

第三讲：黑盒测试技术

LIAN YU
THE SCHOOL OF SOFTWARE AND MICROELECTRONICS
PEKING UNIVERSITY
NO.24 JINYUAN RD, BEIJING 102600

提 纲

1

黑盒测试

2

等价划分

3

边界值分析

4

因果分析法

5

正交数组测试

6

测试插桩

7

总结

黑盒测试

- 黑盒测试又叫做功能测试，是基于系统已实现的功能进行测试的。使用该方法的具体的测试用例设计方法包括等价类划分法、边界值分析法、正交数组测试法、因果分析法等。
 - 黑盒测试注重于测试软件的功能性需求，也即黑盒测试需要软件工程师生成输入条件集来检测程序所有功能需求。
 - 黑盒测试并不是白盒测试的替代品，而是配合白盒测试发现其他类型的错误。
- 黑盒测试有助于测试人员解决在测试过程中的以下问题：
 - 功能的有效性如何测试？
 - 系统的行为如何测试？
 - 哪一类的输入会形成好的测试用例？ (Partition)
 - 系统是否会对某些输入特别敏感？
 - 数据分类的界限怎么被隔离？ (Boundary)
 - 数据的特殊组合对系统操作产生什么效果？ (Combination)

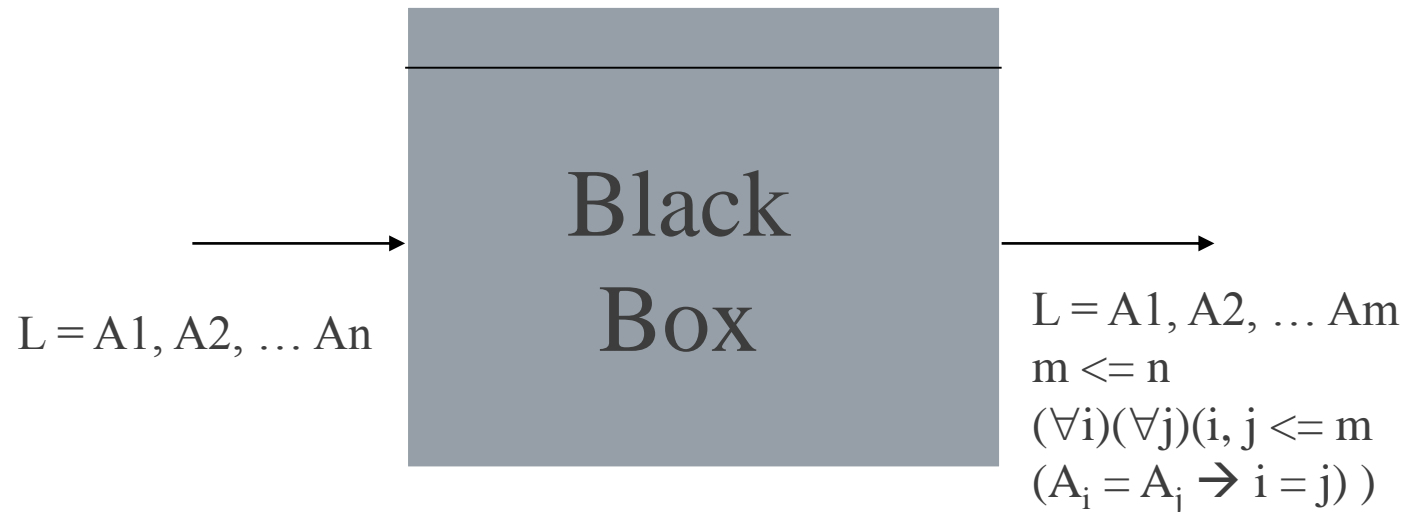
BLACK BOX EXAMPLE

Procedure purge (var L:list)

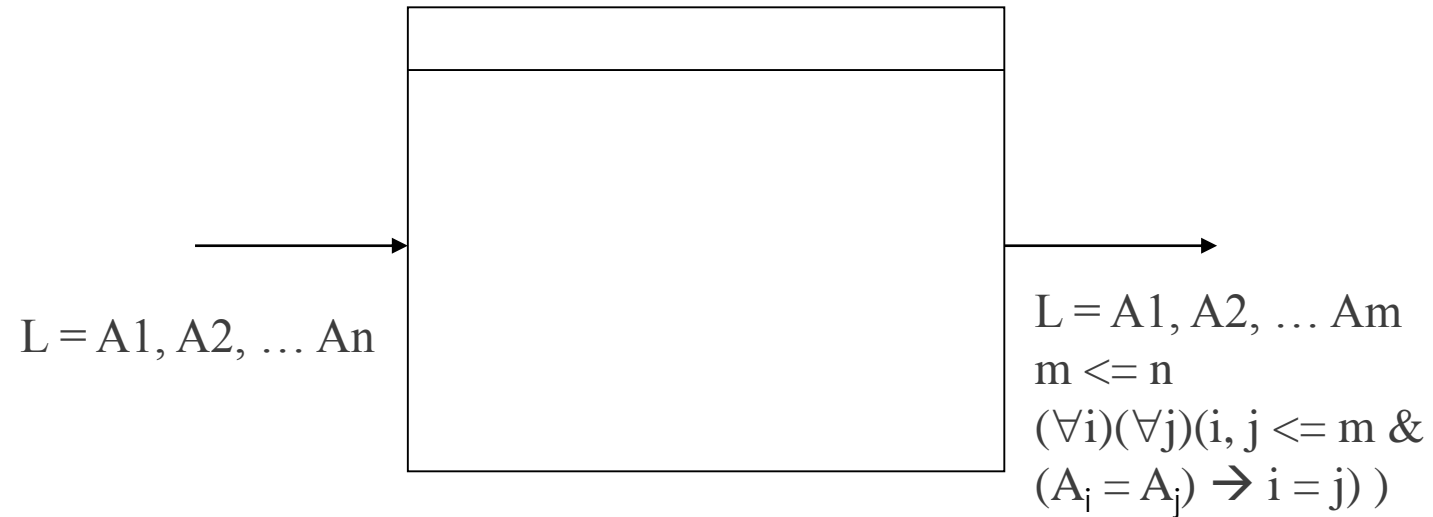
var p: ...

- (1) begin p:= FIRST(L);
- (2) while P <> END(L) do
- (3) begin q:= next(p,L);
- (4) while q <> END(L) do
- (5) if Aq = Ap then
- (6) delete (Aq, L)
- (7) else q:= next(q,L);
- (8) end
- (9) p := next(p,L)
- (10) end;

Black box testing is based on the function performed.



