



北京交通大学
BEIJING JIAOTONG UNIVERSITY



第三章 动态测试技术

hhli@bjtu.edu.cn
2020年10月





第三章 动态测试技术

3.1

黑盒测试技术

3.2

白盒测试技术



3.1 黑盒测试技术

3.1.1 测试用例设计概述

3.1.2 等价类划分法

3.1.2 边界值分析法

3.1.2 因果图法

3.1.5 决策表法

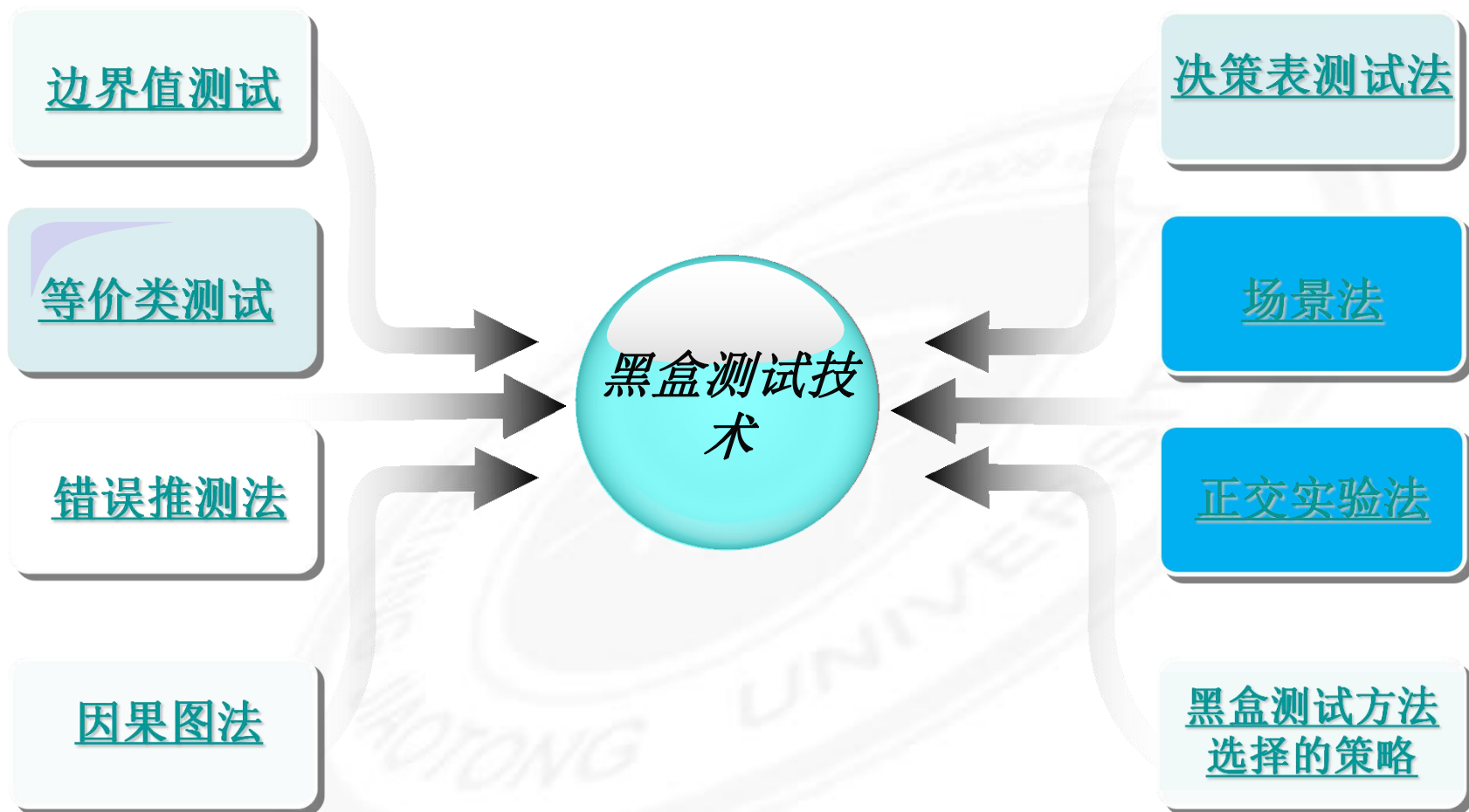
3.1.6 错误推测法

3.1.7 测试方法的选择

黑盒测试法的概念

- 黑盒测试被称为功能测试或数据驱动测试。在测试时，把被测程序视为一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下进行。
- 采用黑盒测试的目的主要是在已知软件产品所应具有的功能的基础上进行，包括：
 - (1) 检查程序功能能否按需求规格说明书的规定正常使用，测试各个功能是否有遗漏，检测性能等特性要求是否满足。
 - (2) 检测人机交互是否错误，检测数据结构或外部数据库访问是否错误，程序是否能适当地接收输入数据而产生正确的输出结果，并保持外部信息（如数据库或文件）的完整性。
 - (3) 检测程序初始化和终止方面的错误。

黑盒测试方法



3.1.1 测试用例设计概述

- 1 测试用例的定义和特征
- 2 测试用例的基本准则
- 3 设计测试用例的着眼点
- 4 测试用例设计书写标准

1、测试用例的定义和特征

- 测试用例的定义:

- (1) 测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果。
- (2) 测试用例是执行的最小实体。

- 测试用例的特征:

- (1) 最有可能抓住错误的;
- (2) 不是重复、多余的;
- (3) 一组相似测试用例中最有效的;
- (4) 既不是太简单, 也不是太复杂。

2、设计测试用例的基本准则

④ 测试用例的代表性

能够代表并覆盖各种合理的和不合理的、合法的和非法的、边界的和越界的以及极限的输入数据、操作和环境设置等。

④ 测试结果的可判定性

即测试执行结果的正确性是可判定的，每一个测试用例都应有相应的期望结果。

④ 测试结果的可再现性

即对同样的测试用例，系统的执行结果应当是相同的。

3、设计测试用例的出发点

- ① 根据产品规格说明，测试基本功能；
