域测试

Domain Test

目录

CONTENTS

目录

CONTENTS

目录

CONTENTS

6.1 域错误-基本概念

输入域和程序路径

输入域：程序所有输入数据的集合

程序路径：从程序入口到程序中一些“关心”点所组成的一组有序的指令。一般对应程序的某个控制流/数据流。如果存在一个输入使得程序执行一条路径，就说这条路径是可行的，否则就是不可行的

6.1 域错误-基本概念

计算错误和域错误

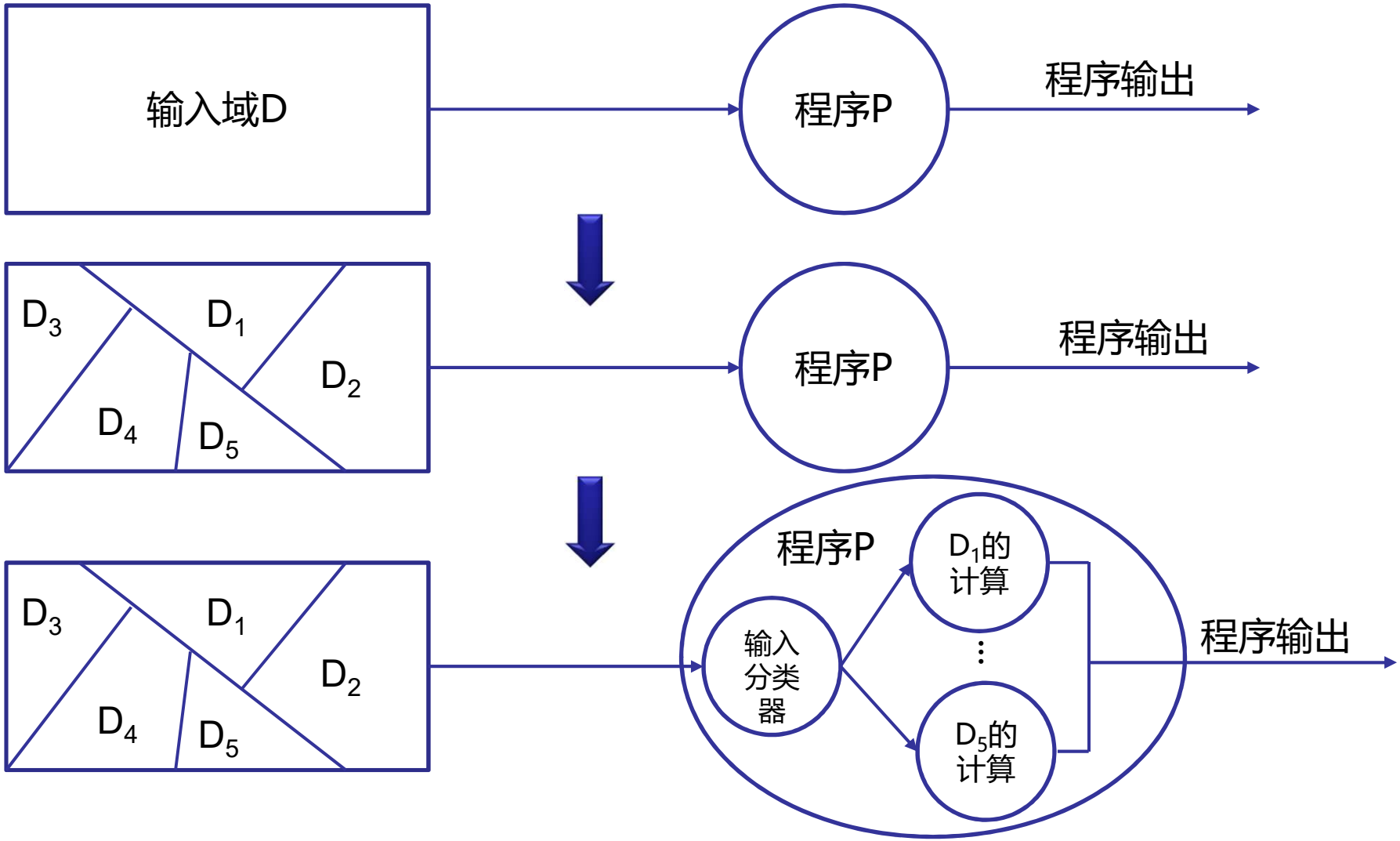
计算错误：当一个具体的输入数据使得程序执行了正确的路径（即所期望的路径）但输出结果是错误的，我们称这类错误为计算错误，这一般是由于执行路径中的赋值语句出现了错误

域错误：当输入一个具体的输入数据后，执行了一条错误的、不是期望的程序路径，这一般是由程序中的谓词出现错误所导致的

6.1 域错误-程序域的概念

可以把一个计算程序看成执行一个抽象的映射函数。引起同样路径执行的输入子集称为一个**输入域（输入子域）**。程序可将一个域映射到它自身的一条路径。因此，可将程序的输入空间分成一组有限数量的**子域**，并给每一个**输入子域**分配一条不同的程序路径