BLACK BOX/FUNCTIONAL TESTING



$L = ()$, $L = (A_p)$, $L = (A_p \ A_p)$, $L = (A_p \ A_q)$,

$L = (A_p \ A_q \ A_p)$

TEST CASES ANALYSIS

- $L = A_1, A_2, \dots, A_m, m \leq n$
 - case 1: $n=0, m=n$, the input is an empty list
 - case 2: $n=1$, the input is a single element list
 - $m < n$, not possible
 - $m = n$
 - case 3: $n > 1$
 - $m < n$, list L contains duplicate elements
 - $m = n$, list L does not contain duplicate elements

TEST CASES ANALYSIS

- $(\forall i)(\forall j)(i, j \leq m \ \& \ (A_i = A_j) \rightarrow i = j)$

- Negation of above is

$$(\exists i)(\exists j)(i, j \leq m \ \& \ (A_i = A_j) \ \& \ i \neq j)$$

- This also suggests a list containing duplicate elements

黑盒测试（续）

- 实际上，这个例子中的前置条件并不足够，其正确的前置条件至少需要表明以下几条：
 - 原始列表中的每一个元素会出现在结果列表中；
 - 结果列表中的每一个元素也会出现在原始列表中；
 - 结果列表中元素的排列与他们出现在原始列表中的顺序相同；
- 简便起见，我们省去了上述条件。在实际测试中应注意。

提 纲

- 黑盒测试
- ➔ ■ 等价划分
- 边界值分析
- 因果分析法
- 正交数组测试
- 测试插桩
 - 测试预言
 - 随机数据生成器

等价类关系

- 假设 $S=\{a_1, a_2, \dots, a_n\}$ 是一系列元素的集合，如果 R 满足以下条件，则 $R \subseteq S \times S$ 是一个等价类关系：
 - 自反性: $\langle a_i, a_i \rangle \in R$;
 - 对称性: $\langle a_i, a_j \rangle \in R \implies \langle a_j, a_i \rangle \in R$;
 - 传递性: $\langle a_i, a_j \rangle \in R \wedge \langle a_j, a_k \rangle \in R \implies \langle a_i, a_k \rangle \in R$ 。
- 例如：
 - 1) $S=\{0, 1, 2, \dots\}$, $(x-y)\%3=0$, 则 $\langle x, y \rangle \in R$;
 - 2) $S=\{\text{Human}\}$, $\langle x, y \rangle \in R$ 生活在同一城市中的关系;
 - 3) $S=\{E\}$, $\langle x, y \rangle \in R$ 拥有相同属性的关系。

等价类划分法

- 等价类划分法是黑盒测试的一种方法，它把一个程序的输入域划分成数据类集合，从而生成测试用例。
- 一个理想的测试用例是指可单独地发现**一类**错误(例如:所有字符数据的错误处理)。否则在这种错误被观察之前，需要执行很多的用例。
 - 等价类划分法试图生成那种可以揭露一类错误的测试用例，从而减少必须生成的测试用例的总数。
- 等价类划分法的测试用例设计是基于对输入条件的等价类评估。

等价类划分法（续）

- 使用前面介绍的概念，如果对象由具有对称性、传递性或自反性的关系连接，就存在等价类 [BEI95] 。
- 等价类表示输入条件的一组有效或无效的状态。典型地，输入条件通常是
 - 一个特定的数值，
 - 一个数值域，
 - 一组相关值
 - 一个布尔条件

定义等价类指南

- (1) 如果输入条件规定了一个范围，可以确立一个有效等价类和两个无效等价类。
- 例如：输入范围为0~100，则确立的等价类为：
 - 则有效等价类为0~100间的整数
 - 无效等价类为
 - 负整数
 - 大于100的整数。

定义等价类指南（续）

- （2）如果输入条件是一个明确的值，可以确立一个有效等价类和两个无效等价类。
 - 例如：期望的输入值为10，
 - 则有效等价类为{10}，
 - 无效等价类为
 - 所有小于10的整数
 - 所有大于10的整数。

定义等价类指南（续）

- (3) 如果输入条件是一个集合的成员，可以确立一个有效等价类和一个无效等价类。
 - 例如：期望的输入条件是“SunOS”，
 - 则有效等价类为{“SunOS”}，
 - 无效等价类为所有的集合：not in {“SunOS”}。

定义等价类指南（续）

- （4）如果输入条件是一个布尔量，可以确立一个有效等价类和一个无效等价类。
 - 例如：
 - 允许的最低GRE成绩，
 - 电话号码可以根据区号来划分等价类，
 - 装船出货可以根据服务和区域来划分等价类。
- 课堂练习：列举几个输入条件和等价类划分的其他例子。

提 纲

- 黑盒测试
- 等价划分
- ➡ ■ 边界值分析
- 因果分析法
- 正交数组测试
- 测试插桩
 - 测试预言
 - 随机数据生成器

边界值分析法

- 边界值分析法是用来补充等价类划分法的一种测试用例的生成方法。
 - 它是在类的“边界”上选择测试用例，
 - 而且也是根据输出域得到测试用例。
- 边界值分析法指南：
 - 如果是一个范围 $[a, b]$ ，那么 a 、 b 、大于 a 、小于 a 、大于 b 、小于 b 的值都应进行测试。
 - 如果是一组数值，那么最小值、最大值、比最小值和最大值大、比最小值和最大值小的数值都应进行测试。
 - 把上述两条指南应用于输出条件。
 - 如果内在程序的数据结构已经被规定了边界（例如，只能存100个数值的数组），那么就在它的边界上设计测试用例进行测试。

边界值分析法（续）

- 例如：
- $a < x < b$ ：
 - $x = a + 1, x = b - 1, x = a, x = b, (x = a - 1, x = b + 1)$
- 特殊数值：
 - 0、1、一个极大的数；
 - 产生输出的为零的数值；
 - 空数组、空列表、空栈；
 - 满的数组。

提 纲

- 黑盒测试
- 等价划分
- 边界值分析
- ➡ ■ 因果分析法
- 正交数组测试
- 测试插桩
 - 测试预言
 - 随机数据生成器

因果分析法

- 因果分析法（Cause-Effect Analysis）是一种测试用例的生成方法，它提供了对**逻辑条件**和**对应的动作**（action）的一个简明的表示。
- 因果分析法是一种黑盒测试方法，从分析软件系统需求规格说明书开始，得到因果列表；基于此列表建立决策表，基于决策表的规则生成测试用例。

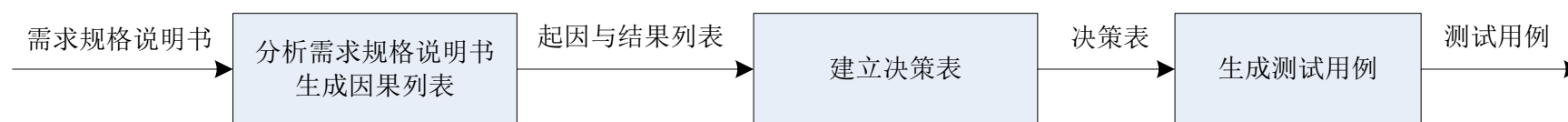


图 2-2用数据流图表示因果分析法的过程。



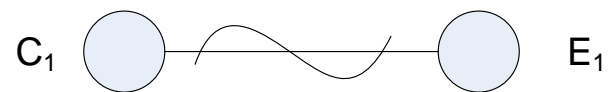
因果图—图形符号

- 在因果图中通常用 C_i 表示原因， E_i 表示结果。
- 原因与结果在图中用节点表示，当原因、结果出现时，称节点状态为“1”，否则为“0”。
- 原因与结果之间的关系有以下四种
 - 恒等（—）：若原因出现，则结果出现；若原因不出现，则结果也不出现（见图（a））。
 - 非（ \sim ）：若原因出现，则结果不出现；若原因不出现，则结果出现（见图（b））。
 - 或（ \vee ）：若几个原因中有一个出现，则结果出现；若几个原因都不出现，则结果不出现（见图（c））。
 - 与（ \wedge ）：若几个原因都出现，则结果出现；若其中有一个原因不出现，则结果不出现（见图（d））。

