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.

ISSN 1673-9418　CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2015/09(07)-0769-12
doi: 10.3778/j.issn.1673-9418.1405041

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

SMT求解技术简述*

金继伟+,马菲菲,张　健
中国科学院 软件研究所,北京 100190

Brief Introduction to SMT Solving*

JIN Jiwei+, MA Feifei, ZHANG Jian
Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
+ Corresponding author: E-mail: jinjw@ios.ac.cn

JIN Jiwei, MA Feifei, ZHANG Jian. Brief introduction to SMT solving. Journal of Frontiers of Computer Science and Technology, 2015, 9(7): 769-780.

Abstract: SMT is the problem of deciding the satisfiability of a first order formula with respect to some theory formulas. It is being recognized as increasingly important due to its applications in different communities, in particular in formal verification, program analysis and software testing. This paper provides a brief overview of SMT and its theories. Then this paper introduces some approaches aiming to improve the efficiency of SMT solving, including eager and lazy approaches and optimum technique which have been proposed in the last ten years. This paper also introduces some state-of-the-art SMT solvers, including Z3, Yices and CVC3/CVC4. Finally, this paper gives a preliminary prospect on the research trends of SMT, which include SMT with quantifier, optimization problems subject to SMT and volume computation for SMT.

Key words: satisfiability modulo theories (SMT); DPLL(T); solver

摘　要: SMT问题是在特定理论下判定一阶逻辑公式可满足性问题。它在很多领域,尤其是形式验证、程序分析、软件测试等领域,都有重要的应用。介绍了SMT问题的基本概念、相关定义以及目前的主流理论。近年来出现了很多提高SMT求解效率的技术,着重介绍并分析了这些技术,包括积极类算法、惰性算法及其优化技术等。介绍了目前的主流求解器和它们各自的特点,包括Z3、Yices、CVC3/CVC4等。对SMT求解技术的前景进行了展望,量词的处理、优化问题和解空间大小的计算等尤其值得关注。

* The National Natural Science Foundation of China under Grant No. 61100064 (国家自然科学基金).
Received 2014-05, Accepted 2015-05.
CNKI网络优先出版:2015-06-16, http://www.cnki.net/kcms/detail/11.5602.TP.20150616.1651.001.html

关键词:可满足性模理论(SMT);DPLL(T);求解器
文献标志码:A **中图分类号**:TP30

1 引言

SAT(satisfiability)问题指的是命题逻辑公式的可满足性问题。随着研究的深入,人们发现SAT在表达能力上有很大的局限性,许多应用用SAT进行编码并不是很明智的选择,它们需要比SAT更强的表达方式。在这种形势下,将SAT问题扩展为SMT,经过扩展,SMT能比SAT更好地表达一些人工智能和形式化方法领域内的问题,比如在资源规划、时序推理、编译器优化等很多方面用到了SMT。

SMT的全称是Satisfiability Modulo Theories,可被翻译为“可满足性模理论”、“多理论下的可满足性问题”或者“特定(背景)理论下的可满足性问题”,其判定算法被称为SMT求解器。简单地说,一个SMT公式是结合了理论背景的逻辑公式,其中的命题变量可以代表理论公式。对于SMT的研究起源于20世纪70年代末80年代初,当时的一些学者为形式化方法设计了一些判定算法,这些算法可以看作最早的SMT求解器[1-5]。到了20世纪90年代,人们开始研究能处理大规模工业界问题的SMT求解技术[6-12]。最近几年在工业界和学术界,这类技术均得到了迅猛的发展。而SMT求解器也被集成到一些大型工具中,比如HOL/Isabelle、ESC/Java 2、ACL2、UCLID、BLAST、Eureka、CUTE和PEX等。

本文主要介绍了SMT的求解技术,并对这些技术的发展进行了探讨。此外还介绍了目前的主流SMT求解器以及它们的特点。这些求解器大都是用于处理工业上的大型例子,可以求解大规模的问题实例。最后探讨了目前SMT求解中的难点以及该领域的发展方向。

本文组织结构如下:第2章介绍了SMT和它的各种理论的相关定义;第3章介绍了SMT的判定算法,主要是惰性算法,内容包括基本的DPLL算法,各种理论判定方法,以及一些提高效率的技术;第4章讨论了目前主要的求解器和它们的特点;第5章分析了目前求解SMT的难点,以及SMT技术的发展方向;最后总结全文。

2 基本概念和定义

SMT问题是判定SMT公式是否可满足的问题。SMT公式是结合了背景理论的一阶逻辑公式,这些理论包括一些数学理论和计算机领域内用到的数据结构理论等。这种结合的体现是在SMT公式中,命题变量有时会被解释为背景理论公式。例如下面的公式。

例1 $x+y<3\wedge y>2$

这是一个SMT公式,它的逻辑形式是 $A\wedge B$,其中命题变量 A 和 B 分别被解释为数学公式 $x+y<3$ 和 $y>2$。

给定一个SMT公式,在通常的逻辑解释和背景理论解释下,如果存在一个赋值使该公式为真,那么称该公式是可满足的;否则称该公式是不可满足的。这样的赋值被称为模型。在算法中,用返回值为真代表输入的公式可满足,用返回值为假代表它不可满足。例如公式 $x<3\wedge x>4$ 是不可满足的,因为根据 $\wedge$ 的解释,要求 $x<3$ 和 $x>4$ 必须同时有解,而根据数学演算(背景理论知识),这是不可能的。而公式 $x<5\wedge x>1$ 是可满足的,它的一个模型是 $\{x=2\}$。

命题变量或者命题变量的否定被称为文字,由文字构成的形如 $l_1\vee l_2\vee\cdots\vee l_m$ 的公式被称为子句(l_i 是文字)。CNF(conjunctive normal form)是命题公式的合取范式形式,也就是形如 $A_1\wedge A_2\wedge\cdots\wedge A_n$ 的公式,其中每一个 A_i 是一个子句。如果公式之间是合取关系,有时也用大括号来表示,例如 $A_1\wedge A_2\wedge\cdots\wedge A_n$ 可以被表示为 $\{A_1,A_2,\cdots,A_n\}$。

对命题变量的赋值有时也被称为模型,通常用大括号括起的文字来表示。其中赋值成真的变量是变量本身,赋值成假的变量是它的否定,例如赋值 $a=1,b=0,c=1$ 记作 $\{a,\neg b,c\}$。模型是常用的术语,此处的模型与SMT公式的模型的意义有所不同,根据上下文能够区分开来。

