蜕变关系构造基本准则与策略研究

王 瑢 贲可荣
(海军工程大学计算机工程系 武汉 430033)

摘 要 蜕变测试可以部分解决软件测试中的 Oracle 问题,其关键步骤和难点是蜕变关系的构造,它将直接影响测试的效果。通过对典型程序测试的案例对蜕变关系的构造进行分析,归纳总结了若干构造蜕变关系的基本准则,并在案例研究中采用变异分析方法验证了构造准则的合理性。提出了蜕变测试与等价类测试结合运用的测试方法,此方法可用于输入空间易于分类的程序。

关键词 软件测试,Oracle 问题,蜕变测试,蜕变关系
中图法分类号 TP311 **文献标识码** A

Researches on Basic Criterion and Strategy of Constructing Metamorphic Relations

WANG Rong BEN Ke-rong
(Department of Computer Engineering, Naval University of Engineering, Wuhan 430033, China)

Abstract Metamorphic testing can alleviate the oracle problem in software testing, and the construction of metamorphic relations in this method is a difficult problem, which will affect the result of testing directly. This paper analysed the construction of effective metamorphic relations based on typical case studies, summarizeed several criterions of selecting useful relations, and the rationality of our criterions was verified through case studies with mutation analysis. The paper also proposed a testing strategy of combining metamorphic testing and equivalence testing. It can be used in the testing of programs whose input spaces are easy to be classified.

Keywords Software testing, Oracle problem, Metamorphic testing, Metamorphic relation

1 引言

软件测试是为发现错误而执行程序的过程。传统的软件测试技术多是通过比较测试用例的执行结果和预期输出是否相同,来判断程序的正确性。但是在很多情况下,测试人员不能或者很难获得程序的预期输出,因而无法将执行结果和预期结果进行比较,这就是软件测试中的 Oracle 问题。为解决这一问题,Chen 等人提出了蜕变测试方法[1],即通过比较程序的多个执行结果间的关系来检验程序的正确性,因而不需要构造程序的预期输出。

成功的测试用例是指在测试中通过的用例。一个成功的用例往往被认为是对测试没有价值的用例,因为它没有发现任何错误。因此,在传统的测试方法中,成功的测试用例在后续测试中通常被抛弃或者极少使用。而 Chen 则认为,所有的测试用例都是有其利用价值的[2],测试中已使用的成功的测试用例包含着丰富的信息,不应该随便抛弃。蜕变测试方法就是充分挖掘了成功测试用例所包含的信息并加以利用。

Chen 指出,运用蜕变测试技术必须意识到两个问题:(1)蜕变测试只是一种生成衍生测试用例的方法。换句话说,单纯的蜕变测试对软件测试来说是不够的。它还必须与其它的测试用例选择策略结合使用。(2)已有的研究结果表明,对不同类型的错误来说,不同的蜕变关系有着不同的错误检测能力。如何选择蜕变关系并使它们以一种协作和互补的方式达到最佳的测试效果,是蜕变测试中的一个重要问题[2]。

在蜕变关系的选择策略以及评价方法上,Chen 认为,测试功能较强的蜕变关系应当包括核心功能的执行及对该功能的有效验证,它应当对错误具有很高的敏感性。当然,蜕变关系对错误的敏感性主要是由错误的特性及错误与关系的关联方式决定的[4]。

Mayer 以一个实例为依据,根据对实验数据的剖析归纳出了 4 条可以用来评价蜕变关系性能的一般性准则:(1)等式两边都只有一个待测程序的输出的蜕变关系,检错能力最差;(2)线性组合形式,即等式中至少有一边有多个待测程序输出的运算的蜕变关系,检错能力较强;(3)含有待测程序的语义信息越丰富的关系,其测试效果越好;(4)与典型的程序或算法使用的策略相似的关系,其适用范围比较有限[5]。

本文在总结前人工作的基础上,对两个程序进行蜕变测试的实例分析,归纳了构造蜕变关系的基本准则,并采用变异分析方法验证构造准则的合理性。本文第 2 节介绍了蜕变测试的基本原理;第 3 节给出了两个运用蜕变测试的案例,并分析了实验结果;第 4 节列出了通过实验分析得出的蜕变关系构造准则以及一个在某些程序的蜕变测试中可用到的一种可

