带数组和循环的路径测试数据自动生成技术研究*)

陈继锋
(湖南涉外经济学院计算机学部 长沙 410205)

摘 要 提出了一种新的带数组和循环的路径测试数据自动生成的方法。该方法只考虑数组中与路径中谓词函数有关的数组元素,将循环中的同一变量名在每一次执行时用不同的变量参数来替代,从而较好地解决了路径中数组循环有效处理的问题。为有效、简单地自动生成测试数据,建立了谓词函数关于输入变量的线性约束系统。当谓词函数为线性表达式时,不需要计算其线性算术表示,仅计算非线性函数谓词函数的线性算术表示,且不需计算路径中的谓词片和确定输入依赖集,以及构造谓词函数关于输入变量增量的线性约束系统。理论分析和实例验证该方法具有简单、直观、有效且计算量小等特点。

关键词 数组和循环,测试数据,谓词函数,线性算术表示

Research on Path-based Automatic Test Data Generation with Arrays and Loops

CHEN Ji-feng
(Department of Computer, Hunan International Economics University, Changsha 410205, China)

Abstract A new approach is proposed for path-based automatic test data generation with arrays and loops. The approach, in which only the array elements related to the branch predicates on the path is selected and all branch predicates on the path are simutaneously considered, is adopted to construct the linear constrain system. The loop is outspreaded on the given path. The same variable is replaced by different variables in each execution of the loop. If the predicate function is linear, it is not needed to compute the linear arithmetic representation of predicate function. Otherwise, this computation is needed. Also in our method, it's not needed to compute the slices, identify the input dependency sets and construct the linear constraint system of predicate function on the increments for the input. Theoretic analysis and practical testing show that our method is simpler, easier, more effective and less cost in computing than the others.

Keywords Arrays and loops, Test data, Predicate function, Linear arithmetic representation

1 引言

软件测试在软件制造业中变得越来越重要,其所占比重也越来越高。软件测试的自动化则使软件工程师们从繁重而枯燥的测试工作中解脱出来,因而具有巨大的发展前途。测试数据的自动生成,则是软件测试自动化最主要、最关键的一环。基于路径的测试数据自动生成是白盒测试的一项重要工作,其测试数据的生成是一项艰巨的任务。而含有数组、循环的路径测试数据的自动生成,则使求解问题更加复杂化。所以,如何较好地解决这一问题就成了软件测试界普遍关心的问题。文献[1]提出了采用函数最小化和回朔的方法可有效地解决这一问题,但它每一次仅考虑一个分支和一个输入变量,在回溯过程中将造成巨大的资源浪费,特别是当数组较大时,这种浪费将会成倍地增加。文献[2]提出了采用迭代方法,只考虑数组中与路径谓词函数有关的元素,并同时考虑路径中各个分支约束条件来建立和求解线性约束系统,获得测试输入数据,因而较好地简化了带数组、循环路径的测试数据自动生成。但该方法需计算路径中的谓词片和输入依赖集,以及非线性函数的线性算术表示,同时还需建立谓词函数关于输入变量增量的线性约束系统,因而还是较为繁琐。

基于以上情况,本文提出了一种新的求解带数组、循环的路径测试数据自动生成的方法,该方法同样只考虑数组中与路径中谓词函数有关的数组元素,并同时考虑路径中各个分支约束条件,但仅建立谓词函数关于输入变量的线性约束系统,来获得测试输入数据。它不需计算路径中的谓词片和输入依赖集,以及线性谓词函数的线性算术表示,也不需构造谓词函数关于输入变量增量的线性约束系统,仅当谓词函数为非线性函数时才计算其线性算术表示,因而具有简单、直观、计算量小等特点。

2 理论分析

```
Var
A: array[1,...,100] of integer;
low, high, step: integer;
min, max, i: integer;

1:   input (low, high, step, A);
2:   min:=A[low];
3:   max:=A[low];
4:   i:=low+step;
P1   while i<high do
P2:      if max < A[i] then
5:           max:= A[i];
P3:      if min > A[i] then
6:           min:= A[i];
7:       i:=i+step;
  endwhile;
8:   output(min, max);
```

图 1 程序示例

*)国家 863 高技术研究发展计划基金项目(2003AA1Z2610),湖南省教育厅资助科研项目(07A034)。陈继锋 博士,副教授,从事软件测试自动化研究。

