

黑盒测试技术

本章学习目标

- 了解黑盒测试基本知识。
- 熟练掌握等价类划分法设计测试用例。
- 熟练掌握边界值分析法设计测试用例。
- 熟练掌握错误猜测法设计测试用例。
- 熟练掌握因果图法设计测试用例。
- 熟练掌握判定表驱动法设计测试用例。
- 熟练掌握场景法设计测试用例。
- 熟练掌握正交试验法设计测试用例。

本章首先介绍黑盒测试的定义和分类,再介绍常用黑盒测试方法——等价类划分、边界值法、错误猜测法、因果图、判定表驱动法、场景法、正交试验法的基本理论及测试用例的设计过程。

3.1 黑盒测试概述

对于黑盒测试技术,需要了解以下内容:

- 黑盒技术的定义和常用的黑盒测试方法。

黑盒测试又称为功能测试,即将软件看成黑盒子,在完全不考虑软件内部结构和特性的情况下,测试软件的外部特性。黑盒测试是一种从用户观点出发,基于规格说明的测试方法,主要验证软件的功能需求和用户最终需求。黑盒测试技术有很多测试方法,常用的有等价类划分、边界值法、错误猜测法、因果图、判定表驱动法、场景法、正交试验法等。

从理论上讲,黑盒测试只有采用穷举输入测试,把输入域里所有可能的输入数据都作为测试情况考虑,才能查出程序中所有的错误。实际上测试情况有无穷多个,不仅要测试所有合法的输入,而且还要对那些不合法但可能的输入进行测试,测试量是非常庞大的,因此完全测试是不可能的,所以要进行有针对性的测试。本章介绍了多种测试方法,其目的是在庞大的输入域中合理地设计有代表性的测试用例,以进行高效的测试。

3.2 等价类划分

为了熟练掌握黑盒测试技术的等价类划分法,需要学习以下内容:

- 认识什么是等价类。
- 等价类概念和等价类的分类。
- 划分等价类的方法。
- 等价类划分法设计测试用例实例分析。

3.2.1 认识等价类

假设 GradeRecord 函数能根据输入的单科成绩,把低于 60 分的成绩登记为“不及格”,大于等于 60 分的成绩按实际成绩登记。

单科成绩的输入域: {单科成绩: $0 \leqslant \text{单科成绩} \leqslant 100$ }。根据题意,当 $0 \leqslant \text{单科成绩} < 60$ 时,程序所做的处理是: 将单科成绩登记为“不及格”; 当 $60 \leqslant \text{单科成绩} \leqslant 100$ 时,程序所做的处理是: 按实际输入的单科成绩登记。在测试时,只要在 $0 \leqslant \text{单科成绩} < 60$ 、 $60 \leqslant \text{单科成绩} \leqslant 100$ 中各选择一个任意的输入数据就可以了,而没有必要将 $0 \leqslant \text{单科成绩} \leqslant 100$ 所有的数据都测试一遍。

$0 \leqslant \text{单科成绩} < 60$ 、 $60 \leqslant \text{单科成绩} \leqslant 100$ 为 $0 \leqslant \text{单科成绩} \leqslant 100$ 的子集,这两个子集互不相交,每个子集里的所有数据都执行相同的处理,相同的处理映射到程序相同的执行路径。因此每个子集中的所有数据都有相同的执行路径,所以只要在每个子集选择一个有代表性的数据就可以把其处理功能测试了。

$0 \leqslant \text{单科成绩} < 60$ 、 $60 \leqslant \text{单科成绩} \leqslant 100$ 就是 GradeRecord 函数测试的两个等价类。

3.2.2 等价类划分概述

等价类划分法是一种典型的、常用的黑盒测试方法。

在做黑盒测试时,需要通过接口把输入域中的数据输入,然后观察输出结果,如果输出结果和预期结果不一致,表示被测试的功能有错误。通常输入域是一个庞大的数据集合,试图用穷举法把输入域中的所有输入数据做上述测试是不可能的,也没有必要。

等价类划分就是解决如何选择适当的数据子集来代表整个数据集的问题,通过降低测试的数目去实现合理的覆盖,覆盖更多的可能数据,以发现更多的软件缺陷。

所谓等价类划分,就是把输入数据(有效的和无效的)的所有可能值划分为若干等价类,使每个等价类中任何一个测试用例都能代表同一等价类中的其他测试用例。划分了等价类,就可以从每个等价类中选取任意的或具有代表性的数据当作测试用例进行测试。

使用等价类划分法进行测试时完全不需要考虑程序的内部结构,只根据需求规格说明书来分析程序的输入域,按照一定的规则把程序的输入域划分为若干合理的互不相交

的子域,然后从这些子域中选择任意的或有代表性的数据作为测试用例。

为了保证测试用例的完整性和代表性。测试用例由有效等价类和无效等价类的代表组成。

1. 有效等价类

有效等价类是指对于程序规格说明来说合理的、有意义的输入数据构成的集合。利用有效等价类可以检验程序是否实现了规格说明预先规定的功能和性能。有效等价类可以是一个,也可以是多个。把软件系统的输入域按一定的规则划分为若干子域(有效等价类),然后从每个子域(有效等价类)中选取少数任意或有代表性的数据作为测试用例的输入数据。

2. 无效等价类

无效等价类和有效等价类相反,无效等价类是指对程序的规格说明来说所有的不合理的或无意义的输入数据所构成的集合。对于具体的问题,无效等价类至少应有一个,也可能有多个。利用无效等价类,可以检查程序对各种异常输入的处理。

