



Session 11

Black Box Testing (5)

Combinatorial Test

# Black Box Testing Techniques

- Boundary Value Analysis
- Equivalence Partitioning
- Decision Table
- Cause-Effect Graph
- Combinatorial Test

# Motivation

- 因素间的复杂关系
  - 输入输出测试
  - 配置测试
  - 兼容性测试
  - ...
- 在实际的软件项目中，输入条件多，而每个条件又有多个取值。 ---组合爆炸

字体

字体(N) 字符间距(R)

西文字体(E): 微软雅黑 字体样式(Y): 常规 大小(S): 40

中文字体(I): +中文正文

所有文字

字体颜色(C) 下划线线型(U) (无) 下划线颜色(I)

效果

☐ 删除线(K) ☐ 小型大写字母(M)

☐ 双删除线(L) ☐ 全部大写(A)

☐ 上标(P) 偏移量(E): 0% ☐ 等高字符(Q)

☐ 下标(B)

确定 取消

40 字体  
4 样式  
30 大小  
10 颜色  
16 下划线  
2^7 效果

9.8 X 10^7

```
int Foo(int a, int b, int c, int d)
{
    int result = 0;

    //calculating with input a, b, c and d
    ...

    return result;
}
```

No.	a	b	c	d
1	-1	0	1	-1
2	0	1	-1	0
3	1	-1	0	1

No.	a	b	c	d
1	-1	-1	-1	-1
2	-1	-1	-1	0
3	-1	-1	-1	1
4	-1	-1	0	-1
5	-1	-1	0	0
6	-1	-1	0	1
7	-1	-1	1	-1
8	-1	-1	1	0
9	-1	-1	1	1
10	-1	0	-1	-1
11	-1	0	-1	0
12	-1	0	-1	1
13	-1	0	0	-1
14	-1	0	0	0
15	-1	0	0	1
16	-1	0	1	-1
17	-1	0	1	0
18	-1	0	1	1
19	-1	1	-1	-1
20	-1	1	-1	0
21	-1	1	-1	1
22	-1	1	0	-1
23	-1	1	0	0
24	-1	1	0	1
25	-1	1	1	-1
26	-1	1	1	0
27	-1	1	1	1

No.	a	b	c	d
28	0	-1	-1	-1
29	0	-1	-1	0
30	0	-1	-1	1
31	0	-1	0	-1
32	0	-1	0	0
33	0	-1	0	1
34	0	-1	1	-1
35	0	-1	1	0
36	0	-1	1	1
37	0	0	-1	-1
38	0	0	-1	0
39	0	0	-1	1
40	0	0	0	-1
41	0	0	0	0
42	0	0	0	1
43	0	0	1	-1
44	0	0	1	0
45	0	0	1	1
46	0	1	-1	-1
47	0	1	-1	0
48	0	1	-1	1
49	0	1	0	-1
50	0	1	0	0
51	0	1	0	1
52	0	1	1	-1
53	0	1	1	0
54	0	1	1	1

No.	a	b	c	d
55	1	-1	-1	-1
56	1	-1	-1	0
57	1	-1	-1	1
58	1	-1	0	-1
59	1	-1	0	0
60	1	-1	0	1
61	1	-1	1	-1
62	1	-1	1	0
63	1	-1	1	1
64	1	0	-1	-1
65	1	0	-1	0
66	1	0	-1	1
67	1	0	0	-1
68	1	0	0	0
69	1	0	0	1
70	1	0	1	-1
71	1	0	1	0
72	1	0	1	1
73	1	1	-1	-1
74	1	1	-1	0
75	1	1	-1	1
76	1	1	0	-1
77	1	1	0	0
78	1	1	0	1
79	1	1	1	-1
80	1	1	1	0
81	1	1	1	1

测试用例数量:  $3 \times 3 \times 3 \times 3 = 81$

## 安卓手机硬件“碎片化”问题

系统版本	CPU	GPU	屏幕模式
Android 1.x	Qualcomm	SGX	4:3
Android 2.x	TexasInstruments	Adreno	16:10
Android 4.x	Intel	GeForce	平板电脑

## Web网站及应用的兼容性问题

操作系统	浏览器	网络接入	屏幕模式
Windows	IE	Wi-Fi	标清(4:3)
Mac OS	Chrome	4G	标清(16:9)
Linux	Safari	LAN	Retina

