构件组合的集成测试

邓永杰,陈 颖
(上海大学 计算机工程与科学学院,上海 200072)

摘 要:模型检验是一种自动化验证技术,其应用主要的困难在于状态空间爆炸问题。针对构件组合形成的状态空间爆炸问题,结合构件抽象组合原理及反例引导的抽象精化框架,提出了一种测试用例自动生成的方法。根据某个待集成构件抽象已集成的其他构件,并通过组合各个抽象构件生成抽象组合模型。利用模型检验工具对组合模型进行集成测试,生成抽象测试用例,再通过精化得到原模型对应的具体测试用例。实验结果表明该方法减小了状态空间,在一定程度上减缓了状态空间爆炸的问题。

关键词:模型检验;构件组合;抽象精化;测试用例生成
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2013)07-0031-05
doi:10.3969/j.issn.1673-629X.2013.07.008

Component Composition Based Integration Test

DENG Yong-jie, CHEN Ying
(College of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)

Abstract: Model checking is an approach to formal automation verification, whose difficulties are due to the state explosion problem. On an account of the space state explosion in components composition, propose a test case generation method, which combines the compositional reasoning theory and the counterexample guided abstraction refinement framework. According to a integrated component abstract the other integrated components, then generate abstract composition model by combining all abstract components. Generate abstract test cases by tools, finally correct test cases are generated by refining abstract models repeatedly. The experimental results indicate that the proposed approach can alleviate the state space explosion to some extent.

Key words: model checking; component composition; abstraction refinement; test case generation

0 引 言

软件测试是在软件投入运行前,通过对软件的需求分析,设计规格说明和编码进行最终审查,以保证软件质量。软件的手动测试非常耗时,劳动强度高,并且非常单调,还会引入一些人为的错误,因此软件测试的自动化势在必行。

模型检验是一种自动验证有穷状态系统的技术,其基本思想为用时态逻辑表达系统的所期望的性质,用有限状态机FSM表示系统的状态转移结构,通过遍历FSM来检验性质是否满足;如果不满足,系统返回一个反例。从测试的角度来看,反例为构造测试用例提供了一种有效的途径。基于模型检验的测试是测试自动化的一个重要研究方向,根据测试准则如覆盖准则、变异等产生陷阱性质诱使模型检验器产生违背性质的反例,然后从反例中抽取测试的输入和预期的输出以构成测试用例[1]。

Ammann等[2]提出了一种使用形式化模型检验方法来生成测试集,或根据时序逻辑表达的安全性质来分析现有的测试集;Gargantini和Heitmeyer[3]描述了一种基于规范构建测试序列包的方法,通过生成一个包含软件规范描述的所有可能的行为的分支谓词的集合,结合模型检验工具生成反例;Zeng和Miao[4]提出了使用模型检验技术来测试web应用,提出了根据对象模型的节点和边覆盖准则来设计陷阱性质的算法;Hamon等[5]提出使用一种基于模型检验的改进的方法来生成有效的测试集。上述这些方法都是基于单个模型的检测,没有考虑多个构件的组合。同时由于模型检验技术是基于穷举搜索的,性能成为阻碍其发展的主要限制因素。基于模型检验的测试的关键技术在

收稿日期:2012-09-28 **修回日期**:2012-12-29 **网络出版时间**:2013-04-08
基金项目:国家自然科学基金资助项目(61073050);上海市自然科学基金(09ZR1412100);上海市重点学科建设项目基金(J50103)
作者简介:邓永杰(1986-),男,硕士研究生,研究方向为软件测试。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130408.1715.061.html

万方数据

于设计如何建模,以表示和遍历大规模的系统模型状态空间;以及解决由多个模型的并行组合而形成的状态空间爆炸问题。

抽象组合技术[6,7]作为应对模型状态空间爆炸问题的主要技术之一,其基本思想是通过去掉一些与验证性质无关的信息,减少模型的状态空间。值得注意的是,抽象具有性质保留性,即若抽象模型满足性质,则原来的模型也必须满足该性质。但是,抽象技术很可能会导致抽象模型引入系统中不存在的附加行为,即当抽象模型违背某一性质时,产生的反例可能只是抽象模型中引入的一个附加行为,具体模型并不违背该性质,这样的反例则称为伪反例(Spurious Counter-example)。因此,需要进行不断精化抽象模型以确保删除伪反例并更进一步检验性质。反例引导的抽象精化(Counterexample Guided Abstraction Refinement, CEGAR)[8]框架于 2000 年最初被 Clarke 提出,并在多种模型检验工具如 SLAM,BLAST 中实现。

集成测试又称为组装测试,即在满足单元测试的基础上,集成测试的关注点在于集成后的整体系统的正确性。构件化软件的集成测试[9,10]将构件单位组合成更大的系统,分析测试新组合系统的满足性。

