一种新的测试集简化的测试覆盖准则

崔 霞 高建华
(上海师范大学计算机科学与工程系 上海 200234)

摘 要 在回归测试过程中,测试集的规模不断的变大增加了测试的成本。结合某种测试准则利用测试简化法对测试集中冗余的测试用例进行删除是一种有效的解决方法。但是用此方法得到的简化测试集,其错误检测能力往往被减弱。因此提出了一种新颖的测试覆盖准则,即二级变量串联覆盖准则和二级变量并联覆盖准则。这两种准则主要考虑了变量间的串、并联关系对程序的影响。用此准则与其它测试覆盖准则相组合,利用HGS测试集简化法对测试集进行选择,既简单高效又保证了最小化测试集的错误检测能力。针对文献[3]中的具体应用实例,验证了该测试覆盖准则的有效性。

关键词 测试覆盖准则,测试集简化,回归测试
中图法分类号 TP31

New Testing Coverage Criteria for Test Suite Reduction

CUI Xia GAO Jian-hua
(Department of Computer Science and Engineering, Shanghai Normal University, Shanghai 200234, China)

Abstract The arising scale of test suite adds the testing cost in regression test. With some testing coverage criteria, the testing suite reduction method is a kind of effective approach to erase those redundant test cases. However, with this method, the fault detect effectiveness of test suite is often decreased. So we presented two new test coverage criteria i. e. two-stage variable serial criteria and two-stage parallel coverage criteria. These two criteria mainly focused on the influence of serial or parallel relations of variables on software. Combining this criterion with other test criteria, we simplified the test suite using HGS reduction method and maintained the fault detect effectiveness of reduced test suite highly effectively and simply. With the specific example in reference paper [3], the new testing coverage criteria were validated.

Keywords Test criteria, Test suite reduction, Regression test

1 引言

在软件开发和维护的各个阶段,为了及时发现程序中的缺陷和错误,需要对程序进行测试以保障程序的质量。由于需求变化、系统环境升级等多种原因,软件需要进行多次的修改,这就导致已经被测试过的软件可能引入新的错误,因此要进行严格的、多次的回归测试。在回归测试中为了满足某种测试覆盖准则,对原有的测试案例需要进行改进,包括新增、删除和修改。新的测试用例不断的生成,使得测试集的规模过大,从而耗费计算机的大量资源。根据某种测试覆盖准则,测试集中的测试用例会重复覆盖该准则下的测试需求,出现多个测试用例测试一个需求的冗余现象。为了节省计算机的资源,必须进行测试集优化或最小化,以便在回归测试的过程中及时删除冗余的测试案例。目前已有多种测试集最小化方法[1-6],每种方法有各自的优缺点和适应环境。

测试集最小化的定义如下:给定一个具有 m 个测试用例的测试集 $T:\{t_1;t_2;t_3;\cdots;t_m\}$,根据某种测试覆盖准则得到的测试需求集为 $R:\{r_1;r_2;r_3;\cdots;r_n\}$,假如测试集 T 和测试需求集之间满足如下关系:T 的划分 $T_1,T_2,\cdots,T_n$ 中的 T_i($T_i\neq\Phi,i\in\{1,2,\cdots,n\}$) 里的元素 t_j 满足 r_i。那么问题是:找一个 T 的最小子集 T' 满足 R 中的所有需求。然而,此问题是一个NP完全问题。

本文介绍的测试集简化方法有:简单贪婪启发式[1]、HGS测试集简化法[2]、RSR有冗余的测试集简化法[3]等。

简单贪婪启发式[1]的主要思想:先选择覆盖测试需求最多的那个测试用例,把它加入到最后的最小化的测试集合中,然后把它所覆盖的所有需求去除,然后再重复这个过程,当出现两个测试用例覆盖相同的需求时,任选其中的一个加入到最后的最小化的测试集合中。简单贪婪启发式算法不是全局最优算法,在解决某些特定问题时,得到简化测试集的错误检测能力低,该算法不具有通用性。

