



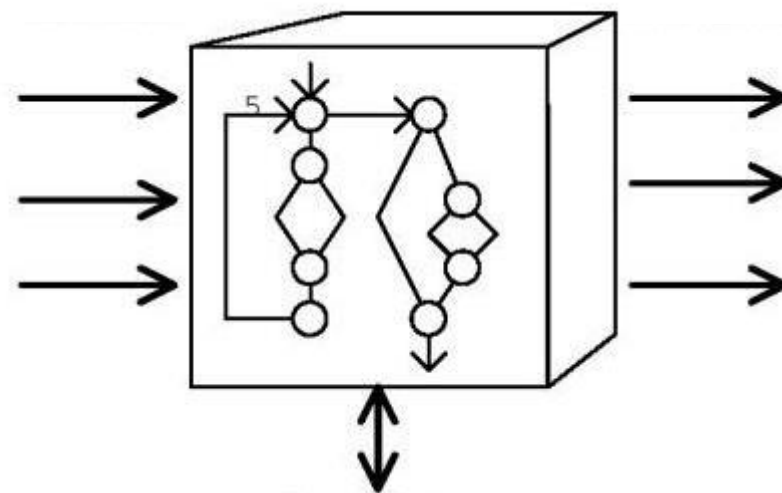
第五部分 软件编码、测试与质量保障

- 5.1 软件编程
- 5.2 软件测试
- 5.3 白盒测试
- 5.4 黑盒测试
- 5.5 变异测试
- 5.6 性能测试

上节回顾：白盒测试的概念

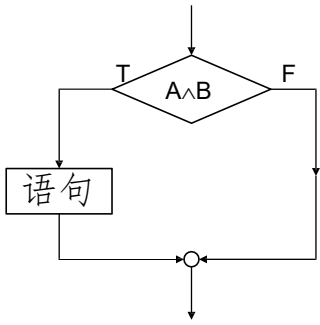
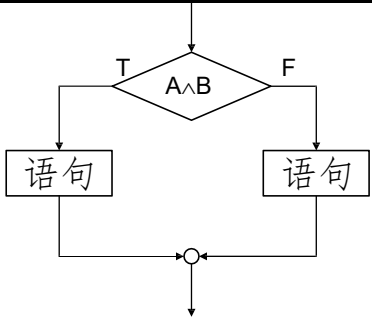
■ 白盒测试(又称为“结构测试”或“逻辑驱动测试”)

- 把测试对象看做一个透明的盒子，它允许测试人员利用程序内部的逻辑结构及有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试。





上节回顾：五种覆盖标准的对比

覆盖标准	程序结构举例	测试用例应满足的条件
语句覆盖		$A \wedge B = T$ 使得被测试程序中的每条可执行语句至少被执行一次。
判定覆盖		$A \wedge B = T$ $A \wedge B = F$ 每一判定的每个分支至少执行一次。



上节回顾：五种覆盖标准的对比

覆盖标准	程序结构举例	测试用例应满足的条件
条件覆盖		$A=T, A=F$ $B=T, B=F$ 每一判定中的每个条件，分别按“真”、“假”至少各执行一次。
判定/条件覆盖		$A \wedge B=T, A \wedge B=F$ $A=T, A=F$ $B=T, B=F$ 同时满足判定覆盖和条件覆盖的要求。
条件组合覆盖		$A=T \wedge B=T$ $A=T \wedge B=F$ $A=F \wedge B=T$ $A=F \wedge B=F$ 求出判定中所有条件的各种可能组合值，每一可能的条件组合至少执行一次。



上节回顾：基本路径测试

- 在程序控制流图的基础上，通过分析控制构造的环路复杂性，导出基本可执行路径集合，从而设计测试用例。
包括以下**4**个步骤：
 1. 以设计或源代码为基础，画出相应的流图
 2. 确定所得流图的环复杂度
 3. 确定线性独立路径的基本集合
 4. 准备测试用例，执行基本集合中每条路径



5.4 黑盒测试

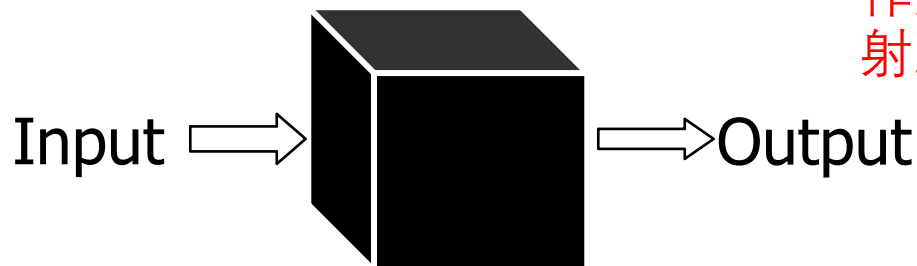
- 黑盒测试概述
- 黑盒测试方法
 - 等价类测试
 - 边界值测试
 - 场景法测试

黑盒测试概念

■ 黑盒测试(black-box testing):

- 又称“功能测试”、“数据驱动测试”或“基于规格说明书的测试”，是一种从用户观点出发的测试。
- 将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说明书，检查程序的功能是否符合它的功能说明。
- 通常在软件接口处进行。

原理：任何程序都可以看作是将输入定义域取值映射到输出值域的函数





黑盒测试能发现的错误

- 是否有不正确或遗漏的功能
- 接口错误
- 数据结构错误或外部信息访问
- 行为或性能错误
- 初始化或终止错误



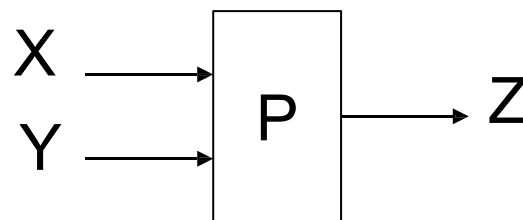
黑盒测试中的“穷举”

- 用黑盒测试发现程序中的错误，必须在所有可能的输入条件和输出条件中确定测试数据，来检查程序是否都能产生正确的输出，但这是不可能的。
——因为穷举测试数量太大，无法完成。



黑盒测试中的“穷举”

- 举例：程序**P**有输入整数**X**和**Y**及输出量**Z**，在字长为**32**位的计算机上运行。
 - 可能采用的测试数据组： $2^{32} \times 2^{32} = 2^{64}$
 - 如果测试一组数据需要**1**毫秒，一年工作**365**×**24**小时，完成所有测试需**5**亿年。