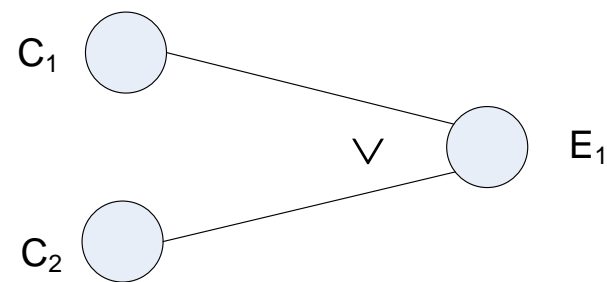
因果图—图形符号（续）



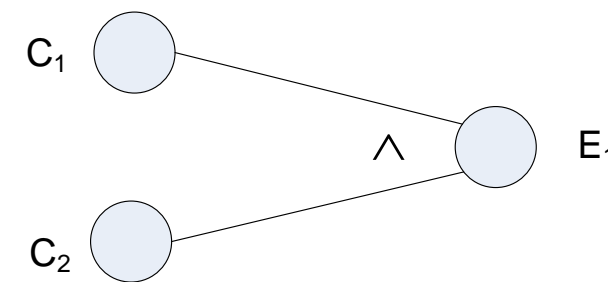
(a) 恒等



(b) 非



(c) 或



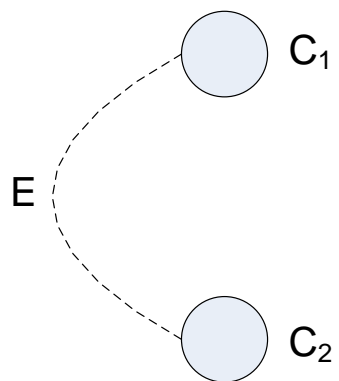
(d) 与

因果图—图形符号（续）

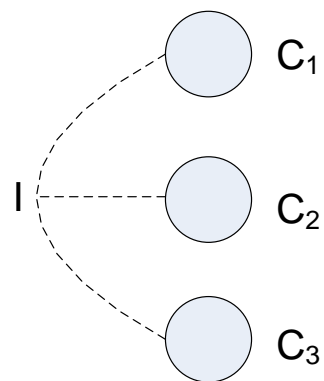
- 为了表示原因与原因之间，结果与结果之间可能存在的约束条件，在因果图中可以附加一些表示约束条件的符号。
- 从输入（原因）考虑，有四种约束，E（互斥）、I（包含）、O（唯一）和R（要求）；
- 从输出（结果）考虑还有一种约束M（屏蔽）

从输入（原因）考虑：	
E（互斥）：	C_1 、 C_2 两个原因不能同时成立，两个中最多有一个能成立（见图（a））。
I（包含）：	C_1 、 C_2 和 C_3 三个原因中至少有一个必须成立（见图（b））。
O（唯一）：	C_1 和 C_2 两个原因中必须有一个成立，且仅有一个成立（见图（c））。
R（要求）：	当 C_1 原因成立时， C_2 原因也必须成立，即不可能 C_1 成立而 C_2 不成立（见图（d））。
从输出（结果）考虑：	
M（屏蔽）：	当 E_1 是1时， E_2 必须是0；而当 E_1 是0时， E_2 的值不定（见图（e））。

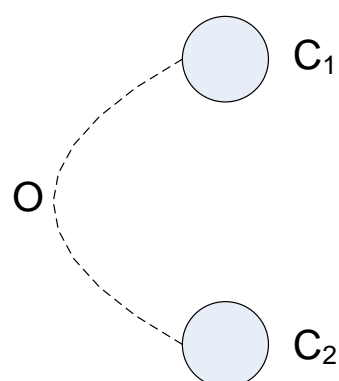
因果图—图形符号（续）



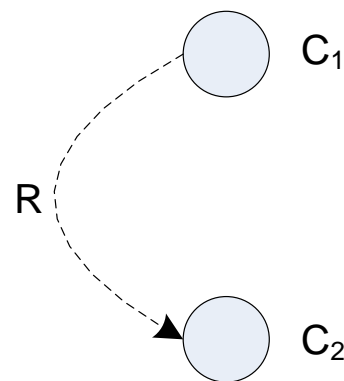
(a) E (互斥)



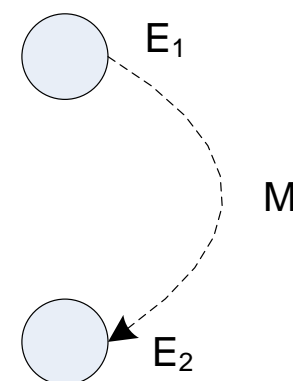
(b) I (包含)



(c) O (唯一)



(d) R (要求)



(e) M (屏蔽)

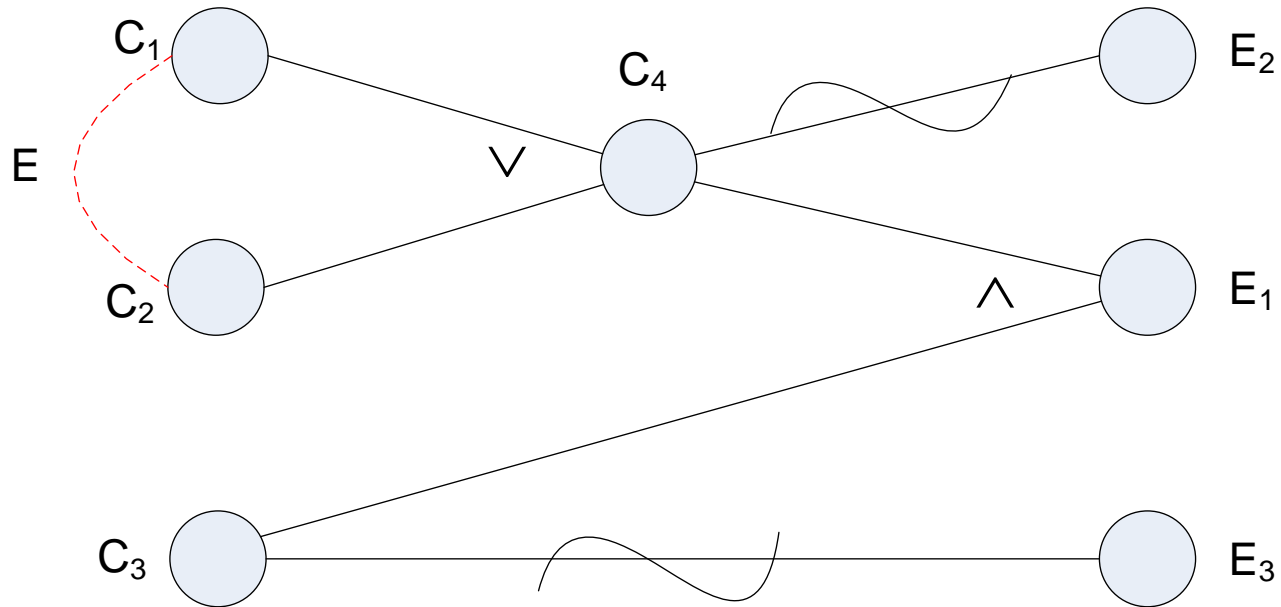
因果图—举例

- 我们假设有以下有关一个文件管理系统的一段规格说明：

- “在文件第一列的字符必须是一个“A”或“B”. 在文件第二列的字符必须是一个数字。在这种情况下，文件是被修改了。如果第一个字符不正确，则打印“X12”消息。如果第二个不是数字，则打印“X13”消息。”