- ② 考虑设计一般用户（非专业人员）的使用方案；
- ③ 考虑设计稀有或特殊的使用方案；
- ④ 与系统其它组成部分的配合；
- ⑤ 考虑特殊情况（如内存和硬件的冲突等）；
- ⑥ 设计极端情况（如内存泄漏、破坏性测试等）；
- ⑦ 能花费最小的代价（人力、物力、财力、时间）做最好的测试。

4、测试用例设计书写标准

在ANSI/IEEE829-2008标准中列出了与测试设计相关的测试用例编写规范和模板（《GB/T 15532-2008 计算机软件测试规范》中也有相关规定）。标准模板中主要元素包括：

- ④ 标识符——唯一标识每一个测试用例。
- ④ 测试项——准确的描述所需要的测试项及其特征。
- ④ 测试环境要求——表征执行该测试用例需要的测试环境。
- ④ 输入标准——执行测试用例的输入需求（这些输入可能包括数据、文件或者操作）。
- ④ 输出标准——按照指定的环境和输入标准得到的期望输出结果。
- ④ 测试用例之间的关联——标识该测试用例与其它测试（或其它测试用例）之间的依赖关系

3.1.2 等价类划分法

1. 等价类划分法概述
2. 等价类划分法的设计原则
3. 等价类划分法的测试用例设计
4. 常见等价类划分法的测试形式
5. 使用等价类划分法测试的实例

1、等价类划分法概述

- ❖ 等价类划分法是一种常用的黑盒测试方法，它将不能穷举的测试过程进行合理分类，从而保证设计出来的测试用例具有完整性和代表性。
- ❖ 例：设计这样的测试用例，来实现一个对所有实数进行开平方根运算（ $y = \text{sqrt}(x)$ ）的程序的测试。
 - 由于开平方根运算只对非负实数有效，这时需要将所有的实数（输入域x）进行划分，可以分成：正实数、0 和 负实数。假设我们选定+1.4444代表正实数，-2.345代表负实数，则为该程序设计的测试用例的输入为+1.4444、0 和 -2.345。

1、等价类划分法（续）

- ❖ 等价类划分法是把所有可能的输入数据，即程序的输入域划分为若干子集（部分），然后从每一个子集中选取少数具有代表性的数据作为测试用例。
- ❖ 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，它们具有等价特性，即每一类的代表性数据在测试中的作用都等价于这一类中的其它数据。这样，对于表征该类的数据输入将能代表整个子集合的输入。因此，可以合理的假定：
 - 测试某等价类的代表值就是等效于对于这一类其它值的测试。