以图1所示Pascal程序为例，来分析讨论带数组、循环的路径测试数据的自动生成。

带数组、循环的路径测试数据的自动生成较一般程序路径要复杂。为实现其测试数据自动生成，我们以文献[1-3]为基础，从以下几个方面对新的方法进行理论分析。

(1)数组元素的选取。数组的大小是产生测试数据的主要问题，并不是数组的每个元素都将被路径中的谓词函数所使用。如何选取数组元素，以及相关元素值的确定，将是测试数据产生的关键。直接搜索法[4]是将路径P分成若干子路径，从程序开始到每一个含有数组元素的谓词函数所在的语句各为一条子路径。如图1所示程序中，设P={1,2,3,4,$P1_1$,$P2_1$,$P3_1$,7,$P1_2$,$P2_2$,$P3_2$,6,7,$P1_3$,8}，则相应的子路径为P1={1,2,3,4,P1,P2}，P2={1,2,3,4,$P1_1$,$P2_1$,$P3_1$}，P3={1,2,3,4,$P1_1$,$P2_1$,$P3_1$,7,$P1_2$,$P2_2$}，P4={1,2,3,4,$P1_1$,$P2_1$,$P3_1$,7,$P1_2$,$P2_2$,$P3_2$}。从初始输入I_0中，依次选取每个数组元素来执行子路径P1，直至所选的数组元素能使P1被经过，再执行子路径P2，同样求出能使P2被经过的数组元素。如此反复，直到P中所有子路径均被经过，则最终所得的数组元素值的集，即为所求测试数据。这种方法不考虑数组元素是否与分支函数相关，带有极大的盲目性，导致巨大的资源浪费。为此，本文采用一种基于动态数据流分析的方法[1]，仅考虑数组中与被执行路径中的谓词函数有关数组元素，这就使得测试数据的生成与数组的大小无关，从而达到了简化求解的目的。

(2)循环的处理。对于循环的处理，较数组容易一些。本文采用将循环按条件展开的方法，形成一条完整的路径来求解测试数据，如前面所设图1中的路径P，共执行了两次循环。由于在循环的展开过程中，变量在循环的每一次执行时，其数值都是动态变化的，因此要将这些因素进行考虑。具体做法是：将循环中的同一变量名在每一次次执行时设置不同的变量参数来替代。

(3)分支谓词函数约束条件的建立。文献[1]求解测试数据，每一次仅考虑一个分支和一个输入变量，并分成多个子路径，使每个子路径均被经过，因而非常繁琐。为解决这一问题，本文同时考虑路径中的各个分支约束条件，来建立整个路径中各个分支谓词函数的线性约束[2]，然后求解线性约束系统，因而使求解问题大大简化，较好地实现了带数组、循环路径的测试数据自动生成。

(4)谓词函数的线性算术表示。针对文献[2,3]中，对所有谓词函数都要计算其线性算术表示，本文将路径中的线性谓词函数直接作为线性算术表示来构造谓词函数关于输入变量的线性约束。仅当谓词函数是输入变量的非线性函数时，才用均差近似导数的方法来计算其线性算术表示。为保证新方法的正确性，特提出以下定理，并给出证明。

定理1 如果路径P中的谓词函数是线性的，则其线性算术表示就是谓词函数本身。

证明：由已知谓词函数是线性的，故可设谓词函数为

$$F(X)=a_1X_1+a_2X_2+\cdots+a_iX_i+\cdots+a_mX_m+a_0$$

其中：X_i为输入变量，a_i为系数，m为输入变量个数，$i\in(1,2,\cdots,m)$，a_0为常数项。

设k为迭代次数，$k\in(0,1,2,\cdots,T)$，则第k次迭代的$F(X)$关于$X_{i,k}$的均差为

$$F[X_{i,k+1},X_{i,k}]=\frac{F(X_{i,k+1})-F(X_{i,k})}{X_{i,k+1}-X_{i,k}}$$

$$=\frac{a_{i,k}(X_{i,k+1}-X_{i,k})}{X_{i,k+1}-X_{i,k}}=a_{i,k}$$

$F(X)$关于$X_{i,k}$的导数为

$$F'_{X_i}=a_{i,k}$$

可见，用均差求得的线性谓词函数的变量系数，等于$F(X)$的一阶导数，而与输入变量无关，因此其线性表示即为谓词函数$F(X)$本身，定理1得证。

(5)谓词函数关于输入变量线性约束的构造。本文采用建立谓词函数关于输入变量线性约束的方法，不构造谓词函数关于输入变量的增量的线性约束，因而不需计算谓词片和确定输入依赖集。相关理论证明如下。

