因为此点不接触任一边界. 提取目标函数 H 的第 1 个元素 y,判断在满足 $y\leqslant 3$ 的基础上,是否存在优于可行解 p_1 的另 1 个可行解 p_2,即:

$$S\equiv[p_1]\subset[p_2]\wedge y(p_2)\geqslant y(p_1),$$

其中,$y(p_i)$ 代表点 p_i 的 y 值. 使用 SMT 求解器判断可行解 p_2 是否存在,得到 $p_2=(3,3)$,由 $\varepsilon(\varphi)$ 可知 x 和 y 的取值都达到了各自的上界,因此,更新最优解的最小上界为

$$UA\equiv y\leqslant 3\wedge x+y\leqslant 6.$$

4) 将 $\varphi\wedge\neg UA$ 送入 SMT 求解器,求得可行解为 $(x=5)\wedge(y=3)$,记为 $p_3=(5,3)$,更新最小上界为

$$UA\equiv y\leqslant 3\wedge x+y\leqslant 8.$$

5) 提取目标函数 H 的第 2 个元素 $x+y$,在满足 $x+y\leqslant\infty$ 的基础上,使用 SMT 求解器求得优于 p_3 的可行解 $p_4=(6,3)$,重复此求解步骤,发现 $x+y$ 的值可以无限的增大,求解过程无法终止,判定此最优解为无穷大,即 $x+y$ 是无上界的,因此,更新最优解的最小上界为

$$UA\equiv y\leqslant 3,$$

此时,$UA\equiv OA$,求解结束,得到最优解为

$$opt_{\text{T}}(\varphi)\equiv y\leqslant 3.$$

因此,$x_{\text{opt}}=\infty\wedge y_{\text{opt}}\leqslant 3$.

4.5.3 SMT 在线性约束优化问题求解中的应用

Sebastiani 等人[111]提出了求解线性实数算术问题全局最优解的方法,主要思想是通过可行解测试(feasibility test)、查找关键点(critical finding)、全局检查(global checking)三个步骤来找到全局最优解. 在可行解测试中,算法利用 SMT 求解器求出符合约束条件的可行解,然后在这些可行解中找出局部最优解作为全局最优解的候选解. 在全局检查步骤中,算法利用 SMT 求解器 MathSAT 对每个候选解进行测试,检查其是否为最优解. 该算法被集成到基于 SMT 的优化求解器 OPT-MathSAT 中.

Li 等人[111]研发的 SYMBA 求解器也可以获取线性实数算术问题的全局最优解,求解步骤主要分为全局推送(GLOBALPUSH)和检测是否无界(UNBOUNDED)2 个阶段. 在 GLOBALPUSH 阶段,SYMBA 使用 Z3 判断是否存在 1 个新的可行解优于当前最优解,如果存在,调整当前最优解的最小下界,如果不存在,则说明当前最优解即为全局最优解. 在 UNBOUNDED 阶段,SYMBA 会检测当前最优解是否无界,并调整其最大上界和最小上界. SYMBA 会重复这 2 个阶段直至找到全局最优解为止.

微软研发的 νZ[112] 可以求解基于 SMT 公式的线性优化问题. νZ 求解线性实数算术问题的方法与 OPT-MathSAT 的方法类似,区别在于 νZ 使用 SMT 求解器 MaxSMT 求解约束可满足性问题(constraint-satisfaction problem)并利用 OptSMT 模块优化线性算数目标函数. 微软在 Z3 中也实现了相同的功能.

4.6 基于 SMT 的其他应用

Arkoudas 等人[19]提出了一种访问控制策略自动分析方法. 该方法使用可编程定理证明系统 Athena 实现了策略的框架表示,并把策略转化为 SMT 公式,调用 Athena 中的 SMT 求解接口 smt-solve 求解公式的可满足性,再对策略进行分析.

Malozemoff 等人[113]提出了一种分组密码加密安全性的自动分析与检测方法,该方法首先将加密步骤抽象为含有不同种类节点的有向无环图,图中不同节点代表不同的操作,边代表节点的输入或输出. 需要将图中的点和边转化为约束可满足性问题并使用 Z3 求解问题的可满足性,根据结果判断模式(mode)是有效的(valid)、解密的(decryptable)还是安全的(secure).

