第4章　黑盒测试技术

4.1　黑盒测试技术概述

黑盒测试，又称数据驱动测试，指的是把被测软件看作一个黑盒子，我们不去关心盒子里的结构如何，只关心软件的输入数据和输出结果。

采用黑盒测试的目的主要是在已知软件产品所应具有的功能的基础上，检查程序功能能否按需求规格说明书的规定正常使用，测试各个功能是否有遗漏，检测性能等特性要求是否满足；检测人机交互是否错误，检测数据结构或外部数据库访问是否错误，程序是否能适当地接收输入数据而产生正确的输出结果，并保持外部信息（如数据库或文件）的完整性；检测程序初始化和终止方面的错误。

黑盒测试的技术方法主要包括等价类划分法、边界值分析法、因果图法、决策表法、场景法等，下面将一一给大家介绍。

4.2　等价类划分法

现有一个小程序，能够求出三个在－10000 到＋10000 间整数中的最大者，程序界面如图 4-1所示，用等价类划分法设计测试用例。

图 4-1　求三个数中最大值的程序界面

4.2.1 等价类划分法的思想

如何用等价类划分法对上面的程序进行测试用例设计呢？在设计测试用例之前，我们来看看什么是等价类划分法？等价类划分法是怎样产生的？

很多原因可能导致完全测试是不可能的，比如，输入量太大，输出结果太多，软件执行路径太多，软件说明书是主观的，没有客观标准等。正是由于实现穷举测试的不可能性，才产生了等价类划分法。等价类划分法是从大量的可能输入数据中选取一部分数据来进行测试，这部分数据代表他所在等价类中的其他数据，其测试结果等同于其他数据的测试结果。即把程序的输入域划分成若干部分(子集)，然后从每一个子集中选取少量具有代表性的数据作为测试用例。如何划分等价类、如何从划分好的各个区域中选取有代表性的数据成为等价类划分法的关键问题。

测试既要对正确的输入进行测试，也要对不正确的输入进行测试。因此等价类分为两种，有效等价类和无效等价类。有效等价类指对软件规格说明书而言，是有意义的、合理的输入数据所构成的集合，利用有效等价类中的数据检验程序是否实现了规格说明书中预先规定的功能和特性。无效等价类指对规格说明书而言，是无意义的、不合理的输入数据所组成的集合，利用无效等价类中的数据检验程序对错误输入的处理能力。

针对是否对无效的数据进行测试，可以将等价类测试分为两种：标准等价类测试(也称一般等价类测试)和健壮等价类测试。

下面通过实例说明标准等价类测试与健壮等价类测试。

实例描述：某函数 F 有两个变量 x1，x2，要求两输入变量的取值范围如下：

a≤x1≤d，有效区间为[a，b)，(b，c)，[c，d]；

e≤x2 ≤g，有效区间为[e，f)，[f，g]；

x1，x2 的无效区间为：x1＜a，x1＞d；x2＜e，x2＞g

分别用各种等价类的划分形式进行测试。

1.标准等价类测试

标准等价类测试在设计测试用例时只考虑有效数据，不考虑无效数据，测试用例使用每个有效等价类中的一个值。标准等价类测试又可以分为强标准等价类测试和弱标准等价类测试。

(1)弱标准等价类测试。

特点：不考虑无效数据，只考虑有效数据，设计测试用例，使测试用例使用到每个有效等价类中的一个值即可，如图 4-2 所示。

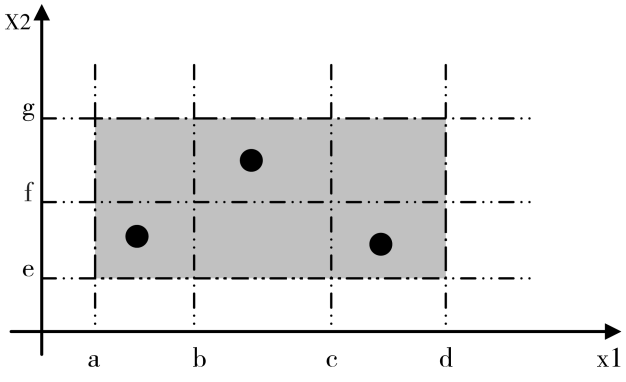

图 4-2 弱标准等价类测试

在图形中，从黑色圆点所在区域选取测试用例，即可满足弱标准等价类测试。

(2)强标准(一般)等价类测试。

特点：不考虑无效数据，只考虑有效数据，设计测试用例，在每一个有效等价类中要选择至少一个测试用例，如图 4-3 所示。

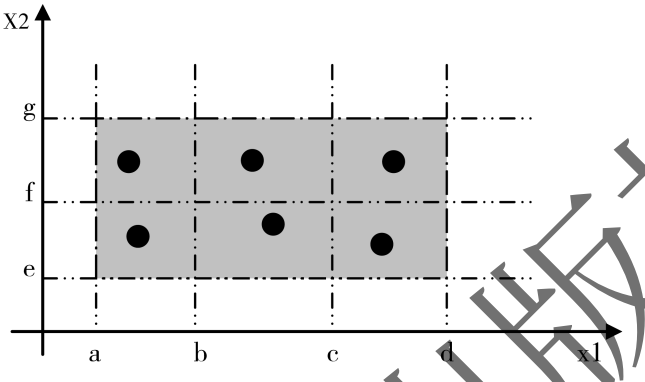

图 4-3　强标准等价类测试

2.健壮等价类测试

健壮等价类测试在设计测试用例时考虑了无效等价类，对于有效输入，测试用例从每个有效等价类中取一个值；对于无效输入，一个测试用例中只有一个无效值，其他值均取有效值。健壮等价类测试又可以分为弱健壮等价类测试和强健壮等价类测试。

(1)弱健壮等价类测试。

特点：既考虑有效数据，又考虑无效数据，对于有效输入数据，设计测试用例，使用每个有效等价类中的一个值；对于无效输入数据，设计测试用例，只使用无效等价类中的一个无效值，其余值都是有效的，如图 4-4 所示。

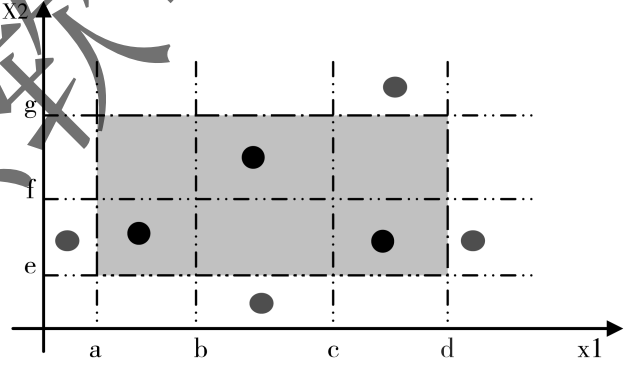

图 4-4　弱健壮等价类测试

(2)强健壮等价类测试。

特点：既考虑有效数据，又考虑无效数据，即从每个有效等价类和每个无效等价类中都至少要选择一个测试用例，如图 4-5 所示。

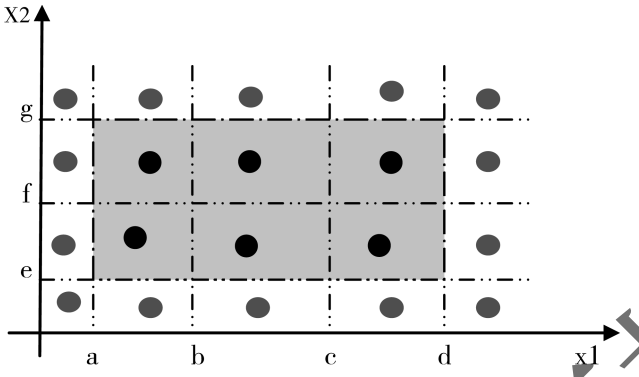

图 4-5　强健壮等价类测试

4.2.2　等价类划分法的原则及方针

1.等价类划分原则

根据不同的输入情况，等价类具有不同的划分方式。

(1)按照区间划分——在输入条件规定了取值范围或值的个数的情况下，可以确定一个有效等价类和两个无效等价类。

例：程序输入条件为小于 100 大于 10 的整数 x，则有效等价类为 10＜x＜100，两个无效等价类为 x≤10 和 x≥100。

(2)按照数值划分——在规定了一组输入数据(假设包括 n 个输入值)，并且程序要对每一个输入值分别进行处理的情况下，可确定 n 个有效等价类(每个值确定一个有效等价类)和一个无效等价类(所有不允许的输入值的集合)。