设计测试用例时,要同时考虑这两种等价类。因为软件不仅要能接收合理的、有意义的数据,也要能对意外不合理、无意义数据的处理能力,这样才能确保软件具有更高的可靠性。

3.2.3 划分等价类的方法

1. 划分等价类

如何确定等价类是使用等价类划分法的重要问题。等价类的划分方法如下。

1) 按区间划分

如果可能的输入数据属于一个取值范围,则可以确定一个有效等价类和两个无效等价类。

例如,输入数据是学生的单科成绩,单科成绩的范围是 $0 \sim 100$,可划分为“ $0 \leqslant$ 单科成绩 $\leqslant 100$ ”一个有效等价类(有效成绩)和“单科成绩 < 0 ”、“单科成绩 > 100 ”两个无效等价类(无效成绩)。

2) 按数值划分

如果规定了输入数据的一组值,并且程序要对每个输入值分别进行不同的处理,则可为每个值确定一个有效等价类,此外针对这组值确立一个无效等价类,它是这组值之外的所有数据的集合。

例如,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 $\{x : x \neq 1, 3, 7, 15\}$ 。

3) 按数值集合划分

如果可能的输入数据属于一个值的集合(假定 n 个),并且程序对该集合的每个输入

值进行不同的处理,这时可确定 n 个有效等价类和一个无效等价类。

4) 按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定条件下,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)。

例如,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。

5) 按布尔值划分

如果输入条件是一个布尔量,可确定一个有效等价类和一个无效等价类。

6) 按处理方式划分

在确定已划分的等价类中各元素在程序处理中方式不同的情况下,则应再将该等价类进一步划分为更小的等价类。

在划分了等价类之后,需要建立等价类表,列出所有划分出的等价类,如表 3-1 所示。

表 3-1 等价类表

输入数据(或输入条件)	有效等价类	无效等价类

2. 等价类划分法设计测试用例的原则

为等价类表中所有的有效等价类和无效等价类设计测试用例的原则如下:

- 为每个等价类规定一个唯一的编号。
- 设计一个新的测试用例,使其尽可能多地覆盖尚未覆盖的有效等价类;重复这一步骤,直到所有的有效等价类都被覆盖为止。
- 设计一个新的测试用例,使其覆盖且仅覆盖一个无效等价类,重复这一步骤,直到所有的无效等价类都被覆盖为止。

3.2.4 等价类划分法实例

【例 3-1】 假设 GradeRecord 函数能根据输入的单科成绩,把低于 60 分的成绩登记为“不及格”,大于等于 60 分的成绩按实际成绩登记。试用等价类划分法为其设计测试用例。

解:第一步:划分等价类。

GradeRecord 函数的输入变量是 Grade(单科成绩),可将其输入域划分成 $0 \leqslant \text{Grade} < 60$ 、 $60 \leqslant \text{Grade} \leqslant 100$ 两个有效等价类,另外还需考虑各种情况的无效等价类,GradeRecord 函数划分的等价类,如表 3-2 所示。

表 3-2 GradeRecord 函数的等价类表

输入数据(或输入条件)	有效等价类	无效等价类
单科成绩	为数字 (1)	有非数字字符 (4)
	$0 \leqslant \text{Grade} < 60$ (2)	$\text{Grade} < 0$ (5)
	$60 \leqslant \text{Grade} \leqslant 100$ (3)	$\text{Grade} > 100$ (6)

第二步：为有效等价类设计测试用例。

为 GradeRecord 函数有效等价类设计的测试用例如表 3-3 所示。

表 3-3 GradeRecord 函数覆盖有效等价类的测试用例

测试用例	输入数据	预期输出	测试范围
Test Case 1	47	不及格	(1)、(2)
Test Case 2	80	登记成绩 80 分	(1)、(3)

第三步：为无效等价类设计测试用例。

为 GradeRecord 函数无效等价类设计的测试用例如表 3-4 所示。

表 3-4 GradeRecord 函数覆盖无效等价类的测试用例

测试用例	输入数据	预期输出	测试范围
Test Case 1	R6	非法成绩	(4)
Test Case 2	-20	非法成绩	(5)
Test Case 3	124	非法成绩	(6)

【例 3-2】 某公司招聘销售人员, 规定报名者为 1975 年 1 月 1 日到 1995 年 1 月 1 日期间出生, 若出生日期为 1995 年 1 月 1 日之后, 将显示“对不起, 您的年龄太小了!!”并拒绝接受; 若出生日期在 1975 年 1 月 1 日之前, 将显示“对不起, 您的年龄超出了!!”并拒绝接受。用等价类划分法为此功能设计测试用例。

解: 第一步: 划分等价类。

假设出生年月日由 8 位数字字符表示, 前 4 位为年, 5、6 位为月, 7、8 位为日。可划分 4 个有效等价类和 9 个无效等价类, 如表 3-5 所示。

表 3-5 报名功能的等价类表

输入数据(或输入条件)	有效等价类	无效等价类
出生年月日	8 位有效数字字符 (1)	有非数字字符 (5) 位数小于 8 (6) 位数大于 8 (7)
数值范围	在 19750101~19950101 之间 (2)	> 19950101 (8) < 19750101 (9)
月的范围	在 1~12 之间 (3)	< 1 (10) > 12 (11)
日的范围	在 1~31 之间 (4)	< 1 (12) > 31 (13)

第二步：为有效等价类设计测试用例。

为报名功能的有效等价类设计测试用例，如表 3-6 所示。Test Case 1 测试用例就可以覆盖编号为(1)、(2)、(3)、(4)的有效等价类。

表 3-6 报名功能覆盖有效等价类的测试用例

测试用例	输入数据	预期输出	测试范围
Test Case 1	19860815	接受报名	(1)、(2)、(3)、(4)

