软件质量保证与测试

第3章 黑盒测试

内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

3.1 概述

- **黑盒测试是在程序接口进行的测试**

  只知道系统输入和预期输出，不需要了解程序内部结构和内部特性的测试方法

  - 程序功能是否能按照规格说明书的规定正常使用
  - 程序能否适当地接收输入数据并产生正确的输出数据
  - 程序运行过程中能否保持外部信息的完整性

概述 (续)

- **黑盒测试的原理：**
  - 任何程序都可以看做是从输入定义域到输出值域的映射
  - 对程序功能的理解仅基于输入和输出，对其实现原理和过程一无所知。

概述 (续)

- **黑盒测试的优点：**
  - 方法简单有效
  - 可以整体测试系统的行为
  - 测试用例可以重用
  - 开发与测试可以并行
  - 对测试人员技术要求相对较低

概述 (续)

- 穷举输入测试可行吗？
- 软件测试是不完备的
- 软件测试是有风险的

概述 (续)

- **测试方法的评价标准**

  在最短时间内，以最少的人力，有利于发现最多的，以及最严重的缺陷。

以用户需求为中心
高效的测试用例设计

概述 (续)

- 测试用例的设计标准
- 测试用例的覆盖度：高
- 测试用例的数量：少
- 测试用例的冗余度：低
- 测试用例的缺陷定位能力：高
- 测试方法的复杂度：高

内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

3.2 等价类划分

例：计算两个1~100之间整数的乘积

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;
    cout<<“请输入两个1到100之间的整数：”<<endl;
    cin>>a>>b;
    if ((a >= 1 && a <= 100) && (b >=1 && b <= 100))
    {
        c = a * b;
        cout<<“两个数的乘积为:”<<c<<endl;
     }
     return 0;
 }
```

等价类划分(续)

用例编号	乘数1	乘数2	乘积
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	5
…	…	…	…

等价类划分(续)

- 对系统进行穷举测试是不可能的
- 选择少量测试用例来测试系统，并满足
  - 测试是完备的
  - 测试没有冗余

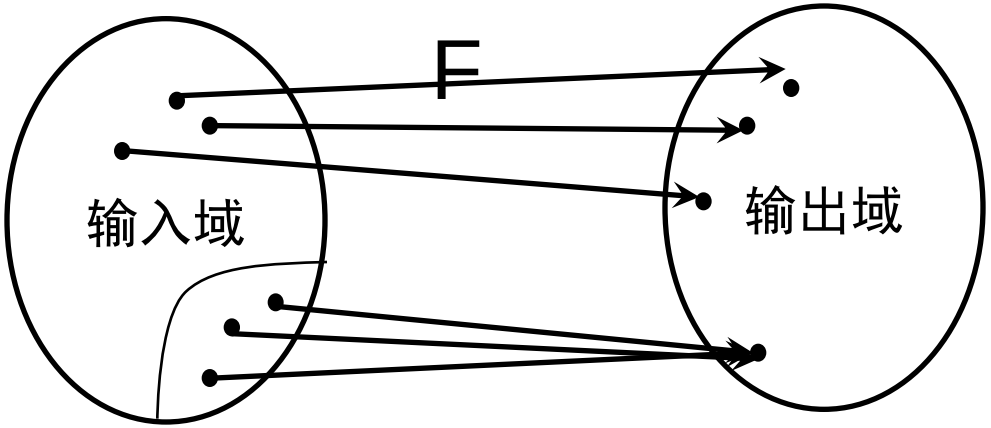
等价类划分(续)

- **等价类划分法**
  - 把所有可能的输入数据，即程序的输入域划分成若干部分(子集)
  - 从每一个子集中选取少数具有代表性的数据作为测试用例

等价类划分(续)

- **等价类划分要求：**
  - 任意两个等价类之间不存在交集 -> 无冗余
  - 所有等价类的并集为输入域 -> 完备性
  - 任意一个等价类中的数据等价 -> 以一代全
    - 对于揭露程序中的错误是等效的
    - 测试某个数据就等于对这一类其他数据的测试

等价类划分(续)



R是一种等价关系:

xRy ⇔ F(x)=F(y)

x,y ∈输入域

- **等价类：一组具有相同处理机制的数据集合**

等价类划分(续)

- 如何划分等价类？
- 如何利用等价类设计测试用例？

等价类划分(续)

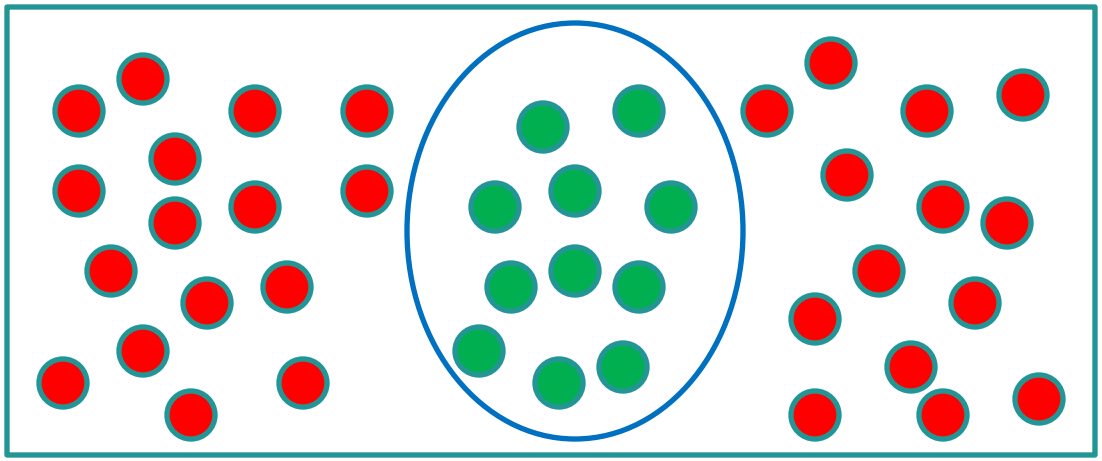
- **等价类划分可有两种不同的情况:**
  - 有效等价类:
    - 指对于程序的规格说明来说是合理的，有意义的输入数据构成的集合
    - 检验程序是否实现了规格说明中所规定的功能和性能。
  - 无效等价类:
    - 与有效等价类的定义恰巧相反，不符合需求规格说明书
    - 检验系统的容错性

等价类划分(续)

- **确定等价类的原则：**
