一种基于分支覆盖的测试数据自动生成算法*)

陈继锋[1] 朱 利[2] 沈钧毅[1] 王志海[1]
(西安交通大学计算机软件研究所[1] 软件学院[2] 西安 710049)

摘 要 通过构造新的程序流图,利用Fibonacci法优化选取路径,为指定的分支生成测试数据。提出了路径测试数据生成代价的概念,并给出了代价的计算方法。当所选路径的分支谓词均为线性表达式时,直接求解线性约束集即可生成测试数据,或判定路径不可行;当分支谓词含有非线性表达式时,利用均差近似导数将非线性函数线性化,通过简单的迭代,亦能容易生成测试数据或判定路径在很大程度上不可行。若所选路径不可行或在很大程度上不可行,则选取新的路径,重复以上过程,直至求出所期望的数据,或无新的路径被选取,给定分支不可达。实例和实验表明,算法可行、有效。

关键词 分支覆盖,谓词函数,线性算术表示

Automatic Test Data Generation Algorithm on Branch Coverage

CHEN Ji-Feng[1] ZHU Li[2] SHEN Jun-Yi[1] WANG Zhi-Hai[1]
(Institute of Computer Software[1], School of Software[2], Xi'an Jiaotong University, Xi'an 710049)

Abstract In order to generate test data for a given branch, a new program flow graph is constructed, and Fibonacci method is used to optimize the path selection. The conception and calculation to path-based test data generation complexity is presented. If all branch predicates on the selected path are linear expression, the linear constrain set is constructed and solved to generate test data, otherwise the path can be determined infeasible. Else, the derivatives of a predicate function is approximated by its divided difference, and the nonlinear function is linearized. The test data can also be generated by simple iteration or the path is determined as likely infeasible. If the selected path is infeasible or likely infeasible, a new path is selected. The process above is repeated until the desired test data is obtained or there is no new path to be selected, and the given branch is infeasible. Example and experiment show that the algorithm is feasible and effective.

Keywords Branch coverage, Predicate function, Linear arithmetic representation

1 引言

软件结构测试是软件测试中的一个重要部分,它需要生成数据来覆盖程序中被选定的特定元素,如语句覆盖[1]、分支覆盖[2]、数据流覆盖[3]以及路径覆盖[4,5]等。结构测试中一个主要的问题就是如何产生输入数据来使被选定要测试的元素被执行。对于路径覆盖,因为路径是给定了的,这样当所选输入不能使路径被通过时,只能通过改变输入数据来使路径被经过。但对于语句覆盖、分支覆盖等,测试数据的产生有很大的灵活性,除了能改变输入数据,还能选择不同的路径来产生测试数据。所以,当程序中的许多路径能被用来执行被测程序元素时,路径的选择是一个非常重要的因素。本文主要是讨论结构测试中的分支覆盖,当然所提出的方法也能用于语句和数据流覆盖,即通过动态选取路径来产生经过给定语句或定义-使用对的测试数据。

2 相关工作

目前,已有很多方法都能为分支覆盖产生测试数据。其中一种是通过使用一些输入数据每次改变一个分支的输出来动态地产生路径。这种方法采用函数极小化,来改变输入数据,试图使选定的分支被经过[6]。文[7]使用数据依赖分析对文[6]的方法进行了改进,通过附加一控制流信息来指导测试数据的产生。在这两种方法中,当控制流不能到达选定的分支时,相关的分支谓词函数将被极小化并以此来改变输入,使所选定的分支被经过。该方法使用函数极小化,一次改变一个输入变量,并且不断循环,直至一个方案被获得或没有程序被使用。如果不成功,则回溯到前面的谓词,并重复以上过程。当从开始节点到被测分支的路径集中包含大量的分支谓词时,这种方法将使程序的执行次数大得令人无法接受。

文[8]描述了一个结合路径选取来产生测试数据的域测试算法,它更有效地使用文[9]中描述的函数极小化方法,并只需从输入域中选取较少的输入。但是对每一个受影响的变量而言,从输入域所选的点数,是一个能被赋予输入变量不同值的最大函数,在一般的程序中,它是非常大的。

