CHAPTER 7

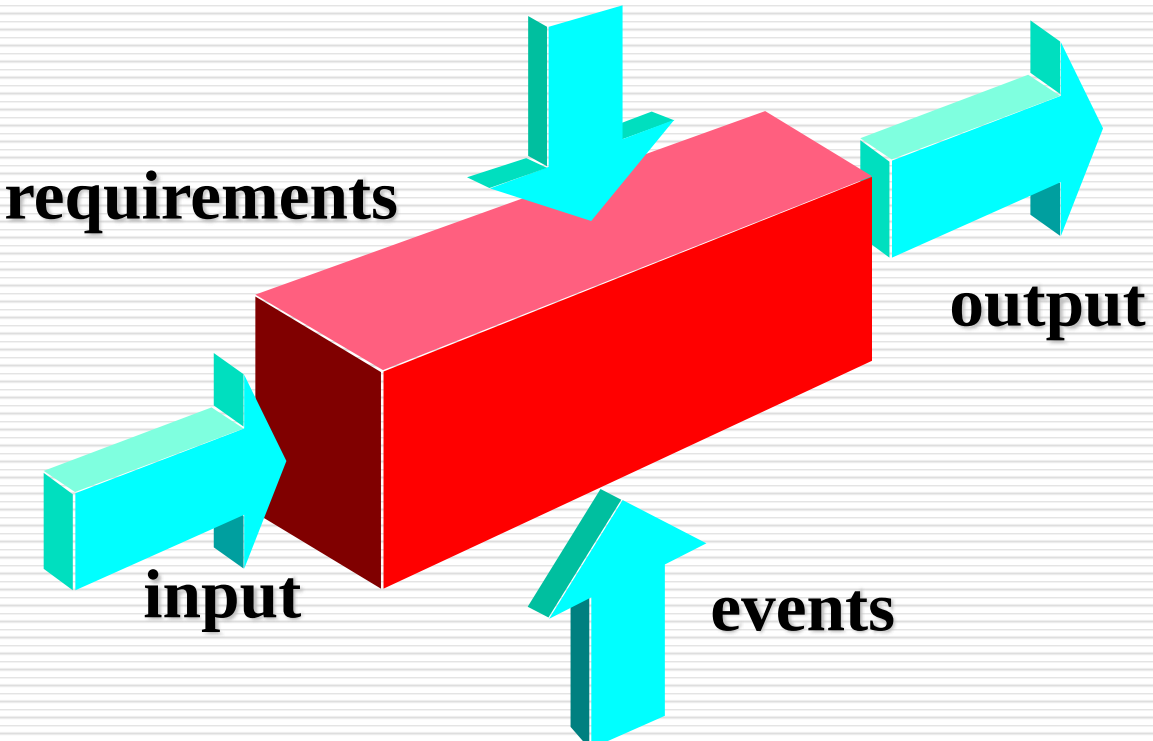
Software Testing

Outline

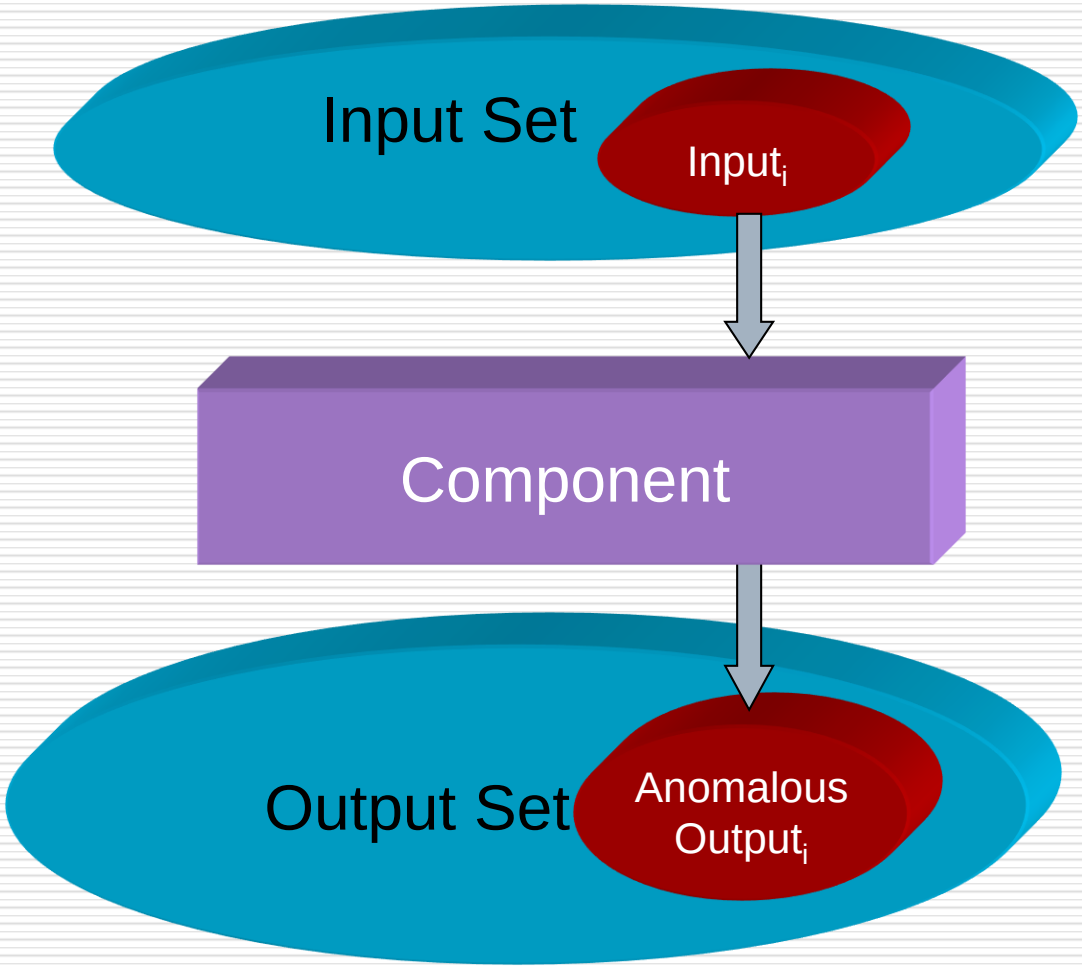
- **Black-box testing in detail**
  - ✓ **equivalence class partitioning**
  - ✓ **boundary-value testing**
  - ✓ **error guessing, random testing**
  - ✓ **cause and effect diagram**