由于使用了背景理论,SMT有着比SAT更灵活的表达方式,可以方便地表示一些工业上和学术上

的问题。目前的SMT求解器能处理的理论主要有一些数学理论、数据结构理论和未解释函数。如果细分的话，主要有以下几个。

（1）未解释函数（uninterpreted function，UF）。它主要包含一些没有经过解释的函数符号和它们的参数，比如 $f(x), g(y+1, z)$ 等。通常这个理论带有等号，因此它也被称为EUF。下面是一个EUF的例子。

例2 $\{a=b, b=f(c), \neg(g(a)=g(f(c)))\}$

在这个例子中，大括号里的公式属于合取关系（如前文所述）。这个例子是不可满足的。因为如果将 $\neg(g(a)=g(f(c)))$ 中的 a 用 b 取代，再将 b 用 $f(c)$ 取代，即可得到 $\neg(g(f(c))=g(f(c)))$，这个公式显然是不成立的。

（2）线性实数演算（linear real arithmetic，LRA）和线性整数演算（linear integer arithmetic，LIA）。这两类理论的公式形如：

$$a_1x_1+a_2x_2+\cdots+a_nx_n \odot c$$

这里 $\odot$ 可以是=、<=、>=、$\neq$，c 是一个常数，$a_1a_2\cdots a_n$ 是常系数，$x_1x_2\cdots x_n$ 是实数变量（在LRA中）或者整数变量（在LIA中）。举例如下。

例3 $\{x-2y=3, 4y-2z<9, x-z>7\}$

这组公式可通过变量替换转化为 $\{x-2y=3, 4y-2z<9, 2y-z>4\}$，并进一步变换为 $\{x-2y=3, 8<4y-2z<9\}$。在LRA理论中，$8<4y-2z<9$ 是有解的，因而这个例子是可满足的。但 $8<4y-2z<9$ 没有整数解，因而在LIA理论中，这个例子是不可满足的。这两类理论统称为LA。

（3）非线性实数演算（non-linear real arithmetic，NRA）和非线性整数演算（non-linear integer arithmetic，NIA）。公式的形式为任意的数学表达式。

（4）实数差分逻辑（difference logic over the reals，RDL）和整数差分逻辑（difference logic over the integers，IDL）。它的一般形式为：

$$x-y \odot c$$

这里 $\odot$ 可以是=、<=、>=、$\neq$，c 是一个整数或者布尔值，x 和 y 可以是实数（在RDL中）或者整数（在IDL中）。这两类理论总称为DL。下面是一个不可满足的DL理论的例子。

例4 $\{x-y=3, y-z<5, \neg(x-z<9)\}$

此例子可通过变量替换转化为 $\{x-y=3, y-z<5, \neg(y-z<6)\}$，进一步可转化为 $\{x-y=3, y-z<5, y-z>=6\}$。$y-z<5$ 与 $y-z>=6$ 相冲突，因而无解。

（5）数组（arrays）和位向量（bit vector，BV）。这两个理论处理计算机中的数据结构，用于处理数组和位向量操作。下面是一个BV例子，是不可满足的。

例5 $(\boldsymbol{w}[31:0])>>16!=O16{:}w[31:16]$

此例子的含义是一个32位的位向量 $\boldsymbol{w}$，向右移动16位后，得到的结果向量和 $\boldsymbol{w}$ 的高16位不同。这是错误的，该公式不可满足。

为了更好地表示应用中的问题，SMT公式经常包含两种以上的理论。例如UFBV指的是包含未解释函数和位向量的公式，UFLIA指的是包含未解释函数和线性整数演算的公式。

3 算法

SMT的判定算法大致可分为积极（eager）类算法和惰性（lazy）类算法。前者是将SMT公式转换为可满足性等价的SAT公式，然后求解该SAT公式；后者是结合了SAT求解和理论求解，先得出一个对命题变量的赋值，然后再判断该赋值是否理论一致。其中SAT求解用的是DPLL算法（Davis-Putnam-Logemann-Loveland algorithm）。下面简单介绍一下积极类算法，然后着重介绍惰性算法，包括DPLL算法和各种理论求解算法。

3.1 积极类算法

积极算法是早期的SMT求解器采用的算法，它是将SMT公式转换成一个CNF型的命题公式，然后用SAT求解器求解[13-16]。这种方法的好处是它可以利用高效的SAT求解器，求解效率依赖于SAT求解器的效率。早期SAT求解技术取得的重大进步推动了积极算法的发展。对于不同的理论，积极算法需要用不同的转换方法和改进方法，这样有助于提高转换和求解的效率。例如对于EUF用到了per-constraint编码[14]，对于DL通常用small-domain编码[9,17]等。

下面以small-domain编码为例来说明积极算法。这种方式首先计算出每个数字变量的值域大小，假设为 N，然后用 N 个布尔变量或者 $\mathrm{lb}N$ 个布

尔变量来表示这个数字变量。前一个方法有助于SAT求解器进行传播,求解速度快,不过生成的公式比较大;后者是模拟计算机硬件的数学演算,生成的公式规模不大,不过由于没有很好的结构特征,SAT求解器的求解比较困难。

积极算法的正确性依赖于编码的正确性和SAT求解器的正确性,而且对于一些大的例子来说,编码成CNF公式很容易引起组合爆炸,也就是公式的长度指数级增长。因此这类方法在实际应用中效果不是很好,解决不了很大的工业界例子,目前主流的SMT求解器大都采用惰性类算法。

3.2 惰性算法的基本过程

惰性算法是目前采用的主流方法,也是被研究得最多的算法[8,18-22]。这种算法是先不考虑理论,将一个SMT公式看作SAT公式求解,然后再用理论求解器判定SAT公式的解所表示的理论公式是否一致。目前多数大型的SMT求解器采用了惰性算法。可以看出这类算法结合了SAT求解和理论求解。在介绍该算法之前,首先介绍一个术语。前文说过SMT是逻辑公式和背景理论公式的结合,称一个SMT公式中的逻辑公式为它的命题框架,比如下列的SMT公式。

例6 $x<2\wedge x+y>3\rightarrow y>2$

这个公式的命题框架是 $A\wedge B\rightarrow C$,其中 A、B、C 分别代表理论公式 $x<2$、$x+y>3$ 和 $y>2$ 。对于一个SMT公式 F ,用 $PS(F)$ 来表示它的命题框架。