C₁: 第一列的字符是“A”
C₂: 第一列的字符是“B”
C₃: 第二列的字符是数字

E₁: 文件修改过
E₂: 打印消息“X12”
E₃: 打印消息“X13”



原因与结果		组合列举					
		1	2	3	4	5	6
输入 (原因)	C_1	0	0	0	0	1	1
	C_2	0	0	1	1	0	0
	C_3	0	1	0	1	0	1
输出 (结果)	E_1	0	0	0	1	0	1
	E_2	1	1	0	0	0	0
	E_3	1	0	1	0	1	0

条件与行动 \ 决策规则		1	2	3	4	5	6
条件	C_1	0	0	0	0	1	1
	C_2	0	0	1	1	0	0
	C_3	0	1	0	1	0	1
行动	A_1	0	0	0	1	0	1
	A_2	1	1	0	0	0	0
	A_3	1	0	1	0	1	0



生成测试用例

- 测试用例：

- 用例1：

- 输入条件：第一列的字符是“C”,第二列的字符是“X”
 - 输出结果：打印消息“X12”，打印消息“X13”

- 用例4：

- 输入条件：第一列的字符是“B”，第二列的字符是“1”
 - 输出结果：文件修改过

因果分析法（续）

- 下面总结一下利用因果图、决策表的因果分析法执行过程：
 - 分析程序规格说明书，识别哪些是原因，哪些是结果。原因往往是输入条件或是输入条件的等价类，而结果常常为输出条件。
 - 分析程序规格说明书，按其语义，在因果图连接各个原因与其相应的结果。用讲述的四种关系符号（—、~、 \vee 、 \wedge ）来描述因果图中原因与结果之间的关系。
 - 标明约束条件。由于语法或环境的限制，有些原因和结果的组合情况是不可能出现的。对于这些特定的情况，在因果图中使用讲述的五种约束符号（E、I、O、R、M）来标明原因间、结果间的约束条件。

决策表一示例（续）

- 下面总结一下利用因果图、决策表的因果分析法执行过程：（续）
 - 把因果图转换成一个因果图列表进而生成决策表(或判定树)来描述哪种输入组合所引起的哪个执行动作的决策规则。
 - 把决策表的规则转换成测试用例。选择测试数据以便使决策表里的每个规则都被测试。
 - 显而易见，如果决策表已被用作设计工具，那么就不必进行因果分析了，可以直接利用设计时所得到的决策表生成测试用例。

提 纲

- 黑盒测试
- 等价划分
- 边界值分析
- 因果分析法
- ➡ ■ 正交数组测试
- 测试插桩
 - 测试预言
 - 随机数据生成器

正交实验 OR 正交设计法

正交试验设计(Orthogonal experimental design)是研究多因素多水平的一种设计方法，它是根据正交性从全面试验中挑选出部分有代表性的点进行试验，

- 这些有代表性的点具备了“**均匀分散，齐整可比**”的特点，正交试验设计是分式析因设计的主要方法。
- 是一种高效率、快速、经济的实验设计方法。

日本著名的统计学家田口玄一将正交试验选择的水平组合列成表格，称为正交表。例如作一个三因素三水平的实验，按全面实验要求，须进行 $3^3=27$ 种组合的实验，且尚未考虑每一组合的重复数。

- 若按 $L_9(3^3)$ 正交表安排实验，只需作9次，按 $L_{18}(3^7)$ 正交表进行18次实验，显然大大减少了工作量。
- 因而正交实验设计在很多领域的研究中已经得到广泛应用。

正交实验法

利用因果图来设计测试用例时,作为输入条件的原因与输出结果之间的因果关系,有时很难从软件需求规格说明中得到。

- 往往因果关系非常庞大,以至于据此因果图而得到的测试用例数目多的惊人,给软件测试带来沉重的负担,为了有效地,合理地减少测试的工时与费用,可利用正交实验设计方法进行测试用例的设计。

正交实验设计方法:依据Galois理论,从大量的(实验)数据(测试例)中挑选适量的、有代表性的点(例),从而合理地安排实验(测试)的一种科学实验设计方法。

- 类似的方法有:聚类分析方法、因子方法方法等。

正交表的符号

正交表是正交试验设计法的基本工具。它是运用组合数学理论在正交拉丁方的基础上构造的一种规格化的表格。正交表的符号是：

$L_n(j^i)$

其中：

L ——正交表的代号；

n ——正交表的行数（试验次数、试验方案数）；

j ——正交表中的数码（因素的水平数）；

i ——正交表的例数（试验因素的个数）；

$N=j^i$ ——全部试验次数（完全因素水平组合数）。

表 1 正交表 $L_8(2^7)$

列号	试验号	1	2	3	4	5	6	7
1		1	1	1	2	2	1	2
2		2	1	2	2	1	1	1
3		1	2	2	2	2	2	1
4		2	2	1	2	1	2	2
5		1	1	2	1	1	2	2
6		2	1	1	1	2	2	1
7		1	2	1	1	1	1	1
8		2	2	2	1	2	1	2

表 2 正交表 $L_9(3^4)$

列号	试验号	1	2	3	4
1		1	1	3	2
2		2	1	1	1
3		3	1	2	3
4		1	2	2	1
5		2	2	3	3
6		3	2	1	2
7		1	3	1	3
8		2	3	2	2
9		3	3	3	1

正交表的正交性

均衡分散性

- 在同一张正交表中，任意两列（两个因素）的水平搭配（横向形成的数字对）是完全相同的。
- 这样就保证了试验条件均衡地分散在因素水平的完全组合之中，因而具有很强的代表性，容易得到好的试验条件。

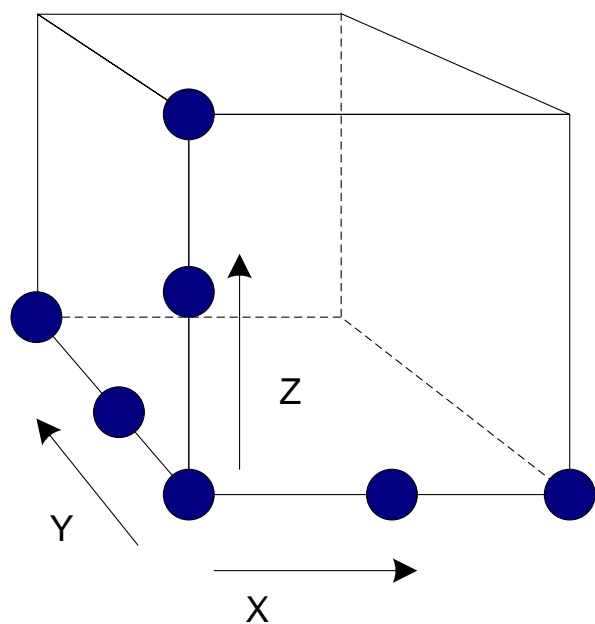
整齐可比性

- 在同一张正交表中，每个因素的每个水平出现的次数是完全相同的。
- 由于在试验中每个因素的每个水平与其它因素的每个水平参与试验的机率是完全相同的，这就保证在各个水平中最大程度的排除了其它因素水平的干扰。
- 因而，能最有效地进行比较和作出展望，容易找到好的试验条件。

