



软件测试基础

第二章 测试方法(2)



【本章知识要点】

➤ 学习完本章后，应该掌握如下知识

- 软件测试与开发的关系
- 软件测试与软件质量的关系
- 软件可测性与可靠性
- 软件测试范围和内容
- 软件测试用例
- 需求文档测试方法
- 设计文档测试方法
- 单元测试方法
- 集成后的系统测试方法
- 测试过程中的集成方法



2.8 单元测试方法

- 单元概念
 - 界面
 - 函数
 - 类
- 动态测试
- 静态分析（代码质量分析）
- 用代码复杂度分析风险

2.8 单元测试方法

动态测试

➤ 黑盒测试

- 边界值
- 等价类划分
- 组合测试
- 边界、等价、输入组合三者的转化
- 因果图测试
- 基于决策表的测试

➤ 白盒测试

- 块测试(语句覆盖)
- 判定覆盖
- 条件覆盖
- 判定条件覆盖
- 条件组合覆盖
- 基路径测试
- 数据流测试(研究)
- 程序切片技术(研究)

2.8 单元测试方法

2.8.1 黑盒测试

➤ 黑盒测试概念

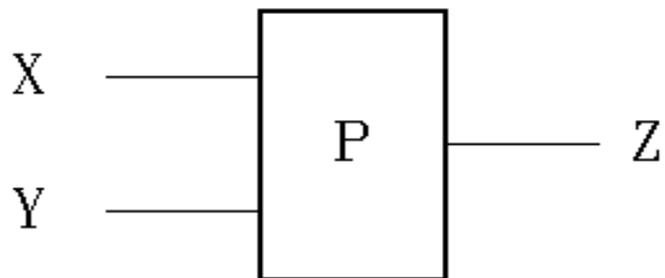
□ 测试又叫做功能测试或数据驱动测试，它是把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格或设计说明书，检查程序的功能是否符合它的功能说明。

2.8 单元测试方法

2.8.1 黑盒测试

一个程序P有输入X和Y及输出Z：

在字长为32位的计算机上运行。如果X，Y只取整数，考虑把所有的X、Y值都作为测试数据，这样做可能采用的测试数据组 (X_i, Y_i) 的最大可能数目为： $2^{32} \times 2^{32} = 2^{64}$ 。如果程序P测试一组X，Y数据需要1ms，且一天工作24h，一年工作365天，要完成 2^{64} 组测试，需要5亿年。





➤ 黑盒测试类型

- 边界值测试
- 等价类划分测试
- 输入组合测试
- 因果图测试
- 基于状态测试
- 基于决策表测试

2.8 单元测试方法

2.8.1 黑盒测试

➤ 2.8.1.1 边界值

□ 定义

- ✓ 针对各种边界情况设计测试用例

□ 设计原则

- ✓ 边界值分析的基本思想是使用在最小值、略高于最小值、正常值、略低于最大值和最大值处取输入变量值。

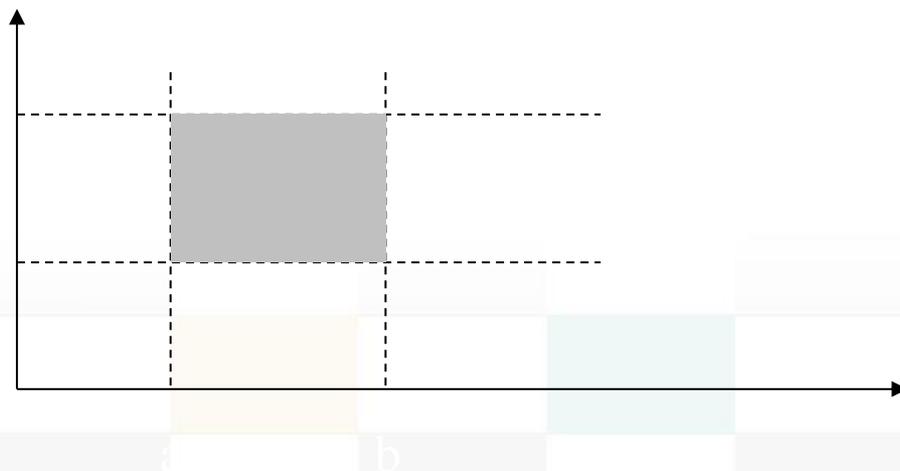


➤ 2.8.1.1 边界值

如果函数F实现一个程序，则输入两个变量X1和X2会有一些边界：

$$a \leq X1 \leq b$$

$$c \leq X2 \leq d。$$





2.8 单元测试方法

2.8.1 黑盒测试

➤ 2.8.1.1 边界值

根据上述原则，对于两变量函数F的边界值分析测试用例是：

$\langle X1_{nom}, X2_{min} \rangle \langle X1_{nom}, X2_{min+} \rangle \langle X1_{nom}, X2_{nom} \rangle \langle X1_{nom}, X2_{max} \rangle \langle X1_{nom}, X2_{max-} \rangle$

$\langle X1_{min}, X2_{nom} \rangle \langle X1_{min+}, X2_{nom} \rangle \langle X1_{nom}, X2_{nom} \rangle \langle X1_{max}, X2_{nom} \rangle \langle X1_{max-}, X2_{nom} \rangle$



➤ 2.8.1.1 边界值

例子：

三角形问题接受三个整数a、b和c作为输入，用做三角形的边。程序的输出是由这三条边确定的三角形类型：等边三角形、等腰三角形、不等边三角形。整数a、b、c必须满足以下条件：