文[2]的方法开始于一个程序中给定的分支和程序输入域中任选的一个输入,如果程序输入不能经过所选的包含待测分支的路径,那么通过改变分支谓词的输出来选取新的路径,然后计算当前路径和被选新路径的阻力系数,并进行比较。如果新路径对测试数据的生成具有较小的阻力,则下次迭代使用新的路径。否则,仍使用当前路径来迭代提炼输入。

*)国家863高技术研究发展计划基金项目(2003AA1Z2610)。陈继锋 博士生,从事软件测试自动化研究;沈钧毅 教授,博士生导师,从事软件工程、数据挖掘等研究。

该方法在路径选择上提出了计算路径阻力系数的思想，在一定程度上解决了路径的选择问题，但路径的选取仍具有较大的盲目性。另外，该方法采用了文[10]的方法来求解路径上分支谓词的线性约束集，故需计算谓词片、谓词残量，以及确定输入依赖集，因而计算量仍是非常大。

文[9,10]的方法是对一条特定的包含待测分支的路径生成测试数据，进行路径覆盖，从而达到对分支的覆盖。如果这条路径不通，则另选一条路径，重复以上过程来产生测试数据，直到获得所求测试数据，或不再有新的路径被选取。这种方法最大的不足就是不可达路径的影响。如果一条不可达的路径被选取，那么在更新的路径之前，将有大量的计算被浪费。

针对上述方法存在的不足，本文提出了一种新的方法，通过优化选取路径来为指定的分支生成测试数据，并通过实例和实验进行了验证。

3 算法设计

以图1中的程序流图G为例。设图中分支B为待测分支，那么算法要解决的问题就是要求出能使分支B被经过的程序输入。

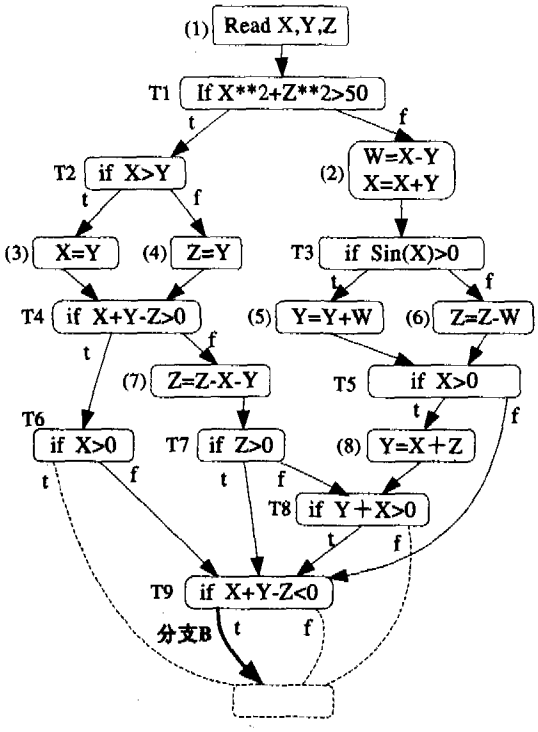

图1 一个程序流图G

3.1 新流图的构造

设给定的程序流图为G，为求经过待测分支B的程序输入I，我们构造一个新的程序流图G'，G'包含G中所有的从开始节点到分支B的所有路径（如图1中实线边部分所示）。这样只需为G'中任意一条路径产生测试数据，即可覆盖分支B。

3.2 路径的选取

本文在文[2]的基础上，提出测试数据生成代价的计算公式。

定义1 设路径P测试数据生成的代价函数为

$R(P)=re^{t}$

其中：r为路径P上的分支谓词数；

t为路径P上非线性分支谓词数。

可见，路径P上的分支谓词越多，其测试数据生成的代价（以下简称代价）也就越大，而非线性分支谓词的增加，则使生成代价呈指数增加。

对G'进行分析，设从开始节点到分支B的所有路径为k条，分别为$P_1,P_2,P_3\cdots P_k$。当路径数k非常大时，如果求出所有路径的代价来寻找代价最小的路径，将耗费大量的时间和资源。为此，我们采取一种优化的方法来寻找一条代价尽可能小的路径。

由于G'中所有的路径是离散的和随机的，故采用Fibonacci法[11]来优化选取路径P_y。引入Fibonacci数列，即序列1,1,2,3,5,8,13,21,34,……其中每个数都是前两者之和，用数学模型表示为：

$$\begin{cases}F_n=F_{n-1}+F_{n-2} & n\geqslant 3\\ F_1=F_2=1\end{cases} \tag{1}$$