首先介绍判定命题公式的DPLL算法。

算法1 DPLL算法

输入:一个CNF公式 F

输出:真,假

1. 对 F 进行预处理,如果公式为假,返回假
2. 选择下一个没有被赋值的变量进行赋值
3. 根据赋值进行推导
4. 如果推导出公式为真,则返回真
5. 如果推导出冲突,则
6. 分析冲突,如果能回溯,则回溯
7. 如果不能回溯,则返回假
8. 如果没有推导出冲突,返回2

第1行的预处理是检查公式的结构,看看是否存在冲突的子句,这个检查消耗的代价很小。如果在预处理阶段没有发现冲突,则按照一定的策略对公式中的变量进行赋值。其中第3行的推导,主要指这样一个操作:对某个变量赋值后,某些子句中的变量就会被强迫赋值,例如子句 $a\vee b$,如果给 a 赋值成0(假),那么 b 就必须被赋值成1,这个步骤可以不断进行下去。推导的作用主要是找出这样的文字,并扩充赋值。第5行中的冲突指的是推导出不可能的赋值,比如一个公式中存在这样两个子句 $a\vee b$ 和 $a\vee\neg b$,如果给 a 赋值成0,那么前一个子句要求 b 为1,后一个子句要求 b 为0,这时就产生了冲突。产生冲突后,用回溯算法返回前面的变量,重新进行赋值。如果没有冲突,而且公式已经被满足,这时就可以返回真,否则要对下一个变量赋值。

惰性算法通常又被称为DPLL(T)算法,T代表理论求解器。这个算法的一般形式是:

算法2 DPLL(T)算法

输入:一个SMT公式 F

输出:真,假

1. 得到 F 的命题框架 $PS(F)$
2. 如果 $PS(F)$ 是不可满足的,返回假
3. 否则对于 $PS(F)$ 的每一个模型 M ,检查 M 所代表的理论是否一致
4. 如果存在一个模型是理论一致的,返回真
5. 如果所有的模型都不是理论一致的,返回假

上面算法中的模型 M 指的是 $PS(F)$ 的一个解。第2行用到的是SAT求解器,第3行中取得模型的方法也是用SAT求解器,检查模型的理论一致性用相应的理论求解器。SAT求解器一般用到DPLL算法。一个逻辑公式的解(模型)可以看作一组文字的合取,检查模型的理论一致性就是检查这组文字代表的理论公式是否有解。理论一致指的是这组文字代表的理论公式有解。下面用一个例子来说明算法的执行过程。给定SMT公式:

例7 $F=(x>2\vee y<5)\wedge x<1$

算法首先取得它的命题框架 $PS(F)$: $(A\vee B)\wedge C$,其中 A、B、C 分别代表 $x>2$、$y<5$、$x<1$ 。然后算法发现 $PS(F)$ 是可满足的,因此检查它的模型的一致性。首先会检查模型 $\{A,C\}$,也就是检查 $x>2$ 和

$x<1$是否有解，这种检查是通过理论求解器来进行的。理论求解器发现模型$\{A, C\}$无解，然后算法检查第二个模型$\{B, C\}$，也就是$y<5$和$x<1$。这个模型有解，因此算法返回真，说明公式F是可满足的。

为了提高上面算法的效率，人们发展出很多技术，其中多数源自SAT求解技术。下面介绍这些技术。

3.3 惰性算法用到的技术

早先的DPLL(T)算法是将SAT求解器(DPLL部分)当作黑盒，也就是说理论求解器检查模型的一致性要等到DPLL算法给出一个解之后才进行[20]，这种方法有时也被称为offline方法[20]，其缺点是理论求解器很少参与DPLL的求解过程。后来出现了online方法对其进行了改进，使得理论求解器参与DPLL，从而提高了求解效率[11-12,19-20,23-24]。这些改进主要有以下几种。

理论预处理：这种改进是对SMT公式进行预处理，并对其进行简化和标准化，必要的时候可以修改模型中的理论公式。理论预处理可以和DPLL预处理相结合。

选择分支：在DPLL中选择下一个被赋值的命题变量是很重要的。在DPLL(T)中，可以使用理论求解器来帮助选择下一个被赋值的命题变量。

理论推导：这种方法主要是通过理论的帮助来推导出一些文字。在DPLL中推导是一个提高效率的重要手段。理论推导可以导致3种结果：第一是推导出一个文字，该文字和目前的部分模型冲突，这时需要回溯；第二是推导出一个可满足的模型，这时就可以返回模型，结束求解；第三是推导终止后，既没有得到可满足的模型，又没有发生冲突，这时需要进行下一步的赋值。

理论冲突分析：在DPLL中，冲突分析有助于回溯到早期的变量，从而提高求解效率。在DPLL(T)中这种分析是通过理论求解器来进行的。

通过一个例子来说明上面的方法是怎样工作的[25]。

例8 给定一个SMT公式，它是一个CNF公式：

$c_1 \wedge c_2 \wedge c_3 \wedge c_4 \wedge c_5 \wedge c_6 \wedge c_7$

其中 c_1: $(A_1 \vee \neg B_1)$; c_2: $(\neg A_2 \vee B_2)$;

c_3: $(A_2 \vee B_3)$; c_4: $(A_1 \vee B_3)$;

c_5: $(\neg A_1 \vee \neg B_4 \vee \neg B_5)$;

c_6: $(\neg A_1 \vee B_6 \vee B_7)$;

c_7: $(A_1 \vee A_2 \vee B_8)$;

B_1: $2x_2 - x_3 > 2$; B_2: $x_1 - x_5 <= 1$;

B_3: $3x_1 - 2x_2 <= 3$; B_4: $2x_3 + x_4 >= 5$;

B_5: $3x_1 - x_3 <= 6$; B_6: $x_1 - x_4 <= 6$;

B_7: $5 - 3x_4 = x_5$; B_8: $3x_5 + 4 = x_3$

A_1、A_2的具体意义无关紧要，略去。

假设现在的部分赋值，或者说部分模型为：

$\{\neg B_5, B_8, B_6, \neg B_1\}$

这时光凭借DPLL算法是无法推导出单元子句的(单元子句就是只包含一个文字的子句)，需要结合理论求解器进行理论推导。理论求解器根据上面的部分模型代表的理论公式进行推导，得出：

$\neg(3x_1 - 2x_2 <= 3)$

这个公式就是$\neg B_3$，这时DPLL推导就可以进行下去，得到部分模型：

$\{\neg B_5, B_8, B_6, \neg B_1, \neg B_3, A_1, A_2, B_2\}$

这个模型是理论上不一致的，因为$\neg B_5$、B_6、A_1这3个文字代表的公式不一致。这时算法进行理论冲突分析，然后将新学习到的子句$B_5 \vee \neg B_8 \vee \neg B_2$加入到原公式中并且回溯到$\{\neg B_5, B_8\}$，这时根据新加入的子句，可以推导出单元子句$\neg B_2$，然后$\neg A_2$和$B_3$也可以被推导出来。

