一致闭的禁忌交互集生成算法

周吴杰[1),2)] 张德平[4)] 徐宝文[2),3)]

[1)](东南大学数学系 南京 210096)

[2)](南京大学软件新技术国家重点实验室 南京 210093)

[3)](南京大学计算机科学与技术系 南京 210093)

[4)](南京航空航天大学信息科学与技术学院 南京 210016)

摘 要 组合测试是侦测软件系统中各因素或配置之间是否有交互作用导致软件系统故障的重要方法,当因素之间的取值组合出现约束时如何生成尽可能少的测试用例是组合测试中的热点问题之一.该文研究了约束出现时由约束导致的禁忌交互集的结构形式,提出了包括所有的显性与隐含极小禁忌交互的一致闭的禁忌交互集的概念,对一般的禁忌交互集,提出了生成一致闭的禁忌交互集的算法,分析了算法的性能,然后对一致闭的禁忌交互集提出了生成禁忌覆盖表的类 AETG 算法.对 Cohen 等人提出的 5 个实际的测试场景以及 30 个人工合成的场景,实验表明生成的一致闭的禁忌交互集的规模是在可接受的范围内,生成的测试用例集规模与 Cohen 等人实验的结果是相当的.最后通过随机实验研究了影响禁忌交互集与其一致闭的禁忌交互集的规模比值的因素.

关键词 组合测试;约束条件;禁忌覆盖表;禁忌交互;一致闭禁忌交互集;类 AETG 算法

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2015.02279

The Generation Algorithm of Consistent Closed Forbidden Interaction Set

ZHOU Wu-Jie[1),2)] ZHANG De-Ping[4)] XU Bao-Wen[2),3)]

[1)](*Department of Mathematics, Southeast University, Nanjing* 210096)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

[3)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

[4)](*College of Information Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing* 210016)

Abstract In the paper, we study the combinatorial testing model that there are some constraints among the parameters or components in the software under testing. These constraints render some interactions invalid. The more forbidden interactions are implied by these explicit forbidden interactions. We study the structure of the forbidden interaction set and propose the concept of consistent closed forbidden interaction set. The algorithm to generate the consistent closed forbidden interaction set is provided. And the AETG-like algorithm to generate the combinatorial testing suites is proposed on the basic of the consistent closed forbidden interaction set. The experiments to observe the relations between the implied and explicit forbidden interactions are performed on the 35 testing scenes investigated by Cohen et al. At last, the additional examinations to look into that relation are inspected on some examples randomly synthesized.

Keywords combinatorial testing; constraints; forbidden interaction; consistent closed forbidden interaction; forbidden covering arrays; AETG-like algorithm

---

收稿日期:2012-02-25;最终修改稿收到日期:2015-03-26.本课题得到国家自然科学基金(90818027,91018005)、国家"八六三"高技术研究发展计划项目基金(2009AA01Z147)、国家"九七三"重点基础研究发展规划项目基金(2009CB320703)资助.**周吴杰**,男,1973 年生,博士,讲师,主要从事软件测试技术、软件可靠性等方面的科研工作.E-mail:zhouwujie@seu.edu.cn.**张德平**,男,1973 年生,博士,讲师,主要从事软件测试技术、软件可靠性、数理统计等方面的教学与科研工作.**徐宝文**,男,1961 年生,博士,教授,博士生导师,主要从事程序设计语言、软件工程、并行与网络软件等方面的教学与科研工作.

1 引 言

测试是软件或硬件开发过程中一个重要并且昂贵的环节.组合测试是侦测软件系统中各个因素或配置之间是否有交互作用导致软件系统故障的重要方法,该方法用尽可能少的测试用例来检测系统两个因素之间或 t 个因素之间是否有交互作用导致系统故障.检测任意两个因素之间所有的可能取值组合称为两两组合覆盖,任意 t 个因素之间所有的可能取值组合称为 t 维组合覆盖,t 称为覆盖强度. Kuhn 等人[1]的研究表明测试两两组合覆盖发现大约 70% 的错误,三维组合覆盖能发现大约 90%的错误.

人们应用组合测试策略进行软件测试已有很长时间,最早组合测试是应用正交拉丁方和正交实验设计来对软件进行测试[2-3].在组合测试的应用中,如何生成尽可能少的组合测试用例集是当前研究的热点问题之一,除了一些特殊的情形,一般生成达到覆盖要求的最优测试用例集是 NP 完全问题.人们通过数学构造,贪心算法以及智能搜索方法来生成尽可能小的测试用例集[4-13].然而在实践中软件系统的因素取值之间有很多约束,这些约束导致有些因素的某些取值不能同时出现,若同时出现在某个测试用例中,运行这个测试用例一定会导致软件系统故障,而在前述方法中很少有人讨论在有许多约束出现时的处理策略.而是否存在一条测试用例满足给定的约束已被 Bryce 和 Colbourn 证明是一 NP-hard 问题[14]. Cohen 等人[15-16]认真分析了当前一些生成组合测试用例集的工具中处理约束的情况,并得出结论:在实际测试场景中,约束对生成组合测试用例集会产生重要影响,在测试用例集生成算法中必须充分考虑相关约束.他们把 SAT 工具集成到 AETG 算法中,把组合测试问题编码成 SAT 问题,在每次选择下一个因素的一个取值时,把已生成的部分测试用例调用 SAT 解法器做一致性检查,看看是否能扩充成一个完全的测试用例[16].后来 Cohen 等人[17]又把 SAT 工具集成到模拟退火算法中来对约束的组合测试集生成问题求解,取得了比较好的结果. Danziger 等人[18]在二维情形下对生成禁忌覆盖表的复杂性问题进行了研究,得到最优的禁忌覆盖表的行数等于相应的图的最小边团覆盖数的结论.

本文研究了带约束的组合测试用例集生成问题,首先讨论了带禁忌交互的组合测试模型,然后对约束导致的禁忌交互集的内部结构进行了研究,提出了一致闭的禁忌交互集的概念,得到了判断禁忌交互集是一致闭的禁忌交互集的充分条件,在此基础上提出了一种将一般的禁忌交互集扩充成一致闭的禁忌交互集的算法,并分析了算法的性能,随后针对一致闭的禁忌交互集设计了生成禁忌覆盖表的类 AETG 算法.对于 Cohen 等人[16]提出的 5 个实际测试场景以及 30 个人工合成场景,通过实验表明其禁忌交互集扩充成一致闭禁忌交互集的规模在可接受范围内,从而可用一致闭的禁忌交互集来引导和生成组合测试用例集,生成的满足约束的覆盖表的规模与 Cohen 等人的实验相当.为了进一步研究禁忌交互集与其一致闭的禁忌交互集的规模比值与哪些因素有关,设计了一些随机实验进行了研究.最后总结了全文并提出未来可进行研究的工作.

2 相关研究以及带禁忌交互的组合测试模型

为了研究带禁忌边的组合测试的生成技术,我们先给出一些记号和形式化定义.

假设影响待测系统(Software Under Test,SUT)的因素共有 k 个,因素 i 有 $v_i(1\leq i\leq k)$ 个可能的取值,用 $0,1,\cdots,v_i-1$ 表示,并用记号 $[0,v_i-1]$ 表示取值域 $\{0,1,\cdots,v_i-1\}$.称 $\boldsymbol{V}=(v_1,v_2,\cdots,v_k)$ 为待测系统的因素可取值向量.

定义 1. 设 k 维向量 $\boldsymbol{T}=(T_1,T_2,\cdots,T_k)$,其中 $T_i\in[0,v_i-1]$,$i=1,2,\cdots,k$,则称这个 k 维向量 $\boldsymbol{T}$ 为第 i 个因素取值为 T_i 的测试用例.