因此，测试人员只能在大量可能的数据中，选取其中一部分作为测试用例。



测试方法的评价标准

测试方法的评价

测试用例的覆盖度: 高

覆盖什么? → 需求

- ✓ 测试以需求为中心
- ✓ 测试用例设计以需求为中心
- ✓ 测试用例应覆盖功能需求
- ✓ 测试用例应覆盖高风险

=

覆盖什么? → 风险

- ✓ 设计测试用例以发现特定缺陷, 确保风险被覆盖
- ✓ 可能的漏测导致遗漏的缺陷, 造成对软件的影响



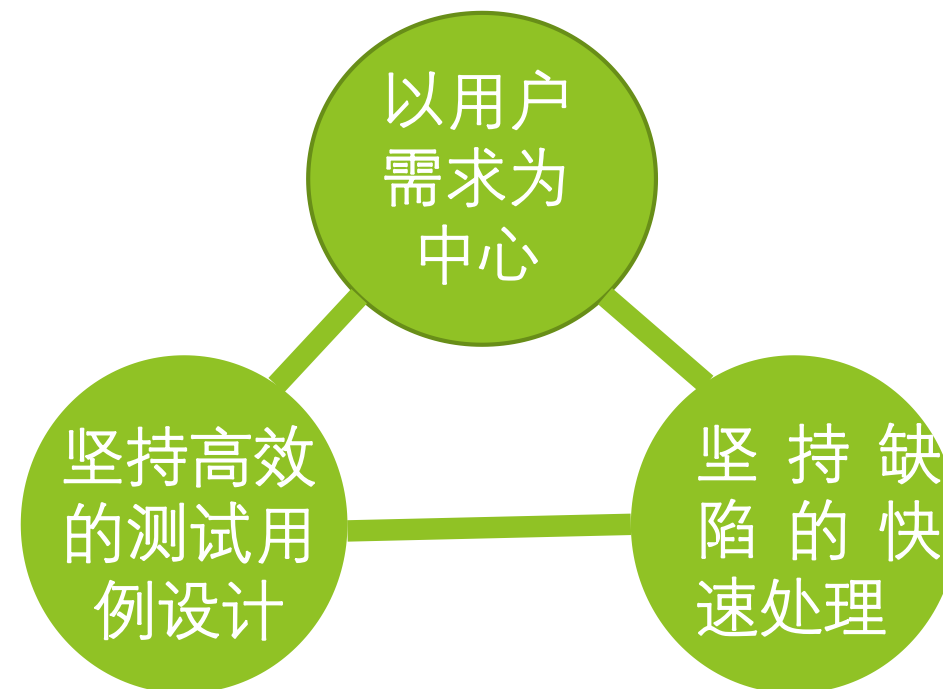


测试方法的评价标准

在**最短时间**内，以**最少的人力**，发现**最多的**，以及**最严重**的缺陷。

测试方法的评价

- 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





黑盒测试方法

黑盒测试并不是白盒测试的替代品，而是用于辅助白盒测试发现其他类型错误。通常由独立测试人员根据用户需求文档来进行，但不一定要求用户参与。





5.4 黑盒测试

- 黑盒测试概述
- 黑盒测试方法
 - 等价类测试
 - 边界值测试
 - 场景法测试



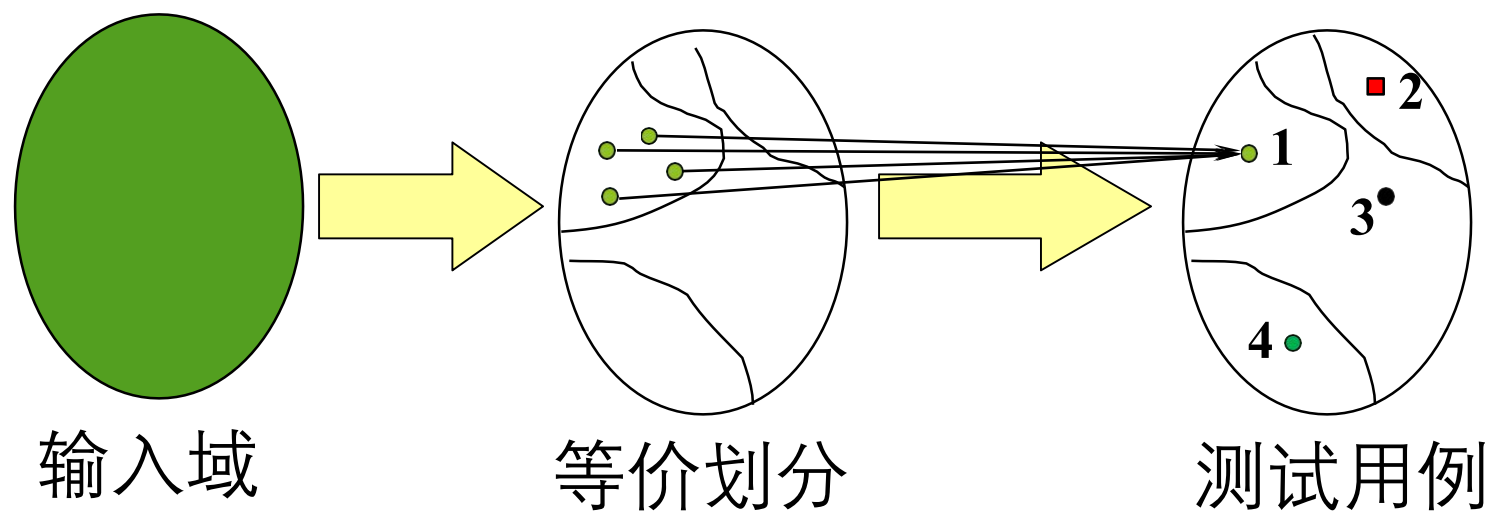
等价类测试



将**无穷多**数据缩减到**有限个**
等价区域中，通过**测试等价**
区域完成穷尽测试。

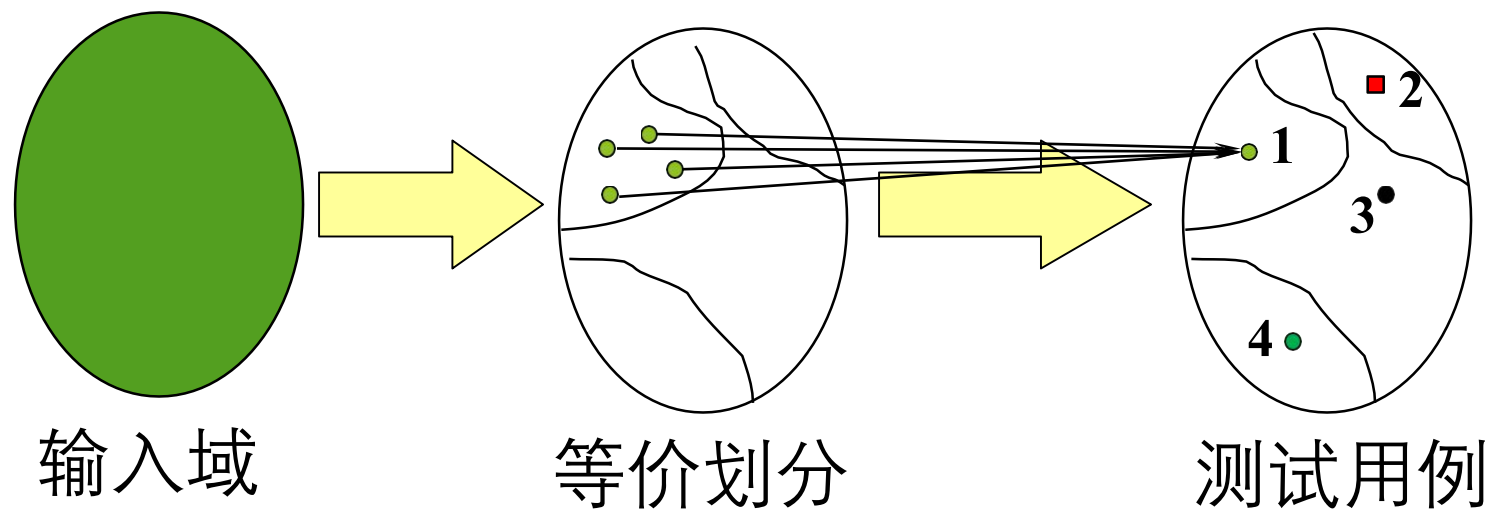
等价类划分

等价类：将程序的输入划分为若干个数据类，从中生成测试用例。
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”。



- 在每一个等价类中选取少量有代表性的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，并取得较好的测试结果。

等价类划分



分而不交：划分出的任意两个等价类之间不存在交集→测试无冗余

合而不交：所有等价类的并集仍然是原始的输入域→测试无漏洞

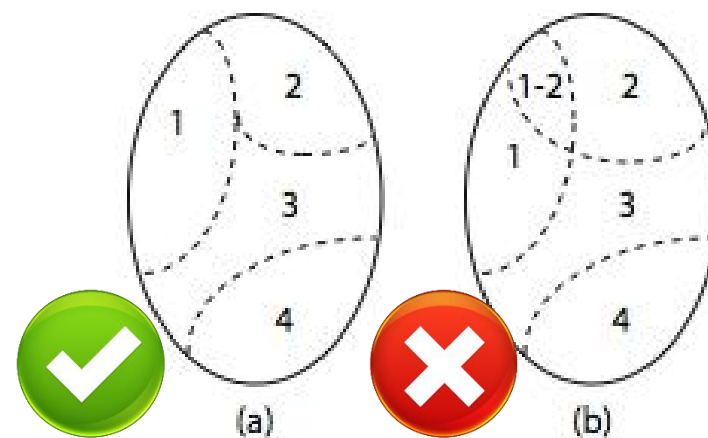
类内等价：任意一个等价类中，所有数据相互“等价”→以一代全

等价类划分(Equivalence partitioning)

■ 关键步骤：确定等价类和选择测试用例

■ 基本原则：

- 每个可能的输入属于某一个等价类
- 任何输入都不会属于多个等价类
- 用等价类的某个成员作为输入时，如果证明执行存在误差，那么用该类的任何其他成员作为输入，也能检查到同样的误差。
- 同一输入域的等价类划分可能不唯一。
- 对于相同的等价类划分，不同测试人员选取的测试用例集可能是不同的，测试用例集的故障检测效率取决于人员经验。



等价类类型

■ 有效等价类

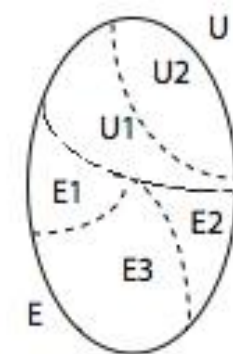
- 输入域中一组有意义的数据的集合。
- 有效等价类被用于检验系统指定功能和性能是否正确实现。

■ 无效等价类

- 输入域中一组无意义的数据的集合。
- 无效等价类被用于检验系统的容错性。
- 无效等价类至少应有一个，也可能有多个。

■ 设计测试用例时，要同时考虑这两种等价类。

- 软件不仅要能接收合理的数据，也要能经受意外的考验，这样的测试才能确保软件具有更高的可靠性。



E 表示所有正常和合法的输入

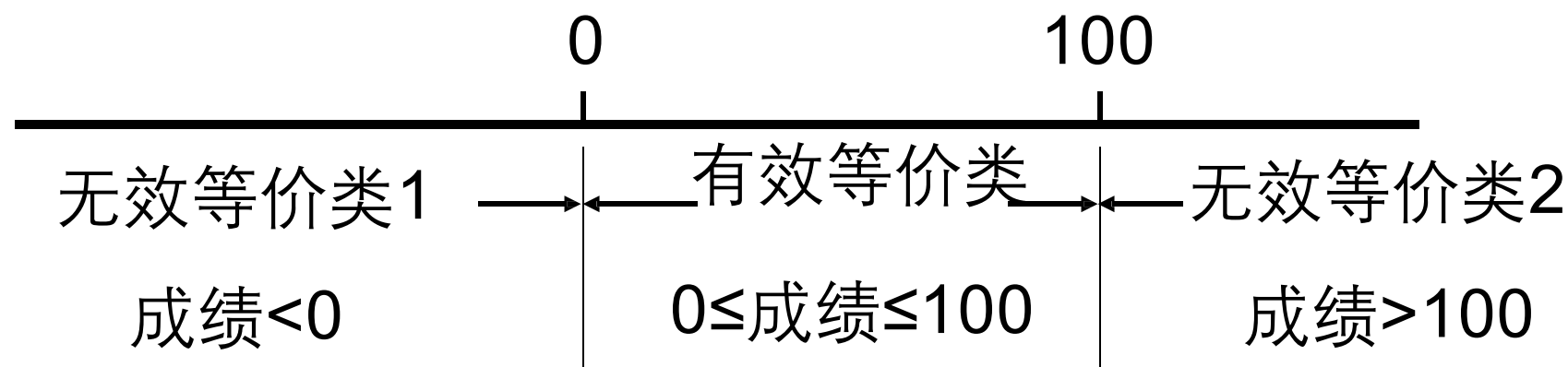
U 表示所有异常和非法的输入



确定等价类的一些建议

建议1: 在输入条件规定了取值范围的情况下，可以确定一个有效等价类和两个无效等价类。

例如：输入值是学生成绩，范围是0~100





确定等价类的一些建议

建议2: 在规定了输入数据必须遵守的规则情况下，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）。

举例: 姓名是长度不超过20的非空字符串，且只由字母组成，数字和其他字符都是非法的。

1个有效等价类: 满足了上述所有条件的字符串

3个无效等价类: 1) 空字符串；2) 长度超过20的字符串；3) 包含了数字或其它字符的字符串



确定等价类的一些建议

建议3：若规定输入数据是一组值（假定 N 个），并且程序要对每一个输入值分别处理，可确定 N 个有效等价类和一个无效等价类。

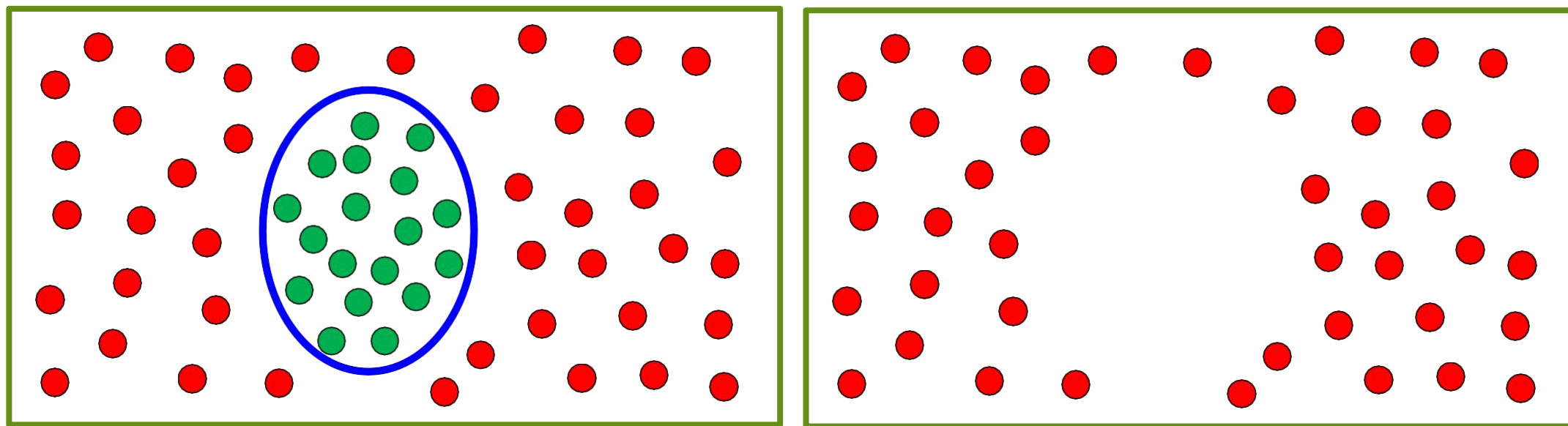
举例：某程序根据不同的学历分别计算岗位工资，其中学历可以是专科、本科、硕士、博士等四种类型。

4个有效等价类：专科、本科、硕士、博士

1个无效等价类：其他学历

确定等价类的一些建议

建议4: 如果某个输入条件指定了一组特定取值 ,
则可以定义一个有效等价类和一个无效等价类。





等价类组合

- **测试用例生成**：测试对象通常有多个输入参数，如何对这些参数等价类进行组合测试，来保证等价类的覆盖率，是测试用例设计首先需要考虑的问题。
- 所有有效等价类的代表值都集成到测试用例中，即覆盖有效等价类的所有组合。任何一个组合都将设计成一个有效的测试用例，也称**正面测试用例**。
- 无效等价类的代表值只能和其他有效等价类的代表值（随意）进行组合。因此，每个无效等价类将产生一个额外的无效测试用例，也称**负面测试用例**。

**等价类的组合将产生数以百计的测试用例，
如何有效地减少测试用例的数目？**



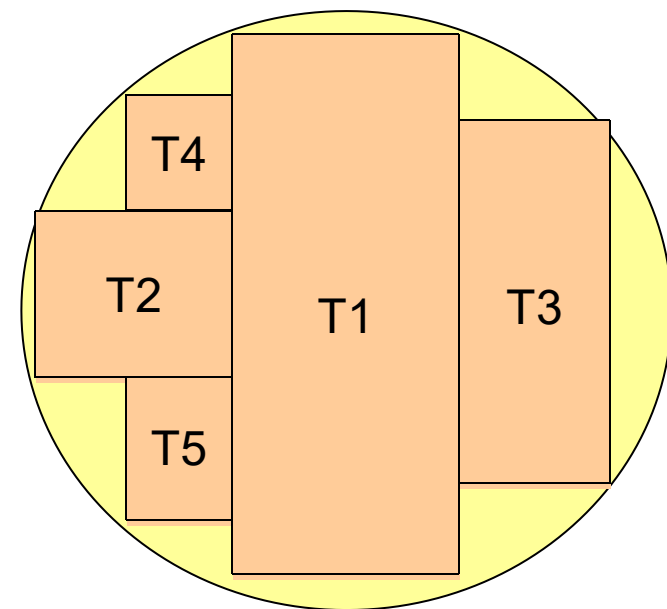
设计测试用例

- 测试用例 = {测试数据+期望结果}
- 测试结果 = {测试数据+期望结果+实际结果}
- 在确立了等价类后，可建立等价类表，列出所有划分出的等价类
输入数据+期望结果：

输入条件	有效等价类	无效等价类
...
...