3.4 惰性方法的优化

前面一节主要介绍了理论求解器怎么帮助SAT求解器提高效率，接下来介绍一些用于提高求解效率的优化技术[25]。

标准化理论公式：有些理论公式看似不同，其实是同一公式，通过标准化，能够进行优化，尽量减少公式的数目。

例9 $(x<y)$和$(x>=y)$本来是两个公式，可以优化成：$\neg(x>=y)$, $(x>=y)$。

例10 $(x+(y+z)=1)$和$((x+y)+z=1)$是两个公式，可以优化成一个公式：$(x+y+z=1)$。

静态学习技术：这种技术是对公式的结构进行简单的分析，以求检测出一些简单的冲突。例如下

面的两个公式都可以通过静态学习检测出冲突。

例11 $x=1, x=2$

例12 $x=y, x-z<2, y-z>=2$

3.5 惰性方法的理论求解器

为了实现前文所介绍的技术，一个好的理论求解器必须有以下的功能。

模型枚举：因为在求解过程中用到理论公式的模型，所以一个好的理论求解器应该能够输出模型。

增量求解：前面介绍的DPLL(T)算法，多数都是不需要等到一个命题模型完全产生后再进行理论求解的。一般来说，产生了部分模型就开始调用理论求解器了，这就需要理论求解器具有增量求解的功能，在以前求解的基础上，添加公式后能继续快速地求解。

理论推导功能：通过理论求解，能够做一些推导，推导出理论上的单元子句，就像前文中的例8。

目前的理论求解器大都能满足上面的要求，下面简单介绍各类理论求解器。

EUF：带等词的未解释函数理论。这个理论的判定算法是基于下面的原理。

对于任何的符号 $x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n, f$ ：

如果 $x_1=y_1, x_2=y_2, \cdots, y_n=x_n$

那么 $f(x_1, x_2, \cdots, x_n)=f(y_1, y_2, \cdots, y_n)$

这里有一个EUF的例子，该例子是可满足的。

例13 $g(g(x))=x \wedge g(g(g(g(x))))=x \wedge g(x)\neq x$

EUF理论的判定算法主要是通过计算未解释符号的最小的等价闭包来进行的，这样的算法一般被称为congruence-closure算法，多数求解器是利用该算法求解EUF[26-27]。其中采用的数据结构为有向无环图(directed acyclic graph，DAG)，通过不断地对DAG进行操作，最后判定公式的可满足性。

LA：这类理论的判定算法有很长的历史，比较有名的是高斯消元法、Fourier-Motzkin法、单纯形法等。其中对于实数，它的求解是多项式时间可解的[28]；对于整数，则是NP完全的[29]。

对于实数来说，为了避免因为溢出而产生的错误，LRA的理论求解器一般要依赖于一些能处理无限精度实数的软件包。

DL：这种理论可以看作LA的子类，理论上它可以采用LA算法求解，不过因为它具有特殊的形式，人们为它开发出新的高效判定算法。前面的EUF和LA的判定算法大都是一些有着很长历史的经典算法，而DL算法较新[30]。该算法是采用图论的相关知识，将公式转换成一个图，然后寻找其中的环。一般的公式形如：

$x-y \odot c$

这里 $\odot$ 可以是=、<=、>=、≠等，c 是一个常数，算法首先要把这些公式转换成一个标准形式：

$x-y<=a$

方式是首先通过如下的规则消去=>、≠：

$x-y=a: x-y<=a \wedge y-x<=-a$

$x-y>a: \neg(x-y<=a)$

$x-y\neq a: \neg(x-y<=a) \vee \neg(y-x<=-a)$

然后消去¬：

$\neg(x-y<=a): y-x<=-a-1$（IDL）

$\neg(x-y<=a): y-x<=-a-\varepsilon$（RDL）

当公式变成标准形式后，算法构造一个加权有向图，每个变量对应于图的一个顶点，对于 $x-y<=a$ ，构造从 y 到 x 的边，该边的权重为 a 。这样，公式的可满足问题就变成了图是否有权值为负数的环的问题。如果存在一个环路，所有边的权值之和为负数，那么公式就是不可满足的。判定图中是否存在这样的环路的问题，可借助于Bellman-Ford算法[31]。

BV：该理论是位向量理论，在SMT中只处理固定位向量，也就是说向量的位数都是固定的。这个理论的主要用途是一些软件验证问题。它需要处理位向量的带模的加减乘除，以及位移、取某些位等。位向量公式的可满足性判定是NP完全的，判定算法大都是将其中每一位编码成一个命题变量，然后模拟计算机对其演算，这被称为blast算法[32-33]。此外还有一些其他算法，比如将其编码成LA处理[34]。

Arrays：这个理论用于为数组和内存操作建模，它的可满足性判定问题是NP完全的[35]。一般来说，它是经由下面的3条公理来进行推理演算的：

(1)对所有的 $a、i、e$: $(read(write(a, i, e), i)=e$。

(2)对所有的 $a、i、j、e$ ：

如果 $i \neq j$，那么 $read(write(a,i,e),j)=read(a,j)$。

(3)对所有的 a、b：

如果对所有的 i 有 $read(a,i)=read(b,i)$，那么 $a=b$。

前两个公理被称为McCarthy公理，第三个被称为扩展公理。

组合理论：很多SMT公式的背景理论不止一个，它们往往由多个理论构成。对于这样的公式的判定，需要用到Nelson-Oppen算法[1,36]。组合理论的求解问题可以简化为这样的问题：给出两个理论的文字的合取 $A_1 \wedge A_2 \wedge \cdots \wedge A_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m$，$A_1A_2\cdots A_n$ 属于理论TA，$B_1B_2\cdots B_m$ 属于理论TB，寻找一个模型满足它。一般的判定方法是根据下面的事实。

对于由两个理论组合成的文字 $A_1 \wedge A_2 \wedge \cdots \wedge A_n \wedge B_1 \wedge B_2 \wedge \cdots \wedge B_m$，假设 $V=\{v_1v_2\cdots v_k\}$ 是出现在公式中的TA和TB的共同变量。公式是可满足的当且仅当满足下列条件：