定义 2. 任取 t 个因素,设集合 $I=\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$,其中因素 i_j 互不相同,$a_{i_j}\in[0,v_{i_j}-1](j=1,2,\cdots,t)$,则称这个集合 I 为一个 t 维交互.称集合 $f_I=\{i_1,i_2,\cdots,i_t\}$ 为交互 I 对应的因素集,一维交互 $\{(i_1,a_{i_1})\}$ 也称为顶点,简记为 (i_1,a_{i_1}),$t(t\geq 2)$ 维交互也称为 t 维边.

定义 3. 设一条测试用例 T 的第 i_j 个因素取值是 $a_{i_j}(j=1,2,\cdots,t)$,即 $T(i_j)=a_{i_j}$,则称这条测试用例 T 覆盖了 t 维交互 $I=\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$.

这样,一条测试用例覆盖了 $\binom{k}{t}$ 个 t 维交互.

为了生成组合测试用例集,我们定义覆盖表如下.

定义 4. 如果 A 是一个 $n\times k$ 表,表中第 i 列元素都取自 $[0,v_i-1]$,且满足:每个可能的 t 维交互都被表中某一行所对应的测试用例所覆盖,即对

任意的$\{(i_1,a_{i_1}),(i_2,a_{i_2}),\cdots,(i_t,a_{i_t})\}$，至少存在一行 r，使得 $A[r,i_j]=a_{i_j},j=1,2,\cdots,t$，则称 A 是一个 t 维混合覆盖表，记为 $MCA(n;t,(v_1,v_2,\cdots,v_k))$，$t$ 称为覆盖表的强度.

给定 t 和 $v_1,v_2,\cdots,v_k$，称使得 $MCA(n;t,(v_1,v_2,\cdots,v_k))$存在的最小的数 n 为混合覆盖表数，记为 $MCAN(t,(v_1,v_2,\cdots,v_k))$，当定义中的 $v_1=v_2=\cdots=v_k=v$ 时，我们简记为 $CA(n;t,k,v)$ 和 $CAN(t,k,v)$.

在不引起混淆的情况下覆盖表或混合覆盖表统称为覆盖表.

一般人们用覆盖表或混合覆盖表来产生测试用例集，表中每列对应一个因素，k 表示待测系统的因素数目，v_i表示每个因素可能取的取值个数，每一行表示一个测试用例，t 维覆盖表产生的测试用例集能覆盖到任意 t 个因素所有可能的取值组合，二维覆盖表产生的测试用例集称为两两组合测试用例集，人们希望在不降低测试标准的情况下来产生尽可能少的测试用例.

然而，有些测试场景中，有些因素的某些取值组合存在约束，例如 1 个移动电话系统有 5 个因素，显示因素有 3 个取值，0 表示 16M 彩显，1 表示 8M 彩显，2 表示黑白. 短信息因素中 0 表示支持彩信，1 表示只支持文本短信，2 表示不支持短信息[16]. 其他因素的相应取值由表 1 所示.

表 1　移动电话系统

显示	短信息	照相机	视频录像	视频铃声
0=16M 彩显 1=8M 彩显 2=黑白	0=彩信 1=文本短信 2=不支持	0=2 兆像素 1=1 兆像素 2=不支持	0=支持 1=不支持	0=支持 1=不支持

为了检测有哪些交互可能对系统的运行产生影响，在不考虑约束时我们可构造二维覆盖表来测试系统，如表 2 所示.

表 2　移动电话系统的无约束两两组合测试用例集

测试用例	显示	短信息	照相机	视频录像	视频铃声
1	0	2	1	0	0
2	1	1	2	1	1
3	0	1	0	1	0
4	2	2	2	1	0
5	1	2	0	0	1
6	0	0	2	0	1
7	2	1	1	0	1
8	1	0	1	1	0
9	2	0	0	0	0

然而，表 1 移动电话系统中并不是所有因素的所有可能的取值组合都是有效的，有些因素的某些取值组合是无效的，例如有如表 3 所示的一些约束以及由这些约束导致的无效交互.

表 3　约束和无效交互

约束	无效的交互
(1) 彩信支持需要彩显支持	{(1,2),(2,0)}
(2) 2 兆像素照相机需要彩显支持	{(1,2),(3,0)}
(3) 彩信不支持 2 兆像素的照相机	{(2,0),(3,0)}
(4) 8M 彩显不支持 2 兆像素的照相机	{(1,1),(3,0)}
(5) 视频录像需要照相机和彩显	{(3,2),(4,0)}, {(1,2),(4,0)}
(6) 不支持视频录像则不支持视频铃声	{(4,1),(5,0)}
(7) 不支持组合 16M 彩显，文本短信和 2 兆像素	{(1,0),(2,1), (3,0)}

所以一条测试用例中不能包含由这些约束导致的这些无效的交互，我们称这些由约束导致的无效的交互为禁忌交互或禁忌边.

定义 5.　设 I 是一个 t 维交互，如果任何一条覆盖 I 的测试用例都会导致系统故障或失效，则称交互 I 是一个禁忌交互或禁忌边.

我们在生成覆盖表时，必须考虑约束，有时在系统中有很少的约束可能导致生成的覆盖表中有很多行覆盖了禁忌交互. Cohen 等人[16]研究了 5 个实际的测试场景与 30 个人工构造的场景，如果不考虑约束用 AETG 算法来生成覆盖表，平均有 96%的行违反了约束，即至少覆盖了一个禁忌交互. 甚至有时约束会导致没有任何能够满足这些约束的测试用例.

假设在某个测试场景下，由外在约束导致某些交互是无效交互，我们称这些无效交互为显性禁忌交互，所有的显性禁忌交互构成的集合记为 Π，而由这些显性交互可以推导出来有些交互也是禁忌交互，若测试用例包含这些交互的某一个，则必定至少包含一个显性禁忌交互，称这些交互为显性交互集 Π 推导出的隐含禁忌交互，简称隐含禁忌交互. 假设某个 s 维交互是禁忌交互，则任何包含这个 s 维交互的交互都是禁忌交互，为此，我们称一个禁忌交互是极小禁忌交互，若其所有的真子集都不再是禁忌交互.

定义 6.　设显性禁忌交互集为 Π，称在 Π 的所有显性禁忌交互与隐含禁忌交互构成的集合中去掉非极小禁忌交互得到的交互集合为 Π 的一致闭禁忌交互集，记为$\overline{\Pi}$.

定义 7.　若一个显性禁忌交互集 Π 本身是一致闭禁忌交互集，即$\overline{\Pi}=\Pi$，则称禁忌交互集 Π 是一致闭的.

例如，对上述的移动电话系统，由约束导致的显性禁忌交互集为

Π={{(1,2),(2,0)},{(1,2),(3,0)},{(2,0),(3,0)},
{(1,1),(3,0)},{(3,2),(4,0)},{(1,2),(4,0)},
{(4,1),(5,0)},{(1,0),(2,1),(3,0)}}.

由这些交互可得到交互{{(2,1),(3,0)},{(2,1),(3,0),(4,0)},{(1,2),(5,0)},{(3,2),(5,0)}}是隐含禁忌交互,例如交互{(2,1),(3,0)},它不能扩充成一个满足这些约束的测试用例,因为对第 1 个因素没有办法赋值,若第 1 个因素赋值为 0,则覆盖了禁忌交互{(1,0),(2,1),(3,0)},若赋值为 1,则覆盖了禁忌交互{(1,1),(3,0)},若赋值为 2,则覆盖了禁忌交互{(1,2),(3,0)}.然而隐含禁忌交互{(2,1),(3,0)}导致了交互{(1,0),(2,1),(3,0)},{(2,1),(3,0),(4,0)}不再是极小禁忌交互,所以在前述的显性及隐含禁忌交互中把{(1,0),(2,1),(3,0)},{(2,1),(3,0),(4,0)}删除得到如下一致闭禁忌交互集,其中共 10 个禁忌交互:

$\overline{\Pi}$={{(1,2),(2,0)},{(1,2),(3,0)},{(2,0),(3,0)},
{(1,1),(3,0)},{(3,2),(4,0)},{(1,2),(4,0)},
{(4,1),(5,0)},{(2,1),(3,0)}, {(1,2),(5,0)},
{(3,2),(5,0)}}.