打印测试，也需要考虑4个因素，每个因素也有多个选项

- 打印范围分：全部、当前幻灯片、给定范围
- 打印内容分：幻灯片、讲义、备注页、大纲视图
- 打印颜色/灰度分：彩色、灰度、黑白
- 打印效果分：幻灯片加框和幻灯片不加框

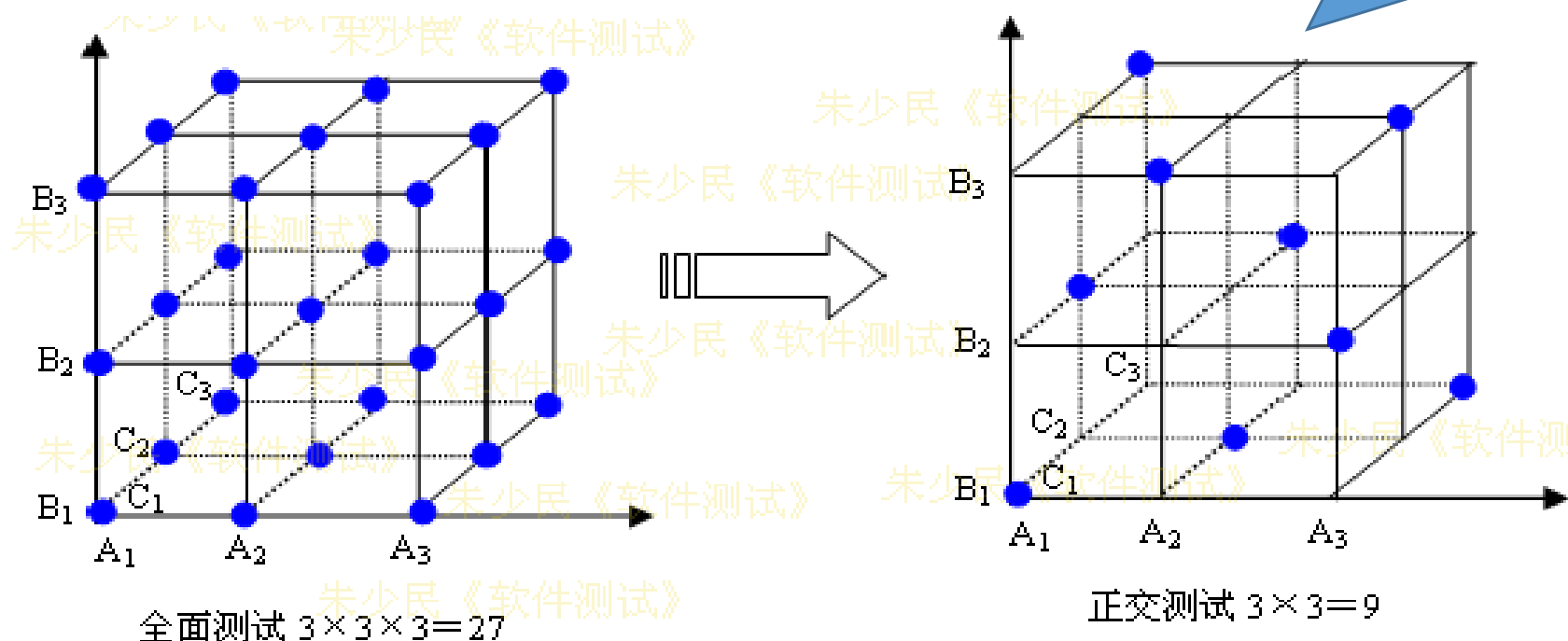


# 如何解决组合爆炸问题？

- 组合测试(Combinatorial Test): 抽样, 优化组合
  - 将被测试应用抽象为一个受到多个因素影响的系统, 其中每个因素的取值是离散且有限的
  - 正交实验设计
  - 多因素 (N- way,  $N > 2$ ) 组合测试可以覆盖任意N个因素的所有取值组合, 在理论上可以发现由N个因素共同作用引发的缺陷