存在一组变量对的集合 $V_{eq} \subseteq V \times V$，对于里面的变量对 $\{v_i, v_j\} \in V_{eq}$，令文字 l_{ij} 为 $v_i = v_j$，另外对于所有的变量对 $\{v_i, v_j\} \in V \times V \backslash V_{eq}$，令文字 l_{ij} 为 $v_i \neq v_j$，令 L 为这些文字的合取。$A_1 \wedge A_2 \wedge \cdots \wedge A_n \wedge L$ 和 $B_1 \wedge B_2 \wedge \cdots \wedge B_m \wedge L$ 都是可满足的。

目前几乎所有的求解器都采用Nelson-Oppen算法来处理组合理论，也就是说，对于组合理论，当得到一个模型后，它们寻找一个满足上面条件的变量对的集合，找到就证明目前的模型是可满足的，找不到就证明目前的模型是不可满足的。

下面用一个例子来说明这种做法[37]，假设理论是EUF和LIA组合而成的，目前的模型是：

$$M(EUF): \neg(f(v_1)=f(v_2)) \wedge \neg(f(v_2)=f(v_4)) \wedge f(v_3)=v_5 \wedge f(v_1)=v_6$$

$$M(LIA): v_1 \geqslant 0 \wedge v_1 \leqslant 1 \wedge v_5 = v_4 - 1 \wedge v_3 = 0 \wedge v_4 = 1 \wedge v_2 \geqslant v_6 \wedge v_2 \leqslant v_6 + 1$$

要求解的模型是 $M(EUF) \wedge M(LIA)$，为了判定它的可满足性，找到一个变量对集合 $\{<v_1, v_4>, <v_3, v_5>\}$，然后发现对于这样的文字集合：$L: v_1 = v_4 \wedge v_3 = v_5 \wedge \neg(v_i = v_j)$：$i$、$j$ 不能同时为 $<1,4>$ 和 $<3,5>$，$M(EUF) \wedge L$ 是可满足的，而且 $M(LIA) \wedge L$ 也是可满足的，因此 $M(EUF) \wedge M(LIA)$ 是可满足的。

4 SMT求解器

本章介绍了一些SMT求解器，包括它们的体系结构和求解特点。这些求解器都是能处理大规模工业化例子的求解器，而且支持多种理论的求解。目前主要的求解器有Ario[38]、CVC3[39]、MathSAT[40]、Simplify[41]、TSAT[42]、Verifun[43]、Yices[44]、Z3[45]。限于篇幅，选择了3个进行介绍，分别是Z3、Yices和CVC3/CVC4。

4.1 Z3

Z3是由微软组织开发的SMT求解器，是目前最好的SMT求解器之一，它支持多种理论，主要的用途是软件验证和软件分析。Z3的原型工具参加了2007年的SMT竞赛，获得了4个理论的冠军和7个理论的亚军，之后在陆续参加的SMT竞赛中获得大多数理论的冠军。目前Z3已经被用于很多项目，比如Pex、HAVOC、Vigilante、Yogi和SLAM/SDV等。它的体系结构如图1所示[45]。

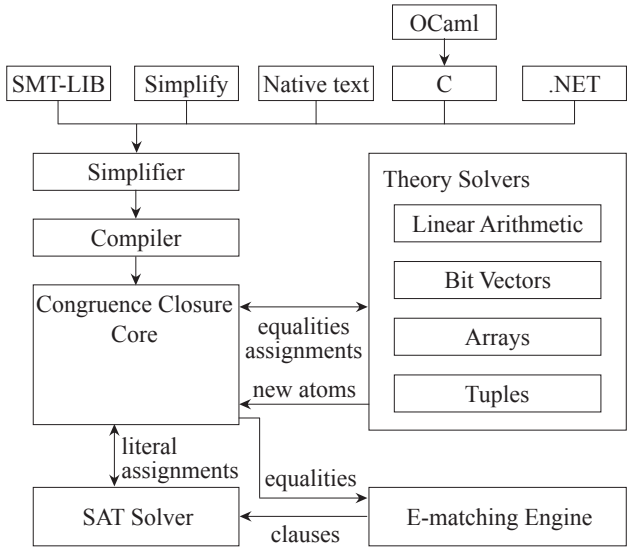


Fig.1 Architecture of Z3

图1 Z3体系结构

Z3的Simplifier采用了一个不完全，但是高效的简化策略。比如它将 $p \wedge \text{true}$ 简化成 p，将 $x=4 \wedge f(x)$ 简化成 $f(4)$ 等。它的Compiler是将输入转换成内部的数据结构和congruence-closure节点。Congruence Closure Core接受来自SAT solver的赋值，然后处理EUF和相关组合理论，它采用的方式称作E-matching。SAT Solver是对公式的命题框架进行求解，并将结果

交给Congruence Closure Core处理。Theory Solvers主要包含4种：Linear arithmetic、BV、Arrays和Tuples。理论求解器是建立在congruence-closure算法上的，这也是目前大多数SMT求解器使用的方式，也就是说，congruence-closure可以看作核心求解器，各个理论求解器是外围求解器。

4.2 Yices

Yices[44]是由SRI开发的，目前被整合在PVS中，后者是SRI开发的一个定理证明器。Yices主要用于一些有界模型检测、定理证明和学习推理项目。Yices的体系结构如图2所示。

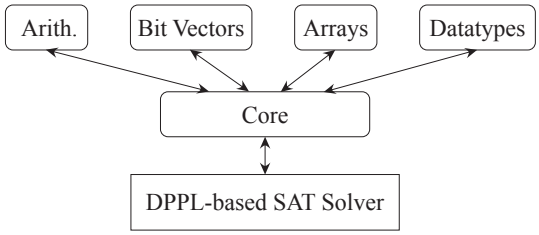


Fig.2 Architecture of Yices

图2 Yices的体系结构

Yices的核心求解器同Z3一样，也是处理未解释函数的。它的理论求解器处理的理论是算术理论、BV、Arrays和一些数据结构理论。Yices中，SAT求解器和理论求解器的结合比较密切，也比较灵活。理论求解器可以为SAT求解器产生子句，并帮助SAT求解器进行传播。Yices的核心求解器采用了改进的congruence-closure，它的算术理论采用了单纯形法，它的Arrays求解器采用lazy instantiation方法，BV求解器是Blasting方法。

Yices有自己的输入文件格式，不过它也接收SMTLIB格式的输入。此外Yices也有自己的API，它还包含了一个MAX-SAT求解器。

4.3 CVC3/CVC4

CVC3和CVC4是NYU和Iowa大学联合开发的，是SVC、CVC、CVC live的后继产品。后面几个都是斯坦福大学的产品，到了CVC3开发小组转为NYU和Iowa大学的研究人员。CVC3比前几个更为成熟，功能更多，它是一个成熟的SMT求解器，支持多种理论。CVC3的体系结构如图3所示[39]。