一个待测系统中,如何由显性的禁忌交互集生成其一致闭的禁忌交互集,其一致闭的禁忌交互集规模有多大?Cohen 等人研究了 35 个测试场景,共有 840 个显性二维禁忌交互,导致了 24 247 个隐含的二维禁忌交互[16].他们讨论的主要是隐含的二维禁忌交互.而不是讨论极小禁忌交互,我们研究表明,这 35 个测试场景中,共有 981 个显性交互,导致了 1897 个极小的禁忌交互,大致是显性交互的 2 倍,其中有 111 个一维隐含禁忌交互,这导致了二维禁忌交互的爆炸性增长.

为了研究带约束的组合测试问题,我们定义带禁忌交互集的覆盖表的概念.

定义 8. 设待测系统(SUT)中,禁忌交互集为 Π,一个 $n\times k$ 表 A,其第 i 列元素取自 $[0,v_i-1]$,$i=1,2,\cdots,k$,且满足:

(1) A 的每一行所对应的测试用例没有覆盖任何 Π 中的交互;

(2) 对任意 t 维交互,若这个 t 维交互不包含 Π 的任何显性或隐含禁忌交互,则都能被 A 的某一行所覆盖.

则称 A 是强度为 t 的禁忌覆盖表,记为 $FCA(n;t,(v_1,v_2,\cdots,v_k),\Pi)$.当定义中的 $v_1=v_2=\cdots=v_k=v$ 时,我们简记为 $FCA(n;t,k,v,\Pi)$.

注:定义中若禁忌交互集 Π 的一致闭的禁忌交互集 $\overline{\Pi}$ 已知,则条件(2)可陈述为:

(2′) 每个不包含 $\overline{\Pi}$ 中交互的任何 t 维交互,都能被 A 的某一行覆盖.

这样,由已知的约束,生成相应的禁忌交互集,根据这些禁忌交互集来生成相应的覆盖表,从而指导测试.

对上述的移动电话系统,生成满足约束的覆盖表如表 4.

表 4 移动电话系统的带约束两两组合测试用例集

测试用例	显示	短信息	照相机	视频录像	视频铃声
1	0	2	0	1	1
2	1	1	1	0	0
3	1	0	2	1	1
4	2	1	1	1	1
5	0	0	1	0	0
6	0	2	0	0	0
7	1	2	1	0	1
8	2	2	2	1	1
9	0	1	2	1	1

3 生成一致闭禁忌交互集的算法及生成禁忌覆盖表的类 AETG 算法

假设待测系统中,由约束导致的显性禁忌交互集为 Π,如何判断 Π 是否是一致闭的,对于非一致闭的禁忌交互集,如何生成一致闭禁忌交互集 $\overline{\Pi}$,其规模有多大?

定义 9. 假设禁忌交互集 Π 中有一组禁忌交互分别包含了某个因素的每个取值,例如:有 v_i 个禁忌交互分别包含了第 i 个因素的每个取值:$I_1=\{(i,0),(i_{11},a_{11}),\cdots,(i_{1t_1},a_{1t_1})\}$,$I_2=\{(i,1),(i_{21},a_{21}),\cdots,(i_{2t_2},a_{2t_2})\},\cdots,I_{v_i}=\{(i,v_i-1),(i_{v_i1},a_{v_i1}),\cdots,(i_{v_it_{v_i}},a_{v_it_{v_i}})\}$,由这些交互得到集合 $I=\{(i_{11},a_{11}),\cdots,(i_{1t_1},a_{1t_1}),(i_{21},a_{21}),\cdots,(i_{2t_2},a_{2t_2}),\cdots,(i_{v_i1},a_{v_i1}),\cdots,(i_{v_it_{v_i}},a_{v_it_{v_i}})\}$.

在集合 I 中删除相同的元素得到新集合,仍记为 I,如果 I 中不存在两点在相同的因素下取值不同,即不包含两点 $(j,a_1),(j,a_2)(a_1\neq a_2)$,则 I 是一个交互,这时称 $I_1,I_2,\cdots,I_{v_i}$ 是可归结禁忌交互组,交互 I 是由一组交互 $I_1,I_2,\cdots,I_{v_i}$ 归结出来的交互,记为 $\mathrm{Resolu}(I_1,I_2,\cdots,I_{v_i})=I$,如果 $I=\varnothing$,即 $I_1,I_2,\cdots,I_{v_i}$ 中每个交互都是单点集,这时可得到 Π 是不可满足的禁忌交互集.若 I 包含两点 $(j,a_1),(j,a_2)(a_1\neq a_2)$,则称交互组 $I_1,I_2,\cdots,I_{v_i}$ 不可归结.

注:上述的归结推理本质上就是命题逻辑中的

归结推理的推广.

根据归结运算，显然有：

定理 1.　设禁忌交互集为 Π，则由 Π 中可归结禁忌交互组归结出来的交互一定是禁忌交互.

要对禁忌交互集 Π 做归结运算，必须找到某个因素 i，使得因素 i 的每个取值都必须在某个交互中出现，即对任意的(i,a)，$(a\in[0,v_i-1])$，都存在交互 $I\in\Pi$，使得$(i,a)\in I$，称这种因素为可归结因素，所有的可归结因素构成的集合称为 Π 的可归结因素集，记为 $FactorsOfResolu(\Pi)$. 我们要生成一致闭的交互集，必须首先找到可归结因素集，这个集合当然越小越好.

显然有：

定理 2.　如果禁忌交互集 Π 中不包含非极小禁忌交互，并且其可归结因素集 $FactorsOfResolu(\Pi)=\varnothing$，即每个因素中至少有一个取值没有被包含在某个禁忌交互中，则禁忌交互集 Π 是一致闭的.

有了归结运算，对于显性禁忌交互集 Π，能否通过反复应用归结操作，得到所有的隐含禁忌交互. 首先我们证明对在归结操作下不变的极小禁忌交互集，一定是一致闭的禁忌交互集.

定理 3.　假设禁忌交互集 Π 中不含非极小禁忌交互，并且所有的可归结禁忌交互组所归结出来的交互都至少包含 Π 中的一个交互，即关于归结运算封闭，则禁忌交互集 Π 是一致闭的.

证明.　如果 $\Pi=\{\varnothing\}$，则所有的交互都是禁忌交互，显然 Π 是一致闭的.

否则只需证明对任意的交互 I，I 不包含 Π 中的任何交互，则 I 不是禁忌交互.

设 $I=\{(i_1,a_1),(i_2,a_2),\cdots,(i_t,a_t)\}$，要证明存在一条测试用例 T 覆盖了 I 但没有覆盖 Π 中任何交互. 如下构造测试用例 T：

第 1 步：$T(i_1)=a_1,T(i_2)=a_2,\cdots,T(i_t)=a_t$；

第 2 步：对余下的因素随机排列 $i_{t+1},i_{t+2},\cdots,i_n$，假设前面 j 个因素的取值都已确定，即 $T(i_1)=a_1,T(i_2)=a_2,\cdots,T(i_t)=a_t,T(i_{t+1})=a_{t+1},\cdots,T(i_j)=a_j$ 确定第 $j+1$ 个因素的取值 $T(i_{j+1})=a_{j+1}$ 时，要求测试用例中已赋值部分没有覆盖 Π 中任何交互，即使得交互 $\{(i_1,a_1),(i_2,a_2),\cdots,(i_j,a_j),(i_{j+1},a_{j+1})\}$ 不包含 Π 中任何交互.

