

基于隐式谓词抽象和属性导向可达的SCADE模型检测

张聪, 陈哲, 王慧杰, 韦依洋

引用本文

张聪, 陈哲, 王慧杰, 韦依洋. 基于隐式谓词抽象和属性导向可达的SCADE模型检测[J]. 计算机科学, 2025, 52(12): 24-31.

ZHANG Cong, CHEN Zhe, WANG Huijie, WEI Yiyang. [SCADE Model Checking Based on Implicit Predicate Abstraction and Property-directed Reachability](#) [J]. Computer Science, 2025, 52(12): 24-31.

相似文章推荐 (请使用火狐或 IE 浏览器查看文章)

Similar articles recommended (Please use Firefox or IE to view the article)

[一种结合静态分析的轻量化内存安全运行时检测方法](#)

Lightweight Memory Safety Runtime Detection Method Combined with Static Analysis
计算机科学, 2025, 52(11A): 241100060-8. <https://doi.org/10.11896/jsjcx.241100060>

[结合动态分析的内存安全漏洞模糊测试方法](#)

Dynamic Analysis Based Fuzz Testing for Memory Safety Vulnerabilities
计算机科学, 2025, 52(11): 382-389. <https://doi.org/10.11896/jsjcx.241000003>

[复杂系统的形式化建模及量化分析方法综述](#)

Survey on Formal Modelling and Quantitative Analysis Methods for Complex Systems
计算机科学, 2025, 52(9): 346-359. <https://doi.org/10.11896/jsjcx.240600022>

[一种基于线性插值的对抗攻击方法](#)

Linear Interpolation Method for Adversarial Attack
计算机科学, 2025, 52(8): 403-410. <https://doi.org/10.11896/jsjcx.240700058>

[MTFuse:基于Mamba和Transformer的红外与可见光图像融合网络](#)

MTFuse: An Infrared and Visible Image Fusion Network Based on Mamba and Transformer
计算机科学, 2025, 52(8): 188-194. <https://doi.org/10.11896/jsjcx.240600106>

基于隐式谓词抽象和属性导向可达的 SCADE 模型检测

张聪¹ 陈哲^{1,2} 王慧杰¹ 韦依洋¹

1 南京航空航天大学计算机科学与技术学院 南京 211106

2 软件新技术与产业化协同创新中心 南京 211106

(1912460139@qq.com)

摘要 SCADE 被广泛应用于航空航天、核电站、轨道交通、医疗设备等关乎生命安全的关键行业。将模型检测应用于这些安全关键领域,能够有效地确保系统的安全。目前,针对 SCADE 模型检测的研究较少,多数研究基于程序转化,借助其他更简单的语言模型检测工具来完成验证,而少有的可实现对 SCADE 程序进行模型检测的全流程工具则验证效率较低。为此,提出了一种基于隐式谓词抽象和属性导向可达的模型检测算法(IAPDR),将其并行集成到现有的针对 SCADE 程序的模型检测工具(PSMC)上,该工具实现了 SCADE 程序的分析、建模和模型检测的全流程。此外,通过理论证明了所提出算法的正确性,通过实验评估了 IAPDR 算法以及扩展后的工具(PSMCWI)的效果和性能。与传统的 BMC, K-Induction 和 CEGAR 算法相比, IAPDR 算法在数据集上具有最高的验证成功覆盖率和最短的验证总耗时。与原生的 PSMC 工具相比, PSMCWI 在数据集上能够多验证 139 个 SCADE 程序,验证成功覆盖率提升了 15.1%,验证的总耗时减少了 43%。与 JKind 的对比实验的结果表明, IAPDR 算法能够正确地对 SCADE 程序进行模型检测,相比于将 SCADE 模型转化为 Lustre 模型后,用 JKind 对 Lustre 模型进行模型检测来实现对 SCADE 程序进行模型检测的方法, PSMCWI 具有更高的效率。

关键词: SCADE; 模型检测; 状态空间; 谓词抽象; 属性导向可达; 插值

中图分类号 TP311

SCADE Model Checking Based on Implicit Predicate Abstraction and Property-directed Reachability

ZHANG Cong¹, CHEN Zhe^{1,2}, WANG Huijie¹ and WEI Yiyang¹

1 College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

2 Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 211106, China

Abstract SCADE is widely used in critical industries related to life safety, such as aerospace, nuclear power plants, rail transit, and medical equipment. Applying model checking to these safety critical areas can effectively ensure the safety of the system. At present, there is relatively little research on SCADE model checking. Most studies are based on program translation and use other simpler language model checking tools to complete verification. However, few tools that have implemented the entire process of model checking for SCADE programs have low verification efficiency. This paper proposes a model checking algorithm based on implicit predicate abstraction and property-directed reachability(IAPDR), which is parallelly integrated into existing model checking tools(PSMC) for SCADE programs. The tool implements the entire process of analysis, modeling, and model checking for SCADE programs. In addition, the correctness of the proposed algorithm is theoretically proven, and the effectiveness and performance of the expanded tool(PSMCWI) are evaluated through experiments. Compared with traditional BMC, K-Induction, and CEGAR algorithms, IAPDR has the highest verification success coverage and the lowest total verification time on the benchmarks. Compared with the native PSMC tool, PSMCWI can verify 139 more SCADE programs on the benchmarks, with a 15.1% increase in successful verification coverage and a 43% reduction in total verification time. The results of the comparative experiment with JKind show that IAPDR can correctly do model checking on SCADE programs. Compared with the method of translating SCADE models into Lustre models and using JKind to do model checking on Lustre models to achieve model checking on SCADE programs, PSMCWI has higher efficiency.

Keywords SCADE, Model checking, States space, Predicate abstraction, Property-directed reachability, Interpolation

到稿日期:2024-11-11 返修日期:2025-02-12

基金项目:国家自然科学基金(62172217);国家自然科学基金委员会-中国民航局民航联合研究基金(U1533130)

This work was supported by the National Natural Science Foundation of China(62172217) and Joint Research Funds of National Natural Science Foundation of China and Civil Aviation Administration of China(U1533130).

