第33卷 第23期 计 算 机 工 程 2007年12月
Vol.33 No.23 Computer Engineering December 2007
·软件技术与数据库· 文章编号：1000—3428(2007)23—0057—03 文献标识码：A 中图分类号：TP311.5

基于区间算法的软件测试数据生成方法

高 月，邵培南，邢洁雯
（华东计算技术研究所，上海 200233）

摘 要：软件测试过程中经常需要对大量的数值计算模块进行穷举测试，传统的软件测试方法就显露出了不足与局限。该文采用区间算法生成测试数据，比较了采用区间算法的软件测试方法和传统的软件测试方法，结果表明，区间代数方法很好地解决了传统测试方法不能解决的问题，发现程序代码中可能的计算异常，判断能否给出程序代码输出变量的上限范围和是否存在不可能达到的分支。

关键词：区间算法；测试数据；软件测试

Software Test Data Generation Method Based on Interval Algorithm

GAO Yue, SHAO Pei-nan, XING Jie-wen
（East-China Institute of Computer Technology, Shanghai 200233）

【Abstract】 The massive values computation module needs exhaustion tests in the software test. Insufficiency and limitation exist in traditional software test method. This paper produces test data by using interval algorithm and comparison is presented between interval algorithm test method and the traditional software test method. Conclusion indicates the interval algorithm can solve the problem that the traditional one can not, such as finding possible computation exception of program and judging whether it can present the upper limit scope of output variable or whether it has any branch hard to achieve.

【Key words】 interval algorithm; test data; software test

软件测试的本质是针对测试内容确定一组测试用例，并用这些测试用例执行程序，以此发现程序中的错误。测试数据(test data)的选择是软件测试的难点和关键，它决定了软件测试的质量，影响软件测试的效率、速度及软件的质量、市场和寿命。不同的测试数据发现程序错误的能力差别很大，为了提高测试效率、降低测试成本，应该选用高效的测试数据。测试中经常需要判断软件中的某些变量的范围是否越界、软件中是否存在不可达条件分支等，对于以上问题需要进行穷举测试，传统的测试方法对此就显露出了缺陷与局限。

为了自动核对计算结果，摩尔于20世纪50年代末提出了区间算法的概念，之后，区间算法很快成为了计算数学的一个活跃的分支。对于传统浮点算法，区间算法是一个根本的改革，它把计算的数存储为区间并对这些区间进行运算，由此避免浮点算法所产生的计算误差。该算法还能使区间参数直接包含在计算中，这在实际应用中具有重要的意义[1~2]。

1 测试数据的生成

广泛使用的软件测试的方法主要分为基于规格说明书的测试和基于程序结构的测试2大类，测试用例的生成分为功能测试数据生成和结构测试数据生成。

1.1 功能测试数据的生成

功能测试是一种按照需求规格说明设计测试数据的方法。它把程序看作内部不可见的黑盒子，完全不考虑程序内部结构和编码结构，以及程序中的语句及路径，只须了解程序输入和输出间的关系或程序的功能，完全依靠能够反映这一关系和程序功能的需求规格说明确定测试数据、判定测试结果的正确性。

1.1.1 边界值法

边界值分析通过输入空间的边界生成测试用例，其基本原理是错误更可能出现在输入变量的极值附近。边界值的基本操作是在最小值、略高于最小值、正常值、略低于最大值和最大值中取测试用例。这样，对于n个变量的函数，边界值分析会产生$4n+1$个测试用例。若想进一步测试程序的健壮性，还须在每个边界值的基础上再取略小于最小值的值和略大于最大值的值，观察超过极限时程序的容错力，同样会产生$6n+1$测试用例。边界值分析采用了可靠性理论的单缺陷假设，若拒绝这种假设，则须考虑当多个变量取极值时会出现的情况。因此，选择最坏情况测试，即对每个变量先进行基本边界值的选择，然后对这些集合进行笛卡儿积计算，以生成测试用例，这种方法比基本边界分析更完备，但大大增加了测试的工作量，n变量函数的最坏情况测试会产生5^n个测试用例。