这样构造出来的测试用例一定不包含 Π 中任何交互. 且第 2 步内的循环能够执行到最后构造出完整的测试用例. 假设已经确定了前面的 j 个因素的赋值，满足：测试用例的已赋值部分未覆盖 Π 中任何交互. 在确定第 $j+1$ 个因素的取值时，满足条件的取值一定存在，若不然，假设满足条件的取值不存在，即对任意的

$a\in[0,v_{j+1}-1]$，都存在 Π 中交互 $I_a=\{(i_{j+1},a),(i_{s_1},T(i_{s_1})),\cdots,(i_{s_t},T(i_{s_t}))\}$ 被测试用例已赋值部分所覆盖. 则这组交互 $I_a(a=0,1,\cdots,v_{j+1}-1)$ 是可归结的，设归结出的交互为 $\overline{I}$，因为 Π 关于归结运算是封闭的，所以在 Π 中至少存在一个交互 $I'\in\Pi$，使得 $I'\subseteq\overline{I}$. 然而 I' 被测试用例的前面 j 个因素的赋值所覆盖，与前面假设测试用例前 j 个因素的赋值部分未覆盖 Π 中任何交互矛盾.　　证毕.

由定理 3 证明可知，任何一个交互，若这个交互没有包含一致闭的禁忌交互集中任何交互，则可用证明中的方法把这个交互扩充成一个不包含任何禁忌交互的测试用例.

有了定理 1 和定理 3，得到算法 1.

算法 1.　生成一致闭的禁忌交互集算法.

输入：k 个参数的可取值向量$(v_1,v_2,\cdots,v_k)$，显性禁忌交互集 Π

输出：一致闭的禁忌交互集$\overline{\Pi}$

0. 初始化，在 Π 中删除所有非极小交互，对新得到的交互集仍记为 Π.

1. 对禁忌交互集 Π 计算可归结因素集 $FactorsOfResolu(\Pi)$，并令 $\Pi_1=\Pi$.

2. 对集合 $FactorsOfResolu(\Pi)$ 中每个因素 i 所对应的每一组可归结的禁忌交互组 $I_1,I_2,\cdots,I_{v_i}$.

计算归结 $I=Resolu(I_1,I_2,\cdots,I_{v_i})$，如果 $I=\varnothing$，输出“禁忌交互集 Π 是不可满足的”，输出$\overline{\Pi}=\{\varnothing\}$，结束；否则把 I 并入到 Π_1.

3. 在步 2 中生成的 Π_1 中删除掉所有的非极小交互，删除后的交互集仍记为 Π_1.

4. 如果 $\Pi_1\neq\Pi$，则令 $\Pi=\Pi_1$，转入步 1；否则令$\overline{\Pi}=\Pi_1$，输出$\overline{\Pi}$，结束.

定理 4.　算法 1 运行有限步后结束.

证明.　因为每个交互若在步 3 被删除掉，则在以后的循环中不可能再出现在步 4 的 Π_1 中，所以每轮循环中步 4 中的 Π_1 互不相同，而系统中交互集合的数目是有限的，所以算法在有限步结束.　　证毕.

定理 5.　算法 1 输出的是包含显性禁忌交互集 Π 的一致闭的禁忌交互集$\overline{\Pi}$.

证明.　由定理 1，在步 2 中每次并入到 Π_1 中的都是隐含禁忌交互，所以最后输出的$\overline{\Pi}$中交互都是禁忌交互，并且由步 3，$\overline{\Pi}$ 中不含非极小交互，由步 4，$\overline{\Pi}$ 关于归结运算封闭，所以根据定理 3，$\overline{\Pi}$ 是一致闭的.　　证毕.

注：算法的时间复杂性是指数级的，它是命题逻

辑中归结推理的推广,输出的$\overline{\Pi}$中禁忌交互数目有可能规模较大,但在一些特殊情况下,我们有:

定理 6. 如果禁忌交互集 Π 的可归结因素集 $FactorsOfResolu(\Pi)=\{i\}$,即只有一个因素 i 其每个取值都至少被包含在一个禁忌交互中,设其包含取值 v 的禁忌交互的数目为 $n_v(v=0,1,\cdots,v_i-1)$,则算法 1 得到的一致闭的禁忌交互集$\overline{\Pi}=\{\varnothing\}$或者比 Π 中的交互数目最多增加 $n_0 n_1\cdots n_{v_i-1}$ 个交互.

定理 7. 如果禁忌交互集 Π 的可归结因素集 $FactorsOfResolu(\Pi)=\{i_1,i_2,\cdots,i_r\}$,即共有 r 个因素,称某个因素的某个取值(顶点)与 $FactorsOfResolu(\Pi)$关联,如果在 Π 中存在某个交互 I 包含了这个因素的这个取值也包含了 $FactorsOfResolu(\Pi)$中的某个因素的某个取值,记所有的关联顶点集 Γ,即 $\Gamma=\{(i,u),u\in[0,v_i-1]|i\in FactorsOfResolu(\Pi),$或 $\exists j\in FactorsOfResolu(\Pi),v\in[0,v_j-1]\},I\in\Pi$ 使得$(i,u),(j,v)\in I\}$设 Γ 中元素数目为 $n=|\Gamma|$,则算法 1 中生成的一致闭禁忌交互集$\overline{\Pi}=\{\varnothing\}$或者比 Π 中交互数目最多增加 $\left(\begin{matrix}n\\ \lfloor\frac{n}{2}\rfloor\end{matrix}\right)$ 个交互.

证明. 因为每次执行归结操作时生成的归结交互都是关联集合 Γ 的子集,所以最后生成的$\overline{\Pi}=\{\varnothing\}$或者$\overline{\Pi}\backslash\Pi$ 是 Γ 的子集. 若$\overline{\Pi}\backslash\Pi$ 是 Γ 的子集,又因为$\overline{\Pi}\backslash\Pi$ 中元素不互相包含,所以$\overline{\Pi}$是集合 Γ 中的一条反链,根据 Sperner 定理[19],Γ 中最长的反链的长度为 $\left(\begin{matrix}n\\ \lfloor\frac{n}{2}\rfloor\end{matrix}\right)$,所以$\overline{\Pi}$比 Π 中交互数目最多多 $\left(\begin{matrix}n\\ \lfloor\frac{n}{2}\rfloor\end{matrix}\right)$ 个交互. 证毕.

注:由定理 6,7 可知,对显性禁忌交互 Π,要使得生成的$\overline{\Pi}$的规模不太大,必须或者 $FactorsOfResolu(\Pi)$中因素不太多或者对其关联集合 Γ 中元素较少,才能使得算法 1 效率较高.

定理 8. 如果禁忌交互集 Π 中每个交互都是二维的,并且每个因素都是二水平的,即 $v_1=v_2=\cdots=v_k=2$,则算法 1 生成一致闭的禁忌交互集$\overline{\Pi}=\{\varnothing\}$或者其交互数目最多为 $2k(k-1)$.

证明. 因为每次执行归结操作时生成的归结交互是二维或一维的或者是空集,所以$\overline{\Pi}=\{\varnothing\}$或者最多等于所有二维交互数目,即 $2k(k-1)$.

注:定理 8 其实等价于命题可满足性问题(SAT)中 2-SAT 问题.

以上讨论了一致闭的禁忌交互集的概念以及生成一致闭的禁忌交互集的算法.

在一般的待测系统中,由显性禁忌交互集通过归结运算生成一致闭的禁忌交互集,然后就可以设计类似的 AETG 算法来得到禁忌覆盖表,与传统的 AETG 算法不同的是:

(1)在初始化阶段,计算未被覆盖的交

互集 Uncover 时,在所有的 t 维交互中删除掉包含了一致闭禁忌交互集中某个交互的 t 维交互.

(2)在生成一条候选测试用例时,假设前 $j-1$ 个参数的取值已确定,在确定第 j 个参数的取值时,要求选取的取值与测试用例的已赋值部分构成的部分测试用例没有覆盖任何$\overline{\Pi}$中的交互并且覆盖了 Uncover 中最多的 t 维交互.

具体算法如下.

算法 2(类 AETG 算法生成禁忌覆盖表).