李舟军等人[114]对安全漏洞检测技术进行了较为全面的总结,并且介绍了 SMT 在自动化白盒模糊测试中的应用:首先利用文中提出的轻量级动态符号执行方法获取程序的执行路径,再借助 SMT 求解器对路径约束求解,用于检测程序的漏洞.

5 结论与展望

SMT 以其丰富的背景理论和高效的组合背景理论求解技术成为了可满足性问题求解领域的核心. SMT 求解器可以求解含有组合背景理论的一阶逻辑公式,直接处理一些含有高层级信息的可满足问题,成为了许多实际应用的基础.

本文力图详尽地介绍 SMT 原理、求解方法、工具及最新的应用进展. 详细地阐述了 SAT,SMT 判定算法及理论组合判定算法的实现. 根据近 11 年 SMT-COMP 竞赛的结果,比较了能够处理大规模工业化应用问题的主流 SMT 求解器的综合求解能力. 从应用的角度阐述了 SMT 在测试用例自动生成、程序缺陷检测、RTL 验证、程序分析与验证以及线性逻辑约束公式优化问题求解等领域中的具体应用.

对于 SMT 在各个领域中的实际应用而言,还存在着许多新的研究热点和难点. 例如,基于有界模型检测的程序缺陷检测受限于界限 K,如果想证明程序不违反安全属性要确保:1)程序在界限 K 内不违反安全属性;2)程序中循环的次数$\leqslant K$,那么如何验证循环次数远大于 K 的程序正确性是 1 个新的研究热点.

基于 SAT 的门级电路验证会产生状态爆炸问题,将 SMT 与 RTL 验证相结合可以很好地缓解这个问题,因此许多学者对此做了相关的研究[10,101],但如何利用 SMT 验证 RTL 电路的完全正确性也是未来研究的难点.

对于 SMT 求解器本身而言,大多数 SMT 求解器只接受无量词一阶逻辑公式作为输入,实际问题中经常会含有全称量词($\forall$),求解此类问题是目前 SMT 求解器的难点之一,仅有少数几个主流的 SMT 求解器支持带有全称量词问题的求解,比如 Z3,CVC4 等. 求解方法主要分为 2 种:1)基于模式(pattern-based)的量词实例化方法,主要是将带有全称量词的变量实例化为带有存在量词($\exists$)的变量,比如将($\forall x, x>1$)实例化为($\exists x=2, x>1$). 但是如何选取 x 的具体实例化数值是有待研究的问题. 2)基于饱和定理证明(saturation theorem proving),此方法使用了叠加演算的思想解决公式中的量词问题. 含有全称量词的一阶逻辑公式可满足性求解算法将是今后的研究热点.

参 考 文 献

[1] Johnson K, Calinescu R. Efficient re-resolution of SMT specifications for evolving software architectures [C] //Proc of the 10th Int ACM Sigsoft Conf on Quality of Software Architectures. New York: ACM, 2014: 93-102

[2] Malik S U R, Khan S U, Srinivasan S K. Modeling and analysis of state-of-the-art VM-based cloud management platforms [J]. IEEE Trans on Cloud Computing, 2013, 1(1): 50-63

[3] Cotrini C, Weghorn T, Basin D, et al. Analyzing first-order role based access control [C] //Proc of the 28th IEEE Computer Security Foundations Symp. Piscataway, NJ: IEEE, 2015: 3-17

[4] Arkoudas K, Chadha R, Chiang J. Sophisticated access control via SMT and logical frameworks [J]. ACM Trans on Information and System Security, 2014, 16(4): 1-31

[5] Voss S, Schatz B. Deployment and scheduling synthesis for mixed-critical shared-memory applications [C] //Proc of the 20th IEEE Int Conf and Workshops on the Engineering of Computer Based Systems. Piscataway, NJ: IEEE, 2013: 100-109