  - 按区间划分，可以确立一个有效等价类和两个无效等价类。

等价类划分(续)

- **确定等价类的原则：**
  - 按数值集划分，可确立一个有效等价类和一个无效等价类。



等价类划分(续)

- **确定等价类的原则：**
  - 按布尔数值划分，可确定一个有效等价类和一个无效等价类，或确定两个有效等价类。
  - 按限制条件或规则划分，可确立一个有效等价类(符合规则)和若干个无效等价类(从不同角度违反规则)。
  - 细分等价类，则应再将该等价类进一步的划分为更小的等价类。

等价类划分(续)

- 计算两个1~100之间数的乘积

输入条件	有效等价类	无效等价类	
1~100间的数a	[1 ,100]	(-∞,1)	(100, +∞)
1~100间的数b	[1 ,100]	(-∞,1)	(100, +∞)

等价类划分(续)

- **设计测试用例**
  - 每一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被覆盖地有效等价类，重复这一步。直到所有的有效等价类都被覆盖为止。
  - 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。

等价类划分(续)

用例编号	所属等价类	乘数a	乘数b	乘积
1	1	3	20	60
2	2	-10	2	提示“请输入1~100之间的整数”
3	3	200	3	提示“请输入1~100之间的整数”

等价类划分(续)

例：输入三个整数a、b和c分别作为三角形的3条边，通过程序判断由这3条边构成的三角形类型是：等边三角形、等腰三角形、一般三角形或非三角形(不能构成一个三角形)。

等价类划分(续)

输入条件	有效等价类		无效等价类	
是否三角形的3条边	(a>0) and (b>0) and (c>0) and (a+b>c) and (b+c>a) and (a+c>b)	(1)	(a≤0)	(2)
			(b≤0)	(3)
			(c≤0)	(4)
			(a+b≤c)	(5)
			(b+c≤a)	(6)
			(a+c≤b)	(7)
是否等腰三角形	(a=b)	(8)	(a≠b) and (b≠c) and (c≠a)	(11)
	(b=c)	(9)		
	(c=a)	(10)		
是否等边三角形	(a=b)and(b=c)and(c=a)，	(12)	(a≠b)	(13)
			(b≠c)	(14)
			(c≠a)	(15)

等价类划分(续)

用例编号	【a，b，c】	覆盖等价类	输出
1	【3，4，5】	(1)	一般三角形
2	【0，1，2】	(2)	不能构成三角形
3	【1，0，2】	(3)	
4	【1，2，0】	(4)	
5	【1，2，3】	(5)	
6	【1，3，2】	(6)	
7	【3，1，2】	(7)	
8	【3，3，4】	(1)，(8)	等腰三角形
9	【3，4，4】	(1)， (9)	
10	【3，4，3】	(1)， (10)	
11	【3，4，5】	(1)， (11)	非等腰三角形
12	【3，3，3】	(1)，(12)	是等边三角形
13	【3，4，4】	(1)， (9)，(13)	非等边三角形
14	【3，4，3】	(1)， (10)，(14)	
15	【3，3，4】	(1)，(8)，(15)	

等价类划分(续)

等价类测试存在两个问题：

- 规格说明往往没有定义无效测试用例的期望输出应该是什么样的。因此，测试人员需要花费大量时间来定义这些测试用例的期望输出。
- 强类型语言没有必要考虑无效输入。

内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

3.3 边界值分析法

- 大量的测试实践经验表明，边界值是最容易出现问题的地方
- 边界值分析为检验边界附近的处理专门设计测试用例，常会取得良好的测试效果
- 边界值分析法是对等价类划分法的补充

  基本思想：选取正好等于、刚刚大于或刚刚小于等价类边界的值作为测试数据，而不是选取等价类中的典型值或任意值做为测试数据。

边界值分析法(续)

- **边界条件**