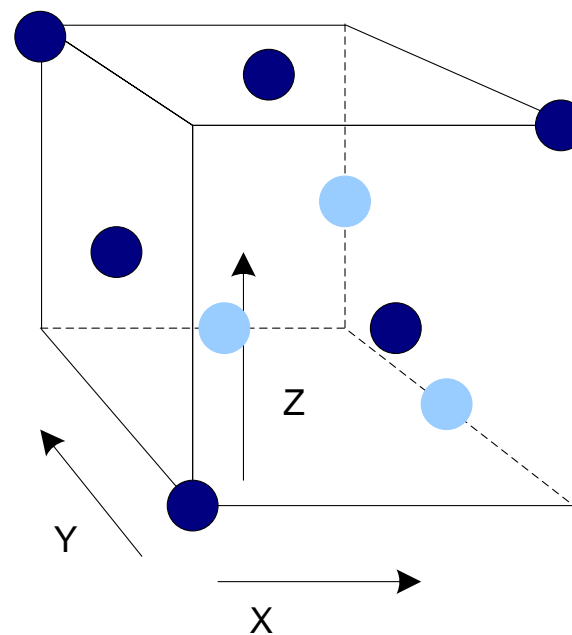
正交数组测试

- 有很多应用，其输入域是有限的。也就是说，输入参数的数量不多，而且每个参数能取的值也是明显确定的。
 - 当这些数量非常小时（例如，三个输入参数，每个参数可取值各为三个离散值），则可以考虑对输入参数值进行排列组合，即用枚举法详尽地测试所有输入域。
 - 然而，随着输入参数的增加和每个输入参数的取值数量的增加，枚举法测试将不再实用。
- 正交数组测试（Orthogonal array testing）可以被应用于输入域相对较小而详尽测试的次数又过于巨大的问题。
 - 正交数组测试方法对于发现和区域错误（region faults）相关的错误特别有用。
 - 区域错误是和软件模块中的逻辑相关的一类错误。
- 为了描述正交数组测试和较为传统的“一次一个输入项(one input item at a time)”方法的区别，考虑一个系统有三个输入项：X、Y和Z。
 - 每一个输入项有三个离散值与之相关，那么就有 $3^3 = 27$ 种可能的测试用例

正交数组测试（续）



(a) 一次一个输入项



(b) L9 正交数组

Phadke[PHA97]建议可以从几何学角度来看这些可能的测试用例

正交数组测试（续）

- 传真机发送功能（send函数）的例子。

- 假设有四个参数：P1、P2、P3和P4，被传入send函数。其中每一个参数都可以取三个离散值。
- 例如，P1取值分别为：

$P_1 = 1$, 立即发送

$P_1 = 2$, 一小时后发送

$P_1 = 3$, 午夜后发送

- 如果采用“同一时间只改变一个输入项的值”的测试策略，那么测试序列（P1, P2, P3, P4）将设为：
 - (1,1,1,1), (2,1,1,1), (3,1,1,1), (1,2,1,1), (1, 3,1,1), (1,1,2,1), (1,1,3,1), (1,1,1,2)和 (1,1,1,3)。

正交数组测试（续）

- Phadke [PHA97]对这些测试用例的评估：
 - 这些测试用例只有在测试参数互不影响的情况下有用。
 - 当只有一个参数的值造成软件故障时，它们可以查出这样的逻辑错误。
 - 这种错误称为单模错误（single mode faults）。
 - 这种方法不能查出当两个或多个参数同时取某些值时导致软件故障的逻辑错误，也就是说，它不能检测参数之间互相影响的情况。因此它的查错能力是有限的。



测试用例	测试参数			
	P ₁	P ₂	P ₃	P ₄
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

L9正交数组

- Phadke[PHA97]对使用L9正交数组后的测试结果进行评估：
 - 检查并且隔离所有的单模错误。单模错误是指任一参数在某个值的情况下都会出现的错误。
 - 检查所有的双模错误。如果两个参数的某些取值同时出现而引起一种固定的错误，则称为双模错误。一个双模错误暗示了两个参数之间的互不相容或者是互相有害的作用。
 - 多模错误。正交数组可能保证检测到单模错误和双模错误。然而，许多多模错误也可由这些测试查找出来。

用正交实验法设计测试用例

一、用正交表设计测试用例的步骤

- (1) 有哪些因素（变量）
- (2) 每个因素有哪几个水平（变量的取值）
- (3) 选择一个合适的正交表
- (4) 把变量的值映射到表中
- (5) 把每一行的各因素水平的组合做为一个测试用例
- (6) 加上你认为可疑且没有在表中出现的组合

用正交实验法设计测试用例

二、如何选择正交表

考虑因素（变量）的个数

考虑因素水平（变量的取值）的个数

考虑正交表的行数

取行数最少的一个

三、设计测试用例时的三种情况

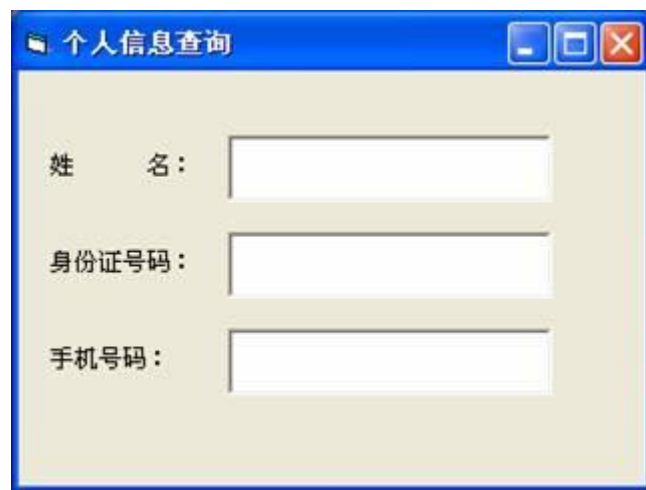
(1) 因素数（变量）、水平数（变量值）相符

(2) 因素数不相同

(3) 水平数不相同

(1) 因素数与水平数刚好符合正交表

举个例子：



- 这是个人信息查询系统中的一个窗口。
- 要测试的控件有3个：
 - 姓名、身份证号码、手机号码 ➡ (也就是要考虑的)因素有三个；
- 每个因素里的状态有两个：填与不填。 ➡ (也就是要考虑的)水平有两个

选择正交表时分析一下：

- 1、表中的因素数 ≥ 3 ；
- 2、表中至少有3个因素数的水平数 ≥ 2 ；
- 3、行数取最少的一个。

从正交表公式中开始查找，结果为： $L_4(2^3)$
变量映射：

行号\列号	1	2	3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0



行号\列号	姓名	身份证号	手机号码
1	填	填	填
2	填	不填	不填
3	不填	填	不填
4	不填	不填	填

测试用例如下：

- 1：填写姓名、填写身份证号、填写手机号
- 2：填写姓名、不填身份证号、不填手机号
- 3：不填姓名、填写身份证号、不填手机号
- 4：不填姓名、不填身份证号、填写手机号

增补测试用例

- 5：不填姓名、不填身份证号、不填手机号

从测试用例可以看出：

- 如果按每个因素两个水平数来考虑的话，需要8个测试用例，而通过正交实验法进行的测试用例只有5个，大大减少了测试用例数。
- 用最小的测试用例集合去获取最大的测试覆盖率。

(2) 因素数不相同

如果因素数不同的话，可以采用包含的方法，在正交表公式中找到包含该情况的公式，如果有N个符合条件的公式，那么选取行数最少的公式。

(3) 水平数不相同

采用包含和组合的方法选取合适的正交表公式。

正交实验法的又一个例子

PowerPoint软件打印功能

假设功能描述如下：

打印范围分：全部、当前幻灯片、给定范围 共三种情况；
打印内容分：幻灯片、讲义、备注页、大纲视图 共四种方式；
打印颜色/灰度分：颜色、灰度、黑白 共三种设置；
打印效果分：幻灯片加框和幻灯片不加框两种方式。