通信作者:陈哲(zhechen@nuaa.edu.cn)

1 引言

在当今高度依赖自动化系统的社会中,软件的安全性和可靠性成为不可忽视的重要议题。尤其在航空航天、核电站、轨道交通、医疗设备等关乎生命安全的关键行业中,任何细微的软件故障都可能导致灾难性的后果。SCADE^[1] (Safety-Critical Application Development Environment)作为一种专为开发安全关键应用而设计的语言及开发环境,因能够高效地设计、验证并生成高质量的嵌入式软件代码而被广泛使用。为了保证 SCADE 程序的安全性,除了常规的测试方法外,还需要在投入使用前对 SCADE 程序的安全属性进行验证。模型检测常被应用于各种软硬件程序的属性验证,它将目标系统进行形式化建模,使用逻辑推理或模型检测算法,检测目标模型是否符合某种需要被关注的安全属性。

目前,针对 SCADE 程序的模型检测的研究中,大部分的实现是将 SCADE 程序建模后转化成语法更简单的其他语言模型,再通过调用转化后模型对应的模型检测工具进行模型检测。这种方法可能无法全面覆盖 SCADE 语法,因为语法更为简单的其他语言模型的表达能力有限,无法表达一些 SCADE 语言结构。此外,模型检测完全依赖于转化后模型对应的模型检测工具,因此验证效率和结果也完全依赖于该工具,没有优化的空间。

Ran 等^[2]提出了一种从 SCADE 模型到 Lustre 模型的转化算法,并使用 JKind^[3]作为模型转化后的模型检测工具以完成对 SCADE 程序的模型检测,但 SCADE 模型中的一些高级操作符和复杂数据结构并未被有效转化,这意味着所提出的转化算法只适用于验证 SCADE 程序的部分模型的安全性。文献[4]将自然语言描述的安全属性转化为 LTL 和 CTL 公式,将 SCADE 模型转化为 nuXmv^[5]可接受的输入,使用 nuXmv 完成模型检测,其中的转化仅包含了 SCADE 的部分语法,对 nuXmv 也具有较强的依赖。Li 等^[6]实现的 StoL 工具从 SCADE 模型合成 Lustre 模型,能够覆盖 SCADE 的大部分语法,但其仍依赖于 JKind 对 Lustre 程序的模型检测能力。

除了上述模型转化后调用模型检测工具的思想外,一些学者将 SCADE 模型转化为语法更简单的语言模型后,再将其转化为 SMT^[7]求解器的输入,之后结合模型检测算法完成对安全属性的验证。文献[8]将 SCADE 程序转化为 LAMA 程序,然后将 LAMA 程序再转化为 SMT 求解器的输入,最后结合 SMT 求解器和模型检测算法完成模型检测。但是,部分 SCADE 语句无法转化为 LAMA 程序,而且它只使用了 BMC^[9]和 K-Induction^[10]两种模型检测算法,且整个求解过程使用单线程完成,因此模型检测的效率低。

类似地,Fang 等^[11]提出了基于多引擎并行协作的 SCADE 模型检测方法,将 SCADE 程序建模后转化为一阶逻辑公式,结合 SMT 求解器和模型检测算法对安全属性进行形式化验证,实现了多引擎并行协作的 SCADE 模型检测工具 PSMC。PSMC 使用了多引擎并行验证的方法,在一定程度上提升了模型检测的效率,但其使用的基本算法还是 BMC 和 K-Induction,虽然引入了抽象精化、无关变量切片等技术,

但仍不能避免算法的局限性,即状态爆炸问题,无法在限定超时时间内完成对复杂程序的模型检测。

抽象技术^[12]简化系统模型,通过缩小系统的状态空间,能够有效解决状态爆炸问题。IC3^[13]算法不展开迁移关系,而是增量地生成逐步逼近可达性信息的归纳相关性子句,能有效地缓解状态爆炸问题。IC3 算法的优秀思想启发了不少新的模型检测算法,UAIR^[14]算法融合了 IC3 算法对可达性信息的提取,基于不满足核心构造不变式,在一些特例上有优秀的验证表现。CAR^[15-16]算法优化了 IC3 算法的公式序列的单调性,双向维护公式序列,引入死状态检测等启发式算法以加速验证过程,具有不错的验证表现,尤其在反例的搜索上优于原生的 IC3 算法。

综上所述,对 SCADE 程序进行模型检测,应当避免对其其他语言模型检测工具的强依赖,提升扩展性。受并行集成引擎、抽象精化等算法的思想能提升模型检测效率和解决状态爆炸问题的启发,本文提出了一种基于隐式谓词抽象和属性导向可达的 SCADE 模型检测方法,将隐式谓词抽象^[12]和 IC3 的优秀思想进行有效融合,提出了 IAPDR 算法,将其并行扩展到 PSMC 工具中,通过理论证明了算法的正确性,通过实验评估了算法的效果和性能。

本文的主要工作包括:

- 1)提出了一种基于隐式谓词抽象和属性导向可达的模型检测算法(IAPDR);
- 2)实现了 IAPDR 算法,并以并行引擎的形式扩展到 PSMC 上;
- 3)从理论上证明了 IAPDR 算法的正确性;
- 4)通过实验评估了 IAPDR 算法的效果和性能,IAPDR 算法的验证效率优于传统的 BMC,K-Induction 和 CEGAR 算法,扩展后的 PSMC 在数据集上的验证成功覆盖率提升了 15.1%,验证总耗时缩短了 43%。

2 基础知识

2.1 基本概念

本文使用标准的一阶逻辑的概念,使用 x, y 来表示变量,使用 X, Y, \bar{X}, \hat{X} 来表示变量的集合,使用 φ, ψ, I, T, P 来表示公式。*literal* 是原子变量或它的取反,*clause* 是对 *literal* 的析取,*cube* 是对 *literal* 的合取。如果 s 是 $l_1 \wedge \dots \wedge l_n$, 则 s 是一个 *cube*, $\neg s$ 是一个 *clause*, 为 $\neg l_1 \vee \dots \vee \neg l_n$ 。对于变量 x, x' 表示 x 在 1 步迁移后的版本, x^2, \dots, x^n 则表示 x 在 2, \dots, n 步迁移后的版本。对于集合 X, X' 表示将集合中所有变量 x 替换为 x' 对应的版本,相应地, \bar{X} 由 \bar{x} 替换, X^n 由 x^n 替换。 $P(X_1, \dots, X_n)$ 表示所有出现在公式 P 中的变量,属于集合 X_1, \dots, X_n 的并集。对于公式 P, P' 是将 P 的所有变量 x 替换为 x' 对应的版本。