```
void BubbleSort(int A[ ], int n)
{
    int i, j, k;
    bool flag = false;
    for (i = 0; i < n-1; i++)
    {
        for (j = 0; j < n–i-1; j++)
        {
            if (A[j] > A[j+1])
                Swap(A[j], A[j+1]);
        }
    }
}
```

- **内外层循环次数的确定**
- **数组的下标**

边界值分析法(续)

- **边界条件是一些特殊情况，是软件计划的操作界限所在的边缘条件。**
- **一些可能与边界有关的数据类型有：数值，速度，字符，地址，位置，尺寸，数量等。同时，考虑这些数据类型的下述特征：**

  第一个/最后一个，最小值 / 最大值，开始 / 完成，超过/在内，空 / 满，最短 / 最长，最慢/最快，最早/最迟，最高 / 最低，相邻 / 最远等。

边界值分析法(续)

- 次边界条件

字符	ASCII值	字符	ASCII值	字符	ASCII值	字符	ASCII值
Null	0	2	50	B	66	a	97
Space	32	9	57	Y	89	b	98
/	47	:	58	Z	90	y	121
0	48	@	64	[	91	z	122
1	49	A	65	′	96	{	123

边界值分析法(续)

- **其他一些边界条件**
  - 当软件要求输入时，根本没有输入任何内容，只按了Enter键
  - 这些值通常在软件中进行特殊处理，不要把它们与合法情况和非法情况混在一起，而要建立单独的等价区间

边界值分析法(续)

- **边界值的选择方法**
  - ① 如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。
  - 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、比最小个数大1、比最大个数多1、比最大个数少1的数作为测试数据。
  - ③ 根据规格说明的每个输出条件，使用前面的原则①。
  - ④ 根据规格说明的每个输出条件，应用前面的原则②。

边界值分析法(续)

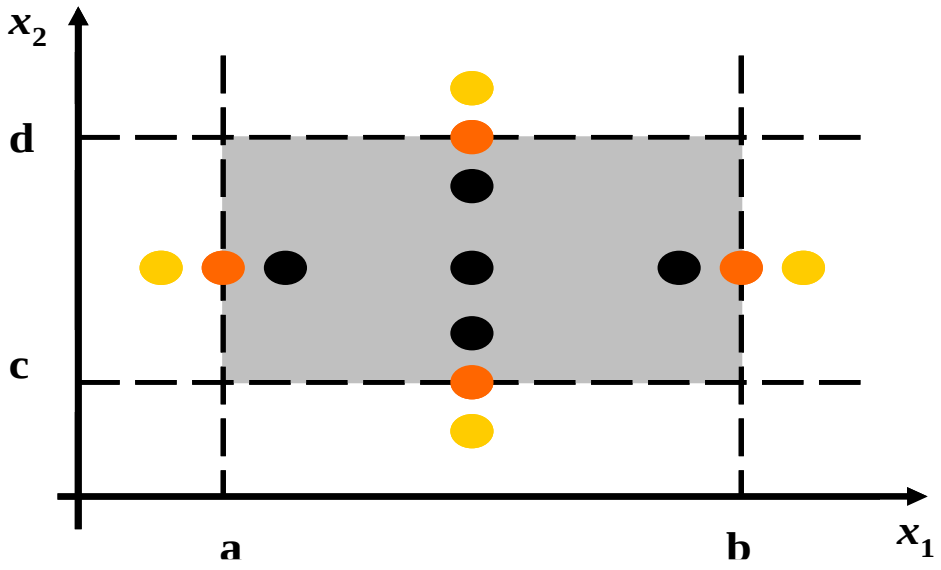
- **边界值的选择方法**
  - ⑤ 如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。