[6] Huang Yu, Mercer E, Mccarthy J. Proving MCAPI executions are correct using SMT [C] //Proc of the 28th IEEE Int Conf on Automated Software Engineerin. Piscataway, NJ: IEEE, 2013: 26-36

[7] Cordeiro L, Fischer B, Marques-Silva J. SMT-based bounded model checking for embedded ANSI-C software [J]. IEEE Trans on Software Engineering, 2012, 38(4): 957-974

[8] Barnat J, Bauch P, Havel V. Model checking parallel programs with inputs [C] //Proc of the 22nd Euromicro Int Conf on Parallel, Distributed and Network-Based Processing (PDP). Piscataway, NJ: IEEE, 2014: 756-759

[9] Liu Leyuan, Kong Werqiang, Ando T, et al. A survey of acceleration techniques for SMT-based bounded model checking [C] //Proc of 2013 Int Conf on Computer Sciences and Applications (CSA). Piscataway, NJ: IEEE, 2013: 554-559

[10] Kunapareddy S, Turaga S D, Sajjan S S T M. Comparision between LPSAT and SMT for RTL verification [C] //Proc of Int Conf on Circuit, Power and Computing Technologies. Piscataway, NJ: IEEE, 2015

[11] Sebastiani R, Tomasi S. Optimization in SMT with LA(ℚ) cost functions [G] //LNAI 7364: Automated Reasoning. Berlin: Springer, 2012: 484-498

[12] Chen Li, Wang Yongji, Wu Jingzheng, et al. Rate-Monotonic optimal design based on tree-like linear programming search [J]. Journal of Software, 2015, 26(12): 3223-3241 (in Chinese)
(**陈力, 王永吉, 吴敬征, 等. 基于树状线性规划搜索的单调速率优化设计[J]. 软件学报**, 2015, 26(12): 3223-3241)

[13] Blackham B, Liffiton M, Heiser G. Trickle: Automated infeasible path detection using all minimal unsatisfiable subsets [C] //Proc of Real-Time and Embedded Technology and Applications Symp. Piscataway, NJ: IEEE, 2014: 169-178

[14] Eldib H, Wang Chao, Taha M, et al. Quantitative masking strength: Quantifying the power side-channel resistance of software code [J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(10): 1558-1568

[15] Leroy X. Formal verification of a realistic compiler [J]. Communications of the ACM, 2009, 52(7): 107-115

[16] Klein G. Operating system verification—An overview [J]. Sadhana, 2009, 34(1): 27-69

[17] Cook S A. The complexity of theorem-proving procedures [C] //Proc of the 3rd Annual ACM Symp on Theory of Computing. New York: ACM, 1971: 151-158

[18] Cousot P, Cousot R, Feret J, et al. The ASTRéE analyzer [G] //LNCS 3444: Programming Languages and Systems. Berlin, Springer, 2005: 21-30

[19] Arkoudas K, Loeb S, Chadha R, et al. Automated policy analysis [C] //Proc of 2012 IEEE Int Symp on Policies for Distributed Systems and Networks(POLICY). Piscataway, NJ: IEEE, 2012

[20] Nuzzo P, Puggelli A, Seshia S A, et al. CalCS: SMT solving for non-linear convex constraints [C] //Proc of the 2010 Conf on Formal Methods in Computer-Aided Design. Piscataway, NJ: IEEE, 2010: 71-80

[21] Chimisliu V, Wotawa F. Category partition method and satisfiability modulo theories for test case generation [C] //Proc of the 7th Int Workshop on Automation of Software Test Piscataway, NJ: IEEE, 2012: 64-70

[22] Riener H, Bloem R, Fey G. Test case generation from mutants using model checking techniques [C] //Proc of the 4th IEEE Int Conf on Software Testing, Verification and Validation Workshops (ICSTW). Piscataway, NJ: IEEE, 2011: 388-397

[23] JoöBstl E, Weiglhofer M, Aichernig B K, et al. When bdds fail: Conformance testing with symbolic execution and SMT solving [C] //Proc of the 3rd Int Conf on Software Testing, Verification and Validation (ICST). Piscataway, NJ: IEEE, 2010: 479-488