文中在反例引导的抽象精化框架基础上,对全局整体模型的测试转化为基于抽象精化的构件集成测试,从一定程度上缓解状态空间爆炸问题。文中的主要任务包括形式化建模和测试用例生成及精化两方面。

1 形式化建模

通常基于模型检验的测试技术使用 Kripke 结构表示系统的状态空间。Kripke 结构是一种数学结构,它实际上是状态转移图的一种变形,可以利用它严格地定义线性时态逻辑(LTL)的语义。

定义 1 (Kripke 结构):Kripke 结构是一个五元组 $M=(S,\mathrm{Init},AP,L,T)$,其中:

(1) S 表示系统的一个有限状态集合;

(2) $\mathrm{Init}\subseteq S$ 表示系统的初始状态集合;

(3) AP 表示所有原子命题集合;

(4) $L:S\rightarrow 2^{AP}$ 为状态标记函数,把状态 $s\in S$ 映射为在状态 s 中成立的原子命题 AP 的一个子集;

(5) $T\subseteq S\times S$ 表示系统状态转移关系。

Kripke 结构的一条路径(Path)为一个无穷状态序列 $\pi=<s_1,s_2,s_3\cdots>$,其中 $\forall i\geqslant 1\cdot(s_i,s_{i+1})\in T$,当且仅当 $s_1\in\mathrm{Init}$ 时,路径即为一次执行;同时,定义 Path(M)为 M 的所有执行的集合。

由于模型检验是基于状态空间的穷举搜索,对于多个构件组合而成的并发系统,其状态数易随着并发状态的增加而呈指数增长,若直接对系统状态空间进行搜索,极易容易引起状态爆炸问题。该问题已经成为基于模型检验的软件测试技术的主要难题之一。许多减少和压缩状态技术被相继提出,例如符号模型检验、on-the-fly 模型检测、抽象方法等。其中抽象方法是有效实现集成和互补模型检验和定理证明两种形式化方法的方法[11]。

抽象的主要目的在于缩减模型的状态空间,一般通过状态抽象函数对模型的状态空间进行划分,每个划分对应一个抽象状态。存在性抽象是抽象中比较常用的一种方法,其基本思想为当且仅当两个抽象状态间分别存在一个具体状态,同时这两个具体状态在具体模型中存在迁移时,这两个抽象状态之间存在迁移[12]。对于由多个构件组成的系统,多个模块间通过原子命题集进行相互通信。

文中假定构件 $M_i(1\leqslant i\leqslant n-1)$ 是满足测试条件的已集成构件,待集成构件 M_n 不变,则已集成构件 M_i 可根据待测构件 M_n 构造抽象函数 h_i,生成相应的抽象模型 M_i^a。

定义 2(抽象函数):设 $M_i=(S_i,\mathrm{Init}_i,AP_i,L_i,T_i)(i\in[1,n-1])$ 是 $n-1$ 个已集成的构件,$M_n=(S_n,\mathrm{Init}_n,AP_n,L_n,T_n)$ 为待集成构件,$AP^a=(\bigcup_{1\leqslant j\leqslant n-1}AP_j)\cap AP_n$,则定义构件 M_i 的抽象函数 $h_i:S_i\rightarrow S_i{}^a$,满足:$\forall s,s'\in S_i,h_i(s)=h_i(s')$,当且仅当 $L(s)\cap AP^a=L(s')\cap AP^a$。

构件间的抽象函数主要是基于待测构件与已集成构件之间的原子命题集的公共部分来得到映射关系,从而对状态集进行抽象,生成相应的抽象模型。

定义 3(构件抽象):设 $M_i=(S_i,\mathrm{Init}_i,AP_i,L_i,T_i)(i\in[1,n-1])$ 是 $n-1$ 个已集成的构件,$M_n=(S_n,\mathrm{Init}_n,AP_n,L_n,T_n)$ 为待集成构件,$h_i:S_i\rightarrow S_i{}^a$ 为满足定义 2 的抽象函数,则 M_i 的抽象模型 $M_i^a=(S_i^a,\mathrm{Init}_i^a,AP^a,L_i^a,T_i^a)$,满足

$$\mathrm{Init}_i^a=\{s^a\mid\exists s\in\mathrm{Init}_i\wedge h_i(s)=s^a\}$$

$$T_i^a=\{(s_1^a,s_2^a)\mid\exists s_1,s_2\cdot T_i(s_1,s_2)\wedge h_i(s_1)=s_1^a\wedge h_i(s_2)=s_2^a\}$$