第三步：为无效等价类设计测试用例。

为报名功能无效等价类设计测试用例，如表 3-7 所示。

表 3-7 报名功能覆盖无效等价类的测试用例

测试用例	输入数据	预期输出	测试范围
Test Case 1	19r41215	输入错误提示	(5)
Test Case 2	199414	输入错误提示	(6)
Test Case 3	1992100556	输入错误提示	(7)
Test Case 4	19981029	对不起,您的年龄太小了!!	(8)
Test Case 5	19700528	对不起,您的年龄超出了!!	(9)
Test Case 6	19800013	输入错误提示	(10)
Test Case 7	19981416	输入错误提示	(11)
Test Case 8	19770200	输入错误提示	(12)
Test Case 9	19920533	输入错误提示	(13)

【例 3-3】 输入 3 个整数 a、b、c 作为三角形的 3 条边。判断所构成的三角形是什么类型的三角形。如果是一般三角形，输出其周长；如果是等边三角形，输出其面积；如果是等腰三角形，输出其 3 个内角。试用等价类划分法为该程序的三角形判断及计算部分进行测试用例设计。

解：第一步：划分等价类。

根据题意，本题可能进行 3 种不同的处理：

- (1) 如果 a、b、c 三边构成一般三角形，输出其周长。
- (2) 如果 a、b、c 三边构成等边三角形，输出其面积。
- (3) 如果 a、b、c 三边构成等腰三角形，输出其 3 个内角。

以上 3 种不同的处理所对应的输入条件如下：

一般三角形的输入条件： $a+b>c; a+c>b; b+c>a$ 。

等边三角形的输入条件： $a=b=c$ 。

等腰三角形的输入条件： $a=b; a=c; b=c$ ；

另外，在等价类划分时还要考虑输入各种可能发生的情况。根据以上的分析和考虑该三角形三边 a、b、c 的等价类划分如表 3-8 所示。

表 3-8 三角形的等价类表

输入数据(或输入条件)	有效等价类	无效等价类
构成三角形的条件	a、b、c 为数字(1)	一条边为非数字(12)、(13)、(14)
		两条边为非数字(15)、(16)、(17)
		三条边为非数字(18)
	a、b、c 三条边(2)	只输入一条边(19)、(20)、(21)
		只输入两条边(22)、(23)、(24)
		输入三个以上的边(25)
	a、b、c 为非零数(3)	一条边为 0(26)、(27)、(28)
		两条边为 0(29)、(30)、(31)
		三条边为 0(32)
	a、b、c 为正数(4)	一条边小于 0(33)、(34)、(35)
		两条边小于 0(36)、(37)、(38)
		三条边小于 0(39)
是否为等腰三角形	$a+b > c$ (5)	$a+b < c$ (40)
		$a+b = c$ (41)
	$a+c > b$ (6)	$a+c < b$ (42)
		$a+c = b$ (43)
	$b+c > a$ (7)	$b+c < a$ (44)
		$b+c = a$ (45)
是否为等边三角形	$a=b$ (8)	
	$b=c$ (9)	
	$c=a$ (10)	
是否为等边三角形	$a=b=c$ (11)	

第二步：为有效等价类设计测试用例。

为三角形问题的有效等价类设计测试用例，如表 3-9 所示。Test Case 1 测试用例可以覆盖编号为(1)、(2)、(3)、(4)、(6)、(7)的有效等价类。

表 3-9 三角形问题覆盖有效等价类的测试用例

测试用例	输入数据			预期输出	测试范围
	a	b	c		
Test Case 1	3	4	5	输出其周长	(1)(2)(3)(4)(5)(6)(7)
Test Case 2	10	10	18	输出其三个内角	(1)(2)(3)(4)(5)(6)(7)(8)
Test Case 3	6	4	4	输出其三个内角	(1)(2)(3)(4)(5)(6)(7)(9)
Test Case 4	52	36	52	输出其三个内角	(1)(2)(3)(4)(5)(6)(7)(10)
Test Case 5	50	50	50	输出其面积	(1)(2)(3)(4)(5)(6)(7)(11)

第三步：为无效等价类设计测试用例。

为三角形问题的无效等价类设计测试用例,如表 3-10 所示。

表 3-10 三角形问题覆盖无效等价类的测试用例

测试用例	输入数据				测试范围	测试用例	输入数据			测试范围
	a	b	c	d			a	b	c	
Test Case 1	L	4	5		(12)	Test Case 18	0	0	20	(29)
Test Case 2	10	h	18		(13)	Test Case 19	20	0	0	(30)
Test Case 3	6	4	u		(14)	Test Case 20	0	20	0	(31)
Test Case 4	e	r	52		(15)	Test Case 21	0	0	0	(32)
Test Case 5	50	r	e		(16)	Test Case 22	-5	3	4	(33)
Test Case 6	r	50	e		(17)	Test Case 23	3	-5	4	(34)
Test Case 7	t	r	e		(18)	Test Case 24	3	4	-5	(35)
Test Case 8	50				(19)	Test Case 25	-3	-4	5	(36)
Test Case 9		50			(20)	Test Case 26	3	-4	-5	(37)
Test Case 10			50		(21)	Test Case 27	-3	4	-5	(38)
Test Case 11	48	36			(22)	Test Case 28	-2	-3	-4	(39)
Test Case 12		48	36		(23)	Test Case 29	2	3	6	(40)
Test Case 13	48		36		(24)	Test Case 30	2	4	6	(41)
Test Case 14*	5	3	4	6	(25)	Test Case 31	3	6	2	(42)
Test Case 15	0	8	12		(26)	Test Case 32	3	5	2	(43)
Test Case 16	8	0	12		(27)	Test Case 33	6	2	3	(44)
Test Case 17	8	12	0		(28)	Test Case 34	5	3	2	(45)