用母函数法可求得

$$F_n=\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

$$\approx\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \tag{2}$$

令$F_{n-1}<k\leqslant F_n$

即$$\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n-1}<k\leqslant\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \tag{3}$$

可求得n（取整数值），由(1)和(2)生成Fibonacci数列$\{1,1,2,3,5,\cdots,F_n\}$。设min，max分别为路径序列中最小和最大的路径编号，但对于初始路径序列$\{P_1,P_2,P_3,\cdots,P_k\}$，令$min=0$，$max=k$。

设$u=F_{n-1}+min$，$v=F_{n-2}+min$

分析路径P(u)和P(v)，其上分支谓词数分别为$r(u)$和$r(v)$，其中非线性分支谓词数分别为$t(u)$和$t(v)$，则其路径测试数据生成的代价分别为：

$R(u)=r(u)e^{t(u)}$

$R(v)=r(v)e^{t(v)}$

若$R(u)\geqslant R(v)$，则$max=u$，取路径$\{P_i\mid i=min,\cdots,v\cdots,u\}$为新的路径序列，否则，$min=v$，取路径$\{P_i\mid i=v,\cdots,u,\cdots,max\}$为新的路径序列。重复以上步骤，直至路径序列中只剩下3条路径，则其中代价最小的路径即为所求路径P_y。

如果P_y不可行，则在已计算代价的路径集中，取没有计算的代价最小的路径作为P_y。若已计算代价的路径都不可行，则删除这些路径，将所剩路径重新组合成新的路径序列，再重复以上求P_y步骤。如果遍历了G'中所有的路径，均不能求得经过分支B的数据，则分支B不可行。

3.3 测试数据的计算

为方便、快捷处理分支谓词均为线性表达式的路径，在文[11]的基础上，提出以下定理并给出证明。

定理1 若路径上各分支谓词函数均为线性表达式，那么当谓词函数关于输入变量的线性约束系统有解时，则路径可达，其解即为所求测试数据；否则，路径不可达。

证明：假设路径P中有p个输入变量和q个分支谓词，所有的分支谓词总可以转变为$F=0$，$F>0$，$F\geqslant 0$这3种形式。故可设分支谓词中有$q1$个使用“=”，有$q2$个使用“>”，有$q3$个使用“≥”，则$q=q1+q2+q3$。根据已知，我们可构建路径P中所有的分支谓词线性约束集为：

$$\begin{cases} AX+A_0=0 \\ BX+B_0>0 \\ CX+C_0\geqslant 0 \end{cases} \quad (4)$$

其中：$A=\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,p} \\ \cdots & \cdots & \cdots & \\ a_{q1,1} & a_{q1,2} & \cdots & a_{q1,p} \end{pmatrix}$，

$B=\begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,p} \\ \cdots & \cdots & \cdots & \\ b_{q2,1} & b_{q2,2} & \cdots & b_{q2,p} \end{pmatrix}$，

$C=\begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,p} \\ \cdots & \cdots & \cdots & \\ c_{q3,1} & c_{q3,2} & \cdots & c_{q3,p} \end{pmatrix}$

$X=(x_1,x_2,\cdots,x_p)^T$，$A_0=(a_{1,0},a_{2,0},\cdots,a_{q1,0})^T$，

$B_0=(b_{1,0},b_{2,0},\cdots,b_{q2,0})^T$，$C_0=(c_{1,0},c_{2,0},\cdots,c_{q3,0})^T$

∵“≥”等价于“>∪=”

∴式(4)可分解为以下两个线性约束集：

$$\begin{cases} AX+A_0=0 \\ BX+B_0>0 \\ CX+C_0>0 \end{cases} \quad (5), \qquad \begin{cases} AX+A_0=0 \\ BX+B_0>0 \\ CX+C_0=0 \end{cases} \quad (6)$$

可见式(5)，(6)中都是由“=”和“>”组成的约束集。求解约束集可知，式(5)，(6)要么有解，要么无解。当式(5)，(6)中至少有一个有解时，则存在一数据集，可遍历路径 P，即 P 可达，其解即为所求测试数据；当式(5)，(6)均无解时，则 P 不可达。故定理 1 得证。