因素状态表:

状态/因素	A打印范围	B打印内容	C打印颜色/灰度	D打印效果
0	全部	幻灯片	颜色	幻灯片加框
1	当前幻灯片	讲义	灰度	幻灯片不加框
2	给定范围	备注页	黑白	
3		大纲视图		

先将中文字转换成字母，便于设计。

状态/因素	A	B	C	D
0	A1	B1	C1	D1
1	A2	B2	C2	D2
2	A3	B3	C3	
3		B4		

我们分析一下：被测项目中一共有四个被测对象，每个被测对象的状态都不一样。

选择正交表：

- 1、表中的因素数 ≥ 4
- 2、表中至少有4个因素的水平数 ≥ 2
- 3、行数取最少的一个

最后选中正交表公式： $L_{16}(4^5)$

行\列号	1	2	3	4	5
1	0	0	0	0	0
2	0	1	1	1	1
3	0	2	2	2	2
4	0	3	3	3	3
5	1	0	1	2	3
6	1	1	0	3	2
7	1	2	3	0	1
8	1	3	2	1	0
9	2	0	2	3	1
10	2	1	3	2	0
11	2	2	0	1	3
12	2	3	1	0	2
13	3	0	3	1	2
14	3	1	2	0	3
15	3	2	1	3	0
16	3	3	0	2	1

用字母替代正交矩阵：

	1	2	3	4	5
1	A1	B1	C1	D1	0
2	A1	B2	C2	D2	1
3	A1	B3	C3	2	2
4	A1	B4	3	3	3
5	A2	B1	C2	2	3
6	A2	B2	C1	3	2
7	A2	B3	3	D1	1
8	A2	B4	C3	D2	0
9	A3	B1	C3	3	1
10	A3	B2	3	2	0
11	A3	B3	C1	D2	3
12	A3	B4	C2	D1	2
13	3	B1	3	D2	2
14	3	B2	C3	D1	3
15	3	B3	C2	3	0
16	3	B4	C1	2	1

	1	2	3	4	5
1	A1	B1	C1	D1	0
2	A1	B2	C2	D2	1
3	A1	B3	C3	D1	2
4	A1	B4	C1	D2	3
5	A2	B1	C2	D1	3
6	A2	B2	C1	D2	2
7	A2	B3	C2	D1	1
8	A2	B4	C3	D2	0
9	A3	B1	C3	D2	1
10	A3	B2	C3	D1	0
11	A3	B3	C1	D2	3
12	A3	B4	C2	D1	2
13	A1	B1	C1	D2	2
14	A2	B2	C3	D1	3
15	A3	B3	C2	D2	0
16	A1	B4	C1	D1	1

第一列水平值为3、第三列水平值为3、第四列水平值3、2都需要由各自的字母替代。

第五列去掉没有意义。通过分析，由于四个因素里有三个的水平值小于3，所以从第13行到16行的测试用例可以忽略。

测试用例编号	PPT—ST—FUNCTION—PRINT—001		
测试项目	测试powerpoint打印功能		
测试标题	测试用例编号	PPT—ST— FUNCTION—PRINT—002	
重要级别	测试项目	测试powerpoint打印功能	
预置条件	测试标题	测试用例编号	PPT—ST—FUNCTION—PRINT—003
输入	重要级别	测试项目	测试powerpoint打印功能
操作步骤	预置条件	测试标题	打印PowerPoint文件A全部的备注页，黑白，加框
	输入	测试用例编号	PPT—ST—FUNCTION—PRINT—004
	操作步骤	重要级别	测试powerpoint打印功能
		预置条件	打印PowerPoint文件A全部的大纲视图，黑白
		输入	中
		重要级别	PowerPoint文件A已被打开，电脑主机已连接有效打印机
		预置条件	文件A: D:\系统测试. ppt
		输入	1、打开打印界面； 2、打印范围选择“全部”； 3、打印内容选择“大纲视图”； 4、颜色/灰度选择“黑白”； 5、点击“确定”。
		操作步骤	
		预期输出	打印出全部大纲视图，黑白
预期输出	预期输出	预期	

SUMMARY: ORTHOGONAL ARRAY TESTING

1、运用正交试验法设计测试用例时遇到的几种情况和解决方法：

- 1) 因素数、水平数与正交表相符
- 2) 因素数不同
- 3) 水平数不同

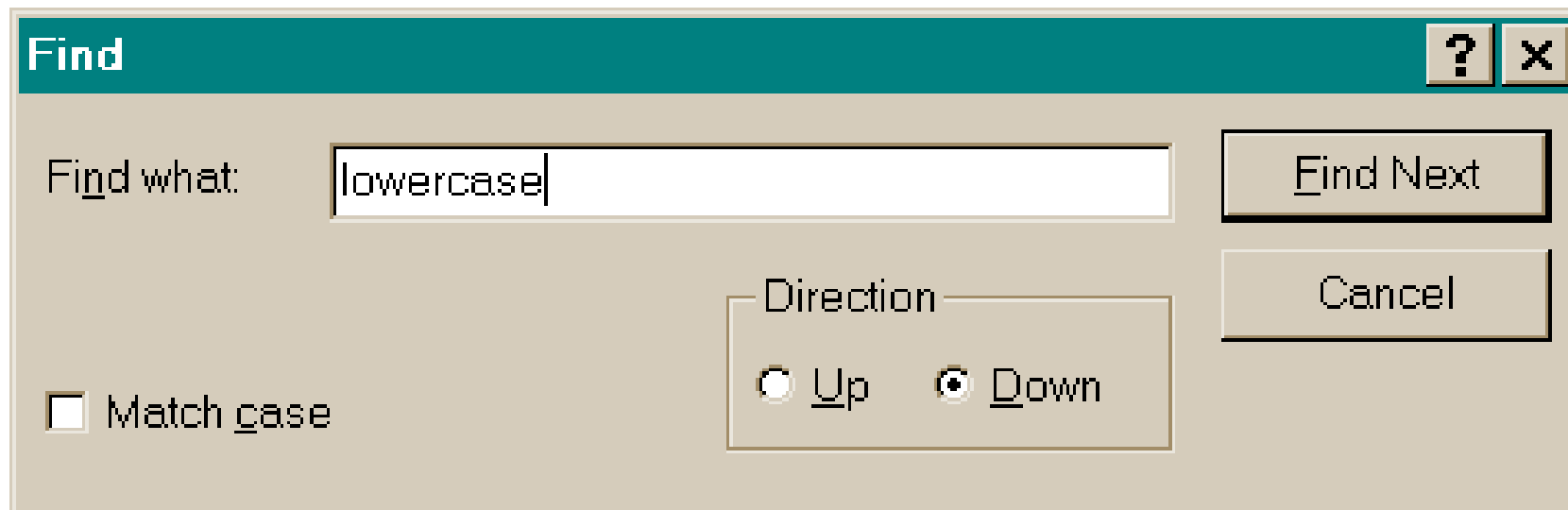
2、用正交表设计测试用例的步骤：

- 1) 确定因素（变量）数
- 2) 确定每个因素的水平数
- 3) 选择一个合适的正交表，不同的正交表生成的用例数会有很大的差别。需要让正交表尽量和因素数和水平数吻合，因素数一般是要一致，水平数要考虑出现最多的水平数
- 4) 将变量的具体取值映射到正交表中
- 5) 将每一行数据的组合抽取出来，形成测试用例
- 6) 再次分析（用错误推测法）可能需要测试的数据组合（查漏补缺），添加到测试用例当中

3、正交表查询地址：

<http://www.york.ac.uk/depts/maths/tables/orthogonal.htm>

COMBINATORIAL EXAMPLE



- Here is a simple Find dialog. It takes three inputs:
 - Find what: a text string
 - Match case: yes or no
 - Direction: up or down
- Simplify this by considering only three values for the text string, “lowercase” and “Mixed Cases” and “CAPITALS”.