迁移系统是对具体程序或系统的抽象描述,它由一个元组进行表示, $S = \langle X, I, T \rangle$, X 表示该系统的变量集合,由 X 中所有变量的赋值合取成的 *cube* 表示系统的一个状态,部分变量的赋值合取成的 *cube* 表示系统的一个状态集合。 $I(X)$ 表示该系统的初始状态集合的公式, $T(X, X')$ 表示系统的

迁移关系, $T(X^i, X^{i+1})$ 表示从初始状态集合经过 i 步迁移后的状态集合与经过 $i+1$ 步迁移后的状态集合之间的迁移关系。状态 s 是对变量的一组赋值, 系统的一条路径 $path$ 由 s_0, s_1, \dots, s_n 组成, 其中 $s_0 \models I$, 且对任意 $0 \leq i < n$, $(s_i, s'_{i+1}) \models T$ 。

模型检测的目标是对给定的迁移系统 S 和待验证属性 P , 检查是否满足 $S \models P$ 。若是, 则说明迁移系统 S 的所有路径 s_0, s_1, \dots, s_k , 都有 $s_i \models P, 0 \leq i < k$, 即属性 P 是安全的。若否, 则说明存在一条路径 s_0, s_1, \dots, s_k , 使得 $s_k \models \neg P$, 即属性 P 是不安全的。

2.2 模型检测算法

2.2.1 有界模型检测

有界模型检测(Bounded Model Checking, BMC)算法通过增量添加状态迁移关系, 搜索状态空间, 尝试在 k 步以内找到一个违反属性 P 的路径。更直观地, BMC 算法的验证过程就是判断如下逻辑公式是否是可满足的。

$$I \wedge T_{01} \wedge T_{12} \wedge \dots \wedge T_{k-1k} \rightarrow P_0 \wedge P_1 \wedge \dots \wedge P_k \quad (1)$$

其中, T_{i+1} 表示 $T(X^i, X^{i+1})$ 。

2.2.2 K-归纳

K-归纳(K-Induction)算法基于有界模型检测算法的验证过程, 引入归纳法的思想来验证属性 P 的全局安全性, 其验证过程分为两个阶段。在阶段 1, 验证 k 步迁移内是否有违反 P 的路径, 即判断式(1)是否可满足; 在阶段 1 验证结果为可满足后(不可满足则存在反例路径), 再在阶段 2 使用归纳法将 k 步扩展至任意时钟对应的状态空间, 即判断式(2)是否是可满足的。

$$T_{m+1} \wedge P_{n+1} \dots \wedge T_{n+k-1, n+k} \wedge P_{n+k} \rightarrow P_{n+k+1} \quad (2)$$

2.2.3 基于插值的模型检测

基于插值的模型检测算法^[17](Interpolation-based Model Checking, IMC)利用 Craig 插值^[18]来生成中间公式, 从而减少对状态空间的探索, 缓解状态爆炸问题, 提高验证效率。

$$\underbrace{I \wedge T_{01} \wedge T_{12} \wedge \dots \wedge T_{k-1k}}_A \wedge \neg P_k \quad (3a)$$

$$\underbrace{C \wedge T_{01} \wedge T_{12} \wedge \dots \wedge T_{k-1k}}_{A'} \wedge \neg P_k \quad (3b)$$

若式(3a)是可满足的, 则存在一个长度为 k 的路径违反属性 P 。否则, 根据 Craig 插值定义, 计算出插值 C 。假设插值包含一些非初始状态, 用这些状态替换阶段 1 中的初始状态集合 I , 即式(3b), 如果式(3b)不可满足, 则存在另一个插值 C' , 是从初始状态经过 2 步迁移可达的状态集合的更大的近似状态集合。重复这样的计算, 当初始状态和所有先前的插值的并集增长到一个不动点时, 所有的初始状态就都不可能经过迁移到违反属性 P 的状态。

2.3 IC3/PDR

与 2.2 节介绍的算法不同, IC3 算法不展开迁移关系, 而是增量地生成逐步逼近可达性信息的归纳相关性子句。基于 IC3 的属性导向可达的特性, 下文用 PDR 表示 IC3 算法。

PDR 在证明 $S \models P$ 的过程中, 尝试找到一个公式 $F(X)$, 使 $F(X)$ 满足: 1) $I(X) \models F(X)$; 2) $F(X) \wedge T(X, X') \models F(X')$; 3) $F(X) \models P(X)$ 。若 $F(X)$ 存在, 则属性 P 是安全的。

为了构造出与 $P(X)$ 有归纳相关性的 $F(X)$, PDR 增量地维护一个公式序列 $F_0(X), \dots, F_k(X)$, 序列 $trace$ 满足如下条件:

- 1) $F_0 = I$;
- 2) $F_i \models F_{i+1}$;
- 3) $F_i(X) \wedge T(X, X') \models F_{i+1}(X')$;
- 4) $F_i \models P, i < k$ 。

$trace$ 中的元素叫作 $frame$, $trace$ 中的每个 $frame$ 都与其前一个 $frame$ 具有归纳相关性。PDR 通过阻塞具有归纳相关性的 $cube$ 来强化这些 $frame$, 这依赖于式(4)的不可满足性。

$$indRel(F, T, c) = F \wedge \neg c \wedge T \wedge c' \quad (4)$$

更具体地, PDR 具有两阶段的交替处理, 分别是阻塞和传播阶段。阻塞阶段, PDR 通过检查 $trace$ 证明 F_k 和 $\neg P$ 不存在交集, 若不能在当前 $trace$ 获得证明, 则存在反例路径使得 $\neg P$ 成立。

在阻塞过程中, PDR 在式(4)可满足时, 创建任务, 反向向前搜索相关的 $cube$, 不断向 $frame$ 中添加 $clause(\neg c)$, 从而强化对可达状态空间的近似估计, 阻塞结束时, $F_k \models P$ 。传播阶段, PDR 为 $trace$ 添加新的 $frame$, 将前序 $frame$ 中的 $clause$ 向后扩展, 在该阶段中, 若存在相邻两个 $frame$ 中的公式相等, 则找到了与 $P(X)$ 有归纳相关性的 $F(X)$, 即证明属性 P 是安全的。

2.4 谓词抽象

谓词抽象在减小状态空间的同时, 保留了具体程序或系统的一些属性的可满足性。如果某个属性在 S 中是可达的, 那么该属性在 S 的抽象状态空间中也是可达的。因此, 若某个属性在 S 的抽象状态空间中是不可达的, 那么在 S 中也是不可达的。

