Ada软件测试用例生成工具*

奚红宇 徐 红 高仲仪

(北京航空航天大学软件工程研究所 北京 100083)

摘要 软件测试是软件开发过程的一个重要环节,它的主要工作是测试用例的选择.由于人为地选择测试用例带有很大的盲目性和倾向性,因此开发一个能够自动生成测试用例的工具是十分必要的.本文阐述了软件测试用例生成的一些概念和方法,重点讨论了函数最小化方法和插装技术在测试用例生成工具中的应用,并通过一个实例介绍了Ada软件测试用例生成工具TCGT(test case generation tool).

关键词 软件工具,软件测试,测试用例,函数最小化,插装.

中图分类号 TP311

在软件的开发过程中,软件测试是保证软件质量和可靠性的关键,它的核心思想是:对于输入域的特定输入,观察程序的执行结果,验证该结果与期望结果是否一致,然后做相应的纠错和调整.在测试过程中,如何选择有代表性的输入数据(测试用例)来驱动被测程序,使其更完全、更彻底地暴露所存在的错误,是人们最关心的问题.目前,测试人员在选择测试用例时通常都是根据直觉与经验来进行,给测试带来很大的盲目性.同时,测试人员的个性及倾向性使得选择的测试用例仅能测试出其所熟悉的某一方面的错误,许多隐含的其它错误不能被检测出来,这就使软件的质量存在着潜在的不稳定性.

从软件开发的经济性和可靠性考虑,迫切需要改进软件测试的方法,尤其是测试用例的选择方法.鉴于人工选择和编写测试用例的局限性,开发一种能够有针对性的自动生成测试用例的工具是十分必要的.因此,我们根据这一目标,研制并开发了Ada软件测试用例生成工具——TCGT(test case generation tool).该工具能够根据用户选择的测试单元和指定路径(未测试过的)自动生成驱动程序,产生用户所需的测试数据(该数据能够使程序按指定的路径执行),从而改变测试的盲目性,提高测试的质量,对于保证软件的质量和可靠性具有重要的作用.

1 基本概念和方法

下面是TCGT所使用的几个主要概念和方法.

* 本文研究得到国家"八五"攻关项目基金资助.
作者奚红宇,女,1970年生,讲师,主要研究领域为软件工程和软件测试技术.徐红,女,1966年生,讲师,主要研究领域为软件测试,面向对象方法,软件工程.高仲仪,1935年生,教授,主要研究领域为形式语言与自动机理论,软件质量与可靠性,软件及软件工程环境,过程工程.

本文通讯联系人:奚红宇,北京 100083,北京航空航天大学软件工程研究所
本文1996-04-16收到修改稿

1.1 测试用例

测试用例设计的根本目的在于确定一组最可能发现某个错误或某类错误的测试数据.在软件测试中,根据测试准则选择适当的测试用例是软件测试的关键,它决定了软件测试的效率及所能达到的目标.测试用例包括驱动程序、桩程序、测试数据和期望结果.

驱动程序相当于一个主程序,它接收不同的测试数据,并把这些数据传送给被测模块,驱动其运行;桩程序用于代替由被测模块调用的子模块;测试数据是被测模块所需的输入数据;期望结果是被测模块在指定输入下的正确输出.

图1表示了对一个被测模块进行单元测试的环境状况.

TCGT支持驱动程序和测试数据的自动生成.由于我们采用自底向上的测试策略,每一个测试模块均已被测试过,因而不必为被测模块构造桩程序.在测试过程中,测试模块将直接调用其下层模块,并根据不同的返回结果来调整当前的输入数据.

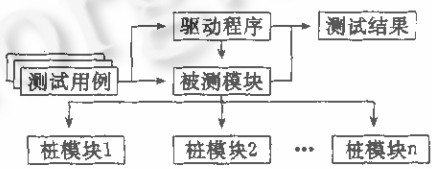

图1 单元测试的运行环境

1.2 测试驱动

由于每个单元在整个软件中并不是孤立的,在对每个模块进行单元测试时,必须考虑它与其它模块的相互联系.一方面,为驱动被测模块运行并建立适当的环境,应构造一个调用被测模块的辅助模块,即驱动程序;另一方面,为使被测模块的功能完整,需构造它所调用的子模块,即桩程序.驱动程序和桩程序的产生就是自动化的测试驱动问题.测试驱动用来模拟被测模块的运行环境,提供明确的测试数据的描述及自动化测试过程.测试驱动是测试用例生成过程中必须解决的一个问题.