$$1 \leq a \leq 200$$

$$1 \leq b \leq 200$$

$$1 \leq c \leq 200$$

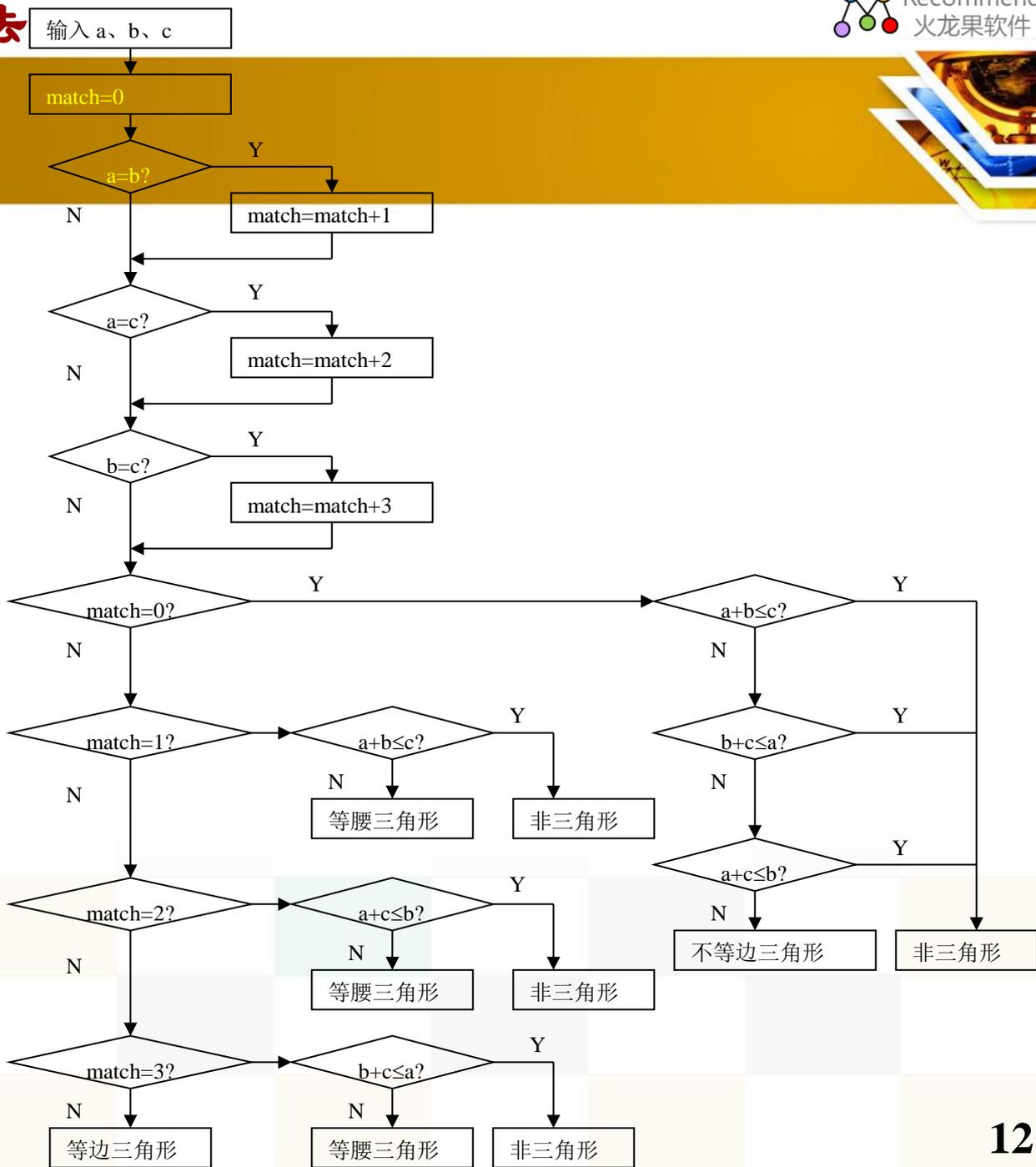
$$a < b + c$$

$$b < a + c$$

$$c < a + b$$



第2章 软件测试方法





➤ 2.8.1.1 边界值

□ 根据设计原则设计测试用例

第2章 软件测试方法

编号	a	b	c	预期输出
1	100	100	1	等腰三角形
2	100	100	2	等腰三角形
3	100	100	100	等边三角形
4	100	100	199	等腰三角形
5	100	100	200	非三角形
6	100	1	100	等腰三角形
7	100	2	100	等腰三角形
8	100	100	100	等边三角形
9	100	199	100	等腰三角形
10	100	200	100	非三角形
11	1	100	100	等腰三角形
12	2	100	100	等腰三角形
13	100	100	100	等边三角形
14	199	100	100	等腰三角形
15	200	100	100	非三角形



➤ 2.8.1.1 边界值

□ 变异：健壮性测试

□ 健壮性测试是边界值分析的一种简单扩展：除了变量的五个边界值分析取值，还要通过采用一个略超过最大值（max+），以及略小于最小值（min-）的取值



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 在数学中，给定一个集合 X 和在 X 上的一个等价关系 \sim ，则 X 中的一个元素 a 的等价类是在 X 中等价于 a 的所有元素的子集：
$$a = \{ X ; X X \sim a \}$$

➤ 如果 X 是轿车的集合，而 \sim 是“颜色相同”的等价类，则一个特定等价类由所有绿色轿车组成。 X / \sim 自然的被认同于所有轿车颜色的集合。



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

- 把所有可能的输入数据，即程序的输入域划分成若干个子集合，然后从每个子集合中选取少数有代表性的数据做为测试用例。
- 等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的，并合理地假定：测试某等价类的代表值就等于对这一类其它值的测试，因此，可以把全部输入数据合理划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件就可以用少量代表性的测试数据取得较好的测试结果。等价类划分可有两种不同的情况：有效等价类和无效等价类。



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 设计步骤总结

- ① 为每一个等价类规定一个唯一的编号.
- ② 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.
- ③ 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.