* Test Case 14 是一个无效等价类的测试用例,用来测试在错误地输入 4 个数据时程序的输出。

3.3 边界值分析

为了熟练掌握黑盒测试技术的边界值分析方法,需要学习以下内容:

- 边界值的确定及设计测试用例的原则。
- 边界值分析法设计测试用例实例分析。

3.3.1 边界值分析概述

使用等价类划分法时,在每个等价类中选择一个有代表性的数据作为测试用例,由于该数据可以代表同一等价类的其他数据,等价类划分法做到了用最少的数据覆盖更多的可能数据。但是对于容易出错的边界,等价类划分法却显得不那么有效。

根据长期的测试工作经验发现,大量的错误是发生在输入或输出范围的边界上,而不是发生在输入输出范围的内部,因此针对各种边界情况设计测试用例,可以发现更多的错误。

边界值分析法就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。

使用边界值分析法设计测试用例,首先应确定边界情况。边界值和等价类密切相关,通常是输入等价类与输出等价类的边界。边界值分析法应着重测试的边界情况,应选取正好等于、刚刚大于或刚刚小于边界的值作为测试数据,而不是选取等价类中有代表性的值或任意值作为测试数据。

1. 边界值分析法与等价类划分法的区别

边界值分析法与等价类划分法的区别如下:

(1) 边界值分析法不是从某等价类中选择一个任意的或有代表性的数据,而是利用这个等价类的每个边界作为测试条件(或测试数据)。

(2) 边界值分析法不仅要考虑输入条件的边界,还要考虑输出域的边界。

通常情况下,软件测试所包含的边界检验有数字、字符、位置、质量、大小、速度、尺寸、空间等几种类型。

以上类型的边界值应该是最大数/最小数、首位/末位、最上/最下、最优/最劣、最大/最小、最快/最慢、最长/最短、满/空等情况。例如:

- (1) 对 16 位的整数而言 32 767 和 -32 768 是边界。
- (2) 屏幕上光标在最左上、最右下位置是边界。
- (3) 报表的第一行和最后一行是边界。
- (4) 数组的第一个下标和最后一个下标是边界。
- (5) 循环的第一次和最后一次是边界。
- (6) 利用边界值作为测试数据。

边界值分析的基本思想是使用输入数据的最小值(min)、略大于最小值(min+)、略小于最小值(min-)、正常值、略小于最大值(max-)、最大值(max)、略大于最大值(max+)处设计测试用例。

2. 边界值分析法设计测试用例的原则

(1) 如果输入数据(或输入条件)规定了取值范围,则应取等于边界及刚刚超出边界的值作为测试用例。

(2) 如果输入数据(或输入条件)规定了输入数据的个数,则应取最大个数、最小个数、比最小个数少一、比最大个数多一的输入数据为测试用例。

例如,一个输入文件里有 255 个记录,则测试用例可取第 1 个、第 255 个记录,还应取 0 及 256。

(3) 将规则(1)和(2)应用于输出数据,针对输出数据设计测试用例。

例如,某程序的规格说明要求计算出“每月扣除保险金为 0~1165.25 元”,其测试用例可取 -0.01、0.00、0.01、1165.24、1165.25、1165.26。

再如,某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”,这时应考虑的测试用例包括 0 条、1 条、4 条、5 条。

(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,则应当考虑内部数据结构的边界来设计测试用例。

(6) 分析规格说明,找出其他可能的边界条件。

3.3.2 边界值分析法实例

针对容易出错的边界,使用边界值分析法为 3.2.4 节例 3-1、例 3-2 补充测试用例。

【例 3-4】 假设 GradeRecord 函数能根据输入的单科成绩,把低于 60 分的成绩登记为“不及格”,大于等于 60 分的成绩按实际成绩登记。用边界值分析法为例 3-1 补充测试用例。

解: 在等价类划分法中将 Grade 变量的输入域划分为 $0 \leqslant \text{Grade} < 60$ 和 $60 \leqslant \text{Grade} \leqslant 100$ 两个有效等价类,如果其中的 \leqslant 错写成 $<$,这类错误在等价类划分法中非常容易被忽略。对于这类错误使用边界值分析法最为有效。

通过对 $0 \leqslant \text{Grade} < 60$ 和 $60 \leqslant \text{Grade} \leqslant 100$ 这两个有效等价类的边界进行分析,补充设计如表 3-11 所示的测试用例。

表 3-11 边界值分析法为 GradeRecord 函数补充的测试用例

测试用例	输入数据	预期输出
Test Case 1	-1	非法成绩
Test Case 2	0	不及格
Test Case 3	1	不及格
Test Case 4	59	不及格
Test Case 5	60	登记成绩 60 分
Test Case 6	61	登记成绩 61 分
Test Case 7	99	登记成绩 99 分
Test Case 8	100	登记成绩 100 分
Test Case 9	101	非法成绩

【例 3-5】 某公司招聘销售人员,规定报名者为 1975 年 1 月 1 日到 1995 年 1 月 1 日期间出生,若出生日期为 1995 年 1 月 1 日之后,将显示“对不起,您的年龄太小了!!”并拒绝接受;若出生日期在 1975 年 1 月 1 日之前,将显示“对不起,您的年龄超出了!!”并拒绝接受。用边界值分析法为例 3-2 补充测试用例。