在我们的TCGT中,驱动程序是按照Ada程序的结构自动生成的.首先要根据指定的路径对被测模块进行插装,并结合到驱动程序中.其次,由于Ada程序的结构比较复杂,支持嵌套的模块声明(过程、函数、任务、包等),因而每个被测模块都有一个很复杂的运行环境,我们必须在驱动程序中构造和模拟这个环境.由于驱动程序是针对被测模块的指定路径而生成的,因此在我们的TCGT中,每次更换指定路径后,必须重新生成驱动程序.

1.3 测试数据的生成

测试数据的生成是测试用例生成中最为困难的问题.根据一定的测试准则和测试方法,可以确定测试数据的生成方式.现有的测试数据生成方式可以分为3类:面向路径的测试数据生成方式;数据规格说明方式;随机测试数据生成方式.在我们的TCGT中,由于它是作为动态测试工具的辅助工具,为某些难于覆盖的执行路径提供测试数据,因而采用了面向路径的测试数据生成方式.面向路径的测试数据生成方式是通过执行被测程序单元的一条路径来派生出测试数据,它又可分为程序直接执行方式和符号执行方式.

· 程序直接执行方式是在被测单元中选定一条路径之后,根据输入域随机地选取测试用例初始值,经程序的多次循环探测执行,最终确定测试数据.

· 符号执行方式是根据覆盖准则(如语句覆盖、条件覆盖、判定覆盖及路径覆盖等准则)抽取出逻辑路径,并对每条逻辑路径进行符号运算,产生该路径的由所有分支条件构成的谓词方程组,所有满足该方程组的输入数据(测试数据)都能驱动该路径的执行.但由于解谓词方程组的复杂性(在理论上是一个不可解问题),以及存在数组下标、循环次数、递归过

程的执行次数在静态时无法确定等问题而难于实用化.

在我们的 TCGT 中,采用的是程序直接执行方式,利用函数最小化方法,逐步探测执行被测模块,最终确定测试数据.

1.4 插　装

插装是动态测试中常用的一种技术. 它是在程序中插入一些探针,当被测程序运行时,就可以记录程序执行的路径、状态等信息. 插装的前提是不能影响原程序的逻辑结构和功能. 在我们的 TCGT 中,为了了解被测程序的执行情况,确定是否覆盖了指定的路径,必须对被测程序进行插装,以便判断所提供的测试数据是否合适,是否满足了测试的准则. 我们严格按照 Ada 文法对被测模块进行静态分析,针对各种语法现象制定相应的插装方案,以保证所收集信息的完整性和正确性.

1.5 分支函数

分支函数[1]是一个实值函数,它是分支谓词到实值的一个映射. 构造分支函数的目的是为了解决程序直接执行过程中的分支冲突问题(后面将详细介绍),这是用函数最小化方法来产生测试数据的一个关键步骤,它的选择与构造对探索寻找测试数据的速度有很大影响. 分支函数应能真实反映分支谓词的真伪情况,且分支函数的内部变化特性是良好的. 如下所示:

分支谓词	分支函数
True	$F(x)=-1.0$
$E_1(x)>E_2(x)$	$F(x)=E_2(x)-E_1(x)$
$E_1(x)\geqslant E_2(x)$	$F(x)=E_2(x)-E_1(x)$
$E_1(x)<E_2(x)$	$F(x)=E_1(x)-E_2(x)$
$E_1(x)\leqslant E_2(x)$	$F(x)=E_1(x)-E_2(x)$
$E_1(x)=E_2(x)$	$F(x)=abs(E_1(x)-E_2(x))$
$E_1(x)\neq E_2(x)$	$F(x)=-abs(E_1(x)-E_2(x))$
$E_1(x)$ and $E_2(x)$	$F(x)=\mathrm{Max}(F_1(E_1(x)),F_2(E_2(x)))$
$E_1(x)$ or $E_2(x)$	$F(x)=\mathrm{Min}(F_1(E_1(x)),F_2(E_2(x)))$
not $(E_1(x))$	$F(x)=-F_1(E_1(x))$

$F(x)$是一个实值函数,当分支谓词为假时,$F(x)$为正(或零);当分支谓词为真时,$F(x)$为负(或零).