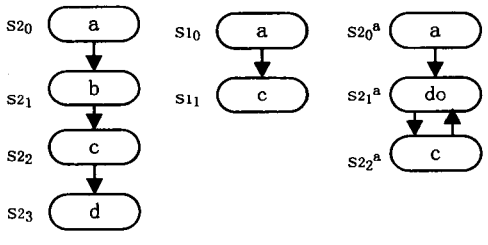
$\forall s^a\in S_i^a,L_i^a(s^a)=L_i(s)\cap AP^a$,其中,$h_i(s)=s^a$。

如图 1 所示,(a)为已集成的构件 1,(b)为待集成的构件 2,则根据待集成构件,可构造已集成构件相应的抽象函数对其进行抽象,得到抽象模型(c)。

模型检验技术可以自动验证抽象模型是否满足给定的性质,如果满足则报告正确;反之则生出抽象反例。假设 φ 为系统的一个 LTL 性质,M^a 为 M 的抽象模型,得出 $M^a\mid=\varphi\rightarrow M\mid=\varphi$,但是 $M^a\mid\neq\varphi\not\rightarrow M\mid\neq\varphi$。若抽象模型中存在路径 $\pi^a=<s_1{}^a,s_2{}^a,s_3{}^a\cdots>\in$

万方数据

Path(M^a),则模型中必须有与其对应的路径 $\pi = < s_1, s_2, s_3 \cdots > \in \text{Path}(M)$;反之如果抽象模型生成的反例没有与之对应的反例在原模型中,则称该反例为伪反例。



(a)构件1　　(b)构件2　　(c)抽象构件2

图 1　构件抽象

构件组合主要采用分而治之的基本思想,通过验证系统构件的局部性质,然后把这些性质组合起来以获得系统的全局性质。构件间的组合有很多种方式,主要可以分为异步并行组合和同步并行组合。两个构件异步并行组合即两个构件相互独立地执行,组合后的状态为两个构件状态的笛卡尔乘积;两个构件同步并行组合是指存在共享动作的两个构件接口自动机,即第一个构件的输入动作可以作为第二个构件的输出动作。文中主要采用构件的同步并行组合方式。

若构件间存在共享动作,则一个构件调用另一个构件的行为可以看作构件间的组合,构件间的共享动作变为组合模型的内部动作。一个构件的行为模型为 Kripke 结构 $M = (S, \text{Init}, AP, L, T)$,对于两个构件 M_a 和 M_b 当且仅当满足条件 $AP_a \cap AP_b \neq \varnothing$ 时才可进行组合。对于多个构件的组合问题可以进行类似定义。

定义 4(构件组合):若构件 $M_a = (S_a, \text{Init}_a, AP_a, L_a, T_a)$ 和 $M_b = (S_b, \text{Init}_b, AP_b, L_b, T_b)$ 可组合,则它们的同步组合构件 $M = M_a \,||\, M_b = (S, \text{Init}, AP, L, T)$,其中:

(1) $S = \{(s_a, s_b) \mid L_a(s_a) \cap AP_b = L_b(s_b) \cap AP_a\}$

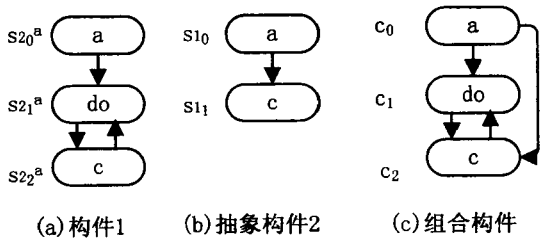
(2) $\text{Init} = (\text{Init}_a \times \text{Init}_b) \cap S$

(3) $AP = AP_a \cup AP_b$

(4) $L(s_a, s_b) = L_a(s_a) \cup L_b(s_b)$

(5) $(s_a, s_a') \in T_a, (s_b, s_b') \in T_b$ 时 $((s_a, s_b), (s_a', s_b') \in T)$

如图 2 所示为已集成的构件抽象模型(a)与待集成构件 1(b)生成的组合构件模型(c)。



(a)构件1　　(b)抽象构件2　　(c)组合构件

图 2　抽象构件组合

2　测试用例生成及精化

模型检验主要通过显示状态搜索或隐式不动点计算来验证有穷状态并发系统的性质,能自动地在系统不满足性质时提供违背性质的路径,即反例。值得注意的是,若要证明一个模型不满足某个性质时,一个反例足矣;但测试的目标在于尽可能多地覆盖系统,测试用例执行的不同行为越多越好。模型的测试用例为一个有穷状态序列,基于模型检验的测试技术可以通过各种覆盖准则获得测试用例集。一般来说,一个模型检验器用来判断抽象模型是否满足 LTL 性质,当模型检验器分析完所有可达的状态仍不存在违反性质的路径,则原模型也满足该性质。反之,当模型检验器发现一条违反性质的可达状态,反例则生成。