6.1 域错误-程序域的概念



6.1 域错误-术语

输入分类器：程序中的一个概念上的内置机制来决定选择一条具体路径所需要的计算方法。实际上指的是程序中的一组**有序谓词**

域：是一个输入值的集合，对于集合中的每一个成员，程序执行相同的计算。我们希望找到这样的**最大域**，它使得在邻域上程序执行不同的计算

域错误：如果程序没有正确执行输入分类，就说这个程序有一个域错误

目录

CONTENTS

6.2 域错误测试

基于流图的测试技术和**域测试技术**的不同

基于流图的测试技术是从控制流图或数据流图中选择路径来满足覆盖标准，没有明确地考虑要检查某种特定的错误类型。即**无目标**测试

域测试首先定义了一类错误，叫做域错误，接着选择测试数据来检测这些错误。即**有目标**测试

6.2 域错误测试

从以下三个方面来讨论域测试

域的源：解释程序谓词如何作为一个输入分类器使用

域错误类型：如何对谓词进行一个细微的改变从而导致域错误，这个细微的改变解释为编程缺陷

选择测试数据来发现域错误：一个测试选择标准，用来说明如何选择输入数据。这些选择的测试数据将会揭示域错误的某些特殊类型

目录

CONTENTS

6.3 域的源

从程序源代码识别域的方法

1、从所给的源代码中画出程序控制流图

2、找出谓词的所有可能（一个谓词可能会有多个解释）的解释（只用输入向量和可能的常量向量来表达谓词）

3、分析被解释的谓词从而识别域

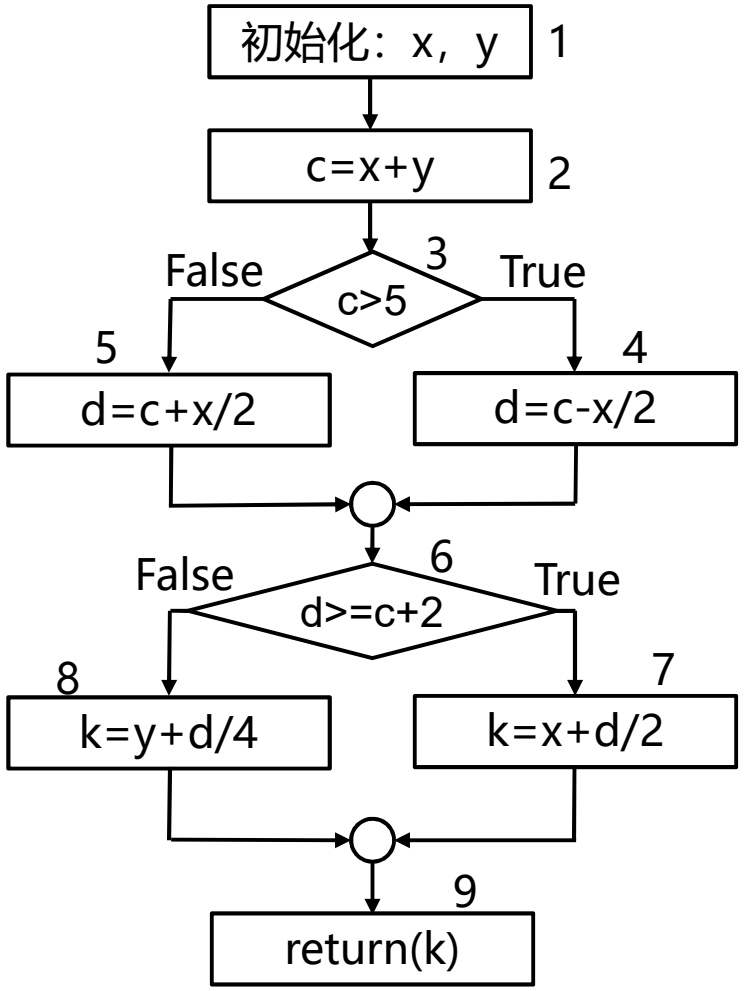
6.3 域的源-例子

解释程序域的函数（例）

```
int codedomain(int x, int y) {
  int c, d, k;
  c = x + y;
  if (c>5) d = c-x/2;
  else     d = c+x/2;
  if (d>=c+2) k=x+d/2;
  else        k=y+d/4;
  return(k);
}
```