2 程序直接执行和函数最小化方法

2.1 程序直接执行方法

这是一种面向路径的测试数据生成方式. 在被测程序单元中选定一条路径 $P=\langle n_1,\cdots,n_q\rangle$,然后根据输入域的定义随机地选取测试数据初始值,经程序多次循环探测执行后,最终确定测试数据,使其覆盖路径 P. 直接执行的方法为:

(1) 随机输入初始测试数据 X^0,执行程序. 如果指定的路径 P 被覆盖,则 X^0 即为所需的测试数据.

(2) 如果未覆盖路径 P,则记 $P_i=\langle n_{k1},\cdots,n_{ki}\rangle$ 为 X^0 上执行路径所经过的 P 的最大子路径,其中 $n_{k1}=n_1,n_{k2}=n_2,\cdots,n_{ki}=n_i,1\leqslant i\leqslant q$.

(3) P_i 下一相邻节点为 n_{ki+1}，分支冲突出现在分支(n_{ki},n_{ki+1})上. 在此点计算分支函数 $minfun(i+1)$，调整输入值 X^0 为 X，使 $minfun(i+1)$的改变最有利于解决(n_{ki},n_{ki+1})上的分支冲突，最终程序将覆盖边(n_{ki},n_{ki+1})，且覆盖 P_i. 这样，程序所覆盖的 P 的最大子路径将增长为 $P_{i+1}=\langle n_k,\cdots,n_{ki},n_{ki+1}\rangle$.

(4) 重复上述操作，直到找到覆盖整个路径 P 的测试数据. 若找不到，则 P 为不可执行路径——死路径.

2.2 函数最小化方法

函数最小化方法也称步长加速法，亦称爬山法.[2]我们在程序直接执行时，使用这种方法来解决分支冲突. 当输入 X^i 使程序的执行路径覆盖指定路径 P 的最大子路径 P_i 时，此点的分支函数为 $F_i(x_i)$，通过函数最小化方法的步长探索和模块探索，寻找新的输入 X_{i+1}满足 $F_i(X_{i+1})\leqslant F_i(x_i)$，并重复这种探索过程，直到找到 X^*，使函数 $F_i(x)$达到最小，即 $F_i(X^*)\leqslant 0$，且覆盖子路径 P_i. 根据分支函数的构造方法，当 $F_i(X^*)\leqslant 0$ 时，就表示程序的执行路径已覆盖了更长的子路径 P_{i+1}，从而解决了分支冲突，找到了更有利的测试数据.

在我们的测试用例生成工具中，采用了函数最小化方法的一维搜索过程. 该过程包括2个主要阶段：

(1)探测性搜索. 在这种搜索中，输入变量 X^i 先以一个很小的步长增大或减小. 这时监测程序的执行情况，计算分支函数，以便确定变量 X^i 的前进方向(增大或减小)是否正确. 若已探测出变量 X^i 的前进方向，则进行下面的模式性搜索.

(2)模式性搜索. 在这种搜索中，由于已探测出变量 X^i 的前进方向，因此将 X^i 在此方向上做一个较大的移动，加大步长，只要每次模式移动都能够引起分支函数的改善，那么一系列的模式移动就会沿着这个方向进行下去. 如果分支函数没有改善，那么就要通过探测性搜索重新指示一个新的方向.

函数最小化方法是一种传统的搜索方法，善于找到局部最优解. 它使用迭代改进法，每次迭代从现时点的邻域选择一个新的点. 显然，在多峰搜索空间中，它易于陷入假的局部峰值，该峰值是否为全局最优解依赖于起始点的选择. 因此，我们在选择初始输入数据时，采用自动生成和人工输入相结合的方法，通过大量的起始点来改进解的局部性.

3 Ada 软件测试用例生成工具 TCGT

TCGT 是我们利用函数最小化方法开发的一个测试用例生成工具. 该工具作为 Ada 软件动态测试工具的一个辅助工具，实现了自动生成驱动程序和测试数据的功能，为提高动态测试的质量提供了有力支持.

下面我们通过一个实例来说明 TCGT 的工作过程. 例1是一个被测的 Ada 源程序.