到稿日期:2011-02-14 返修日期:2011-07-15
王 瑢(1987—),女,硕士生,主要研究方向为软件质量保证技术,E-mail:trista_cool@126.com;贲可荣(1963—),教授,博士生导师,主要研究方向为软件工程和人工智能等。

选蜕变测试构造方法；最后对本文进行了总结，并对进一步的发展方向进行了预测。

2 蜕变测试

蜕变测试是一种利用未发现错误的成功测试用例来生成衍生测试用例的技术[1]，适用于各种输入的软件测试[5]，具有普遍的适用范围。下面将介绍它的基本原理。

假设程序 P 是计算函数 f 的被测程序，$x_1,x_2,\cdots,x_n(n>1)$是 f 的 n 个自变量，$f(x_1),f(x_2),\cdots,f(x_n)$是它们所对应的函数结果。当自变量 $x_1,x_2,\cdots,x_n$ 满足关系 R 时，$f(x_1),f(x_2),\cdots,f(x_n)$满足关系 R_f，即

$$R(x_1,x_2,\cdots,x_n)=>R_f(f(x_1),f(x_2),\cdots,f(x_n))$$

则称(R,R_f)是程序 P 的蜕变关系[3]。

如果 P 是正确的，那么它满足下面的推导式：

$$R(I_1,I_2,\cdots,I_n)=>R_f(P(I_1),P(I_2),\cdots,P(I_n))$$

式中，$I_1,I_2,\cdots,I_n$ 是程序 P 对应于 $x_1,x_2,\cdots,x_n$ 的输入，$P(I_1),P(I_2),\cdots,P(I_n)$是相应的输出。因此，可以通过检测上式是否成立来判定程序 P 的正确性。蜕变测试即是一种基于蜕变关系的测试[3]。

使用蜕变关系(R,R_f)测试程序 P 时，起初给定(利用其他测试用例选择策略生成)的测试用例是原始测试用例；由原始测试用例根据蜕变关系 R 计算出的用例，是该原始用例基于蜕变关系(R,R_f)的衍生测试用例[3]。

下面给出一个蜕变测试的例子。测试计算正弦函数的程序时，选择 $\sin x=\sin(180-x)$为它的一条蜕变关系。通过其它策略选择 $t=46.8$ 为它的一条原始测试用例。假设程序输出的结果为 $P(t)=0.7213$，此时很难确定 sin46.8 的值，因此无法判断这一结果的正确性。然而，无论这一结果正确与否，都可以根据这条原始测试用例和蜕变关系 $\sin x=\sin(180-x)$为它构造一条衍生测试用例 $t'=180-46.8=135.2$，再用程序计算出 $P(t')$的输出结果。假设计算结果为 $P(t')=0.7346$，则两次的计算结果不满足 $P(t)=P(t')$，此时可以确定程序中存在错误。 有效缓解

蜕变测试可以有效地解决 Oracle 问题和充分利用成功的测试用例，是一种实用的软件测试方法。实验表明，蜕变测试具有合理的成本效益，并且在检错方面有着比传统测试方法更大的潜力[7]。文献[8]中作者用控制实验的方法研究了使用蜕变测试的成本及效率，实验结果表明，与断言测试相比，蜕变测试的错误检测率更高而效率有待提高，可适用于较为粗粒度的测试需求。

3 实例研究

本节将通过两个案例就蜕变关系构造原则问题展开讨论。案例一 TriSquare 是计算三角形面积的程序。案例二 MatMul 是计算两个矩阵乘积的程序，在对这两个程序进行测试的过程中，均存在着预期输出难以获取的问题，即 Oracle 问题。