在谓词抽象中, 抽象状态空间由一个谓词集合进行描述。给定迁移系统 S , 一个谓词集合 $PSet$, 每个谓词是由具体系统的变量集合组成的公式。每个 $p \in PSet$ 被定义为抽象系统的变量 x_p , 抽象系统与具体系统之间的关系由 $R_{PSet}(X, X_p) := \bigwedge_{p \in PSet} x_p \leftrightarrow p(X)$ 定义, 给定公式 $\varphi(X)$, 其抽象版本为:

$$\widehat{\varphi}_{PSet} = \exists X, X' \cdot (\varphi(X, X') \wedge R_{PSet}(X, X_{PSet}) \wedge R(X, X_{PSet}'))$$

3 IAPDR

本文提出了一种基于隐式谓词抽象和属性导向可达的模型检测算法——IAPDR。IAPDR 基于原生 PDR 算法, 融合隐式谓词抽象、BMC 模拟、插值, 在 PDR 迭代中寻找新谓词, 精化对模型的过度抽象, 将状态空间逐步收缩, 逼近不动点, 以证明属性的安全性, 并在精化过程中判断是否存在反例路径。

IAPDR 用 $PCube$ 表示 $cube$, $PCube$ 由谓词的赋值组成, 而不是具体系统中的变量。待处理的任任务用 $TPCube$ 表示, 包含 $PCube$ 和 $index$, $index$ 表示该任务相关的 $frame$ 在公式序列 F 中的下标。

3.1 PDR 优化

Bradley^[13]提出的 PDR 算法在主迭代进行前, 需要对 0 和 1 步迁移到达的状态空间进行检查, 在属性 P 满足后才进

入主迭代。本文实现的算法将两阶段合并,并在算法结构上进行调整,使得算法的框架^[19]更加清晰。

另外,对于在某次迭代的反向搜索的过程中产生的任务 $TPCube_{pc}$,将 tpc 的 $PCube$ 向后共享,让后续的 $frame$ 也在本次迭代中处理该 $PCube$,以避免后续迭代中的重复处理。

除此之外,在每次对坏 $PCube$ pc_0 进行阻塞时,对所有前置 $frame$ 中已有的 $PCube$ 进行检查,对任何一个 $PCube$ pc ,若 pc_0 是 pc 的子集,则删除该 $frame$ 中的 pc 。一方面,累计的 $PCube$ 在求解时只会增加求解器的压力,及时清除冗余的 $PCube$ 能够有效提升求解效率。另一方面,原生的 PDR 在传播阶段检查是否存在一对相邻的 F_i 和 F_{i+1} 相等,以判断是否退出迭代。在去除冗余 $PCube$ 后,上述判断条件可转化为是否有一个 $frame$ 阻塞的 $PCube$ 的数量为 0,不再需要去比较相邻的 $frame$ 是否相等。

3.2 隐式谓词抽象

IAPDR 的核心思想是将 PDR 运行在谓词抽象出的状态空间上,使用抽象反例路径提供的信息构建出新谓词,以精化模型或者获取到具体系统的反例路径。

在谓词抽象的一般实现中,需要显式指定谓词集合,这种方式需要对程序有深入的理解,不当的谓词选择可能使抽象效果不理想。而隐式谓词抽象自动从程序的执行过程中提取谓词,无需手动指定,自动化程度高,不需人工干预,同时能够根据程序的实际运行情况动态调整抽象,更适合于快速迭代和大规模程序的验证。

隐式谓词抽象将谓词抽象和 $path$ 绑定在一起,隐式地将具有相同特性的具体状态关联为同一个抽象状态(见图 1),依赖于下式:

$$H_{PSet}(X, \bar{X}) = \bigwedge_{p \in PSet} p(X) \leftrightarrow p(\bar{X})$$

其中, H 将两个具体状态关联成一个抽象状态。具体状态对应具体系统,使用具体的变量进行描述;抽象状态对应抽象系统,使用谓词进行描述。隐式谓词抽象不对迁移关系进行抽象,而直接使用具体系统的迁移关系 T ,而谓词由具体的变量构成,使得 T 能够作为抽象系统的迁移关系。由图 1 可知,抽象系统中的某条路径,在映射到具体系统时可能是不存在的,将这种路径称为伪反例。伪反例的出现意味着抽象系统的抽象程度过高,可以从伪反例中提取出新谓词,从而精化抽象系统以降低抽象程度。

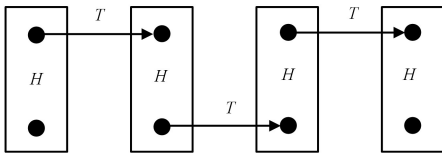


图 1 抽象路径

Fig. 1 Abstract path

3.3 IAPDR

IAPDR 的初始化阶段,向谓词集合中添加谓词 $Init$ 和 P ,创建 F_0 和 F_∞ , F_∞ 用于限定迭代次数。与 PDR 的验证流程基本一致,IAPDR 也有阻塞和传播两个阶段,不同之处在于:

1)将初始状态集合进行 0 和 1 步迁移的检查合并到主迭代中,直接通过 $F_{cur} \wedge \neg P$ 获取坏 $PCube$,再进行反向搜索;

2)在算法反向搜索相关 $PCube$ 并触达抽象初始状态集合时,即在抽象系统中发现反例路径时,需要对该抽象反例路径进行检查;

3)抽象反例路径的检查过程使用 BMC 算法对该路径在具体系统中进行模拟,若路径是可满足的,则说明反例成立,若不可满足,则该抽象反例路径是伪反例,说明当前抽象需要精化;

4)精化过程对具体系统的不可满足的 BMC 路径进行插值计算,将获取到的插值拆解为具有原子性的子式,将这些子式作为新谓词添加到谓词集合中以具体化抽象系统。

IAPDR 初始化阶段的伪代码如算法 1 所示。

算法 1 IAPDR

输入: $S=(I, T, P)$

输出: P is valid or invalid with counterexample

```

1. function check( $M; S$ )
2.    $F \leftarrow \{F_0 = I, F_\infty\}$ 
3.    $PSet \leftarrow \{Init, P\}$ 
4.   addT2Solver( $T$ )
5.   while true do
6.      $c := getBadPCube(F_{cur} \wedge \neg P)$ 
7.     if  $c \neq nullptr$  then
8.       try blockPCube( $TPCube(c, cur)$ )
9.       catch FalseException return false
10.    else
11.      addNewFrame()
12.       $res := propagateBlockedPCube()$ 
13.      if  $res == true$  then return true
14.      else continue
15.    end if
16.  end if
17. end while