Black Box Testing

- Complementary to white box testing.
- Derive external conditions that fully exercise all functional requirements without internal details.



黑盒测试



黑盒测试

also named:

- ✓ **input/output test**
- ✓ **external test**
- ✓ **functional test**
- ✓ **data driven test**
- ✓ **specification based test**

又称：

输入/输出测试

外部测试

功能测试

数据驱动测试

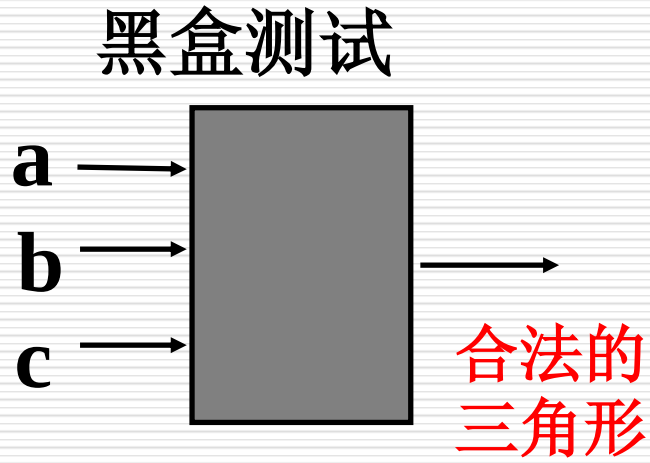
基于规格说明书的测试

黑盒测试的特点

- Attempts to find errors in the following categories
  - ✓ Incorrect or missing functions
  - ✓ Interface errors
  - ✓ Errors in data structures or external database access
  - ✓ Behavior or performance errors
  - ✓ Initialization or termination errors
- Black box testing is performed during later stages of software testing
- easy to put in practice
- easy to understand

Exhaustive black-box testing (infeasible)

例: 输入三角形三条边长, 判断是否可构成合法的三角形?



设a, b,c 为16位2进制数

可用的测试用例数为:

$$2^{16} \times 2^{16} \times 2^{16} \approx 3 \times 10^{14}$$

执行时间: 设测试一次需1ms，共需一万年.

Test case design in black-box testing

- **Methods:**
  - ✓ **equivalence class partitioning**
  - ✓ **boundary-value testing**
  - ✓ **error guessing, random testing**
  - ✓ **cause and effect diagram**

Equivalence class partitioning

- 等价类划分法

Why equivalence partitioning ?

Answer: An example, at first.

a simple example

program's specification:

This program is designed to add two numbers, which you will enter

- *Each number should be one or two digits*
- *Press Enter after each number and the program will print the sum.*
- *To start the program, type* **ADDER**

Sum = $X_1 X_2 + Y_1 Y_2$

Before you start testing, do you have any questions?

Testing demo

? ADDER ↵

? 3 ↵

? 7 ↵

10 ↵

? _ 不断记录测试数据……

For the first test, try a pair of easy values, such as 3 plus 7.