例：程序输入 x 取值于一个固定的枚举类型{1,3,7,15}，且程序中对这 4 个数值分别进行了处理，则有效等价类为 x＝1、x＝3、x＝7、x＝15，无效等价类为 x≠1,3,7,15 的值的集合。

(3)按照数值集合划分——在输入条件规定了输入值的集合或规定了“必须如何”的条件下，可以确定一个有效等价类和一个无效等价类(该集合有效值之外)。

例：程序输入用户口令的长度必须是 4 位的串，可以确定一个有效等价类是串的长度为 4，一个无效等价类是长度不为 4 的串。

(4)按照限制条件或规则划分——在规定了输入数据必须遵守的规则或限制条件的情况下，可确定一个有效等价类(符合规则)和若干个无效等价类(从不同角度违反规则)。

例：程序输入条件为取值为奇数的整数 x，则有效等价类为 x 的值为奇数的整数，无效等价类为 x 的值不为奇数的整数。

(5)细分等价类——在确知已划分的等价类中各元素在程序中的处理方式不同的情况下，则应再将该等价类进一步划分为更小的等价类，并建立等价类表。

例：程序输入条件为以字符‘a’开头、长度为 8 的字符串，并且字符串不包含‘a’～‘z’之外

的其他字符，则有效等价类为满足了上述所有条件的字符串，无效等价类为不以‘a’开头的字符串、长度不为 8 的字符串和包含了‘a’～ ‘z’之外其他字符的字符串。

2.等价类测试的指导方针

(1)如果实现的语言是强类型语言(无效值输入会引起系统运行时出错)，则没有必要使用健壮等价类测试。

(2)如果错误输入检查非常重要，则应进行健壮等价类测试。

(3)如果输入数据以离散区间或集合的形式定义，则等价类测试是合适的，当然也适用于变量值越界会造成故障的系统。

(4)在发现合适的等价关系之前，可能需要多次尝试。

4.2.3　等价类划分法测试用例设计

在了解了等价类划分法设计测试用例的思想，掌握了等价类划分法的形式后，我们对本节开始给出的程序进行测试用例设计。

在设计测试用例时，根据需求规格说明书的要求及程序的具体情况，决定采用哪种等价类划分方式，本程序适合于采用弱健壮等价类测试。

通常情况下，用等价类划分法设计测试用例的步骤如下：

(1)划分等价类。

通常划分等价类的思想是先从程序的规格说明书中找出各个输入条件，再为每个输入条件划分两个或多个等价类，形成若干互不相交的子集。通常分为以下几个步骤：

①先考虑输入数据的类型(合法型和非法型)；

②再考虑数据范围(合法型中的合法区间和非法区间)；

③画出示意图，区分等价类；

④为每一个等价类编号；

⑤考虑输出，进行补充。

(2)建立等价类表，列出所有划分出的等价类，并为每一个等价类规定一个唯一的编号。

(3)从划分出的等价类中按以下的原则设计测试用例。

①设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步，直到所有的有效等价类都被覆盖为止。

②设计一个新的测试用例，使其仅覆盖一个尚未被覆盖的无效等价类，重复这一步，直到所有的无效等价类都被覆盖为止。

对于本节提出的问题，我们是这样划分等价类的：

按照如何划分等价类的步骤进行等价类划分，示意图如图 4-6 所示。

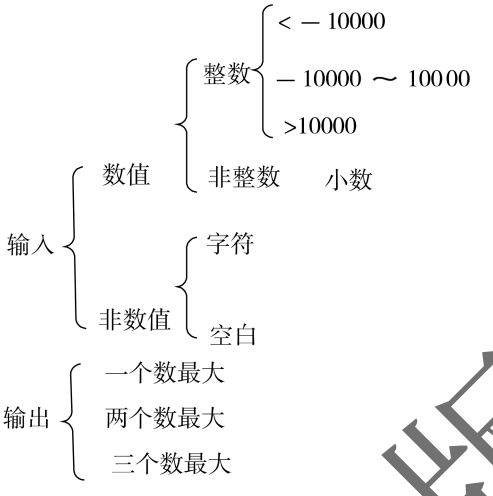

图 4-6　等价类划分示意图

根据已经划分好的等价类建立等价类表，如表 4-1 所示。

表 4-1　　等价类表

条件	有效等价类		编号	无效等价类	编号
输入	整数		1	小数	12
				字符	13
				空白	14
	三个有效数	−10000≤a≤10000	2	a＜−10000	15
				a＞10000	16
		−10000≤b≤10000	3	b＜−10000	17
				b＞10000	18
		−10000≤c≤10000	4	c＜−10000	19
				c＞10000	20
输出	最大值是一个数	a 最大	5		
		b 最大	6		
		c 最大	7		
	最大值是两个数	a＝b＞c	8		
		b＝c＞a	9		
		a＝c＞b	10		
	最大值是三个数	a＝b＝c	11		

根据等价类表设计测试用例，如表 4-2 所示。

表 4-2　　测试用例

用例编号	测试用例	覆盖等价类	预期输出
1	(5000,0,－5000)	1、2、3、4、5	a 最大
2	(0,5000,－5000)	1、2、3、4、6	b 最大
3	(0,－5000,5000)	1、2、3、4、7	c 最大
4	(2000,2000,0)	1、2、3、4、8	a、b 最大
5	(0,2000,2000)	1、2、3、4、9	b、c 最大
6	(2000,0,2000)	1、2、3、4、10	a、c 最大
7	(2000,2000,2000)	1、2、3、4、11	a、b、c 最大
8	(2.6，5.5，8)	12	输入有小数，错误
9	(三,3,3)	13	输入有字符，错误
10	(3，　,6)	14	输入有空白，错误
11	(－20000,10,100)	15	数字 a 超出范围
12	(20000,10,100)	16	数字 a 超出范围
13	(10,－20000,100)	17	数字 b 超出范围
14	(10,20000,100)	18	数字 b 超出范围
15	(10,100,－20000)	19	数字 c 超出范围
16	(10,100,20000)	20	数字 c 超出范围

4.2.4　应用实例

实例描述：NextDate 函数

NextDate 函数有三个变量 month，day，year 的函数，输出为输入日期下一天的日期。如：输入为 2007 年 7 月 19 日，输出为 2007 年 7 月 20 日。要求三个变量都为整数，且满足：

条件 1：1 ≤month ≤12；

条件 2：1 ≤day ≤31；

条件 3：1912 ≤year ≤2050。

对于 NextDate 函数问题如何设计测试用例？如何将等价类划分法应用到 NextDate 函数问题中？又如何划分等价类？

我们可以从以下几个方面来考虑：

(1)首先确定项目中的输入条件为 month、day、year；

(2)再根据输入条件的输入域划分等价类；

(3)采用两种方式进行等价类划分，设计弱标准等价类测试用例、强标准等价类测试用例、弱健壮等价类测试用例、强健壮等价类测试用例。

具体步骤：

(1)划分等价类。

①有效等价类：Year、Month、Day 的有效值区间定义如下：

- M1＝{month：1≤month ≤12}
- D1＝{day：1≤ day ≤31}
- Y1＝{year：1912≤year ≤2050}

②无效等价类：

- M2＝{month：month＜1}
- M3＝{month：month＞12}
- D2＝{day：day＜1}
- D3＝{day：day＞31}
- Y2＝{year：year＜1912}
- Y3＝{year：year＞2050}

NextDate 函数的弱健壮等价类测试用例如表 4-3 所示。

表 4-3　　弱健壮等价类测试用例

编号	测试用例(month,day,year)			覆盖的等价类	预期输出
Test1	6	15	1912	M1,D1,Y1	1912.6.16
Test2	－1	15	2005	M2,D1,Y1	month 不在有效值内
Test3	13	15	2005	M3,D1,Y1	month 不在有效值内
Test4	6	－1	2005	M1,D2,Y1	day 不在有效值内
Test5	6	32	2005	M1,D3,Y1	day 不在有效值内
Test6	6	15	1911	M1,D1,Y2	year 不在有效值内
Test7	6	15	2051	M1,D1,Y3	year 不在有效值内