2、等价类的划分原则

- 等价类是输入域的某个子集合，而所有等价类的并集就是整个输入域。因此，等价类对于测试有两个重要的意义：
 - 完备性——整个输入域提供一种形式的完备性。
 - 无冗余性——若互不相交则可保证一种形式的无冗余性。
- 如何划分？——先从程序的规格说明书中找出各个输入条件，再为每个输入条件划分两个或多个等价类，形成若干的互不相交的子集。
- 采用等价类划分法设计测试用例通常分两步进行：
 - (1) 确定等价类，列出等价类表。
 - (2) 确定测试用例。

2、等价类的划分原则（续）

④ 划分等价类可分为两种情况：

(1) 有效等价类

——是指对软件规格说明而言，是有意义的、合理的输入数据所组成的集合。利用有效等价类，能够检验程序是否实现了规格说明中预先规定的功能和性能。

(2) 无效等价类

——是指对软件规格说明而言，是无意义的、不合理的输入数据所构成的集合。利用无效等价类，可以鉴别程序异常处理的情况，检查被测对象的功能和性能的实现是否有不符合规格说明要求的地方。



2、等价类的划分原则（续）

① 进行等价类划分的依据：

(1) **按照区间划分**：在输入条件规定了取值范围或值的个数的情况下，可以确定一个有效等价类和两个无效等价类。

例：程序输入条件为小于100大于10的整数 x ，则有效等价类为 $10 < x < 100$ ，两个无效等价类为 $x \leq 10$ 和 $x \geq 100$ 。

(2) **按照数值划分**：在规定了一组输入数据（假设包括 n 个 输入值），并且程序要对每一个输入值分别进行处理的情况下，可确定 n 个有效等价类（每个值确定一个有效等价类）和一个无效等价类（所有不允许的输入值的集合）。

例：程序输入 x 取值于一个固定的枚举类型 $\{1, 3, 7, 15\}$ ，且程序中对这4个数值分别进行了处理，则有效等价类为 $x=1$ 、 $x=3$ 、 $x=7$ 、 $x=15$ ，无效等价类为 $x \neq 1, 3, 7, 15$ 的值的集合。

等价类的划分原则（续）

(3) 按照数值集合划分：在输入条件规定了输入值的集合或规定了“必须如何”的条件下，可以确定一个有效等价类和一个无效等价类（该集合有效值之外）。

例：程序输入条件为取值为奇数的整数 x ，则有效等价类为 x 的值为奇数的整数，无效等价类为 x 的值不为奇数的整数。

(4) 按照限制条件或规则划分：在规定了输入数据必须遵守的规则或限制条件的情况下，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）。

例：程序输入条件为以字符‘a’开头、长度为8的字符串，并且字符串不包含‘a’～‘z’之外的其它字符，则有效等价类为满足上述所有条件的字符串，无效等价类为不以‘a’开头的字符串、长度不为8的字符串和包含了‘a’～‘z’之外其它字符的字符串。

(5) 细分等价类：在**确知**已划分的等价类中各元素在程序中的处理方式不同的情况下，则应再将该等价类进一步划分为更小的等价类，并建立等价类表。



3 等价类划分法的测试用例设计

- 在设计测试用例时，应同时考虑有效等价类和无效等价类测试用例的设计。
- 根据已列出的等价类表可确定测试用例，具体设计过程如下：
 - (1) 首先为等价类表中的每一个等价类分别规定一个**唯一的编号**。
 - (2) 设计一个新的测试用例，使它能够**尽量覆盖尚未覆盖的有效等价类**。重复这个步骤，直到所有有效等价类均被测试用例所覆盖。
 - (3) 设计一个新的测试用例，使它仅覆盖一个**尚未覆盖的无效等价类**。重复这一步骤，直到所有无效等价类均被测试用例所覆盖。



4 常见等价类划分测试形式

- 针对是否对无效数据进行测试，可以将等价类测试分为标准等价类测试和健壮等价类测试。
- 标准等价类测试——不考虑无效数据值，测试用例使用每个等价类中的一个值。
- 健壮等价类测试——主要的出发点是考虑了无效等价类。对有效输入，测试用例从每个有效等价类中取一个值；对无效输入，一个测试用例有一个无效值，其它值均取有效值。