反例是一个从有效的初始状态开始至违背性质处的可达状态序列。由于抽象模型的行为多于原模型,必须判断反例是否为有效反例,若原模型中存在对应的具体执行迹,则该执行迹即为违背性质的具体反例,抽象反例是有效的;反之,抽象反例则为伪反例。

假设 $M = (S, \text{Init}, AP, L, T)$ 为原模型,$M^a = (S^a, \text{Init}^a, AP^a, L^a, T^a)$ 为其抽象模型,抽象模型中违反 LTL 性质的反例(π^a)为一个有穷迹或者无限循环的无穷迹。若 π^a 为有穷迹 $\pi^a = < s_1^a, s_2^a, \cdots, s_n^a >$,则每一个状态 s_i^a 对应原模型的等价类,因此 π^a 必定对应 M 中的一个具体执行迹的集合:$\text{CTrace}(\pi^a) = \{\langle s_1, s_2, \cdots, s_n\rangle \mid \bigwedge_{1 \leqslant i \leqslant n} (s_i \in S_i^a) \wedge (s_1 \in \text{Init}) \bigwedge_{1 \leqslant i \leqslant n-1} ((s_i, s_{i+1}) \in T)\}$,且 π^a 为有效的反例当且仅当 $\text{CTrace}(\pi^a) \neq \varnothing$[13]。

假设构件 $M_i = (S_i, \text{Init}_i, AP_i, L_i, T_i)(i = 1, 2)$,$\pi = \langle (s_1^1, s_1^2), (s_2^1, s_2^2), \cdots, (s_n^1, s_n^2)\rangle$ 为 $M_1^a \,||\, M_2^a$ 的执行迹,则 $\pi \downarrow M_i = \langle s_1^i, s_2^i, \cdots, s_n^i\rangle$ 为 M_i 中 π 的投影,且仅当 $(\text{CTrace}(\pi^a \downarrow M_1^a) \neq \varnothing) \wedge (\text{CTrace}(\pi^a \downarrow M_2^a) = \varnothing)$ 才满足 $\text{CTrace}(\pi^a) \neq \varnothing$[13]。因此抽象构件的组合模型可以通过 $M_i^a (1 \leqslant i \leqslant n)$ 中的迹投影来验证反例的有效性。

若反例为伪反例,必须精化抽象模型直至完全消除伪反例。精化算法主要通过不断重新抽象构件并组合新的抽象构件以生成新的抽象模型,直至抽象模型产生的违反性质的反例原模型也满足为止。

当使用模型检验进行验证时,任何能说明违反性质的反例都是合适的;但用于测试时,这并不准确,很可能会导致很多伪反例或者大量同样的测试用例,同时基于模型检验的测试技术还存在很大的性能问题,尤其是状态爆炸问题,状态空间的不断变大可能会导致测试用例不可行甚至更高的代价。文中在不实际构造系统的全局状态空间的情况下,将整体模型的抽象

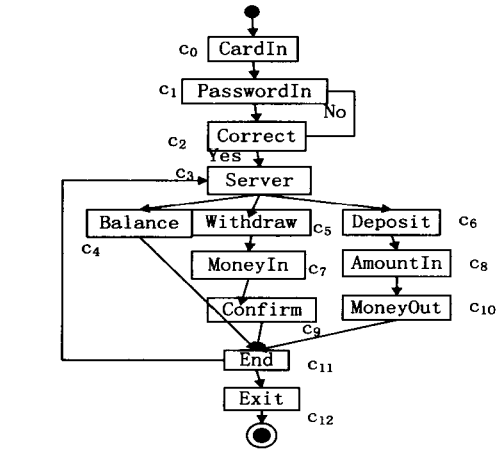
万方数据

精化转化为构件局部抽象精化;在反例引导的抽象精化以及构件组合的集成测试的基础上,为基于模型检验的测试用例的生成提供了一定的解决方法。

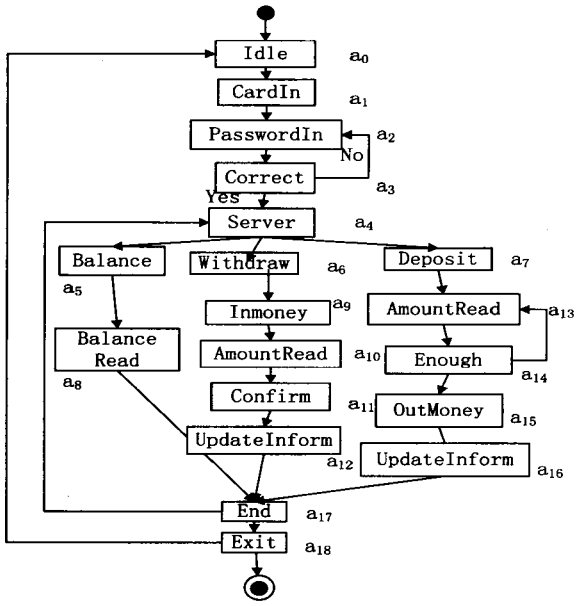
3 实例研究

文中主要通过 NuSMV 来生成测试用例,并以大家熟悉的 ATM 机交易模型为实例来说明基于模型检验的构件组合的测试用例的生成过程,且通过对比模型状态数的大小来说明抽象精化解决状态空间爆炸问题的效果。