[24] Ansótegui C, Bofill M, Manyà F, et al. Building automated theorem provers for infinitely-valued logics with satisfiability modulo theory solvers [C] //Proc of the 42nd IEEE Int Symp on Multiple-Valued Logic (ISMVL). Piscataway, NJ: IEEE, 2012: 25-30

[25] Shostak R E. An algorithm for reasoning about equality [J]. Communications of the ACM, 1978, 21(2): 583-585

[26] Shostak R E. Deciding combinations of theories [J]. Journal of the ACM, 1984, 31(1): 209-222

[27] Shostak R E. A practical decision procedure for arithmetic with function symbols [J]. Journal of the ACM, 1979, 26(2): 351-360

[28] Armando A, Giunchiglia E. Embedding complex decision procedures inside an interactive theorem prover [J]. Annals of Mathematics & Artificial Intelligence, 1993, 8(3/4): 475-502

[29] Armando A, Castellini C, Giunchiglia E. SAT-based procedures for temporal reasoning [G] //LNAI 1809: Proc of the 5th European Conf on Planning: Recent Advances in AI Planning. Berlin: Springer, 2000: 97-108

[30] Bryant R E, German S, Velev M N. Exploiting positive equality in a logic of equality with uninterpreted Functions [C] //Proc of the 11th Int Conf on Computer Aided Verification. Berlin: Springer, 1999: 470-482

[31] Giunchiglia F, Sebastiani R. Building decision procedures for modal logics from propositional decision procedures: The case study of modal $K(m)$ [J]. Information & Computation, 2000, 162(1): 158-178

[32] Zhang Jian. Specification analysis and test data generation by solving Boolean combinations of numeric constraints [C] //Proc of the 1st Asia-Pacific Conf on Quality Software. Piscataway, NJ: IEEE, 2000: 267-274

[33] Malik S, Zhang Lintao. Boolean satisfiability from theoretical hardness to practical success [J]. Communications of the ACM, 2009, 52(8): 76-82

[34] De Moura L, Bjørner N. Z3: An efficient SMT solver [G] //LNCS 4936: Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008: 337-340

[35] Barrett C, Deters M, Conway C L, et al. CVC4 [G] //LNCS 6806: Proc of the 23rd Int Conf on Computer Aided Verification. Berlin: Springer, 2011: 171-177

[36] Dutertre B. Yices2. 2 [G] //LNCS 8559: Proc of the 16th Int Conf on Computer Aided Verification. Berlin: Springer, 2014: 737-744

[37] Bruttomesso R, Cimatti A, Franzén A, et al. The mathsat4 smt solver [G] //LNCS 5123: Proc of the 20th Int Conf on Computer Aided Verification. Berlin: Springer, 2008: 299-303

[38] Brummayer R, Biere A. Boolector: An efficient SMT solver for bit-vectors and arrays [G] //LNCS 5505: Proc of Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2009: 174-177

[39] Godefroid P, De Halleux P, Nori A V, et al. Automating software testing using program analysis [J]. IEEE Software, 2008, 25(5): 30-37

[40] Henzinger T A, Jhala R, Majumdar R, et al. Lazy abstraction [J]. ACM SIGPLAN Notices, 2002, 37(1): 58-70

[41] Wang Jianxin, Guan Lina, Jiang Guohong. A survey of algorithms for SAT problem [J]. Computing Technology and Automation, 2009, 28(4): 138-143 (in Chinese)
(**王建新, 管利娜, 江国红. 可满足性问题的研究综述**[J]. **计算技术与自动化**, 2009, 28(4): 138-143)

[42] Sheini H M, Sakallah K A. From propositional satisfiability to satisfiability modulo theories [G] //LNCS 4121: Proc of the 9th Int Conf Seattle. Berlin: Springer, 2006

[43] Moura L, Dutertre B, Shankar N. A tutorial on satisfiability modulo yheories [G] //LNCS 4590: Proc of the 19th Int Conf on Computer Aided Verification. Berlin: Springer, 2007: 20-36