解: 对于例 3-2 的表 3-5 中编号为(2)、(3)、(4)的有效等价类补充测试用例,如表 3-12 所示。

表 3-12 边界值分析法为报名功能补充测试用例

测试用例	输入数据	预期输出
Test Case 1	19741231	对不起,您的年龄超出了!!
Test Case 2	19750101	接受报名
Test Case 3	19750102	接受报名
Test Case 4	19941231	接受报名

续表

测试用例	输入数据	预期输出
Test Case 5	19950101	接受报名
Test Case 6	19950102	对不起,您的年龄太小了!!
Test Case 7	19760021	输入错误提示
Test Case 8	19770113	接受报名
Test Case 9	19780228	接受报名
Test Case 10	19911129	接受报名
Test Case 11	19921230	接受报名
Test Case 12	19941308	输入错误提示
Test Case 13	19850500	输入错误提示
Test Case 14	19850501	接受报名
Test Case 15	19850502	接受报名
Test Case 16	19900530	接受报名
Test Case 17	19900731	接受报名
Test Case 18	19900732	输入错误提示

3.4 错误猜测法

为了熟练掌握黑盒测试技术的错误猜测方法,需要学习以下内容:

- 错误猜测法概念。
- 错误猜测法设计测试用例实例分析。

3.4.1 错误猜测法概述

错误猜测法是指在测试程序时根据经验、知识或直觉推测程序中可能存在的各种错误,从而有针对性地编写检查这些错误的测试用例的方法。

错误猜测方法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。

例如,测试一个对线性表(比如数组)进行排序的程序,根据测试经验列出以下几项容易出错的地方:

- (1) 输入的线性表为空表。
- (2) 表中只含有一个元素。
- (3) 输入表中所有元素已排好序。
- (4) 输入表已按逆序排好。
- (5) 输入表中部分或全部元素相同。

针对以上容易发生错误的情况来设计测试用例。

3.4.2 错误猜测法实例

【例 3-6】 对于例 3-2、例 3-5，用错误猜测法为其补充设计测试用例。

解：对于报名功能，虽然在例 3-2、例 3-5 中使用了等价类划分法及边界值分析法为其设计了测试用例，但每个月最后一日的输入，仍然是容易发生错误的地方。

- (1) 对于 4、6、9、11 月，如果输入了 31 日，会发生错误。
- (2) 对于闰年的 2 月，如果输入了 30、31 日，会发生错误。
- (3) 对于非闰年的 2 月，如果输入了 29、30、31 日，会发生错误。
- (4) 在例 3-5 中，2、4、6、9、11 月的边界问题没考虑到，而这些地方是容易出错的地方，也需考虑设计补充用例。
- (5) 出生日期为 0。
- (6) 年月日次序颠倒。

用错误猜测法针对上面容易出错的地方补充设计测试用例，如表 3-13 所示。

表 3-13 错误猜测法为报名功能设计测试用例

测试用例	输入数据	预期输出
Test Case 1	19880430	接受报名
Test Case 2	19880431	输入错误提示
Test Case 3	19920229	接受报名
Test Case 4	19920230	输入错误提示
Test Case 5	19920231	输入错误提示
Test Case 6	19930228	接受报名
Test Case 7	19930229	输入错误提示
Test Case 8	19930230	输入错误提示
Test Case 9	19930231	输入错误提示
Test Case 10	0	输入错误提示
Test Case 11	01199502	输入错误提示
Test Case 12	11301991	输入错误提示

3.5 因 果 图

为了熟练掌握黑盒测试技术的因果图法，需要学习以下内容：

- 因果图基本知识和测试用例生成步骤。
- 因果图法设计测试用例实例分析。

3.5.1 因果图概述

等价类划分法和边界值分析法都着重考虑输入条件，而没有考虑输入条件之间的相互联系及各种组合情况，也没有考虑输入条件及输出之间的相互制约关系。对于输入条

件之间没有什么联系的程序,采用等价类划分法和边界值分析法是一种比较有效的测试方法。

在很多时候,测试时必须考虑输入条件的各种组合,因为输入条件之间的相互组合可能会产生一些新的情况。但要检查输入条件的组合不是一件容易的事情,即使把所有输入条件划分成等价类,它们之间的组合情况也相当多。因此必须考虑采用一种适合描述多种条件的组合,相应产生多个动作的形式来考虑设计测试用例。这就需要利用因果图(逻辑模型)。

因果图(cause effect graphics)是一种形式化语言,是一种组合逻辑网络图。它是把输入条件视为“因”,把对输入数据经过执行了一系列计算后得到的输出或程序状态的改变视为“果”,将黑盒看成是从因到果的网络图,采用逻辑图的形式来表达功能说明书中输入条件的各种组合与输出的关系。

1. 因果图的图形符号

因果图有两种类型的图形符号,即关系符号和约束符号。

1) 关系符号

因果图中使用了简单的图形符号,以直线连接左右节点。左节点表示输入状态(原因),右节点表示输出状态(结果)。

在因果图中用4种图形符号表示4种常用的因果关系。其中 c_i 表示原因,通常位于因果图的左部; e_i 表示结果,位于因果图的右部。 c_i 和 e_i 取值可为0或1,0表示某状态不出现,1表示某状态出现。