定理2 谓词函数关于输入变量的线性约束，与谓词函数关于输入变量增量的线性约束等价。

证明：设谓词函数为(其中各参数意义同定理1)

$$F(X)=a_1X_1+a_2X_2+\cdots+a_iX_i+\cdots+a_mX_m+a_0$$

$$=\sum_{i=1}^{m}a_iX_i+a_0$$

由迭代松弛法知，$F(X)$第k次迭代的谓词残量为

$$R(X_{i,k})=\sum_{i=1}^{m}a_{i,k}X_{i,k}+a_{0,k}$$

$F(X)$第k次迭代的关于输入变量的增量的线性约束表达式为

$$\sum_{i=1}^{m}a_{i,k}\Delta X_{i,k}+R(X_{i,k})\ \text{op}\ 0 \qquad (1)$$

其中 op$\in(<,\leqslant,>,\geqslant,=,\neq)$

$\because\Delta X_{i,k}=X_{i,k+1}-X_{i,k}$

将$\Delta X_{i,k}$和$R(X_{i,k})$代入(1)式，可得

$$\sum_{i=1}^{m}a_{i,k}X_{i,k+1}+a_{0,k}\ \text{op}\ 0 \qquad (2)$$

(2)式即为谓词函数$F(X)$关于输入变量的线性约束表达式，其求得的$X_{i,k+1}$即为第$k+1$次迭代的输入变量值I_{k+1}。由于(2)式是由(1)式所推导来的，所以(1)式和(2)式等价，故定理2得证。

3 算法描述

根据第2节中的理论分析，新方法的主要思想是：先将数据组、循环进行初始化处理，再从输入变量的值域中任选一组输入，考察给定路径上各分支谓词。当谓词函数是线性表达式时，直接将其作为线性算术表示来构造线性约束；当谓词函数是非线性表达式时，则计算其关于当前输入的线性算术表示，然后将得到的线性算术表示与路径上的线性谓词函数一起构造输入变量的线性约束系统，进而建立输入变量的线性方程系统，求解出输入变量值，从而获得一组新的输入。若新的输入仍不能使给定的路径被经过，则重复以上过程，直至求出所期望的输入，或达到规定的迭代次数上限，求解结束。

具体算法描述如下：

```
输入：路径 P={n1,n2,…,nk}，程序的初始输入 I0 以及迭代次数上
      限 T；
输出：使路径 P 被经过的程序输入 If；
数组、循环初始化处理；
procedure TESTGEN(P,I0)
  if I0 能让 P 被经过 then If= I0 return
    k=0; Done=false
  while (not Done) and (k≤T) do
    for P 上每个谓词结点 ni do
      if 路径 P 上 ni 的谓词函数为非线性函数 then
step1          计算 ni 的谓词函数的线性算术表示 L(ni,Ik,P)
      endif
    endfor
step2  用 P 中的线性谓词函数和 L(ni,Ik,P)构造输入变量的线性约
       束系统
step3  求解线性约束系统，得到新的输入 Ik+1
```

```
    if I_{k+1}能让 P 被经过 then I_f = I_{k+1};
      Done=true
    else k++ endif
    endwhile
  endprocedure
```

4 实例分析

4.1 应用实例

考虑到文献[1,2]所提方法的比较,我们亦采用图 1 所示的程序。同样取 $P=\{1,2,3,4,P1_1,P2_1,P3_1,7,P1_2,P2_2,P3_2,6,7,P1_3,8\}$, $I_0=(low=39,high=93,step=12,A[1]=1,\cdots,A[100]=100)$。

为简单起见,令 l,h,s 分别表示 low,high,step。由于每次执行循时,数组元素 $A[i]$是变化的,故根据路径 P,可令 $X=A[l],Y=A[l+s],Z=A[l+2s]$。执行算法:

由于 I_0 不能使 P 通过,故继续执行算法后面的步骤。

因 P 中谓词函数 $P1_1,P2_1,P3_1,P1_2,P2_2,P3_2,P1_3$ 都是线性的,所以直接执行算法的第 2 步,构造谓词函数关于输入变量 I_0 的线性约束系统:

step2

$$\begin{cases} l+s-h<0 \\ X-Y\geqslant 0 \\ X-Y\leqslant 0 \\ l+2s-h<0 \\ X-Z\geqslant 0 \\ X-Z>0 \\ l+3s-h\geqslant 0 \end{cases}$$

step3 求解线性约束系统。

设 $a,d,f>0,b,c,e\geqslant 0$,则线性约束系统可变成以下线性方程组:

$$\begin{cases} l+s-h=-a \\ X-Y=b \\ X-Y=-c \\ l+2s-h=-d \\ X-Z=e \\ X-Z=f \\ l+3s-h=g \end{cases}$$