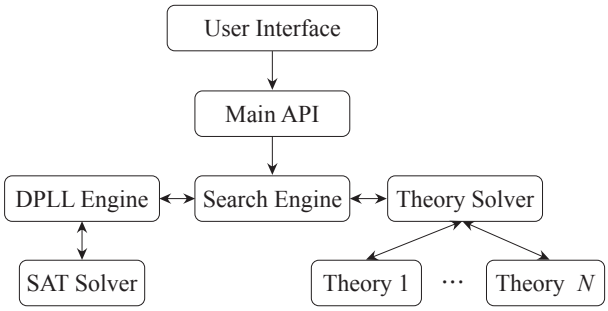


Fig.3 Architecture of CVC3

图3 CVC3的体系结构

CVC3提供多种用户接口，包括C和C++。它还支持输入文件和命令行驱动。主API支持两类主要的操作：创建公式和可满足性问题检测。它的Search Engine结合了SAT求解器和理论求解器。CVC3的一大特点是里面使用了两个SAT求解器，并吸取了两者的长处。CVC3及其前几个版本被用于HOL、一些C语言的验证工具和一些编译工具。CVC4在CVC3的基础上对代码进行了优化，并且支持了SMTLIB2.0的输入标准。

5 难点和发展

目前SMT求解技术发展迅速，不过仍存在一些瓶颈。本章介绍了一些难以处理的SMT问题，以及SMT技术未来可能的发展方向。

5.1 处理量词

一般的SMT求解器只处理不带量词的公式，不过有时候带量词的公式更有用，例如在程序验证中经常会遇到这样的公式：

$$\forall x,y:(x<y\ read(a,x)<read(a,y))$$

因为带量词的公式有很多是不可判定的，所以很多理论不存在完备算法。有的理论虽然具有完备算法，是可判定的，但是它们的计算复杂性往往很高，是PSPACE完全的，因此开发具有实际意义的求解器比较困难。尽管这样，仍然有一些研究人员开发出一些处理带量词的公式的求解器[46]。

处理带量词的公式目前采用的主要方法是实例化，简单说就是用某些项代入那些带量词的变量，例如公式：

$$f(a)<1\wedge(\forall x:f(x)>2)$$

可以选择将后面的 x 用 a 来代入,这样的话得到:

$$f(a)<1 \wedge f(a)>2$$

因此公式是不可满足的。对可满足的公式,需要更巧妙的方法来判定。Z3将某些类型的公式转换成一个等可满足性的无量词的公式,然后判定该公式的可满足性[46]。

对于特定的理论,已经存在一些处理量词的方法,比如对于LA,有各式各样的量词消去方法。不过总的来说,能处理带量词的公式的SMT求解器还不多,目前只有Z3、CVC3/CVC4等少数几个。带量词的公式的处理方法将会是以后研究的重点和热点。

5.2 优化问题

对于一些SMT理论,优化问题指的是求出满足某一条件的最优解,比如LA理论,给定一个公式,然后求出使得 $a_1x_1+a_2x_2+\cdots+a_nx_n$ 取值最大的解,这样的条件通常被称为目标函数。这类问题在实际应用中很重要,不过在SMT中起步较晚,直到近几年,才有了一些研究工作[47-49],这些工作主要集中在LA的优化上。也就是给定一个LA理论的SMT公式和一个目标函数,求使目标函数取得最大或者最小值的解。这类问题也将会是以后研究的热点。

5.3 SMT解空间大小的计算

命题逻辑和SMT公式的可满足性问题都是判定问题,但在现实世界和科学中的很多问题不仅关心解(模型)是否存在,而且要计算出解的个数。命题逻辑公式的模型计数问题即计算一个命题逻辑公式有多少个模型,也被称为#SAT问题。#SAT是一个标准的#P完全问题,#P完全问题通常比NP完全问题更加困难。SMT的计数版本即计算满足SMT公式的解的个数,或者说解空间的大小,将之称为SMT的体积计算问题。它在许多领域都有潜在的应用价值,比如近似推理和程序分析与验证。近年来,线性算术理论上的SMT公式的解空间体积计算和估算方法分别得以研究[50-51],并在软件工程领域得到了很多关注[52-53]。最近,SMT计数问题也被用于分析隐私协议[54]。

6 总结

本文分析了求解SMT的相关技术以及各个求解器的特点。SMT是SAT问题的扩展,可以为多种实际应用问题建模。SMT技术起步比较晚,但是发展迅猛,目前多数理论的求解器技术已经发展得比较成熟,但是仍有很大的研究空间。本文最后分析了目前研究的难点和未来发展方向。

致谢 关于SMT这个名词的翻译,王戟、赵建华、冯新宇、刘万伟、陈立前、沈胜宇、魏欧、吴志林等同行参与了讨论,在此表示感谢。

References:

[1] Nelson G, Oppen D C. Simplification by cooperating decision procedures[J]. ACM Transactions on Programming Languages and Systems, 1979, 1(2): 245-257.

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

[3] Shostak R E. An algorithm for reasoning about equality[J]. Communications of the ACM, 1978, 21(7): 583-585.

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

[5] Shostak R E. Deciding combinations of theories[J]. Journal of the ACM, 1984, 31(1): 1-12.

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

[7] Giunchiglia F, Sebastiani R. Building decision procedures for modal logics from propositional decision procedures—the case study of modal K[C]//LNCS 1104: Proceedings of the 13th International Conference on Automated Deduction, New Brunswick, USA, Jul 30-Aug 3, 1996. Berlin, Heidelberg: Springer, 1996: 583-597.

[8] Armando A, Castellini C, Giunchiglia E. SAT-based procedures for temporal reasoning[C]//LNCS 1809: Proceedings of the 5th European Conference on Planning, Durham, UK, Sep 8-10, 1999. Berlin, Heidelberg: Springer, 2000: 97-108.

[9] Pnueli A, Rodeh Y, Shtrichman O, et al. Deciding equality formulas by small domains instantiations[C]//LNCS 1633: Proceedings of the 11th International Conference on Computer Aided Verification, Trento, Italy, Jul 6-10, 1999. Berlin,

Heidelberg: Springer, 1999: 455-469.

[10] Bryant R E, German S, Velev M N. Exploiting positive equality in a logic of equality with uninterpreted functions[C]//LNCS 1633: Proceedings of the 11th International Conference on Computer Aided Verification, Trento, Italy, Jul 6-10, 1999. Berlin, Heidelberg: Springer, 1999: 470-482.