HGS测试集简化法[2]是Harrold,Gupta和Soffa三人于1993年提出的,HGS方法的主要思想:首先依据某种测试准则得到测试覆盖需求 $R:\{r_1;r_2;r_3;\cdots;r_n\}$,将测试集 T 划分成:$T_1,T_2,T_3,\cdots,T_n,T_i$ 中的每个测试用例都覆盖了测试需求 r_i;其次把基数(集合中包含的元素的个数)为1的 T 的子

到稿日期:2008-01-22 本文得到国家自然科学基金(No:60673067)资助。
崔 霞(1976-),女,讲师,硕士研究生,主要研究方向为软件可靠性理论与设计、软件测试,E-mail:cuixiahappy2008@year.net;高建华(1963-),男,教授,工学博士,主要研究领域为软件可靠性理论与设计、软件开发环境与开发技术、数据安全与计算机安全、网络测试、LSI/VLSI测试等领域。

集 T_i 放入到最后的最小化的测试集合 T' 中，然后考虑基数是 2 的子集中出现次数最多的测试用例加入最后的最小化的测试集合 T' 中，当测试用例出现在子集中的次数相同时，任选其中一个加入到最小化的测试集合 T′ 中，并标记所有包含该测试用例的子集。基数增大重复此过程，直到所有的测试需求都被满足时，即得最小化测试集。HGS 是一种局部优化算法，得到的简化后的测试集并不一定是最优的结果。HGS 可以和任何一种测试准则结合对测试集进行简化，因此 HGS 算法的有效性很大程度上依赖程序本身的特点和测试准则的选择。

RSR 有冗余的测试集简化法的主要思想[3]是：利用 HGS 算法依据某种覆盖准则把某些测试用例选入最小化的测试集中之后，被判断为冗余的测试用例继续在另外一种覆盖准则下判断是否冗余，若存在冗余则舍弃，否则就保留。该算法的有效性很大程度上也依赖程序本身的特点和测试准则的选择，并且该算法要经过多次测试准则的筛选，大大提高程序的复杂性，降低了简化过程的时间效率。

以上介绍了各种测试简化方法的主要思想和存在的缺点。我们可以看到，在每种方法中要结合某种测试覆盖准则来完成简化工作，在回归测试中需要用各种测试覆盖准则和错误检测能力来评估简化后的最小化测试集的性能和质量。

当前的测试覆盖准则主要有语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、点覆盖、边覆盖、路径覆盖、全使用覆盖、定义-使用覆盖、C-Use 覆盖、P-Use 覆盖[7]等。当前，有大量实验案例对错误检测能力和测试覆盖准则[8-11]之间的关系进行研究。不同测试覆盖原则确定的测试用例可以检测程序中具有不同结构和功能的部分。用单一的测试覆盖判断会大大减少原测试集中的测试用例，降低最小化测试集的错误检测能力(FDE)。由此，用两种不同的测试覆盖组合起来作为判断该测试集的标准可以大大提高错误检测能力。

本文提出了二级变量串联测试覆盖准则及二级变量并联测试覆盖准则，这两种准则依据变量之间的相互关系来检测程序中的错误。将二级变量串联测试覆盖准则和其他测试覆盖准则组合产生测试覆盖要求，用 HGS 算法来减小测试集的规模，既简单快捷又能有效提高错误检测的能力。

本文的结构安排如下：第 1 节是引言，介绍现有的测试集最小化技术、测试准则和本文研究的主要问题。第 2 节提出了二级变量串联测试覆盖准则和二级变量并联测试覆盖准则。第 3 节是应用实例及结果对比，针对具体实例程序，依据二级变量串联测试覆盖准则和其他测试覆盖准则相组合所产生的测试需求，利用 HGS 算法准确有效地简化了测试集。通过和其它方法的对比，说明了本文方法的有效性。最后是全文的总结。