# 正交实验设计

- 从大量的数据（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



# 正交实验设计

- 采用正交设计法设计测试用例主要包括以下步骤：
  - 确定影响因素
  - 确定每个因素的水平
  - 选择正交表
    - 根据确定的因素和水平，选择合适的正交表
    - 如果没有合适的正交表可用或需要的测试用例个数太多，则要对因素和水平进行调整
  - 设计测试用例

# 示例1

```
int Foo(int a, int b, int c, int d)
{
    int result = 0;

    //calculating with input a, b, c and d
    ...

    return result;
}
```

No.	a	b	c	d
1	-1	0	1	-1
2	0	1	-1	0
3	1	-1	0	1

# 正交表

$t=2$  (两两组合),  $\lambda=1$  (组合出现的次数)

No.	a	b	c	d
1	-1	-1	-1	-1
2	-1	0	0	0
3	-1	1	1	1
4	0	-1	0	1
5	0	0	1	-1
6	0	1	-1	0
7	1	-1	1	0
8	1	0	-1	1
9	1	1	0	-1

$L_9(3^4)$

用例数, 即行数

因子数, 即列数

因子的水平数

# 示例2

- 为提高某化工产品的转化率，选择了三个有效因素进行条件试验，A=反应温度，B=反应时间，C=用碱量，且
  - A：80~90度
  - B：90~150分钟
  - C：5%~7%
- 目的：弄清因子A，B，C哪个对转化率是主要的，次要的，从而确定最适合的条件。
- 在试验范围内选取三个水平：
  - A：A1=80度，A2=85度，A3=90度
  - B：B1=90分钟，B2=120分钟，B3=150分钟
  - C：C1=5%，C2=6%，C3=7%