考察所选路径 P_y 上的各分支谓词，如果各分支谓词均为线性表达式时，则执行算法 1，否则执行算法 2。

算法 1 直接使用路径上各分支谓词函数构造关于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 I。由定理 1 知，此输入变量值即为所求测试数据。如果约束系统无解，则路径不可行。

算法 2 从输入变量的值域中，任选一组输入 I_0，如 I_0 能使待测分支 B 被经过，则 I_0 为所求，算法结束。否则，考察所选路径上各分支谓词，当谓词函数是非线性表达式时，则计算其关于当前输入变量的线性算术表示，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关于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待测分支 B 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，返回到选取新的路径 P_y。

3.4 算法描述

输入：程序流程图 G，测试分支 B

输出：使路径 P 被经过的程序输入 I_f

返回：成功生成用例时 true，否则 false

```
function GENCASE(G, B)
step 1  构造包含经过 B 所有路径的程序流图 G′，求出 G′中
        所有路径集{P1,P2…Pk}，并依次放入数组 H 中；
        while H 不为空数组
step2   计算 n，生成 Fibonacci 数列，有：
          F[1]=F[2]=1,F[3]=2,F[4]=3,…,F[n]>=
          k。数列存放在数组 F 中；
           min=0, max=k, total=max−min;
        while total>2
             u=F_{n−1}+min, v=F_{n−2}+min;
          计算 R(u)和 R(v)并依次加入到数组 R 中；
          if R(u) ≥ R(v) then max= u;
          else min= v;
          endif
            total=max−min;
            设置新 n 值。方法为在数组 F 中找到 F[n]=to-
            tal;
        end while
        比较 R(min),R(min+1),R(max)，选取最小的代价
        所对应的路径作为 Py；
        对 R 从小到大排序；
        do
step3   for Py 上每个谓词结点 mj do
         if 路径 Py 上 mj 的谓词函数为非线性函数 then
           if TESTGEN(Py, I0) then
              return true;
           end if
         end if
        end for
        if TESTGEN(Py) then
           return true;
        end if
step4   设置新的 Py，该 Py 对应 R 中没有计算的最小的路
        径；
        while R 中还有对应的路径没有计算；
          从数组 H 中删除已计算代价的路径，并将剩下的
          路径组成新的序列，更新 k；
        end while
          return false;
end function

function TESTGEN(P)
step5   用 Py 中的线性谓词函数构造输入变量的线性约束系
        统；
        求解线性约束系统；
step6   if 约束系统有解 then If= I return true;
           else return false;
        endif
end function

function TESTGEN(P, I0)
step7   选取程序的初始输入 I0 以及给定的迭代次数上限 T；
        if I0 能让 Py 被经过 then If=I0
            return true;
        h=0; Done=false;
        while (not Done) and (h≤T)
           for Py 上每个谓词结点 mj do
             if 路径 Py 上 mi 的谓词函数为非线性函数 then
step8       计算 mj 的谓词函数的线性算术表示 L(mj, Ih,
            Py);
            endif
           endfor
step9   用 Py 中的线性谓词函数和 L(mj, Ih, Py)构造输入变
        量的线性约束系统；
step10  求解线性约束系统，得到新的输入 Ih+1；
```

```
        if I_{h+1}能让 P_y 被经过 then I_f = I_{h+1};
            return true;
            Done=true
          else h++ endif
        endwhile
    return false;
end function
```

4 实例和实验

4.1 应用实例

以图1程序流图G为例，求使分支B被经过的测试数据 I_f。

step1 构造包含分支B的所有路径的流图G′，如图1中实线部分所示。对G′进行分析，可知从开始节点到分支B的所有路径有 $k=10$ 条，$H=\{P_1,P_2\cdots P_{10}\}$，分别是：