(1) 恒等。若 c_1 是1,则 e_1 也是1;若 c_1 是0,则 e_1 也是0。其图形表示如图3-1所示。

(2) 非(\sim)。若 c_1 是1,则 e_1 是0;若 c_1 是0,则 e_1 是1。其图形表示如图3-2所示。

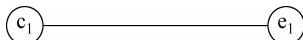


图 3-1 恒等关系图形符号



图 3-2 非关系图形符号

(3) 或(\vee)。若 c_1 、 c_2 、 c_3 有一个或一个以上是1,则 e_1 是1;若 c_1 、 c_2 、 c_3 都是0,则 e_1 是0。其图形表示如图3-3所示。“或”可有两个以上的输入。

(4) 与(\wedge)。若 c_1 、 c_2 、 c_3 都是1,则 e_1 是1;否则 e_1 为0。其图形表示如图3-4所示。“与”也可有两个以上的输入。

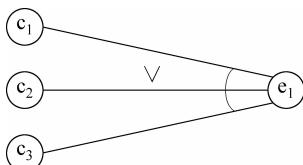


图 3-3 或关系图形符号

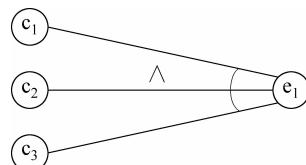


图 3-4 与关系图形符号

2) 约束符号

在输入条件(或状态)之间还可能存在某些依赖关系。例如,某些输入条件不可能同时出现。在多个输出结果(或状态)之间也可能存在强制的约束关系。这些关系对测试是非常重要的。在因果图中,用特定的符号标明这些约束或强制关系。

(1) E(互斥)约束。a 和 b 两个原因最多只能有一个为 1,即 a 和 b 不能同时为 1。其图形表示如图 3-5 所示。

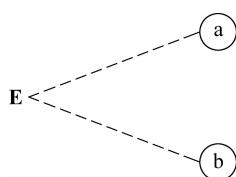


图 3-5 E 约束图形符号

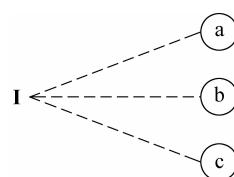


图 3-6 I 约束图形符号

(3) O(唯一)约束。a 和 b 两个原因必须有一个且仅有一个为 1。其图形表示如图 3-7 所示。

(4) R(要求)约束。a 和 b 两个原因,a 是 1 时,b 必须是 1,即不可能 a 是 1 时 b 是 0。其图形符号如图 3-8 所示。

(5) M(强制)约束。若输出结果 a 是 1,则输出结果 b 强制为 0;而输出结果 a 是 0 时,输出结果 b 的值不定。其图形表示如图 3-9 所示。

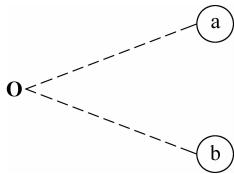


图 3-7 O 约束图形符号

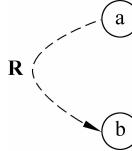


图 3-8 R 约束图形符号

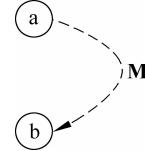


图 3-9 M 约束图形符号

以上(1)~(4)为对输入状态(或输入条件)的约束,只有(5)为对输出结果的约束。

2. 因果图生成测试用例步骤

利用因果图生成测试用例的基本步骤如下:

(1) 分析被测试部分的规格说明,找出原因与结果。

根据软件被测试部分的规格说明描述,分析哪些是原因(即输入条件或其等价类),哪些是结果(即输出结果或输出状态),并给每个原因和结果赋予一个标识符。

(2) 创建因果图。

根据软件被测试部分的规格说明描述中的语义,找出原因与结果之间的因果关系以及原因与原因之间、结果与结果之间的约束或强制关系。根据这些关系画出因果图。

- (3) 把因果图转换为判定表。
- (4) 根据判定表中的每一列设计测试用例。

3.5.2 因果图法实例

【例 3-7】 某企业工资管理软件一个模块的需求规格说明书中描述如下：

- (1) 行政管理员工：严重过失，扣年终奖的 6%；过失，扣年终奖的 4%。
 - (2) 一线生产员工：严重过失，扣年终奖的 10%；过失，扣年终奖的 6%。
- 用因果图法为其设计测试用例。

解：第一步：对需求规格说明书进行分析，得到原因和结果。

原因：

- 1——行政管理员工。
- 2——一线生产员工。
- 3——严重过失。
- 4——过失。

结果：

- 21——扣年终奖的 4%。
- 22——扣年终奖的 6%。
- 23——扣年终奖的 10%。

第二步：画出因果图

根据第一步分析出的原因和结果及需求规格说明书中的描述，将原因和结果之间的关系及原因和原因之间的约束关系用因果图相应的图形符号表示出来，得出的因果图如图 3-10 所示。

第三步：将图 3-10 所示的因果图转换为对应的判定表，如表 3-14 所示。

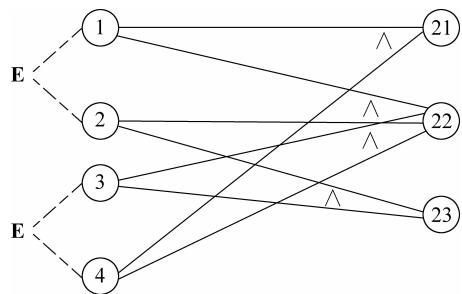


图 3-10 企业工资管理扣年终奖因果图

表 3-14 企业工资管理扣年终奖判定表

选项	规则		1	2	3	4	5	6
	原因(输入)	结果(输出)	1	2	3	4	5	6
原因(输入)	行政管理员工	扣年终奖的 4%	1	1	1	0	0	0
	一线生产员工	扣年终奖的 6%	2	0	0	1	1	1
	严重过失	扣年终奖的 10%	3	1	0	1	0	0
	过失	扣年终奖的 10%	4	0	1	0	1	0
结果(输出)	行政管理员工	扣年终奖的 6%	21	0	1	0	0	0
	一线生产员工	扣年终奖的 10%	22	1	0	0	1	0
	严重过失	扣年终奖的 10%	23	0	0	1	0	0