```

IAPDR 算法以 $check(\dots)$ 为入口,初始化 $frame$ 序列和谓词集合后进入主迭代,在属性被验证为安全时返回 true,反之返回 false 并在日志中打印出具体系统的反例路径。

与 PDR 获取经过 1 步迁移到达 $\neg P$ 的坏 $cube$ 不同, $getBadPCube(\dots)$ 获取当前迭代轮次 cur 对应的 $frame$ 所映射的状态空间中的 $\neg P$ 状态,即坏 $PCube$ 。主迭代根据坏 $PCube$ 的获取情况进入阻塞或者传播阶段。

```

1. function blockPCube( $tpc_0: TPCube$ )
2.    $L \leftarrow \{\emptyset\}$ 
3.    $Q \leftarrow \{tpc_0\}$ 
4.   while  $Q$  isn't empty do
5.      $tpc := Q.front()$ 
6.      $Q.pop()$ 
7.      $L.push\_front(tpc)$ 
8.     if  $tpc.index == 0$  then
9.       refine( $L$ ) return
10.    end if
11.    if isBlocked( $tpc$ ) == false then
12.       $tpc1 := relativeSearch(tpc)$ 
13.      if  $tpc1.index \neq -1$  then
14.         $tpc1 = generalize(tpc1)$ 

```

```

15.     forward(tpc1)
16.     addBlockedPCube(tpc1)
17.     else
18.         tpc1.setIndex(tpc.getIndex()-1)
19.         Q.push(tpc1)
20.         Q.push(tpc)
21.     end if
22. end if
23. end while

```

阻塞阶段, IAPDR 维护一个先序任务队列 Q 和一个 $PCube$ 链表 L 。 Q 用于坏 $PCube$ 的反向搜索, 任务的 $index$ 越小, 任务越先被处理。 L 记录当前的搜索路径, 在路径触达初始状态集合时, 即找到抽象反例路径, 需对具体系统的路径进行模拟和精化。

$isBlocked(\dots)$ 判断当前任务对应的 $frame$ 是否已阻塞当前任务的 $PCube$, 避免重复处理。

$relativeSearch(\dots)$ 反向地向前 1 步搜索可达当前处理任务的 $PCube$ 的相关 $PCube$, 以及该 $PCube$ 所关联的 $frame$, 构造新的任务并入队。其搜索过程对应 PDR 对式(4)的求解。PDR 在处理任务时最多会求解式(4) $cur-index$ 次。IAPDR 将这个过程用一次求解替代, 并从 SMT 求解器返回的 $unsatcore$ 中提取需要的信息来减少对求解器的调用次数。当式(5)不可满足时, 说明没有相关 $PCube$, 反向搜索的路径断裂, 返回的新任务根据 $unsatcore$ 进行构造, 若式(5)可满足, 则说明有相关 $PCube$, 新任务的 $PCube$ 为求解器返回的 $model$ 在谓词集合上的赋值, 新任务的 $index$ 设置为 -1 。

$$F_{index-1} \wedge \dots \wedge F_{cur} \wedge \neg c \wedge T \wedge c' \quad (5)$$

$generalize(\dots)$ 对任务的 $PCube$ 进行泛化, 将 $PCube$ 中的无关 $literal$ 剔除, 让 $PCube$ 所能映射的状态空间尽可能大, 在后续阻塞时能更快地收缩状态空间, 加速验证过程。

$forward(\dots)$ 将任务的 $PCube$ 向后共享, 让后续的 $frame$ 也在本次迭代中处理该 $PCube$, 避免后续迭代中的重复处理。

$addBlockedPCube(\dots)$ 在反向搜索出现路径断裂时, 对当前任务的 $PCube_{pc_0}$ 进行阻塞。首先对任务对应的 $frame$ 的前序 $frame$ 中的已阻塞 $PCube_{pc}$ 进行检查, 剔除满足 pc_0 是 pc 的子集的 pc 。消除冗余 $PCube$, 以减轻求解器的压力和简化迭代退出的判定条件。

```

1. function refine(L:List(PCube))
2.   res.uc:=simulate(L)
3.   switch res
4.     casesat:throw FlaseException
5.     case unsat:
6.       pres:=getPredicates(uc)
7.       updatePredicates(pres)
8.   end switch

```

$refine(\dots)$ 对具体系统进行 BMC 模拟并根据伪反例提取新谓词。传入的链表 L 是一条抽象反例路径, 使用 BMC 算法进行具体系统的路径模拟。当路径可满足时, 便返回具体的路径作为反例; 当路径不可满足时, 则说明该路径是伪反例。 $getPredicates(\dots)$ 提取谓词的方式类似于 IMC 算法, 当公式

$$C_0 \wedge T_{0,1} \wedge C_1 \wedge T_{1,2} \wedge \dots \wedge T_{k-1,k} \wedge C_k$$

不可满足时, 分别对:

$$\underbrace{C_0 \wedge T_{0,1}}_A \wedge \underbrace{C_1 \wedge T_{1,2} \wedge \dots \wedge T_{k-1,k}}_B \wedge C_k$$

⋮

$$\underbrace{C_0 \wedge T_{0,1} \wedge C_1 \wedge T_{1,2}}_A \wedge \dots \wedge \underbrace{T_{k-1,k} \wedge C_k}_B$$

求插值。若链表 L 中存在 k 个 $PCube$, 则能够获得到 $k-1$ 个插值, 对插值进行子式的分解以构建新谓词。假设插值为 $a=1 \wedge b>5 \wedge a>c \wedge \neg e$, 则提取的谓词为 $\{a=1, b>5, a>c, e\}$ 。 $updatePredicate(\dots)$ 将提取到的新谓词去重后添加到全局谓词集 $PSet$ 中。

```

1. function propagateBlockedPCube()
2.   for i=1 to k
3.     for pc:PCube in Fi
4.       tpc:=TPCube(pc,i+1)
5.       tpc1:=relativeSearch(tpc)
6.       if tpc1.index != -1 then
7.         addBlockedPCube(tpc1)
8.       end if
9.     end for
10.  if Fi.isEmpty() then return true
11.  end if
12. end for
13. return false