3.1 案例一：计算三角形面积

图 1 给出了计算三角形面积程序的代码。这个程序首先判断 a,b,c 3 个大于 0 的实数能否构成一个三角形。如果可以，则输出所构成三角形的面积。

```
1 double TriangleSquare (float a,float b,float c) {
2 int match=0;
3 if (a==b)
4 match=match+1;
5 if (a==c)
6 match=match+2;
7 if (b==c)
8 match=match+3;
9 if (match==0) /*if a,b and c are not equals to each other*/
10   if (a+b <=c) {
11   System.out.println(" Not a triangle");
12   return 0.0;
13   } else if (b+c <=a) {
14   System.out.println (" Not a triangle");
15   return 0.0;
16   } else if (a+c <=b) {
17   System.out.println (" Not a triangle");
18   return 0.0;
19   } else {
20   double p=(a+b+c)/2.0;
21   System.out.println (" Scalene");
22   return sqrt (p*(p-a)*(p-b)*(p-c));/* compute square */
23   }
24   else if (match==1) /* if (a=b≠c) */
25   if (a+b <=c) {
26   System.out.println (" Not a triangle");
27   return 0.0;
28   } else {
29   double h=sqrt (pow (a,2)-pow(c/2.0,2));
30   System.out.println (" Isosceles");
31   return (c*h)/2.0;/* compute square */
32   }
33   else if (match==2) /* if (a=c≠b) */
34   if (a+c <=b) {
35   System.out.println (" Not a triangle");
36   return 0.0;
37   } else {
38   double h=sqrt (pow (a,2)-pow (b/2.0,2));
39   System.out.println (" Isosceles");
40   return (b*h)/2.0;/* compute square */
41   }
42   else if (match==3) /* if (b=c≠a) */
43   if (b+c <=a) {
44   System.out.println (" Not a triangle.");
45   return 0.0;
46   } else {
47   double h=sqrt (pow (b,2)-pow (a/2.0,2));
48   System.out.println (" Isosceles");
49   return (a*h)/2.0;/* compute square */
50   }
51   else { /* if (a=b=c) */
52   System.out.println (" Equilateral");
53   return (sqrt (3.0)*a*a)/4.0;/* compute square */
54   }}
```

图 1 三角形面积计算代码

董国伟等根据三角形的基本性质为此程序构造了 7 条蜕变关系 MR_1-MR_7[6]。在此基础上进行扩展，构造了如表 1 所列的 10 条蜕变关系，以便于对蜕变关系构造的基本准则进行分析和归纳总结。其中 MR_5-MR_7 的构造原理如图 2(a)

所示[6]，MR_8－MR_{10}的构造原理如图 2(b)所示。

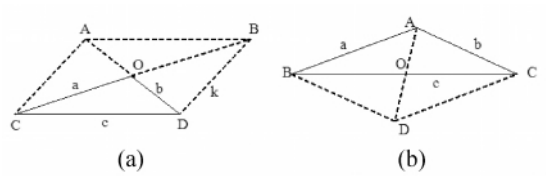

图 2　蜕变关系 MR_5，MR_8 构造原理

表 1　TriSquare 蜕变关系

MR_i	R	R_f
MR_1	$(a',b',c')=(b,a,c)$	TriSquare(a',b',c')= TriSquare(b,a,c)
MR_2	$(a',b',c')=(c,b,a)$	
MR_3	$(a',b',c')=(a,c,b)$	
MR_4	$(a',b',c')=(ka,kb,kc)$	TriSquare(a',b',c')= k2×TriSquare(b,a,c)
MR_5	$(a',b',c')=(a,b,\sqrt{2a^2+2b^2-c^2})$	TriSquare(a',b',c')= TriSquare(b,a,c)
MR_6	$(a',b',c')=(a,\sqrt{2a^2+2c^2-b^2},c)$	
MR_7	$(a',b',c')=(\sqrt{2b^2+2c^2-a^2},b,c)$	
MR_8	$(a',b',c')=(a,\sqrt{2a^2+2b^2-c^2}/2,c/2)$	TriSquare(a',b',c')= TriSquare$(b,a,c)/2$
MR_9	$(a',b',c')=(\sqrt{2a^2+2c^2-b^2},b/2,c)$	
MR_{10}	$(a',b',c')=(a/2,b,\sqrt{2b^2+2c^2-a^2}/2)$	