第四步：针对表 3-14 的每一列设计测试用例，如表 3-15 所示。

表 3-15 企业工资管理扣年终奖测试用例

测试用例	输入数据		预期输出
	员工类型	过失	
Test Case 1	行政管理员工	严重过失	扣年终奖的 6%
Test Case 2	行政管理员工	过失	扣年终奖的 4%
Test Case 3	行政管理员工	无	不扣年终奖
Test Case 4	一线生产员工	严重过失	扣年终奖的 10%
Test Case 5	一线生产员工	过失	扣年终奖的 6%
Test Case 6	一线生产员工	无	不扣年终奖

【例 3-8】有一个处理单价为 3 元 5 角瓶装饮料的自动售货机软件。若投入 3 元 5 角硬币,按下“红茶”或“绿茶”按钮,相应的饮料就送出来。若投入 4 元硬币,在送出饮料的同时退还 5 角硬币。试用因果图法为其设计测试用例。

解:第一步:对题目进行分析,得到原因和结果。

原因:

- 1——投 4 元硬币。
- 2——投 3 元 5 角硬币。
- 3——按“红茶”按钮。
- 4——按“绿茶”按钮。

中间节点:

- 11——已投币。
- 12——已按按钮。

结果:

- 21——退 5 角硬币。
- 22——送出“红茶”饮料。
- 23——送出“绿茶”饮料。

第二步:画出因果图。

将原因和结果之间的关系及原因和原因之间的约束关系用因果图相应的图形符号表示出来,得出的因果图如图 3-11 所示。

第三步:将图 3-11 所示的因果图转换为相应的判定表,如表 3-16 所示。

表 3-16 自动售货机判定表

选项	规则		1	2	3	4	5	6	7	8
	原因(输入)	结果	1	2	3	4	5	6	7	8
原因(输入)	投 4 元硬币	1	1	1	0	0	0	0	0	0
	投 3 元 5 角硬币	2	0	0	0	1	1	1	0	0
	按“红茶”按钮	3	1	0	0	1	0	0	1	0
	按“绿茶”按钮	4	0	1	0	0	1	0	0	1

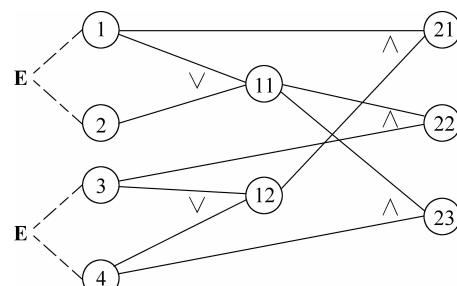


图 3-11 自动售货机因果图

续表

规则		1	2	3	4	5	6	7	8
选项									
中间节点	已投币	11	1	1	1	1	1	0	0
	已按钮	12	1	1	0	1	1	0	1
结果(输出)	退 5 角硬币	21	1	1	0	0	0	0	0
	送出“红茶”饮料	22	1	0	0	1	0	0	0
	送出“绿茶”饮料	23	0	1	0	0	1	0	0

针对表 3-16 的每一列设计测试用例(略)。

【例 3-9】 某软件规格说明书中对“订餐处理”的描述为：如果订单未过期，则向顾客发出菜单和用餐日期提醒；如果订餐金额不足 300 元，已过期的订单不发通知单；如果订餐金额超过 300 元但不足 800 元，对已经过期的订单发过期通知单；如果订餐金额超过 800 元，对已经过期的订单发出菜单和修改用餐日期提醒。

解：第一步：对规格说明书进行分析，得到原因和结果。

原因：

- 1——订餐金额不足 300 元。
- 2——订餐金额超过 300 元但不足 800 元。
- 3——订餐金额超过 800 元。
- 4——已过期。
- 5——未过期。

中间节点：

- 11——11 为节点 1 或节点 2 或节点 3。

结果：

- 21——发菜单。
- 22——发用餐日期提醒。
- 23——发过期通知单。
- 24——发修改用餐日期提醒。

第二步：画出因果图。

将原因和结果之间的关系及原因和原因之间的约束关系用因果图相应的图形符号表示出来，得出的因果图如图 3-12 所示。

第三步：将因果图转换为判定表。

将图 3-12 转换为判定表，如表 3-17 所示。

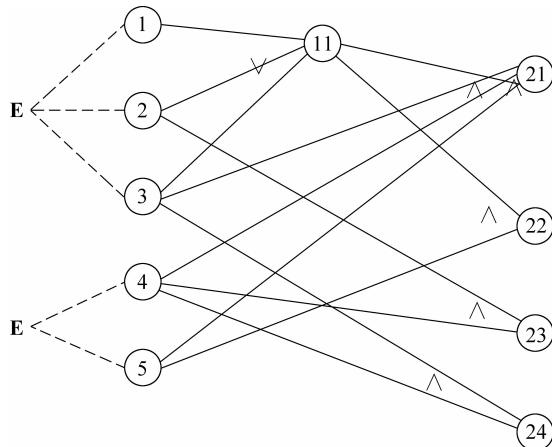


图 3-12 订餐处理因果图

表 3-17 订餐处理判定表

选项	规则		1	2	3	4	5	6
	原因(输入)	中间节点						
原因(输入)	订餐金额不足 300 元	1	1	1	0	0	0	0
	订餐金额超过 300 元但不足 800 元	2	0	0	1	1	0	0
	订餐金额超过 800 元	3	0	0	0	0	1	1
	已过期	4	1	0	1	0	1	0
	未过期	5	0	1	0	1	0	1
中间节点	为节点 1 或节点 2 或节点 3	11	1	1	1	1	1	1
结果(输出)	发菜单	21	0	1	0	1	1	1
	发用餐日期提醒	22	0	1	0	1	0	1
	发过期通知单	23	0	0	1	0	0	0
	发修改用餐日期提醒	24	0	0	0	0	1	0