[44] Jin Jiwei, Ma Feifei, Zhang Jian. Brief introduction to SMT solving [J]. Journal of Frontiers of Computer Science & Technology, 2015, 9(7): 769-780 (in Chinese)
(**金继伟, 马菲菲, 张健**. SMT **求解技术简述**[J]. **计算机科学与探索**, 2015, 9(7): 769-780)

[45] Singer D. Parallel Resolution of the Satisfiability Problem: A Survey [M]. New York: John Wiley & Sons, 2006: 123-148

[46] Hansen P, Jaumard B. Algorithms for the maximum satisfiability problem [J]. Computing, 1990, 44(4): 279-303

[47] Bistarelli S, Montanari U, Rossi F, et al. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison [J]. Constraints, 1999, 4(3): 199-240

[48] Mahajan M, Raman V. Parameterizing above guaranteed values: Maxsat and maxcut [J]. Journal of Algorithms, 1999, 31(2): 335-354

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

[50] Davis M, Logemann G, Loveland D. A machine program for theorem-proving [J]. Communications of the ACM, 1962, 5(7): 394-397

[51] Prestwich S. Lynce I: Local search for unsatisfiability [G] //LNCS 4121: Proc of Theory and Applications of Satisfiability Testing-SAT 2006. Berlin: Springer, 2006: 283-296

[52] Audemard G, Simon L. GUNSAT: A greedy local search algorithm for unsatisfiability [C] //Proc of 2007 Int Joint Conf on Artificial Intelligence. San Francisco: Morgan Kaufmann, 2007: 2256-2261

[53] Xu Daoyun Liu Changyun. DPLL algorithm with literal renaming strategy [J]. Journal of Frontiers of Computer Science & Technology, 2007, 1(1): 116-125 (in Chinese)
(**许道云, 刘长云. 带文字改名策略的** DPLL **算法**[J]. **计算机科学与探索**, 2007, 1(1): 116-125)

[54] Nieuwenhuis R, Oliveras A, Rodríguez-Carbonell E, et al. Challenges in satisfiability modulo theories [G] //LNCS 4533: Proc of Term Rewriting and Applications. Berlin: Springer, 2007: 2-18

[55] Barrett C, Stump A, Tinelli C. The satisfiability modulo theories library (smt-lib) [EB/OL]. 2010 [2016-04-03]. http://www.SMT-LIB.org

[56] Mccarthy J. Towards a Mathematical Science of Computation [M]. Berlin: Springer, 1993: 35-56

[57] Barrett C, Tinelli C. Cvc3 [G] //LNCS 4590: Proc of the 19th Int Conf on Computer Aided Verification. Berlin: Springer, 2007: 298-302

[58] Nelson G, Oppen D C. Simplification by cooperating decision procedures [J]. ACM Trans on Programming Language System, 1979, 1(2): 245-257

[59] Bozzano M, Bruttomesso R, Cimatti A, et al. Efficient satisfiability modulo theories via delayed theory combination [G] //LNCS 3676: Proc of Computer Aided Verification. Berlin: Springer, 2005: 335-349

[60] Nelson G, Oppen D C. Fast decision procedures based on congruence closure [J]. Journal of the ACM, 1980, 27(2): 356-364

[61] Huang Zhuo, Zhang Jian. Generating SAT instances from first-order formulas [J]. Journal of Software, 2005, 16(3): 327-335

[62] Seshia S A, Lahiri S K, Bryant R E. A hybrid SAT-based decision procedure for separation logic with uninterpreted functions [C] //Proc of the 40th Annual Design Automation Conf. New York: ACM, 2003: 425-430

[63] Bryant R E, Lahiri S K, Seshia S A. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions [G] //LNCS 2404: Proc of the 2002 Int Conf on Computer Aided Verification. Berlin: Springer, 2002: 78-92

[64] Pnueli A, Rodeh Y, Shtrichman O, et al. Deciding equality formulas by small domains instantiations [G] //LNCS 1633: Proc of the 1999 Int Conf on Computer Aided Verification. Berlin: Springer, 1999: 455-469