$P_1=\{①,T1,T2,③,T4,T6,T9\}$

$P_2=\{①,T1,T2,③,T4,⑦,T7,T9\}$

$P_3=\{①,T1,T2,③,T4,⑦,T7,T8,T9\}$

$P_4=\{①,T1,T2,④,T4,T6,T9\}$

$P_5=\{①,T1,T2,④,T4,⑦,T7,T9\}$

$P_6=\{①,T1,T2,④,T4,⑦,T7,T8,T9\}$

$P_7=\{①,T1,②,T3,⑤,T5,⑧,T8,T9\}$

$P_8=\{①,T1,②,T3,⑤,T5,T9\}$

$P_9=\{①,T1,②,T3,⑥,T5,⑧,T8,T9\}$

$P_{10}=\{①,T1,②,T3,⑥,T5,T9\}$

step2 路径的优化选取

根据式(3)，可求得 $n=7$。故对应的Fibonacci数列为{1,1,2,3,5,8,13}。数列存放在数组F中，有：$F[1]=F[2]=1$，$F[3]=2$，$F[4]=3$，…，$F[7]=13$。

$min=0$，$max=10$，$total=max-min=10$。进入第一次循环，$v=F_{n-2}+min=5$，$u=F_{n-1}+min=8$，计算 $R(5)$ 和 $R(8)$：

$R(5)=r(5)\ e^{t(5)}=5e=13.59$

$R(8)=r(8)e^{t(8)}=5e^2=29.56$

将 $R(5)$ 和 $R(8)$ 存放在数组 R 中。

由于 $R(5)<R(8)$，故 $max=8$，$total=max-min=8$，在数组 F 中可找到 $F[5]=total=8$，进入第2次循环。直到第4次循环后，$total=2$，循环结束，此时 $min=3$，$max=5$，$R=\{R(8),R(5)\ R(3),R(6),R(4)\}$，而

$R(min)=R(3)=r(3)e^{t(3)}=6e=16.31$，

$R(min+1)=R(4)=r(4)e^{t(4)}=5e=13.59$，

$R(max)=R(5)=r(5)\ e^{t(5)}=5\ e=13.59$

可见 $R(4)=R(5)=13.59$ 为最小，故取 $P_y=P_4$。

对 R 从小到大排序得：$R=\{R(4),R(5)R(3),R(6),R(8)\}=\{13.59,13.59,16.31,16.31,29.56\}$。

step3 考察路径 P_4 可知，T1为非线性谓词，故执行step7。

step7 任选 $I_0=(X_0,Y_0,Z_0)=(1,2,3)$，迭代增量 $\Delta X=\Delta Y=\Delta Z=1$。

可知，I_0 不能使 P_4 通过，故继续执行算法后面的步骤。

因 P_4 中谓词函数T1是非线性的，故执行step8：

step8 求T1的线性算术表示：

T1的谓词函数为 $F1=X^2+Z^2-50$，因而可令其线性算术表示为

$L(BT1,I_k,P_4)=aX+cZ+d$

利用均差近似导数，可求得谓词函数 $F1=X^2+Z^2-50$，关于 I_0 的线性算术表示为

$L(BT1,I_0,P_4)=3X+7Z-64$

Step9 用 P_4 中的线性谓词函数T2，T4，T6，T9和 $L(BT1,I_0,P_4)$ 构造关于输入变量的线性约束系统

$$\begin{cases}3X+7Z-64\leqslant 0\\ X-Y<0\\ X>0\\ X<0\\ X<0\end{cases}$$

上式无解，故路径 P_4 不可行。执行step 4，设置新的 P_y。R 中没有计算的最小的路径为 $R(5)$，故取 $P_y=P_5$。再次执行step 3，step 7，step 8，step 9，用 P_5 中的线性谓词函数T2，T4，T7，T9和 $L(BT1,I_0,P_5)$ 构造关于输入变量的线性约束系统。

$$\begin{cases}3X+7Z-64>0\\ X-Y\leqslant 0\\ X\leqslant 0\\ X<0\\ 2X+Y<0\end{cases}$$

step10 求解线性约束系统。

根据文[5,12]中求解线性约束系统的方法，可求得 $X=-1$，$Y=-1$，$Z=10$。则得到新的输入 $I_1=(-1,-1,10)$。由于 I_1 能使待测分支 B 被经过，故算法结束，I_1 即为所求 I_f。