NextDate 函数的强健壮等价类测试用例(部分)如表 4-4 所示。

表 4-4　　测试用例

编号	测试用例(month,day,year)			覆盖的等价类	预期输出
Test1	6	15	1912	M1,D1,Y1	1912.6.16
Test2	－1	15	1918	M2,D1,Y1	month 不在有效值内
Test3	6	－1	1918	M1,D2,Y1	day 不在有效值内
Test4	6	15	1911	M1,D1,Y2	year 不在有效值内
Test5	－1	－1	2005	M2,D2,Y1	month,day 不在有效值内
Test6	6	－1	1911	M1,D2,Y2	day,year 不在有效值内
Test7	－1	15	1911	M2,D1,Y2	month,year 不在有效值内
Test8	－1	－1	1911	M2,D2,Y2	month,day,year 不在有效值内
…	…	…	…	…	…

(2)详细的等价类划分。

考虑对输入日期的处理：

M1＝{month：month＝1,3,5,7,8,10,12}

M2＝{month：month＝4,6,9,11}

M3＝{month：month＝2}

D1＝{day：1≤day ≤28}

D2＝{day：day＝29}

D3＝{day：day＝30}

D4＝{day：day＝31}

Y1＝{year：year 是闰年}

Y2＝{year：year 是平年}

NextDate 函数的弱标准等价类测试用例如表 4-5 所示。

表 4-5　　测试用例

编号	测试用例(month,day,year)			覆盖的等价类	预期输出
Test1	6	30	2000	M2,D3,Y1	2000.7.1
Test2	7	31	1997	M1,D4,Y2	1996.8.1
Test3	2	28	2000	M3,D1,Y1	2000.2.29
Test4	2	29	2000	M3,D2,Y1	2000.3.1

NextDate 函数的强标准等价类测试用例如表 4-6 所示。

表 4-6　　测试用例

编号	测试用例(month,day,year)			覆盖的等价类	预期输出
Test1	7	15	2000	M1,D1,Y1	2000.7.16
Test2	7	15	2002	M1,D1,Y2	2002.7.16
Test3	7	29	2000	M1,D2,Y1	2000.7.30
Test4	7	29	2002	M1,D2,Y2	2002.7.30
Test5	7	30	2000	M1,D3,Y1	2000.7.31
Test6	7	30	2002	M1,D3,Y2	2002.7.31
Test7	7	31	2000	M1,D4,Y1	2000.8.1
Test8	7	31	2002	M1,D4,Y2	2002.8.1
Test9	6	15	2000	M2,D1,Y1	2000.6.16
Test10	6	15	2002	M2,D1,Y2	2002.6.16
Test11	6	29	2000	M2,D2,Y1	2000.6.30
Test12	6	29	2002	M2,D2,Y2	2002.6.30
Test13	6	30	2000	M2,D3,Y1	2000.7.1
Test14	6	30	2002	M2,D3,Y2	2002.7.1
Test15	6	31	2000	M2,D4,Y1	无效输入日期
Test16	6	31	2002	M2,D4,Y2	无效输入日期
Test17	2	15	2000	M3,D1,Y1	2000.2.16
Test18	2	15	2002	M3,D1,Y2	2002.2.16
Test19	2	29	2000	M3,D2,Y1	2000.3.1
Test20	2	29	2002	M3,D2,Y2	无效输入日期
Test21	2	30	2000	M3,D3,Y1	无效输入日期
Test22	2	30	2002	M3,D3,Y2	无效输入日期
Test23	2	31	2000	M3,D4,Y1	无效输入日期
Test24	2	31	2002	M3,D4,Y2	无效输入日期

4.3 边界值分析法

假设有两个变量 x 和 y 的程序 F(x,y)＝x＋y,x、y 在下列范围内取值:10≤x≤100,20≤y≤200,区间[10,100]和[20,200]是 x、y 的输入域,程序 F 的输入定义域如图 4-7 所示,即带阴影矩形中的任何点都是程序 F 的有效输入,用边界值分析法设计测试用例。

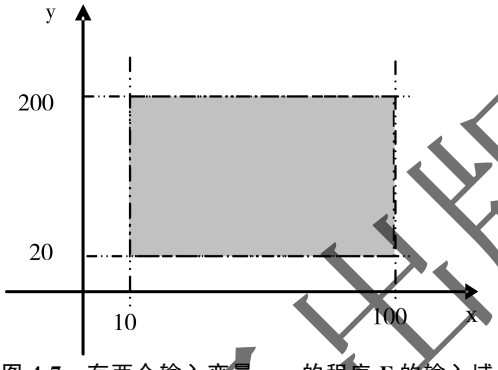

图 4-7　有两个输入变量 x、y 的程序 F 的输入域

4.3.1 边界值分析法的思想

如何用边界值分析法对上述问题进行测试用例设计呢？在设计测试用例之前,我们来看看什么是边界值分析法？边界值分析法是怎样产生的？

无数的测试实践表明,大量的故障往往发生在输入定义域或输出值域的边界上,而不是在其内部。因此,针对各种边界情况设计测试用例,通常会取得很好的测试效果。例如,一个循环条件为“≤”时,却错写成“<”;计数器发生少计数一次等等。边界值法恰恰是针对边界问题进行测试用例设计的一种方法。

边界值分析法是对输入或输出的边界值进行测试的一种黑盒测试方法。通常边界值分析法是作为对等价类划分法的补充,这种情况下,其测试用例来自等价类的边界。

本教材所讲的边界值分析法是基于可靠性理论中称为“单故障”的假设,即有两个或两个以上故障同时出现而导致软件失效的情况很少,也就是说软件失效基本上是由单故障引起的。

4.3.2 边界值分析法的原则

(1)如果输入条件规定了值的范围,则应取刚达到这个范围边界的值,以及刚刚超越这个范围边界的值作为测试输入数据。

例如,如果程序的规格说明中规定:“重量在 10 公斤至 50 公斤范围内的邮件,其邮费计算公式为……”。作为测试用例,我们应取 10 及 50,还应取 10.01,49.99,9.99 及 50.01 等。

(2)如果输入条件规定了值的个数,则用最大个数、最小个数、比最小个数少 1、比最大个数多 1 的数作为测试数据。

例如，一个输入文件应包括 1～255 个记录，则测试用例可取 1 和 255，还应取 0 及 256 等。

(3)将规则 1 和 2 应用于输出条件，即设计测试用例使输出值达到边界值及其左右的值。

例如，某程序的规格说明要求计算出“每月保险金扣除额为 0 至 1165.25 元”，其测试用例可取 0.00 及 1165.24、还可取 0.01 及 1165.26 等。

再如一程序属于情报检索系统，要求每次“最少显示 1 条、最多显示 4 条情报摘要”，这时我们应考虑的测试用例包括 1 和 4，还应包括 0 和 5 等。

(4)如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。

(5)如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。

(6)分析规格说明书，找出其他可能的边界条件。

4.3.3　标准性测试

有两个输入变量的程序 F 的边界值分析法设计标准性测试的测试用例思想，如图 4-8 所示。

对于一个含有 n 个变量的程序，保留其中一个变量，让其余的变量取正常值，被保留的变量依次取最小值 min、略大于最小值 min＋、略小于最大值 max－、最大值 max，对每个变量都重复进行，最后再取一个所有变量都是正常值 nom 值的情况。这样，对于一个有 n 个变量的程序，边界值分析测试程序会产生 4n＋1 个测试用例。

$<x_{min}, y_{nom}>$；

$<x_{min+}, y_{nom}>$；

$<x_{max}, y_{nom}>$；

$<x_{max-}, y_{nom}>$；

$<x_{nom}, y_{min}>$；

$<x_{nom}, y_{min+}>$；

$<x_{nom}, y_{max}>$；

$<x_{nom}, y_{max-}>$；

$<x_{nom}, y_{nom}>$；

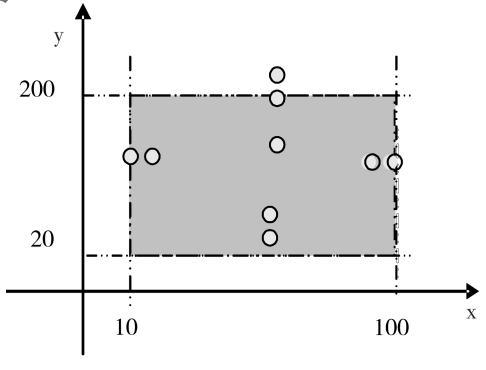

图 4-8　有两个输入变量的程序 F 的标准性测试用例

4.3.4　健壮性测试

有两个输入变量的程序 F 的边界值分析法设计健壮性测试的测试用例思想，如图 4-9 所示。

边界值分析测试的一种扩展，除了取 5 个边界值外，还需要考虑采用一个略超过最大值(max＋)及略小于最小值(min－)的取值，检查超过极限值时系统的情况，健壮性测试最有意义的部分不是输入，而是预期的输出。