2 二级变量串联和并联覆盖准则

在程序中各个变量或参数之间是有联系的，程序中出现的很多错误往往是由多个变量的串联和并联的作用所引起的，因此本文提出了一种新的基于变量的串、并联关系的测试覆盖准则，即二级变量串联测试覆盖。

首先介绍变量间的串联、并联的含义，以及它们对程序的影响。

定义 1(变量间串联关系) 变量 var_1 影响变量 var_2 的值。串联关系如图 1 所示。

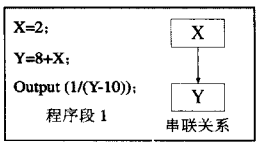

图 1 程序段 1 及变量间的串联关系

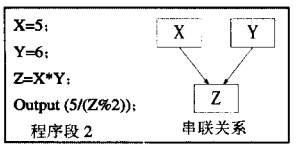

图 2 程序段 2 及变量间的并联关系

程序段 1 表示了变量间的串联影响关系：X 影响 Y 的值，Y 影响 Z 的值，最终的被 0 除的错误是由 X 引起的。

定义 2(变量间并联关系) 若变量 var_1，var_2 同时影响变量 var_3 的值，则变量 var_1，var_2 之间为并行关系。变量间并联关系如图 2 所示。

程序段 2 表示了变量间的并联影响关系：X 和 Y 同时影响变量 Z 的值，X 和 Y 之间为并联关系。

定义 3(变量间串、并联关系) 如图 3 所示，得：

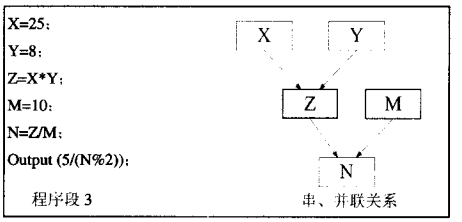

图 3 程序段 3 及变量间的串、并联关系

程序段 3 表示了变量间的串、并联关系：X 和 Y，Z 和 M 为并联关系。X 和 Z，Y 和 Z，Z 和 N，M 和 N 之间为串联关系。

在介绍了变量间的串、并联关系后，以下提出了二级变量串联准则和二级变量并联准则。

二级变量串联准则(程序中所有相关联的两个变量间的串联关系) 它可以表示为一个五元组(var_1，def，$use\text{-}def$，var_2，use)，此五元组表示变量 var_1 和变量 var_2 之间的串联影响关系。其中 var_1，var_2 分别是两个变量的名字，含有 $var_2 = f(var_1)$，f 表示函数关系，def 是 var_1 在程序中的定义行号，$use\text{-}def$ 是 var_1 在程序中的使用行号和 var_2 在程序中的定义行号，use 是 var_2 在程序中的使用行号。

二级变量串联测试覆盖要求测试有足够多的测试案例，使得程序中的每一个变量的二级需求都要被执行到。例如：表 1 为第三部分实例中二级变量串联需求以及测试案例在这些需求中的覆盖信息。(X,2,6,Y,13) 是变量 X 和 Y 的一个串联使用串联关系，2 是 X 的定义行号，6 是 X 的使用行号也是 Y 的定义行号，13 是 Y 的使用行号。

表 1 二级变量串联覆盖信息

Test case	(X,2,6,Y,13)	(X,4,6,Y,13)
t_1	X	
t_2		
t_3		X
t_4		
t_5		

二级变量并联准则(程序中所有相关联的变量间的并联关系) 它可以表示为一个七元组(var_1，def_1，var_2，def_2，

$use\text{-}def, var_3, use)$，此七元组表示变量 var_1 和变量 var_2 之间的并联关系，同时影响变量 var_3 的值。其中 var_1, var_2, var_3 分别是 3 个变量的名字，含有 $var_3 = f(var_1, var_2)$，f 表示函数关系，def_1, def_2 分别是 var_1, var_2 在程序中的定义行号，$use\text{-}def$ 是 var_1, var_2 在程序中的使用行号和 var_3 在程序中的定义行号，use 是 var_3 在程序中的使用行号。