健壮等价类测试存在两个问题：

- (1) 需要花费精力定义无效测试用例的期望输出
- (2) 对强类型的语言可能没有必要考虑无效的输入

强类型定义语言

- ④ 它是一种总是强制类型定义的语言，要求变量的使用要严格符合定义，所有变量都必须先定义后才能使用。如，**Java、C++等都是强制类型定义的。**
- ④ 也即，一旦一个变量被指定了某个数据类型，如果不经强制转换，那么它就永远是这个数据类型了。
- ④ 例如你有一个整数，如果不显式地进行转换，你不能将其视为一个字符串。
- ④ 与其相对应的是弱类型语言：数据类型可以被忽略的语言。它与强类型定义语言相反，一个变量可以赋不同数据类型的值。



5、等价类划分法测试实例

- 实例1 三角形问题
- 输入三个整数a、b、c，分别作为三角形的三条边，现通过程序判断由三条边构成的三角形的类型为等边三角形、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2 $1 \leq b \leq 100$

条件3 $1 \leq c \leq 100$

条件4 $a < b + c$

条件5 $b < a + c$

条件6 $c < a + b$



实例（续）

- **分析：**
- 如果输入值 a 、 b 、 c 不满足条件1、条件2和条件3，程序给出“边的取值超出允许范围”的信息。
- 如果输入值 a 、 b 、 c 满足条件1、条件2和条件3，则输出下列四种情况之一：
 - (1) 如果不满足条件4、条件5和条件6中的一个，则程序输出为“非三角形”。
 - (2) 如果三条边相等，则程序输出为“等边三角形”。
 - (3) 如果恰好有两条边相等，则程序输出为“等腰三角形”。
 - (4) 如果三条边都不相等，则程序输出为“一般三角形”。
- **结论：** 三角形问题的复杂之处在于输入与输出之间的关系比较复杂。



实例（续）

分析：在多数情况下，是从输入域划分等价类的，但并非不能从被测程序的输出域反过来定义等价类，事实上，这对于三角形问题却是最简单的划分方法。在三角形问题中，有四种可能的输出：等边三角形、等腰三角形、一般三角形和非三角形。利用这些信息能够确定下列输出（值域）等价类。

$R1 = \{ \langle a, b, c \rangle : \text{边为 } a, b, c \text{ 的等边三角形} \}$

$R2 = \{ \langle a, b, c \rangle : \text{边为 } a, b, c \text{ 的等腰三角形} \}$

$R3 = \{ \langle a, b, c \rangle : \text{边为 } a, b, c \text{ 的一般三角形} \}$

$R4 = \{ \langle a, b, c \rangle : \text{边为 } a, b, c \text{ 不能组成三角形} \}$

三角形问题

测试用例	a	b	c	预期输出
Test1	10	10	10	等边三角形
Test2	10	10	5	等腰三角形
Test3	3	4	5	一般三角形
Test4	4	1	2	非三角形

三角形问题的4个标准等价类测试用例



三角形问题。

测试用例	a	b	c	预期输出
Test1	5	6	7	一般三角形
Test2	-1	5	5	a值超出输入值定义域
Test3	5	-1	5	b值超出输入值定义域
Test4	5	5	-1	c值超出输入值定义域
Test5	101	5	5	a值超出输入值定义域
Test6	5	101	5	b值超出输入值定义域
Test7	5	5	101	c值超出输入值定义域

三角形问题的7个健壮等价类测试用例

3.1.3 边界值分析法

- 1 边界值分析法概要
- 2 边界值分析法测试用例
- 3 边界值分析法测试举例



1、边界值分析法概要

- 边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法。通常边界值分析法是作为对等价类划分法的补充，这种情况下，其测试用例来自等价类的边界。
- 使用边界值分析法意义：

无数的测试实践表明，大量的故障往往发生在输入定义域或输出值域的边界上，而不是在其内部。因此，针对各种边界情况设计测试用例，通常会取得很好的测试效果。



1、边界值分析法概要

④ 怎样用边界值分析法设计测试用例？

- (1) 首先确定边界情况。通常输入或输出等价类的边界就是应着重测试的边界。
- (2) 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。

举例 —— 常见的边界值

- 对16-bit 的整数而言 32767 和 -32768 是边界
- 屏幕上光标在最左上、最右下位置
- 报表的第一行和最后一行
- 数组元素的第一个和最后一个
- 循环的第 0 次、第 1 次和倒数第 2 次、最后一次
-