6.3 域的源-例子

例子函数的程序控制流图



P_1: **x+y>5**

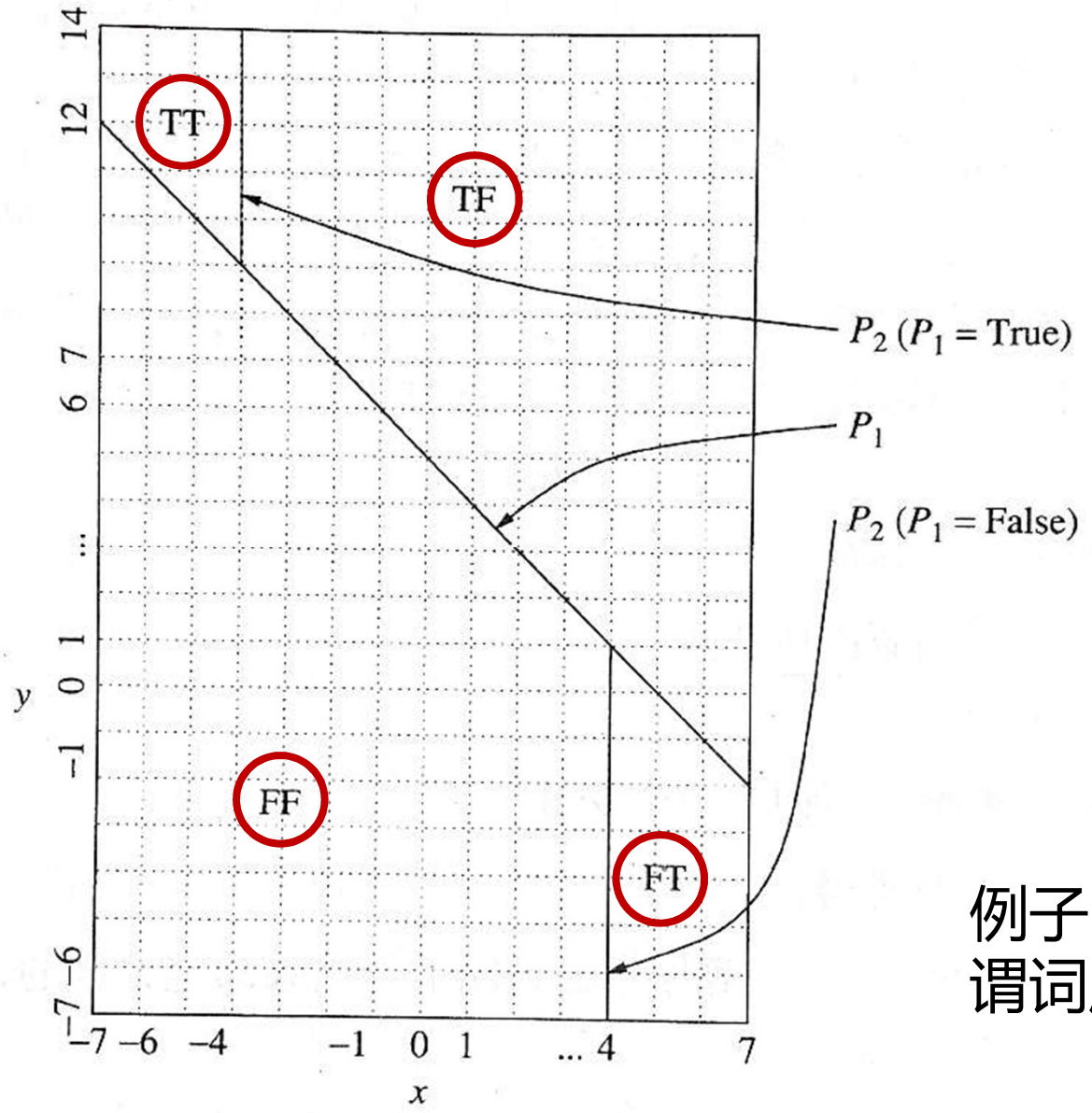
P_1: **False**
P_2: **x>=4**

P_1: **True**
P_2: **x<=-4**

6.3 域的源-例子

例子函数的程序控制流图中第二个if语句的两种解释

P1的取值	P2的取值
True	x<=-4
False	x>=4



例子函数解释的谓词所得到的域(图6.4)

目录

CONTENTS

6.4 域错误的类型-一些术语

域的属性

一个域是一组值的集合，这个集合让程序执行相同的计算

一个域可以表示成一个谓词集合。域的每个元素都要满足域的这组判定

6.4 域错误的类型-一些术语

从几何学的角度来看，一个域可以用一组称为**边界不等式**的**约束集**来定义

例如：图6.4中域**TT**的边界不等式的约束集:

P1：x+y>5 ≡ True

P2： x<=-4 ≡ True

6.4 域错误的类型-一些术语

封闭边界：如果一个边界上的点都在感兴趣的域中，就称这个边界为封闭的

例：

图6.4域TT的一条封闭边界：

P2：x<=-4

6.4 域错误的类型-一些术语

开放边界：如果一个边界上的点不属于感兴趣的域中，就称这个边界为开放的

例：

图6.4域TT的一条开放边界：

P1：x+y>5

6.4 域错误的类型-一些术语

封闭域：如果一个域的所有边界都是封闭的

6.4 域错误的类型-一些术语

开放域：如果一个域有开放的边界

6.4 域错误的类型-一些术语

极点：如果一个点是两条或者多条边界的交点，那么就称这个点为极点

6.4 域错误的类型-一些术语

邻域：如果两个域有一个相同的边界不等式，那么这两个域互为邻域

6.4 域错误的类型

域错误的原因

—个不正确的指定谓词

—个不正确的赋值影响了谓词中的一个变量

6.4 域错误的类型

不同类型的域错误

封闭错误：意味着一个边界原本是封闭的但结果是开放的，或者相反。其一般是由于边界不等式中关系算符对“=”运算的包含错误

边界移动错误：边界移动错误表示实际实现的边界平行于原本边界，这一般是由于边界不等式中的常量错误

边界倾斜错误：如果定义边界的判定中变量的常量系数取错误的值，那么就产生了边界倾斜错误

6.4 域错误的类型

不同类型的域错误：

例子

封闭错误：

关系运算符<=实现为<

关系运算符<实现为<=

边界移动错误：

本意：P1: x+y>5

错误实现为：P1’：x+y>4

边界倾斜错误：

本意：P1: x+y>5

错误实现为：P1’：x+0.5y>5

6.4 域错误的类型

如果出现了这几类域错误，则输入点将要落入错误的域中。（输入域和功能域的对应关系：对于一个域中的所有点，程序都执行相同的操作）

目录

CONTENTS

6.5 ON点和OFF点

域测试的目标是发现域错误，是要发现对域错误**最敏感的数据点（落在边界或是靠近边界的数据点）**，通过用这些数据点执行程序来发现错误

6.5 ON点和OFF点

ON点

给定一个边界，一个**ON点**就是在边界上的点或者是非常靠近边界上的点

6.5 ON点和OFF点

两种方法来选择ON点

如果要选择的点正好落在边界上，那么选择这样的点作为**ON点**。如果边界不等式得到一个精确解，那么选择这个精确解作为**ON点**

如果边界不等式得到一个近似解，则选择非常靠近边界的点。对于开放域，选择外部点；对于封闭域，选择内部点。

6.5 ON点和OFF点

选择ON点例子

P_{ON1}: x+7y>=6 (封闭的域)

选x=0，y的近似解y=0.8571428，则点(0, 0.858)为ON点，在域内

P_{ON2}: x+7y<6 (开放的域)

点(0, 0.858)为ON点，在域外

6.5 ON点和OFF点

OFF点

边界上的**OFF点**就是那些远离边界上的点

6.5 ON点和OFF点

两种方法来选择OFF点

如果对于边界这个域是开放的，那么对应边界的**OFF点**就是在域内的一个内部点，离边界距离为**ε**

如果对于边界这个域是封闭的，那么对应边界的**OFF点**就是在域外的一个外部点，离边界距离为**ε**

符号**ε**：表示一个任意小的值

6.5 ON点和OFF点

选择OFF点

P_{OFF1}: x+7y>=6 (封闭的域)

（-1， 0.99）在域的外部，是一个OFF点

例子

P_{OFF2}: x+7y<6 (开放的域)