Here is the screen display that results from that test.

Are there any bug reports that you would find from this?

为了保证无误，所有的测试如下：

0+0	1+0	2+0	3+0		10+0	99+0
0+1	1+1	2+1	3+1		10+1	99+1
0+2	1+2	2+2	3+2		10+2	99+2
0+3	1+3	2+3	3+3		10+3	99+3
0+4	1+4	2+4	3+4		10+4	99+4
0+5	1+5	2+5	3+5		10+5	99+5
0+6	1+6	2+6	3+6		10+6	99+6
0+7	1+7	2+7	3+7	……	10+7	99+7
0+8	1+8	2+8	3+8		10+8	99+8
0+9	1+9	2+9	3+9		10+9	99+9
0+10	1+10	2+10	3+10		10+10	99+10
0+11	1+11	2+11	3+11		10+11	99+11
0+12	1+12	2+12	3+12		10+12	99+12
…	…	…	…		…	…
0+99,	1+99,	2+99,	3+99,		10+99,	99+99,

✓ ***All possible input cases:***

100×100

✓***We cannot afford to run every possible test.***

✓***We need a method for choosing a few tests that will represent the rest.***

✓**Equivalence analysis is the most widely used approach.**

什么是等价类

- 等价类：输入域(问题域)的一个子集，在该子集中，各个输入数据对于揭示程序中的错误都是等效的。即：以等价类中的某代表值进行的测试，等价于对该类中其它取值的测试。
- 有效等价类：指那些对于软件的规格说明书而言，是合理的、有意义的输入数据所构成的集合。（用于实现功能和性能的测试）。
- 无效等价类：指那些对于软件的规格说明书而言，是不合理的、无意义的输入数据所构成的集合。（用于测试那些所实现的功能和性能不符合规格说明书的要求）。

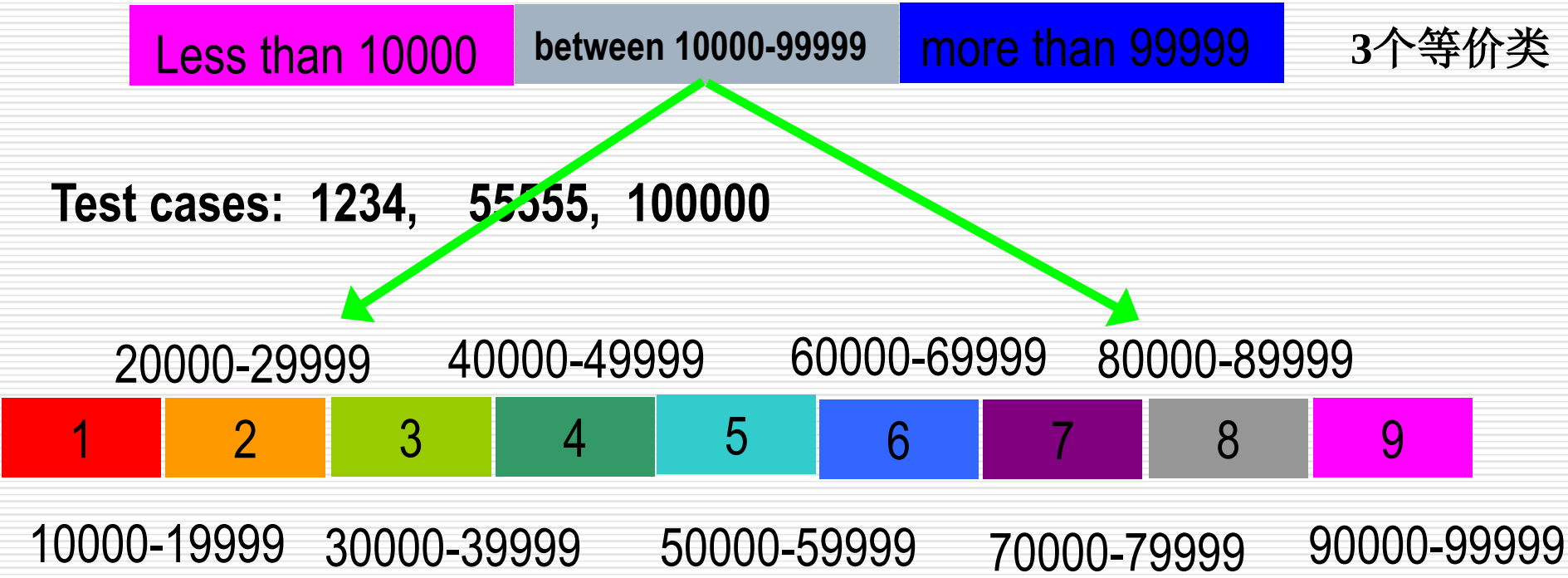
什么是等价类划分

- ✓ **Divide the set of possible values of a problem field into subsets, pick some values to represent each subset.**
- ✓ **The goal is to find a "best representative" for each subset, and to run tests with these representatives so that we can find errors efficiently.**
- ✓ **Equivalence class is the subset**
- ✓ **every element in this subset has the same capability to find errors**