二级变量串联覆盖准则所表示的变量间的串联关系可以是二级，也可以扩充到三级以上，三级即为 3 个变量间的串联关系，如图 4 所示。二级变量并联准则也可以做进一步的扩充，即多个变量间存在并联关系，如图 5 所示，同时影响一个变量的值。

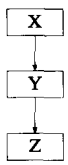

图 4　三级变量串联

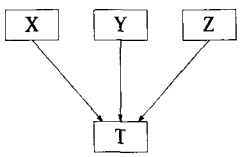

图 5　三级变量并联

3　应用实例及结果对比

实例描述：测试一个具有 4 个参数 a, b, c, d 的函数 $read$，参数取不同的值，得到以下 5 个测试用例，其构成的集合称为 T，如图 6 所示。其程序实例的案例图和流程图如图 7 和图 8 所示。表 2 为 T 在分支覆盖准则下满足各个分支需求的信息。从表中我们可以观察到 T 是充分覆盖集，如何求一个 T 最小子集满足分支覆盖准则下的测试需求。

```
1:   read(a,b,c,d);                 A Branch Coverage Adequate Suite T
B1:  if (a > 0)                     t1: (a = 1, b = 1, c = -1, d = 0)
2:       x = 2;                     t2: (a = -1, b = -1, c = 1, d = -1)
3:   else                          t3: (a = -1, b = 1, c = -1, d = 0)
4:       x = 5;                     t4: (a = -1, b = 1, c = 1, d = 1)
5:   endif                         t5: (a = -1, b = -1, c = 1, d = 1)
B2:  if (b > 0)
6:       y = 1 + x;
7:   endif
B3:  if (c > 0)
B4:      if (d > 0)
8:           output(x);
9:       else
10:          output(10);
11:      endif
12:  else
13:      output(1/(y-6));
14:  endif
```

图 6　程序实例、分支充分覆盖测试集 T

表 2　分支覆盖信息表

Test:Case	B_1^T	B_1^F	B_2^T	B_2^F	B_3^T	B_3^F	B_4^T	B_4^F
t_1:	X		X			X		
t_2:		X		X	X			X
t_3:		X	X			X		
t_4:		X	X		X		X	
t_5:		X		X	X		X	

首先，简单贪婪算法、HGS 分别结合分支覆盖（表 2）、边覆盖（表 3）、定义-使用串联覆盖，全使用覆盖、分支-定义-使用串联组合覆盖得到的最小化测试集为 T'：$\{t_1, t_2, t_4\}$。虽然 T' 满足了各覆盖准则生成的测试需求，但是我们分析程序可以发现只有 t_3 可以发现被 0 除的错误，从而不包含测试用例 t_3 的最小化测试集 T' 的错误检测能力减弱了。

其次，RSR 在分支覆盖信息表上（表 2），运用 HGS 算法在简化集中加入 t_1, t_2 后，发现 t_3 冗余，此时继续在定义-使用串联覆盖中判断 t_3，若存在冗余，则舍去，否则，则保留。可以看到 t_3 对于定义-使用是非冗余的，则 t_3 被保留到最小化测试集中。最后的最小化测试集为 $\{t_1, t_2, t_3, t_4\}$。RSR 算法虽然得到了具有强检测错误能力的简化集，但是其简化的过程要分两部，增加了程序处理的复杂性和计算机内存空间开销。