边界值分析

- ④ **边界值分析使用**与等价类划分法相同，只是边界值分析假定错误更多地存在于划分的边界上，因此，在等价类的边界上以及两侧的情况设计测试用例。
- ④ 例：测试计算平方根的函数
 - 输入：实数
 - 输出：实数
 - 规格说明：当输入一个0或比0大的数的时候，返回其正平方根；当输入一个小于0的数时，显示错误信息“平方根非法-输入值小于0”并返回0；库函数Print-Line可以用来输出错误信息。



边界值分析

④ 等价类划分分析:

- 可以考虑作出如下划分:
 - **输入** $(1) < 0$ 和 $(2) \geq 0$
 - **输出** $(a) \geq 0$ 和 (b) Error
- 测试用例有两个:
 - **输入** 4, 输出 2。对应于 (2) 和 (a) 。
 - **输入** -10, 输出 0 和 错误提示。对应于 (1) 和 (b) 。

④ 边界值分析:

- 等价类 (2) 的边界为 0 和 最大正实数; 等价类 (1) 的边界为 最小负实数和 0。由此得到以下测试用例:
 - 输入 {最小负实数}
 - 输入 {绝对值很小的负数}
 - 输入 0
 - 输入 {绝对值很小的正数}
 - 输入 {最大正实数}



举例 —— 利用边界值作为测试数据

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个到 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	假设某软件的数据输入域要求输入 5 位的数据值，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 数值来作为边界条件。
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。



内部边界值分析

- 在多数情况下，边界值条件是基于应用程序的功能设计而需要考虑的因素，可以从软件的规格说明或常识中得到，也是最终用户可以很容易发现问题的。然而，在测试用例设计过程中，某些边界值条件是不需要呈现给用户的，或者说用户是很难注意到的，但同时确实属于检验范畴内的边界条件，称为内部边界值条件或子边界值条件。

- 内部边界值条件主要有下面几种：

- 数值的边界值检验



- 字符的边界值检验



- 其它边界值检验

■ 小结：

在实际的测试用例设计中，需要将基本的软件设计要求和程序定义的要求结合起来，即结合基本边界值条件和内部边界值条件来设计有效的测试用例。



数值的边界值检验

- 计算机是基于二进制进行工作的，因此，软件的任何数值运算都有一定的范围限制。

计算机数值运算的范围

项	范围或值
位 (bit)	0 或 1
字节 (byte)	0 ~ 255
字 (word)	0~65535 (单字) 或 0~4294967295 (双字)
千 (K)	1024
兆 (M)	1048576
吉 (G)	1073741824



字符的边界值检验

- 在计算机软件中，字符也是很重要的表示元素，其中ASCII和Unicode是常见的编码方式。下表中列出了一些常用字符对应的ASCII码值。

字符	ASCII码值	字符	ASCII码值
空 (null)	0	A	65
空格 (space)	32	a	97
斜杠 (/)	47	Z	90
0	48	z	122
冒号 (:)	58	单引号 (`)	96
@	64		



选择测试用例的原则

- (1) 如果**输入条件**规定了值的范围，则应取刚达到这个范围的边界值以及刚刚超过这个范围边界的值作为测试输入数据。
- (2) 如果**输入条件**规定了值的个数，则用最大个数、最小个数和比最大个数多1个、比最小个数少1个的数作为测试数据。
- (3) 根据程序规格说明的每个**输出条件**，使用原则（1）。
- (4) 根据程序规格说明的每个**输出条件**，使用原则（2）。
- (5) 如果程序的规格说明给出的输入域或输出域是有序集合（如有序表、顺序文件等），则应选取集合中的第一个和最后一个元素作为测试用例。
- (6) 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构的边界上的值作为测试用例。
- (7) 分析程序规格说明，找出其它可能的边界条件。

2、边界值分析法测试用例设计

- 采用边界值分析测试的基本思想是：故障往往出现在输入变量的边界值附近。

——因此，边界值分析法利用输入变量的最小值(min)、略大于最小值(min+)、输入值域内的任意值(nom)、略小于最大值(max-)和最大值(max)来设计测试用例。