1.1.2 等价类划分

如果软件输入域可以分为几个子区域，而且每个测试子域中的任何一个输入能测试相同的目标或者暴露相同的软件缺陷，那么这个子域就是一个等价类，整个测试域的划分就是等价类划分。等价类的重点是集合的划分，其主要思想是利用每个等价类的通用特点选择一个或一组数据作为测试用例。等价类划分的难点是选择类的等价关系。实际测试中，在任何一个等价类当中选择用例时，要同时考虑合法的输入空间及非法的输入空间。例如，计算超出范围的循环计数值的测试数据、失效的复杂数据串、实时系统中超出时间保护机制的测试用例等。另外，一定要考虑建立处理默认值、空白、空值、零值或者无输入等条件的等价区间。这些值具有

作者简介：高 月(1976－)，女，硕士、工程师，主研方向：软件测试；邵培南，硕士、研究员；邢洁雯，助理工程师

收稿日期：2006-12-01 **E-mail：**gyue2008@yahoo.com.cn

特殊性，在软件中必须作为单独等价区间进行特殊处理。

1.1.3　决策表法

在功能测试中，由决策表生成的测试用例是最完备的，因为决策表有逻辑的严格性。把条件解释为输入，把行动解释为输出，如果变量之间不是独立的，并且有一定的逻辑关系（例如存在 if-then-else 语句等），均可采用决策表法产生测试用例。输入变量若是布尔值，在边界值分析中没有任何意义，就可以采用决策表法生成测试用例。

1.2　结构测试数据生成

现实条件下人们还无法保证软件需求规格说明书的完全正确性和充分完备性。根据统计，测试中大部分未发现的缺陷，是没有正确覆盖流程中的各种错误路径[3]，因此，实际测试中还需更缜密的测试用例生成方法。结构测试数据生成方法就是很好的补充。与功能测试数据生成方法不同，结构生成方法是知道产品内部工作过程和了解程序内部逻辑结构，按照程序内部的结构对所有逻辑路径进行测试，检验程序中的每条路径是否都能按预定的要求正确工作，而不强调程序的具体功能。它主要用于高可靠性软件的测试，尤其是嵌入式软件测试。结构数据生成的核心问题可理解为：存在程序P，程序中的路径集合为$W\{w_1,w_2,\cdots, w_n\}$，运行程序P，寻找可通过所有W 的数据集合D。

结构生成方法主要依据各种逻辑覆盖准则，从算法和/或程序的结构来导出测试用例，每个测试用例由一组动作组成，这一组动作覆盖该算法的一个特定路径。选择测试用例使每种控制结构在程序运行中得到执行，这个过程称为覆盖，不同的覆盖要求得到对程序不同深度和不同完备性的测试。无论是控制流覆盖中的分支覆盖(设计 D 使 P 中的每一个分支至少通过一次，即真值通过一次，假值执行一次)、语句覆盖 (设计 D 使 P 中的每条语句至少被执行一次)、条件覆盖(设计 D 使 P 的判定中每个条件获得各种可能的结果)、路径覆盖(设计 D 使 P 中所有可能的路径都执行一次)、条件组合覆盖(设计 D 使 P 中每个判定中条件的各种组合至少出现一次)还是数据流的定义覆盖、引用覆盖，都可以看作是在不同的测试标准下生成测试用例。但对于测试人员而言，覆盖工作量极大、效率极低，尤其是复杂的大型程序，测试用例常以几何级数增长，测试人员要设计、选择、执行、分析大量数据，因此，选择合适的算法自动生成测试用例可以有效提高测试的质量和效率。

2　问题的提出

在武器装备软件的测试过程中，经常遇到下列问题：

(1)在程序语言检查中，变量的数据类型定义是否合理。在软件开发过程中，通常需要对变量进行类型定义。如果这些变量的定义范围超出这些变量类型的定义，就会出现计算异常。

(2)在内存检查中，软件中的数组及变量是否越界。在武器装备软件中，专业的特殊性使得某些变量对应物理模型中具体的物理量，因此，它们具有一定的物理意义约束，如果这些变量在程序中的范围超出物理意义的约束，就越界了。

(3)在控制流与逻辑检查中，是否存在失败的递归调用或不可达条件分支。条件分支是一种重要的程序结构，但是每个分支是否都是理论可以执行到的，还是即使经过穷举测试也存在某些分支执行不到[3]。

对于上述问题，传统的软件测试方法根本无法对其作出判断，该软件投入实际运行后可能导致严重后果。

以某计算模块程序代码为例：

```
float test1(float x,float y)
{ if(x<y) /*分支 1*/
        {z=sqrt(x+y-1)；}
  else /*分支 2*/
        { z=sqrt(x-y+1)；}
 return(z)；}
```