[11] Zhang Jian, Wang Xiaoxu. A constraint solver and its application to path feasibility analysis[J]. International Journal of Software Engineering & Knowledge Engineering, 2001, 11(2): 139-156.

[12] Zhang Jian. Specification analysis and test data generation by solving Boolean combinations of numeric constraints[C]// Proceedings of the 1st Asia-Pacific Conference on Quality Software, Hong Kong, China, Oct 30-31, 2000. Piscataway, NJ, USA: IEEE, 2000: 267-274.

[13] Bryant R E, German S, Velev M N. Processor verification using efficient reductions of the logic of uninterpreted functions to propositional logic[J]. ACM Transactions on Computational Logic, 2001, 2(1): 93-134.

[14] 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[C]//LNCS 2404: Proceedings of the 14th International Conference on Computer Aided Verification, Copenhagen, Denmark, Jul 27-31, 2002. Berlin, Heidelberg: Springer, 2002: 78-92.

[15] Bryant R E, Velev M N. Boolean satisfiability with transitivity constraints[J]. ACM Transactions on Computational Logic, 2002, 3(4): 604-627.

[16] Strichman O. On solving presburger and linear arithmetic with SAT[C]//LNCS 2517: Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design, Portland, USA, Nov 6-8, 2002. Berlin, Heidelberg: Springer, 2002: 160-170.

[17] Talupur M, Sinha N, Strichman O, et al. Range allocation for separation logic[C]//LNCS 3114: Proceedings of the 16th International Conference on Computer Aided Verification, Boston, USA, Jul 13-17, 2004. Berlin, Heidelberg: Springer, 2004: 148-161.

[18] Wolfman S, Weld D. The LPSAT engine & its application to resource planning[C]//Proceedings of the 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, Jul 31-Aug 6, 1999. San Francisco, CA, USA: Morgan Kaufmann, 1999: 310-317.

[19] Audemard G, Bertoli P, Cimatti A, et al. A SAT based approach for solving formulas over Boolean and linear mathematical propositions[C]//Proceedings of the 18th International Conference on Automated Deduction, Copenhagen, Denmark, Jul 27-30, 2002. Berlin, Heidelberg: Springer, 2002: 195-210.

[20] Barrett C, Dill D, Stump A. Checking satisfiability of first-order formulas by incremental translation into SAT[C]//LNCS 2404: Proceedings of the 14th International Conference on Computer Aided Verification, Copenhagen, Denmark, Jul 27-31, 2002. Berlin, Heidelberg: Springer, 2002: 236-249.

[21] Ball T, Cook B, Lahiri S K, et al. Zapato: automatic theorem proving for predicate abstraction refinement[C]//LNCS 3114: Proceedings of the 16th International Conference on Computer Aided Verification, Boston, USA, Jul 13-17, 2004. Berlin, Heidelberg: Springer, 2004: 457-461.

[22] Een N, Biere A. Effective preprocessing in SAT through variable and clause elimination[C]//LNCS 3569: Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing, St Andrews, UK, Jun 19-23, 2005. Berlin, Heidelberg: Springer, 2005: 61-75.

[23] Ganzinger H, Hagen G, Nieuwenhuis R, et al. DPLL(T): fast decision procedures[C]//LNCS 3114: Proceedings of the 16th International Conference on Computer Aided Verification, Boston, USA, Jul 13-17, 2004. Berlin, Heidelberg: Springer, 2004: 175-188.

[24] Bozzano M, Bruttomesso R, Cimatti A, et al. Efficient theory combination via Boolean search[J]. Information and Computation, 2006, 200(10): 1493-1525.

[25] Sebastiani R. Lazy satisfiability modulo theories[J]. Journal on Satisfiability, Boolean Modeling and Computation, 2007, 3(3/4): 141-224.

[26] Nieuwenhuis R, Oliveras A. Congruence closure with integer offsets[C]//LNCS 2850: Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, Almaty, Kazakhstan, Sep 22-26, 2003. Berlin, Heidelberg: Springer, 2003: 78-90.

[27] Nieuwenhuis R, Oliveras A. Proof producing congruence closure[C]//LNCS 3467: Proceedings of the 16th International Conference on Term Rewriting and Applications, Nara, Japan, Apr 19-21, 2005. Berlin, Heidelberg: Springer, 2005: 453-468.

[28] Khachiyan L G. A polynomial algorithm in linear programming[J]. Soviet Mathematics Doklady, 1979, 20: 191-194.

[29] Papadimitriou C H. On the complexity of integer programming[J]. Journal of the ACM, 1981, 28(4): 765-768.

[30] Cherkassky B V, Goldberg A V. Negative-cycle detection algorithms[J]. Mathematical Programming, 1999, 85(2): 277-311

[31] Bellman R. On a routing problem[J]. Quarterly of Applied Mathematics, 1958, 16: 87-90.

[32] Johannsen P, Drechsler R. Speeding up verification of RTL designs by computing one-to-one abstractions with reduced signal widths[C]//Proceedings of the 11th International Conference on Very Large Scale Integration of Systems-on-Chip, Montpellier, France, Dec 3-5, 2001: 361-374.

[33] Manolios P, Srinivasan S K, Vroon D. Automatic memory reductions for RTL level verification[C]//Proceedings of the 2006 International Conference on Computer-Aided Design, San Jose, USA, Nov 5-9, 2006. New York, NY, USA: ACM, 2006: 786-793.

[34] Brinkmann R, Drechsler R. RTL-datapath verification using integer linear programming[C]//Proceedings of the 7th Asia and South Pacific and the 15th International Conference on VLSI Design, Bangalore, India, Jan 7-11, 2002. Piscataway, NJ, USA: IEEE, 2002: 741-746.

[35] Stump A, Dill D L, Barrett C W, et al. A decision procedure for an extensional theory of arrays[C]//Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science, Boston, USA, Jun 16-19, 2001. Piscataway, NJ, USA: IEEE, 2001: 29-37.

[36] Tinelli C, Harandi M T. A new correctness proof of the Nelson-Oppen combination procedure[C]//Proceedings of the 1st International Workshop on Frontiers of Combining Systems, Munich, Germany, Mar 26-29, 1996: 103-119.

[37] Bruttomesso R, Cimatti A, Franzen A, et al. Delayed theory combination vs. Nelson-Oppen for satisfiability modulo theories: a comparative analysis[C]//LNCS 4246: Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, Phnom Penh, Cambodia, Nov 13-17, 2006. Berlin, Heidelberg: Springer, 2006: 527-541.