- 边界值分析法是基于可靠性理论中称为“单故障”的假设，即有两个或两个以上故障同时出现而导致软件失效的情况很少，也就是说，软件失效基本上是由单故障引起的。

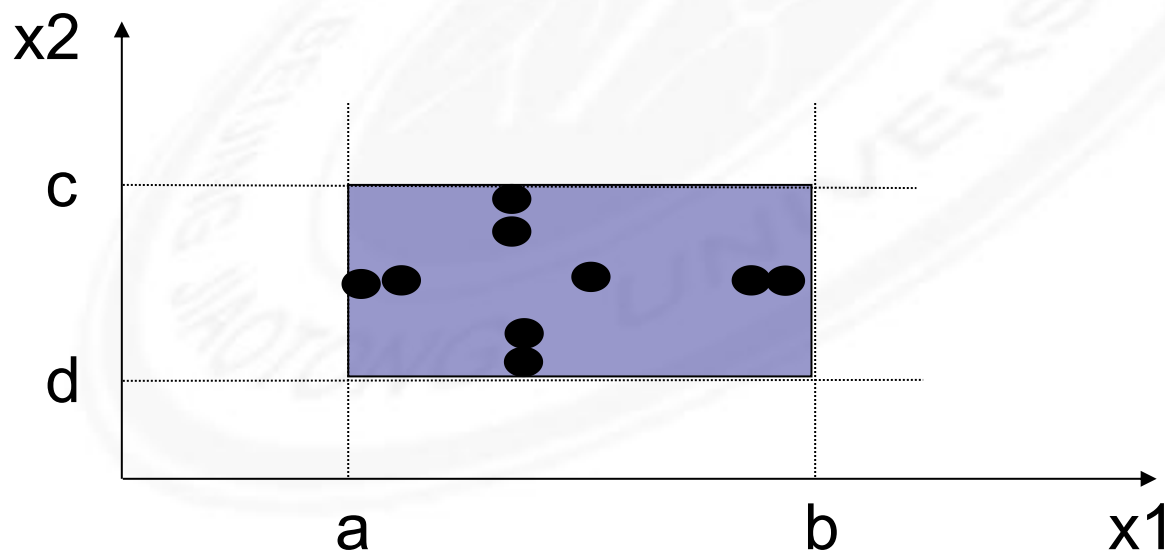
——因此，在边界值分析法中获取测试用例的方法是：

- (1) 每次保留程序中一个变量，让其余的变量取正常值，被保留的变量依次取min、min+、nom、max-和max。
- (2) 对程序中的每个变量重复 (1) 。

2. 边界值分析法测试用例设计 (续)

- ① 例1: 有两个输入变量 x_1 ($a \leq x_1 \leq b$) 和 x_2 ($c \leq x_2 \leq d$) 的程序F的边界值分析测试用例如下:

{ $\langle x_{1nom}, x_{2min} \rangle$, $\langle x_{1nom}, x_{2min+} \rangle$, $\langle x_{1nom}, x_{2nom} \rangle$,
 $\langle x_{1nom}, x_{2max} \rangle$, $\langle x_{1nom}, x_{2max-} \rangle$, $\langle x_{1min}, x_{2nom} \rangle$,
 $\langle x_{1min+}, x_{2nom} \rangle$, $\langle x_{1max}, x_{2nom} \rangle$, $\langle x_{1max-}, x_{2nom} \rangle$ }



2、边界值分析法测试用例（续）

- 例2：有二元函数 $f(x, y)$ ，其中 $x \in [1, 12]$ ， $y \in [1, 31]$ 。
则采用边界值分析法设计的测试用例是：

{ $\langle 1, 15 \rangle$, $\langle 2, 15 \rangle$, $\langle 11, 15 \rangle$, $\langle 12, 15 \rangle$, $\langle 6, 15 \rangle$, $\langle 6, 1 \rangle$,
 $\langle 6, 2 \rangle$, $\langle 6, 30 \rangle$, $\langle 6, 31 \rangle$ }