2.1　功能测试

对上述例子中的代码进行黑盒测试，要求变量x是[0,3]中的浮点数，y是[1,2]中的浮点数。根据文档设计了 2 组测试用例：(1)x=0, y=1；(2)x=3, y=2。实际运行结果分别是z=0 和z=sqrt2，和预期的结果一样。但是不能仅根据上述 2 组测试用例的结果就保证输入空间的所有可能数据的运行都无计算异常，除非对输入空间的所有可能数据都进行测试，才能下此结论，但这就成了穷举测试，而穷举测试针对连续型的数值变量在实际工程中往往难以实现[4]。由于选择测试用例的有限性 (不可能对所有可能的输入一一进行测试)，因此通过黑盒测试的程序不能保证程序代码中不存在计算异常，只能认为程序在抽样验证的意义下具备了正常的功能。

2.2　结构测试

对上述例子中的代码进行白盒测试。根据例中的程序结构，设计了 3 组测试用例(如图 1 所示)[5]：(1)x=0, y=2 (沿路径abd执行)；(2)x=3, y=2 (沿路径acd执行)；(3)x=2, y=2 (沿路径acd执行)。

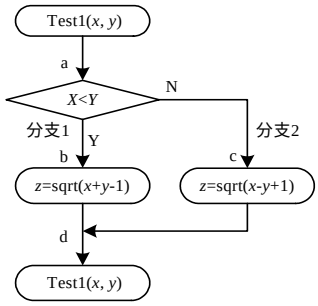

图 1　算例流程

这 3 组测试用例覆盖了程序代码中的所有语句和所有分支，语句覆盖率和分支覆盖率都达到了 100%，虽然满足了覆盖率的测试要求，但仍无法保证程序对输入空间的所有可能输入其运行无计算异常。

2.3　传统测试方法的不足和局限

对例 1 的代码进行黑盒测试和白盒测试，其结果显示：

(1)黑盒的功能测试和白盒的结构测试都无法保证程序中不会出现计算异常；

(2)黑盒的功能测试和白盒的结构测试无法给出程序中输出变量的取值范围；

(3)白盒的结构测试无法断言未覆盖分支是尚未覆盖到，还是无法覆盖到。

导致这样结果的主要原因是传统测试方法选择测试用例的有限性，传统的测试方法不可能对程序所有可能的输入进行一一测试，否则就成了穷举测试，但穷举测试往往是不可能的，尤其对于连续的输入空间而言。因此，传统的测试方法不能对上述问题作出正确与否的判断。

3　区间算法的数学理论基础[1~2]

3.1　区间数的定义

定义 1 对于给定的数对 $\underline{x}, \overline{x} \in \mathrm{R}$，若满足 $\underline{x} \le \overline{x}$，则有

界闭集合，有界闭区间数 $\tilde{x}$ 为

$$\tilde{x}=[\underline{x},\overline{x}]=\{x\in \mathrm{R}\mid \underline{x}\le x\le \overline{x}\} \tag{1}$$

其中，$\underline{x}$ 和 $\overline{x}$ 分别称为区间数 $\tilde{x}$ 的下端点和上端点；若 $\underline{x}=\overline{x}$，则定义区间数 $\tilde{x}$ 为点区间数。

3.2 区间四则运算的定义

给定区间数 $\tilde{x}=[\underline{x},\overline{x}]$，$\tilde{y}=[\underline{y},\overline{y}]$，$\tilde{x},\tilde{y}\in \mathrm{I(R)}$，区间的四则运算定义为

$$\tilde{x}+\tilde{y}=[\underline{x}+\underline{y},\overline{x}+\overline{y}] \tag{2}$$

$$\tilde{x}-\tilde{y}=[\underline{x}-\underline{y},\overline{x}-\overline{y}] \tag{3}$$

$$\tilde{x}\,\tilde{y}=[\min(\underline{x}\,\underline{y},\underline{x}\,\overline{y},\overline{x}\,\underline{y},\overline{x}\,\overline{y}),\max(\underline{x}\,\underline{y},\underline{x}\,\overline{y},\overline{x}\,\underline{y},\overline{x}\,\overline{y})] \tag{4}$$

$$\frac{\tilde{x}}{\tilde{y}}=[\underline{x},\overline{x}][\underline{y}]\quad 0\notin\tilde{y} \tag{5}$$