设计测试用例

- 为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，使其尽可能多的覆盖尚未被覆盖的有效等价类；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





例1：数据录入问题

- **[例1]**某一程序要求输入数据满足以下条件：
 - 可输入1个或多个数据，每个数据由1-8个字母或数字构成，且第一个字符必须为字母；
 - 如果满足上述条件，则判断“合法数据”；否则，判断“非法数据”；
- 用等价类划分方法为该程序进行测试用例设计。



例1：数据录入问题

划分等价类：

输入条件	有效等价类	号码	无效等价类	号码
数据个数	1个	(1)	0个	(6)
	多个	(2)		
标识符字符数	1~8个	(3)	0个	(7)
			>8个	(8)
标识符组成	数字与字母	(4)	含有非“字母或数字”	(9)
第一个标识符	字母	(5)	非字母	(10)



例1：数据录入问题

设计测试用例：

测试用例编号	测试用例内容	覆盖的等价类
1	a2ku83t	(1) (3) (4) (5)
2	a2ku83t, fta, b2	(2) (3) (4) (5)
3		(6)
4	a2ku83t, , b2	(7)
5	a2ku83t29, fta	(8)
6	a2ku8\$t	(9)
7	92ku83t	(10)



例2：日期检查

- **[例2]** 档案管理系统，要求用户输入以年月表示的日期。假设日期限定在**1990**年**1**月~**2049**年**12**月，并规定日期由**6**位数字字符组成，前**4**位表示年，后**2**位表示月。
- 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”。



例2： 日期检查

划分等价类：

输入等价类	有效等价类	无效等价类
日期的类型及长度	①6位数字字符	②有非数字字符 ③少于6位数字字符 ④多于6位数字字符
年份范围	⑤在1990~2049之间	⑥小于1990 ⑦大于2049
月份范围	⑧在01~12之间	⑨等于00 ⑩大于12



例2： 日期检查

设计有效等价类的测试用例：

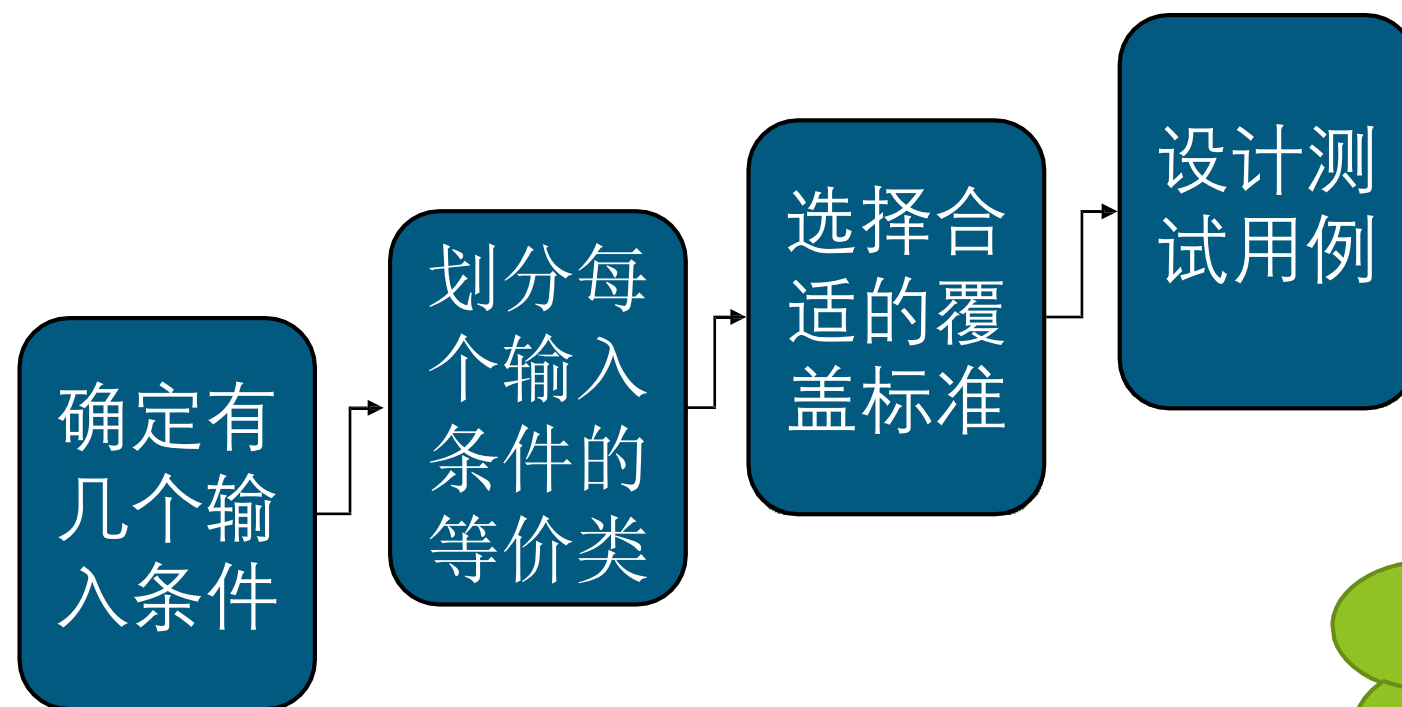
测试用例编号	测试用例内容	覆盖的等价类
1	200711	(1) (5) (8)

设计无效等价类的测试用例：

测试用例编号	测试用例内容	覆盖的等价类
1	07June	(2)
2	20076	(3)
3	2007011	(4)
4	198912	(6)
5	205401	(7)
6	200700	(8)
7	200713	(10)



等价类测试的流程





等价类测试的陷阱1

确定输入条件时，可能会改变原始输入域

- 后一日问题。
- 针对从1800年到2050年之间的任意一个日期，计算出其下一天的日期；
- 否则，给出错误提示。



其中闰年有61个，共包含91676个日期



等价类测试的陷阱1

确定输入条件时，可能会改变原始输入域

输入条件	有效等价类	无效等价类	
年	[1800, 2050]	$(-\infty, 1800)$	$(2050, +\infty)$
月	[1, 12]	$(-\infty, 1)$	$(12, +\infty)$
日	[1, 31]	$(-\infty, 1)$	$(31, +\infty)$

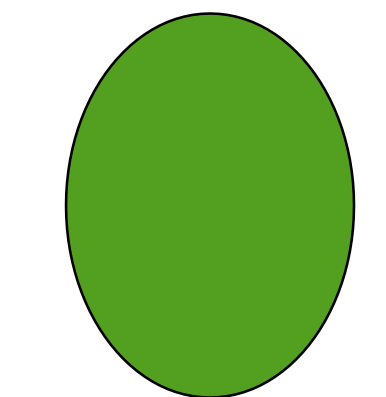
日期的有效输入域
是1号到31号



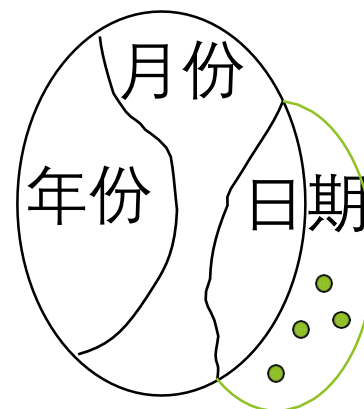
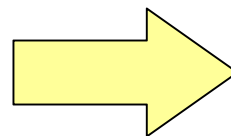
其中闰年有61个，共包含91676个日期

