基于矩阵行列变换的测试用例约简算法*

周冲波，楼俊钢，程　龙
（湖州师范学院 信息与工程学院，浙江 湖州 313000）

摘　要：针对测试用例约简问题，定义了一种不会改变测试需求与测试用例覆盖关系的布尔运算。应用此运算，辅以不同的测试需求、用例集优先策略，经矩阵的列变换得到精简的测试需求集，然后使用行变换对测试用例集进行约简。该方法不受测试用例输入顺序的影响。实验表明，与一些常用的约简算法相比，提出的算法在有序树生成程序测试用例约简的几个实例上都能得到较优的用例集。

关键词：软件测试；测试用例；测试需求约简；测试用例约简
中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1001-3695(2013)03-0779-04
doi:10.3969/j.issn.1001-3695.2013.03.035

Test suites reduction based on matrix transformation

ZHOU Chong-bo, LOU Jun-gang, CHENG Long
(*College of Computer Science & Technology*, *Huzhou Teachers College*, *Huzhou Zhejiang* 313000, *China*)

Abstract: In allusion to the problem of test requirements reduction, this paper defined a new Boolean operation of the relationships which didn't change the coverage of test suites to test requirements. Redundance requirements was reducted by column transformation operation of matrix basis of the interrelations among the testing requirements, then test suites could be reducted by row transformation operation of matrix. The experimental results based on several cases shows that, compared with some other existed methods, this method has better property that can generate the minimal test suite to test all the test requirements sufficiently.

Key words: software testing; test suites; test suites reduction; test requirements reduction

0　引言

测试用例的约简问题可以描述为[1~3]：给出一个测试用例集 T，一个测试用例需求集 R，R 满足给定程序要求的某种测试覆盖准则，T 可以完成对 R 中所有 r_i 的测试，要求从 T 中找到一个测试用例优化集可以满足所有 r_i 的测试，需要在大量的测试用例中筛选出最优的测试用例，即能用最少的测试用例覆盖最多的测试需求。

最常见的测试用例约简方法有贪心算法[4,5]、启发式算法[6]、整数规划算法[7]以及扩张集算法[8,9]等。贪心算法（简称 G 算法）每次将满足测试需求 R 最多的一条测试用例选取出来，从 R 中删除已被该测试用例满足了的测试需求，在剩余的测试用例中反复执行，直到满足所有的测试需求，即 R 为空集[4]。在此基础上，Chen 等人提出了 GE 和 GRE 算法[5]。GE 算法改进了贪心算法，首先找出某项需求仅能被唯一一条测试用例满足的对应测试用例，即必不可少测试用例，再使用贪心算法进行约简。GRE 算法反复地选取出必不可少测试用例，剔除 1-1 冗余测试用例，再使用贪心算法进一步地约简测试用例。Harrold 等人提出了一种启发式算法，按照测试用例的重要程度来约简测试用例集（简称 H 算法）[6]。该算法将测试需求进行划分，若某项需求 r_k 能被 i 条测试用例满足，则划分至 R_i，由此得到一个新的测试集合 $R_1, R_2, \cdots, R_d (d \leqslant n)$。由 R_1 得到必不可少测试用例，然后从 R_2 开始，选择能最大限度满足需求的测试用例，即运用贪心算法。以此类推，从剩余 R 中依次取得最优的测试用例。

这几种算法都是对测试用例集的简化策略，它们的效果取决于初始化的测试用例集，不能从根本上保证根据测试目标对测试用例进行最优化。测试用例集是根据测试需求来确定的，如果不考虑测试需求集的精简而只着眼于测试用例集的算法研究，所取得的效果将很有限[8]。为此，许多研究人员建议在简化测试用例前先进行测试需求的约简。如聂长海等人[10]首先充分考虑测试目标中测试需求之间的相互关系，对可用测试用例进行划分，生成一个测试用例集，再利用启发式、贪心、整数规划方法消除冗余，进行进一步优化，从而提出一种最小测试用例集生成方法；章晓芳等人[11]着重考虑了测试需求间的相互关系，提出一种基于测试需求约简的测试用例集优化方法。然而这些方法约简测试需求的过程十分频琐，先去除包含关系的需求，再利用部分重合关系进行一步约简，无法一步到位，并且也不能保证每次都得到最简的测试用例集。