使用 ATM 机进行交易已经成为生活中很频繁的一件事,文中将该系统交易过程从顾客(Client 端)和 ATM 机端分别进行建模,假定 ATM 机端构件满足测试,根据抽象组合原理生成抽象组合模型,根据测试准则进行集成测试,生成反例,最后对反例进行精化生成测试用例。Client 和 ATM 构件模型的 Kripke 结构图如图 3 所示。



(a)Client构件图

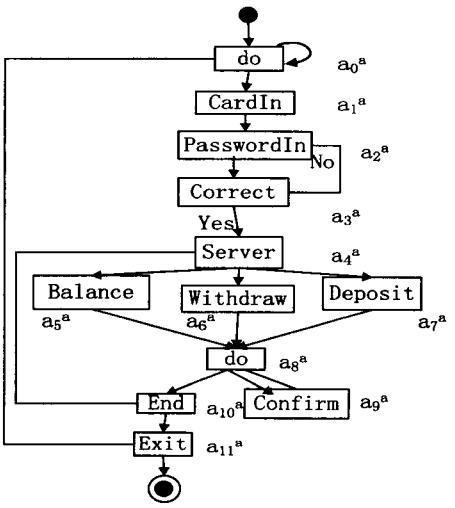


(b)ATM 构件图

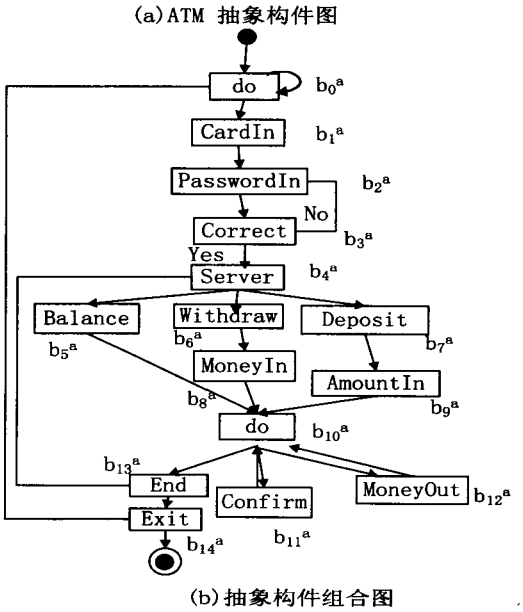
图 3 Client,ATM 构件

假定构件 ATM 是满足测试条件的构件,$AP_1 = AP_1^a$,Client 是待测构件,根据两个构件间的原子命题集的公共部分,得到抽象函数,对 ATM 构件进行抽象。从而得到了关于 ATM 构件的构件抽象模型,如图 4(a)所示,其中 $a_n^a = \{a_n\}(0 \leqslant n \leqslant 7)$;$a_8^a = \{a_8, a_9, a_{10}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}\}$;$a_9^a = \{a_{11}\}$;$a_{10}^a = \{a_{17}\}$;$a_{11}^a = \{a_{18}\}$。

根据组合规则,因 Client 构件与 ATM 构件满足组合条件 $AP_1 \cap AP_2 \neq \varnothing$,则生成组合模型如图4(b)所示,其中 $b_0^a = \{a_0^a, c\}$;$b_n^a = \{a_n^a, c_m\}(1 \leqslant n \leqslant 7, 0 \leqslant m \leqslant 6)$;$b_8^a = \{a_8^a, c_7\}$;$b_9^a = \{a_8^a, c_9\}$;$b_{10}^a = \{a_8^a, c\}$;$b_{11}^a = \{a_9^a, c_8\}$;$b_{12}^a = \{a_8^a, c_{10}\}$;$b_{13}^a = \{a_{10}^a, c_{11}\}$;$b_{14}^a = \{a_{11}^a, c_{12}\}$。