备注：1)蜕变关系的定义域都是$\{(a,b,c)|(a+b>c)\wedge(b+c>a)\wedge(a+c>b)\}$，即 a,b,c 可以构成三角形；2)k 为任意正数。

MR_5 构造原理：如图 2(a)所示，设三角形 OCD 的 3 边分别为 a,b,c，则 $OC=OB=a$，令 $BD=k$。根据三角形性质，$\triangle ODC$ 和 $\triangle ODB$ 面积相等，即 $TriSquare(a,b,c)=TriSquare(a,b,k)$。又根据平行四边形对角线的平方和等于 4 条边的平方和的性质，得 $k^2=2a^2+2b^2-c^2$，从而构造出 MR_5。同理可以构造出 MR_6 和 MR_7[6]。

MR_8 构造原理：如图 2(b)所示，假设三角形 ABC 的 3 边分别为 a,b,c，则 $AB=CD=a$，$AC=BD=b$，O 为 BC 边的中点，$\triangle OAB$ 的面积是 $\triangle ABC$ 面积的一半，OA 的长度可根据平行四边形 4 条边的平方和等于其对角线的平方和的性质求得 $OA=\sqrt{2a^2+2b^2-c^2}/2$，因此可得 MR_8。同理可构造蜕变关系 MR_9 和 MR_{10}。

变异分析是一种评估测试用例集质量或确定其充分性的有效技术[9]。在对蜕变测试技术的研究中，这种技术可用来评估各条蜕变关系的检错能力。董国伟等在程序 TriSquare 中，基于算术符号替代(AOR)和数据语句变更(DSA)[9]等几种基本的变异操作分别置入 4 条变异，以使得每条路径至少出现一次错误，如图 1 中被标记的代码所示。这些变异都是编写 TriSquare 时容易发生的，等边、等腰和不等边三角形均有涉及，详细介绍如下[6]。

变异 1　将第 4 行代码和第 6 行代码交换位置；

变异 2　将第 20 行代码替换为 $p=(a+b+c)*2.0$；

变异 3　将第 31，40，49 行代码中的“/2.0”同时替换为 *2.0；

变异 4　将第 53 行代码替换为 return sqrt(3.0)*a*a/2.0。

此案例原始测试用例的选择采用特殊值和随机值相结合的方法，并引入等价类划分，将输入划分为等腰、等边、不等边和不能构成三角形 4 类，其中等腰三角形和不等边三角形还可再次划分为直角三角形和非直角三角形。其中直角三角形的面积可以很容易获得，故在此次测试中只选用较少的测试用例。而蜕变关系的定义域为 a,b,c，可以构成三角形，故在输入的 3 边不能构成三角形时不需用到蜕变测试。我们选择了 25 个原始测试用例，其中 5 个不能构成三角形用例、3 个等边三角形用例、7 个等腰三角形用例和 10 个不等边三角形用例，以此保证测试覆盖了各个类型的输入。分别在含有变异 $i(i=1,2,3,4)$的程序中测试 5 个不能构成三角形的用例，然后用后 3 类总共 20 个用例使用蜕变测试方法，验证它们是否满足蜕变关系 MR1－MR10。

在实际操作中需要注意的是，由于计算精度的影响，使得计算结果有可能不能完全满足蜕变关系。尤其是对于 MR5－MR10这类蜕变关系，计算包含平方和开根号的运算，等式两边的值很难完全一致。此时在判断输出结果是否满足蜕变关系时需放松条件。若其计算结果在一定精度范围内，则认为该用例满足蜕变关系。

在变异分析方法中，一般使用 MS(Mutation Score)来评估一个测试用例集[10]。此处 MS 定义为一个测试用例集可检测出的非等价变异的比例，即

$$MS=\frac{M_k}{M_t-M_q}$$

式中，M_k 为测试用例集检测出的变异体的数量，M_t 为变异体总数，M_q 为等价变异体数量。此处的 4 个变异均为非等价变异，故此案例中 $M_S=M_k/M_t$。

由表 1 可看出，MR1－MR3，MR5－MR7，MR8－MR10 都是对称蜕变关系。在分析试验结果时将蜕变关系分成 3 组，分别为 MR1－MR4，MR5－MR7，MR8－MR10。这 3 组用例的 MS 值如表 2 所列。