  暴露错误的能力是等价的！

举例

If input is a 5-digit integer between 10,000 and 99,999



Test cases: 11111, 22222, 33333, 44444, 55555, 66666, 77777,

88888, 99999

If input is a 4-Chinese words box for user name

Valid class: 2 words
3 words
4 words

Invalid class: 1 Chines word
English letters
numbers
hidden, control synbols
space

Test cases: 王芳， 王 芳
李国强
诸葛孔明
张
chao qiang, 123456,

How to divide equivalence class ?

? Valid equivalence class (rational …)

? Invalid equivalence class (irrational …)

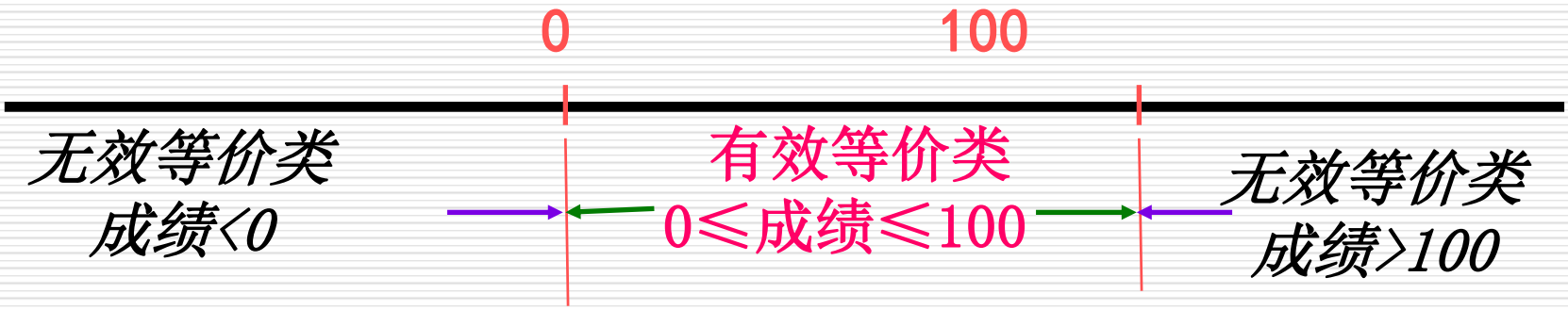
General standard:

- ✓ Cover as more error as possible
- ✓ not intersectant
- ✓ representative
- ✓ 测试精度要求

划分等价类的经验规则

(1)如果输入条件规定了取值范围，可定义一个有效等价类和两个无效等价类。

例： 输入值是学生成绩，范围是0～100



(2)如果输入条件代表集合的某个元素，则可定义一个有效等价类和一个无效等价类。

划分等价类的规则(续)

(3)如规定了输入数据的一组值，且程序对不同输入值做不同处理，则每个允许的输入值是一个有效等价类，并有一个无效等价类(所有不允许的输入值的集合)。

例：输入条件说明学历可为:专科、本科、硕士、博士四种之一，则分别取这四个值作为四个有效等价类，另外把四种学历之外的任何学历作为无效等价类

(4)如果规定了输入数据必须遵循的规则，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)。

(5)如已划分的等价类各元素在程序中的处理方式不同，则应将此等价类进一步划分成更小的等价类。

举例:

某报表处理系统要求用户输入处理报表的日期，日期限制在2003年1月至2008年12月，即系统只能对该段期间内的报表进行处理，如日期不在此范围内，则显示输入错误信息。

系统日期规定由年、月的6位数字字符组成，前四位代表年，后两位代表月。

如何用等价类划分法，设计测试用例，来测试程序的日期检查功能？

Step1：

“报表日期”输入条件的等价类表

输入条件	有效等价类	无效等价类
报表日期的类型及长度	6位数字字符(1)	有非数字字符 (4) 少于6个数字字符 (5) 多于6个数字字符 (6)
年份范围	在2003～2008之间 (2)	小于2003 (7) 大于2008 (8)
月份范围	在1～12之间(3)	小于1 (9) 大于12 (10)

Step2：

为有效等价类设计测试用例

对表中编号为1,2,3的3个有效等价类用一个测试用例覆盖

测试数据	期望结果	覆盖范围
200306	输入有效	等价类(1)(2)(3)