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 例子2：保险费率计算

人人保险公司承担人寿保险已有多年历史，该公司保费计算方式为投保额 * 保险率，保险率又依点数不同而有别，10点以上费率为0.6%，10点以下费率为0.1%



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 例子2：保险费率计算

程序界面：

保险费率计算

投保额	<input style="width: 90%;" type="text"/>	年龄	<input style="width: 90%;" type="text"/>
性别	<input style="width: 90%;" type="text" value="请选择"/> ▼	婚姻	<input style="width: 90%;" type="text" value="请选择"/> ▼
抚养人数	<input style="width: 90%;" type="text"/>		
<input style="border: 1px solid gray; padding: 5px 20px;" type="button" value="计算保险费率"/>			
点数	<input style="width: 90%;" type="text"/>	保险费率	<input style="width: 90%;" type="text"/>

2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 例子2：保险费率计算：输入数据说明

年龄	20 ~ 39	6点
	40 ~ 59	4点
	60岁以上20岁以下	2点
性别	男	5点
	女	3点
婚姻	已婚	3点
	未婚	5点
抚养人数	1人扣0.5点，最多3点(四舍五入取整数)	



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 例子2：保险费率计算：输入数据说明

投保额：1000元至100000元，只能输入数字

年龄：只能输入数字

抚养人数：只能输入数字



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 例子2：保险费率计算

一、划分等价类（见文档）

二、设计测试用例（测试数据）



2.8 单元测试方法

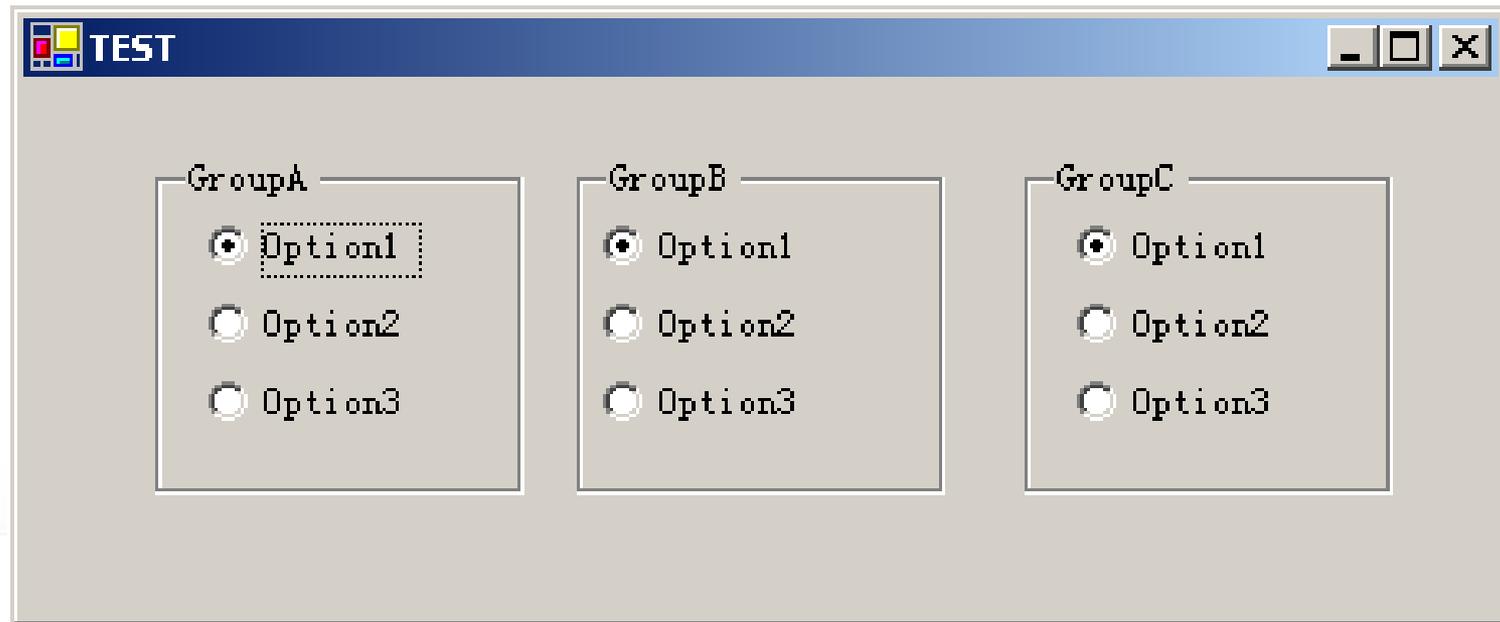
2.8.1 黑盒测试

2.8.1.2 等价类划分

➤ 习题：三角形判定函数



2.8.1.3 输入组合





2.8.1.3 输入组合

各种变量可能的取值								
A			B			C		
1	2	3	1	2	3	1	2	3

2.8 单元测试方法

2.8.1 黑盒测试



ID	A	B	C
1	1	1	1
2	1	1	2
3	1	1	3
4	1	2	1
5	1	2	2
6	1	2	3
7	1	3	1
8	1	3	2
9	1	3	3

ID	A	B	C
10	2	1	1
11	2	1	2
12	2	1	3
13	2	2	1
14	2	2	2
15	2	2	3
16	2	3	1
17	2	3	2
18	2	3	3

ID	A	B	C
19	3	1	1
20	3	1	2
21	3	1	3
22	3	2	1
23	3	2	2
24	3	2	3
25	3	3	1
26	3	3	2
27	3	3	3



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.3 组合测试

- 正交试验(见单独讲义)
- 组合覆盖

2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.3 组合测试

➤ 组合覆盖（全对偶）





2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.3 组合测试

➤ 组合覆盖 (全对偶)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
粗体	c	c	c	c	u	u	u	c	u	u	c	c	u	c	u	u
斜体	u	c	c	c	c	c	c	c	u	u	u	u	u	u	c	u
底纹	u	u	c	c	c	c	c	u	c	c	c	c	u	u	u	u
下划线	u	u	u	c	u	u	c	c	u	c	u	c	c	c	c	u