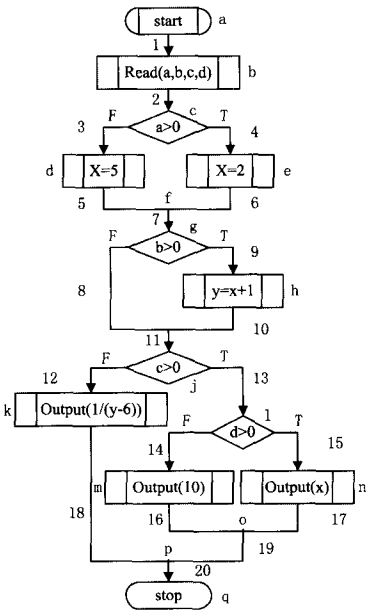

图 7　图 6 程序实例的案例图

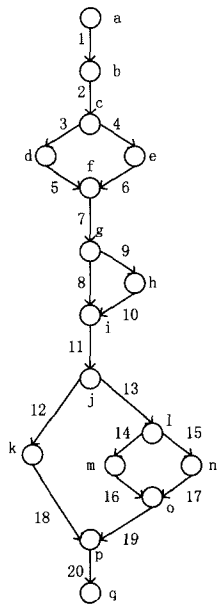

图 8　图 6 程序实例的流程图

表 3　边覆盖信息

Test case	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
t_1	X	X		X		X	X		X	X	X	X						X		X
t_2	X	X	X		X		X	X			X		X	X		X			X	X
t_3	X	X	X		X		X		X	X	X	X						X		X
t_4	X	X	X		X		X		X	X	X		X		X		X		X	X
t_5	X	X	X		X		X	X			X		X		X		X		X	X

表 4　分支-二级变量串联组合覆盖信息

Test case	B_1T	B_1F	B_2T	B_2F	B_3T	B_3F	B_4T	B_4F	(X,2,6,Y,13)	(X,4,6,Y,13)
t_1	X		X			X			X	
t_2		X		X	X			X		
t_3		X	X			X				X
t_4		X	X		X		X			
t_5		X		X	X		X			

本文所采用的方法是，在分支-二级变量串联覆盖组合（表 4）的基础上应用 HGS 算法，得到的最小化测试集为 $\{t_1, t_2, t_3, t_4\}$。经 HGS 算法一次判断，就可以保留 t_3。简化后的测试集就可以发现 t_3 侦测的被 0 除的错误，同时剔除了 t_5 这一冗余项。HGS 结合分支-二级串联覆盖组合原则既简洁高效，又能很好地保持最小化测试集的错误检测能力。

在本例中所运用的简单贪婪算法、HGS 算法，其分别利用分支覆盖、边覆盖、定义-使用串联覆盖、全使用覆盖作为判断准则，所得到的最小化测试集简化结果并不理想。在文献[3]中所提到的 RSR 算法，虽然得到了相对正确的分析结果，但使用了二个步骤才得出了结论。本文在提出的分支-二级串联覆盖组合（表 4）的基础上，运用 HGS，只通过一步分析就

（下转第 272 页）

[11] 陈炜. 基于谓词划分的完备谓词抽象研究. 硕士学位论文. 清华大学,2007
[12] Clarke E M, Grumberg O, Jha S. Counterexample-guided abstraction refinement for symbolic model checking. Journal of the ACM,2003,50(5):752-794
[13] Clarke E M, Grumberg O, Jha S, et al. Counterexample-guided abstraction refinement// Computer Aided Verification, CAV00. Springer-Verlag,2000:154-169
[14] 周旻. 基于惰性抽象的软件模型检验算法研究. 学士学位论文. 清华大学,2007
[15] 杨柳春. 基于反例的抽象细化研究. 硕士学位论文. 清华大学,2007
[16] Clarke E M, Grumberg O, Long D E. Model checking and abstraction. ACM Transactions on Programming Languages and System(TOPLAS),1994,16(5):1512-1542
[17] Dams D, Gerth R, Grumberg O. Abstract interpretation of reactive systems. ACM Transactions on Programming Languages and System(TOPLAS),1997,19(2):253-291
[18] 袁志斌,徐正权,王能超. 软件模型检测中的抽象. 计算机科学,2006,33(7):276-279
[19] Lee W, Pardo A, Jang J, et al. Tearing based abstraction for CTL model checking// International Conference of Computer-Aided Design. 1996:76-81
[20] http://www.kenmcmil.com/foci.html
[21] http://www.cs.ucla.edu/~rupak/Vampyre/
[22] Henzinger T A, Jhala R, Majumdar R, et al. Abstraction from proofs// Proc. of the 31th ACM SIGPLAN-SIGACT Symposium on Principles of Program Languages. ACM Press,2004:232-244
[23] McMillan K L. An interpolating theorem prover// Proc. of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. volume 2988 of LNCS. Springer,2004:16-30
[24] 李力. On the Improvements of Abstract-Check-Refinement Verification. 技术报告
[25] 李力. Fast Finding Abstract Counter-example. 技术报告
[26] 李江. Game Theory-based Optimization of Predicates Abstraction Model Checking. 技术报告