为判定表的每一列设计测试用例(略)。

【例 3-10】 公交一卡通自动充值软件系统需求如下：

- (1) 系统只接收 50 元或 100 元纸币,一次充卡只能使用一张纸币,一次充值金额只能为 50 元或 100 元。
- (2) 若输入 50 元纸币,并选择充值 50 元,完成充值后退卡,提示充值成功。
- (3) 若输入 50 元纸币,并选择充值 100 元,提示输入金额不足,并退还 50 元。
- (4) 若输入 100 元纸币,并选择充值 50 元,完成充值后退卡,提示充值成功,找零 50 元。
- (5) 若输入 100 元纸币,并选择充值 100 元,完成充值后退卡,提示充值成功。

- (6) 若输入纸币后在规定时间内不选择充值按钮,退回输入的纸币,并提示错误。
 (7) 若选择充值按钮后不输入纸币,提示错误。

解: 第一步: 对公交一卡通自动充值软件系统需求进行分析, 得到原因和结果。

原因:

- 1——投 50 元纸币。
- 2——投 100 元纸币。
- 3——充值 50 元。
- 4——充值 100 元。
- 5——超时。

中间节点:

- 11——节点 1 和节点 2 不能同时存在, 11 为节点 1 或节点 2。
- 12——节点 3 和节点 4 不能同时存在, 12 为节点 3 或节点 4。

结果:

- 21——完成充值后退卡, 提示充值成功。
- 22——提示输入金额不足。
- 23——找零 50 元。
- 24——退回 50 元。
- 25——退回 100 元。
- 26——提示错误。

第二步: 分析公交一卡通自动充值软件系统需求, 将原因和结果之间的关系及原因和原因之间的约束关系用因果图相应的图形符号表示出来, 得出的因果图如图 3-13 所示。

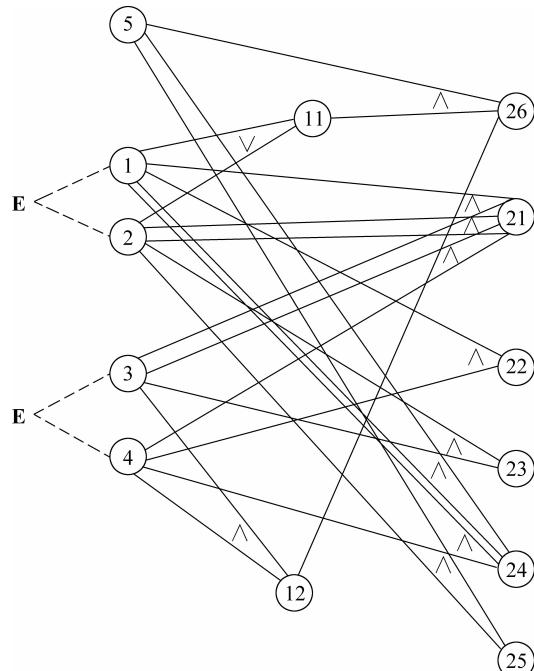


图 3-13 公交一卡通自动充值软件因果图

第三步：将因果图转换为判定表。

将图 3-13 转换为判定表，如表 3-18 所示。

表 3-18 公交一卡通自动充值软件判定表

选项		规则	1	2	3	4	5	6	7	8
原因(输入)	输入 50 元	1	1	1	0	0	0	0	0	0
	输入 100 元	2	0	0	0	1	1	1	0	0
	充值 50 元	3	1	0	0	1	0	0	1	0
	充值 100 元	4	0	1	0	0	1	0	0	1
	超时	5	0	0	1	0	0	1	0	0
中间节点	输入 50 元或 100 元	11	1	1	1	1	1	1	0	0
	充值 50 元或 100 元	12	1	1	0	1	1	0	1	1
结果(输出)	充值后退卡并提示	21	1	0	0	1	1	0	0	0
	提示输入金额不足	22	0	1	0	0	0	0	0	0
	找零 50 元	23	0	0	0	1	0	0	0	0
	退回 50 元	24	0	1	1	0	0	0	0	0
	退回 100 元	25	0	0	0	0	0	1	0	0
	提示错误	26	0	0	1	0	0	1	1	1

为判定表的每一列设计测试用例(略)。

3.6 判定表驱动法

为了熟练掌握黑盒测试技术的判定表驱动法，需要学习以下内容：

- 认识什么是判定表。
- 判定表的组成和创建判定表的步骤。
- 判定表驱动法设计测试用例实例分析。

3.6.1 认识判定表

假设规格说明书中的输入条件有条件 1、条件 2。当条件 1 和条件 2 均为真时，执行的动作是操作 1；当条件 1 和条件 2 均为假时，执行的动作是操作 2；当条件 1 和条件 2 中有一个为真时，执行的动作是操作 3；条件为真时用 1 表示，条件为假时用 0 表示，可用表 3-19 表示规格说明书中输入和输出的关系，该表即为判定表。