等价类测试的陷阱1

确定输入条件时，可能会改变原始输入域



原始输入域



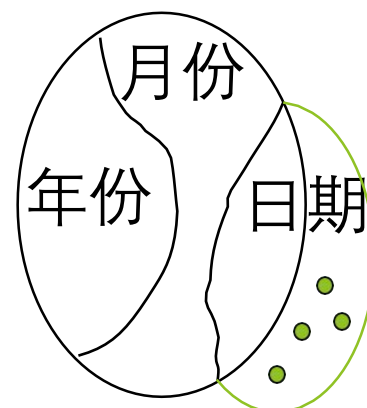
等价划分

原始输入域不等于
等价划分的域



等价类测试的陷阱1

在对原始输入域进行等价划分时，如果这种等价划分改变了原始的输入域，还能使用等价类测试方法设计测试用例吗？



等价划分

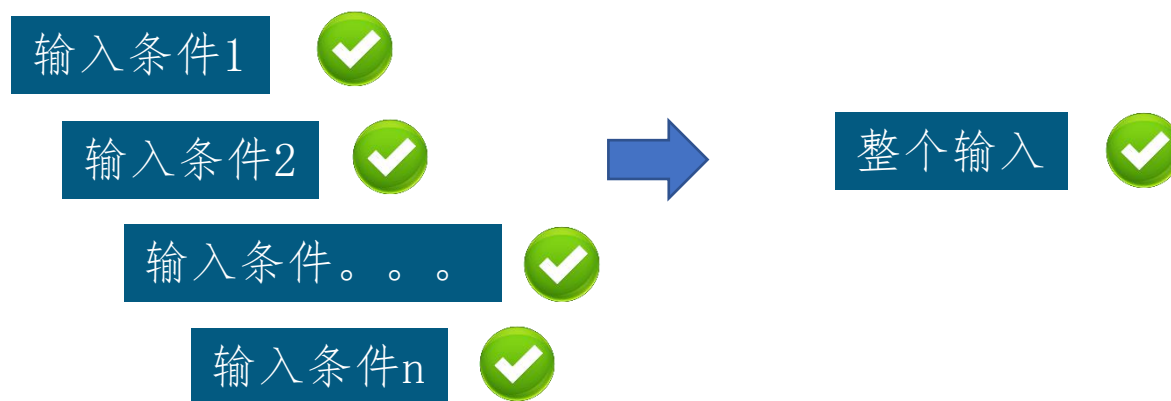


等价类测试的陷阱2

对有效域和无效域可以用相同方式进行等价类测试吗？

有效等价类

- 有效域内的测试用例如果要正常运行，要求所有的输入条件都必须是有有效的。



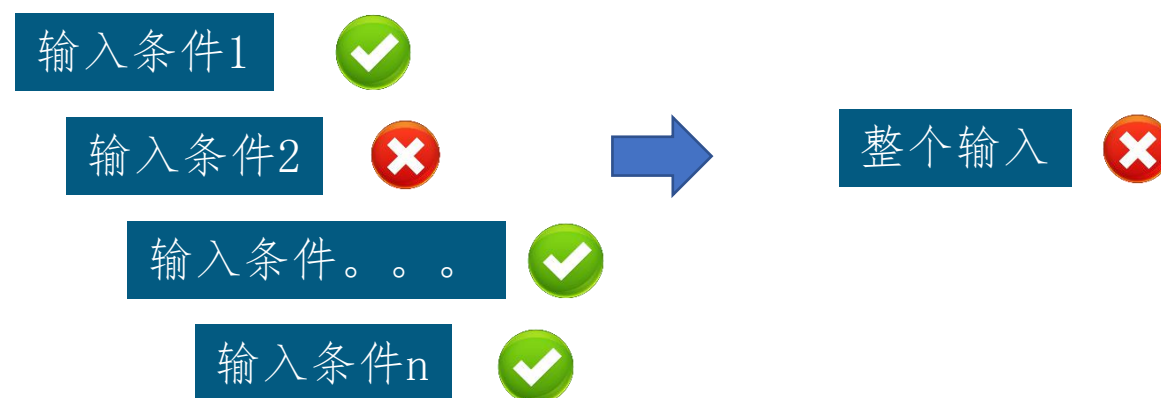


等价类测试的陷阱2

对有效域和无效域可以用相同方式进行等价类测试吗？

无效等价类

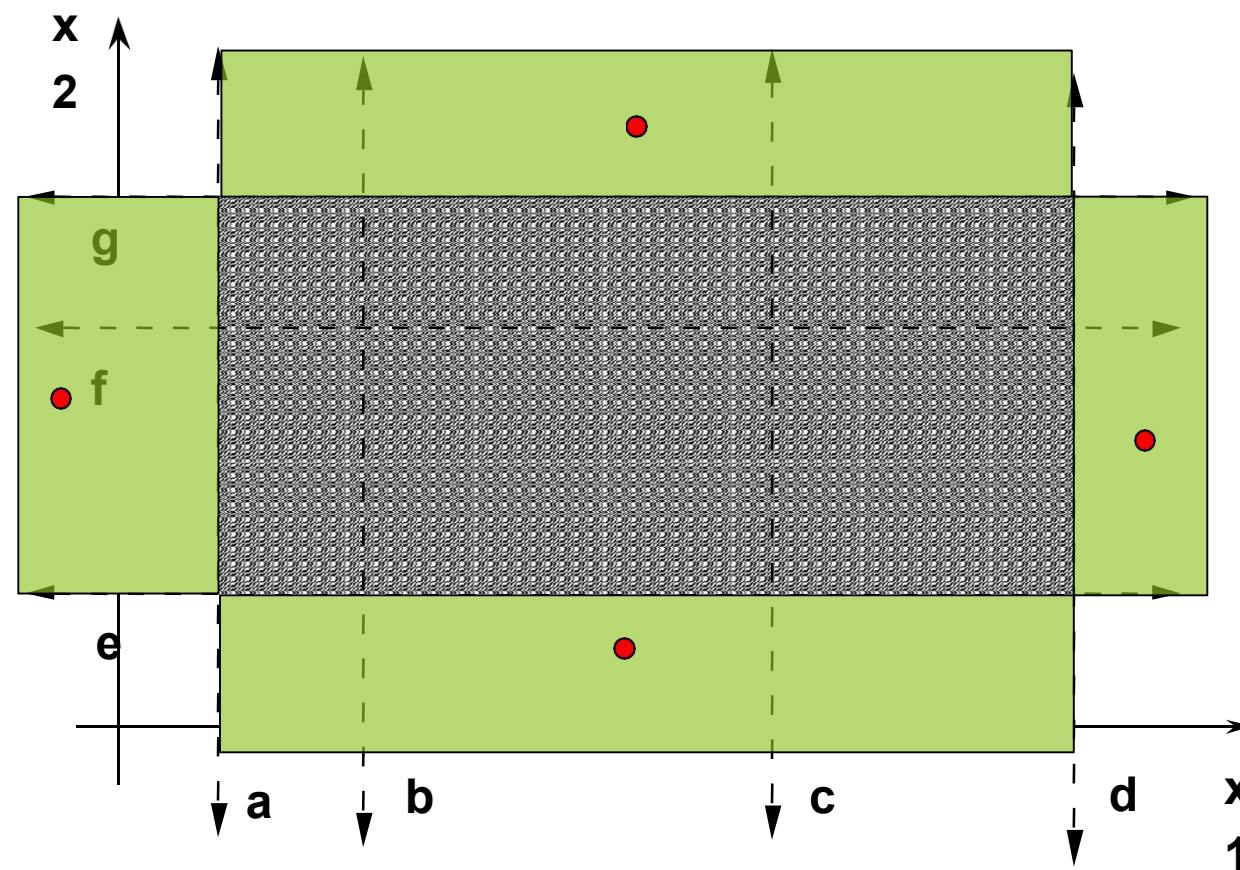
- 无效等价类对应的都是系统所不能接受的



等价类测试的陷阱2

对无效等价类设计测试用例时：**采用单缺陷原则**

有效域内设计测试用例，应确保每个测试用例覆盖的均为有效等价类，在无效域内，则应基于单缺陷原则设计测试用例，不应该出现多个无效等价类组合成测试用例的情况。





选取等价类的测试用例

- 由所有代表值组合而成的测试用例按使用频率（典型的使用特征）进行排序，并按照这个序列设置优先级，这样就能仅对相关的测试用例（典型的组合）进行测试。
- 优先考虑包含边界值或者边界值组合的测试用例。
- 将一个等价类的每个代表值和其他等价类的每个代表值进行组合来设计测试用例（即双向组合代替完全组合）。
- 保证满足最小原则：一个等价类的每个代表值至少在一个测试用例中出现。
- 无效等价类的代表值不与其他无效等价类代表值进行组合。



举例：判断三角形类型

输入三个整数**a**、**b**、**c**，分别作为三角形的三条边，现通过一个程序判断这三条边构成的三角形类型，包括等边三角形、等腰三角形、一般三角形（特殊的还有直角三角形）以及构不成三角形。

现在要求输入的三个整数**a**、**b**、**c**必须满足以下条件：

条件1: $1 \leq a \leq 100$

条件4: $a < b + c$

条件2: $1 \leq b \leq 100$

条件5: $b < a + c$

条件3: $1 \leq c \leq 100$

条件6: $c < a + b$

请使用等价类划分方法，设计该程序的测试用例。



举例：判断三角形类型

等价类划分：

①按输入取值划分

$\{<1, 1\sim 100, >100\}$

②按输出的几何特性划分

$\{\text{等腰且非等边三角形, 等边三角形, 不等边三角形, 非三角形}\}$

你会选择哪一种方法划分等价类？



举例：判断三角形类型

标准等价类测试用例

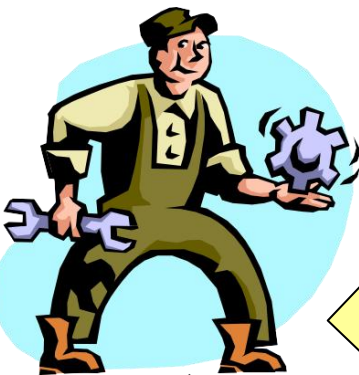
序号	测试用例描述	输入参数			期望输出
		a	b	c	
1	$a > 0, b > 0, c > 0$ $a + b > c, b + c > a, a + c > b$ $a = b = c$	10	10	10	等边三角形
2	$a > 0, b > 0, c > 0$ $a + b > c, b + c > a, a + c > b$ $a = b \neq c$ 或 $b = c \neq a$ 或 $a = c \neq b$	10	10	5	等腰三角形
3	$a > 0, b > 0, c > 0$ $a + b > c, b + c > a, a + c > b$ $a \neq b \neq c$	3	4	5	一般三角形
4	$a > 0, b > 0, c > 0$ $a + b \leq c$ 或 $b + c \leq a$ 或 $a + c \leq b$	4	1	2	非三角形



举例：判断三角形类型

增加无效等价类测试用例

序号	测试用例描述	输入参数			期望输出
		a	b	c	
1-4	有效等价类同前面（略）				
5	$a < 0, b > 0, c > 0$	-1	5	5	a 值越界
6	$a > 0, b < 0, c > 0$	5	-1	5	b 值越界
7	$a > 0, b > 0, c < 0$	5	5	-1	c 值越界
8	$a > 100, b > 0, c > 0$	101	5	5	a 值越界
9	$a > 0, b > 100, c > 0$	5	101	5	b 值越界
10	$a > 0, b > 0, c > 100$	5	5	101	c 值越界



生产

最低销售要求：一支完整的步枪

月供货：枪机70个，枪托80个，
枪管90个



步枪

月销售提成
比例

- ◆ $[0, 1000] \rightarrow 0.1$
- ◆ $(1000, 1800] \rightarrow 0.15$
- ◆ $(1800, +\infty) \rightarrow 0.20$

输出：销售商的月佣金



供货

单价



销售商





动动手：设计测试用例

需求

- 输入：枪机、枪托、枪管的月销售量
- 输出：销售商的提成

枪机销售量的边界点是：1, 70

枪托销售量的边界点是：1, 80

枪管销售量的边界点是：1, 90



序号	枪机销售量 (个)	枪托销售量 (个)	枪管销售量 (个)	销售额 预 (美元)	期输出 (提成 (美元)
1	35	40	45	3900	

有效等价类
$L1 = \{\text{枪机销售量} \mid 1 \leq \text{枪机销售量} \leq 70\}$
$S1 = \{\text{枪托销售量} \mid 1 \leq \text{枪托销售量} \leq 80\}$
$B1 = \{\text{枪管销售量} \mid 1 \leq \text{枪管销售量} \leq 90\}$

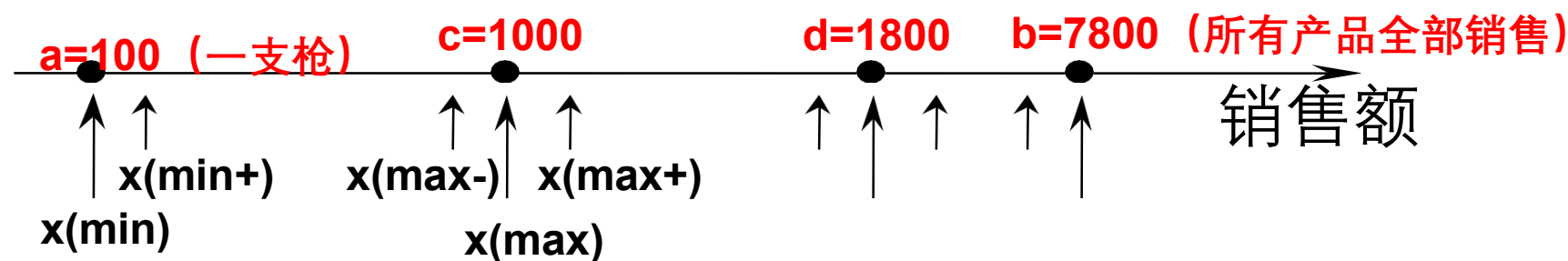
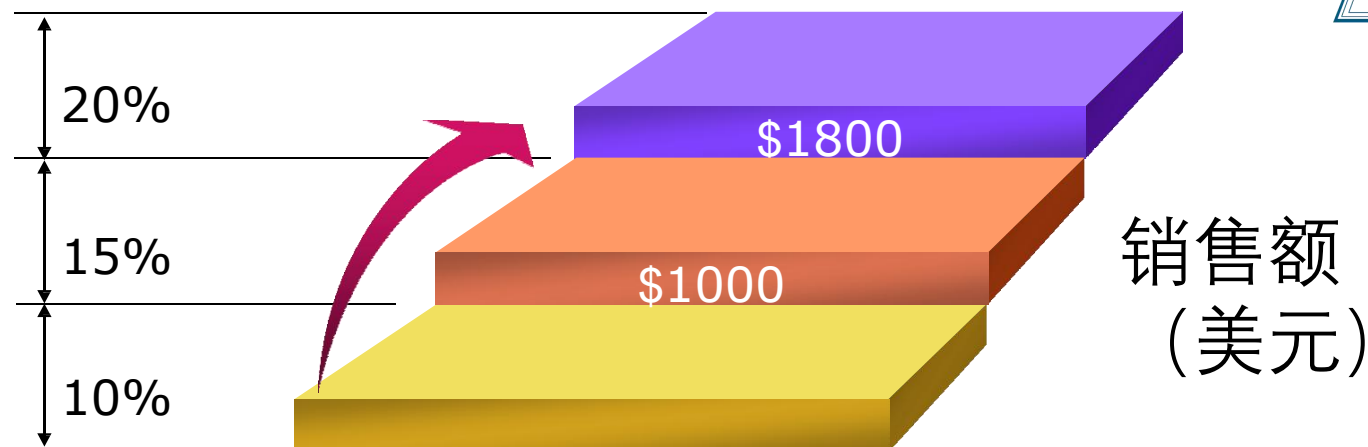
↓
>1800 美元