本文参考 Quine 和 McCluskey 在对布尔函数进行代数化简法时提出的 Q-M 方法[12,13]，即用 0-1 来表示测试用例与测试需求之间的关系。为了保证算法不受测试用例排列顺序的影响，并且尽可能实现对测试用例更优更准确的选择，定义了

收稿日期：2012-07-11；**修回日期**：2012-08-27　　**基金项目**：国家自然科学基金资助项目(61103051)；浙江省自然科学基金资助项目(Y1101237)

作者简介：周冲波(1990-)，女，浙江宁波人，学士，主要研究方向为软件测试、软件可靠性；楼俊钢(1982-)，男，浙江义乌人，副主任，讲师，博士，主要研究方向为软件可靠性建模、可信计算、系统性能评测等(loujungang0210@hotmail.com)；程龙(1989-)，男，河南光山人，学士，主要研究方向为软件测试、云计算.

一种新的关系运算——减变换运算⊖,该运算不会改变测试用例集对测试需求的覆盖率;同时提出一种基于矩阵行列变换的测试用例约简算法(test cases reduction based on matrix transformation,TRMT),从而建立一个新的测试用例约简模型。在算法中,约简时设定以下两种不同的优先策略:a)在实现列变换(测试需求约简)过程中,优先选择当前"1"个数最少的那一列;b)在实现行变换(测试用例约简)过程中,优先选择当前"1"个数最多的那一行。

1 基于矩阵变换的测试用例集约简模型

1.1 相关定义

定义1 对于待测软件系统,测试需求集合用 $R=\{r_1,r_2,\cdots,r_m\}$ 表示,测试用例集合用 $T=\{t_1,t_2,\cdots,t_n\}$ 表示,则可以用二维数组的元素 $a[i][j](i=1,2,\cdots,m;j=1,2,\cdots,n,)$ 表示第 i 个测试用例和第 j 个测试需求之间的一一对应关系,所有的关系集合组成一个 $n\times m$ 的二维矩阵

$$A=\begin{pmatrix} a[1][1] & a[1][2] & \cdots & a[1][m] \\ a[2][1] & a[2][2] & \cdots & a[2][m] \\ \vdots & \vdots & & \vdots \\ a[n][1] & a[n][2] & \cdots & a[n][m] \end{pmatrix}$$

定义2 二维矩阵中 $a[i][j]$ 的取值为0或1。若第 i 个测试用例能够满足第 j 个测试需求,则 $a[i][j]=1$;若第 i 个测试用例不能够满足第 j 个测试需求,则 $a[i][j]=0$。

定义3 对于任意两个关系 $a[i][j]$ 和 $a[i][j]'$,定义关系间的运算⊖,运算法则包括如下四种情况:

a) $1\ominus 1=0$

b) $1\ominus 0=1$

c) $0\ominus 1=0$

d) $0\ominus 0=0$

定义4[9] 对于测试需求集 R 与 R',若 $|R'|\leqslant|R|$,$\forall r\in R,\exists r'\in R',r'\subseteq r$,且 $\forall r'\in R',\exists r\in R,r'\subseteq r$,则称测试需求集 R' 为 R 的精简测试需求集。若不存在满足 $|R''|<|R'|$ 的精简测试需求集 R'',则称 R' 为 R 的最小精简测试需求集。

1.2 TRMT约简模型

该模型的主要思想是建立一个测试需求集与测试用例集之间一一对应的0-1矩阵,矩阵的行表示测试用例,矩阵的列表示测试需求。利用测试需求间的相互关系,通过所定义的⊖变换操作,优先选出不被其他测试需求包含的测试需求,达到约简测试需求集的目的。需要注意的是,由于计算得到最小精简需求集的复杂度非常大,在本文中只是去除了具有等价或者包含关系的需求,因此根据测试需求集间的相互关系精简得到的测试需求集并不一定是最小精简需求集。在此基础上消除测试用例间的冗余关系,最终达到测试用例的约简。