(a)ATM 抽象构件图



(b)抽象构件组合图

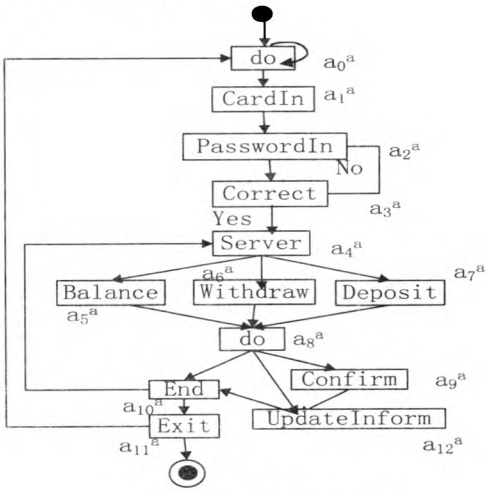
图 4 抽象组合构件图

根据边覆盖准则,如果选择 Client 构件中的一条边从 Confirm 到 End,可以得到一个包含这条边的陷阱性质 $\varphi = G(\text{MoneyIn} \rightarrow X(\text{Confirm} \rightarrow X\neg\ \text{End}))$。根据基于模型检验的测试,可以得到在组合模型中存在反例 $< b_0^a, b_1^a, b_2^a, b_3^a, b_4^a, b_6^a, b_8^a, b_{10}^a, b_{11}^a, b_{10}^a, b_{12}^a, b_{10}^a, b_{13}^a >$ 违背了该性质,其在 ATM 的抽象构件中的执行迹为 $< a_0^a, a_1^a, a_2^a, a_3^a, a_4^a, a_6^a, a_8^a, a_8^a, a_9^a, a_8^a, a_8^a, a_8^a, a_{10}^a >$,根据

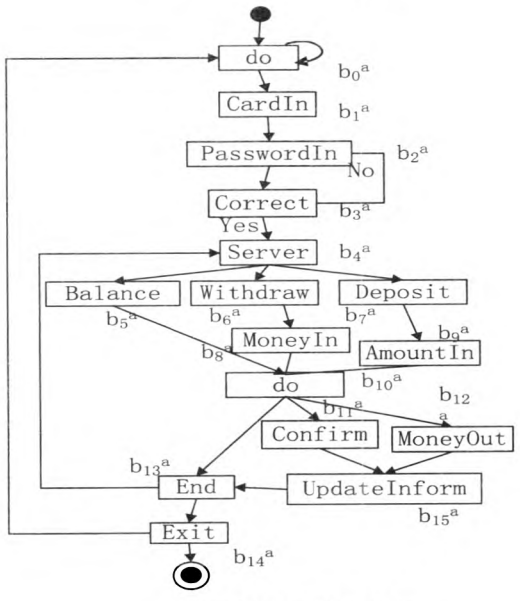
万方数据

CTrace($a_0^a, a_1^a, a_2^a, a_3^a, a_4^a, a_6^a, a_8^a, a_8^a, a_9^a, a_8^a, a_8^a, a_{10}^a$) = ∅, 可以得出结论该反例为一个伪反例。

对于伪反例,必须精化抽象组合模型直至完全消除伪反例。对 ATM 抽象构件进行精化,添加原子 UpdateInform 到集合 AP_2^a 中,得到精化抽象 ATM 模型如图 5(a)所示,其中 $a_n^a = \{a_n\}(0 \leqslant n \leqslant 7)$;$a_8^a = \{a_8, a_9, a_{10}, a_{13}, a_{14}, a_{15}\}$;$a_9^a = \{a_{11}\}$;$a_{10}^a = \{a_{17}\}$;$a_{11}^a = \{a_{18}\}$;$a_{12}^a = \{a_{12}, a_{16}\}$。