```

传播阶段, 对 $i \in [1, cur-1]$, F_i 中的每个 $PCube$ 进行检查, 若 F_i 对应的状态空间中的所有状态经过 1 步迁移都不能到达 $PCube$ 对应的状态空间, 那么在 F_{i+1} 中阻塞该 $PCube$ 。传播完成后, 若任意一个 $frame$ 不再存在 $PCube$, 则完成验证, 属性成立。

4 理论证明

IAPDR 算法的核心思想是将 PDR 算法运行在谓词抽象出的状态空间上。本章就 IAPDR 算法对隐式谓词抽象与 PDR 算法结合的正确性进行了理论证明。

IAPDR 在对抽象系统进行相关 $PCube$ 的反向搜索时, 使用式(4)的抽象版本——式(6)来确定某个 $PCube$ 与对应的 $frame$ 具有归纳相关性, 即对于抽象系统 \hat{S} 和某个 $PCube \hat{c}$ 及其对应的 $\hat{F}, \hat{S} \models indRel(\hat{F}, \hat{T}, \hat{c})$, 如定理 1 所示。

$$absIndRel(F, T, c, PSet) := F(X) \wedge \neg c(X) \wedge H_{PSet}(X, \bar{X}) \wedge T(\bar{X}, \bar{X}') \wedge H_{PSet}(\bar{X}', X') \wedge c(X') \quad (6)$$

定理 1 给定谓词集合 $PSet$ 、公式 F_{PSet} 和 $PCube_{c}$, 当且仅当 $absIndRel(F, T, c, PSet)$ 可满足时, $indRel(\hat{F}, \hat{T}, \hat{c})$ 可满足。当具体系统 $S \models absIndRel(F, T, c, PSet)$ 时, 抽象系统 $\hat{S} \models indRel(\hat{F}, \hat{T}, \hat{c})$ 。

证明: 假设 $S \models absIndRel(F, T, c, PSet)$ 。用 \bar{t} 表示 S 在集合 $\bar{X} \cup \bar{X}'$ 上的映射, t 表示 S 在集合 $X \cup X'$ 上的映射。因为 $\bar{t} \models T$, 所以 $\bar{t} \models \hat{T}$ 。因为 $S \models H_{PSet}(X, \bar{X}) \wedge H_{PSet}(\bar{X}', X')$, 且 \hat{t} 和 \bar{t} 具有相同的抽象迁移关系, 所以 $\hat{t} \models \hat{T}$ 。因为 $t \models F \wedge$

$\neg c$, 所以 $\hat{t} \models \hat{F} \wedge \neg \hat{c}$ 。因为 $t \models c'$, 所以 $\hat{t} \models \hat{c}'$ 。综上可得, $\hat{t} \models \hat{F} \wedge \neg \hat{c} \wedge \hat{T} \wedge \hat{c}'$, 又因为 $\hat{S} \models \hat{t}$, 所以, $\hat{S} \models \text{indRel}(\hat{F}, \hat{T}, \hat{c})$ 。

相对地, 假设 $\hat{t} \models \text{indRel}(\hat{F}, \hat{T}, \hat{c})$, 显然存在 t 是在集合 $X \cup X'$ 上的映射, 使得 $t \models T$ 且 $\hat{t} = \bar{t}$, 因此 $t \models \text{absIndRel}(F, T, c, PSet)$ 。

定理 2 给定系统模型 S 、待验证属性 P , IAPDR 返回 true, 则属性安全, 反之, 则属性不安全。IAPDR 返回的结果是正确的。

证明: IAPDR 维护的公式序列 F 满足如下条件:

$$1) F_0 = I;$$

$$2) F_i \models F_{i+1}, i < k;$$

$$3) F_i(X) \wedge H_{PSet}(X, \bar{X}) \wedge T(\bar{X}, \bar{X}') \wedge H_{PSet}(\bar{X}', X') \models F_{i+1}, i < k;$$

$$4) F_i \models P, i < k。$$

假设 IAPDR 返回 true。IAPDR 返回 true 的条件是 F 的某个 $frame$ 为空, 因为阻塞时会对冗余 $PCube$ 进行剔除, 所以上述条件等价于 $\hat{F}_k = F_{k+1}$, 因此 $\hat{S} \models \hat{P}$ 。又因为 2.4 节介绍的抽象的性质, 所以 $S \models P$, 即属性 P 是安全的。

假设 IAPDR 返回 false。IAPDR 返回 false 的条件是 BMC 模拟发现反例路径。抽象反例路径在 S 上的模拟不具有任何抽象特性, 任意满足抽象反例路径的状态 $s_k \models \neg P$, 就能确定属性 P 是不安全的。

5 实验与分析

本文使用由 Ran 等^[2]构造的 SCADE 数据集进行实验评估。本文实验运行在 64 位 Ubuntu 22.04 操作系统上, 处理器为 Intel i5-12400, 内存为 32GB, 编译器为 gcc-11.4.0。将

程序的验证超时时间门限设定为 240s, 超时时间门限内能够输出验证结果则认定为验证成功。验证结果中的 valid 表示程序的待验证属性是安全的, invalid 表示程序的待验证属性是不安全的, unknown 表示验证时间超过设定的门限。本文将 IAPDR 算法并行集成到 PSMC 工具中, 下文用 PSMCWI (PSMC With IAPDR) 表示集成后的工具。

PSMC 使用 Z3^[20]求解器作为唯一的逻辑公式求解器, 在 IAPDR 算法的实现中, 因为 Z3 求解器不支持插值计算功能, 所以引入 MATHSAT^[21]求解器, 将公式转化为 SMTLib2^[22]格式并作为 MATHSAT 的输入, 完成插值计算后再逆向转化为 Z3 格式的公式。