  - ⑥ 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构边界上的值作为测试用例。
  - ⑦ 分析规格说明，找出其他可能的边界条件。

边界值分析法(续)

- 例：一个有两个输入变量x_1和x_2的程序P。假设输入变量x_1和x_2在下列范围内取值:

$$a \le x_1 \le b,\ c \le x_2 \le d$$

边界值分析利用 输入变量的最小值（min）稍小于最小值(min-)稍大于最小值(min+)最大值(max)稍小于最大值(max-)稍大于最大值（max+）来设计测试用例。



边界值分析法(续)

- **乘法器边界值测试用例**

测试用例	输入数据		预期输出
	乘数1	乘数2	乘积
Test1	1	50	50
Test 2	2	50	100
Test 3	99	50	4950
Test 4	100	50	5000
Test 5	50	1	50
Test 6	50	2	100
Test 7	50	99	4950
Test 8	50	100	5000
Test 9	0	50	提示“请输入1～100间的整数”
Test 10	50	0	提示“请输入1～100间的整数”
Test11	101	50	提示“请输入1～100间的整数”
Test12	50	101	提示“请输入1～100间的整数”

内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

3.4 因果图法

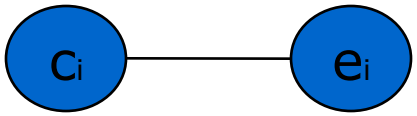
- **等价类划分法和边界值分析法着重考虑输入条件**
  - 没有考虑输入情况的各种组合
  - 没有考虑各个输入情况之间的相互制约关系
- **如果输入之间有关系，必须考虑描述多种条件的组合，产生多个相应动作的测试方法**
- **因果图法是从自然语言书写的程序规格说明书的描述中找出原因(输入条件)和结果(输出或程序状态的改变)**

因果图法(续)

1. 因果图
2. 因果图法的步骤

1、因果图

（1）基本符号

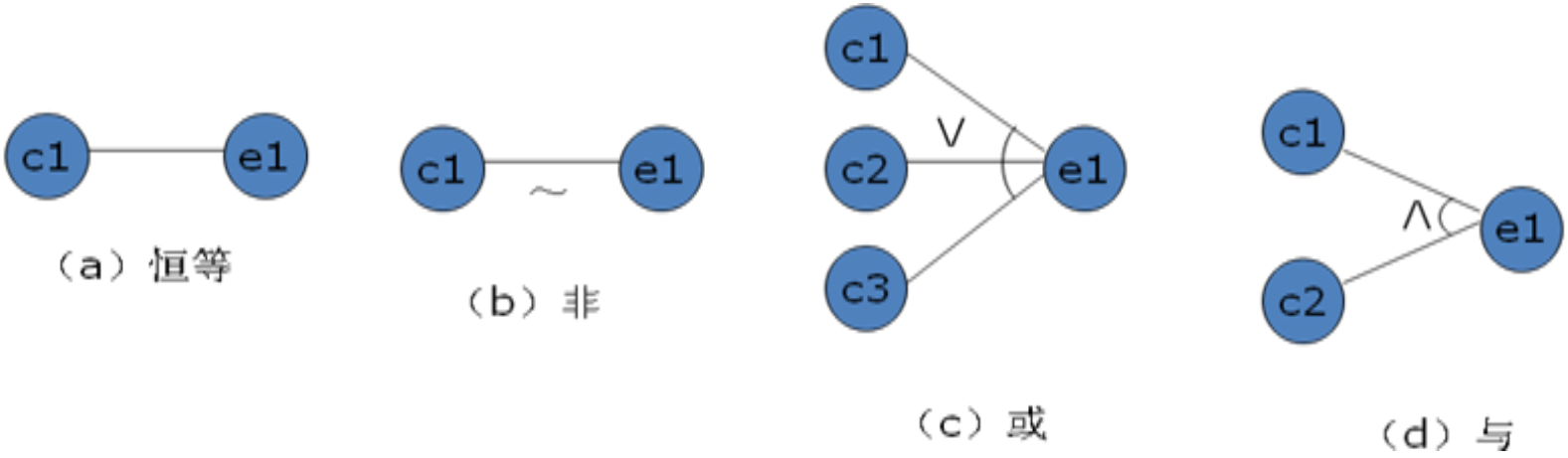


- ci表示原因（输入）
- ei表示结果（输出）
- 结点的状态：“0”或“1”

因果图（续）

（2）关系符号

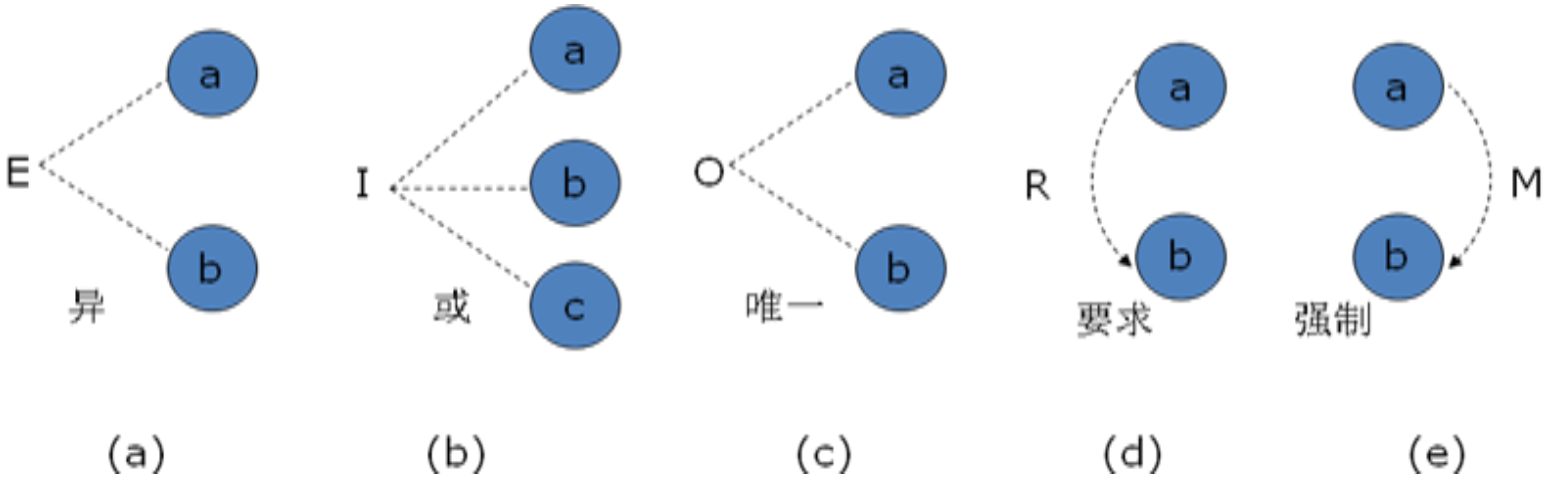
表示原因和结果之间的关系



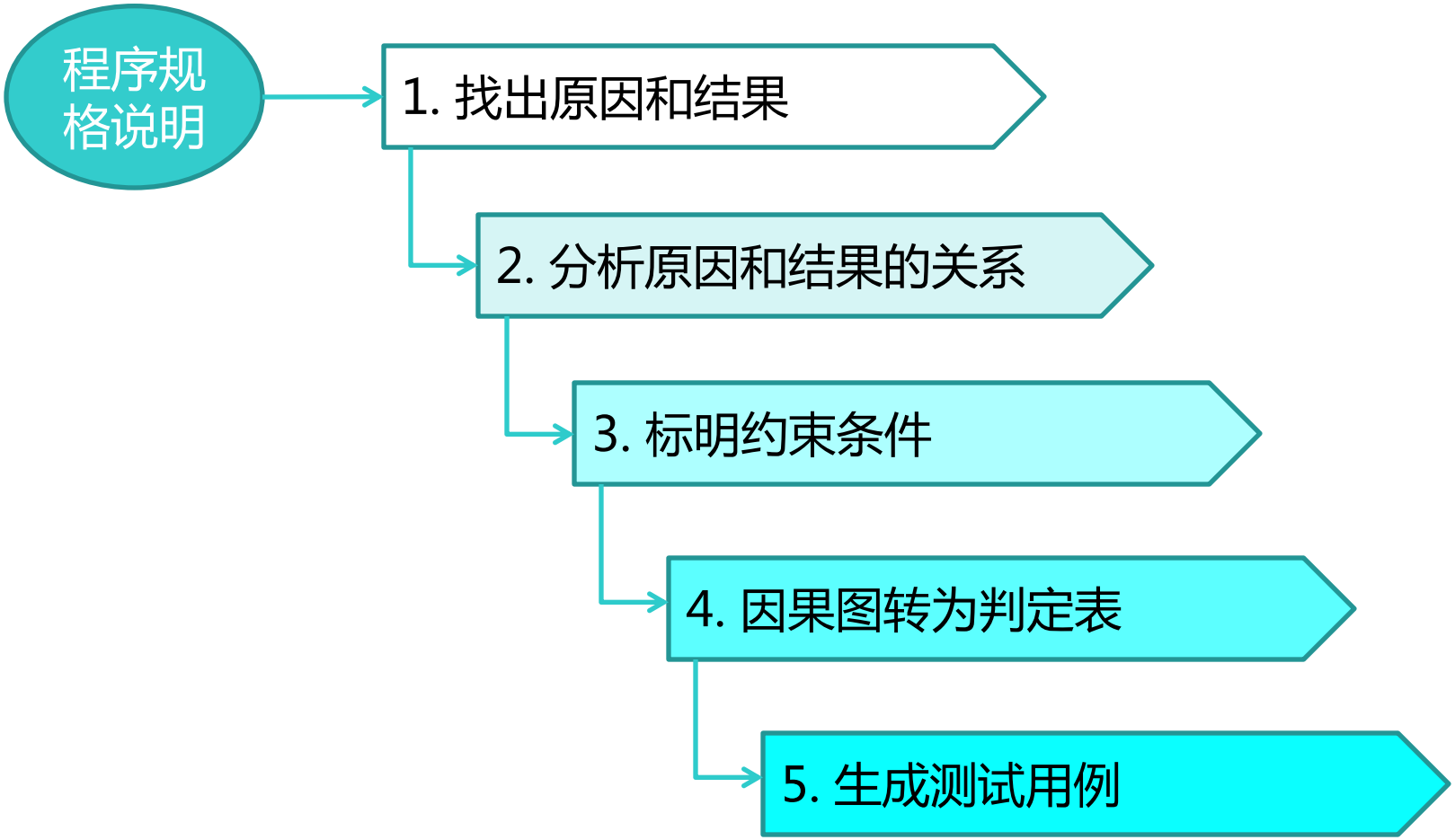