(1)6位数字字符
(2)年在2003～2008之间
(3)月在1～12之间

Step3：

为每一个无效等价类设至少设计一个测试用例

测试数据	期望结果	覆盖范围
003MAY	输入无效	等价类(4)
20035	输入无效	等价类(5)
2003005	输入无效	等价类(6)
200105	输入无效	等价类(7)
200905	输入无效	等价类(8)
200300	输入无效	等价类(9)
200313	输入无效	等价类(10)

不能出现相同的测试用例

本例的10个等价类至少需要8个测试用例

举例:对考试系统“输入学生成绩”子模块设计测试录入准考证号的测试用例

准考证号数据格式定义：共6为数字组成，其中

第一位为专业代号:1-行政专业，2-法律专业，3-财经专业

后5位为考生顺序号，编码范围为：

行政专业准考证号码为：110001～111215

法律专业准考证号码为：210001～212006

财经专业准考证号码为：310001～314015

准考证号码的等价类划分

有效等价类: (1) 110001 ～ 111215

(2) 210001 ～ 212006

(3) 310001 ～ 314015

无效等价类: (4) - ∞ ～ 110000

(5) 111216 ～ 210000

(6) 212007 ～ 31000

(7) 314016 ～ + ∞

Boundary Testing 边界测试

Why？ 现象？

More errors tend to occur at the boundaries of the input domain.

Program is more likely to fail at a boundary.