对于一个含有 n 个变量的程序，保留其中一个变量，让其余的变量取正常值，被保留的变量

依次取最小值 min、略大于最小值 min＋、略小于最小值 min－、略小于最大值 max－、最大值 max，略大于最大值 max＋，对每个变量都重复进行，最后再取一个所有变量都是 nom 值的情况。这样，对于一个有 n 个变量的程序，边界值分析测试程序会产生6n＋1个测试用例。

$<x_{min}, y_{nom}>$；
$<x_{min-}, y_{nom}>$；
$<x_{min+}, y_{nom}>$；
$<x_{max}, y_{nom}>$；
$<x_{max-}, y_{nom}>$；
$<x_{max+}, y_{nom}>$；
$<x_{nom}, y_{min}>$；
$<x_{nom}, y_{min-}>$；
$<x_{nom}, y_{min+}>$；
$<x_{nom}, y_{max}>$；
$<x_{nom}, y_{max-}>$；
$<x_{nom}, y_{max+}>$；
$<x_{nom}, y_{nom}>$；

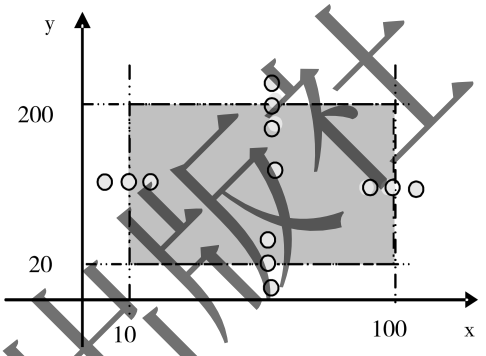

图 4-9　有两个输入变量的程序 F 的健壮性测试用例

4.3.5　最坏情况测试

边界值分析采用可靠性理论中的单缺陷假设，如果不考虑这种假设，那么，应该关心当多个变量取极值时会出现什么情况。

使用这种思想生成最坏情况的测试用例，首先对每个变量进行包含最小值 min，略大于最小值 min＋，正常值 nom，略小于最大值 max－和最大值 max 五个元素集合的测试，然后对这些集合进行笛卡儿积计算，以生成测试用例。

最坏情况测试比边界值分析测试以及健壮性测试要彻底，边界值分析测试用例是最坏情况测试用例的真子集。有 n 个变量的函数的最坏情况测试，会产生 5^n 个测试用例，而边界值分析只产生 4n＋1 个测试用例。有两个输入变量程序 F 的最坏情况测试如图 4-10 所示。

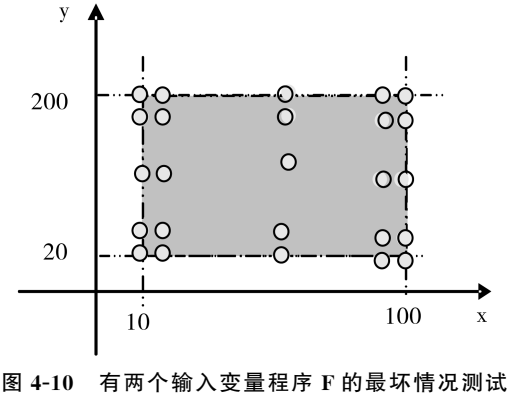

图 4-10　有两个输入变量程序 F 的最坏情况测试

4.3.6　健壮最坏情况测试

首先对每个变量进行包含略小于最小值 min－，最小值 min，略大于最小值 min＋，正常值 nom，略小于最大值 max－，最大值 max 和略大于最大值 max＋七个元素集合的测试，然后对这些集合进行笛卡儿积计算，以生成测试用例。有两个输入变量程序 F 的健壮最坏情况测试如图 4-11 所示。

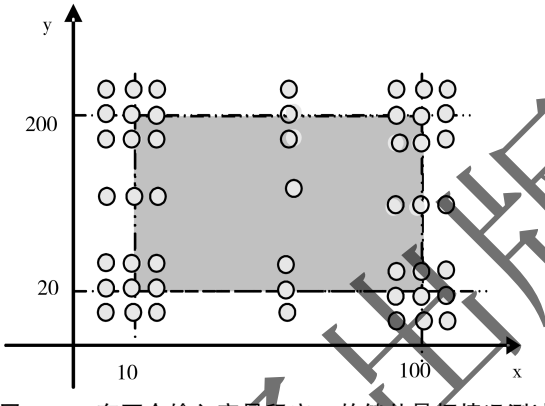

图 4-11　有两个输入变量程序 F 的健壮最坏情况测试

4.3.7　边界值分析法设计测试用例

针对本节开始提出的问题，我们来用边界值分析法设计测试用例。

1.标准性测试

对于变量 x 来说，x 的最小边界为 10，最大边界为 100；即 $x_{min}=10$，$x_{min+}=11$，$x_{max}=100$，$x_{max-}=99$。对于变量 y 来说，y 的最小边界为 20，最大边界为 200；即 $y_{min}=20$，$y_{min+}=21$，$y_{max}=200$，$y_{max-}=199$。

表 4-7 为标准性测试的测试用例。

表 4-7　　测试用例

编号	测试用例(x,y)	预期输出
Test1	(6,20)	26
Test2	(6,21)	27
Test3	(6,200)	206
Test4	(6,199)	205
Test5	(10,108)	118
Test6	(11,108)	119
Test7	(100,108)	208
Test8	(99,108)	207
Test9	(5,119)	224

2.健壮性测试

对于变量 x 来说，x 的最小边界为 10，最大边界为 100；即 $x_{min}=10$，$x_{min+}=11$，$x_{min-}=9$，$x_{max}=100$，$x_{max+}=101$，$x_{max-}=99$。对于变量 y 来说，y 的最小边界为 20，最大边界为 200；即 $y_{min}=20$，$y_{min+}=21$，$y_{min-}=19$，$y_{max}=200$，$y_{max}=200$，$y_{max+}=201$，$y_{max-}=199$。

表 4-8 为健壮性测试的测试用例。

表 4-8　　测试用例

编号	测试用例(x,y)	预期输出
Test1	(6,20)	26
Test2	(6,21)	27
Test3	(6,19)	输入数据不合法
Test4	(6,200)	206
Test5	(6,201)	输入数据不合法
Test6	(6,199)	205
Test7	(10,108)	118
Test8	(11,108)	119
Test9	(9,118)	输入数据不合法
Test10	(100,108)	208
Test11	(101,108)	输入数据不合法
Test12	(99,108)	207
Test13	(5,119)	224

4.3.8　基本应用

NextDate 函数有三个变量 month，day，year 的函数，输出为输入日期下一天的日期。如：输入为 2007 年 7 月 19 日，输出为 2007 年 7 月 20 日。要求三个变量都为整数，且满足：

条件 1：1 ≤month ≤12；

条件 2：1 ≤day ≤31；

条件 3：1912 ≤year ≤2050。

对于 NextDate 函数问题如何设计测试用例？采用什么方法？

思考：

(1)项目的输入条件为 month、day、year；

(2)各个条件的输入域为：条件 1：1 ≤month ≤12；

条件 2 ：1 ≤day ≤31；

条件 3：1912 ≤year ≤2050；

month 的最小边界为 1、最大边界为 12；day 的最小边界为 1、最大边界为 31；year 的最小边界为 1912、最大边界为 2050。

(3)根据条件的输入域确定各个条件的最大边界、最小边界。

NextDate 函数的边界值分析法设计测试用例如表 4-9 所示。

表 4-9　　测试用例

测试用例	month	day	year	预期输出
Test1	6	15	1911	year 超出[1912,2050]
Test2	6	15	1912	1912.6.16
Test3	6	15	1913	1913.6.16
Test4	6	15	1975	1975.6.16
Test5	6	15	2049	2049.6.16
Test6	6	15	2050	2050.6.16
Test7	6	15	2051	year 超出[1912,2050]
Test8	6	0	2001	day 超出[1,31]
Test9	6	1	2001	2001.6.2
Test10	6	2	2001	2001.6.3
Test11	6	30	2001	2001.7.1
Test12	6	31	2001	输入日期超界
Test13	6	32	2001	day 超出[1,31]
Test14	−1	15	2001	month 超出[1,12]
Test15	1	15	2001	2001.1.16
Test16	2	15	2001	2001.2.16
Test17	11	15	2001	2001.11.16
Test18	12	15	2001	2001.12.16
Test19	14	15	2001	month 超出[1,12]