因果图（续）

（3）约束符号

表示原因与原因，结果与结果之间的约束条件



2. 因果图法的步骤



因果图法的步骤（续）

案例：自动售货机软件

若投入1元5角硬币，按下“可乐”、“雪碧”或“红茶”按钮，相应的饮料就送出来。

若投入2元硬币，按下“可乐”、“雪碧”或“红茶”按钮，相应饮料就送出来的同时退还5角硬币。

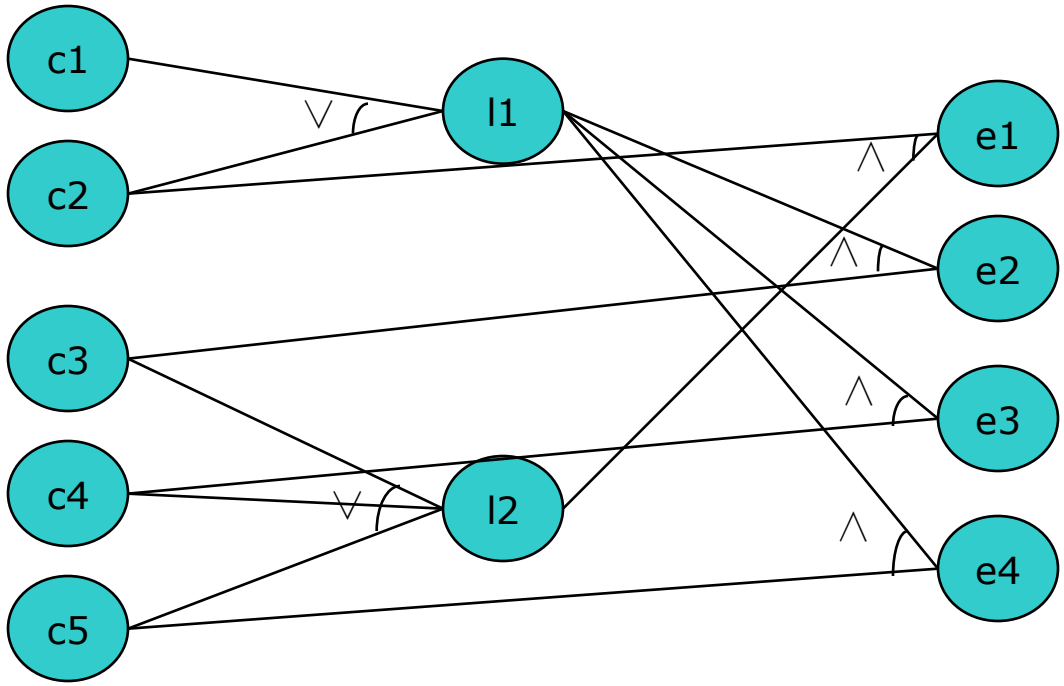
因果图法的步骤（续）

步骤1：找出原因和结果

原因	中间状态	结果
c1: 投入1元5角硬币	11：已投币	e1: 退还5角硬币
c2: 投入2元硬币	12：已按钮	e2: 送出“可乐”饮料
c3: 按“可乐”按钮		e3: 送出“雪碧”饮料
c4: 按“雪碧”按钮		e4: 送出“红茶”饮料
c5: 按“红茶”按钮		

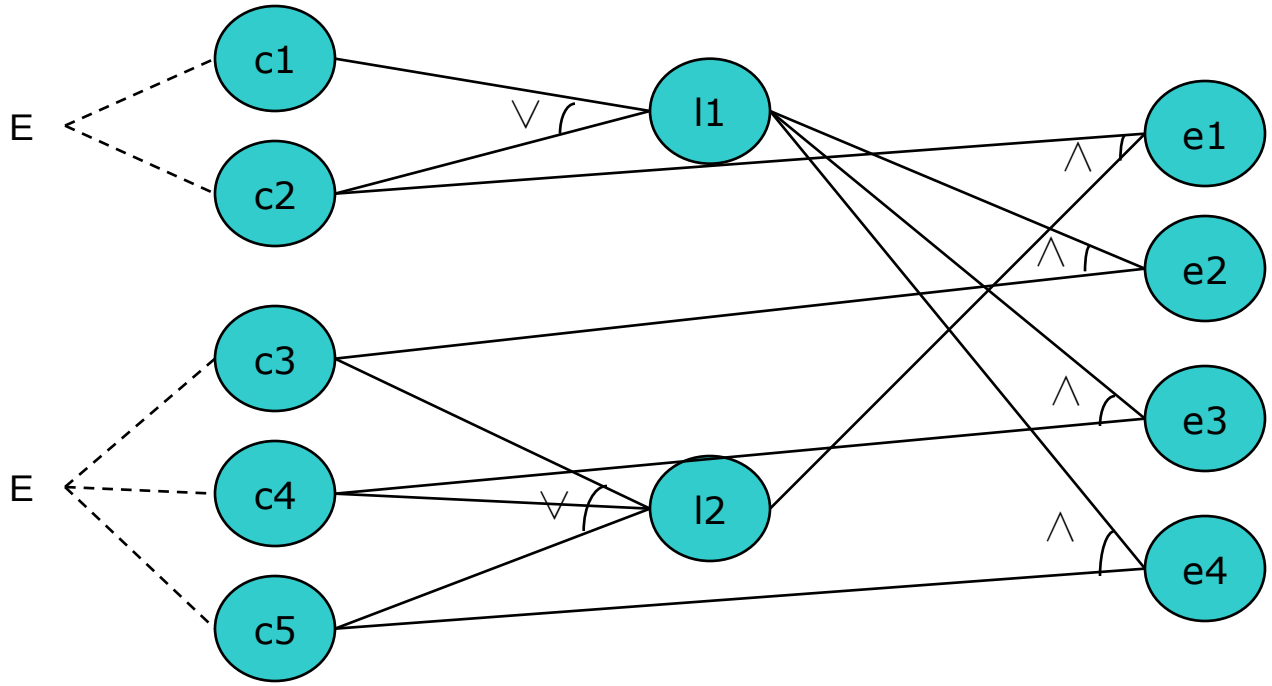
因果图法的步骤（续）

步骤2：分析原因和结果的关系



因果图法的步骤（续）

步骤3：表明约束条件



内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

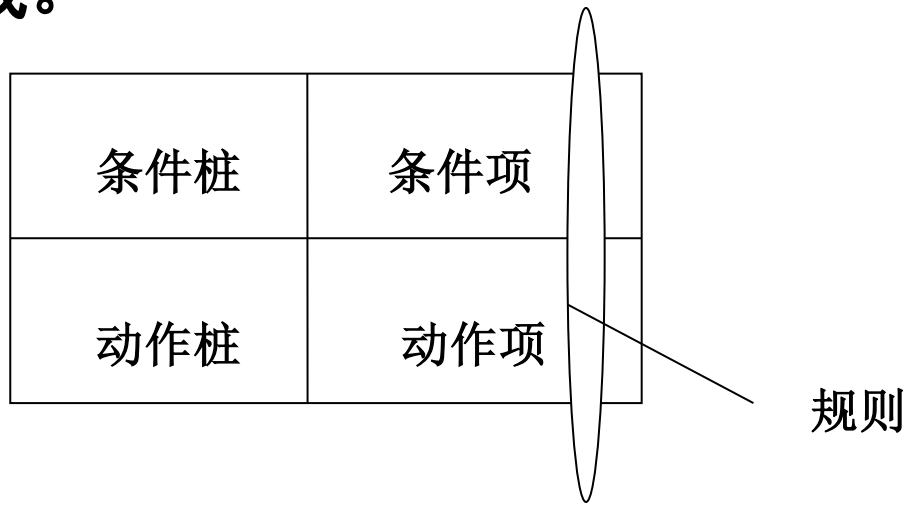
3.5 决策表法

- 决策表是把作为条件的所有输入的各种组合值以及对应输出值都罗列出来而形成的表格。

  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。
- 在所有的黑盒测试方法中，基于决策表的测试是最严格，最具有逻辑性的测试方法。

决策表法(续)

- 决策表通常由条件桩、条件项、动作桩和动作项4部分组成。



- 动作项和条件项紧密相关，指出在条件项的各组取值情况下应采取的动作。

决策表法(续)

- **决策表的构造：**
  - ① 列出所有的条件桩和动作桩。
  - ② 确定规则的个数。
  - ③ 填入条件项。
  - ④ 填入动作项，得到初始决策表。
  - ⑤ 简化决策表，合并相似规则。

决策表法(续)

- **决策表的简化：**
  - 对于n个条件的决策表，相应有2^n个规则（每个条件分别取真、假值），当n较大时，决策表很繁琐。
  - 实际使用决策表时，常常先将它简化。决策表的简化是以合并相似规则为目标。即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。

决策表法(续)