输入:$t,(v_1,v_2,\cdots,v_k)$,一致闭禁忌交互集$\overline{\Pi}$

输出:禁忌覆盖表 $FCA(n;t,(v_1,v_2,\cdots,v_k),\overline{\Pi})A$

```
begin
  初始化表 A 为空表;
  如果∅∈Π̄, 则输出 A 为空表;
  生成所有的未被覆盖的 t 维交互集 Uncover,初始化
      为所有的 t 维交互集;
  在Uncover 中删除掉所有的包含Π̄中某个交互的 t 维
    交互,仍记删除后的集合为 Uncover;
  初始化空表 A;
  while (Uncover 不为空)
    选择t-1 维{(i_1,a_{i_1}),(i_2,a_{i_2}),…,(i_{t-1},a_{i_{t-1}})},
        使得{(i_1,a_{i_1}),(i_2,a_{i_2}),…,(i_{t-1},a_{i_{t-1}})}在
        Uncover 中出现次数最多;
    随机排序 i_1,i_2,…,i_{t-1};
    for tt=1 to M          //生成 M 条新的候选测试
      //用例,其他参数随机排列为 i_t,i_3,…,i_k;
      for j=t to k
        //此时前 j-1 个参数的取值已确定,分为
        //a_{i_1},a_{i_2},…,a_{i_{j-1}}
        选取第 j 个因素的取值 a_j, 使得 a_j 与测试
          用例的已赋值部分构成的部分测试用
          例没有覆盖Π̄中的任何交互并且覆盖
          了 Uncover 中最多的 t 维交互;
      endfor
    endfor
        从 M 条测试用例中选择一条测试用例 T,
          使得 T 覆盖了 Uncover 中最多未被
          覆盖的交互,将其作为一行并到 A 的
          下面,得到新的表 A;
        在 Uncover 中删除所有被 T 覆盖的 t 维交
          互,仍记为 Uncover;//更新 Uncover
  endwhile
    return A;
end
```

注：由定理 3 中的证明，可以知道算法 2 正确地返回一个禁忌覆盖表.

4　实验研究

实验 1.　Cohen 等人[16]研究的 35 个测试场景.

我们对 Cohen 等人研究的 35 个测试场景进行研究，见表 5，对场景中的显性禁忌交互集运用算法 1 得到一致闭禁忌交互集，其禁忌交互数目对比如表 5 所示，其中 CA 模型栏中的记号如 $2^{189}3^{10}$ 表示待测系统有 199 个因素，其中 189 个因素有 2 个取值，10 个因素有 3 个取值，禁忌交互记号如 $2^{37}3^{3}$ 表示待测系统中有 40 个禁忌交互，其中有 37 个是二维交互，3 个是三维交互，$\overline{\Pi}/\Pi$ 这一栏表示一致闭禁忌交互数目与显性禁忌交互数目的比值. 从表 5 中可以看出，35 个场景中，$\overline{\Pi}/\Pi$ 比最大的是 5.0930，有 8 个测试场景的数目比小于 1，最小的是 0.3333，有 3 个测试场景的一致闭禁忌交互集与显性禁忌交互集相等. 35 个测试场景中，共有 981 个显性交互，生成的一致闭的禁忌交互集中交互数为 1897，$\overline{\Pi}/\Pi$ 比平均值是 1.9337，可以看出生成的极小禁忌交互数目在可接受范围内. 在 $\overline{\Pi}/\Pi$ 比小于 1 的测试场景中有 7 个其一致闭禁忌交互集中出现一维禁忌交互，并且所有的 1897 个极小的禁忌交互中有 111 个一维禁忌交互，这些一维禁忌交互的出现可能导致了隐含的极小禁忌交互数目不是太快的增长.

表 5　显性禁忌交互集与一致闭的禁忌交互集数目对比

场景	CA 模型	显性 Π	一致闭 $\overline{\Pi}$	$\overline{\Pi}/\Pi$	场景	CA 模型	显性 Π	一致闭 $\overline{\Pi}$	$\overline{\Pi}/\Pi$
SPIN-S	$2^{13}4^{5}$	2^{13}	2^{13}	1	14	$2^{81}3^{5}4^{3}6^{3}$	$2^{13}3^{2}$	$1^{3}2^{7}3^{1}$	0.7333
SPIN-V	$2^{42}3^{2}4^{11}$	$2^{47}3^{2}$	$2^{56}3^{10}$	1.3469	15	$2^{50}3^{4}4^{1}5^{2}6^{1}$	$2^{20}3^{2}$	$1^{1}2^{25}3^{33}4^{32}$	4.1364
GCC	$2^{189}3^{10}$	$2^{37}3^{3}$	2^{39}	0.9750	16	$2^{81}3^{3}4^{2}6^{1}$	$2^{30}3^{4}$	$1^{12}2^{4}$	0.4706
Apache	$2^{158}3^{8}4^{4}5^{1}6^{1}$	$2^{3}3^{1}4^{2}5^{1}$	$2^{3}3^{1}4^{2}5^{1}$	1	17	$2^{128}3^{3}4^{2}5^{1}6^{3}$	$2^{25}3^{4}$	$2^{66}3^{33}$	3.4138
Bugz.	$2^{49}3^{1}4^{2}$	$2^{4}3^{1}$	$2^{4}3^{1}$	1	18	$2^{127}3^{2}4^{4}5^{6}6^{2}$	$2^{23}3^{4}4^{1}$	$2^{28}3^{8}4^{2}$	1.3571
1	$2^{86}3^{3}4^{1}5^{5}6^{2}$	$2^{20}3^{3}4^{1}$	$1^{2}2^{35}3^{31}$	2.8333	19	$2^{172}3^{9}4^{9}5^{3}6^{4}$	$2^{38}3^{5}$	$2^{114}3^{69}4^{36}$	5.0930
2	$2^{86}3^{3}4^{3}5^{1}6^{1}$	$2^{19}3^{3}$	$1^{1}2^{35}3^{10}$	2.0909	20	$2^{138}3^{4}4^{5}5^{4}6^{7}$	$2^{42}3^{6}$	$1^{9}2^{46}3^{21}$	1.5833
3	$2^{27}4^{2}$	$2^{9}3^{1}$	$2^{14}3^{5}$	1.9000	21	$2^{76}3^{3}4^{2}5^{1}6^{3}$	$2^{40}3^{6}$	$1^{11}3^{35}$	1.0000
4	$2^{51}3^{4}4^{2}5^{1}$	$2^{15}3^{2}$	$1^{3}2^{15}3^{3}$	1.2353	22	$2^{73}3^{3}4^{3}$	$2^{31}3^{4}$	$1^{2}2^{50}3^{5}$	2.5909
5	$2^{155}3^{7}4^{3}5^{5}6^{4}$	$2^{32}3^{6}4^{1}$	$2^{73}3^{19}4^{18}5^{2}$	2.8718	23	$2^{25}3^{1}6^{1}$	$2^{13}3^{2}$	$1^{3}2^{13}3^{1}$	1.1333
6	$2^{73}4^{3}6^{1}$	$2^{26}3^{4}$	$1^{11}2^{7}$	0.6000	24	$2^{110}3^{2}5^{3}6^{4}$	$2^{25}3^{4}$	$1^{2}2^{45}$	1.6207
7	$2^{29}3^{1}$	$2^{13}3^{2}$	$1^{4}2^{1}$	0.3333	25	$2^{118}3^{6}4^{2}5^{2}6^{6}$	$2^{23}3^{3}4^{1}$	$2^{52}3^{8}4^{12}5^{2}$	2.7407
8	$2^{109}3^{2}4^{2}5^{3}6^{3}$	$2^{32}3^{4}4^{1}$	$1^{4}2^{53}3^{1}$	1.5676	26	$2^{87}3^{1}4^{3}5^{4}$	$2^{28}3^{4}$	$1^{3}2^{49}3^{27}$	2.4688
9	$2^{57}3^{1}4^{1}5^{1}6^{1}$	$2^{30}3^{7}$	$1^{14}2^{2}$	0.4324	27	$2^{55}3^{2}4^{2}5^{1}6^{2}$	$2^{17}3^{3}$	$2^{43}3^{7}4^{4}$	2.7000
10	$2^{130}3^{6}4^{5}5^{2}6^{4}$	$2^{40}3^{7}$	$1^{6}2^{65}3^{46}$	2.4894	28	$2^{167}3^{16}4^{2}5^{3}6^{6}$	$2^{31}3^{6}$	$2^{75}3^{73}4^{11}$	4.2973
11	$2^{84}3^{4}4^{2}5^{2}6^{4}$	$2^{28}3^{4}$	$1^{3}2^{51}3^{16}$	1.2875	29	$2^{134}3^{7}5^{3}$	$2^{19}3^{3}$	$2^{32}3^{17}4^{3}$	2.3636
12	$2^{136}3^{4}4^{3}5^{1}6^{3}$	$2^{23}3^{4}$	$1^{3}2^{30}3^{1}$	1.2593	30	$2^{72}3^{4}4^{1}6^{2}$	$2^{20}3^{2}$	$1^{8}2^{16}$	0.6857
13	$2^{124}3^{4}4^{1}5^{2}6^{2}$	$2^{22}3^{4}$	$1^{6}2^{15}3^{3}$	0.9231	sum		$2^{851}3^{122}4^{7}5^{1}$	$1^{111}2^{1211}3^{450}4^{120}5^{5}$	1.9337