# 简单对比法

首先固定B、C于B1、C1，使A变化：

B1C1——A1

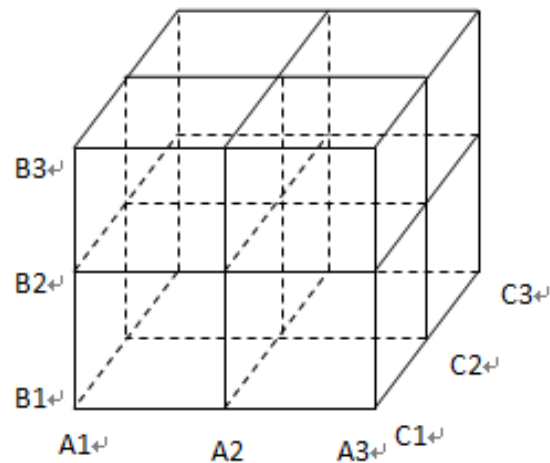
——A2

——A3（好结果）

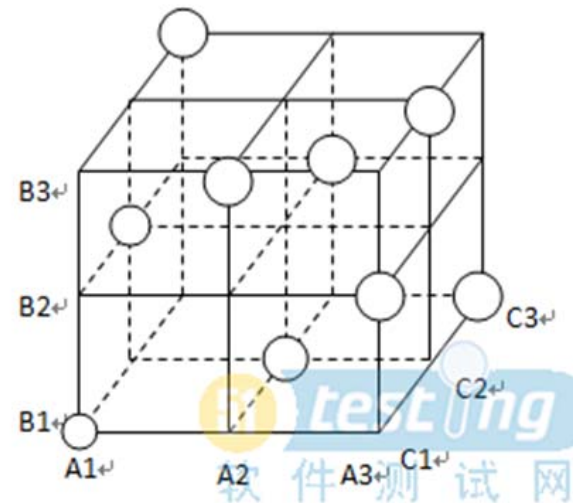
若结果A3最好，则固定A于A3，C还是C1，使B变化  
依次下去对比

优点：试验次数少，但选择代表性差

## 全面试验法



优点：对各因子与指标间的关系剖析的比较清楚，但试验次数太多



# 简单对比法

首先固定B、C于B1、C1，使A变化：

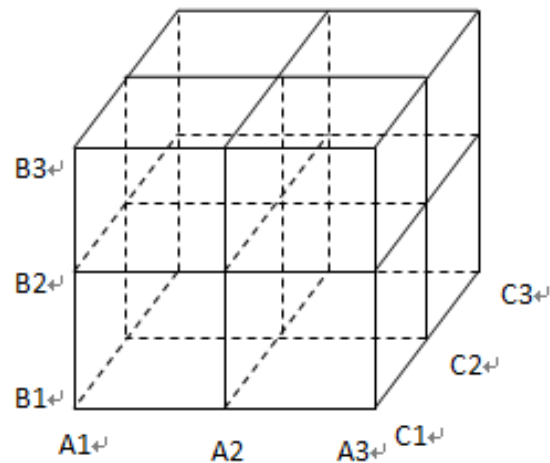
B1C1—A1/A2/A3 （假设A3最好） 3次

A3C1—B2/B3 （假设B2最好） 2次

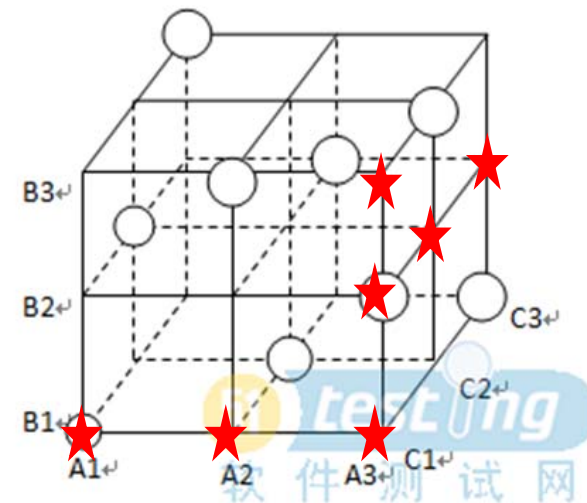
A3B2—C2/C3 （假设C2最好） 2次

优点：试验次数7次，但选择代表性差

## 全面试验法



优点：对各因子与指标间的关系剖析的比较清楚，但试验次数太多





- 用正交试验设计法所需的行数是多少？
  - 试验次数（行数）= 求和【各个列数\*（水平-1）】+1
  - 示例1， 示例2    --相同水平正交表  $L_9(3^4)$
  - 示例3： --混合正交表
    - 五个3水平因子及一个2水平因子， 则试验次数至少为
$$5 * (3-1) + 1 * (2-1) + 1 = 12$$
    - 可以选择L18 ( $2*3^7$ ) 的正交表

# 正交试验设计工具

- 常用正交表查询
  - [http://support.sas.com/techsup/technote/ts723\\_Designs.txt](http://support.sas.com/techsup/technote/ts723_Designs.txt)
- SPSS
- SAS
- 正交设计助手

# 正交表

- 何为正交表？

- 任意 $t$ 个输入变量间
- 每个 $t$ 元输入取值组合排列方式齐全而且均衡

出现次数 $\lambda$ 相等

优势：

- 对组合的覆盖
- 空间分布均匀

劣势：

- 正交表构造困难
- 难以判定存在性

# 正交表

$t=2, \lambda=1$

No.	a	b	c	d
1	-1	-1	-1	-1
2	-1	0	0	0
3	-1	1	1	1
4	0	-1	0	1
5	0	0	1	-1
6	0	1	-1	0
7	1	-1	1	0
8	1	0	-1	1
9	1	1	0	-1

$L_9(3^4)$

用例数，即行数

因子数，即列数

因子的水平数

# 正交表

$t=2, \lambda=1, 2$

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

$L_8 (4^1 \times 2^4)$

用例数，即行数

因子数，即列数

因子的水平数

# 组合覆盖表

- 何为组合覆盖表?
  - 任意 $t$ 个输入变量间
  - 每个 $t$ 元输入取值组合出现至少一次

优势:

- 对组合的覆盖
- 空间分布均匀

劣势:

- 正交覆盖表必然存在
- 难以覆盖表便于构造

# 组合覆盖表

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3

# 组合覆盖表

<i>InputA</i>	<i>InputB</i>	<i>InputC</i>	<i>InputD</i>
A1	B1	C1	D1
A1	B1	C2	D2
A1	B2	C1	D3
A2	B1	C2	D3
A2	B2	C1	D2
A2	B2	C2	D1