（-1， 0.99）在域的内部，是一个OFF点

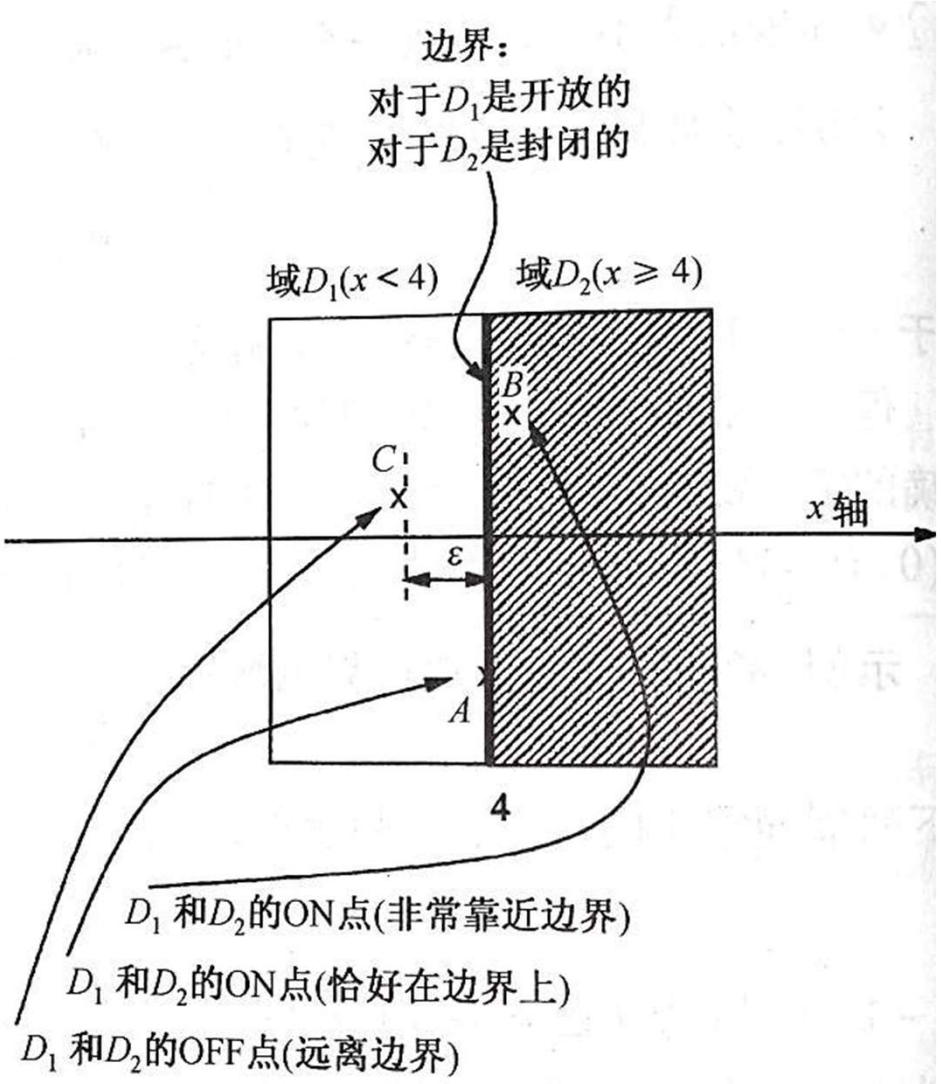
6.5 ON点和OFF点

结论

当测试一个封闭边界时，**ON点**在被测域中，**OFF点**在邻域中

当测试一个开放边界时，**ON点**在邻域中，**OFF点**在被测域中

6.5 ON点和OFF点-例子



D1:x<4
D2:x>=4

A=4
B=4.00001
C=3.95

目录

CONTENTS

6.6 测试选取标准

域测试的假设

程序在相邻的邻域中执行不同的计算（保证数据落在错误域中肯定会导致失败-计算结果不同）

边界谓词是输入向量的线性函数（便于计算和可视化）

6.6 测试选取标准

域测试选取标准：对于每个域的每个边界，按照**ON-OFF-ON**的次序选取三个点A、C和B，可以找到以下几种**错误**：

1、封闭不等式边界：a、边界移动导致域减少；b、边界移动导致域增加；c、边界倾斜；d、封闭错误

2、开放不等式边界：a、边界移动导致域减少；b、边界移动导致域增加；c、边界倾斜；d、封闭错误

3、等式边界

6.6 测试选取标准-(封闭不等式)边界移动导致域减少

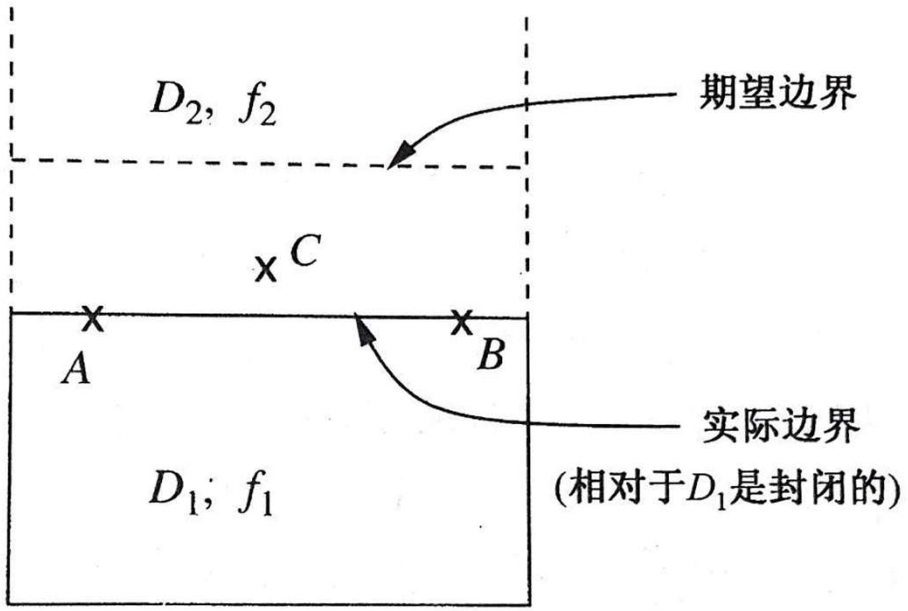


图 6.7　边界移动导致域减少(封闭不等式)

与D1和D2相关的计算分别为f1和f2，且f1！=f2

表6.2 边界移动导致域减少的错误检测（封闭不等式）

测试数据	实际输出	期望输出	检测出缺陷
A	f1(A)	f1(A)	否
B	f1(B)	f1(B)	否
C	f2(C)	f1(C)	是

6.6 测试选取标准-(封闭不等式)边界移动导致**域增加**

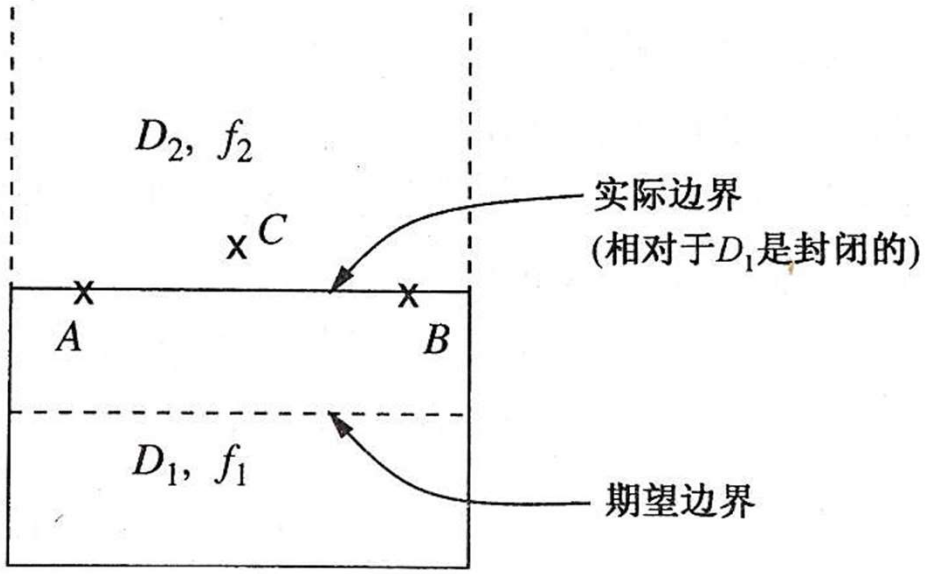


图 6.8　边界移动导致域增加(封闭不等式)