COMBINATIONS EXAMPLE

- 1 How many combinations of these three variables are possible?
- 2 List ALL the combinations of these three variables.
- 3 Now create combination tests that cover all possible pairs of values, but don't try to cover all possible triplets. List one such set.
- 4 How many test cases are in this set?

COMBINATIONS EXAMPLE

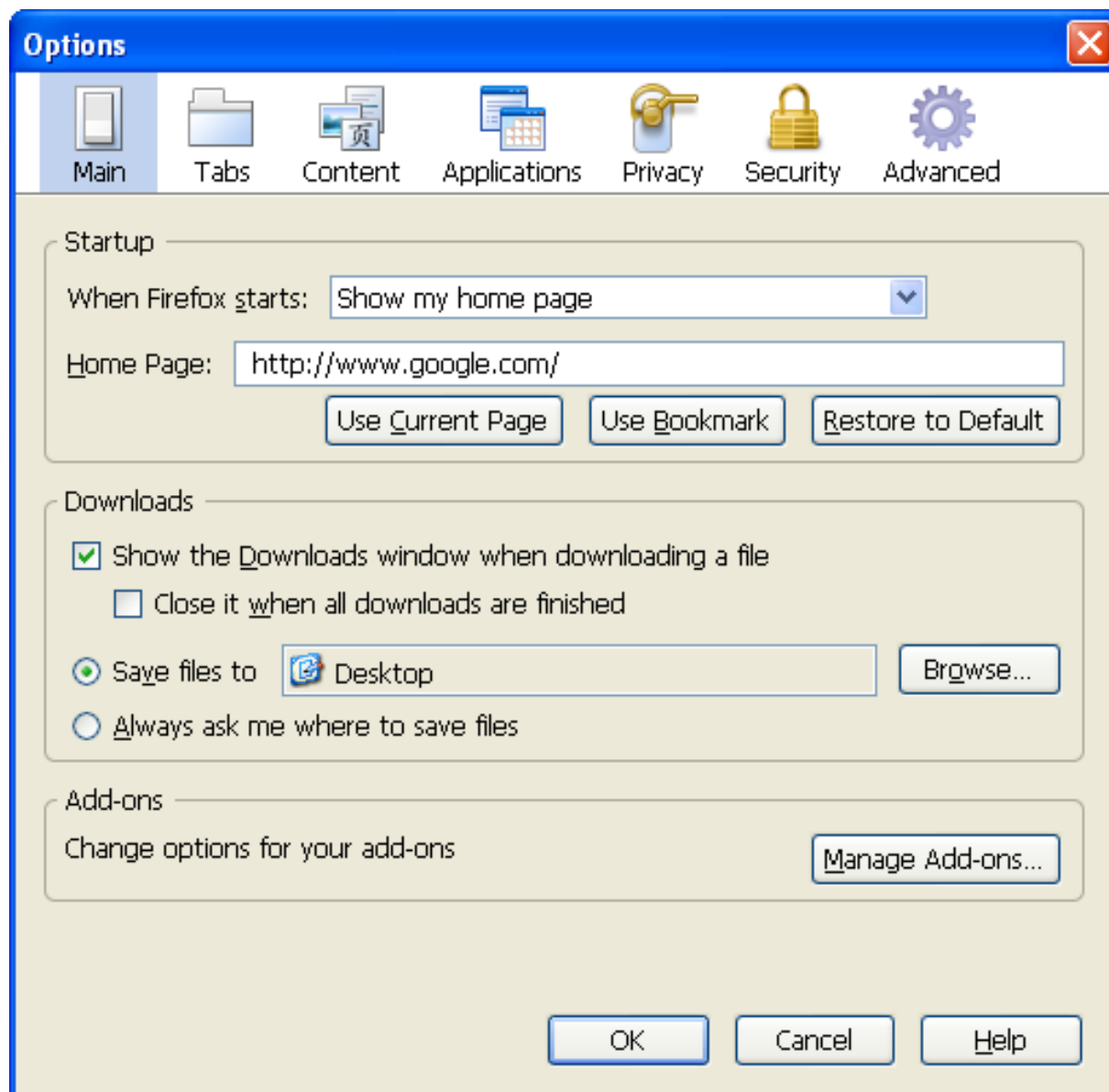
1. How many combinations of these three variables are possible?

- **Find what** has 3 values (lowercase, mixed, caps) (L M C)
- **Match case** has 2 values (Yes / No) (Y N)
- **Direction** has 2 values (Up / Down) (U D)

So there will be $3 \times 2 \times 2 = 12$ tests

2. List ALL the combinations of these three variables.

L Y U	M Y U	C Y U
L Y D	M Y D	C Y D
L N U	M N U	C N U
L N D	M N D	C N D



提 纲

- 黑盒测试
- 等价划分
- 边界值分析
- 因果分析法
- 正交数组测试
- ➡ ■ 测试插桩
 - 测试预言
 - 随机数据生成器

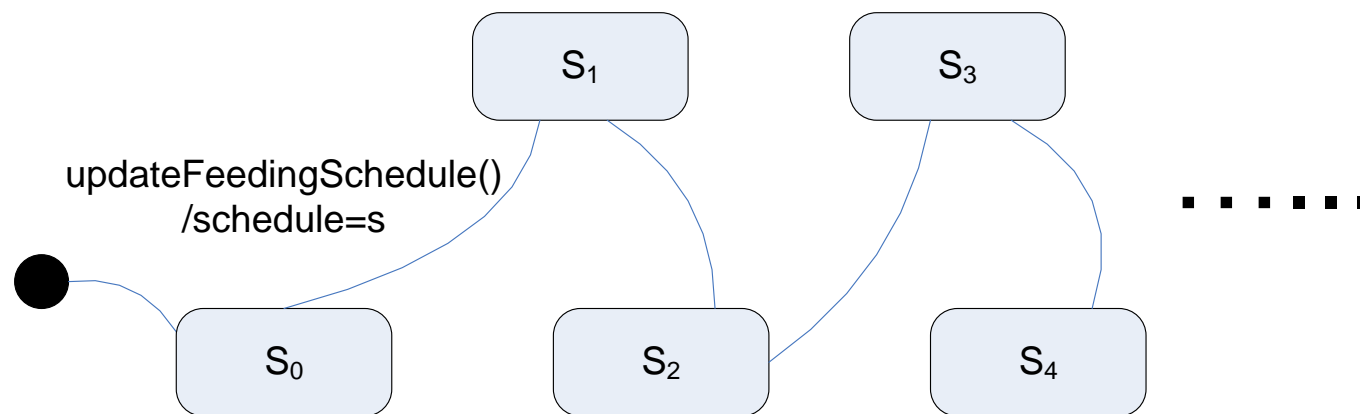
测试插桩

- 测试插桩 (Instrumentation) 是指在程序的特定部位附加一些操作或功能，可用来检验程序运行结果以及执行特性，以便支持测试系统。
- 目前有两大类测试插桩技术,分别支持黑盒测试和白盒测试。
- 介绍两种支持黑盒测试的插桩技术，测试预言与随机数生成器：
 - 测试预言(oracle)：一个程序，是用来确定对于给定的系统输入数据或输入序列，被测系统是否有一个特定输出。
 - 随机数据生成器：用来产生测试数据的一个程序。