4.4　决策表法

现有一“阅读指南”程序，可以根据读者的当前阅读情况，为读者做出是否继续阅读的决定。如果读者感到疲倦则休息；如果读者不感到疲倦并且很感兴趣，在此情况下，如果读者糊涂则重新读一遍，如果读者不糊涂则继续阅读下一部分；如果读者不感到疲倦但对阅读内容不感兴趣，在此情况下，不管读者是否糊涂都跳到下一章继续学习。

根据以上描述，用决策表法进行测试用例设计。

4.4.1　决策表法的思想

在一些数据处理当中，某些操作的实施依赖于多个逻辑条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。决策表很适合于处理这类问题，它能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用决策表能够设计出完整的测试用例集合。决策表是分析和表达多逻辑条件下执行不同操作情况的工具。

决策表通常由以下 4 部分组成，如图 4-12 所示。

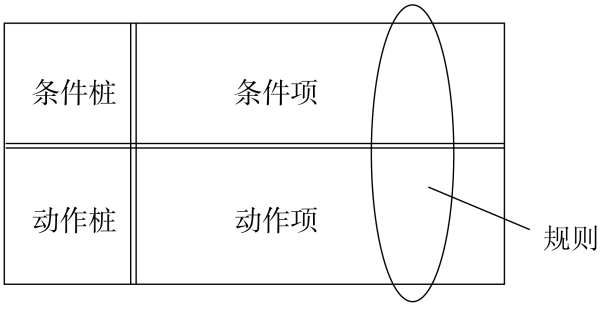

图 4-12　决策表的基本组成

对图 4-12 的说明如下：

(1)条件桩(Condition Stub)：列出问题的所有条件；

(2)条件项(Condition Entry)：针对条件桩给出的条件列出所有可能的取值；

(3)动作桩(Action Stub)：列出问题规定的可能采取的操作；

(4)动作项(Action Entry)：列出在条件项的各组取值情况下应采取的动作。

其中，将任何一个条件组合的特定取值及相应要执行的动作称为一条规则。在决策表中贯穿条件项和动作项的一列就是一条规则。

决策表最突出的优点是，能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏；利用决策表能够设计出完整的测试用例集合。

4.4.2　决策表法的建立步骤

根据软件规格说明，建立决策表的步骤如下：

(1)确定规则的个数，如果有 n 个条件，每个条件有两个取值，那么就有 2^n 种规则；

(2)列出所有的条件桩和动作桩；

(3)填入条件项；

(4)填入动作项，得到初始决策表；

(5)化简：合并相似规则(相同动作)。

条件桩和动作桩通过程序的规格说明书找到。至于规则的个数的确定，没有固定的公式，具体问题具体分析。通常情况下，对于有 n 个条件的决策表，如果每个条件只有真、假两种取值，则规则的个数为 2n 个；如果决策表中有多个条件，则用每个条件的可能取值数相乘得到规则数，例如，决策表中有 3 个条件，条件一有 2 种取值，条件二有 3 种取值，条件三有 2 种取值，则规则数为 2 * 3 * 2＝12 条。但以上两种方法确定规则数都不是所有问题通用的，只是常用的两种方法。有些时候，这样确定的规则是有冗余的，因此还要合并规则。

如何简化决策表，合并相似规则。若表中有两条以上规则具有相同的动作，并且在条件项之间存在极为相似的关系，便可以合并。合并后的条件项用符号“－”表示，说明执行的动作与该条件的取值无关，称为无关条件。

决策表的简化，如图 4-13 所示。

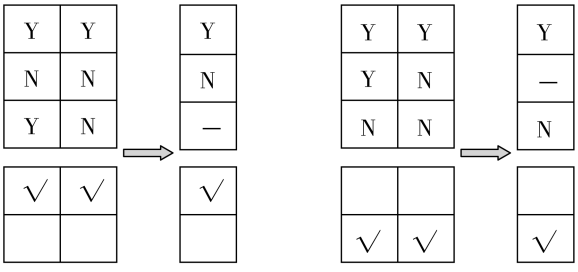

图 4-13　决策表简化示意图

决策表的化简规则如下：

1.合并

如果两个或者多个条件项产生的动作项是相同的，且其条件项对应的每一行的值只有一个是不同的，则可以将其合并；合并的项除了不同值变成无关项外，其余的保持不变。

2.包含

如果两个条件项的动作是相同的，即对任意条件 1 中任意一个值，条件 2 中对应的值满足：

如果条件 1 的值是 Y，则条件 2 中的值也是 Y；

如果条件 1 的值是 N，则条件 2 中的值也是 N；

如果条件 1 的值是—，则条件 2 中的值是 Y、N、—；

那么称条件 1 包含条件 2，此时，条件 2 可以删除。

4.4.3　决策表测试的适用范围

以下情况，适于使用决策表法设计测试用例。

(1)规格说明以决策表形式给出，或较容易转换为决策表；

(2)条件的排列顺序不会也不应影响执行的操作；

(3)规则的排列顺序不会也不应影响执行的操作；

(4)当某一规则的条件已经满足，并确定要执行的操作后，不必检验别的规则；

(5)如果某一规则的条件要执行多个操作任务，这些操作的执行顺序无关紧要；

(6)if-else 逻辑突出；

①恒等：IF A THEN B

②非：IF (NOT A) THEN B

③或：IF (A OR B) THEN C

④与：IF (A AND B) THEN C

(7)输入变量之间存在逻辑关系；

(8)涉及输入变量子集的计算；

(9)输入与输出之间存在因果关系。

4.4.4 决策表法设计测试用例

下面我们针对本节开始提出的问题来构造决策表。构造决策表的步骤如下：

(1)先列出所有的条件桩和动作桩。

条件桩有 3 个， c1:觉得疲倦?

c2:感兴趣吗?

c3:糊涂吗?

动作桩有 4 个， a1:重读

a2:继续

a3:跳下一章

a4:休息

(2)确定规则个数,因为三个条件中,每个条件的可能取值均为真或假,因此规则数为 $2^3=8$ 个,填入条件项及动作项,构造的决策表如表 4-10 所示。

表 4-10　　"阅读指南"决策表

规则选项		1	2	3	4	5	6	7	8
问题	觉得疲倦?	Y	Y	Y	Y	N	N	N	N
	感兴趣吗?	Y	Y	N	N	Y	Y	N	N
	糊涂吗?	Y	N	Y	N	Y	N	Y	N
建议	重读					√			
	继续						√		
	跳下一章							√	√
	休息	√	√	√	√				

(3)简化决策表,合并相似规则。得到简化后的决策表如表 4-11 所示。

表 4-11　　简化后的"阅读指南"决策表

规则选项		1～4	5	6	7～8
问题	觉得疲倦?	Y	N	N	N
	感兴趣吗?	—	Y	Y	N
	糊涂吗?	—	Y	N	—
建议	重读		√		
	继续			√	
	跳下一章				√
	休息	V			

(4)根据简化后的决策表设计测试用例,如表 4-12 所示。

表 4-12　　"阅读指南"的测试用例

Test3	读者不觉得疲倦,也感兴趣,不糊涂	继续
Test4	读者不觉得疲倦,但是不感兴趣	跳下一章
Test5	C2	给出信息 N
Test6	CM	给出信息 M 和 N

4.4.5　应用实例

实例描述:NextDate 函数

NextDate 函数有三个变量 month,day,year 的函数,输出为输入日期下一天的日期。如:输入为 2007 年 7 月 19 日,输出为 2007 年 7 月 20 日。要求三个变量都为整数,且满足:

条件 1:1 ≤month ≤12;

条件 2:1 ≤day ≤31;

条件 3:1912 ≤year ≤2050。

对于 NextDate 函数问题如何设计测试用例?采用什么方法?