表6.3 边界移动导致域增加的错误检测（封闭不等式）

测试数据	实际输出	期望输出	检测出缺陷
A	f1(A)	f2(A)	是
B	f1(B)	f2(B)	是
C	f2(C)	f2(C)	否

6.6 测试选取标准-(封闭不等式)边界倾斜

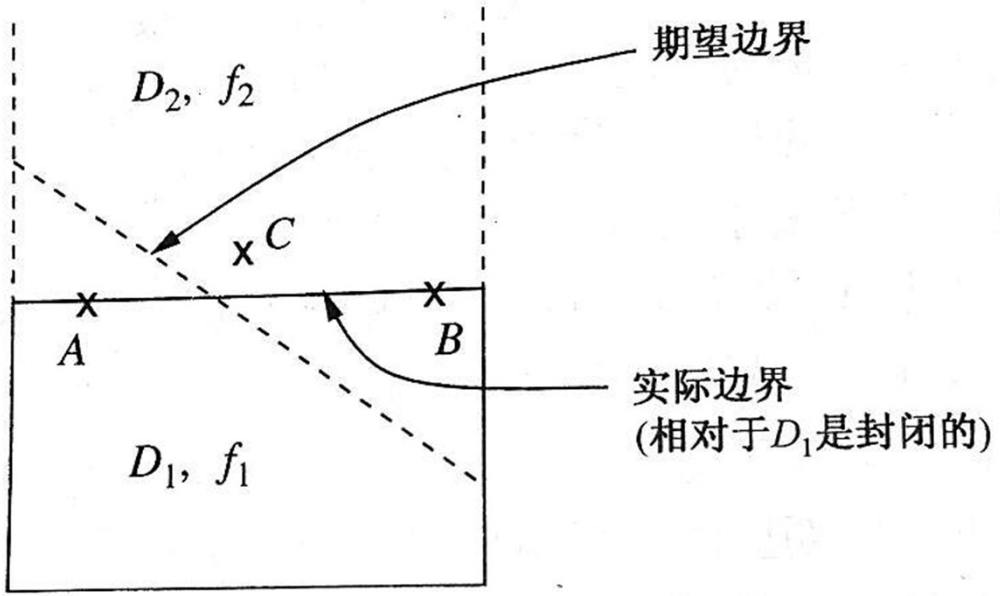


图 6.9　边界倾斜(封闭不等式)

表6.4 边界倾斜的错误检测(封闭不等式)

测试数据	实际输出	期望输出	检测出缺陷
A	f1(A)	f1(A)	否
B	f1(B)	f2(B)	是
C	f2(C)	f2(C)	否

6.6 测试选取标准-(封闭不等式)封闭错误

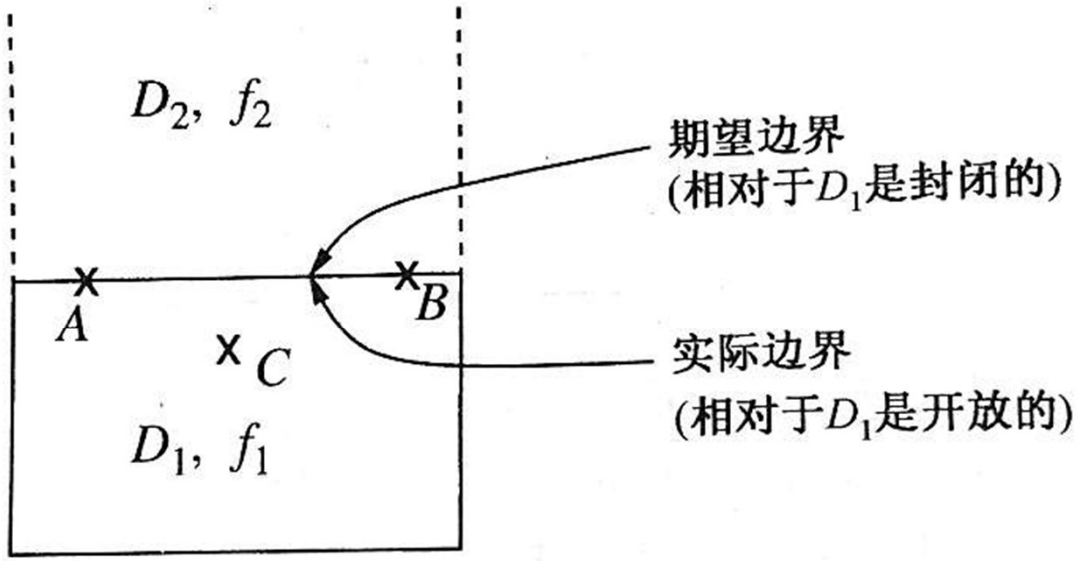


图 6.10　封闭错误(封闭不等式)

表6.5 封闭错误的检测（封闭不等式)