↓
提成比例: 20%

↓
无法覆盖提成比例
: 10%, 15%



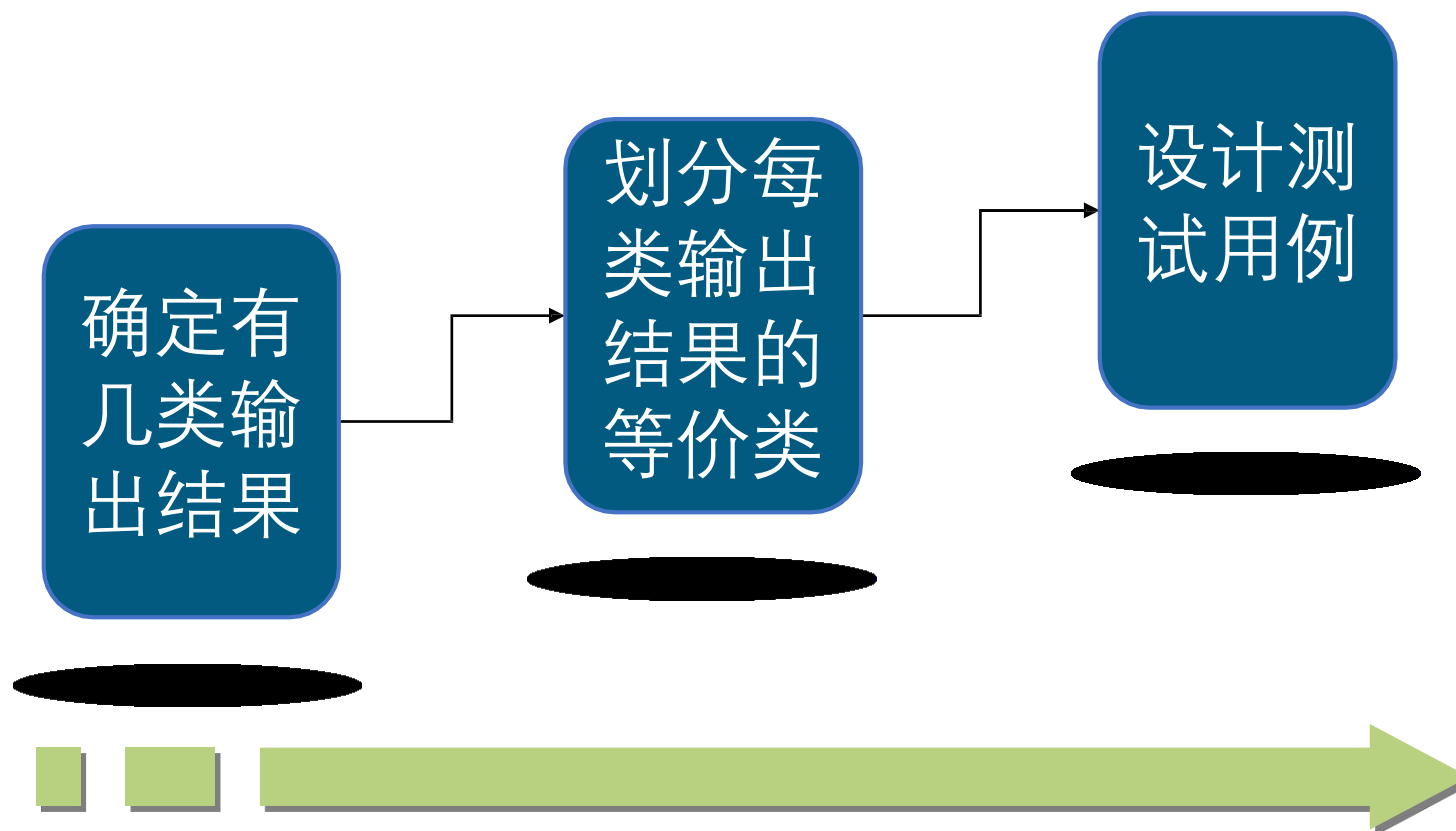
最大值
\$1800
\$1000
最小值



序号	枪机销售量 (个)	枪托销售量 (个)	枪管销售量 (个)	销售额 (美元)	预期输出 (提成) (美元)
1				600	
2				1400	
3				4800	



输出域的等价类测试的流程



输入域: 需要严格区分有效域和无效域

输出域: 不存在无效输出域的概念



5.4 黑盒测试

- 黑盒测试概述
- 黑盒测试方法
 - 等价类测试
 - 边界值测试
 - 场景法测试



边界值测试

产生的原因

- 经过长期的测试工作经验表明，在输入域的边界或边界附近，常常会发现大量缺陷
- 边界值测试倾向于选择系统边界或边界附近的数据来设计测试用例



边界值测试

边界值分析是对输入或输出的边界值进行测试的一种方法，它通常作为等价类划分法的补充，这种情况下的测试用例来自等价类的边界。

- 首先确定边界情况。通常输入或输出等价类的边界就是应该着重测试的边界情况。
- 选取正好等于、刚刚大于或刚刚小于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。



边界值测试

- 边界在哪里?
- 如何定义边界的邻域?
- 如何选择测试数据?
- 如何设计测试用例?





动动手：设计边界值测试用例

`int Add(int x1, int x2)`

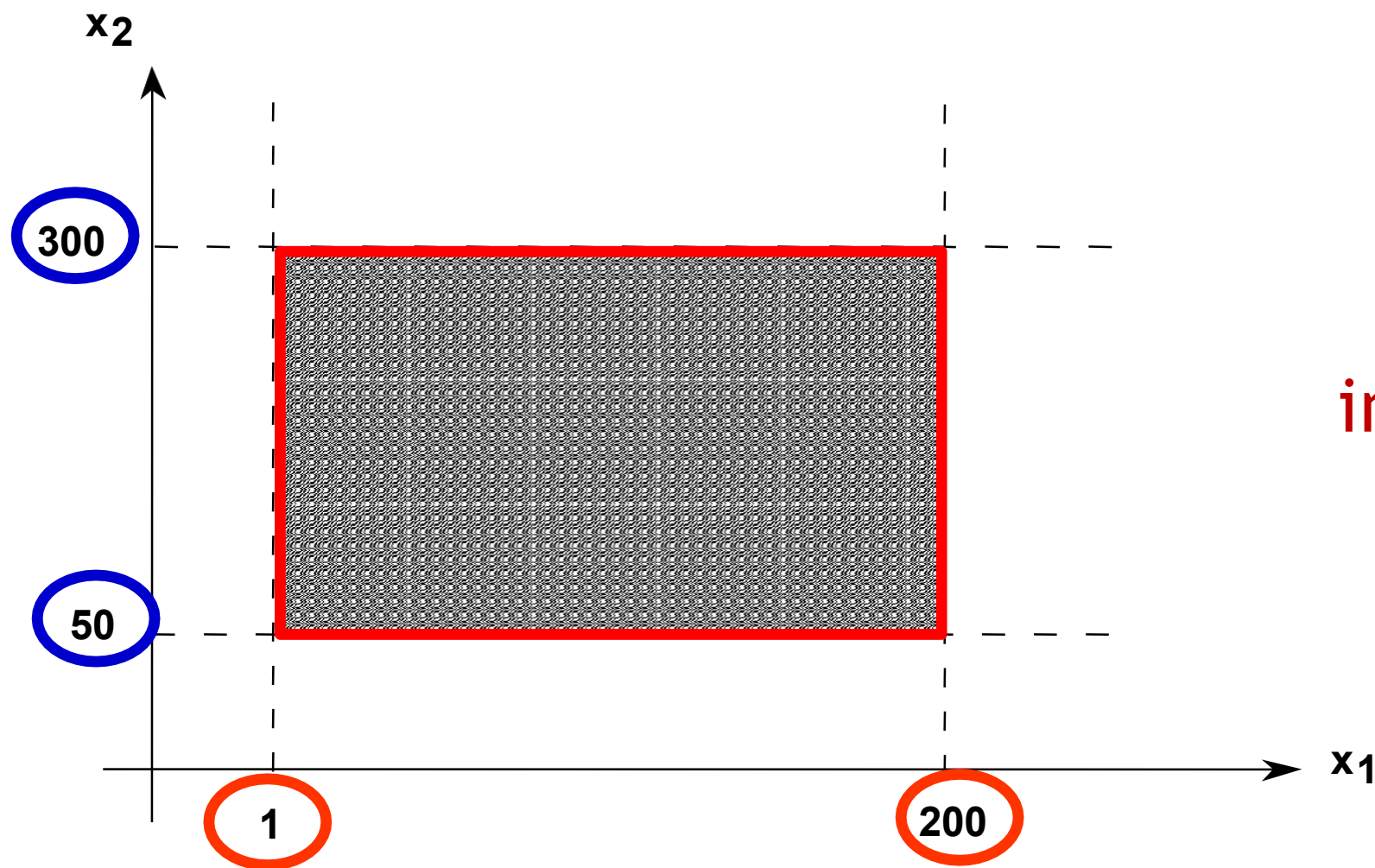
- $1 \leq x1 \leq 200$
- $50 \leq x2 \leq 300$

需求简述：

- 对于有效输入，函数返回 **x1** 与 **x2** 的和；
- 对于无效输入，函数返回 **-1**；



1. 边界在哪里?



`int Add(int x1, int x2)`

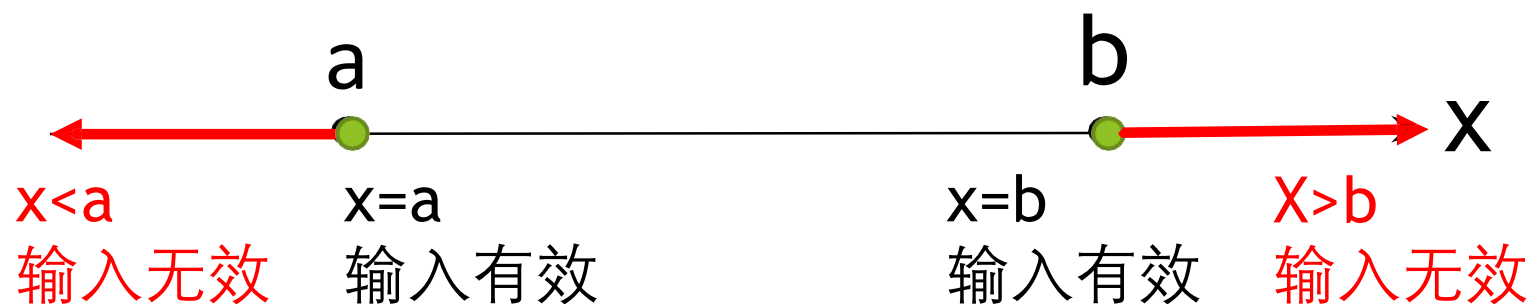
$1 \leq x_1 \leq 200$

$50 \leq x_2 \leq 300$



1. 边界在哪里?

边界点就是可能导致被测系统内部处理机制发生变化的点



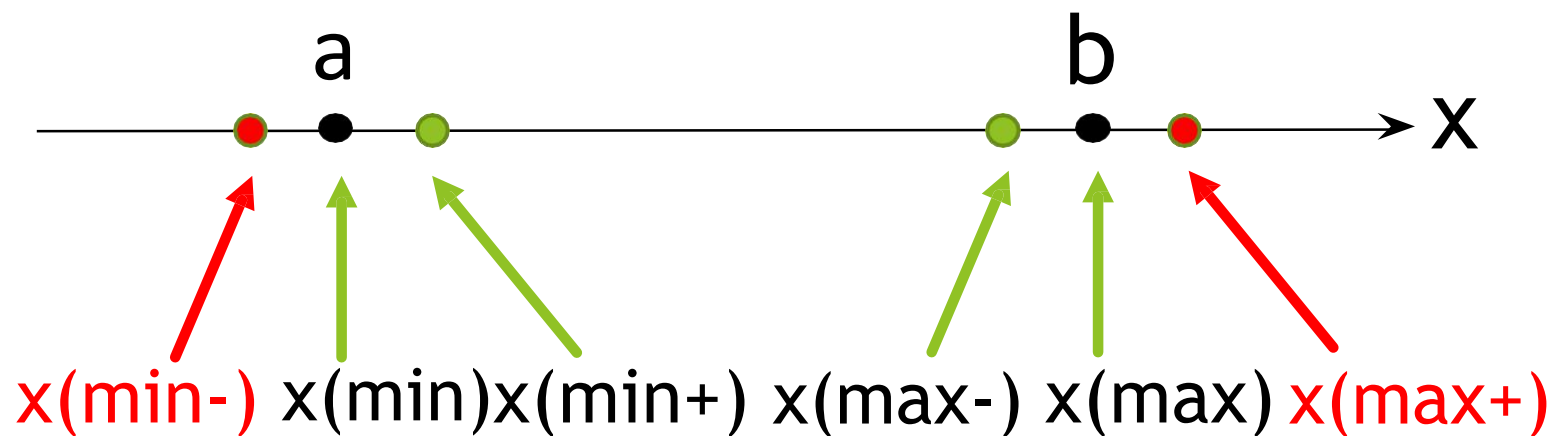
- $a \leq x \leq b$
- 边界点: a, b



1. 边界在哪里？

输入项	边界值	测试用例的设计思路
字符	起始 - 1个字符 结束 + 1个字符	假设一个文本输入区域允许输入1到255个字符，输入1个和255个字符作为有效等价类；输入0个和256个字符作为无效等价类，这几个数值都属于边界条件值。
数值	最小值 - 1 最大值 + 1	假设某软件要求输入5位十进制整数值，可以使用10000作为最小值、99999作为最大值；然后使用刚好小于5位和大于5位的数值作为边界条件。
空间	小于空余空间一点 大于满空间一点	例如在用U盘存储数据时，使用比剩余磁盘空间大一点（几 KB）的文件作为边界条件。