NextDate 函数的决策表测试用例设计如下所示:

(1)month 变量的有效等价类:

M1:{month=4,6,9,11}　　M2:{month=1,3,5,7,8,10}

M3:{month=12}　　M4:{month=2}

(2)day 变量的有效等价类:

D1:{1≤day≤27}　　D2:{day=28}　　D3:{day=29}

D4:{day=30}　　D5:{day=31}

(3)year 变量的有效等价类:

Y1:{year 是闰年}　　Y2:{year 不是闰年}

(4)程序中可能采取的操作有以下六种:

a1:不可能　　a2:day+1　　a3:day=1

a4:month+1　　a5:month=1　　a6:year+1

用等价类划分法确定了每个输入条件的值,确定了可能的动作,列出决策表,如表 4-13 所示。

表 4-13　　Nextdate 函数的决策表

规则选项	1	2	3	4	5	6	7	8	9	10	11
条件:c1:month	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M3
c2:day	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1
c3:year	—	—	—	—	—	—	—	—	—	—	—
动作:a1:不可能					√						
a2:day 加 1	√	√	√			√	√	√	√		√
a3:day 复位				√						√	
a4:month 加 1				√						√	
a5:month 复位											
a6:year 加 1											
规则选项	12	13	14	15	16	17	18	19	20	21	22
条件:c1:month	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
c2:day	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
c3:year	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—
动作:a1:不可能									√	√	√
a2:day 加 1	√	√	√		√	√					
a3:day 复位				√			√	√			
a4:month 加 1							√	√			
a5:month 复位				√							
a6:year 加 1				√							

简化 NextDate 函数决策表:

(1)规则 1、2、3 都涉及有 30 天的月份 day 类 D1、D2 和 D3,并且它们的动作项都是 day 加 1,因此可以将规则 1、2、3 合并。

(2)类似地,有 31 天的月份 day 类 D1、D2、D3 和 D4 也可合并,2 月的 D4 和 D5 也可合并。

简化后的 NextDate 函数决策表如表 4-14 所示。

表 4-14　　Nextdate 函数简化后的决策表

选项规则	1～3	4	5	6～9	10	11～14	15	16	17	18	19	20	21～22
条件:c1:month	M1	M1	M1	M2	M2	M3	M3	M4	M4	M4	M4	M4	M4
c2:day	—	D4	D5	—	D5	—	D5	D1	D2	D2	D3	D3	D4、D5
c3:year	—	—	—	—	—	—	—	—	Y1	Y2	Y1	Y2	—
动作:a1:不可能			√									√	√
a2:day 加 1	√			√		√		√	√	√	√		
a3:day 复位		√			√		√			√	√		
a4:month 加 1		√											
a5:month 复位							√						
a6:year 加 1							√						

根据决策表，设计测试用例如表 4-15 所示。

表 4-15　　Nextdate 函数的测试用例

编号	测试用例(month,day,year)			预期输出
Test1～Test3	6	16	2001	17/6/2001
Test4	6	30	2004	1/7/2004
Test5	6	31	2001	不可能
Test6～Test9	8	16	2004	17/8/2004
Test10	8	31	2001	1/9/2001
Test11～Test14	12	16	2004	17/12/2004
Test115	12	31	2001	1/1/2002
Test16	2	16	2004	17/2/2004
Test17	2	28	2004	29/2/2004
Test18	2	28	2001	1/3/2001
Test19	2	29	2004	1/3/2001
Test20	2	29	2001	不可能
Test21～Test22	2	30	2004	不可能

4.5　因果图法

有一个程序，其规格说明要求：输入的第一个字符必须是“#”或“*”，第二个字符必须是一个数字，在此情况下进行文件的修改；如果第一个字符不是“#”或“*”，则给出信息 N；如果第二个字符不是数字，则给出信息 M。

根据以上描述，用因果图进行测试用例设计。

4.5.1　因果图法的思想

等价类划分法和边界值分析方法都是着重考虑输入条件，但没有考虑输入条件的各种组合、输入条件之间的相互制约关系。这样虽然各种输入条件可能出错的情况已经测试到了，但多个输入条件组合起来可能出错的情况却被忽视了。因果图法正适合解决输入条件组合复杂的情况。

因果图是一种利用图解法分析输入的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件的各种组合情况。它的特点主要表现在以下几个方面：

(1)考虑到了输入情况的各种组合以及各个输入情况之间的相互制约关系；

(2)能够帮助测试人员按照一定的步骤，高效率地开发测试用例；

(3)因果图法是将自然语言规格说明转化成形式语言规格说明的一种严格的方法,可以指出规格说明存在的不完整性和二义性。

因果图中出现的基本符号:

通常在因果图中用 Ci 表示原因,用 Ei 表示结果,各结点表示状态,可取值"0"或"1"。"0"表示某状态不出现,"1"表示某状态出现。

在因果图中,原因与结果之间主要有以下几种关系,如图 4-14 所示。

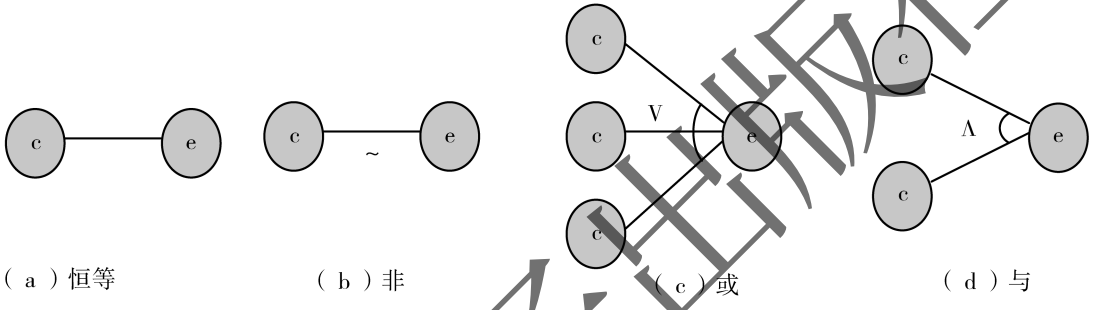

图 4-14　因果图中各种关系示意图

(a)恒等:若 c1 是 1,则 e1 也为 1,否则 e1 为 0;

(b)非:若 c1 是 1,则 e1 为 0,否则 e1 为 1;

(c)或:若 c1 或 c2 或 c3 是 1,则 e1 是 1,否则 e1 为 0,"或"可有任意个输入;

(d)与:若 c1 和 c2 都是 1,则 e1 为 1,否则 e1 为 0,"与"也可有任意个输入。

在实际问题当中输入状态相互之间还可能存在某些依赖关系,称为"约束",如图 4-15 所示。

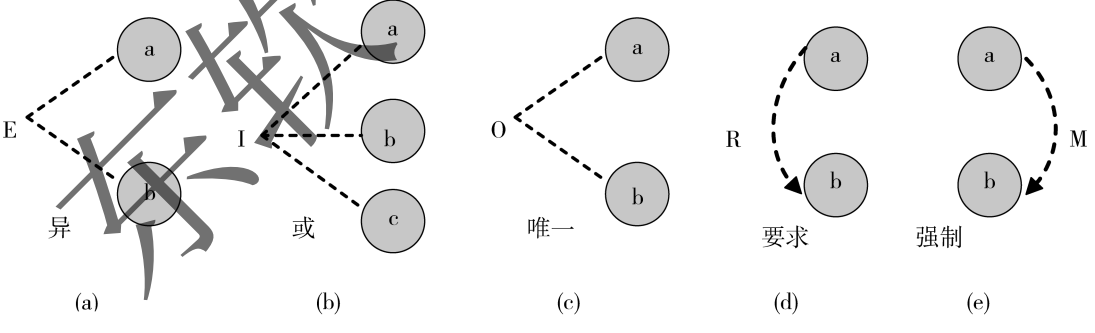

图 4-15　因果图中各种约束示意图

对于输入条件的约束有 4 种:

(a)E 约束(异):a 和 b 中最多有一个可能为 1,即 a 和 b 不能同时为 1;

(b)I 约束(或):a、b、c 中至少有一个必须是 1,即 a、b、c 不能同时为 0;

(c)O 约束(唯一):a 和 b 必须有一个且仅有一个为 1;

(d)R 约束(要求):a 是 1 时,b 必须是 1;

对于输出条件的约束只有 M 约束：

(e)M 约束(强制)：若结果 a 是 1，则结果 b 强制为 0。

4.5.2　因果图法的步骤

用因果图法设计测试用例，首先从程序规格说明书的描述中，找出原因(输入条件)和结果(输出结果或者程序状态的改变)，然后通过因果图转换为判定表，最后为判定表中的每一列设计一个测试用例。具体步骤如下：