该实例已在系统配置为CPU P4 1.6G，512M DDR，Linux OS(Red Flag 4.1)的计算机上验证通过。

4.2 结果分析

对比文[9,10]，本文采用Fibonacci法来优化选取代价较小的路径，大大地减小了路径选取的盲目性。其路径的选取，使用Fibonacci法，迭代收敛较快，循环次数为 $n-3$ 次(因为最后剩下的三条路径直接比较路径的代价，不再循环)。上述实例中 $n=7$，故循环了4次。若程序有1000条路径，由式(3)知 $n=16$，则只需迭代13次，即可找出期望的路径 P_y，所以算法比较有效。与文[2]相比，虽然本文的算法亦采用均差近似导数，将非线性函数线性化，但不需计算谓词片、谓词残量，确定输入依赖集以及计算线性谓词函数的线性算术表示，因而计算量大为减少。另外，由于语句、数据流与分支一样，都是程序路径中的元素，所以本文的算法同样可用于语句覆盖、数据流覆盖的测试数据的自动生成。

结论 本文通过对分支覆盖的测试数据自动生成的几种方法进行分析，指出了各种方法的特点及存在的不足，进而在此基础上提出了一种新的测试数据自动生成的算法。该算法有如下特点：

1)通过构造新的程序流图，使用Fibonacci法优化选取路径来自动生成分支覆盖测试数据。

2)提出了路径测试数据生成代价的概念，并给出了代价的计算方法。

3)当路径上的谓词函数均为线性函数时，直接构造谓词函数关于输入变量的线性约束系统，来获得测试数据，求解约束系统时不需迭代和初始输入，其解即为所求测试数据；否则，路径不可达。

(下转第273页)

12 个完全测试序列，而采用本文中的测试覆盖标准图 1 中 b 图仅需要 4 个完全测试序列就可以满足对所有消息结点至少有一次覆盖的要求。4 个完全测试序列中，既经过结点 m2 又经过结点 m6 的路径有 3 个：p1＝($start_{m1}$，m2，$start_{m2}$，m4，$start_{m4}$，end_{m4}，end_{m2}，m6，$start_{m6}$，m2，$start_{m2}$，m4，$start_{m4}$，end_{m4}，end_{m2}，end_{m6}，end_{m1})，p2＝($start_{m1}$，m2，$start_{m2}$，m3，$start_{m3}$，end_{m3}，m4，$start_{m4}$，end_{m4}，end_{m2}，m6，$start_{m6}$，m2，$start_{m2}$，m3，$start_{m3}$，end_{m3}，m4，$strat_{m4}$，end_{m4}，end_{m2}，end_{m6}，end_{m1})，p3＝($start_{m1}$，m2，$start_{m2}$，m3，$start_{m3}$，end_{m3}，m4，$start_{m4}$，end_{m4}，m5，$start_{m5}$，end_{m5}，end_{m2}，m6，$start_{m6}$，m2，$start_{m2}$，m3，$start_{m3}$，end_{m3}，m4，$start_{m4}$，end_{m4}，m5，$start_{m5}$，end_{m5}，end_{m2}，end_{m6}，end_{m1})。只经过 m2 结点的路径有一个：p4＝($start_{m1}$，m2，$start_{m2}$，m3，$start_{m3}$，end_{m3}，m4，$start_{m4}$，end_{m4}，m5，$start_{m5}$，end_{m5}，end_{m2}，end_{m1})。

结论 本文描述了一个基于 UML 顺序图生成测试用例的方法，该方法通过覆盖与 UML 顺序图等价的 IRCFG 图而生成测试序列。该方法与传统的覆盖标准相比，能够成倍降低产生测试用例的数量。把本文提出的测试方法应用到实际的测试工作中，尤其是在测试大型复杂系统时，不仅有效解决对系统消息事件交互的理解，提高测试效率，而且有效地减少了测试数据的数据量，降低了软件测试所需要的资源和成本。

下一步的工作是研究结合 UML 状态图与顺序图为包含更多信息的消息顺序图，以及基于该顺序图的软件测试方法。在 UML 图中往往只包含引起状态转移的消息事件，而顺序图则包含许多不会引起状态转移的消息事件。期望由 UML 状态图中状态转移产生的消息事件序列与顺序图消息事件时间顺序性相结合，构成包含更多信息的顺序图，使测试达到更全面。