2.8.1.3 组合测试

➤ 组合覆盖（全对偶）

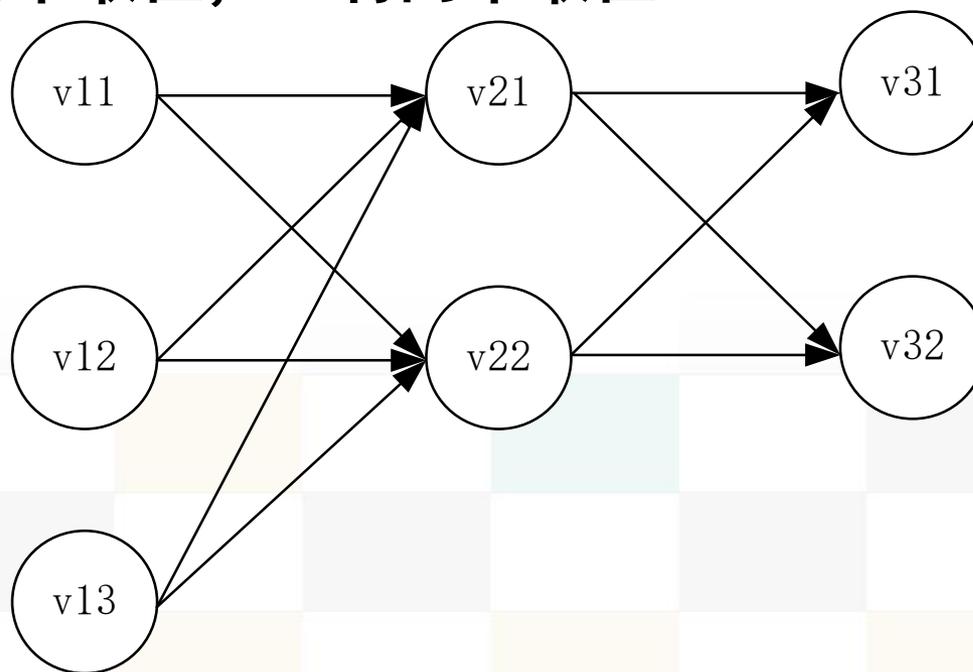
➤ 效果：例如，一个拥有 20 个参数的系统，每个参数有 5 个取值，则需要 $5^{20} = 95367431640625$ 个测试组合数据。而配对组合测试集仅需要 45 个即可覆盖所有配对。



2.8.1.3 组合测试

► 组合覆盖（全对偶）：

□ 一个详细例子：一个函数有三个参数 v_1, v_2, v_3 ; v_1 有三个取值， v_2 有两个取值， v_3 有两个取值。





2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.3 组合测试

➤ 组合覆盖（全对偶）：

- 一个详细例子：一个函数有三个参数 v_1, v_2, v_3 ; v_1 有三个取值， v_2 有两个取值， v_3 有两个取值。
- 黑边演示推导过程



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.3 组合测试

➤ 组合覆盖（全对偶）：

- 在上述列子中再增加一个变量 v_4 （两个取值）
- 黑边演示推导过程



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.3 组合测试

➤ 组合覆盖（全对偶）：

- 在上述列子中再增加一个变量v5（两个取值）
- 黑边演示推导过程



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.3 组合测试

➤ 组合覆盖（全对偶）：

- 在上述列子中再增加一个变量v6（两个取值）
- 黑边演示推导过程



2.8.1.3 组合测试

➤ 组合覆盖（全对偶）：

- 多个变量和多个取值呢？
- 需要配对组合算法来解决！



2.8.1.3 组合测试

➤ 组合覆盖（全对偶）（要求了解）：

□ AETG策略：

AETG(Automatic Efficient Testcase Generator)是以一个空的测试集开始。每次往测试集加入一个测试案例。为了得到一个新的测试案例，系统首先根据贪心算法产生一组候选案例，然后选择其中能覆盖最多未覆盖配对的案例。

详细介绍见论文“配对组合的软件测试方法研究与实现.pdf”



2.8.1.3 组合测试

➤ 组合覆盖（全对偶）（要求了解）：

□ IPO 配对组合测试集生成策略

不同于AETG，IPO策略的基本思想是以参数为对象，初始时先生成两个参数所有配对组合的测试案例集合T。随后每次加入一个参数，通过扩展算法覆盖该参数并“测试集内已有参数组成的新的配对集合，并尽可能使测试集保持较优的大小。如此进行直至所有的参数都加入到测试集中。测试集合T的扩展分为两个部分：水平扩展和垂直扩展。水平扩展是在现有测试案例的基础上加入新参数的某一个取值，不产生新的测试案例；垂直扩展在水平扩展之后进行，对仍未被覆盖的配对组合生成新的测试案例将其覆盖。

详细介绍见论文“配对组合的软件测试方法研究与实现.pdf”



2.8.1.3 组合测试

➤ 组合覆盖（全对偶）（要求了解）：

□ 优先级参数配对组合测试集生成策略

见论文“优先级参数配对组合测试集生成策略.pdf”



2.8.1.3 组合测试

➤ 组合覆盖 (要求了解) :

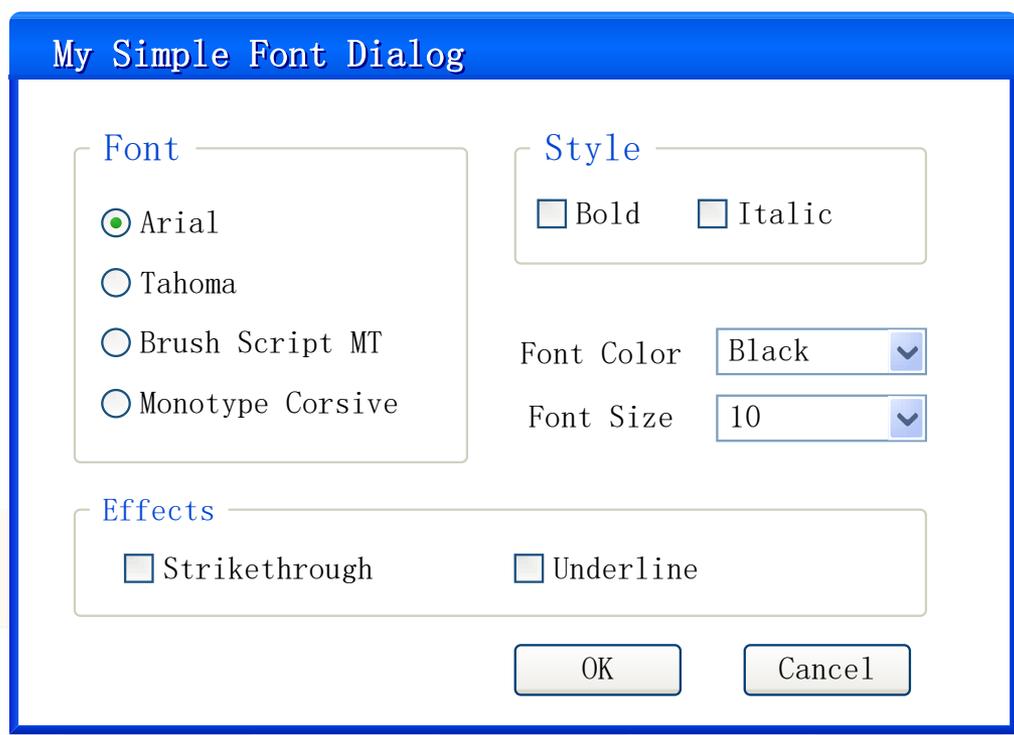
□ 多元配对组合测试算法

见论文“多元配对组合测试的算法研究与实现.pdf”