[38] Sheini H M, Sakallah K A. A scalable method for solving satisfiability of integer linear arithmetic logic[C]//LNCS 3569: Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing, St Andrews, UK, Jun 19-23, 2005. Berlin, Heidelberg: Springer, 2005: 241-256.

[39] Barrett C, Tinelli C. CVC3[C]//LNCS 4590: Proceedings of the 19th International Conference on Computer Aided Verification, Berlin, Germany, Jul 3-7, 2007. Berlin, Heidelberg: Springer, 2007: 298-302.

[40] Bozzano M, Bruttomesso R, Cimatti A, et al. An incremental and layered procedure for the satisfiability of linear arithmetic logic[C]//Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Held as Part of the Joint European Conferences on Theory and Practice of Software, Edinburgh, UK, Apr 4-8, 2005. Berlin, Heidelberg: Springer, 2005: 317-333.

[41] Detlefs D, Nelson G, Saxe J B. Simplify: a theorem prover for program checking, HPL-2003-148[R]. HP Laboratories Palo Alto, 2003.

[42] Armando A, Castellini C, Giunchiglia E, et al. A SAT-based decision procedure for the Boolean combination of difference constraints[C]//LNCS 3542: Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing, Vancouver, Canada, May 10-13, 2004. Berlin, Heidelberg: Springer, 2004: 16-29.

[43] Flanagan C, Joshi R, Ou Xinming, et al. Theorem proving using lazy proof explication[C]//LNCS 2725: Proceedings of the 15th International Conference on Computer Aided Verification, Boulder, USA, Jul 8-12, 2003. Berlin, Heidelberg: Springer, 2003: 355-367.

[44] Dutertre B, de Moura L. A fast linear-arithmetic solver for DPLL(T)[C]//LNCS 4144: Proceedings of the 18th International Conference on Computer Aided Verification, Seattle, USA, Aug 17-20, 2006. Berlin, Heidelberg: Springer, 2006: 81-94.

[45] de Moura L, Bjorner N. Z3: an efficient SMT solver[C]//LNCS 4963: Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Held as Part of the Joint European Conferences on Theory and Practice of Software, Budapest, Hungary, Mar 29-Apr 6, 2008. Berlin, Heidelberg: Springer, 2008: 337-340.

[46] Ge Yeting, de Moura L. Complete instantiation for quantified formulas in satisfiabiliby modulo theories[C]//LNCS 5643:

Proceedings of the 21st International Conference on Computer Aided Verification, Grenoble, France, Jun 26-Jul 2, 2009. Berlin, Heidelberg: Springer, 2009: 306-320.

[47] Ma Feifei, Yan Jun, Zhang Jian. Solving generalized optimization problems subject to SMT constraints[C]//LNCS 7285: Proceedings of the Joint International Conference on Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, Beijing, China, May 14-16, 2012. Berlin, Heidelberg: Springer, 2012: 247-258.

[48] Sebastiani R, Tomasi S. Optimization in SMT with LA(Q) cost functions[C]//LNCS 7364: Proceedings of the 6th International Joint Conference on Automated Reasoning, Manchester, UK, Jun 26-29, 2012. Berlin, Heidelberg: Springer, 2012: 484-498.

[49] Li Yi, Albarghouthi A, Kincaid Z, et al. Symbolic optimization with SMT solvers[C]//Proceedings of the 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Diego, USA, Jan 20-21, 2014. New York, NY, USA: ACM, 2014: 607-618.

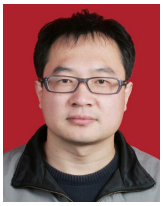
[50] Ma Feifei, Liu Sheng, Zhang Jian. Volume computation for Boolean combination of linear arithmetic constraints[C]//LNCS 5663: Proceedings of the 22nd International Conference on Automated Deduction, Montreal, Canada, Aug 2-7, 2009. Berlin, Heidelberg: Springer, 2009: 453-468.

[51] Zhou Min, He Fei, Song Xiaoyu, et al. Estimating the volume of solution space for satisfiability modulo linear real arithmetic[J]. Theory of Computing Systems, 2015, 56(2): 347-371.

[52] Liu Sheng, Zhang Jian. Program analysis: from qualitative analysis to quantitative analysis[C]//Proceedings of the 33rd International Conference on Software Engineering, Honolulu, USA, May 21-28, 2011. New York, NY, USA: ACM, 2011: 956-959.

[53] Geldenhuys J, Dwyer M B, Visser W. Probabilistic symbolic execution[C]//Proceedings of the 2012 International Symposium on Software Testing and Analysis, Minneapolis, USA, Jul 15-20, 2012. New York, NY, USA: ACM, 2012: 166-176.

[54] Fredrikson M, Jha S. Satisfiability modulo counting: a new approach for analyzing privacy properties[C]//Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science, Vienna, Austria, Jul 14-18, 2014. New York, NY, USA: ACM, 2014.



JIN Jiwei was born in 1977. He received the Ph.D. degree in mathematics logic from Sun Yat-sen University in 2010. Now he is a post-doctor at Institute of Software, Chinese Academy of Sciences. His research interests include SAT, CSP and SMT, etc.

金继伟(1977—),男,山东德州人,2010年于中山大学获得博士学位,现为中国科学院软件研究所博士后,主要研究领域为SAT,CSP,SMT等。

MA Feifei was born in 1982. She received the Ph.D. degree from Institute of Software, Chinese Academy of Sciences in 2010. Now she is an associate professor at Institute of Software, Chinese Academy of Sciences. Her research interests include automated reasoning and constraint solving, etc.

马菲菲(1982—),女,河南南阳人,2010年于中国科学院软件研究所获得博士学位,现为中国科学院软件研究所副研究员,主要研究领域为自动推理,约束求解等。

ZHANG Jian was born in 1969. He received the Ph.D. degree from Institute of Software, Chinese Academy of Sciences in 1994. Now he is a professor and Ph.D. supervisor at Institute of Software, Chinese Academy of Sciences, and the senior member of CCF, ACM and IEEE. His research interests include automated reasoning, constraint solving, semantic Web, program static analysis and error detection, software testing and data generation, etc.

张健(1969—),男,安徽庐江人,1994年于中国科学院软件研究所获得博士学位,现为中国科学院软件研究所研究员,博士生导师,CCF、ACM和IEEE高级会员,主要研究领域为自动推理,约束求解,语义Web,程序静态分析与检错,软件测试,数据生成等。