[65] Talupur M, Sinha N, Strichman O, et al. Range allocation for separation logic [G] //LNCS 3114: Proc of the 2004 Int Conf on Computer Aided Verification. Berlin: Springer, 2004: 148-161

[66] Goel A, Sajid K, Zhou H, et al. BDD based procedures for a theory of equality with uninterpreted functions [C] //Proc of the 1998 Int Conf on Computer Aided Verification. Berlin: Springer, 1998: 244-255

[67] Barrett C W, Dill D L, Stump A. Checking satisfiability of first-order formulas by incremental translation to SAT [C] //Proc of the 2002 Int Conf on Computer Aided Verification. Berlin: Springer, 2002: 236-249

[68] Wolfman S A, Weld D S. The LPSAT engine & its application to resource planning [C] //Proc of the 16th Int Joint Conf on Artificial Intelligence. San Francisco: Morgan Kaufmann, 1999: 310-317

[69] Ball T, Cook B, Lahiri S K, et al. Zapato: Automatic theorem proving for predicate abstraction refinement [G] //LNCS 3114: Proc of Computer Aided Verification. Berlin: Springer, 2004: 457-461

[70] Bruno D, Leonardo D M. A fast linear-arithmetic solver for DPLL(T) [G] //LNCS 4144: Proc of Computer Aided Verification. Berlin: Springer, 2006: 81-94

[71] Ganzinger H, Hagen G, Nieuwenhuis R, et al. DPLL(T): Fast decision procedures [G] //LNCS 3114: Proc of Computer Aided Verification. Berlin: Springer, 2004: 175-188

[72] Bozzano M, Bruttomesso R, Cimatti A, et al. Efficient theory combination via Boolean search [J]. Information and Computation, 2006, 204(10): 1493-1525

[73] Sebastiani R. Lazy satisfiability modulo theories [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2007, 3(9): 141-224

[74] Nieuwenhuis R, Oliveras A, Tinelli C. Solving SAT and SAT modulo theories: From an abstract davis-putnam-logemann-loveland procedure to DPLL (T) [J]. Journal of the ACM, 2006, 53(6): 937-977

[75] Flanagan C, Joshi R, Ou X, et al. Theorem proving using lazy proof explication [G] //LNCS 2725: Proc of Computer Aided Verification. Berlin: Springer, 2003: 355-367

[76] Bozzano M, Bruttomesso R, Cimatti A, et al. An incremental and layered procedure for the satisfiability of linear arithmetic logic [G] //LNCS 3440: Proc of Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2005: 317-333

[77] Jha S, Limaye R, Seshia S A. Beaver: Engineering an efficient smt solver for bit-vector arithmetic [G] //LNCS 5643: Proc of Computer Aided Verification. Berlin: Springer, 2009: 668-674

[78] Armando A, Castellini C, Giunchiglia E, et al. A SAT-based decision procedure for the Boolean combination of difference constraints [G] //LNCS 3542: Proc of Theory and Applications of Satisfiability Testing. Berlin: Springer, 2005: 16-29

[79] Bofill M, Nieuwenhuis R, Oliveras A, et al. The barcelogic SMT solver [G] //LNCS 5123: Proc of the 20th Int Conf on Computer Aided Verification. Berlin: Springer, 2008: 294-298

[80] Bouton T, De Oliveira D C B, Déharbe D, et al. VeriT: An open, trustable and efficient SMT-solver [G] //LNAI 5663: Proc of 2009 Int Conf on Automated Deduction. Berlin: Springer, 2009: 151-156

[81] Lahiri S, Qadeer S. Back to the future: Revisiting precise program verification using SMT solvers [J]. ACM SIGPLAN Notices, 2008, 43(1): 171-182

[82] Gulavani B S, Henzinger T A, Kannan Y, et al. SYNERGY: A new algorithm for property checking [C] //Proc of the 14th ACM SIGSOFT Int Symp on Foundations of Software Engineering. New York: ACM, 2006: 117-127

[83] Ball T, Rajamani S K. The SLAM project: Debugging system software via static analysis [C] //Proc of ACM SIGPLAN Notices. New York: ACM, 2002: 1-3