表 2　TriSquare 3 组蜕变关系生成的测试用例集 MS 值分布情况

蜕变关系	MS				
	0%	25%	50%	75%	100%
MR_1－MR_4	12	8	0	0	0
MR_5－MR_7	0	10	4	6	0
MR_8－MR_{10}	0	3	7	10	0

由表 2 可以明显看出，MR_1－MR_4 这组检出变异的能力尤其差，它们只能检测出一个变异，其中 MR_4 通过了所有的测试。MR_5－MR_7 和 MR_5－MR_7 这两组检出变异的能力较强，结果表明它们任意一组蜕变关系在各个等价类中各选择一个原始用例就可以检测出所有的变异。而且由表 2 也可以看出，蜕变关系 MR_8－MR_{10} 的检错能力要强于 MR_5－MR_7，其 MS 值为 75%的用例数较多。这两组用例的生成方法都是基于平行四边形的性质，而且检出变异的类型也很接近。然而通过表 1 比较这两组蜕变关系的特点可以发现，后一组的 3 个元素在生成新的测试用例时有两个元素被改变，而前一组只有一个元素改变。由此推断出，存在多元输入的情况下，改变元素数量较多的蜕变关系具有更强的检错能力。

3.2　案例二：矩阵的乘法运算

矩阵乘法运算函数的代码如图 3 所示。

根据矩阵的性质，吴鹏等在稀疏矩阵的乘法运算中构造了 9 条蜕变关系 MR_1－MR_9[10]。为了对蜕变关系构造的基本准则进行分析和归纳总结，在此基础上定义了如表 3 所列的 11 条蜕变关系。

基于基本的变异操作，在程序 MatMul 中分别植入 5 条变异，如图 3 中被标记的代码所示。变异详细介绍如下：

变异 1　将第 6 行代码替换为 for(i=0;i<m;i++)；

变异 2　将第 6 行代码替换为 for(i=0;i<m;i++)，将第 8 行代码替换为 for(j=0;j<n;j++)；

变异 3　将第 11 行代码替换为

c[i*m+j]+=a[i*ja+t];

变异 4　将第 11 行代码替换为

c[i*m+j]+=b[t*jb+j];

变异 5　将第 11 行代码替换为

c[i*m+j]+=a[i*ja+t]+b[t*jb+i]。

```
1 void MatMul (int n,int m,const double *a,const int ia,const int ja,
              const double *b,const int ib,const int jb,double *c)
2 {
3   int i,j,t;
4   if(ja==ib)
5   {
6   for ( i=0;i<n;i++ )
7   {
8     for ( j=0;j<m;j++ )
9     {
10      for ( t=0;t<ja;t++ )
11        c[i*m+j] +=a[i*ja+t];
12      }
13    }
14  }
15 }
```

图 3　矩阵乘法运算代码

表 3　MatMul 蜕变关系

MR_i	R	R_f	备注
MR_1	$A'=BT,B'=AT$	$A'B'=(AB)T$	矩阵 A,B 转置后仍可进行乘法运算
MR_2	$A'=PA,B'=B$	$A'B'=P(AB)$	P 为将单位矩阵 I 任意两行互换后得到的矩阵，即 A′为 A 任意两行互换后得到的矩阵
MR_3	$A'=A,B'=BP$	$A'B'=(AB)P$	
MR_4	$A'=QA,B'=B$	$A'B'=Q(AB)$	Q 为将单位矩阵 I 主对角线元素乘以数 k 后得到的矩阵
MR_5	$A'=A,B'=BQ$	$A'B'=(AB)Q$	
MR_6	$A'=kA,B'=B$	$A'B'=k(AB)$	k 为任意数
MR_7	$A'=A,B'=kB$		
MR_8	$A'=A+I,B'=B$	$A'B'=AB+IB$	I 为单位矩阵，A,B 为与 I 等阶的方阵
MR_9	$A'=A,B'=B+I$	$A'B'=AB+AI$	
MR_{10}	$A'=A,B'=A-1$	$A'B'=I$	I 为单位矩阵，A,B 为可逆矩阵
MR_{11}	$A'=B-1,B'=A-1$	$A'B'=(AB)-1$	