对这 35 个测试场景，运用算法 1 得到了每个待测系统的一致闭的禁忌交互集，然后用算法 2 就可以生成相应的二维禁忌覆盖表，得到结果如表 6 所示，其中我们的算法（一致闭算法）行的数据是运行五次算法得到禁忌覆盖表行数的平均值. AETG-SAT 行是运行 50 次 AETG-SAT 算法得到的二维禁忌覆盖表的行数的平均值[16].

表中加黑的数字表示对应的算法生成的禁忌覆盖表具有更少的行. 从表中可看出我们的算法生成的禁忌覆盖表与 AETG-SAT 算法生成的覆盖表规模相当. Cohen 等人在 AETG-SAT 算法中集成了 SAT 解法器，需要把组合测试约束问题编码成命题逻辑公式，而我们的算法简单，易理解，且容易扩展，在把显性禁忌交互集生成一致闭的禁忌交互集之后，

表 6　35 个测试场景中 2 维覆盖表的规模比较

场景	SPIN-S	SPIN-V	GCC	Apache	Bugz.	1	2	3	4	5	6	7
一致闭算法	**26.8**	**42.4**	**24.4**	**42.6**	**24.8**	56.2	40.6	**20.8**	30.0	64.6	**32.8**	15.6
AETG-SAT	27.1	42.5	24.8	42.8	24.9	**54.7**	**40.1**	21.0	**28.7**	**64.1**	34.0	**12.0**
场景	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0	17.0	18.0	19.0
一致闭算法	58.6	33.6	65.2	**60.0**	58.2	54.6	**56.2**	40.8	33.2	**56.2**	58.2	**64.8**
AETG-SAT	**57.3**	**27.2**	**64.1**	61.1	**57.1**	**51.0**	**56.2**	**40.7**	**33.0**	57.0	**57.7**	66.3
场景	20.0	21.0	22.0	23.0	24.0	25.0	26.0	27.0	28.0	29.0	30.0	
一致闭算法	72.8	58.2	29.8	16.2	59.6	66.8	44.2	50.2	**67.6**	**40.4**	**47.6**	
AETG-SAT	**71.6**	**55.0**	**28.7**	**15.7**	**55.9**	**66.2**	**43.3**	**49.8**	68.4	41.2	50.2	

许多生成一般的组合测试用例集的算法都可扩展成生成避免禁忌交互的测试用例集的算法.在得到一致闭的禁忌交互集后,可增进我们对系统的理解,帮助我们对系统进行分析,如在测试过程中发现系统失效时可帮助我们对错误交互进行准确的定位.

实验 2. 随机实验.

Cohen 等人研究的 35 个测试场景中因素大部分是二元的,并且显性禁忌交互大部分是二维的,这可能导致隐含禁忌交互数目比较少.隐含的禁忌交互数目与显性禁忌交互数目比值到底有多大?它与哪些因素有关?它应该与它显性交互的可归结因素的数目有关.影响隐含的禁忌交互数目与显性禁忌交互数目比值的另一个因素可能是测试场景中显性禁忌交互数目所包含的顶点(一维交互)数目与待测系统中所有顶点数目的比值,

这个比值越小,可能出现的可归结因素就越少,从而隐含禁忌交互数目就可能越少.为此,我们设计随机实验模型来研究它们之间的联系.为了简单起见,我们只研究二维与三维禁忌交互.

假设待测系统中,随机选择二维和三维交互作为显性禁忌交互,即在所有二维交互中等可能的选择一批二维交互,然后在所有的三维交互中等可能的选择一批三维交互作为显性禁忌交互集,然后我们计算对应的可归结因素集的大小,为了对比,我们也计算了理论上的可归结因素数目的期望值.

定理 8. 设待测系统有 k 个因素,每个因素的取值个数都为 v,在所有二维交互中等可能的随机选择 n_1 个,在所有的三维交互中等可能的随机选择 n_2 个,这些交互构成显性禁忌交互集 Π,设其对应的可归结因素集 $FactorsOfResolu(\Pi)$ 中因素数目为 X,则其期望

$$E(X)=k\left(\sum_{j=0}^{v}(-1)^j\binom{v}{j}\left(1-\frac{(k-1)v\cdot j}{w_1}\right)^{n_1}\cdot\left(1-\frac{(w_1-(k-1)v^2)\cdot j}{w_2}\right)^{n_2}\right),$$

其中 w_1,w_2 分别为待测系统中所有的二维交互和三维交互的数目.

证明. 设待测系统中所有的二维交互数目为 w_1,所有的三维交互数目 w_2,则

$$w_1=\frac{k(k-1)}{2}v^2,\ w_1=\frac{k(k-1)(k-2)}{6}v^3.$$

设随机选择显性禁忌交互时,第 1 个因素的第 j 个取值被选到的事件是 A_j,则第 j 个取值没有被选到的概率为

$$P(\overline{A_j})=\left(1-\frac{(k-1)v}{w_1}\right)^{n_1}\cdot\left(1-\frac{w_1-(k-1)v^2}{w_2}\right)^{n_2},$$

类似的,第 1 个因素有 j 个取值 $0,1,\cdots,j-1$ 没有被选到的概率为

$$P(\overline{A_1A_2}\cdots\overline{A_j})=\left(1-\frac{(k-1)v\cdot j}{w_1}\right)^{n_1}\cdot\left(1-\frac{(w_1-(k-1)v^2)\cdot j}{w_2}\right)^{n_2}.$$

再令 B_i 指示第 i 个因素是可归结因素这个事件的随机变量,即 $B_i=1$ 表示第 i 个因素是可归结因素,$B_i=0$ 表示第 i 个因素不是可归结因素,则 $P(B_i)=P(A_1A_2\cdots A_v)$.