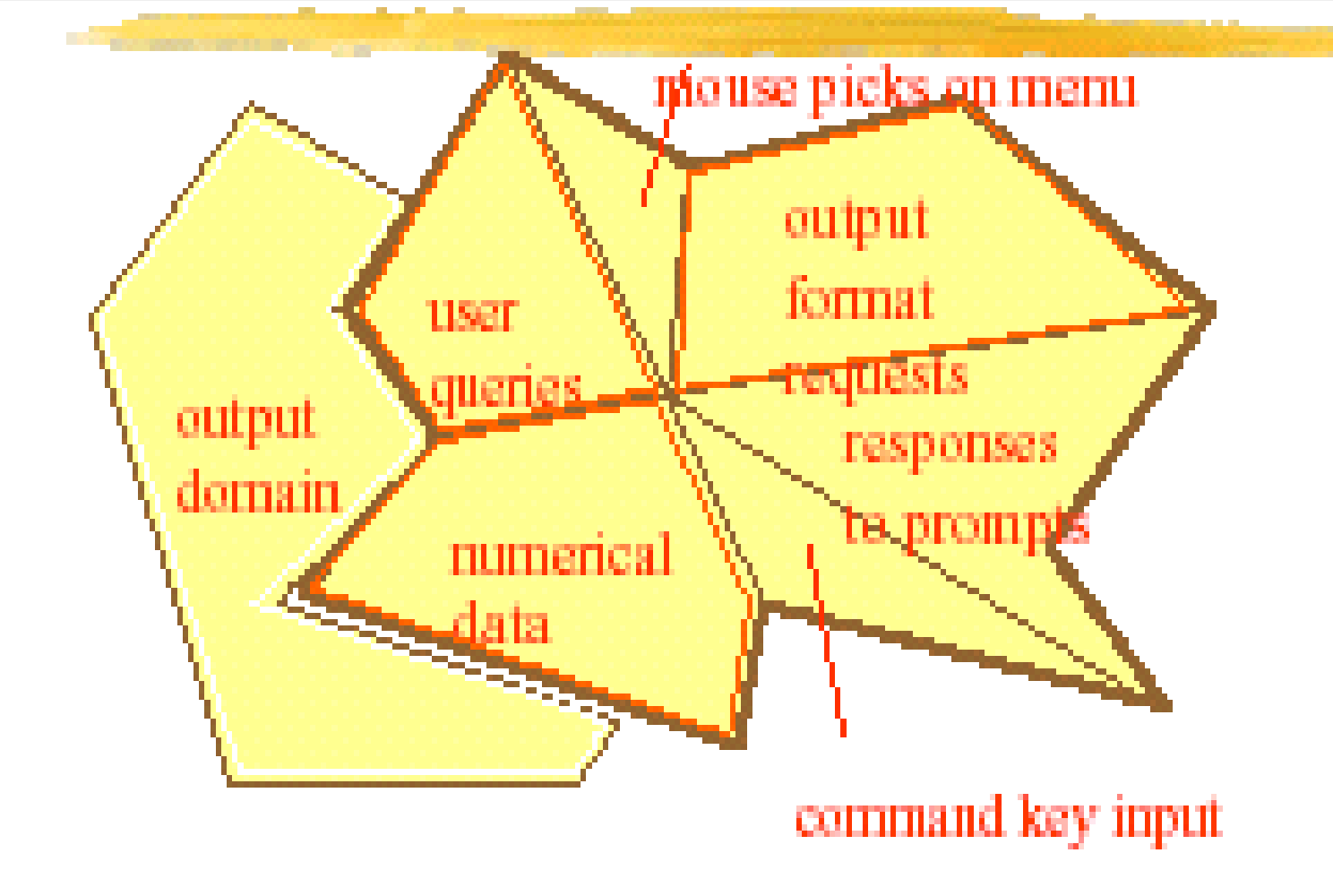
Analogy

软件边界与山路悬崖很类似

如果在悬崖峭壁边可以自信地安全行走，平地就不在话下。

如果软件在能力达到极限时能够运行，那么在正常情况下就不会出什么问题。

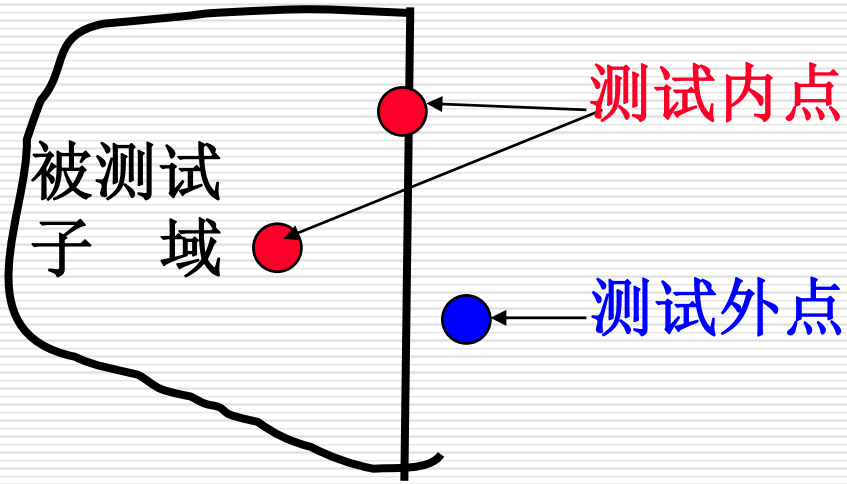
Software boundary



Boundary Vs. Equivalency

(1)边界值分析不是从某等价类中随便挑一个作为代表，而是使这个等价类的每个边界都要作为测试条件。

(2)边界值分析不仅考虑输入条件，还要考虑输出空间产生的测试情况



边界条件类型

如果软件测试问题包含确定的边界，那么数据类型可能是：

- ✓数值
- ✓字符
- ✓位置
- ✓数量
- ✓速度
- ✓地址
- ✓尺寸
- ✓……

还要考虑数据类型的特征

- ✓第一个/最后一个
- ✓最小值/最大值
- ✓开始/完成
- ✓空/满
- ✓最慢/最快
- ✓相邻/最远
- ✓超过/在内
- ✓……

Test case design by boundary

- 总体上：

Select test cases that exercises bounding values

- ✓ 测试边界上的合法数据，以及刚超过边界的非法数据
- ✓ 越界测试通常简单地加1或1个单位；
  减1或1个单位；

“报表日期”边界值分析法测试用例

输入条件	测试用例说明	测试数据	期望结果	选取理由
报表日期的类型及长度	1个数字字符	5	显示出错	仅有1个合法字符
	5个数字字符	20035	显示出错	比有效长度少1
	7个数字字符	2003005	显示出错	比有效长度多1
	有1个非数字字符	2003.5	显示出错	只有1个非法字符
	全部是非数字字符	MAY---	显示出错	6个非法字符
	6个数字字符	200305	输入有效	类型及长度均有效
日期范围	在有效范围边界上选取数据	200301	输入有效	最小日期
		200812	输入有效	最大日期
		200300	显示出错	刚好小于最小日期
		200813	显示出错	刚好大于最大日期
月份范围	月份为1月	200301	输入有效	最小月份
	月份为12月	200312	输入有效	最大月份
	月份<1	200300	显示出错	刚好小于最小月份
	月份>12	200313	显示出错	刚好大于最大月份

Error Guessing(错误推测法)

- 根据经验、直觉和预感来进行测试

例如：

- ✓ 一定要考虑建立处理下列等价类:
  - 缺省值
  - 空白
  - 空值
  - 零值
  - 无输入条件
- ✓ 在已经找到软件缺陷的地方，再找找错误！

Cause and effect diagram(因果图法）

Combination Testing （组合测试）

Edit box1: name

Edit box2: passwd

Edit box3: department

Edit box4: ……

Testing independently ? When interacting among them ?