模型具体实现步骤如图1所示。

a)在执行列运算过程中,优先选择当前"1"个数最少的那一列,认为其最有可能被其他列所包含,因此保留下来,并移到当前矩阵(排除已被选择过的列)的最左端,即与当前矩阵的最左列交换,使得最后得到的精简需求集按"1"个数多少升序排列。

b)选中的列与剩余各列做减变换,若能使被减列各元素都为0,则说明选中列包含于被减列,因此删去被减列,即置被减列所有元素为0。

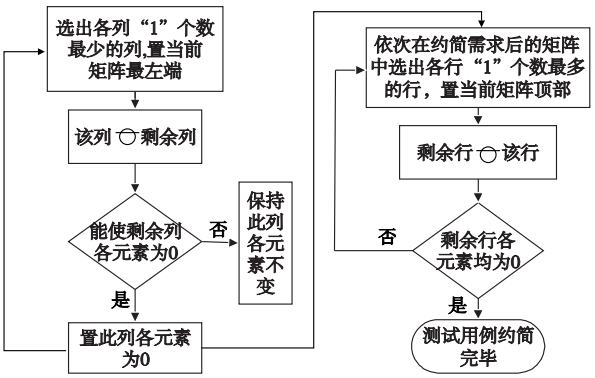

图1 TRMT约简模型的基本实现步骤

c)测试需求约简完毕后,已经自动按照满足各需求的用例个数对需求进行升序排列。某一列所含"1"的个数越少,则该列所对应的需求能被测试用例集中的用例覆盖的数量越少,即认为该需求对应的测试用例重要性越高。故从第一列开始依次对变量为1的元素进行优先选择。

d)每行中"1"的个数表示该行所代表的测试用例所能测试的需求个数,实际上是该测试用例对测试需求集的覆盖度。如果行中"1"个数越多,可以认为该测试用例的覆盖度越高,因此优先选择该用例,将其置于当前矩阵的顶部,并选择其余行与该行做⊖变换。若能使剩余行各元素为0,则表示剩余行是冗余的,剔除,即置该行所有元素为0。通过多次行之间的减运算,可以把用例集中冗余的测试用例删除掉。

定理1 测试用例集与测试需求集组成的关系矩阵经过行列之间的⊖运算后,测试用例集对测试需求的覆盖率保持不变。

证明 可以分行、列变换两种情况进行证明。

a)对于测试需求的列变换约简,每次⊖运算的结果均可以包含在下面四种情形:

(a) $1\ominus 1=0$ 的情形,$a[i][k]=a[j][k]=1$,表示 r_k 能同时被测试用例 t_i 和 t_j 满足,因此相减后,无论 $a[i][k]=0$ 或者 $a[j][k]=0$,均可以保证 r_k 能被覆盖,保持覆盖率不变;

(b) $1\ominus 0=1$ 的情形,$a[i][k]=1$,$a[j][k]=0$,表示 r_k 能被 t_i 覆盖而不能被 t_j 覆盖,相减后对应的布尔关系保持不变,因此保持覆盖关系不会变化;

(c) $0\ominus 1=0$ 的情形,$a[i][k]=0$,$a[j][k]=1$,表示 r_k 能被 t_j 覆盖而不能被 t_i 覆盖,$a[i][k]-a[j][k]=0$ 意味着布尔关系保持不变,覆盖率自然不变;

(d) $0\ominus 0=0$ 的情形,$a[i][k]=a[j][k]=0$,表示第 k 个测试需求都不能被 t_i 和 t_j 覆盖,运算后,t_i 和 t_j 还是不能覆盖 r_k,覆盖率依然不变。

从而,经过列变换不会改变测试用例集与测试需求集之间的覆盖关系。

b)对于测试用例的行变换约简,也可以分为四种情形说明:

(a)若测试用例 t_i 能满足测试需求 r_k,而另一个测试用例 t_j 也能满足 r_k,则这两个测试用例冗余,删除任意一个后并不改变覆盖率;

(b)若测试用例 t_i 能满足测试需求 r_k,与另一个无法满足 r_k 的测试用例 t_j 做减变换,则 t_i 仍能满足该测试需求,因此覆盖率不变;

(c)若测试用例 t_i 无法满足测试需求 r_k,与一个能满足 r_k

的测试用例 t_j 做减变换,则 t_i 仍无法满足该测试需求,因此覆盖率不变;

(d)若测试用例 t_i 无法满足测试需求 r_k,再与一个同样无法满足 r_k 的测试用例 t_j 做减变换,则 t_i 仍无法满足该测试需求,因此覆盖率不变。