根据矩阵乘法的特点，将其输入的可能性分为 3 类：$m\times n$ 矩阵与 $n\times m$ 矩阵($m\neq n$)的乘法；$m\times k$ 矩阵与 $k\times n$ 矩阵($m\neq n$)的乘法；两个 n 阶矩阵的乘法。同样，采取特殊值和随机值结合的方式，基于这个分类选择了 8 个原始测试用例，包含了 3 种不同的类型。然后根据蜕变关系 MR_1-MR_{11} 生成相应的衍生测试用例，执行衍生测试用例并判断其与原始测试用例的执行结果是否满足蜕变关系。

根据所定义的蜕变关系的不同特点，表 4 对 MR_1-MR_5，MR_6-MR_7，MR_8-MR_9 和 $MR_{10}-MR_{11}$ 这几组蜕变关系的 MS 值进行了比较。

表 4　MatMul 4 组蜕变关系生成的测试用例集 MS 值分布情况

蜕变关系	MS					
	100%	0%	20%	40%	60%	80%
MR_1-MR_5	0	0	0	4	0	4
MR_6-MR_7	0	0	1	5	1	1
MR_8-MR_9	5	0	0	3	0	0
$MR_{10}-MR_{11}$	6	0	0	2	0	0

通过观察表 4 可以发现，第一组即 MR_1-MR_5 的 MS 值分布普遍较高，其检错能力最强；而通过进一步分析实验结果还发现，当将后 3 组蜕变关系依次加入第一组蜕变关系中后，即分别生成 MR_1-MR_7，MR_1-MR_9，MR_1-MR_{11} 3 组蜕变关系，其 MS 值的分布与原第一组一模一样，即后加入的蜕变关系对测试结果无影响。

对比分析后 3 组蜕变关系的特点可以发现以下 3 点：(1) MR_1-MR_5 和 MR_6-MR_7 两组蜕变关系的原理比较类似，在程序正确的情况下都是将矩阵的元素扩大了 k 倍，只是 MR_1-MR_5 是通过矩阵的乘法将其扩大。这样在执行用例的过程中，等式两边都需要执行两次被测程序，而蜕变关系 MR_6-MR_7 只需要执行一次被测程序。实验结果表明，由同一个原始用例通过蜕变关系 MR_6-MR_7 发现的变异 MR_4-MR_5 都能发现，反之则不成立。这一结果验证了 Mayer 提出的评价蜕变关系性能的准则，即等式中至少有一边有多个待测程序输出的运算的蜕变关系检错能力较强[5]。(2) MR_8-MR_{11} 这 4 条蜕变关系只能用于方阵，尤其是 $MR_{10}-MR_{11}$ 只能用于可逆矩阵，适用范围过窄，因此它们的 MS 值普遍偏低。(3) 在 $MR_{10}-MR_{11}$ 这组蜕变关系中，需要计算矩阵的逆矩阵。而计算 n 阶矩阵逆矩阵的时间复杂度与计算矩阵乘法的时间复杂度都为 $O(n^3)$，在测试资源普遍比较紧张的情况下，构造这样的蜕变关系是不明智的。

4　蜕变关系的构造

通过对以上两个案例的分析可以看出，蜕变关系对蜕变测试的结果有很大的影响，蜕变关系的构造可以说是蜕变测试方法的灵魂所在。根据上述两个实验以及对实验结果的分析，得出了 5 个构造蜕变关系的基本准则，以及一个在某些程序的蜕变测试中可用到的一种可选蜕变关系构造方法。