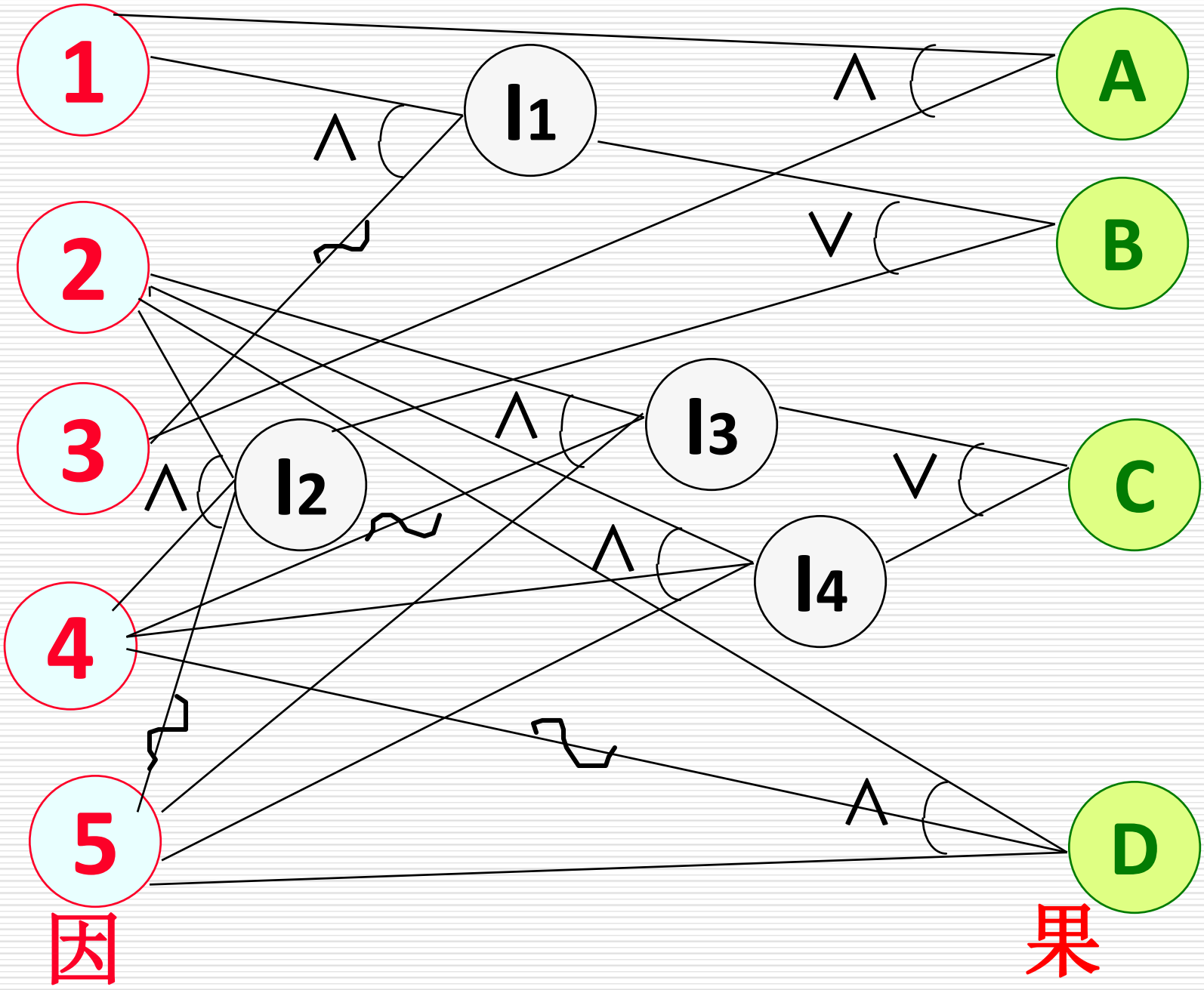
因果图法

等价类划分方法和边界值分析方法，都是着重考虑输入条件，但未考虑输入条件之间的联系，相互组合等。考虑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，他们之间的组合情况也相当多。因此必须考虑采用一种适合于描述对于多种条件的组合，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：