[84] Costa M, Crowcroft J, Castro M, et al. Vigilante: End-to-end containment of internet worms [C] //Proc of 2005 ACM SIGOPS Operating Systems Review. New York: ACM, 2005: 133-147

[85] Godefroid P, Levin M Y, Molnar D. SAGE: Whitebox fuzzing for security testing [J]. Communications of the ACM, 2012, 55(3): 40-44

[86] Anand S, Păsăreanu C S, Visser W. JPF-SE: A symbolic execution extension to Java pathfinder [G] //LNCS 4424: Proc of Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2007: 134-138

[87] Arcaini P, Gargantini A, Vavassori P. Generating tests for detecting faults in feature models [C] //Proc of the 8th IEEE Int Conf on Software Testing, Verification and Validation. Piscataway, NJ: IEEE, 2015

[88] Lin Huimin, Zhang Wenhui. Model checking: Theories, technigues and applications [J]. Acta Electronica Sinica, 2002, 30(12): 1907-1912 (in Chinese)
(林惠民, 张文辉. 模型检测: 理论, 方法与应用[J]. 电子学报, 2002, 30(12): 1907-1912)

[89] Ramalho M, Freitas M, Sousa F, et al. SMT-based bounded model checking of C++ programs [C] //Proc of the 20th IEEE Int Conf and Workshops on Engineering of Computer Based Systems(ECBS). Piscataway, NJ: IEEE, 2013: 147-156

[90] Cordeiro L. SMT-based bounded model checking of multi-threaded software in embedded systems [C] //Proc of the 32nd ACM/IEEE Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2011: 373-376

[91] Cordeiro L, Fischer B. Verifying multi-threaded software using smt-based context-bounded model checking [C] //Proc of the 33rd Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2011: 331-340

[92] Mueller F. A library implementation of POSIX threads under UNIX [C] //Proc of the USENIX Conf. Berkeley, CA: USENIX Association, 1993: 29-42

[93] Pereira P A, Albuquerque H F, Marques H M, et al. Verifying CUDA programs using SMT-based context-bounded model checking [EB/OL]. [2016-04-02]. http://svlab.hussamaismail.eti.br/papers/sac2016.pdf

[94] Cheng J, Grossman M, Mckercher T. Professional Cuda C Programming [M]. New York: John Wiley & Sons, 2014

[95] Morse J. Expressive and efficient bounded model checking of concurrent software [D]. Southampton: University of Southampton, 2015

[96] Ball T, Rajamani S K. Automatically validating temporal safety properties of interfaces [G] //LNCS 2057: Proc of the 8th Int SPIN Workshop on Model Checking of Software. Berlin: Springer, 2001: 103-122

[97] Lingappan L, Ravi S, Jha N K. Satisfiability-based test generation for nonseparable RTL controller-datapath circuits [J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2006, 25(3): 544-557

[98] Duraira V, Kalla P. Exploiting hypergraph partitioning for efficient Boolean satisfiability [C] //Proc of the 9th IEEE Int High-Level Design Validation and Test Workshop. Piscataway, NJ: IEEE, 2004: 141-146

[99] Kroening D, Seshia S A. Formal verification at higher levels of abstraction [C] //Proc of IEEE/ACM Int Conf on Computer-Aided Design. Piscataway, NJ: IEEE, 2007: 572-578

[100] Clarke E, Grumberg O, Jha S, et al. Counterexample-guided abstraction refinement [G] //LNCS 1855: Proc of Computer Aided Verification. Berlin: Springer, 2000: 154-169

[101] Puri P, Bradley M S H. SI-SMART: Functional test generation for RTL circuits using loop abstraction and learning recurrence relationships [C] //Proc of the 33rd IEEE Int Conf on Computer Design. Piscataway, NJ: IEEE, 2015: 38-45

[102] Brady B, Bryant R E, Seshia S, et al. ATLAS: Automatic term-level abstraction of RTL designs [C] //Proc of the 8th IEEE/ACM Int Conf on Formal Methods and Models for Codesign. Piscataway, NJ: IEEE, 2010: 31-40