本文分别使用 PSMC 中集成的 BMC, K-Induction, CEGAR 以及 IAPDR 算法对 992 个 SCADE 程序进行模型检测, 其中 CEGAR 算法融合了抽象技术和 BMC 算法, 结果如表 1 所列。表 1 统计了 4 个不同引擎在单独验证 922 个 SCADE 程序时, 验证结果为 valid, invalid 和 unknown 的程序的量、验证总耗时和验证成功覆盖率。其中, 结果为 unknown 的程序的耗时记为 240s, 表中数据用 X/Y 表示数量/耗时。由表 1 可知, IAPDR 算法具有最高的验证成功覆盖率, 尤其是对属性安全的程序, IAPDR 具有明显的优势, 其验证总耗时也是最短的。BMC 和 CEGAR 算法仅能对属性不安全的 SCADE 程序进行验证, 而 K-Induction 和 IAPDR 对属性安全或不安全的 SCADE 程序都能进行验证。另外, IAPDR 对属性不安全的程序进行验证的效果弱于其他算法, 原因在于, IAPDR 获取反例的时机是在对抽象路径进行模拟的过程中, 且使用 BMC 算法进行模拟, 模拟前需要大量的搜索、阻塞操作, 而其他算法是在 BMC 算法未发现反例的情况下才进入下一算法步骤的。

表 1 算法对比

Table 1 Algorithms comparison

| Algorithm | Valid/s | Invalid/s | Unknown/s | TimeCost/s | Success-Rate/% |
|-------------|---------------|----------------|-------------|-------------|----------------|
| BMC | 0/0 | 355/105.6880 | 567/136.080 | 136185.6880 | 38.5 |
| CEGAR | 0/0 | 370/131.6710 | 552/132.480 | 132611.6710 | 40.1 |
| K-Induction | 303/17.051 | 355/110.9724 | 265/63.600 | 63728.0234 | 72.9 |
| IAPDR | 441/2.088.440 | 339/3.089.4910 | 142/34.080 | 39257.9310 | 84.6 |

本文分别使用 PSMCWI 和 PSMC 对 922 个 SCADE 程序进行模型检测。表 2 统计了在超时时间门限内, PSMCWI 和 PSMC 对数据集中不同类型的 SCADE 程序进行模型检测时, 验证成功的程序的数量。由表 2 可知, 相较于 PSMC, PSMCWI 能够多验证 139 个程序, 从数量上来看, 数据集验证成功的覆盖率提升了 15.1%。

表 2 PSMCWI vs. PSMC(数量)

Table 2 PSMCWI vs. PSMC(count)

| Program Class | 程序数量 | PSMC | PSMCWI | Gap |
|---------------|------|------|--------|------|
| functional | 26 | 5 | 8 | +2 |
| large | 96 | 69 | 85 | +16 |
| memory1 | 282 | 195 | 250 | +55 |
| memory2 | 182 | 152 | 176 | +24 |
| misc | 110 | 71 | 95 | +24 |
| protocol | 36 | 31 | 36 | +5 |
| simulation | 190 | 150 | 163 | +13 |
| 总和 | 922 | 673 | 812 | +139 |

除此之外, 还对 PSMCWI 和 PSMC 对 922 个 SCADE 程序的验证总耗时进行了统计, 如表 3 所列, 完成对 922 个程序的验证, PSMCWI 的耗时更短, 相比于 PSMC, 效率提升了 43%。

表 3 PSMCWI vs. PSMC(耗时)

Table 3 PSMCWI vs. PSMC(time cost)

| Tools | Count | TimeCost/s |
|--------|-------|------------|
| PSMC | 673 | 63754.0064 |
| PSMCWI | 812 | 36300.1110 |

Ran 等^[2]并未对数据集中属性是否安全给出标准答案, 所以本文针对上述的 139 个 SCADE 程序, 进行了进一步的实验, 以验证 IAPDR 算法的正确性, 下文用 OptSet 来表示这 139 个程序。由于目前没有其他的直接针对 SCADE 程序的模型检测工具, 而当前对 Lustre 程序进行模型检测的研究中, JKind 具有非常优秀的表现, 其对 Lustre 程序的验证结果

具有一定的权威性,且 Ran 等构造的数据集所涉及的 SCADE 语法均能等价地使用 Lustre 语法进行描述,所以本文将 PSMCWI 对 SCADE 程序的验证结果与 JKind 对等价的 Lustre 程序的验证结果进行对比,以证明 IAPDR 算法的正确性。

由表 3 可知,PSMCWI 和 JKind 对等价的 SCADE 和 Lustre 程序的验证结果完全一致,因此 IAPDR 算法能够对 SCADE 程序的安全属性进行正确的验证。另外,表 3 显示,OptSet 中的所有程序的验证结果均为 valid,原因在于,PSMC 工具所集成的 BMC 和 CEGAR 算法在处理属性 invalid 的程序时,具有不错的效果,但在处理状态空间较大且属性 valid 的程序时,通常会因为展开迁移关系而验证超时。而 IAPDR 引入了谓词抽象,缩小了状态空间,融合了 PDR,不展开迁移关系,使得 IAPDR 能够有效且快速地完成对 OptSet 中的 SCADE 程序的验证。

表 3 OptSet 验证结果对比

Table 3 Comparison of verification result for OptSet

| Tool | Total | Valid | Invalid |
|--------|-------|-------|---------|
| PSMCWI | 139 | 139 | 0 |
| JKind | 139 | 139 | 0 |

关于验证效率,统计了 PSMCWI 和 JKind 对等价的 922 个 SCADE 和 Lustre 程序进行验证时的时间消耗(包含超时的程序)。如图 2 所示,横坐标是程序的序号,纵坐标为时间消耗,其中,黑色数据点是 JKind 对 Lustre 程序进行验证的时间消耗,灰色数据点则是 PSMCWI 对 SCADE 程序进行验证的时间消耗。

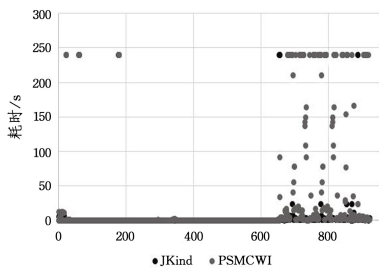


图 2 PSMCWI vs. JKind 散点图(电子版为彩图)

Fig. 2 PSMCWI vs. JKind scatter plot

由图 2 可知,对于大部分程序,两者的时间消耗差距并不大,由此可知,相比于将 SCADE 模型转化为 Lustre 模型后,用 JKind 对 Lustre 模型进行模型检测来实现对 SCADE 程序进行模型检测的方法,PSMCWI 具有更高的效率,因为转化过程也需要一定的时间开销。但 PSMCWI 在个别程序的验证上耗时较多,其原因有两点。第一,PSMCWI 和 JKind 在进行模型检测前,会对程序进行分析并转化为由逻辑公式表示的模型,相较于 Lustre,SCADE 的语法更为复杂,在转化阶段需要引入临时变量来替换复杂语法,尤其是复杂的程序得到的模型相较于 Lustre 会大很多,导致向求解器传递更长的公式,从而降低了求解器的效率。第二,IAPDR 额外引入了 MATHSAT 求解器来计算插值,每次计算都需要经历 Z3 格式、SMTLIB2 格式、MATHSAT 格式的公式间的转化,这也会消耗较多时间。