定义 2 设映射：$\mathrm{R}^n\to\mathrm{R}$，若存在区间映射：$F:\mathrm{IR}^n\to\mathrm{IR}$ (其中，IR是定义在R所有闭区间的集合)，如果对于 $\forall x\in\mathrm{R}^n$ 有 $f(x)=F(x)$ 和对于 $\forall x\in\mathrm{IR}^n$ 有 $f(x)\subseteq F(x)$ 成立，则称 $F(x)=F(x_1,x_2,\cdots,x_n)$ 是点函数 f 的区间扩展。

定义 3 设区间函数 $F:\mathrm{IR}^n\to\mathrm{IR}$，如果对于 $\forall x,Y\in\mathrm{IR}^n$ 且 $X\subseteq Y$，都有 $f(x)\subseteq F(Y)$ 成立，则称区间函数 F 具有单调包含性；如果任意的 $X\subset Y$，都有 $f(x)\subset F(Y)$ 成立，则称区间函数 F 具有严格的单调包含性。

3.3 一类条件函数的区间扩展

引理 1 设 X_1，X_2，Y_1，Y_2 是闭区间，其中，$X_1=[\underline{x},\overline{x}]$，$Y_1=[\underline{y},\overline{y}]$，$X_2=\{x\in\underline{x}\mid\exists\, y\in y_1,\partial x\prec y\}$，$Y_2=\{y\in Y\mid\exists\, x\in x_1,\partial x\prec y\}$，如果 $\underline{x}\prec\overline{y}$，那么 $X_2=[\underline{x},\overline{y}]\cap X_1$，$Y_2=[\underline{x},\overline{y}]\cap Y_1$；如果 $\underline{x}\ge\overline{y}$，那么 $X_2=\varnothing$，$Y_2=\varnothing$。

引理 2 设 X_1，X_2，Y_1，Y_2 是闭区间，其中，$X_1=[\underline{x},\overline{x}]$，$Y=[\underline{y},\overline{y}]$，$X_2=\{X\in X_1\mid\exists\, Y\in Y_1,\partial x=y\}$，$Y_2=\{Y\in Y\mid\exists\, X\in X_1,\partial x=y\}$，那么 $X_1=X_1\cap Y_1$，$X_2=X_2\cap Y_2$。

定理 1 设 X_1，X_2，Y_1，Y_2 是闭区间，其中，$X_1=[\underline{x_1},\overline{x_1}]$；$Y_1=[\underline{y_1},\overline{y_1}]$；$X_2=\{X\in X_1\mid\exists\, Y\in Y_1,\partial x\le y\}$；$Y_2=\{y\in Y_1\mid\exists\, x\in X,\partial x\le y\}$。如果 $\underline{x}\le\overline{y}$，则 $X_2=[\underline{x},\overline{y}]\cap X_1$，$Y_2=[\underline{x},\overline{y}]\cap Y_1$；如果 $\underline{x}\succ\overline{y}$，那么 $X_2=\varnothing$，$Y_2=\varnothing$。

定理 2 区间函数 $F(X,Y)=\{F_1(X_1,Y_1),\ X_1=[\underline{x},\overline{y}]\cap X_1\ Y_1=[\underline{x},\overline{y}]\cap Y$；$F_2(X_2,Y_2)$，$X_2=[\underline{y},\overline{x}]\cap X_2\ Y_2=[\underline{y},\overline{x}]\cap Y$；其中，$X_1=[\underline{x},\overline{x}]$，$Y_1=[\underline{y},\overline{y}]$ 是R上的闭区间。对于任意的 $XX\subseteq X$，$YY\subseteq Y$，都有 $F(XX)\subseteq F(X,Y)$ 成立。

根据上面的引理和定理，给出条件函数 $f(x,y)=\{f_1(x,y),x\prec y,\ f_2(x,y),x\ge y\}$ 的区间扩展形式：

$$F(X,Y)=\{F_1(X_1,Y_1),\ X_1=[\underline{x},\overline{y}]\cap X_1\ Y_1=[\underline{x},\overline{y}]\cap Y$$

$$F_2(X_2,Y_2),\ X_2=[\underline{y},\overline{x}]\cap X_2\ Y_2=[\underline{y},\overline{x}]\cap Y$$

其中，$X_1=[\underline{x_1},\overline{x_1}]$，$Y_1=[\underline{y_1},\overline{y_1}]$ 分别是变量 X，Y 的取值区间；$F(X,Y)=F_1(X_1,Y_1)\cup F_2(X_2,Y_2)$。