2.8.1.3 组合测试

➤ 组合覆盖（全对偶）： PICT工具产生数据





2.8.1.3 组合测试

➤ 组合覆盖（全对偶）

□ PICT工具产生数据（实际操作演示，注意Style和Effects改变形式）



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.4 边界值向等价类划分转化

➤ 例子： $a < x_1 < b; c < x_2 < d$ （黑板板书）



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.5 等价类划分向输入组合转化

➤ 例子：nextday（黑板板书），正交试验，PICT



2.8.1.6 因果图测试

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。

如果在测试时必须考虑输入条件的各种组合，则可能的组合数目将是天文数字，因此必须考虑采用一种适合于描述多种条件的组合、相应产生多个动作的形式来进行测试用例的设计，这就需要利用因果图（逻辑模型）。



2.8.1.6 因果图测试

➤ 因果图设计测试用例思想

首先从程序规格说明书的描述中,找出因(输入条件)和果(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.



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.6 因果图测试

➤ 因果图法定义

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。



2.8.1.6 因果图测试

► 因果图测试步骤

(1) 分析软件规格说明描述中，哪些是原因（即输入条件或输入条件的等价类），哪些是结果（即输出条件），并给每个原因和结果赋予一个标识符；

(2) 分析软件规格说明描述中的语义，找出原因与结果之间，原因与原因之间对应的是有什么关系？根据这些关系，画出因果图；

(3) 由于语法或环境限制，有些原因与原因之间，原因与结果之间的组合情况不可能出现。为表明这些特殊情况，在因果图上用一些记号标明约束或限制条件；

(4) 把因果图转换成判定表；

(5) 把判定表的每一列拿出来作为依据，设计测试用例。



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.6 因果图测试

➤ 因果图画法

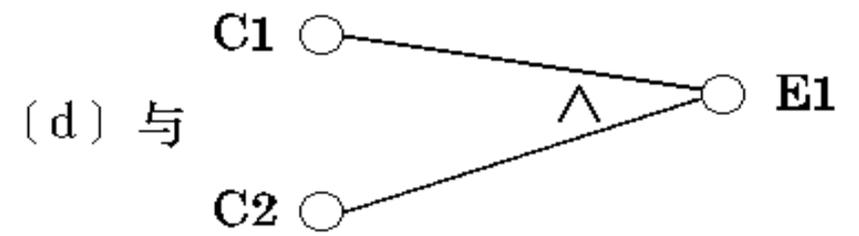
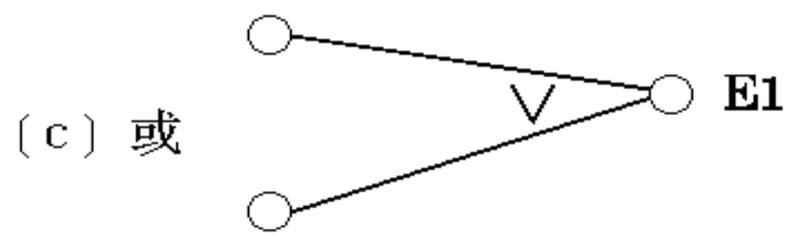
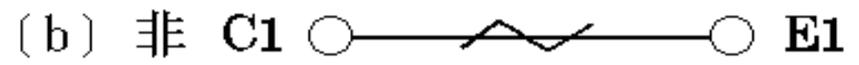
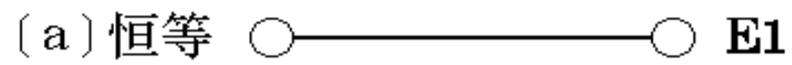


图1 因果图的基本符号



2.8.1.6 因果图测试

► 因果图中基本图形介绍

如图1中所示， c_i 表示原因，一般置于图左部； e_i 表示结果，一般置于图右部。 c_i 和 e_i 都可以取值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 为1，则 e_1 也为1；否则， e_1 为0。

(2) 非：若 c_1 为1，则 e_1 为0；否则 e_1 为1。

(3) 或：若 c_1 或 c_2 或 c_3 为1，则 e_1 为1；否则 e_1 为0；“或”可有任意个输入。

(4) 与：若 c_1 和 c_2 都为1，则 e_1 为1；否则 e_1 为0。“与”也可有任意个输入



2.8.1.6 因果图测试

➤ 因果图中基本图形介绍

因果图中使用了简单的逻辑符号，以直线连接左右结点，左结点表示输入状态（或称原因），右结点表示输出状态（或称结果），而在实际问题中，输入条件相互之间还可能存在着某些依赖关系，我们称之为“约束”，比如某些输入条件本身不可能同时出现，输出状态间也往往存在着约束。在因果图中用特定符号表明这些约束，见下图：