总体来看,PSMCWI 能够有效提升原生 PSMC 在数据集上的验证成功覆盖率,验证的正确性也能够得到保证,同时具有不错的验证效率。从算法的设计和实验上来看,IAPDR 能够有效解决因展开迁移关系而无法验证状态空间较大的程序所表现出的状态爆炸问题。

结束语 本文提出了一种基于隐式谓词抽象和属性导向可达的模型检测算法——IAPDR,并以并行引擎的形式扩展到 PSMC 上。得益于并行集成引擎、抽象精化、IC3 等优秀算法的思想,PSMCWI 在对数据集进行模型检测时,比 PSMC 多验证 139 个程序,在数据集上的验证成功覆盖率提升了 15.1%,数据集验证总耗时缩短了 43%。另外,使用 PSMC 的基本框架,使得对 SCADE 程序的验证过程可以全部自主实现,避免了对其他模型检测工具的强依赖,具有良好的可扩展性。除此之外,SCADE 官方提供的 SCADE Suite 工具可以对 SCADE 程序进行有效的模型检测,但该工具是一款价格昂贵的商业软件,且已经限制在国内使用,因此本文所做的工作,为解决技术封锁贡献了力量,同时也响应了自主可控的国家战略。在未来工作中,未来会针对当前 SCADE 数据集中无法在超时时间中被验证的 SCADE 程序进行分析,优化现有算法引擎或引入新的算法引擎来针对性地解决该问题。另外,当前扩展了 IAPDR 后的 PSMC 工具仍不能对 SCADE 的全部语法进行模型检测,我们会在 PSMC 工具上补充未覆盖的语法的分析、建模和验证功能,并构造相应的数据集用于实验。

参考文献

- [1] COLAÇO J L, PAGANO B, POUZET M. SCADE 6: A formal language for embedded critical software development[C]// 2017 International Symposium on Theoretical Aspects of Software Engineering (TASE). IEEE, 2017: 1-11.
- [2] RAN D, CHEN Z, SUN Y, et al. SCADE model checking based on program translation [J]. Computer Science, 2021, 48(12): 125-130.
- [3] GACEK A, BACKES J, WHALEN M, et al. The JKind model checker[C]// International Conference on Computer Aided Verification. Cham: Springer, 2018: 20-27.
- [4] WANG H, YANG Z, ZHOU Y, et al. AGVTS: Automated Generation and Verification of Temporal Specifications for Aeronautics SCADE Models[C]// International Symposium on Formal Methods. Cham: Springer, 2024: 338-355.
- [5] CAVADA R, CIMATTI A, DORIGATTI M, et al. The nuXmv symbolic model checker[C]// International Conference on Computer Aided Verification. Cham: Springer, 2014: 334-342.
- [6] LI T F, SUN J F, LYU X J, et al. Formal verification of synchronous reactive model for SMT based regional controller [J]. Journal of Software, 2023, 34(7): 3080-3098.
- [7] BARRETT C, TINELLI C. Satisfiability modulo theories[M]// Handbook of Model Checking. Cham: Springer, 2018: 305-343.
- [8] BASOLD H, GUNTHER H, HUHN M, et al. An open alternative for SMT-based verification of SCADE models[C]// International Workshop on Formal Methods for Industrial Critical Systems. Cham: Springer, 2014: 124-139.

- [9] BJESSE P,CLAESSEN K. SAT-based verification without state space traversal[C]// International Conference on Formal Methods in Computer-Aided Design. Berlin: Springer, 2000: 409-426.
- [10] DE MOURA L,RUEB H,SOREA M. Bounded model checking and induction:From refutation to verification[C]//International Conference on Computer Aided Verification. Berlin: Springer, 2003:14-26.
- [11] FANG Y Y,ZHANG C. SCADE model checking based on multi engine parallel collaboration[J]. Computer Technology and Development,2023,33(11):86-90.
- [12] CIMATTI A,GRIGGIO A,MOVER S, et al. IC3 modulo theories via implicit predicate abstraction[C]// International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Berlin:Springer,2014:46-61.
- [13] BRADLEY A R. SAT-based model checking without unrolling [C]//International Workshop on Verification, Model Checking, and Abstract Interpretation. Berlin:Springer,2011:70-87.
- [14] YU Z Q,ZHANG X Y,LI J W, et al. Approximate reachability analysis based on unsatisfiable kernels [J]. Journal of Software, 2023,34(8):3467-3484.
- [15] LI J,ZHU S,ZHANG Y, et al. Safety model checking with complementary approximations[C]//2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2017:95-100.
- [16] ZHANG X,XIAO S,XIA Y, et al. Accelerate safety model checking based on complementary approximate reachability [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2023,42(9):3105-3117.
- [17] MCMILLAN K L. Interpolation and SAT-based model checking [C]//International Conference on Computer Aided Verification. Berlin:Springer,2003:1-13.
- [18] CRAIG W. Linear reasoning:A new form of the Herbrand-Genzzen theorem[J]. Symbolic Logic,1957,22(3):250-268.
- [19] EEN N,MISHCHENKO A,BRAYTON R. Efficient implementation of property directed reachability[C]//2011 Formal Methods in Computer-Aided Design (FMCAD). IEEE, 2011: 125-134.
- [20] DE MOURA L,BJORNER N. Z3:An efficient SMT solver [C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008: 337-340.
- [21] CIMATTI A,GRIGGIO A,SCHAAF SMA B J, et al. The mathsat5 smt solver[C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer,2013:93-107.
- [22] COK D R. The smt-libv2 language and tools:A tutorial[J]. Language C,2011:14-55.



ZHANG Cong, born in 1999, postgraduate. His main research interests include model checking and software safety.



CHEN Zhe, born in 1981, professor, is a senior member of CCF (No. 22234S). His main research interests include formal methods, software verification and software engineering.

(责任编辑:何杨)