例：NextDate函数输入为month(月份)、day(日期)和year(年)，输出为输入后一天的日期。要求输入变量month、day和year都是整数值，并且满足以下条件：

C1. 1≤month≤12

C2. 1≤day≤31

C3. 1900≤year≤2050

决策表法(续)

为了获得下一个日期，NextDate函数需要执行的操作有如下6种：

A1：不可能

A2：day变量值加1；

A3：day变量值复位为1；

A4：month变量值加1；

A5：month变量值复位为1；

A6：year变量值加1。

决策表法(续)

为了处理NextDate函数的日和月问题，仔细研究动作桩。可以在以下的有效等价类集合上建立决策表。

C1 1≤month≤12
M_1: {month有30天};
M_2: {month有31天，
12月除外};
M_3: {month是12月};
M_4: {month是2月};

C2 1≤day≤31
D_1:{1≤day≤27};
D_2: {day=28};
D_3: {day=29};
D_4: {day=30};
D_5: {day=31};

C3 1900≤year≤2050
Y_1:{year:year是闰年};
Y_2: {year:year不是闰年}

决策表法(续)

- **NextDate函数的决策表**

选项 \ 规则		1	2	3	4	5	6	7	8	9	10	11
条件	C_1:month	M_1	M_1	M_1	M_1	M_1	M_2	M_2	M_2	M_2	M_2	M_3
	C_2:day	D_1	D_2	D_3	D_4	D_5	D_1	D_2	D_3	D_4	D_5	D_1
	C_3:year	—	—	—	—	—	—	—	—	—	—	—
动作	A_1:不可能					√						
	A_2:day加1	√	√	√			√	√	√	√		√
	A_3:day复位				√						√	
	A_4:month加1				√						√	
	A_5:month复位											
	A_6:year加1											

决策表法(续)

- **NextDate函数的决策表**

规则 选项		12	13	14	15	16	17	18	19	20	21	22
条件	C_1:month	M_3	M_3	M_3	M_3	M_4	M_4	M_4	M_4	M_4	M_4	M_4
	C_2:day	D_2	D_3	D_4	D_5	D_1	D_2	D_2	D_3	D_3	D_4	D_5
	C_3:year	—	—	—	—	—	Y_1	Y_2	Y_1	Y_2	—	—