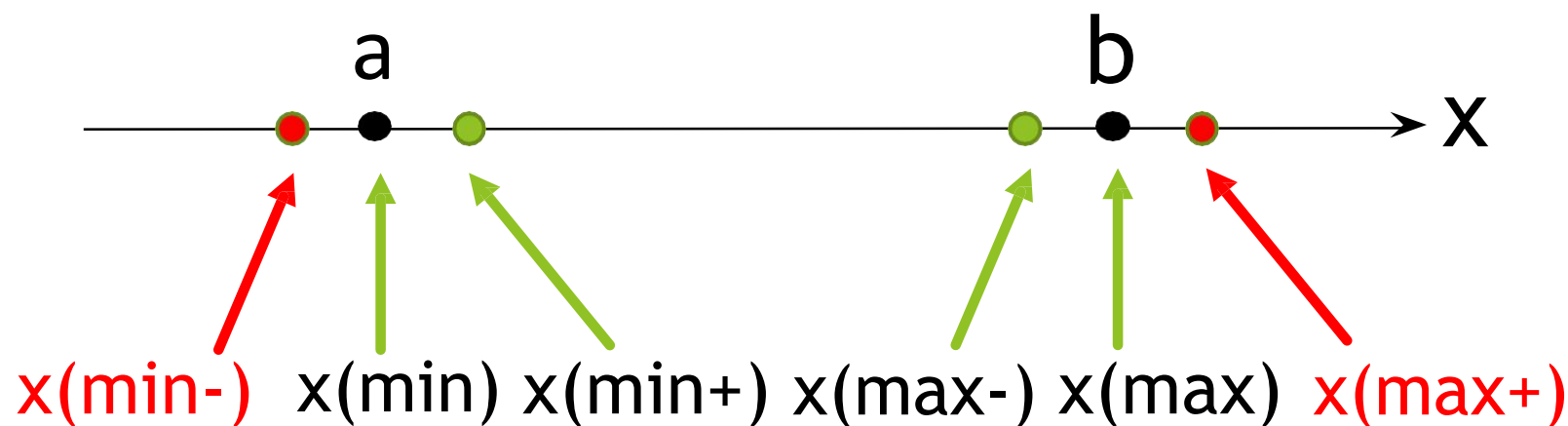
2. 如何定义邻域?



- $a \leq x \leq b$
- 边界点: a, b
- 邻域: $[a-\delta 1, a+\delta 1], [b-\delta 2, b+\delta 2]$
- 提问: $\delta 1 = \delta 2 ?$



3. 如何选择测试数据？



给定: $a \leq x \leq b$

x 的边界点: a, b

邻域: $[a-\delta_1, a+\delta_1], [b-\delta_2, b+\delta_2]$

测试数据: $[a-\delta_1, a+\delta_1], [b-\delta_2, b+\delta_2]$



编号	x1	x2	预期输出
1	0	200	-1
2	1	200	201
3	2	200	202
4	199	200	399
5	200	200	400
6	201	200	-1
7	100	49	-1
8	100	50	150
9	100	51	151
10	100	299	399
11	100	300	400
12	100	301	-1

int Add(int x1, int x2)

$1 \leq x1 \leq 200$

$50 \leq x2 \leq 300$



边界值分析举例

- 例如：如果程序的规格说明中规定“重量在**10**公斤至**50**公斤范围内的邮件，其邮费计算公式为.....”。
- 作为测试用例，可以为多少？



边界值分析举例

- 例如：如果程序的规格说明中规定“重量在**10**公斤至**50**公斤范围内的邮件，其邮费计算公式为.....”。
 - 作为测试用例，可以为多少？
 - 10、50、10.01、49.99、9.99、50.01



边界值分析举例

- 例如：一个输入文件应包括**1~255**个记录.....”。
- 作为测试用例，可以为多少？



边界值分析举例

- 例如：一个输入文件应包括**1~255**个记录.....”。
- 作为测试用例，可以为多少？
- 输入**1**和**255**，**2**及**254**个记录，以及**0**和**256**个记录



边界值分析举例

- 例如，某程序的规格说明要求计算出“每月保险金扣除额为0至1165.25元”。。。。
 - 作为测试用例，可以为多少？

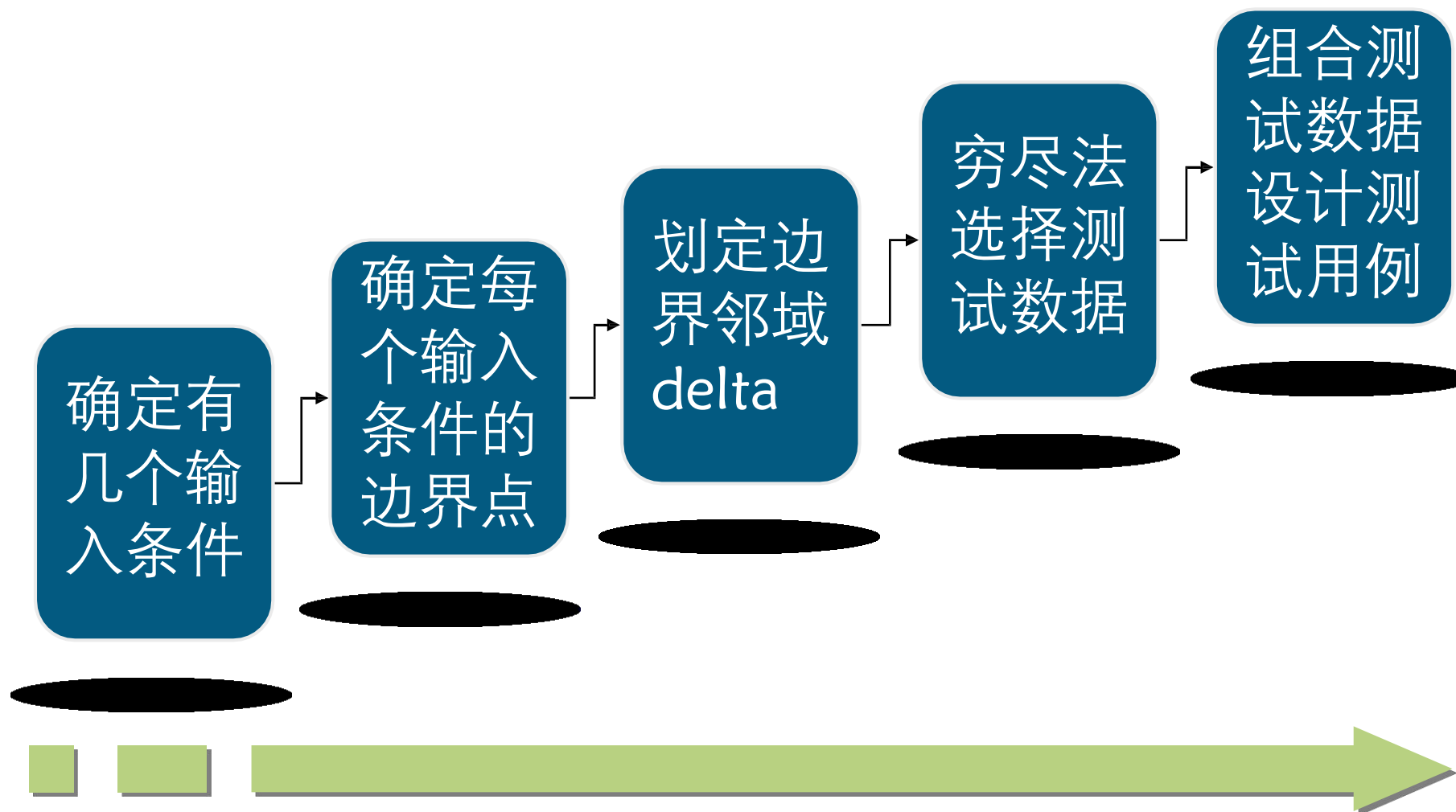


边界值分析举例

- 例如，某程序的规格说明要求计算出“每月保险金扣除额为0至1165.25元。。。”
 - 作为测试用例，可以为多少？
 - 创建测试用例使输出为0.00、1165.25和0.01、1165.24以及-0.01、1165.26等。