(a) ATM 抽象精化构件图



(b) 抽象精化构件组合图

图 5　抽象精化组合模型

同时再次生成组合模型如图 5(b)所示,$b_0^a = \{a_0^a, c\}$;$b_n^a = \{a_n^a, c_m\}(1 \leqslant n \leqslant 7, 0 \leqslant m \leqslant 6)$;$b_8^a = \{a_8^a, c_7\}$;$b_9^a = \{a_8^a, c_9\}$;$b_{10}^a = \{a_8^a, c\}$;$b_{11}^a = \{a_9^a, c_8\}$;$b_{12}^a = \{a_8^a, c_{10}\}$;$b_{13}^a = \{a_{10}^a, c_{11}\}$;$b_{14}^a = \{a_{11}^a, c_{12}\}$;$b_{15}^a = \{a_{12}^a, c\}$。可以得到违背性质 $\varphi = G(\text{MoneyIn} \rightarrow X(\text{Confirm} \rightarrow X\neg\ \text{End}))$ 的反例 $< b_0^a, b_1^a, b_2^a, b_3^a, b_4^a, b_6^a, b_8^a, b_{10}^a, b_{11}^a, b_{15}^a, b_{13}^a >$,该反例在 ATM 精化抽象构件中的执行迹为 $< a_0^a, a_1^a, a_2^a, a_3^a, a_4^a, a_6^a, a_8^a, a_8^a, a_9^a, a_8^a, a_{10}^a >$,相应的 CTrace($a_0^a, a_1^a, a_2^a, a_3^a, a_4^a, a_6^a, a_8^a, a_8^a, a_9^a, a_8^a, a_{10}^a$) ≠ ∅,因此 $< b_0^a, b_1^a, b_2^a, b_3^a, b_4^a, b_6^a, b_8^a, b_{10}^a, b_{11}^a, b_{15}^a, b_{13}^a >$ 为有效的反例,即根据测试准则 $\varphi = G(\text{MoneyIn} \rightarrow X(\text{Confirm} \rightarrow X\neg\ \text{End}))$ 可以得到一个对于 ATM 机交易模型的有效的测试用例 $< (a_0, c), (a_1, c_0), (a_2, c_1), (a_3, c_2), (a_4, c_3), (a_6, c_5), (a_9, c_7), (a_{10}, c), (a_{11}, c_8), (a_{12}, c), (a_{17}, c_{11}) >$。当然如果对于精化过的抽象组合模型仍存在伪反例,必须再次精化,不断重复,直至消除伪反例为止。

如图 6 所示,可以通过直接对比构件模型大小与精化抽象组合模型大小,很直观地发现抽象精化方法可以一定程度上缓解状态空间爆炸问题。

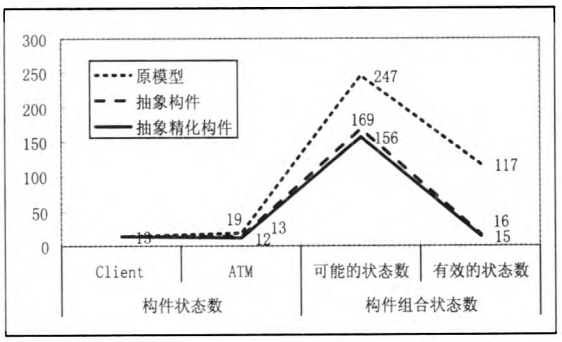


图 6　精化抽象组合模型与原模型大小对比

4　结束语

模型检验技术是基于穷举式搜索的,系统的有穷状态模型的状态数量往往会随其模型的并发分量的增加呈指数增长,极易导致状态空间爆炸。抽象是作为缓减状态空间爆炸问题的主要技术之一,通过去掉原来模型中与待验证性质无关的信息以获得简化模型的方法来减小模型的规模。反例引导的抽象精化(CEGAR)技术提出了不断精化抽象模型直至消除违反性质的伪反例。

文中介绍了一种基于模型检验的构件组合抽象精化的集成测试技术。根据某个待测构件,对其他满足单元测试的构件进行抽象后组合各个抽象构件;根据测试准则集成测试组合系统,生成反例,并通过验证各构件中反例的有效性以判断反例是否为伪反例;若为伪反例则不断精化抽象构件,直至完全消除伪反例,生成有效的测试用例。

接下来的主要工作为通过借助现有的模型检验工具,例如 SMV,SPIN 等等,开发一个通过构件组合抽象精化的集成测试,自动生成测试用例的工具。

参考文献:

[1] Chen Ying, Tian Ye, Zeng Hongwei. Test generation by using component composition abstraction refinement[C]//

(下转第 39 页)

万方数据

密度下的 NMSE 值精度明显高于另外三种算法,尤其在高密度噪声情况下,文中算法仍然具有较高精度的 NMSE 值,与主观视觉效果一致,显示出了算法稳定而高效的细节保持性能。

表 2　几种滤波方法的 NMSE 值比较

噪声密度	NMSE 值			
	传统中值	MLM+算法	Max-min 算法	文中算法
10%	0.0019	0.0013	0.0008	0.0002
20%	0.0048	0.0023	0.0011	0.0004
30%	0.0164	0.0040	0.0019	0.0008
40%	0.0470	0.0064	0.0040	0.0013
50%	0.1089	0.0100	0.0098	0.0022
60%	0.2142	0.0160	0.0233	0.0034
70%	0.3657	0.0267	0.0587	0.0078
80%	0.5617	0.0437	0.1326	0.0184

4　结束语

算法采用了开关策略,借鉴多级中值滤波算法思想,将滤波窗口划分纵横子窗口,并利用图像细节统计信息结合检测阈值减少了对噪声点的误判和漏判,去噪过程中充分利用了图像可用信号点信息,采用子窗口奇数信息点取中值、偶数信息点取均值的方法对噪声像素点进行恢复,使算法在图像去噪和边缘等细节保持方面均达到了十分理想的效果,具有较高的实用价值。

参考文献:

[1] Loupas T, Mcdicken W N, Allan P L. An Adaptive Weighted Median Filter for Speckle Suppression in Medical Ultrasonic Images[J]. IEEE Trans. on Circuits Syst., 1989, 36(1): 129-135.