例1：----------Formated_ada_program----------

```
1  procedure IFMAIN is
2    A, B:INTEGER;
3    procedure IFSUB(C: in INTEGER) is
4      VAR:INTEGER;
5    begin
6      if A>C then
7        if A>20 then
```

```
8          VAR:= 20;
9        end if;
10     else
11       if C< -10 then
12         VAR:= -10;
13       end if;
14     end if;
15   end IFSUB;
16 begin
17   A:= 2;
18   B:= 3;
19   IFSUB(B);
20 end IFMAIN;
```

在这个例子中,包括一个主模块 IFMAIN 和一个子模块 IFSUB,我们选择过程 IFSUB 作为被测模块. 按照 Ada 的语义及 TCGT 中的定义,该模块共有 8 条逻辑路径,当用户选择了某一条逻辑路径后,TCGT 就会自动为其生成驱动程序,包括为被测模块构造运行环境,对被测模块进行插装,及在指定的逻辑路径上插分支函数. 驱动程序的框架结构如例 2 所示.

例 2:

```
LIB_LISTS;                          -- 可见的外部模块;
procedure IFSUB_DRIVER_MAIN is
  C,A:INTEGER;
  procedure IFSUB is                -- 被测模块;
    VAR: INTEGER;
  begin
    MINFUN(1) := -1.0;              -- 分支函数;
    STUB_FUNC(1);                   -- 插装函数;
    MINFUN(2) := FLOAT(C-A);
    if A>C then
      STUB_FUNC(2);
      MINFUN(3) := FLOAT(20-A);
      if A>20 then
        STUB_FUNC(3);
        VAR:= 20;
        MINFUN(4) := -1.0;
      end if;
      STUB_FUNC(5);
    else
      STUB_FUNC(6);
      if C<-10 then
        STUB_FUNC(7);
        VAR:=-10;
      end if;
      STUB_FUNC(9);
      MINFUN(5) := -1.0;
      end if;
    STUB_FUNC(10);
    MINFUN(6) := -1.0;
  end IFSUB;
begin
  READ_INITIAL_VALUE;               -- 产生测试数据的初始输入值;
  loop
    IFSUB;                          -- 调用被测模块;
    DECIDE_PATH;                    -- 判断所执行的路径是否与指定的路径吻合;
```

```
        MINFUNED;                    -- 用函数最小化方法调整测试数据;
    end loop;
end IFSUB_DRIVER_MAIN;
```

在操作系统下编译、连接、运行该驱动程序. 在执行过程中,TCGT 会根据被测模块的执行情况不断调整测试数据,最后产生覆盖指定路径的测试数据,如例 3 所示.

例 3:
```
Exec_Line_No :
        5 6 7 9 14 15
Test_Data :
        C: 0
        A: 21
```

4 结束语

本文介绍了测试用例生成的一些概念和方法,着重讨论了如何用函数最小化方法来解决测试数据的生成问题,并介绍了一个实际的 Ada 软件测试用例生成工具 TCGT. 这一工具的开发为动态测试提供了强有力的支持,并为实现测试的自动化创造了条件. 当然,本工具只是一个实验系统,在很多方面还有待于完善和进一步研究. 我们希望通过此文与有关的专家共同探讨,力图使这一课题的研究有更大突破.

致谢 北京航空航天大学计算机系的彭建军同学、杨芳同学和吴银树同学为本工具的开发与实现做了大量的工作,在此对他(她)们表示衷心的感谢.

参考文献

1 Bogdan Korel. Automated_software test data generation. IEEE Trans. on Software Eng., 1990,16(8).

2 魏权龄,王日爽,徐兵. 数学规划引论. 北京:北京航空航天大学出版社,1991.

ADA SOFTWARE TEST CASE GENERATION TOOL

XI Hongyu XU Hong GAO Zhongyi

(*Software Engineering Institute Beijing University of Aeronautics and Astronautics Beijing* 100083)

Abstract Software testing, whose main task includes the selection of test cases, is very important in software development. Since selecting test cases manually is usually blind and with great tendency, it is necessary to develop a tool which can generate test cases automatically. This paper first introduces some concepts and methods used in Ada software test case generation, and then puts emphasis on the application of function minimization method and instrumentation techniques, finally it describes the design and implementation of TCGT——the Ada software test case generation tool.

Key words Software tool, software test, test case, function_minimization, instrumentation.

Class number TP311