边界值测试

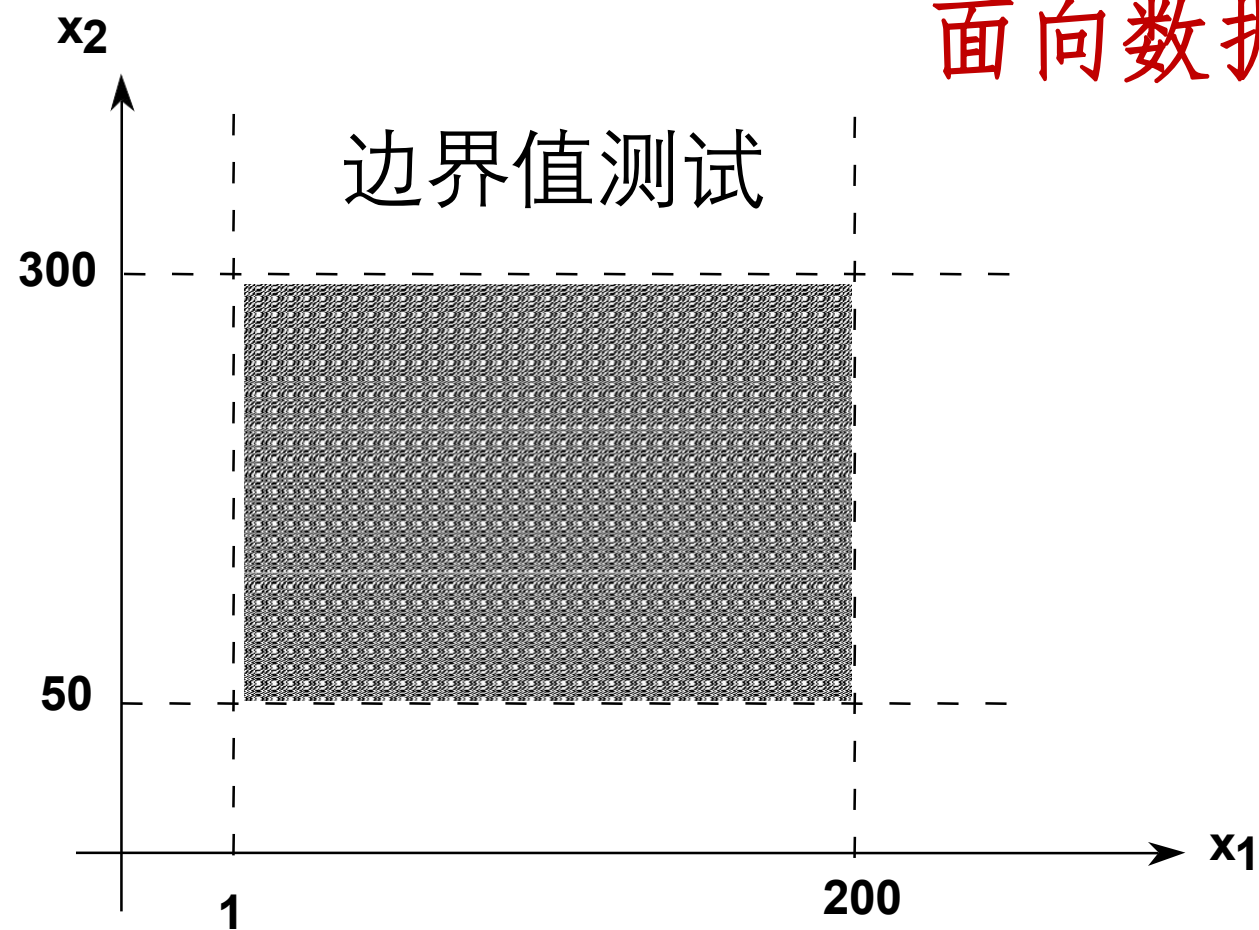




5.4 黑盒测试

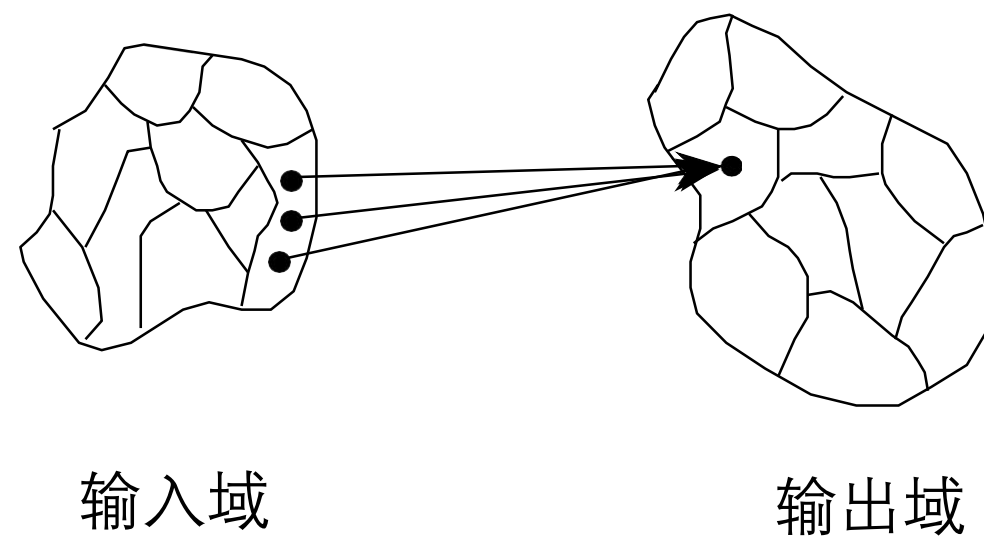
- 黑盒测试概述
- 黑盒测试方法
 - 等价类测试
 - 边界值测试
 - 场景法测试