4.1　蜕变关系构造准则

测试时构造蜕变关系的基本准则如下。

准则一　正确性准则。

这是构造蜕变关系时最基本的原则。进行蜕变测试的第一步就是要全面分析程序所包含的性质，或者程序的规约，这些都是构造蜕变关系的基石。然而需要注意的是，测试人员所了解的性质或者规约都必须相当可靠，每个程序都有其独特性，切忌犯经验主义错误。例如，在利用蜕变测试方法测试计算三角形面积的程序时，可能会构造出这样的蜕变关系 $R\{a'\geqslant a,b'\geqslant b,c'\geqslant c\}=>R_f\{\mathrm{TriSquare}(a',b',c')\geqslant\mathrm{TriSquare}(a,b,c)\}$；在测试稀疏矩阵乘法的程序时，也可能会构造蜕变关系 $R\{A'=Ak,B'=Bk\}=>R_f\{A'B'=(AB)k\}$。明显这两条蜕变关系都是错误的。万一在测试过程中加入了这样的蜕变关系，将造成时间和资源的极大浪费。因此，为保证所构造蜕变关系的正确性，测试人员必须对程序和相关学科相当了解，必要时需与软件开发者或者领域专家沟通把关。

准则二　衍生测试用例易于生成原则。

测试人员在构造蜕变关系时，要考虑到衍生测试用例生成算法的时间复杂度。例如在计算矩阵乘法的案例中，蜕变关系 MR_{10} 和 MR_{11} 在生成衍生测试用例时需要计算矩阵 A，B 的逆矩阵，而计算逆矩阵比矩阵乘法的运算更复杂。这样的衍生测试用例生成方法运用在实践中得不偿失。

准则三　优先选取变化元素多或表达式复杂的蜕变关系。

通过对案例一中表 1 和表 2 的分析可以看出，表达式复杂的蜕变关系 MR_5 — MR_{10} 的错误检测能力明显强于表达式简单的蜕变关系 MR_1 — MR_4。同时，通过对 MR_5 — MR_7 和 MR_8 — MR_{10} 两组蜕变关系的比较可以看出，存在多元输入的情况下，改变元素数量较多的蜕变关系具有更强的检错能力。

准则四　优先选择可产生唯一衍生测试用例的蜕变关系。

在案例一中构造蜕变关系 MR_4：$R\{(a',b',c')=(ka,kb,kc)\}=>R_f\{\text{TriSquare}(a',b',c')=k2\times\text{TriSquare}(b,a,c)\}$（$k$ 为任意正数），此时 k 的值不确定；案例二中构造的蜕变关系 MR_4 — MR_7 中 k 的值也不确定，这样由它们构造的衍生用例就存在着无限种可能。而无论 k 取何值，对蜕变关系的影响是相同的，故此处不妨将 k 赋值为 2，例如可构造 MR_4 为 $R\{(a',b',c')=(2a,2b,2c)\}=>R_f\{\text{TriSquare}(a',b',c')=4\times\text{TriSquare}(b,a,c)\}$，这样既简化了后续计算，又有利于蜕变测试方法的自动化。

准则五　选择适用范围大的蜕变关系，蜕变关系集应覆盖尽可能大的范围。

蜕变测试方法针对存在 Oracle 问题的程序，而这样的程序可能在它的整个输入范围内都存在着 Oracle 问题。因此在构造蜕变关系时，必须充分发掘程序的特性，构造覆盖范围尽可能大的蜕变关系集合。同时，在构造蜕变关系时，还应兼顾到所选关系的适用范围。例如案例二中 MR_8 和 MR_9 的适用范围只有 n 阶方阵，而 MR_{10} 和 MR_{11} 的适用范围就只有可逆矩阵。本程序针对的对象是所有矩阵的乘法，在选择原始测试用例时可逆矩阵的比例就不大，导致这类蜕变关系在测试中的实用价值很低。

4.2　蜕变测试与等价类测试的结合应用

在案例一的实验过程中，将程序的输入空间划分为了 4 个等价类：等腰三角形、等边三角形、不等边三角形和不能构成三角形的输入。除了不能构成三角形的输入，案例一中构造的蜕变关系全部应用于剩下的 3 种类型中。然而在实验过程中发现，当输入为等边三角形时，由蜕变关系 MR_1 — MR_3 生成的衍生测试用例与原始测试用例的输入是一模一样的，蜕变测试中这样的用例是没有实际意义的。由蜕变关系 MR_5 — MR_7 利用等边三角形的原始用例生成的衍生测试用例时，将生成 3 个边长相等的等腰三角形，虽然 3 边输入顺序不同，但这个类型输入的测试已经在以等腰三角形为原始用例，由蜕变关系 MR_1 — MR_3 生成的衍生测试用例测试过了，这样生成的测试用例集将存在大量的冗余。