因果图适合于描述对于多种输入条件的组合，相应产生多个动作的形式来设计测试用例。

因果图方法最终生成的是判定表。

用因果图表明输入和输出间的逻辑关系

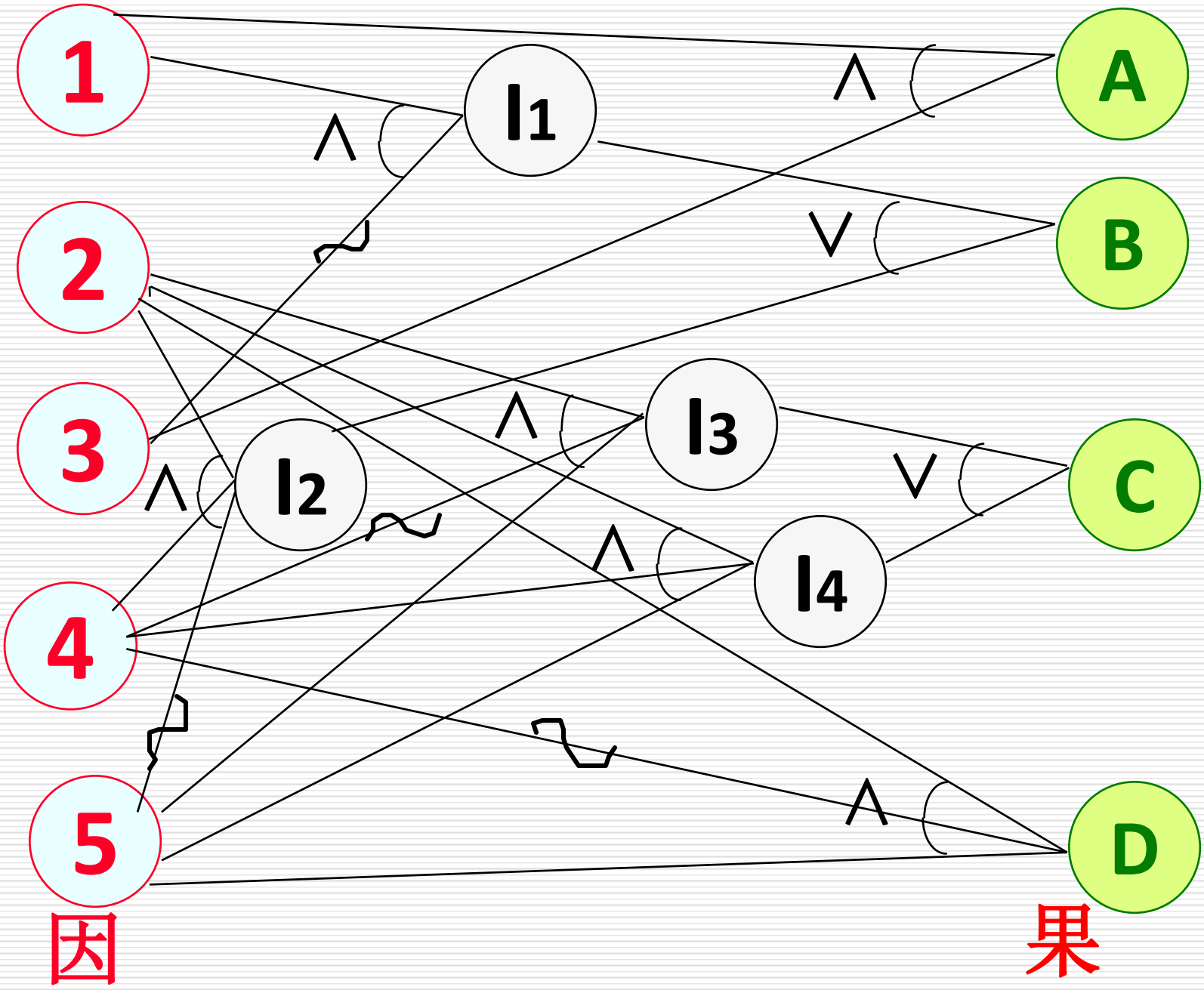


因果图方法实例

某电力公司有A、B、C、D四类收费标准，
并规定：

居民用电	<100度/月		按A类收费
	≥100度/月		按B类收费
动力用电	<10000度/月，	非高峰，	B类收费
	≥10000度/月，	非高峰，	C类收费
	<10000度/月，	高峰，	C类收费
	≥10000度/月，	高峰，	D类收费

用因果图表明输入和输出间的逻辑关系



把因果图转换为判定表

组合条件		1	2	3	4	5	6
条件（原因）	1	1	1	0	0	0	0
	2	0	0	1	1	1	1
	3	1	0				
	4			1	0	1	0
	5			0	0	1	1
动作（结果）	A	1	0	0	0	0	0
	B	0	1	1	0	0	0
	C	0	0	0	1	1	0
	D	0	0	0	0	0	1
测试用例							

为判定表每一列设计一个测试用例：

条件组合	测试用例（输入数据）	预期结果（输出动作）
1列	居民电，90度/月	A
2列	居民电，110度/月	B
3列	动力电，非高峰，8000度/月	B
4列	动力电，非高峰，1.2万度/月	C
5列	动力电，　高峰，0.9万度/月	C
6列	动力电，　高峰，1.1万度/月	D

Automating software testing

- ✓ **Manual software testing and producing a test report is time consuming**
- ✓ **Software testing has to be repeated after every change (regression testing)**
- ✓ **Write test drivers that can run automatically**