(上接第 246 页)

得出了与 RSR 算法相同的结论。体现了简单高效并保持简化集的错误检测能力的优点。

结束语 本文中基于新颖测试覆盖的测试集简化方法的特点在于:(1)提出了一种新颖的测试覆盖准则,二级变量串联准则;(2)运用上述准则与分支测试覆盖准则相组合,形成了新的测试覆盖需求;(3)在所形成的新的测试覆盖需求的基础上,运用 HGS 算法,针对特定的被测程序取得了较好的测试集简化效果,过程简便并保持了简化后的测试集的错误检测能力。

该准则的扩充是今后要进一步研究的工作,当程序中存在变量繁多、关系复杂的情况时,变量间多重并联关系的表示以及冗余关系的剔除都是研究的难点。

参 考 文 献

[1] Chvatal V. A greedy heuristic for the Set-Covering problem. Mathematics of Operations Research,1979,4(3):233-235
[2] Harrold M J, Gupta R, Soffa M L. A methodology for controlling the size of a test suite. ACM Transactions on Software Engineering and Methodology,1993,2(3):270-285
[3] Jeffrey D, Gupta N. Improving fault detection capability by selectively retaining test cases during test suite reduction. IEEE Transactions on Software Engineering,2007,33(2):108-123
[4] Jones J A, Harrold M J. Test-Suite Reduction and Prioritization for Modified Condition/Decision Coverage. IEEE Trans. Software Eng.,2003,29(3):195-209
[5] Tallam S, Gupta N. A Concept Analysis Inspired Greedy Algorithm for Test Suite Minimization. PASTE,2005:35-42
[6] McMaster S, Memon A M. Call Stack Coverage for Test Suite Reduction// icsm'05. Vol. 00, Sept. 2005:25-30
[7] Rapps S, Weyuker E J. Selecting Software Test Data Using Data Flow Information. IEEE Trans. Software Eng.. 1985,11(4):367-375
[8] Frankl P G, Iakounenko O. Further Empirical Studies of Test Effectiveness// Proc. Sixth ACM SIGSOFT Int'l Symp. Foundations of Software Eng.,Nov. 1998:153-162
[9] Frankl P G, Weiss S N. An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing. IEEE Trans. Software Eng.,1993,19(8):774-787
[10] Hutchins M, Froster H, Goradia T, et al. Experiments on the Effectiveness of Dataflow- and Controlflow-Based Test Adequacy Criteria// Proc. 16th IEEE Int'l Conf. Software Eng.. May 1994:191-200
[11] Frankl P G, Weiss S N. An Experimental Comparison of the Effectiveness of the All-Uses and All-Edges Adequacy Criteria// Proc. Fourth Symp. Testing, Analysis, and Verification. 1991:154-164
[12] Huang Chenliang, Gao Jianhua. Research of tracing dependency based on scenario-driven approach. Jisuanji Gongcheng/Computer Engineering(Language:Chinese),2005,31(19):102-104
[13] Zhu Yuan, Gao Jianhua. Method of improving software reliability by controlling logical complexity. Jisuanji Gongcheng/Computer Engineering(Language:Chinese),2004,30(1):73
[14] Elbaum S, Kallakuri P, Malishevsky A, et al. Understanding the effects of changes on the cost-effectiveness of regression testing techniques. Software testing, verification and reliability, 2003(13):65-83
[15] Mansour N, Bahsoon R. Reduction-based methods and metrics for selective regression testing. Information and Software Technology,2002,44:431-443