参 考 文 献

1 颜炯，王戟，陈火旺. 基于模型的软件测试综述[J]. 计算机科学，2004，31(2)：184～187

2 Stefania G, Diego L, Mieke M. Formal Test-case Generation for UML Statecharts[C]. In: ICECCS, Florence, 2004

3 Kansomkeat S, Rivepiboon W. Automated-Generating Test Case Using UML Statechart Diagrams[C]. Proceedings of SAICSIT, Pretoria, 2003

4 Pender T, 著. UML 宝典[M]. 耿国桐，等译. 北京：电子工业出版社，2004

5 Johannes R, Martin G. A Practical Approach to Validating and Testing Software Systems Using Scenarios[C]. In: Third Int'l Software Quality Week Europe Conf., Brussels, 1999

6 Fraikin F, Leonhardt T. SeDiTeC-Testing Based on Scquence Diagrams[C]. In: the 17th IEEE Intl. Confon Automated Software Engneering, Edinburgh, 2002

7 Bertolino A, Marchetti E. Introducing a Reasonably Complete and Coherent Approach for Model-based Testing [J]. Theoretical Computer Science, 2005, 116(1): 85～97

8 Binder R. Testing Object-oriented Systems: Models, Patterns, and Tools[M]. Gary McGraw, Boston: Addison -Wesley Longman Publishing Co, 1999

9 Jorgenson P, Erickson C. Object-oriented integration testing[J]. Communications of the ACM, 1994, 37(9): 30～38

10 Rountev A, Kagan S, Sawin J. Coverage Criteria for Testing of Object Interactions in Sequence Diagrams[C]. In: Fundamental Approaches to Software Engineering, Edinburgh, 2005

11 Milanova A, Rountev A, Ryder B G. Parameterized object sensitivity for points-to analysis for Java[J]. ACM Transactions on Software Engineering and Methodology, 2004, 14(1): 1～41

12 Bodik R, Gupta R, Soffa M L. Refining data flow information using infeasible paths[J]. ACM SIGSOFT Software Engineering Notes, 1997, 22(6): 361～377

(上接第 264 页)

4)该算法能处理带非线性表达式的谓词分支，以及不同语言的程序。

参 考 文 献

1 Jiang Tai-Ying, Liu Chien-Nan Jimmy, Jou Jing-Yang. An observability measure to enhance statement coverage metric for proper evaluation of verification completeness[A]. In: Proceedings of the ASP- Design Automation Conference, 2005[C]. Asia and South Pacific, 2005. 323～326

2 Gupta N, Mathur A P, Soffa M L. Generating test data for branch coverage [A]. In: Automated Software Engineering, 2000. Proceedings ASE 2000[C]. The Fifteenth IEEE International Conference on 11-15 Sept. 2000. 219～227

3 Sze S K S, Lyu M R. ATACOBOL-a COBOL test coverage analysis tool and its applications[A]. In: Proceedings 11th International Symposium of Software Reliability Engineering [C], 2000. 327～335

4 SHAN Jin-Hui, WANG Ji, QI Zhi-Chang, et al. Improved Method to Generate Path-Wised Test Data[J]. Journal of Computer Science and Technology, 2003, 18(2): 235～240

5 陈继锋，朱利，沈钧毅，等. 一种基于路径的测试数据自动生成算法[J]. 控制与决策，2005，20(9)：1065～1068

6 Korel B. A Dynamic Approach of Test Data Generation[A]. In: Conference on Software Maintenance[C]. San Diego, 1990. 311～317

7 Ferguson R, Korel B. The Chaining Approach for Software Test Data Generation[J]. A CM Transactions on Software Engineering Methodology, 1996, 15(1): 63～86

8 Forgacs I, Hajnal A. An Applicable Test Data Generation Algorithm for Domain Errors[A]. In: Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis [C]. Clearwater Beach, Florida, March 1998

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

10 Gupta N, Mathur A P, Soffa M L. Automated Test Data Generation Using An Iterative Relaxation Method[A]. Proceedings of the 6th ACM SIGSOFT International Symposium on Foundations of Software Engineering[C]. Florida, United States, 1998. 231～244

11 卢开澄，卢华明. 组合数学. 第 3 版[M]. 北京：清华大学出版社，2002. 115～126

12 Edvardsson J, Kamkar M. Analysis of the constraint solver in UNA based test data generation[J]. ACM SIGSOFT Software Engineering Notes, 2001, 26(5): 237～245