强度 $t=2$



# 组合覆盖表

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A1	B2	C2	D1
A2	B1	C2	D1
A2	B2	C1	D1
A1	B1	C2	D2
A1	B2	C1	D2
A2	B1	C1	D2
A2	B2	C2	D2
A1	B1	C1	D3
A1	B2	C2	D3
A2	B1	C2	D3
A2	B2	C1	D3

强度t=3

# 组合测试

- 两因素组合测试（也称配对测试、全对偶测试）
  - 测试集可以覆盖任意两个变量的所有取值组合。在理论上，可以暴露所有由两个变量共同作用而引发的缺陷。
- 多因素（t-way,  $t > 2$ ）
  - 测试集可以覆盖任意t个变量的所有取值组合。在理论上，可以发现所有t个因素共同作用引发的缺陷。
- 基于选择的覆盖
  - 选出一个基础的组合，且基础组合中包含每个参数的基础值，建议选择最常用的有效值作为基础值。
  - 基于基础组合，每次只改变一个参数值，来生成新的组合用例。

# Pair-wise

- 成对组合（Pair-Wise），又称两两组合、对对组合，它是将所有因素的水平按照两两组合的原则而产生的。
- Mandl于1985年在测试Ada编译程序时提出的。
- Pairwise基于如下2个假设：
  - 每一个维度都是正交的，即每一个维度互相都没有交集。
  - 根据数学统计分析，73%的缺陷（单因子是35%，双因子是38%）是由单因子或2个因子相互作用产生的。19%的缺陷是由3个因子相互作用产生的。

# 示例

- 假设有3个维度，每个维度有几个因子。如下：
  - 浏览器：M, O, P
  - 操作平台：W(windows), L(linux), i(ios)
  - 语言：C(Chinese), E(English)
- 使用pairwise算法，有多少个测试case？具体是什么case？

# 示例

- 全覆盖需要  $3*3*2=18$ 个测试case

M W C	M W E	M L C	M L E	M I C	M I E
O W C	O W E	O L C	O L E	O I C	O I E
P W C	P W E	P L C	P L E	P I C	P I E

P I E, 两两组合是 P I, P E, I E。P I在17号, P E在16号, I E在12号出现过。以此类推, 最终剩下的如下:

1, M W C	4, M L E	6, M I E
7, O W E	9, O L C	11, O I C
14, P W E	15, P L C	17, P I C

# 示例

M W C	M W E	M L C	M L E	M I C	M I E
O W C	O W E	O L C	O L E	O I C	O I E
P W C	P W E	P L C	P L E	P I C	P I E

从1号开始优化，最终剩下？

2, MWE

4, MLE

5, MIC

8, OWE

10, OLE

11, OIC

13, PWC

15, PLC

18, PIE

# 组合测试中的一些问题

- 默认取值问题
- 约束问题

# 组合测试中的默认取值问题



## 案例1



案例2



首页

国航假期

优

机票预订

☐ 往返 ☒ 单程 ☐ 多段

 航线图

国航假期

出发城市

出发日期

办理乘机

请选择 ( 热点城市机场 )

航班动态

服务预定

信息查询

北京	上海浦东	上海虹桥	广州	深圳
成都	重庆	西安	大连	杭州
洛杉矶	纽约肯尼迪	纽约纽瓦克	东京成田	新加坡
伦敦	首尔仁川	温哥华	悉尼	法兰克福

优惠促销代码

查询预订

## 案例2



中國東方航空

CHINA EASTERN



搜索

首页

预订行程

自助服务

信息服务

积分商城

旅游度假

关于浦东-马累航线免费退改的紧急通知

预订行程

机票

酒店

自助服务

信息服务

移动端

单程

往返

国际缺口程

多目的地

出发城市

到达城市

热门城市 (可直接输入城市名称或城市拼音)

国内

国际/地区

热门城市

ABCD

EFGHJ

KLMNOP

QRSTW

XYZ

上海

北京

昆明

西安

广州

成都

南京

深圳

青岛

杭州

长沙

太原

## 案例2



中国南方航空  
CHINA SOUTHERN AIRLINES



Hi, 8

首页 ▾ 预订管理 ▾ 服务大厅 ▾ 南航假期 ▾ 明珠会员 ▾ 1

 预订

 服务

 团购

 信息

 移动端

 团队

 机票

 酒店

 用车

 签证

单程 往返 联程>>

出发城市

去往城市 

国内

国际/地区

出发日期 

热门城市

ABCDE

FGHIJ

KLMNP

QRSTU

VWXYZ

成人(≥ 12岁) 