2.8.1.6 因果图测试

➤ 因果图中基本图形介绍

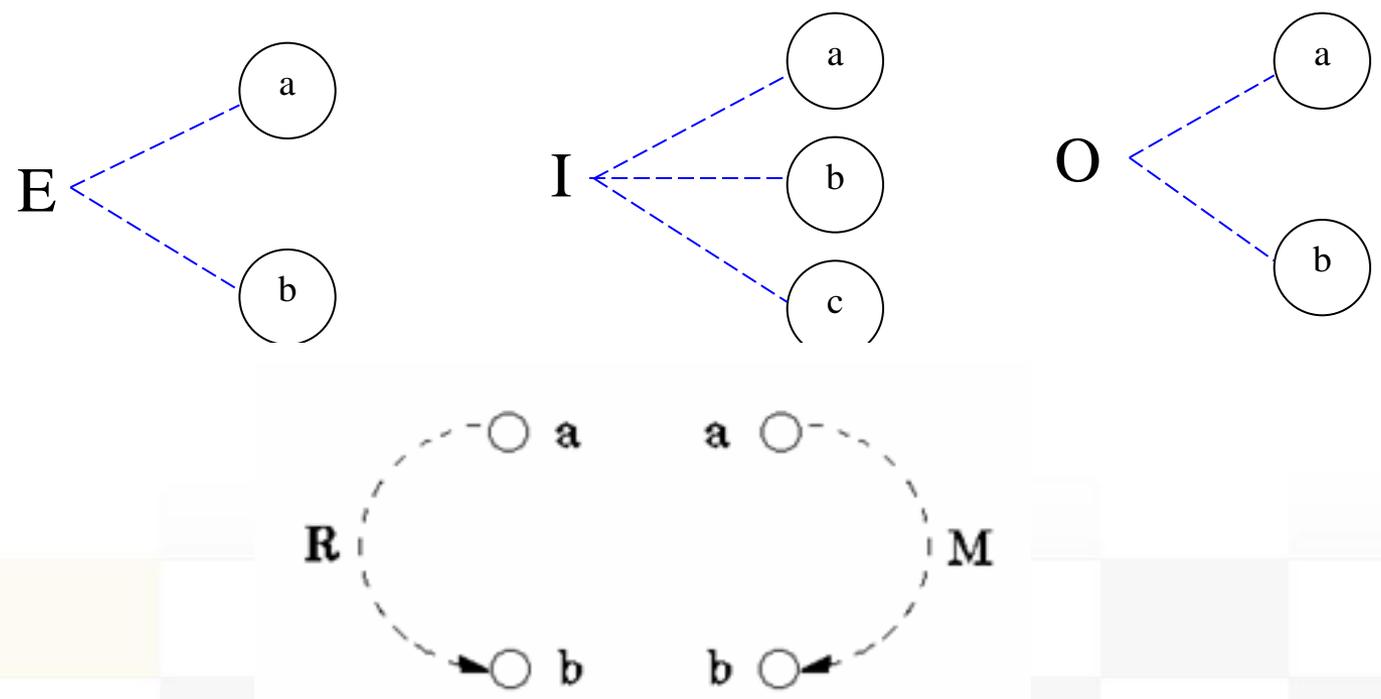


图2约束符号



2.8.1.6 因果图测试

► 因果图中基本图形介绍

- (1) E约束（异）：a、b中至多有一个可能为1，即a、b不可能同时为1。
- (2) I约束（或）：a、b、c中至少有一个必须为1，即a、b、c不能同时为0。
- (3) O约束（唯一）：a、b中必须有一个，且仅有一个为1。
- (4) R约束（要求）：a是1时，b必须是1，即不可能出现a是1时，b是0。
- (5) M约束（强制）：如果结果a为1，则结果b强制为0。a为零时，b值不定。

注意：1—5中只有5是对结果的约束。



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.6 因果图测试（例子1）

如想对文件进行修改，输入的第一列字符必须是A或B，第二列字符必须是一个数字，如果第一列字符不正确，则给出信息L；如果第二列字符不正确，则给出信息M。

2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.6 因果图测试（例子1）

第一步：分析了上面的规格说明的要求后，我们可以很明确的把原因和结果分开。

原因：

1. 第一列字符为A
2. 第一列字符为B
3. 第二列字符为一数字

结果：

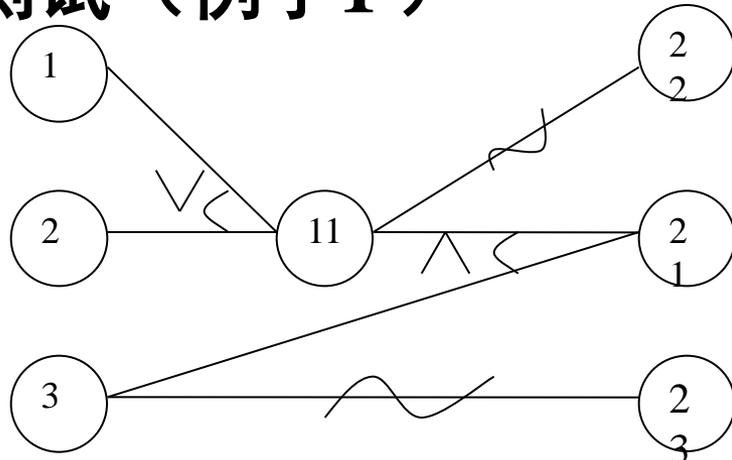
21. 修改文件
22. 给出信息 L
23. 给出信息 M

第二步：这个例子规格说明中，很明确的给出了原因和结果之间的对应关系，将原因和结果根据它们之间的对应关系用相应的逻辑符号连接起来，画出因果图如下：

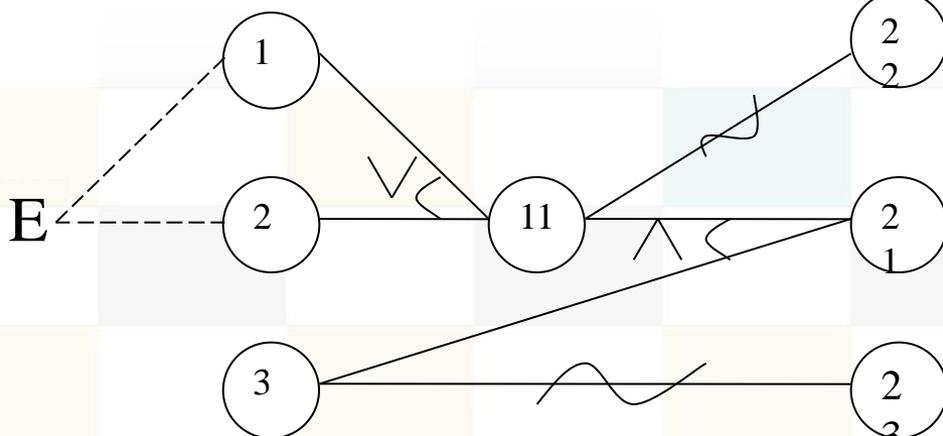


2.8 单元测试方法

2.8.1.6 因果图测试 (例子1)