根据容斥原理,有

$$\begin{aligned}P(B_i)=P(A_1A_2\cdots A_v)&=1-\sum P(\overline{A_i})+\sum P(\overline{A_iA_j})+\cdots+(-1)^vP(\overline{A_1A_2}\cdots\overline{A_v})\\&=1-vP(\overline{A_1})+\binom{v}{2}P(\overline{A_1A_2})+\cdots+(-1)^j\binom{v}{j}P(\overline{A_1A_2}\cdots\overline{A_j})+\cdots+(-1)^vP(\overline{A_1A_2}\cdots\overline{A_v})\\&=\sum_{j=0}^{v}(-1)^j\binom{v}{j}\left(1-\frac{(k-1)v\cdot j}{w_1}\right)^{n_1}\cdot\left(1-\frac{(w_1-(k-1)v^2)\cdot j}{w_2}\right)^{n_2}.\end{aligned}$$

而可归结因素集 $FactorsOfResolu(\Pi)$ 中因素数目为 $X=\sum_{i=1}^{k}B_i$

所以可归结因素集 $FactorsOfResolu(\Pi)$ 中因素数目的期望

$$\begin{aligned}E(X)=E\left(\sum_{i=1}^{k}B_i\right)&=\sum_{i=1}^{k}P(B_i)\\&=k\left(\sum_{j=0}^{v}(-1)^j\binom{v}{j}\left(1-\frac{(k-1)v\cdot j}{w_1}\right)^{n_1}\cdot\left(1-\frac{(w_1-(k-1)v^2)\cdot j}{w_2}\right)^{n_2}\right).\end{aligned}$$

证毕.

我们的实验结果见表 7,表中显性模式栏是随机生成的显性禁忌交互集 Π 的模式,如(60,0.9)表示在待测系统中先等可能的选择 54 个二维禁忌交互,再等可能的选择 6 个三维交互,即二维交互占 90%,然后我们根据这个模式在待测系统中生成的 5 个显性禁忌交互集 Π,再根据算法 1 生成相应的 5 个一致闭的禁忌交互集$\overline{\Pi}$,一致闭模式栏是这 5 个一致闭的禁忌交互集$\overline{\Pi}$的平均模式,$\overline{\Pi}/\Pi$ 栏是 5 次运行下一致闭的禁忌交互集中交互数目与显性禁忌交互数目的比

值的平均值，顶点比栏是显性禁忌交互集中所包含的顶点数目平均值(相同的顶点只算一个)与待测系统中所有顶点数目的比值. $E(X)$栏是可归结因素数目X的期望，可归结数栏是在每个模式下随机生成的5个显性禁忌交互中可归结因素数目的平均值，时间栏是每个模式下，生成一致闭的禁忌交互集的平均时间，运行环境是 Matlab, Windows 2007, INTEL(R) core(TM)2 Duo 2.20GHz，运行时间单位为 s.

表 7　随机生成禁忌交互集与一致闭禁忌交互集数目对比

CA 模型	显性模式	一致闭模式	$\overline{\Pi}/\Pi$	顶点比	$E(X)$	可归结数	时间/s
2^{100}	(60,0.9)	$2^{103}3^{29}4^{1}$	2.2233	0.4770	21.8226	21.8	1.0156
2^{100}	(60,0.8)	$1^{1}2^{97}3^{71}4^{30}5^{14}6^{1}$	3.5567	0.4890	23.3292	24.2	19.3114
2^{100}	(60,0.65)	$2^{56}3^{77}4^{64}5^{55}6^{31}7^{14}8^{4}$	5.0133	0.5070	25.5968	25.4	110.8002
2^{100}	(60,0.5)	$2^{43}3^{89}4^{109}5^{124}6^{125}7^{121}8^{109}9^{78}10^{44}11^{25}12^{8}13^{1}$	14.6100	0.5170	27.8646	27.6	1834.0000
2^{100}	(80,0.9)	$1^{1}2^{172}3^{111}4^{53}5^{24}6^{8}$	4.6175	0.5590	32.3484	33.2	41.2384
2^{100}	(80,0.75)	$1^{1}2^{120}3^{160}4^{118}5^{53}6^{35}7^{16}$	6.2825	0.5670	35.2961	32.2	217.0869
2^{100}	(80,0.65)	$1^{1}2^{105}3^{221}4^{198}5^{203}6^{178}7^{148}8^{67}9^{15}10^{1}$	14.2175	0.6172	37.2323	38.0	2111.2000
3^{100}	(100,0.9)	$2^{90}3^{26}4^{16}5^{12}6^{8}7^{7}8^{4}9^{3}10^{1}$	1.6820	0.5080	12.6648	12.0	3.4201
3^{100}	(100,0.75)	$2^{75}3^{35}4^{21}5^{17}6^{14}7^{12}8^{10}9^{6}10^{3}11^{2}12^{1}13^{1}$	1.9680	0.5427	14.5968	14.8	10.8946
3^{100}	(100,0.65)	$2^{65}3^{39}4^{17}5^{20}6^{16}7^{16}8^{19}9^{20}10^{22}11^{21}12^{20}13^{16}14^{9}15^{5}$	3.0500	0.5460	15.9313	14.8	190.8406
3^{100}	(110,0.9)	$2^{99}3^{36}4^{27}5^{26}6^{39}7^{51}8^{50}9^{38}10^{18}11^{4}$	3.5200	0.5313	15.4033	14.4	273.5874
3^{100}	(110,0.8)	$2^{88}3^{43}4^{26}5^{28}6^{36}7^{36}8^{39}9^{39}10^{34}11^{29}12^{25}13^{23}14^{25}15^{27}16^{27}17^{25}18^{19}19^{12}20^{6}21^{2}$	5.3382	0.5540	16.8935	16.8	6869.1000
3^{100}	(120,0.9)	$2^{108}3^{47}4^{42}5^{36}6^{33}7^{35}8^{35}9^{39}10^{41}11^{31}12^{20}13^{6}14^{1}$	3.9467	0.5567	18.2869	18.0	611.6262
3^{100}	(120,0.85)	$2^{102}3^{45}4^{37}5^{30}6^{32}7^{33}8^{31}9^{32}10^{31}11^{28}12^{26}13^{22}14^{15}15^{8}16^{3}17^{1}$	3.9767	0.5653	19.1303	17.6	8205.0000
4^{100}	(130,0.9)	$2^{117}3^{13}4^{14}5^{5}6^{7}7^{5}8^{2}9^{3}10^{3}12^{1}13^{1}$	1.3077	0.4920	5.8936	6.4	2.1705
4^{100}	(130,0.75)	$2^{97}3^{33}4^{8}5^{6}6^{8}7^{4}8^{7}9^{5}10^{6}11^{2}12^{4}$	1.3908	0.5280	7.1686	6.2	4.0945
4^{100}	(130,0.5)	$2^{65}3^{65}4^{2}5^{6}6^{12}7^{12}8^{7}9^{10}10^{17}11^{21}12^{14}13^{25}14^{34}15^{28}17^{28}18^{28}19^{35}20^{32}21^{24}22^{27}23^{24}24^{17}25^{13}26^{5}27^{5}$	4.3677	0.5625	9.4602	11.8	3769.1000
4^{100}	(140,0.9)	$2^{126}3^{14}4^{13}5^{6}6^{4}7^{4}$	1.1986	0.5255	7.2447	6.8	0.6249
4^{100}	(140,0.75)	$2^{105}3^{35}4^{6}5^{16}6^{12}7^{12}8^{16}9^{16}10^{12}11^{12}12^{16}13^{11}14^{5}15^{5}16^{2}17^{1}18^{1}$	2.0200	0.5450	8.7208	10.0	87.5902
4^{100}	(160,0.8)	$2^{128}3^{32}4^{13}5^{18}6^{15}7^{16}8^{12}9^{12}10^{17}11^{9}12^{11}13^{5}14^{6}15^{4}16^{2}17^{1}18^{1}$	1.8813	0.5925	11.6346	12.2	71.8214
4^{100}	(170,0.8)	$2^{136}3^{34}4^{11}5^{15}6^{17}7^{19}8^{20}9^{21}10^{18}11^{23}12^{32}13^{30}14^{32}15^{30}16^{27}17^{21}18^{11}19^{6}20^{1}$	2.9647	0.6070	13.5238	11.0	2123.0000

从表 7 中可看到：

(1) 随机生成的禁忌交互集与其对应的一致闭的禁忌交互集的大小比值与禁忌交互集中包含的不同的顶点数目有关，在几个不同的模式中，当不同的顶点数目与总的顶点数目小于 50%时，一致闭的禁忌交互集能很快生成出来，并且$\overline{\Pi}/\Pi$栏数值小于5，当不同的顶点数目与总的顶点数目大于等于50%时，生成一致闭的禁忌交互集的时间显著增长，并且在实验过程中可看出在同一个模式下有时生成的时间很长，有时很短，很不稳定.