1

▼

历史记录查询 ▾

立即查询



里上网来

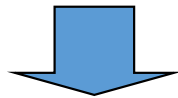
第2步

# Base Choice

基于选择的覆盖

- 选出一个基础的组合，且基础组合中包含每个参数的基础值，建议选择最常用的有效值作为基础值。
- 基于基础组合，**每次只改变一个参数值**，来生成新的组合用例。

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3



每个变量均有一个基本/默认选项

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2	C2	D3



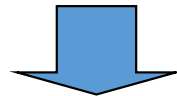
测试用例集

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2	C2	D3
A2	B2	C2	D3
A1	B1	C2	D3
A1	B2	C1	D3
A1	B2	C2	D1
A1	B2	C2	D2

测试用例数量？

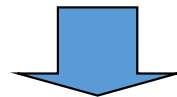
# Multiple Base Choice

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3



某些变量有多个基本/默认选项

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2	C2	D1
			D3



<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2	C2	D1 D3

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2	C2	D1
A2	B2	C2	D1
A1	B1	C2	D1
A1	B2	C1	D1
A1	B2	C2	D2
A1	B2	C2	D3

A1	B2	C2	D3
A2	B2	C2	D3
A1	B1	C2	D3
A1	B2	C1	D3
A1	B2	C2	D1
A1	B2	C2	D2

# 组合测试中的约束问题



# 组合测试中的两种约束问题

强制约束：取值组合为非法

非强制约束：取值组合无需覆盖

```
Bool Fun(bool a ,bool b, bool c, bool d,  
          bool e, bool f, bool g, bool h)  
{  
    if (a && !b)  
        return false;  
    return (a&&c || b&&d) && e && (f && g || !f && h);  
}
```

强制约束a=true b=false

```
Bool Fun(bool a ,bool b, bool c, bool d,  
          bool e, bool f, bool g, bool h)  
{  
    a = true;  
    b = false;  
    return (a&&c || b&&d) && e && (f && g || !f && h);  
}
```

非强约束 a=true b=false

# 如何处理非强约束问题？

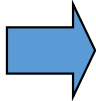
1 直接忽略

2 算法预处理

如何处理强制约束问题？

# 合并输入变量

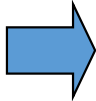
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3
Constraint			
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2		



<i>(Input A Input B)</i>	<i>Input C</i>	<i>Input D</i>
(A1 B1)	C1	D1
(A2 B1)		D2
(A2 B2)	C2	D3

# 重构输入区域

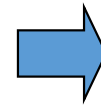
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3
Constraint			
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2		



<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
		C2	D2
			D3
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A2	B1	C1	D1
	B2	C2	D2
			D3

# 修改测试用例

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3
Constraint			
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B2		



(A1, B2, C1, D1)



(-, B2, C1, D1)

(A1, -, C1, D1)

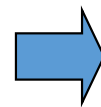


(A2, B2, C1, D1) (A1, B1, C1, D1)



# 练习

<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
A1	B1	C1	D1
A2	B2	C2	D2
			D3
Constraint			
<i>Input A</i>	<i>Input B</i>	<i>Input C</i>	<i>Input D</i>
		C2	D2



合并输入变量

重构输入区域

修改测试用例

# Pairwise小结

- Pairwise VS. 单因素
  - Pairwise能够覆盖到两个维度的组合，能适当减少遗漏的测试。
- Pairwise VS. 全覆盖设计法 VS. 正交表法
  - 全覆盖设计法测试case太多，投入的成本太大。
  - 正交表法是对组合的等概率覆盖，构造困难。
  - Pairwise较之全覆盖设计法，减少了测试用例。较之正交表法，构造相对简单，提高了测试效率。

# Pairwise程序

- 微软的PICK
- ReduceArray
- SmartDesign
- <http://www.pairwise.org./tools.asp>

# Black Box Testing Techniques

- Boundary Value Analysis
- Equivalence Partitioning
- Decision Table
- Cause-Effect Graph
- Combinatorial Test

基于输入/输出域的方法

基于组合及其优化的方法