动作	A_1:不可能									√	√	√
	A_2:day加1	√	√	√		√	√					
	A_3:day复位				√			√	√			
	A4:month加1							√	√			
	A5:month复位				√							
	A6:year加1				√							

决策表法(续)

- **NextDate函数简化的决策表**

规则 / 选项		1~3	4	5	6~9	10	11~14	15	16	17	18	19	20	21~22
条件	C_1	M_1	M_1	M_1	M_2	M_2	M_3	M_3	M_4	M_4	M_4	M_4	M_4	M_4
	C_2	D_1~3	D_4	D_5	D_1~4	D_5	D_1~4	D_5	D_1	D_2	D_2	D_3	D_3	D_4, D_5
	C_3	–	–	–	–	–	–	–	–	Y_1	Y_2	Y_1	Y_2	–
动作	A_1			√									√	√
	A_2	√			√		√		√	√				
	A_3		√			√		√			√	√		
	A_4		√			√					√	√		
	A_5							√						
	A_6							√						

决策表法(续)

- **NextDate函数的测试用例**

测试用例2	Month	day	year	预期输出
Test1-3	6	16	2001	2001.6.17
Test4	6	30	2001	2001.7.1
Test5	6	31	2001	不可能
Test6-9	1	16	2001	2001.1.17
Test10	1	31	2001	2001.2.1
Test11-14	12	16	2001	2001.12.17
Test15	12	31	2001	2002.1.1
Test16	2	16	2001	2001.2.17
Test17	2	28	2004	2004.2.29
Test18	2	28	2001	2001.3.1
Test19	2	29	2004	2004.3.1
Test20	2	29	2001	不可能
Test21-22	2	30	2001	不可能

决策表法(续)

- 自动售货机软件基于因果图的决策表

		1	2	3	4	5	6	7	8	9	10	11
输入	投入1元5角硬币	1	1	1	1	0	0	0	0	0	0	0
	投入2元硬币	0	0	0	0	1	1	1	1	0	0	0
	按“可乐”按钮	1	0	0	0	1	0	0	0	1	0	0
	按“雪碧”按钮	0	1	0	0	0	1	0	0	0	1	0
	按“红茶”按钮	0	0	1	0	0	0	1	0	0	0	1