(1)分析程序规格说明书描述的语义内容，找出“原因”和“结果”，将其表示成连接各个原因与各个结果的“因果图”；

(2)由于语法或环境限制，有些原因与原因之间或与结果之间的组合情况不能出现，用记号标明约束或限制条件；

(3)将因果图转换成判定表；

(4)根据判定表表中每一列设计测试用例。

4.5.3　因果图法设计测试用例

针对本节开始提出的问题，具体实施过程如下：

(1)根据程序的规格说明书确定原因和结果。

原因：c1——第一个字符是“#”；

　　c2——第一个字符是“*”；

　　c3——第二个字符是一个数字；

结果：a1——给出信息 N；

　　a2——修改文件；

　　a3——给出信息 M；

(2)根据找到的原因和结果，画出因果图，如图 4-16、图 4-17 所示。

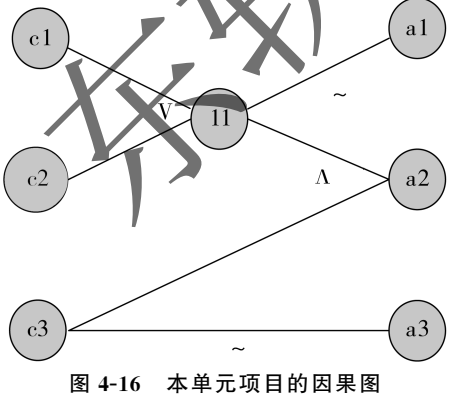

图 4-16　本单元项目的因果图

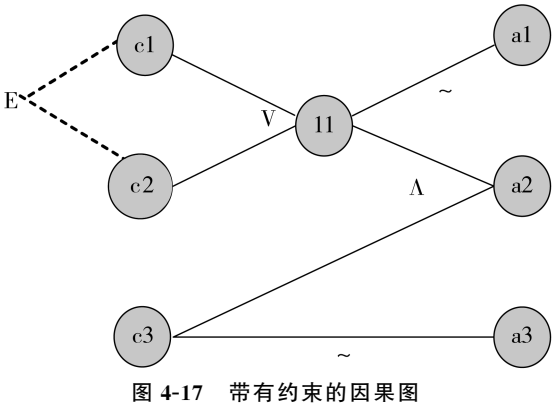

图 4-17　带有约束的因果图

(3)将因果图转化为判定表，如表 4-16 所示。

表 4-16　　单元项目的判定表

规则 选项	1	2	3	4	5	6	7	8
c1	1	1	1	1	0	0	0	0
c2	1	1	0	0	1	1	0	0
c3	1	0	1	0	1	0	1	0
11			1	1	1	1	0	0
a1							√	√
a2			√		√			
a3				√		√		√
不可能	√	√						
测试用例			＃3	＃B	＊7	＊M	C2	CM

(4)根据判定表,设计测试用例。

最左边两列,原因 c1 和 c2 同时为 1 不可能,排除掉,根据判定表可设计出 6 个测试用例,如表 4-17 所示。

表 4-17　　单元项目的测试用例

编号	测试用例	预期输出
Test1	＃3	修改文件
Test2	＃B	给出信息 M
Test3	＊7	修改文件
Test4	＊M	给出信息 M
Test5	C2	给出信息 N
Test6	CM	给出信息 M 和 N

4.6 场景法

下面我们以一个大家常见的案例来讲述场景法。

ATM 机必须能为用户提供以下服务:

(1)用户必须能从 ATM 卡的任一有效账户上提取现金,提取的金额为 50.00 元的整数倍,每次现金支付时,必须得到银行的认可;

(2)用户必须能从 ATM 卡的任一有效账户上存款;

(3)用户必须能在 ATM 卡的任一有效账户之间进行货币转账;

(4)用户必须能查询 ATM 卡的任一有效账户上存款余额;

(5)如果银行确认用户的 PIN 无效,在事务进行之前,要求用户再输入 PIN。如果用户输

入 3 次都不成功，ATM 将永久地保留 ATM 卡，用户必须与银行联系方可取回 ATM 卡；

(6)ATM 机每次交互都通知银行以获得银行的验证；

(7)对于每一个成功的事务处理，ATM 机给用户打印一个收据，提示日期、时间、ATM 机位置、交互类型、账户、数额、转出与转入账户余额；

(8)ATM 机有一个带有钥匙的操作开关面板，安置在银行内部，让银行操作员启动或停止用户服务。

ATM 机的用例图如图 4-18 所示。

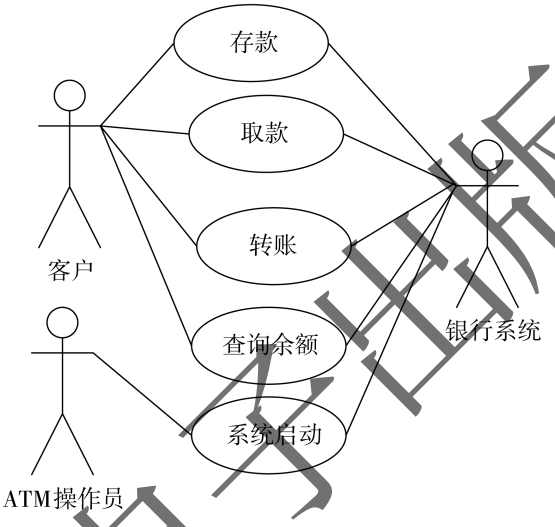

图 4-18　ATM 的用例图

如何采用场景法来对 ATM 机系统进行测试用例设计呢？

4.6.1　场景法的思想

现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。将这种软件设计方面的思想引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。

用例场景用来描述流经用例的路径，从用例开始到结束，遍历这条路径上所有基本流和备选流。场景法测试用例设计就是基于系统的用例场景。

场景法有三个基本概念，分别是基本流、备选流和场景。下面我们通过图示的方式来进行讲解。

图 4-19 中经过用例的每条路径都用基本流和备选流来表示。直黑线表示基本流，是经过用例的最简单的路径。备选流用不同的彩色表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中(如备选流 1 和备选流 3)；也可能起源于另一个备选流(如备选流 2)，或者终止用例而不再重新加入到某个流(如备选流 2 和备选流 4)。

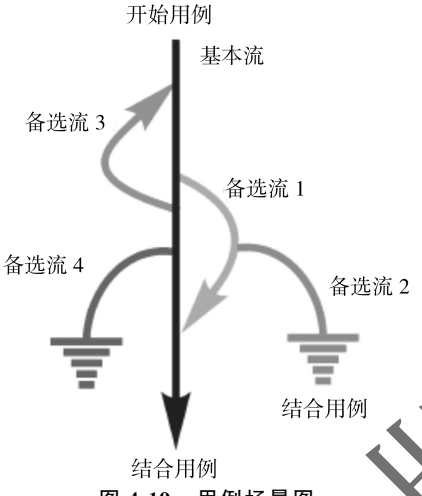

图 4-19　用例场景图

场景如下：

场景 1：基本流；

场景 2：基本流，备选流 1；

场景 3：基本流，备选流 1，备选流 2；

场景 4：基本流，备选流 3；

场景 5：基本流，备选流 3，备选流 1；

场景 6：基本流，备选流 3，备选流 1，备选流 2；

场景 7：基本流，备选流 4；

场景 8：基本流，备选流 3，备选流 4。

4.6.2　场景法的步骤

场景法设计测试用例的步骤如下：

(1)根据说明，描述出程序的基本流及各项备选流；

(2)根据基本流和各项备选流生成不同的场景；

(3)对每一个场景生成相应的测试用例；

(4)对生成的所有测试用例重新复审，去掉多余的测试用例，测试用例确定后，对每一个测试用例确定测试数据值。

4.6.3　场景法设计测试用例

对于 ATM 系统的场景法设计测试用例的设计步骤是：

1.描述程序的基本流程

本用例的开端是 ATM 处于准备就绪状态。

(1)准备提款，客户将银行卡插入 ATM 机的读卡机。

(2)验证银行卡,ATM 机从银行卡的磁条中读取账户代码,并检查它是否属于可以接收的银行卡。

(3)输入 PIN,ATM 要求客户输入 PIN 码(6 位)。

(4)验证账户代码和 PIN,验证账户代码和 PIN 以确定该账户是否有效以及所输入的 PIN 对该账户来说是否正确。对于此事件流,账户是有效的而且 PIN 对此账户来说正确无误。

(5)ATM 选项,ATM 显示在本机上可用的各种选项。在此事件流中,银行客户通常选择“提款”。

(6)输入金额,要从 ATM 中提取的金额。对于此事件流,客户需选择预设的金额(100 元、200 元、500 元或 1000 元)。ATM 通过将卡 ID、PIN、金额以及账户信息作为一笔交易发送给银行系统来启动验证过程。对于此事件流,银行系统处于联机状态,而且对授权请求给予答复,批准完成提款过程,并且据此更新账户余额。