场景法测试



面向数据的测试

等价类测试





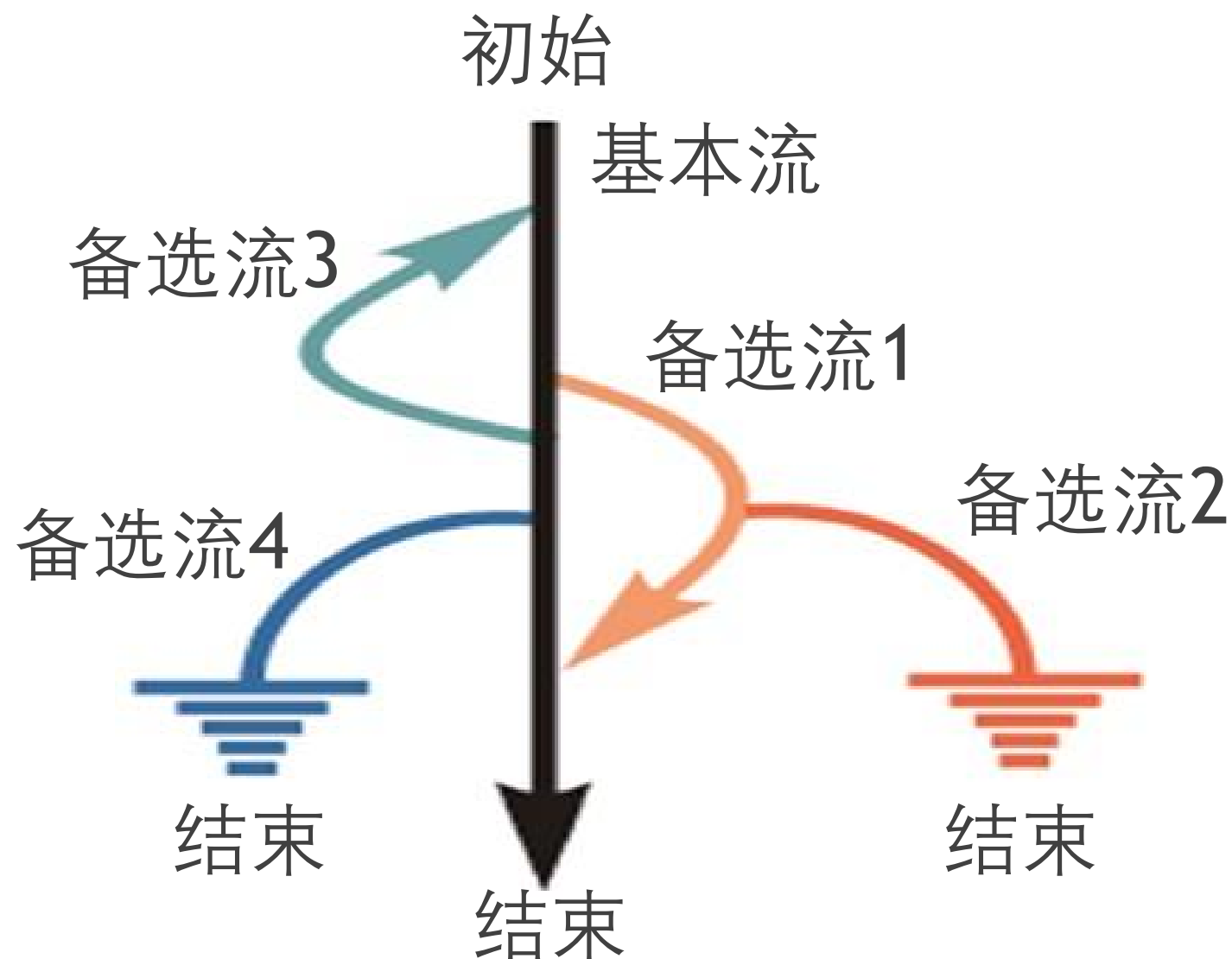
场景法

- 越来越多的软件系统采用事件触发来控制流程
- 事件触发时的情景形成场景
- 同一事件不同的触发顺序和处理结果形成事件流



场景法基本原理

场景法通过运用场景来对系统的功能点或业务流程的描述，从而提高测试效果的一种方法。

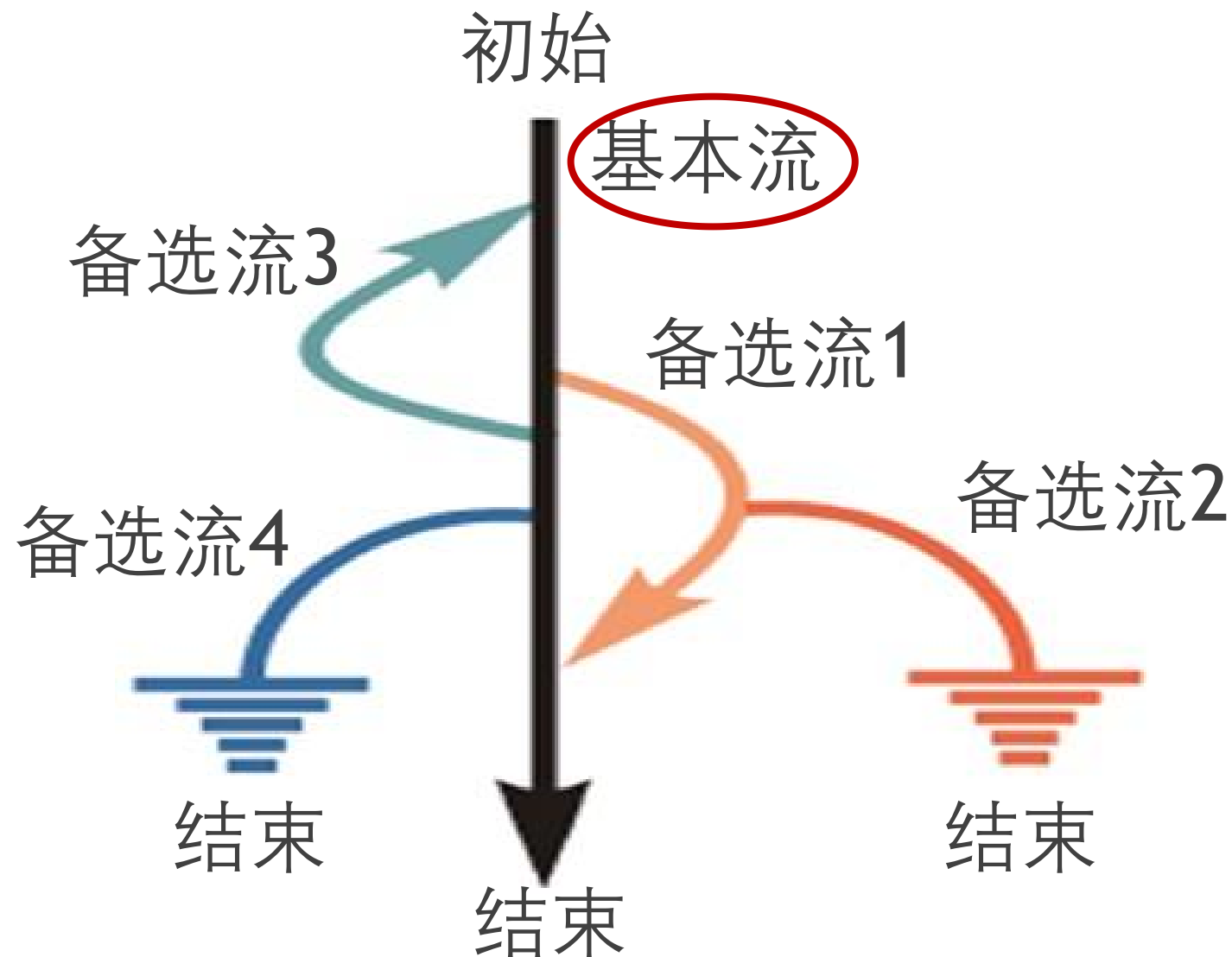


基本流和备选流

基本流: 系统从初始态到终止态的最主要的业务流程。测试中至少要确保系统基本流的执行是完全正确的。

高风险事件流

- 操作频率高
- 涉及业务规则复杂
- 涉及重要功能
- 涉及用户类型广泛
- 涉及用户数量大
- 涉及交互复杂

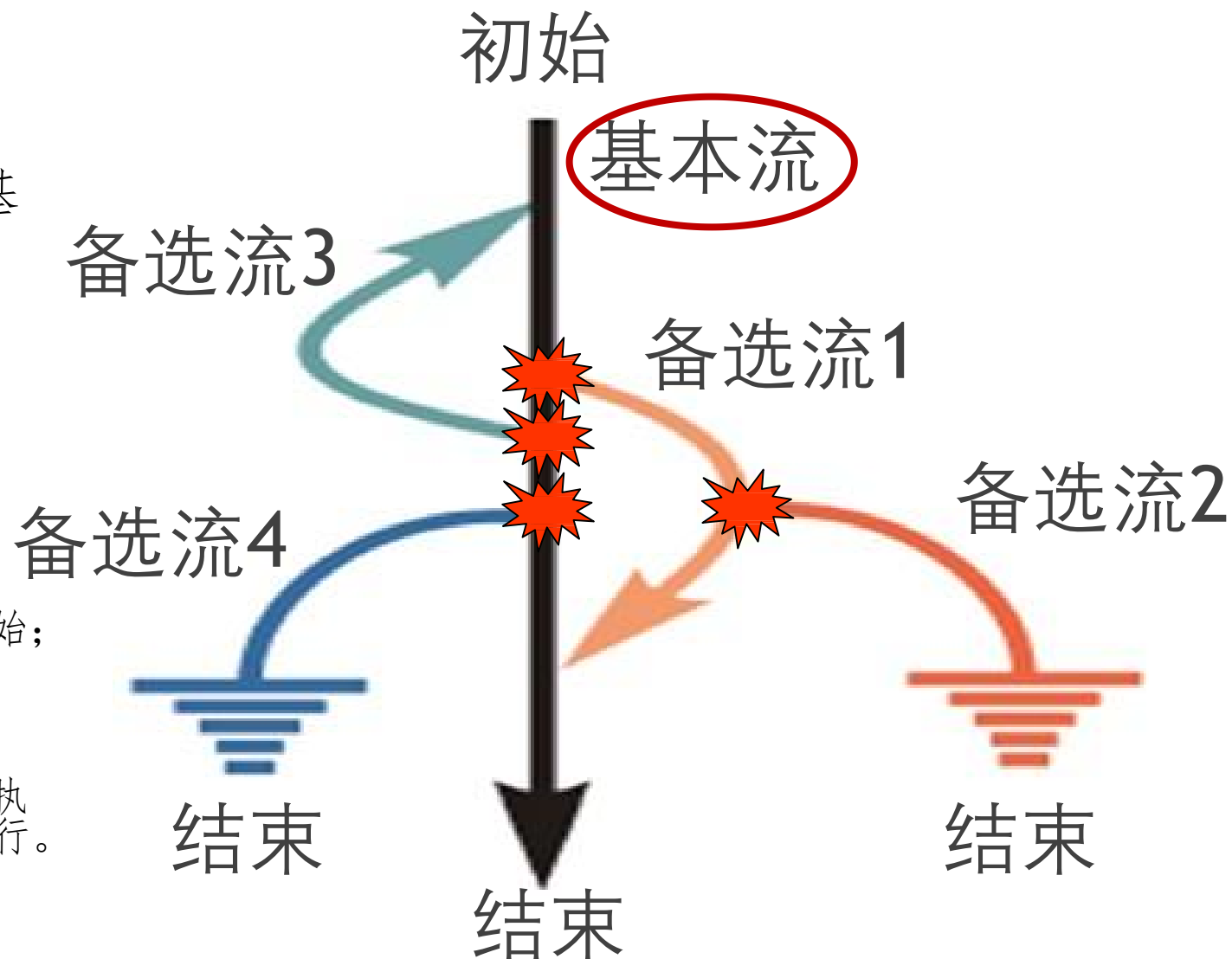


基本流和备选流

备选流: 备选事件流，以基本流为基础，在基本流所经过的每个判定节点处满足的不同触发条件而导致的其他事件流。

节点形式

- 起始节点从基本流的某个判定节点开始；
- 起始节点从其他备选流的某个判定节点开始；
- 终止节点是基本流上的某个状态；
- 终止节点是其他的系统终止状态。
- 备选流上的每个节点执行后可以继续往下执行，也可以返回基本流上的某个节点继续执行。



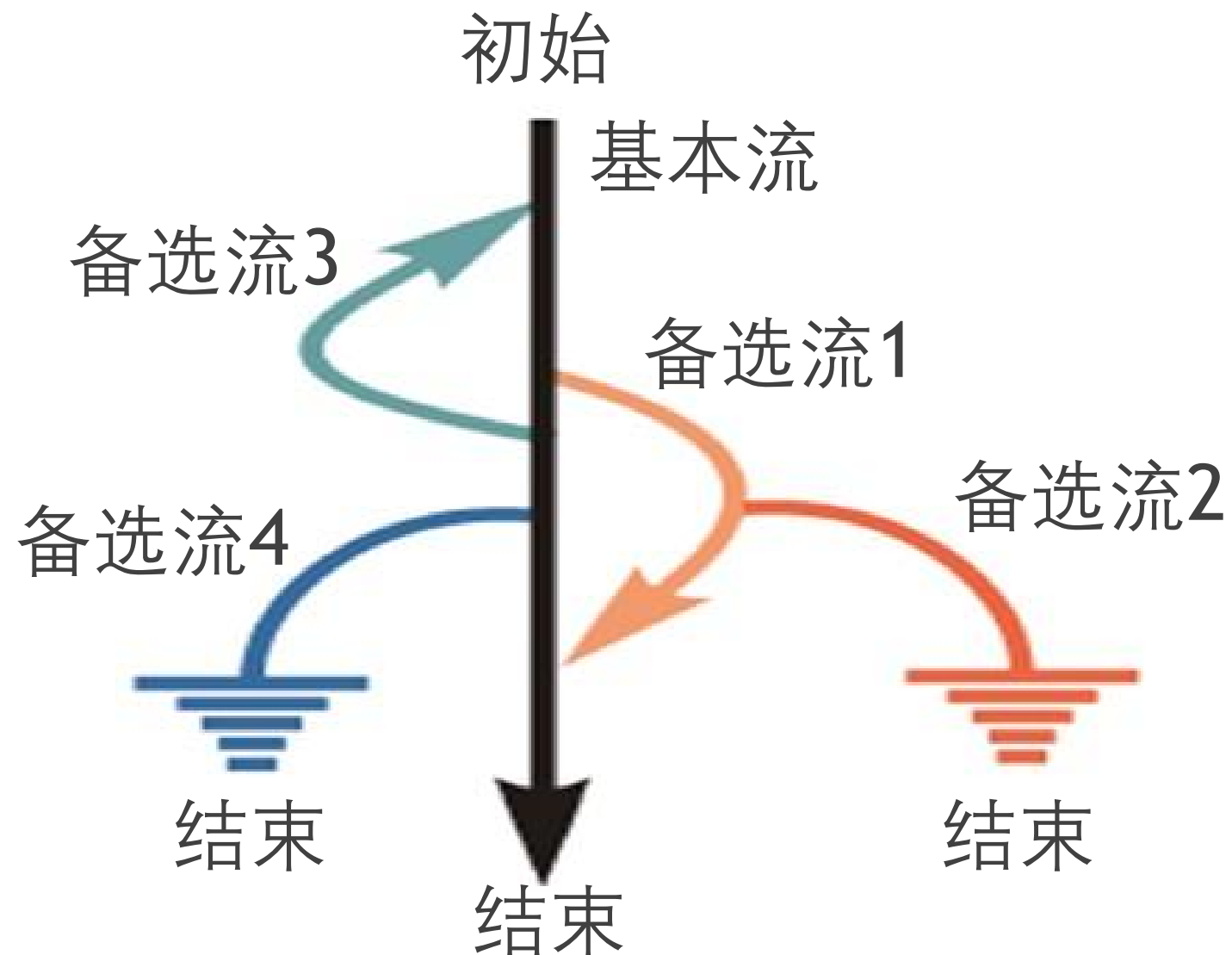


基本流与备选流的区别

	基本流	备选流
测试重要性	重要	次要
数目	1条	1条或多条
初始节点位置	系统初始状态	基本流或其他备选流
终止节点位置	系统默认终止状态	基本流或系统其他终止状态
是否是完整的 业务流程	是	否，仅为业务流程的执行片段

定义场景

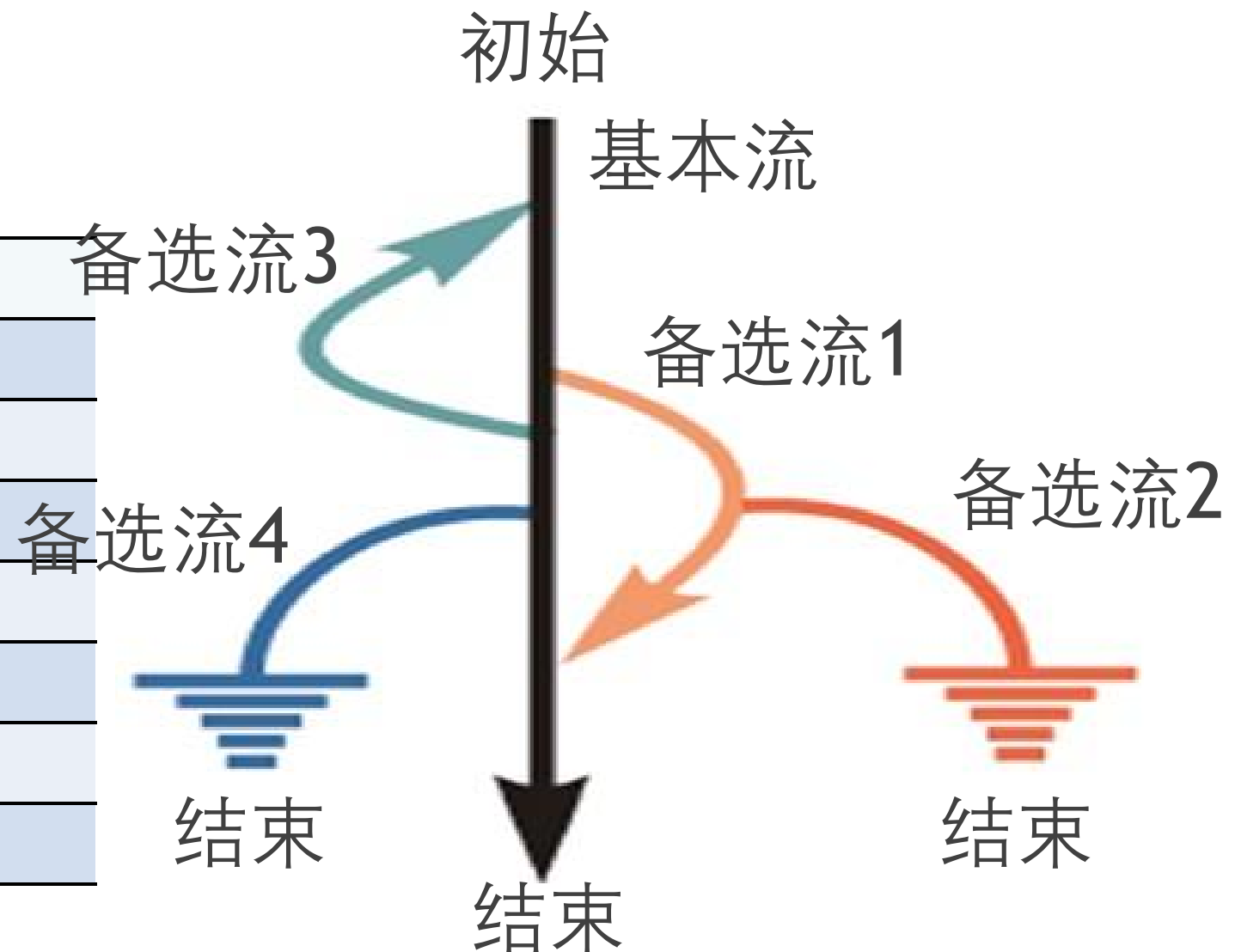
场景:可以看做是基本流与备选流的有序集合。一个场景中至少包含一条基本流。





定义场景

场景1	基本流
场景2	基本流、备选流1
场景3	基本流、备选流1、备选流2
场景4	基本流、备选流3
场景5	基本流、备选流3、备选流1
场景6	基本流、备选流3、备选流1、备选流2
场景7	基本流、备选流4
场景8	基本流、备选流3、备选流4





举例：ATM取款

用例描述	
基本流	成功提款
备选2	ATM内没有现金
备选3	ATM内现金不足
备选4	PIN有误（还有输入机会）
备选5	PIN有误（没有输入机会）
备选6	账户不存在 / 账户类型有误
备选7	账户余额不足



举例：ATM取款

场景设计

场景1：成功提款	基本流
场景2：ATM内没有现金	基本流、备选流2
场景3：ATM内现金不足	基本流、备选流3
场景4：PIN有误（还有输入机会）	基本流、备选流4
场景5：PIN有误（没有输入机会）	基本流、备选流5
场景6：账户不存在 / 账户类型有误	基本流、备选流6
场景7：账户余额不足	基本流、备选流7



举例：ATM取款

测试用例设计

序号	场景	PIN	账号	取款金额	账面金额	ATM 现金	预期结果
1	场景1：成功提款	V	V	V	V	V	成功提款
2	场景2：ATM里没有现金	V	V	V	V	X	提款选项不可用，用例结束
3	场景3：ATM里现金不足	V	V	V	V	X	警告信息，返回基本流相应步骤，重新输入金额
4	场景4：PIN有误 (还有不止一次机会)	X	V	n/a	V	V	警告信息，返回基本流相应步骤，重新输入PIN
5	场景4：PIN有误 (还有一次机会)	X	V	n/a	V	V	警告信息，返回基本流相应步骤，重新输入PIN
6	场景5：PIN有误 (不再有输入机会)	X	V	n/a	V	V	警告信息，卡被没收，用例结束
						