[2] 宋焕生,梁德群,刘春阳. 一种新的自适应多级中值滤波器[J]. 信号处理, 1996, 12(4): 297-305.

[3] Brownrigg D R K. The weighted median filter[J]. Commun. Assoc. Computer, 1984, 27(8): 807-818.

[4] Wang J H, Lin L D. Improved median filter using min-max algorithm for image processing[J]. Electronics Letters, 1997, 33(16): 1362-1363.

[5] 陶剑锋,殷志祥,廖光洪. 基于模糊中值的图像处理方法[J]. 信息与电子工程, 2007, 5(5): 391-394.

[6] 邢藏菊,王守觉,邓浩江,等. 一种基于极值中值的新型滤波算法[J]. 中国图象图形学报, 2001, 6(6): 533-536.

[7] Nieminen A, Heinonen P, Neuvo Y. A new class of detail preserving filters for image processing[J]. IEEE Trans. on PAMI, 1987, 9(1): 74-90.

[8] Arce G R, Foster R E. Detail preserving ranked order based filters for image processing[J]. IEEE Trans. on ASSP, 1989, 37(1): 83-98.

[9] 吴昌东,江　桦,邱晓初. 基于多级中值滤波-提升小波技术的图像去噪[J]. 激光杂志, 2010, 31(6): 23-25.

[10] Gallagher N C, Wise G L. A theoretical analysis of the properties of the median filter[J]. IEEE Trans. on ASSP, 1981, 29(6): 1136-1141.

[11] 卢桂馥,王　勇,窦易文. 一种新的图像椒盐噪声的非线性滤波算法[J]. 计算机技术与发展, 2008, 18(1): 90-92.

(上接第 35 页)

Proceedings of the 2011 10th IEEE/ACIS International Conference on Computer and Information Science. Washington, D C: IEEE Computer Society, 2011: 307-311.

[2] Ammann P, Ding W, Xu D. Using a Model Checker to Test Safety Properties[C]//Proceedings of the Seventh International Conference on Engineering of Complex Computer Systems. Washington, D C: IEEE Computer Society, 2001: 212-221.

[3] Gargantini A, Heitmeyer C. Using model checking to generate tests from requirements specifications[C]//ESEC'99. [s. l.]: Springer-Verlag, 1999: 146-162.

[4] 曾红卫,缪淮扣. 一种验证 Web 应用设计的方法[J]. 上海大学学报(自然科学版), 2007, 13(5): 578-582.

[5] Hamon G, Hamon G E, de Moura E, et al. Generating Efficient Test Sets with a Model Checker[C]//2nd International Conference on Software Engineering and Formal Methods. Washington, D C: IEEE Computer Society, 2004: 261-270.

[6] 文艳军,王　戟,齐治昌. 并发反应式系统的组合模型检验与组合精化检验[J]. 软件学报, 2007, 18(6): 1270-1281.

[7] 胡　军,于笑丰,张　岩,等. 基于场景规约的构件式系统设计分析与验证[J]. 计算机学报, 2006, 29(4): 513-525.

[8] Clarke E, Grumberg O, Jha S, et al. Counterexample-guided abstraction refinement[C]//International Conference on Computer Aided Verification (CAV'00). Heidelberg: Springer-Verlag, 2000: 154-169.

[9] 奚和平. 基于构件的软件测试模型及方法[J]. 解放军理工大学学报(自然科学版), 2006, 7(3): 236-241.

[10] 张书杰,于学军,阎健卓,等. 基于构件软件系统集成测试的初步研究[J]. 北京工业大学学报, 2004, 30(2): 3-5.

[11] Rushby J. Using model checking with automated abstraction, invariant generation and theorem proving[C]//Proc of 5th and 6th International SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking. Heidelberg: Springer-Verlag, 1999: 1-11.

[12] Pnueli A. The Temporal Semantics of Concurrent Programs[C]//Proc of International Symposium on Semantics of Concurrent Computation. [s. l.]: [s. n.], 1979: 1-20.

[13] 曾红卫,缪淮扣. 构件组合的抽象精化验证[J]. 软件学报, 2008, 19(5): 1149-1159.

万方数据

构件组合的集成测试

作者： 邓永杰， 陈颖， DENG Yong-jie， CHEN Ying
作者单位： 上海大学计算机工程与科学学院,上海,200072
刊名： 计算机技术与发展
英文刊名： Computer Technology and Development
年，卷(期)： 2013,23(7)

本文链接：http://d.wanfangdata.com.cn/Periodical_wjfz201307008.aspx