(7)出钞,提供现金。

(8)返回银行卡,银行卡被返还。

(9)收据,打印收据并提供给客户。ATM 相应地更新内部记录。

用例结束时 ATM 又回到准备就绪状态。

2.描述程序的备选流程

备选流 1:银行卡无效。在基本流步骤 2 中,验证银行卡,如果卡是无效的,则卡被退回,同时会通知相关消息。

备选流 2:ATM 内没有现金。在基本流步骤 5 中,ATM 选项,如果 ATM 内没有现金,则“提款”选项将无法使用。

备选流 3:ATM 内现金不足。在基本流步骤 6 中,输入金额,如果 ATM 机内金额少于请求提取的金额,则将显示一则适当的消息,并且在步骤 6 输入金额处重新加入基本流。

备选流 4:PIN 有误。在基本流步骤 4 中,验证账户和 PIN,客户有三次机会输入 PIN。如果 PIN 输入有误,ATM 将显示适当的消息;如果还存在输入机会,则此事件流在步骤 3 输入 PIN 处重新加入基本流。如果最后一次尝试输入的 PIN 码仍然错误,则该卡将被 ATM 机保留,同时 ATM 返回到准备就绪状态,本用例终止。

备选流 5:账户不存在 。在基本流步骤 4 中,验证账户和 PIN,如果银行系统返回的代码表明找不到该账户或禁止从该账户中提款,则 ATM 显示适当的消息并且在步骤 9 返回银行卡处重新加入基本流。

备选流 6:账面金额不足。在基本流步骤 7,授权中,银行系统返回代码表明账户余额少于在基本流步骤 6 输入金额内输入的金额,则 ATM 显示适当的消息并且在步骤 6 输入金额处重新加入基本流。

备选流 7:达到每日最大的提款金额。在基本流步骤 7,授权中,银行系统返回的代码表明包括本提款请求在内,客户已经或将超过在 24 小时内允许提取的最多金额,则 ATM 显示适当的消息并在步骤 6,输入金额上重新加入基本流。

备选流 x:记录错误。如果在基本流步骤 10,收据中,记录无法更新,则 ATM 进入“安全模式”,在此模式下所有功能都将暂停使用。同时向银行系统发送一条适当的警报信息表明 ATM 已经暂停工作。

备选流 y:退出。客户可随时决定终止交易(退出)。交易终止,银行卡随之退出。

备选流 z:“翘起”。ATM 包含大量的传感器,用以监控各种功能,如电源检测器、不同的门和出入口处的测压器以及动作检测器等。在任一时刻,如果某个传感器被激活,则警报信号将发送给警方而且 ATM 进入“安全模式”,在此模式下所有功能都暂停使用,直到采取适当的重启/重新初始化的措施。

3.根据基本流和备选流生成场景

以“取款”用例为例。“取款”用例的事件流:

- 基本流:预设提取金额(100 元、200 元、500 元、1000 元);
- 备选流 2:ATM 内没有现金;
- 备选流 3:ATM 内现金不足;
- 备选流 4:PIN 有误;
- 备选流 5:账户不存在/账户类型有误;
- 备选流 6:账面金额不足。

根据“取款”用例的事件流,生成的场景有:

- 场景 1:成功的取款:基本流;
- 场景 2:ATM 内没有现金:基本流,备选流 2;
- 场景 3:ATM 内现金不足:基本流,备选流 3;
- 场景 4:PIN 有误(还有输入机会):基本流,备选流 4;
- 场景 5:PIN 有误(不再有输入机会):基本流,备选流 3,备选流 4;
- 场景 6:账户不存在/账户类型有误:基本流,备选流 5;
- 场景 7:账户余额不足:基本流,备选流 6。

4.对每一个场景生成对应的测试用例

对于这 7 个场景中的每一个场景都需要确定测试用例。可以采用矩阵或决策表来确定和管理测试用例。下面显示了一种通用格式,其中各行代表各个测试用例,而各列则代表测试用例的信息。本示例中,对于每个测试用例,应有一个测试用例 ID、条件(或说明)、测试用例中涉及的所有数据元素(作为输入或已经存在于数据库中)以及预期结果。

通过从确定执行用例场景所需的数据元素入手构建矩阵。然后,对于每个场景,至少要确定包含执行场景所需的适当条件的测试用例。例如,在表 4-18 的矩阵中,V(有效)用于表明这个条件必须是 VALID(有效的)时才可执行基本流,而 I(无效)用于表明这种条件下将激活所需备选流。表中使用的“n/a”(不适用)表明这个条件不适用于测试用例。

表 4-18　　ATM 的测试用例矩阵表示

(测试用例) ID 号	场景/条件	账户	PIN	选择金额	账面金额	ATM 内金额	预期结果
Test1.	场景 1 成功的提款	V	V	V	V	V	成功的提款
Test 2.	场景 2 ATM 内没有现金	V	V	V	V	I	提款选项不可用,用例结束
Test 3.	场景 3 ATM 内现金不足	V	V	V	V	I	警告消息,返回基本流步骤 6
Test 4.	场景 4 PIN 有误(还有不止一次输入机会)	V	I	n/a	V	V	警告消息,返回基本流步骤 3
Test 5.	场景 4 PIN 有误(还有一次输入机会)	V	I	n/a	V	V	警告消息,返回基本流步骤 3
Test 6.	场景 4 PIN 有误(不再有输入机会)	V	I	n/a	V	V	警告消息,卡予保留,用例结束

4.7　错误推测法

使用边界值分析法和等价类划分法,可以帮助开发人员设计具有代表性的、用以暴露程序错误的测试用例。但是,不同类型、不同特点的程序通常有一些特殊的容易出错的情况。此外,有时分别使用每组测试数据时程序都能正常工作,这些输入数据的组合却可能检测出程序的错误。一般来说,即使是一个比较小的程序,可能的输入组合也往往十分巨大,因此必须依靠测试人员的经验和直觉,从各种可能的测试用例中选出一些最可能引起程序出错的方案。

错误推测法就是基于经验和直觉推测程序中所有可能存在的各种错误,从而有针对性地设计测试用例的方法。它的基本思想是列举出程序中所有可能有的错误和容易发生错误的特殊情况,根据它们选择测试用例。对于程序中可能存在哪类错误的推测,是挑选测试用例时的一个重要因素。例如:在单元测试时曾列出的许多在模块中常见的错误、以前产品测试中曾经发现的错误等,这些就是经验的总结。还有,输入数据和输出数据为 0 的情况、输入表格为空格或输入表格只有一行等,这些都是容易发生错误的情况,可选择这些情况下的例子作为测试用例。

思考题

1. 黑盒测试的主要方法有哪些?

2. 三角形问题:输入三个整数 a、b、c 分别作为三角形的三条边,现通过程序判断这三条边

能否构成三角形，如果能构成，那么构成的是一般三角形、等腰三角形还是等边三角形呢？请用等价类划分法，进行弱健壮性测试，设计测试用例。

3.保险公司计算保费费率的程序：某保险公司的人寿保险的保费计算方式为：投保额×保险费率，其中，保险费率依点数不同而有别，10点及10点以上保险费率为0.6%，10点以下保险费率为0.1%；而点数又是由投保人的年龄、性别、婚姻状况和抚养人数来决定，具体规则如下：

年龄			性别		婚姻		抚养人数
20～39	40～59	其他	M	F	已婚	未婚	1人扣0.5点 最多扣3点(四舍五入取整)
6点	4点	2点	5点	3点	3点	5点	

请用等价类划分法，进行弱健壮性测试，设计测试用例。

4.有二元函数f(x,y)，其中x∈[1,12]，y∈[1,31]，请采用边界值分析法设计测试用例。

5.有函数f(x,y,z)，其中x∈[1900,2100]，y∈[1,12]，z∈[1,31]，请写出该函数采用边界值分析法设计的测试用例。

6.某软件的一个模块的需求规格说明书中描述：

“对于功率大于50马力的机器或者维修记录不全的或已经运行10年以上的机器应予以优先的维修处理……”。这里假定“维修记录不全”和“优先维修处理”有严格的定义。请建立该需求的决策表，并绘制出化简(合并规则)后的决策表，设计测试用例。

7.某软件的一个模块的需求规格说明书中描述：

(1)年薪制员工：严重过失，扣年终风险金的4%；过失，扣年终风险金的2%。

(2)非年薪制员工：严重过失，扣当月薪资的8%；过失，扣当月薪资的4%。

请绘制出因果图和判定表，并给出相应的测试用例。