测试数据	实际输出	期望输出	检测出缺陷
A	f2(A)	f1(A)	是
B	f2(B)	f1(B)	是
C	f1(C)	f1(C)	否

6.6 测试选取标准-(开放不等式)边界移动导致**域减少**

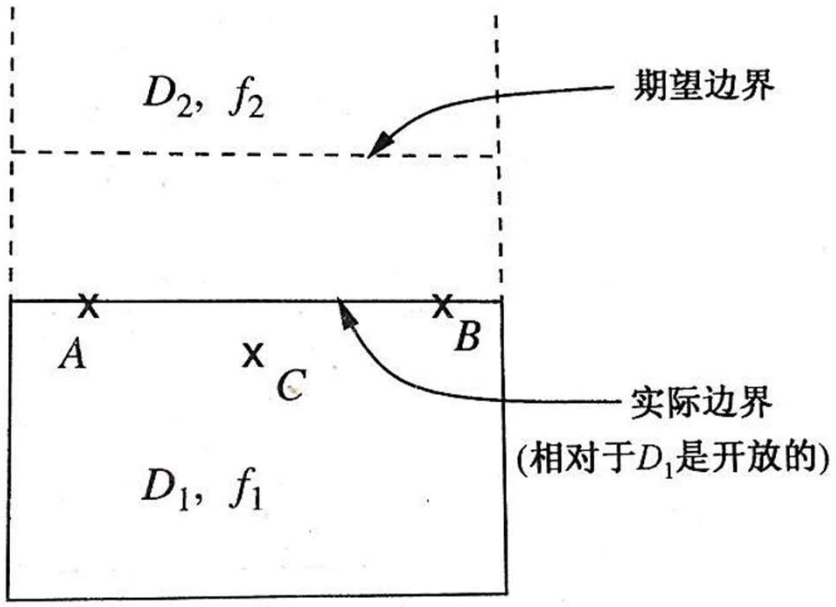


图 6.11　边界移动导致域减少(开放不等式)

表6.6 边界移动导致域减少的错误检测（开放不等式）

测试数据	实际输出	期望输出	检测出缺陷
A	f2(A)	f1(A)	是
B	f2(B)	f1(B)	是
C	f1(C)	f1(C)	否

6.6 测试选取标准-(开放不等式)边界移动导致**域增加**

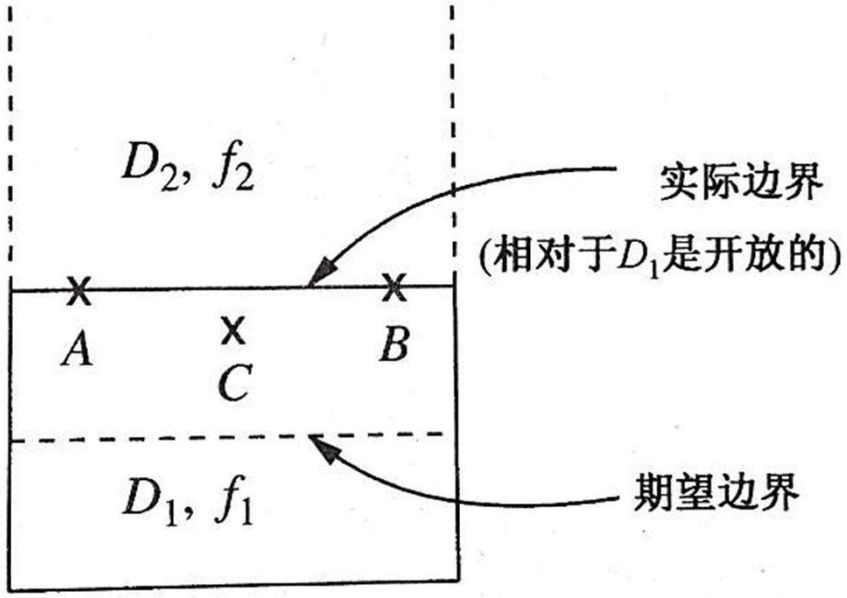


图 6.12　边界移动导致域增加(开放不等式)

表6.7 边界移动导致域增加的错误检测（开放不等式）

测试数据	实际输出	期望输出	检测出缺陷
A	f2(A)	f2(A)	否
B	f2(B)	f2(B)	否
C	f1(C)	f2(C)	是

6.6 测试选取标准-(开放不等式)边界倾斜

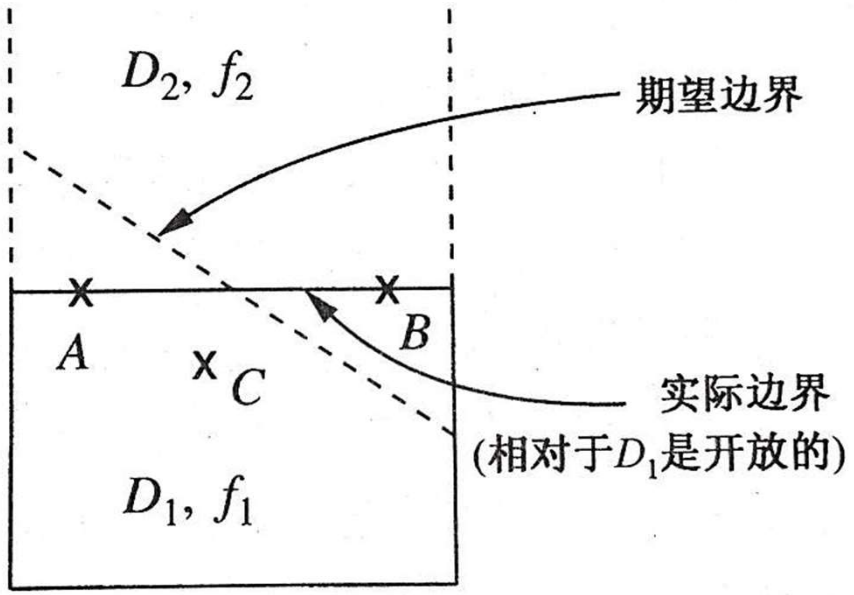


图 6.13　边界倾斜(开放不等式)