从而,经过行变换不会改变测试用例集与测试需求集之间的覆盖关系。

综上所述,测试用例集与测试需求集组成的关系矩阵经过行列之间的⊖运算后,测试用例集对测试需求的覆盖率保持不变。定理得证。

1.3　TRMT 算法

输入:测试用例个数和测试需求个数,对应的布尔矩阵。

输出:最优测试用例编号。

a)输入测试用例与测试需求的覆盖关系矩阵 A,存放于 $a[\,][\,]$;

b)取 $\min(a[\text{row}+1][j])$,将对应需求编号记录到新的数组 req[ ]中,即 $A \xrightarrow{\text{优先选择一个 } r_i} A$;

c)该列与剩余列($a[\text{row}+1][j]\neq 0$)做⊖运算,并初始化 $a[\text{row}+1][j]$为 0;

d)循环 c)d)至测试需求约简完毕;

e)在所有 $T_i(i=1,2,3\cdots)$中,取 $\min(a[i][\text{column}+1])$,对满足 $r_i(i=1,2,3\cdots)$的 t_i 比较 $a[i][\text{column}+1]$,并取最大的值对应的测试用例,将对应用例编号记录到新的数组 test[ ]中,即 $A \xrightarrow{\text{优先选择一个 } t_i} A$;

f)该行与剩余行($a[i][\text{column}+1]\neq 0$)做减变换,遵循定义 3,并初始化 $a[i][\text{column}+1]$为 0;

g)循环 e)f)至测试用例优化完毕;

h)输出最优测试用例集编号。

TRMT 算法进行测试需求约简的最坏时间复杂度为 $O(n^2\cdot m)$,假设约简后需求个数为 k,测试用例约简的最坏时间复杂度为 $O(m^2\cdot k)$,实际上,随着测试用例的消去,矩阵维数的变小,消去一行或一列的时间复杂度迅速递减,实际复杂度远远好于最坏情况。

2　实例分析

为验证所提出方法的有效性,开发相关的测试用例约简工具,并利用 Harrold 等人[4]在提出 H 算法中运用的程序,在比较 GE、GRE 和 H 算法基础上,对这四种方法进行实验比较。测试对象是一个为输入数据生成有序树的程序,具体测试用例表和程序的流程图分别如表 1 和图 2 所示。

根据流程图以及测试用例表,可以得到初始的测试用例和测试需求的对应关系布尔矩阵如下:

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}
t_1	1	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0
t_2	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	0	0
t_3	1	1	1	1	0	1	0	0	0	0	1	1	1	1	1	0	1	1	0
t_4	1	1	1	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	1
t_5	1	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
t_6	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1
t_7	1	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0
t_8	1	1	1	1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	1
t_9	1	1	1	1	1	1	0	1	0	1	0	1	0	0	1	0	1	1	0
t_{10}	1	1	1	1	1	1	1	0	1	1	1	0	1	1	0	0	0	0	0
t_{11}	1	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
t_{12}	1	1	1	1	1	1	1	0	1	1	0	1	0	0	1	0	1	1	0

表 1　有序树生成程序测试用例

测试用例	A[i].key	first	last
t_1	2 1	1	2
t_2	2 1 3 4	1	4
t_3	5 1 3 4 2	1	5
t_4	3 2 1 4 5 6 7	1	7
t_5	1 2	1	2
t_6	1 2 3	1	3
t_7	1 4 3 2	2	4
t_8	2 1 3 4 5	1	5
t_9	3 2 1 4 5 6	1	6
t_{10}	4 1 3 2	1	4
t_{11}	1 2 3 4	2	4
t_{12}	6 2 1 4 5 3	1	6

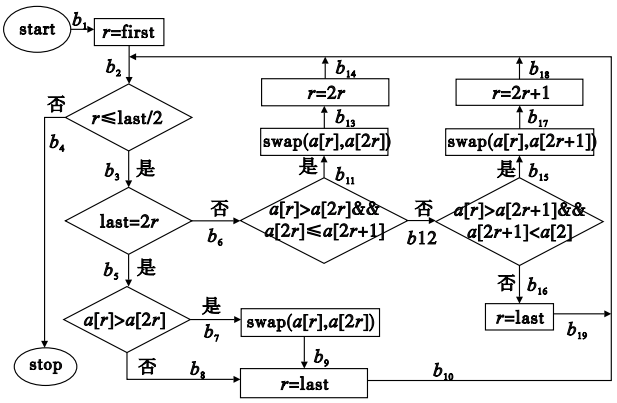

图2　输入数据生成有序树程序流程

在本例子上,当选择所有的测试用例集时,算法实现具体步骤如下:

a)统计各列“1”的个数,选出当前“1”个数最少的列为 r_{16},与 r_1 交换。

b)r_{16} 中各元素分别与剩余列进行减变换,得到如下矩阵:

	r_{16}	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}	r_1	r_{17}	r_{18}	r_{19}
t_1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0
t_2	0	0	0	0	1	0	0	1	0	1	1	0	1	1	0	0	0	0	0
t_3	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0
t_4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
t_5	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
t_6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
t_7	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0
t_8	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0
t_9	0	0	0	0	1	0	0	1	0	1	0	0	0	0	1	0	1	1	0
t_{10}	0	0	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	0	0
t_{11}	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
t_{12}	0	0	0	0	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0

c)对当前矩阵重复以上步骤,得到约简后的测试需求集为 r_{16}、r_7、r_8、r_{11}、r_{15},对应的矩阵为

	r_{16}	r_7	r_8	r_{11}	r_{15}
t_1	0	1	0	0	0
t_2	0	0	1	1	0
t_3	0	0	0	1	1
t_4	1	0	0	0	1
t_5	0	0	1	0	0
t_6	1	0	0	0	0
t_7	0	1	0	0	0
t_8	1	0	0	1	0
t_9	0	0	1	0	1
t_{10}	0	1	0	1	0
t_{11}	0	0	1	0	0
t_{12}	0	1	0	0	1

d)从第一列开始,比较布尔值为 1 的元素所在行“1”的个数,并选择“1”个数最多的用例,此处选择 t_4,置当前矩阵顶部,即与 t_1 所在行交换。

e)剩余行分别与 t_4 所在行做减变换,得到以下矩阵:

	r_{16}	r_7	r_8	r_{11}	r_{15}
t_4	1	0	0	0	1
t_2	0	0	1	1	0
t_3	0	0	0	1	0
t_1	0	1	0	0	0
t_5	0	0	1	0	0
t_6	0	0	0	0	0
t_7	0	1	0	0	0
t_8	0	0	0	1	0
t_9	0	0	1	0	1
t_{10}	0	1	0	1	0
t_{11}	0	0	1	0	0
t_{12}	0	1	0	0	0

f)对当前矩阵重复以上步骤,得到最简测试用例集为:t_4、t_{10}、t_5。

同理,分别选择 T_1:{t_1, t_2, t_3, t_4, t_5, t_6, t_7},T_2:{t_1,t_2, t_3, t_4, t_8, t_9}和 T_3:{t_1, t_3, t_4, t_5, t_6, t_8, t_{10}, t_{11}, t_{12}}作为初始的测试用例集,并运用 GE、H、GRE 算法和 TRMT 算法分别在 T_1、T_2、T_3 三个测试用例—测试需求集中求解约简测试用例集,结果如表 2 所示。

表 2 矩阵行列变换程序测试用例约简结果比较

R-T 集	算法			
	GE 算法	H 算法	GRE 算法	TRMT 算法
T_1	{t_3,$t_1(t_7)$, $t_4(t_6)$,$t_2(t_5)$}	{t_3,$t_1(t_7)$, $t_4(t_6)$,$t_2(t_5)$}	{t_2,t_4,$t_1(t_7)$}	{t_1,t_2,t_4}
T_2	{t_1,t_3, $t_2(t_9)$,$t_4(t_8)$}	{t_1,$t_4(t_8)$, $t_2(t_9)$}	{t_1,t_3, $t_2(t_9)$,$t_4(t_8)$}	{t_1,t_2,t_4}
T_3	{t_{12},t_8,$t_5(t_{11})$}	{$t_5(t_{11})$,t_3,t_1 (t_{10},t_{12}), $t_4(t_6$,$t_8)$}	{$t_5(t_{11})$,t_3, $t_{10}(t_{12})$,$t_4(t_8)$}	{t_4,t_{10},t_5}