因此，在对此类程序进行蜕变测试时，可先将程序的输入空间进行等价类划分，然后针对各个等价类的特点，并遵循上述基本准则，构造专用或者通用的蜕变关系。

例如在对案例一构造蜕变关系时，对等边三角形的输入只使用蜕变关系 MR_5 和 MR_8，对等腰三角形和不等边三角形则使用蜕变关系 MR_1 — MR_{10}。这样构造的总测试用例数将由原来的 220 个减少为 179 个，而所得的测试结果与原用例集的结果相同。

结束语　蜕变测试方法可以有效地利用成功的测试用例应对测试中的 Oracle 问题。其过程简单易懂且易于实现自动化，有着合理的成本效益，对软件测试人员来说是一种实用且有效的测试技术。然而对大多数存在 Oracle 问题的程序来说，测试人员都可以构造若干不同的蜕变关系，它们在蜕变测试中表现出的检错能力可能会相差很大。

本文在对使用蜕变测试的案例进行实验和分析的基础上，总结出了 5 条构造蜕变关系的基本准则。这是一些通用的基本准则，适用于存在 Oracle 问题的软件测试。本文还提出了蜕变测试与等价类测试结合运用的测试方法，此方法适用于使用等价类进行测试并且不同等价类之间性质相差较大的程序。

已有的针对蜕变关系选取的研究，主要采用实例分析的方法，即如本文一样针对某些特定的待测程序，构造出一系列的蜕变关系，然后通过分析比对它们的测试效果和结构特点，总结出有效蜕变关系的选择策略。然而，此类研究只给出了蜕变关系选取的一般性指导方针，且大多以数值型程序为研究的对象，并没有讨论如何构造实用的蜕变关系，即没有提供具体的构造方法，这是在蜕变关系的构造技术方面下一步研究的主要方向。

参 考 文 献

[1] Chen T Y, Cheung S C, Yiu S M. Metamorphic testing: A new approach for generating next test cases[R]. HKUST-CS98-01. Hong Kong, 1998

[2] Chen T Y, Huang D H, Tse T H, et al. Case studies on the selection of useful relations in metamorphic testing [C]// Proceeding of the 4th Ibero -American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004). Polytechnic University of Madrid, Madrid Spain, 2004: 569-583

[3] 董国伟，徐宝文，陈林，等. 蜕变测试技术综述[J]. 计算机科学与探索，2009，3(2)：130-143

[4] Chen T Y, Kuo F C, Liu Y, et al. Metamorphic testing and testing with special values[C]// Proceeding of the 5th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2004). 2004: 128-134

[5] Mayer J, Guderlei R. An empirical study on the selection of good metamorphic relations[C]// Proceeding of the 30th Annual International Computer Software and Applications Conference (COMPSAC2006). 2006: 475-484

[6] 董国伟，聂长海，徐宝文. 基于程序路径分析的有效蜕变测试[J]. 计算机学报，2009，32(5)：1002-1013

[7] Hu P F, Zhang Z Y, Chan W K, et al. An empirical comparison between direct and indirect test result checking approaches[C]// Proceeding of the 3rd International Workshop on Software Quality Assurance (SOQUA 2006), in Conjunction with the 14th ACM SIGSOFT Symposium on Foundations of Software Engineering. New York: ACM Press, 2006: 6-13

[8] Experimental Study to Compare the Use of Metamorphic Testing and Assertion Checking[J]. Journal of Software, 2009, 20(10): 2673-2654

[9] Offutt A J, Lee A, Rot hermel G, et al. An experimental determination of sufficient mutant operators[J]. ACM Transactions on Software Engineering and Methodology, 1996, 5 (2): 99-118

[10] Wu P, Shi X C, Tang J J, et al. Metamorphic testing and special case testing: A case study[J]. Journal of Software, 2005, 16(7): 1210-1220