[103] Zhao Yanni, Bian Jinian, Deng Shujun. Constraints decomposition for RTL verification by SMT [J]. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(2): 234-239 (in Chinese)
(赵燕妮, 边计年, 邓澍军. 利用 SMT 约束分解方法求解 RTL 可满足性问题[J]. 计算机辅助设计与图形学学报, 2010, 22(2): 234-239)

[104] Gent K, Hsiao M S. Functional test generation at the RTL using swarm intelligence and bounded model checking [C] //Proc of the 22nd Asian Test Symp. Piscataway, NJ: IEEE, 2013: 233-238

[105] Liu L, Vasudevan S. Efficient validation input generation in RTL by hybridized source code analysis [C] //Proc of Design, Automation and Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2011

[106] He Yanxiang, Wu Wei, Chen Yong, et al. Path sensitive program verification based on SMT solvers [J]. Journal of Software, 2012, 23(10): 2655-2664 (in Chinese)
(何炎祥, 吴伟, 陈勇, 等. 基于 SMT 求解器的路径敏感程序验证[J]. 软件学报, 2012, 23(10): 2655-2664)

[107] Paganelli G, Ahrendt W. Verifying (in-) stability in floating-point programs by increasing precision, using SMT solving [C] //Proc of the 15th Int Symp on Symbolic and Numeric Algorithms for Scientific Computing. Piscataway, NJ: IEEE, 2013: 209-216

[108] Kroening D, Weissenbacher G. Interpolation-based software berification with WOLVERINE [G] //LNCS 6806: Proc of the 23rd Int Conf on Computer Aided Verification. Berlin: Springer, 2011: 573-578

[109] Pigorsch F, Scholl C. Lemma localization: A practical method for downsizing SMT-interpolants [C] //Proc of the Conf on Design, Automation and Test in Europe. San Jose, CA: EDA Consortium, 2013: 1405-1410

[110] Cohen E, Dahlweid M, Hillebrand M, et al. VCC: A practical system for verifying concurrent C [G] //LNCS 5674: Proc of Theorem Proving in Higher Order Logics. Berlin: Springer, 2009: 23-42

[111] Li Yi, Albarghouthi A, Kincaid Z, et al. Symbolic optimization with SMT solvers [C] //Proc of the 41st ACM SIGPLAN-SIGACT Symp on Principles of Programming Languages. New York: ACM, 2014: 607-618

[112] Bjørner N, Phan A-D, Fleckenstein L. νZ-an optimizing SMT solver [G] //LNCS 9035: Proc of Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2015: 194-199

[113] Malozemoff A J, Katz J, Green M D. Automated analysis and synthesis of block-cipher modes of operation [C] //Proc of the 27th IEEE Computer Security Foundations Symp. Piscataway, NJ: IEEE, 2014: 140-152

[114] Li Zhoujun, Zhang Junxian, Liao Xiangke, et al. Survey of software vulnerability detection techniques [J]. Chinese Journal of Computers, 2015, 38(4): 717-732 (in Chinese)
(李舟军, 张俊贤, 廖湘科, 等. 软件安全漏洞检测技术 [J]. 计算机学报, 2015, 38(4): 717-732)

Wang Chong, born in 1991. PhD candidate. Student member of CCF. His main research interests include satisfiability modulo theories, multicore scheduling algorithm, bounded model checking.

Lü Yinrun, born in 1991. PhD candidate. His main research interests include satisfiability modulo theories, optimization algorithm.

Chen Li, born in 1989. PhD candidate. His main research interests include satisfiability modulo theories, real-time system, optimization algorithm.

Wang Xiuli, born in 1997. Associate professor and PhD supervisor. His main research interests include information security and game theory.

Wang Yongji, born in 1962. Professor and PhD supervisor. Senior member of CCF. He is the winner of the 2002 One-Hundred-Talent People Program sponsored by the Chinese Academy of Sciences. His main research interests include computer-controlled real-time systems, network optimization theory, intelligent software engineering, nonlinear optimization theory, real-time hybrid control theory, real-time embedded operating system, etc.