表6.8 边界倾斜的错误检测(开放不等式)

测试数据	实际输出	期望输出	检测出缺陷
A	f2(A)	f1(A)	是
B	f2(B)	f2(B)	否
C	f1(C)	f1(C)	否

6.6 测试选取标准-(开放不等式)封闭错误

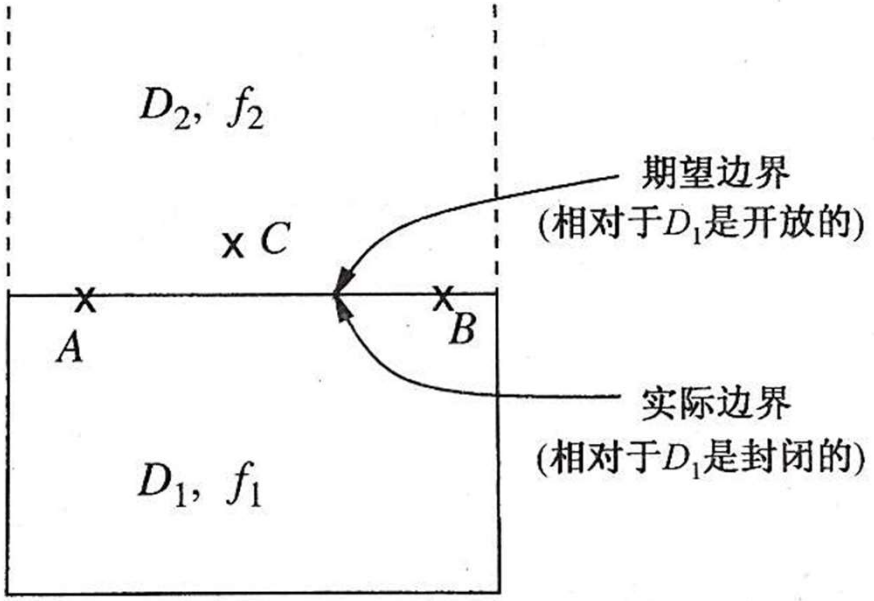


图 6.14　封闭错误(开放不等式)

表6.9 封闭错误的检测（开放不等式)

测试数据	实际输出	期望输出	检测出缺陷
A	f1(A)	f2(A)	是
B	f1(B)	f2(B)	是
C	f2(C)	f2(C)	否

6.6 测试选取标准-等式边界

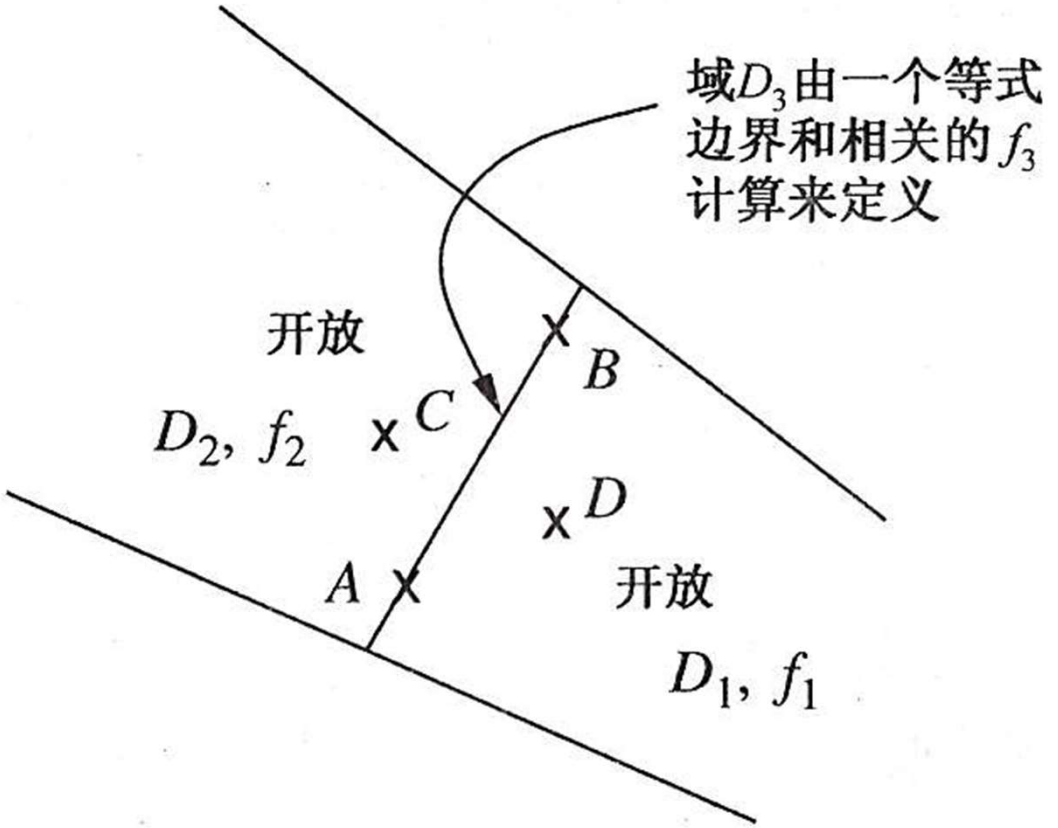


图 6.15　等式边界

域测试

End