测试预言

- 测试预言是实现黑盒测试规格说明中的一个模块里的谓词 (predicates) 或条件。
- 我们以自动化喂鱼 (fish feeding) 系统中“喂鱼视图”模块为例，介绍测试预言技术的应用。
 - 喂鱼视图模块规定两次喂鱼时间间隔，容许修改喂鱼时间表，但必须满足规定的喂鱼时间间隔。

Using UML Statechart



State(schedule:FeedingSchedule);
Initially at S₀, schedule = null;
Invariant: at all states except S₀, f.end - f.start ≤ 30 minutes;
Event message: updateFeedingSchedule();
Action on Transition: schedule = s;

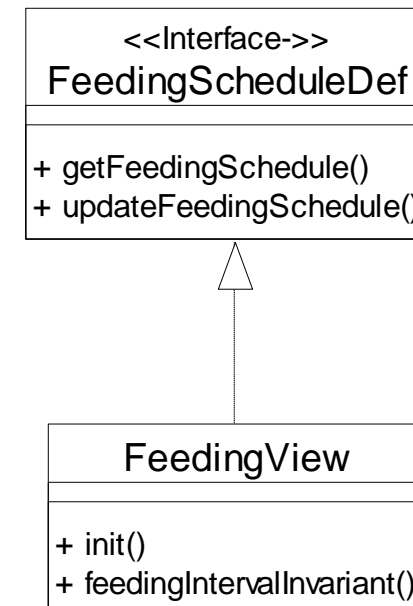
```

package ffcs;
//import second from time unit
//import feeding, feedings from feeding_schedule
public class FeedingView implements FeedingScheduleDef {
(1)     FeedingSchedule schedule=null; // initially empty
        // omit implementation for feeding interval invariant
(2)     //INVARIANT: ALL(f:feedings(schedule):: f.end-f.start <=30 minute)

(3)     //implement the two methods in interface of FeedingScheduleDef
        public FeedingSchedule getFeedingSchedule(){
            return schedule;
        }
        public void updateFeedingSchedule (FeedingSchedule s){
            schedule = s;
(4)     }
        // ...
}

```

(a)



(b)

```
public class TestFeedingView {  
(1)     FeedingView fv = new FeedingView();  
        //check invariant  
(2)     private boolean checkInvariant() {  
(3)         FeedingSchedule fs=fv.getFeedingSchedule();  
(4)         return (fs.getEnd()-fs.getStart() > 30*60*1000)? false: true;  
(5)     }  
        //test method of updateFeedingSchedule()  
(6)     public boolean testUpdateFeedingSchedule(){  
(7)         FeedingSchedule fs=new FeedingSchedule();  
(8)         Calendar now = Calendar.getInstance();  
(9)         fs.setStart(now);  
(10)        now.add(now.MINUTE, 31);  
(11)        fs.setEnd(now);  
(12)        fv.updateFeedingSchedule(fs);  
(13)        return checkInvariant();  
(14)    }  
        //execute test  
(15)    public static void main (String[] args) {  
(16)        TestFeedingView tfv=new TestFeedingView();  
(17)        if (! tfv.testUpdateFeedingSchedule())  
(18)            System.out.println ("testUpdateFeedingSchedule() failed.");  
(19)        else  
(20)            System.out.println ("testUpdateFeedingSchedule() succeeded.");  
(21)    }  
}
```


随机数据生成器

- 随机测试技术（random testing technique）是一种黑盒测试方法，通过在所有可能的输入值中选取一个任意的子集来对软件进行测试[1]。
 - 随机测试有助于避免只测试你所知道的将奏效的问题。1990年Dick Hamlet比较利用等价划分的子域边界测试结果与忽视等价划分的随机测试结果，指出等价划分的作用微乎其微。
 - 但完全的随机测试方法难以进行有针对性有目标的测试，从而导致低效测试活动。
 - 一般认为，在生成测试用例时，结合使用等价划分、边界值和随机测试三种技术是一种有效方法。即先将软件输入域进行等价划分，在各个子域边界上及附近选取中边界值，然后再在各个子域里用随机测试方法选择软件输入样本点。
- [1] 参考URL: <http://computing-dictionary.thefreedictionary.com/random+testing> 的定义。

随机数据生成器（续）

- 例如一个软件有效输入域是1~100之间的整数，包括1和100。用等价划分法，测试输入域是：

无效域1：输入数据 < 1

有效域： $1 \leq \text{输入数据} \leq 100$

无效域2：输入数据 > 100

- 用边界值分析法，可生成三个子域的边界值，例如，分别为 $\{x \mid x \leq 0\}$ 、 $\{1, 2-99, 100\}$ 和 $\{y \mid y \geq 101\}$ 。然后在三个子域里用随机测试方法选择软件输入样本点。

随机数据生成器（续）

- 随机数据生成器是运用了Java API 中的public static double random()。
 - 调用后返回一个带正号的double值，大于或等于0.0，小于1.0。返回值伪随机地均匀地分布在这个范围内。
 - 即， $0.0 \leq \text{Math.random()} < 1.0$
- 若要产生一个1~100之间的随机数，则执行 `(int)(Math.random() * 100)+1`;
- 设有一个字符数组char[n]，若是要产生一个长度小于或等于n的字符串，可以用Math.random()来产生一个随机数k。代码如下：

```
.....  
String s="";  
int k = (int)(Math.random()*n)+1;  
for(int i=0; i<k; i++) s += char[i];  
return s;  
.....
```

测试ORACLE和随机数据生成器的问题

- 但在使用以上两种插装时，会出现以下问题，应加以注意：
 - 没有量词的或者有限范围内没有量化的变量的谓词容易实现；
 - 一般的谓词不容易实现，但是可以手动解决；
 - 随机的数据生成器对于抽象的数据类型可以递归的生成。

测试插桩

- 支持白盒测试的测试插桩技术可提供以下功能：
 - 生成给定状态或者内部数据表示法以强迫程序进入一个特定的状态，以便检验状态的可达性。
 - 显示或读取内部数据或者私有数据：这类数据一般不能从程序外部获得。
 - 监测具体的数据不变特性：程序中的某些数据在执行过程中保持不变，在程序不同位置插入语句可以观测此数据不变特性。
 - 监测前提条件：在执行某操作或过程之前，插入语句可以观测操作或过程前提条件。
 - 监测后置条件：在执行某操作或过程之后，插入语句可以观测操作或过程后置条件。
 - 人为触发时间事件：当某事件发生所需时间太短或太长，在程序不同位置插入适当功能可以控制时间事件的发生。
 - 监测事件时间来测试时间约束：在程序中事件发生前后位置插入语句可以记录事件发生时间以检验时间约束。

提 纲

- 黑盒测试
- 等价划分
- 边界值分析
- 因果分析法
- 正交数组测试
- 测试插桩



- 总结

总结

黑盒测试概念

等价划分,边界值分析

因果分析法,正交数组测试

测试插桩