上图中左列表示原因，右列表示结果，编号为11的结点是导出结果的进一步原因。考虑到原因1和2不可能同时为1（即第一列不能同时为A和B），我们在图上可对其施加E约束，这样就有了具有约束的因果图，如下：



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.6 因果图测试 (例子1)

第三步：把因果图转换成判定表

规则 桩									
		1	2	3	4	5	6	7	8
条件 (原因) 桩	1	1	1	1	1	0	0	0	0
	2	1	1	0	0	1	1	0	0
	3	1	0	1	0	1	0	1	0
	11	——	——	1	1	1	1	0	0
动作 (结果) 桩	22	——	——	0	0	0	0	1	1
	21	——	——	1	0	1	0	0	0
	23	——	——	0	1	0	1	0	1

注：规则是指任何一个条件组合的特定取值及其相应要执行的操作.在判定表中贯穿条件项和动作项的一列就是一条规则.显然,判定表中列出多少组条件取值,也就有多少条规则,既条件项和动作项有多少列.由于原因1和2不可能同时为1,故规则1、2属于不可能发生组合,编辑测试用例时可以考虑不用考虑。



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.6 因果图测试（例子1）

条件桩（Condition Stub）：列出了问题得所有条件.通常认为列出得条件的次序无关紧要.

动作桩（Action Stub）：列出了问题规定可能采取的操作.这些操作的排列顺序没有约束.

条件项（Condition Entry）：列出针对它左列条件的取值.在所有可能情况下的真假值.

动作项（Action Entry）：列出在条件项的各种取值情况下应该采取的动作.



2.8.1.6 因果图测试（例子1）

第四步：把判定表的每一列拿出来作为依据，设计测试用例,如下：

规则	3	4	5	6	7	8
用例	A2、A9	AM、A?	B2、B9	BM、B?	C7、D8	DE、XY

注：规则3中A2表示输入的第一列为A,第二列为数字（见表中规则3），“？”表示除数字外的任何值。



2.8.1.6 因果图测试（例子2）

有一个处理单价为5角的盒装饮料的自动售货机软件。若投入5角硬币，按下“可乐”，“雪碧”或“红茶”按钮，相应的饮料就送出来。若投入的是1元硬币，在送出饮料的同时退还5角硬币。

2.8.1.6 因果图测试（例子2）

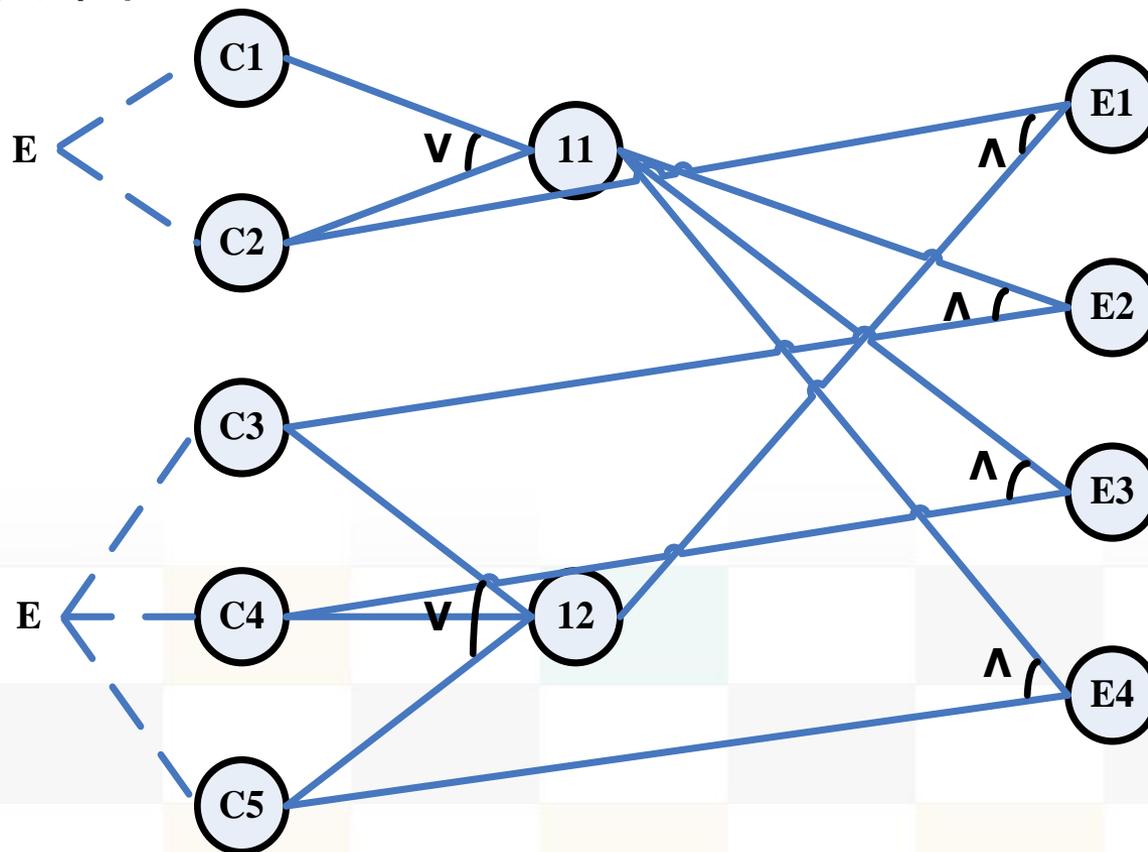
第一步分析原因及结果

原因	<p>c1:投入5角硬币;</p> <p>c2:投入1元硬币;</p> <p>c3:按“可乐”按钮;</p> <p>c4:按“雪碧”按钮;</p> <p>c5:按“红茶”按钮;</p>
结果	<p>a1:退还5角硬币;</p> <p>a2:送出“可乐”饮料;</p> <p>a3:送出“雪碧”饮料;</p> <p>a4:送出“红茶”饮料;</p>



2.8.1.6 因果图测试 (例子2)