(2) 随机生成的禁忌交互集与其对应的一致闭的禁忌交互集的大小比值也与可归结因素数目相关，在2^{100}的 CA 模型中，可归结因素数目达到 30 左右时，生成时间才显著增长，而在3^{100}的 CA 模型中，可归结因素数目达到 15 左右时就显著增长，而在4^{100}的 CA 模型中，可归结因素数目只到 11 左右时就显著增长，所以随着每个因素的取值越多，随机生成的禁忌交互集中禁忌交互越分散，这时很少的可归结因素就可能导致$\overline{\Pi}/\Pi$比很大，生成一致闭禁忌交互集的时间很长.

(3) 在同一个 CA 模型下生成同样数目禁忌交互集时，若禁忌交互集中三维禁忌交互越多，$\overline{\Pi}/\Pi$比越大，生成一致闭的禁忌交互集的时间也越长，也越不稳定，并且生成的一致闭的禁忌交互集中禁忌交互的维数也越来越分散.

5　结论及进一步工作

本文研究了在待测系统中各个因素的取值之间存在约束时如何处理约束，如何生成尽可能少的测试用例. 我们先由约束生成显性禁忌交互集Π，再在禁忌交互集中进行归结运算，这是 SAT 问题中的归结推理的推广，反复应用归结运算来生成一致闭的禁忌交互集$\overline{\Pi}$，然后我们对一致闭的禁忌交互集提出了类 AETG 算法来生成尽可能少的测试用例，随后的实验研究了 Cohen 等人提出的 35 个测试场景，对每个场景中显性禁忌交互集来生成一致闭禁忌交互集，发现得到一致闭的禁忌交互集中交互数平均是原始禁忌交互数的二倍左右，再用这个类 AETG 算法来生成相应的测试用例，其规模与Cohen 等人[16]用集成了 SAT 解法器的 AETG 算法生成的测试用例规模相当，但是我们算法不需要把组合测试模型转化成 SAT 模型，并且我们计算出了所有

的禁忌交互,对模型的理解与测试非常有帮助.随后我们设计了随机实验,通过随机生成显性禁忌交互集,然后生成一致闭的禁忌交互集,研究它们之间的数目比,是否在可接收范围内,与哪些因素有关.

由于我们生成一致闭的禁忌交互集的算法时空复杂性都是指数增长的,对于小的待测系统,显性禁忌交互集比较小时适用,对于大的复杂的带约束系统,本文的方法失效,所以推广 SAT 解法器中的一些如回溯、约束传播等技术来生成尽可能少的测试用例是一个重要研究的问题.另外,元启发式算法是生成测试用例的另一类重要方法,所以研究在禁忌交互集是一致闭的和不是一致闭的情形下生成测试用例的元启发式算法也是需要研究的问题.

参 考 文 献

[1] Kuhn D R, Reilly M J. An investigation of the applicability of design of experiments to software testing//Proceedings of the 27th NASA/IEEE Software Engineering Workshop, NASA Goddard Space Flight Center. Los Alamitos, USA, 2002: 91-95

[2] Mandl R. Orthogonal latin squares: An application of experimental design to compiler testing. Communications of the ACM, 1985, 28(10): 1054-1058

[3] Brownlie R, Prowse J, Phadke M. Robust testing of AT&T PMX/StarMail using OATS. AT&T Technical Journal, 1992, 71(3): 41-47

[4] Cohen D M, Dalal S R, Fredman M L, et al. The AETG system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering, 1997, 23(7): 437-444

[5] Tung Y W, Aldiwan W S. Automating test case generation for the new generation mission software system//Proceedings of the IEEE Arospace Conference. Big Sky, USA, 2000: 431-437

[6] Bryce R C, Colbourn C J. The density algorithm for pairwise interaction testing. Software Testing, Verification and Reliability, 2007, 17(3): 159-182

[7] Bryce R C, Colbourn C J. A density-based greedy algorithm for higher strength covering arrays. Software Testing, Verification and Reliability, 2009, 19(1): 37-53

[8] Lei Y, Tai K C. In_Parameter_Oder: A test generation strategy for pairwise testing. Department of Computer Science, North Carolina State University. Raleigh, North Carolina: Technical Report TR-2001-03, 2001

[9] Lei Y, Kacker R, Kuhn D R, et al. IPOG/IPOG-D: Efficient test generation for multi-way combinatorial testing. Software Testing, Verification and Reliability, 2008, 18(3): 125-148

[10] Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pair-wise covering designs for software testing. Information Processing Letters, 2002, 81(2): 85-91

[11] Cohen M B, Colbouns C J, Gibbons P B, Mugridge W B. Constructing test suites for interaction testing//Proceedings of the International Conference on Software Engineering (ICSE2003). Portland, USA, 2003: 38-48

[12] Cohen M B, Colbouns C J, Ling A C H. Augmenting simulated annealing to build interaction test suites//Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE 2003). Denver Colorado, USA, 2003: 394-405

[13] Toshiaki Shiba, Tatsuhiro Tsuchiya, Tohru Kikuno. Using artificial life techniques to generate test cases for combinatorial testing//Proceedings of the COMPSAC04. Hong Kong, China, 2004: 72-78

[14] Bryce R C, Colbourn C J. Prioritized interaction testing for pair-wise coverage with seeding and constraints. Journal of Information and Software Technology, 2006, 48(10): 960-970

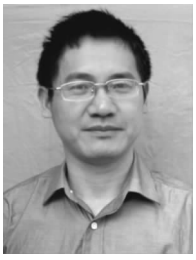
[15] Cohen M B, Dwyer M B, Jiangfan Shi. Interaction testing of highly-configurable systems in the presence of constraints//Proceedings of the International Symposium on Software Testing and Analysis. London, UK, 2007: 129-139

[16] Cohen M B, Dwyer M B, Shi Jiang-Fan. Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach. IEEE Transactions on Software Engineering, 2008, 34(5): 633-650

[17] Garvin B J, Cohen M B, Dwyer M B. An improved meta-heuristic search for constrained interaction testing//International Symposium on Search Based Software Engineering (SSBSE). Windsor, UK, 2009: 13-22

[18] Danziger P, Mendelsohn E, Moura L, Stevens B. Covering arrays avoiding forbidden edges. Theoretical Computer Science, 2009, 410(52): 5403-5414

[19] Katona G. Two applications (for search theory and truth functions) of Sperner type theorems. Periodica Mathematica Hungarica, 1973, 3: 19-26



ZHOU Wu-Jie, born in 1973, Ph.D., lecturer. His research interests include software testing and statistical testing etc.

ZHANG De-Ping, born in 1973, Ph.D., lecturer. His research interests include teaching and researching on software testing, statistical testing and mathematical statistics.

XU Bao-Wen, born in 1961, Ph.D., professor. His research interests include teaching and researching on programming language, software engineering, parallel and network, acquisition technique on knowledge and information etc.

Background

This paper is supported by the National Natural Science Foundation of China under Grant (Nos. 90818027, 91018005), the National High-Tech Research Subjects and Development Plan of China under Grant (No. 2009AA01Z147), and the National Grand Fundamental Research 973 Program of China under Grant (No. 2009CB320703). These projects mainly research on the follows fields: Combinatorial coverage method and its effectiveness for software testing, the existence and the generation algorithm for several minimal combinatorial covering table, some techniques for software fault diagnosis and debugging based on combinatorial testing, the application of the combinatorial testing in Web testing and software configuration testing. The authors have made some works on the test case generation algorithm and the applications. This paper focuses on the research of generations of combinatorial testing suites in the presence of constraints and the structure of the implied forbidden interaction set.