令 $a=d=f=1,b=c=e=g=0$,根据文献[5]中求解线性约束系统的方法,求线性方程组的最小二乘解,可求得 $l=1,h=4,s=1,X=2,Y=2,Z=1$。由于数组其它元素与路径中分支谓词函数无关,可令其与 I_0 相同,故新的输入 $I_1=(low=1,high=4,step=1,A[1]=2,A[2]=2,A[3]=1,A[4]=4,\cdots,A[100]=100)$。

由于 I_1 可使路径 P 被经过,故算法结束。

我们使用文献[2]的方法和本文所提出的方法,在系统配置为 CPU P4 1.6G; 512M DDR; OS Red Flag 4.1 的计算机上,对上述实例分别进行了实验,其实验结果(两种方法的运行时间)见表 1。

表 1 实例的执行时间 (ms)

实验次数	1	2	3	4	5	平均时间
文献[2]方法	60	70	60	60	70	64
本文方法	30	30	30	40	30	32

4.2 结果分析

从上述实例中可看出,对于同一程序中的同一路径,其测试数据的生成,文献[1]执行了 21 次,而本文的方法仅需执行 8 次,计算量大大减少,可节省大量的资源,因而具有求解速度快、简单直观等特点。同时,迭代 1 次即可求得所需测试数据,或确保路径不可达。与文献[2]比较,虽然两者都只需执行 8 次,但本文的方法不必计算路径中的谓词片和确定输入依赖集,也不必计算谓词函数的线性算术表示、谓词残量以及构造谓词函数关于输入变量的增量的线性约束,因而计算量大大减少。从实验的情况看,文献[2]的方法求解测试数据的平均时间为 64ms,而本文的方法仅为 32ms。可见,理论分析和实验结果均表明:本文的方法明显优于文献[1,2]的方法。

结束语 本文提出了一种新的带数组、循环的路径测试数据自动生成的方法。该方法对于给定的路径能很好地自动地生成测试数据,较好地解决了带数组、循环的路径测试数据的自动生成的问题,并能容易实现工具化。它比目前其它方法更有效。由于本方法同时考虑了路径中所有的分支谓词,所以对于线性约束路径,通过一次迭代即可求出测试数据,或确保路径不可达;对于含有非线性约束的路径,通过多次迭代亦可求出测试数据,或在相当程度上能保证路径不可达。另外,该方法同样适用于其它的语言程序的路径测试数据的自动生成。

参 考 文 献

[1] Korel B. Automated Software Test Data Generation[J]. IEEE Transactions on Software Engineering, 1990, 16(8): 870-879

[2] Neelam Gupta Aditya P. Mathur Mary Lou Soffa. Automated Test Data Generation Using An Iterative Relaxation Method [A]// ACM SIGSOFT Sixth International Symposium on Foundations of Software Engineering (FSE-6). Florida, United States: ACM, 1998: 231-244

[3] 单锦辉. 面向路径的测试数据自动生成方法研究[D]. 长沙:国防科学技术大学研究生院,2002

[4] Glass H, Cooper L. Sequential search: A method for solving constrained optimization problems[J]. J. ACM, 1965, 12(1): 71-82

[5] Neelam Gupta Aditya P. Mathur Mary Lou Soffa. UNA based Iterative Test Data Generation and its Evaluation[A]// 14th IEEE International Conference on Automated Software Engineering (ASE'99). Cocoa Beach, Florida, USA, October 1999: 224-232

(上接第 243 页)

现 IEEE 的标准 PSL(属性描述语言)到自动机的转换。

参 考 文 献

[1] 易锦,张文辉. 从基于迁移的扩展 Büchi 自动机到 Büchi 自动机. 软件学报,2006,17(4)

[2] Gastin P, Oddoux D. Fast LTL to Buchi Automata Translation, Lecture Notes In Computer Science; Vol. 2102 archive // Proceedings of the 13th International Conference on Computer Aided Verification. ISBN: 3-540- 42345-1, 2001: 53-65

[3] Vardi M Y. An Automata-Theoretic Approach to Linear Temporal Logic. www. cs. rice. edu/~vardi/papers/banff94rj. ps. gz

[4] 蒋立源,康目宁. 编译原理(第二版). 西北工业大学出版社,2003