第二步画出因果图





2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.6 因果图测试（例子2）

第三步判定表

	1	2	3	4	5	6	7	8	9	10	11
c1:投入5角硬币	1	1	1	1	0	0	0	0	0	0	0
c2:投入1元硬币	0	0	0	0	1	1	1	1	0	0	0
c3:按“可乐”按钮	1	0	0	0	1	0	0	0	1	0	0
c4:按“雪碧”按钮	0	1	0	0	0	1	0	0	0	1	0
c5:按“红茶”按钮	0	0	1	0	0	0	1	0	0	0	1
11: 已投币	1	1	1	1	1	1	1	1	0	0	0
12: 已按钮	1	1	1	0	1	1	1	0	1	1	1
a1:退还5角硬币					√	√	√				
a2:送出“可乐”饮料	√				√						
a3:送出“雪碧”饮料		√				√					
a4:送出“红茶”饮料			√				√				

2.8 单元测试方法 2.8.1 黑盒测试



2.8.1.6 因果图测试（例子2）

第四步设计测试用例

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



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.6 因果图测试的优点

- (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，高效率的开发测试用例。
- (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。



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.6 因果图测试的缺点

输入条件的组合数 2^a 随 a 的上升急剧增长，当 a 较大时，因果图的结构将变得十分复杂，而把因果图转换为判定表则更为麻烦。



2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.6 因果图测试（例子3，习题）

分析中国象棋中走马的实际情况（下面未注明的均指的是对马的说明）

- 1、如果落点在棋盘外，则不移动棋子；
- 2、如果落点与起点不构成日字型，则不移动棋子；
- 3、如果落点处有自己方棋子，则不移动棋子；
- 4、如果在落点方向的邻近交叉点有棋子（绊马腿），则不移动棋子；
- 5、如果不属于1-4条，且落点处无棋子，则移动棋子；
- 6、如果不属于1-4条，且落点处为对方棋子(非老将)，则移动棋子并除去对方棋子；
- 7、如果不属于1-4条，且落点处为对方老将，则移动棋子，并提示战胜对方，游戏结束。

2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.7 基于决策表的测试

- 在所有功能测试方法中，基于决策表的测试方法是最严格的，因为决策表具有逻辑严格性。
- 决策表很适合描述不同条件集合下采取行动的若干组合的情况。
- 决策表的完备性保证一种完备的测试。



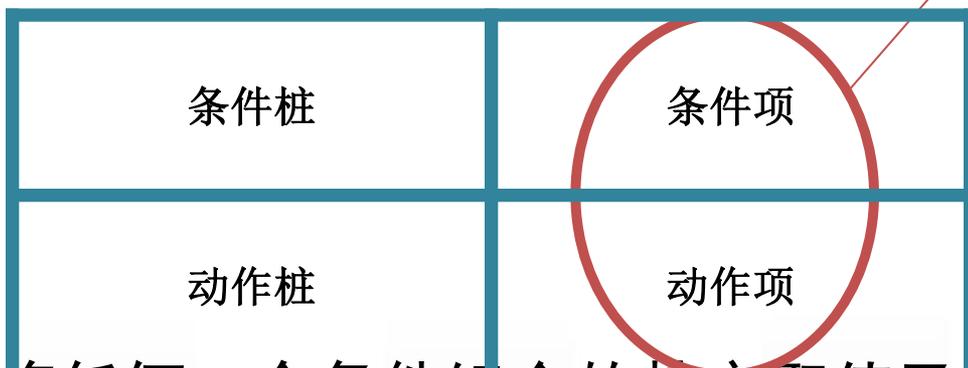
2.8.1.7 基于决策表的测试

► 决策表的组成：

- 1.条件桩：列出了问题的所有条件。
- 2.动作桩：列出了问题规定可能采取的操作。
- 3.条件项：列出针对它所列条件的取值，在所有可能情况下的真假值。
- 4.动作项：列出在条件项的各种取值情况下应该采取的动作。
- 5.规则：任何一个条件组合的特定取值及其相应要执行的操作。
在决策表中贯穿条件项和动作项的一列就是一条规则。

2.8.1.7 基于决策表的测试

➤ 任何一个条件组合的特定取值及其相应要执行的操作。在决策表中贯穿条件项和动作项的一列就是 **规则** 一条规则。



➤ **【规则】** 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。

2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.7 基于决策表的测试

桩	规则1	规则2	规则3、4	规则5	规则6	规则7、8
c1	T	T	T	F	F	F
c2	T	T	F	T	T	F
c3	T	F	-	T	F	-
a1	X	X		X		
a2	X				X	
a3		X		X		
a4			X			X



2.8 单元测试方法 2.8.1 黑盒测试

2.8.1.7 基于决策表的测试

➤ 任何一个条件组合的特定取值及其相应要执行的操作。在决策表中贯穿条件项和动作项的一列就是一条规则。

2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.7 基于决策表的测试

▶ 三角形问题决策表

桩	1	2	3	4	5	6	7	8	9
C1: a,b,c构成三角形?	N	Y	Y	Y	Y	Y	Y	Y	Y
C2: a=b?	—	Y	Y	Y	Y	N	N	N	N
C3: a=c?	—	Y	Y	N	N	Y	Y	N	N
C4: b=c?	—	Y	N	Y	N	Y	N	Y	N
A1: 非三角形	X								
A2: 不等边三角形									X
A3: 等腰三角形					X		X	X	
A4: 等边三角形		X							
A5: 不可能			X	X		X			

2.8 单元测试方法

2.8.1 黑盒测试

2.8.1.7 基于决策表的测试

➤ 经过修改的三角形问题决策表

桩	1	2	3	4	5	6	7	8	9
C1:a<b+c?	N	Y	Y	Y	Y	Y	Y	Y	Y
C2:b<a+c?	—	N	Y	Y	Y	Y	Y	Y	Y
C3:c<a+b?	—	—	N	Y	Y	Y	Y	Y	Y
C4:a=b吗?	—	—	—	Y	Y	Y	Y	N	N
C5:a=c吗?	—	—	—	Y	Y	N	N	Y	Y
C6:b=c吗?	—	—	—	Y	N	Y	N	Y	N
A1: 非三角形	X	X	X						
A2: 不等边三角形									
A3: 等腰三角形							X		X
A4: 等边三角形				X					
A5: 不可能					X	X		X	