从表 2 中可以看出,选择 T_1 作为初始测试用例集情况下,GRE 算法得到精简测试用例集为{t_2,t_4,$t_1(t_7)$},而 H、GE 算法约简后的测试用例集均为{t_3,$t_1(t_7)$,$t_4(t_6)$,$t_2(t_5)$};选择 T_2 作为初始测试用例集情况下,H 算法得到精简测试用例集为{t_1,$t_4(t_8)$,$t_2(t_9)$},GE、GRE 算法约简后测试用例集中用例数量均为 4 个;选择 T_3 作为初始测试用例集情况下,GE 算法得到精简测试用例集为{t_{12},t_8,$t_5(t_{11})$},相应地,H、GRE 算法得到结果稍差。也就是说,GE、H、GRE 这三种方法在针对不同的测试用例—测试需求集时具有不同的优势。相比之下,本文提出的 TRMT 算法在三种情况下均能得到相对较优解。其主要原因是:TRMT 算法在进行行列变换时采用不同的策略,在执行列运算过程中,优先选择当前“1”个数最少的那一列,并把得到的精简需求集按“1”个数多少升序排列;而执行行运算时,按照已经排列好的顺序,依次优先选择每行中“1”个数越多的行,这样就保证了算法能在最大限度上把 更重要的测试用例优先选择出来,因此使用 TRMT 算法约简后得到更精简的测试用例集的可能性更大。

3 结束语

从原始测试用例集中选择规模较小的测试用例从而有效降低测试费用是软件测试行业的主要目标之一,挑选出 100% 覆盖测试需求且代价最小的最精简测试用例子集是一个 NP 难问题[14,15]。本文从需求集、用例集约简两方面着手,提出了一种新的 TRMT 算法。该算法利用矩阵行列变换,通过列变换剔除具有等价关系或包含关系的测试需求,并对简化了的矩阵进行进一步的行变换,以删除冗余的测试用例,从而获得精简的测试用例集。在行列变换时通过设定不同的优先策略,最大限度地在有限复杂度情况下得到更优的精简测试用例集。

本文提出的方法要求测试用例集和测试需求集具有明确对应关系,进一步工作包括:

a)进一步对测试需求进行约简,充分考虑需求间的部分包含等关系,定义一种新的列之间的运算,但同时算法的复杂度也会有所增加,如何保证以较低的复杂度实现更好的需求约简是下一步要进行的主要研究内容之一。

b)考虑应用布尔类型矩阵及相应算法,通过检错率的高低来选择合适的测试用例进行高效率的回归测试。

参考文献:

[1] GRAVES T L. An empirical study of regression test selection techniques [J]. ACM Trans on Software Engineering and Methodology,2001,10(2):185-207.

[2] NIE C H, LEUNG H. The minimal failure-causing schema of combinatorial testing [J]. ACM Trans on Software Engineering and Methodology,2011,20(4):1-38.

[3] BIBLE J, ROTHERMEL G,ROSENBLUM D S. A comparative study of coarse and fine-grained safe regression test selection techniques [J]. ACM Trans on Software Engineering and Methodology, 2001,10(2):149-183.

[4] HARROLD M J, GUPTA R, SOFFA M L. A methodology for controlling the size of a test suite [J]. ACM Trans on Software Engineering and Methodology,1993,2(3):270-285.

[5] CHEN T Y, LAU M F. A new heuristic for test suite reduction [J]. Information and Software Technology,1998,40(5):347-354.

[6] CHEN T Y, LAU M F. On the divide-and-conquer approach towards test suite reduction [J]. Information Sciences,2003,152(1):89-119.

[7] LEE J G, CHUNG C G. An optimal representative sets election method [J]. Information and Software Technology,2000,42(1):17-25.

[8] MARRE M, BERTOLINO A. Using spanning sets for coverage testing [J]. IEEE Trans on Software Engineering,2003,29(11):974-984.

[9] JEFFREY D, GUPTA N. Improving fault detection capability by selectively retaining test cases during test suite reduction [J]. IEEE Trans on Software Engineering,2007,33(2):108-123.

[10] 聂长海, 徐宝文. 一种最小测试用例集生成方法[J]. 计算机学报,2006,26(12):1690-1695.

[11] 章晓芳,徐宝文,聂长海,等. 一种基于测试需求约简的测试用例集优化方法[J]. 软件学报,2007,18(4):821-831.

[12] 周南良. 数字逻辑[M]. 长沙:国防科技大学出版社, 1992.

[13] ROTHERMEL G, UNTCH R, CHU C, *et al*. Prioritizing test cases for regression testing[J]. IEEE Trans of Software Engineering, 2001,27(10):929-948.

[14] JAMES A, HARROLD M. Test-suite reduction and prioritization for modified condition/decision coverage[J]. IEEE Trans on Software Engineering,2003,29(3):195-209.

[15] NIE C H, LEUNG H. A survey of combinatorial testing[J]. ACM Computing Surveys,2011,43(2):1-29.