中间结点	已投币	1	1	1	1	1	1	1	1	0	0	0
	已按钮	1	1	1	0	1	1	1	0	1	1	1
输出	退还5角硬币	0	0	0	0	1	1	1	0	0	0	0
	送出“可乐”饮料	1	0	0	0	1	0	0	0	0	0	0
	送出“雪碧”饮料	0	1	0	0	0	1	0	0	0	0	0
	送出“红茶”饮料	0	0	1	0	0	0	1	0	0	0	0

决策表法(续)

- 自动售货机软件的测试用例

用例编号	测试用例	预期输出
1	投入1元5角，按“可乐”	送出“可乐”饮料
2	投入1元5角，按“雪碧”	送出“雪碧”饮料
3	投入1元5角，按“红茶”	送出“红茶”饮料
4	投入2元，按“可乐”	找5角，送出“可乐”
5	投入2元，按“雪碧”	找5角，送出“雪碧”
6	投入2元，按“红茶”	找5角，送出“红茶”

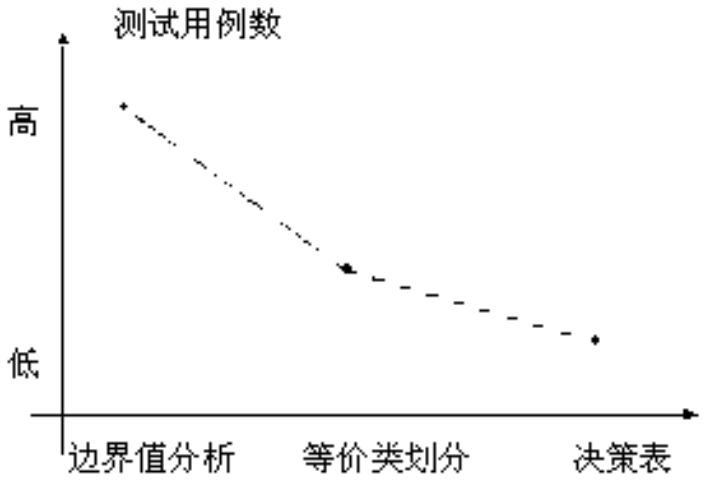
内容提要

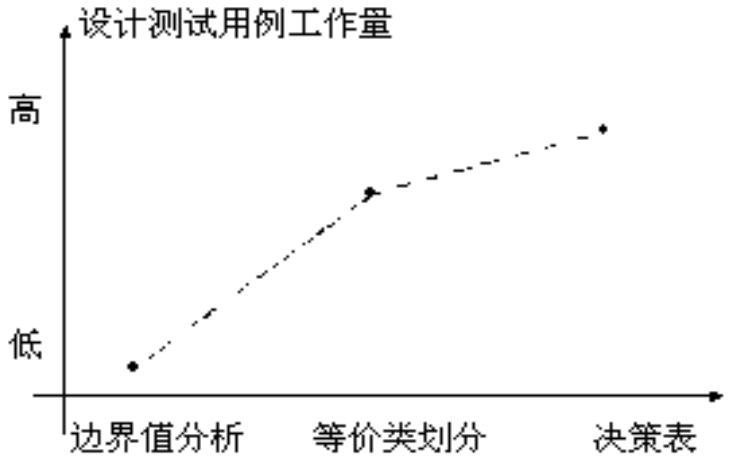
- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

3.6 黑盒测试方法的比较与选择

- **共同点：都把程序看作是一个打不开的黑盒，只知道输入到输出的映射关系，根据软件规格说明设计测试用例**
- **区别**
  - 等价类分析测试：通过等价类划分来减少测试用例的绝对数量。
  - 边界值分析方法：通过分析输入变量的边界值域设计测试用例。
  - 因果图和决策表测试：通过分析被测程序的逻辑依赖关系，构造决策表，进而设计测试用例

黑盒测试方法的比较与选择(续)





黑盒测试方法的比较与选择(续)

- **综合使用各种方法才能有效地提高测试效率和测试覆盖度：**
  - 首先进行等价类划分，包括输入条件和输出条件的等价划分，将无限测试变成有限测试，这是减少工作量和提高测试效率最有效的方法。
  - 在任何情况下都必须使用边界值分析方法。经验表明，用这种方法设计出的测试用例发现程序错误的能力最强。
  - 可以用错误推测法追加一些测试用例，这需要依靠测试工程师的智慧和经验。
  - 如果程序的功能说明中含有输入条件的组合情况，则一开始就可选用因果图法和决策表驱动法。

内容提要

- 3.1 概述
- 3.2 等价类划分
- 3.3 边界值分析法
- 3.4 因果图法
- 3.5 决策表法
- 3.6 黑盒测试方法的比较与选择
- 3.7 小结

3.7 小结

- 等价类划分法
- 边界值分析法
- 因果图法
- 决策表法