3.4 基于区间算法的测试数据的生成

根据以上区间数和区间计算的理论基础，以算例为例将区间算法应用于测试数据的生成。

3.4.1 基于区间算法的测试数据的生成分析

基于区间变量的对应转换程序中，X 为变量 X 的区间值，X=[0,3]；Y 为变量 Y 的区间值，Y=[1,2]。Test1([0,3],[1,2])的分析流程如下：

{(分支 1 区间计算结果)

{$X=[\underline{x},\overline{y}]\cap X$=[0,2]∩[0,3]=[0,2]

$Y=[\underline{x},\overline{y}]\cap Y$=[0,2]∩[1,2]=[1,2]

$X+Y$-1= [0, 2] + [1, 2]-1= [0, 3]

Z=sqrt($X+Y$-1) = sqrt([0,3])= [0, $\sqrt{3}$]}

{(分支 2 区间计算结果)

{$X=[\underline{y},\overline{x}]\cap X$ =[1,3]∩[0,3]=[1,3]

$Y=[\underline{y},\overline{x}]\cap Y$=[1,3]∩[1,2]=[1,2]

$X-Y$+1=[1, 3]-[1, 2] +1= [0, 3]

Z=sqrt($X-Y$+1) = sqrt([0,3])= [0, $\sqrt{3}$]}

Z=[0, $\sqrt{3}$]∪[0, $\sqrt{3}$]= [0, $\sqrt{3}$]}

经过区间算法的分析，得出如下结论：(1)在规定的输入变量取值范围内，程序计算不会出现诸如开方为负数、除数为 0 之类的计算异常；(2)在规定的输入变量取值范围内，程序输入量 z 的范围在[0, $\sqrt{3}$]之内；(3)在规定的输入变量取值范围内，程序的 2 个分支理论上都是可达的。

3.4.2 采用区间算法对计算异常的测试能力

为了说明区间算法对程序中存在计算异常可能情况的测试能力，假设变量 X 的取值范围是 X=[-∂,3+∂]，其中，∂ 大于 0；变量 Y 的取值范围 Y=[1,2]。Test1([-∂,3+∂],[1,2])的分析流程如下：

{(分支 1 区间计算结果)

{$X=[\underline{x},\overline{y}]\cap X$=[-∂,2]∩[-∂,3+∂]= [-∂,2]

$Y=[\underline{x},\overline{y}]\cap Y$=[-∂,2]∩[1,2]= [1,2]

$X+Y$-1=[-∂,2]+ [1,2]-1= [-∂,3]

Z=sqrt($X+Y$-1)出现计算异常！}

{(分支 2 区间计算结果

$X=[\underline{y},\overline{x}]\cap X$ =[1,3＋∂]∩ [-∂,3+∂]= [1, 3+∂]

$Y=[\underline{y},\overline{x}]\cap Y$=[1, 3+∂]∩[1,2]= [1,2]

$X-Y$+1= [1, 3+∂]-[1, 2] +1= [0, 3+∂]

Z=sqrt($X-Y$+1) = sqrt([0, 3+∂])= [0, $\sqrt{3+\partial}$]}

Z 的计算在分支 1 出现计算异常，在分支 2 计算正常。

经过区间算法的分析，可以发现在规定的输入变量 X=[-∂, 3+∂], Y=[1,2]的取值范围内，程序在分支 1 出现计算异常，程序在分支 2 计算正常。因此，利用区间算法方法不用执行程序就可以分析程序代码中是否存在计算异常。

3.4.3 采用区间算法对不可能到达分支的测试能力

为了说明区间代数方法对程序中不可能达到分支情况的测试能力，假设变量 X 的取值范围是 X=[2.5,3]，变量 Y 的取值范围是 Y=[4,5]。Test1 ([2.5,3], [4,5])的分析流程如下：

{(分支 1 区间计算结果)

{ $X=[\underline{x},\overline{y}]\cap X$=[2.5,5]∩[2.5,3]= [2.5,3]

$Y=[\underline{x},\overline{y}]\cap Y$=[2.5,5]∩[4,5]= [4,5]

$X+Y$-1= [2.5, 3] + [4, 5]-1= [5.5, 7]

Z=sqrt($X+Y$-1) =sqrt {[5.5, 7]} = [$\sqrt{5.5}$, $\sqrt{7}$]}

经过区间代数方法的分析可以发现，在规定的输入变量取值范围内，程序存在不可能达到的分支。

（下转第 62 页）