- 推论：对于一个含有 n 个变量的程序，采用边界值分析法测试程序会产生 $4n+1$ 个测试用例。

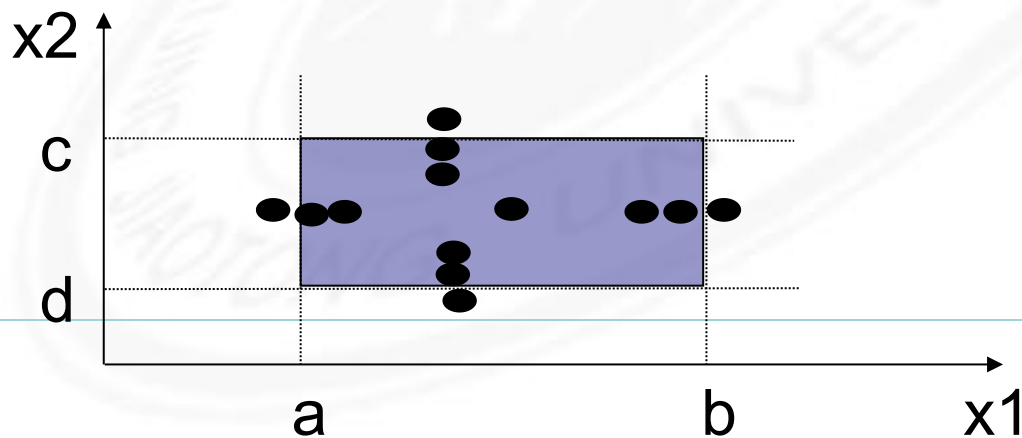
- 练习1：有函数 $f(x, y, x)$ ，其中 $x \in [1900, 2100]$ ， $y \in [1, 12]$ ， $z \in [1, 31]$ 的。请写出该函数采用边界值分析法设计的测试用例。

答：{ $\langle 2000, 6, 1 \rangle$, $\langle 2000, 6, 2 \rangle$, $\langle 2000, 6, 30 \rangle$, $\langle 2000, 6, 31 \rangle$,
 $\langle 2000, 1, 15 \rangle$, $\langle 2000, 2, 15 \rangle$, $\langle 2000, 11, 15 \rangle$, $\langle 2000, 12, 15 \rangle$,
 $\langle 1900, 6, 15 \rangle$, $\langle 1901, 6, 15 \rangle$, $\langle 2099, 6, 15 \rangle$, $\langle 2100, 6, 15 \rangle$,
 $\langle 2000, 6, 15 \rangle$ }



健壮性测试

- 健壮性测试是作为边界值分析的一个简单的扩充，它除了对变量的5个边界值分析取值外，还需要增加一个略大于最大值(max+)以及略小于最小值(min-)的取值，检查超过极限值时系统的情况。因此，对于有n个变量的函数采用健壮性测试需要 $6n+1$ 个测试用例。
- 前面例1中的程序F的健壮性测试如下图所示：





3、边界值分析法测试举例*

❖ 实例1 三角形问题的边界值分析测试用例

在三角形问题描述中，除了要求边长是整数外，没有给出其它的限制条件。在此，我们将三角形每边边长的取值范围值设置为 $[1, 100]$ 。

说明：如果程序规格说明中没有显式地给出边界值，则可以在设计测试用例前先设定取值的下限值和上限值。

测试用例



边界值分析法测试举例（续）。

测试用例	a	b	c	预期输出
Test 1	60	60	1	等腰三角形
Test2	60	60	2	等腰三角形
Test3	60	60	60	等边三角形
Test4	50	50	99	等腰三角形
Test5	50	50	100	非三角形
Test6	60	1	60	等腰三角形
Test7	60	2	60	等腰三角形
Test8	50	99	50	等腰三角形
Test9	50	100	50	非三角形
Test10	1	60	60	等腰三角形
Test11	2	60	60	等腰三角形
Test12	99	50	50	等腰三角形
Test13	100	50	50	非三角形

3.1.4 因果图法

- 1 因果图法的概要
- 2 因果图
- 3 因果图法测试举例



1、因果图法的概要

● 背景：

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）。

1、因果图法的简介（续）

- ④ 因果图法的思想：一些程序的功能可以用判定表（或称决策表）的形式来表示，并根据输入条件的组合情况规定相应的操作。
- ④ 因果图法的定义：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。
- ④ 采用因果图法设计测试用例的思路：
 - （1）根据程序规格说明书描述，分析并确定因（输入条件）和果（输出结果或程序状态的改变），画出因果图。
 - （2）将得到的因果图转换为判定表。
 - （3）为判定表设计测试用例。

因果图法的简介（续）

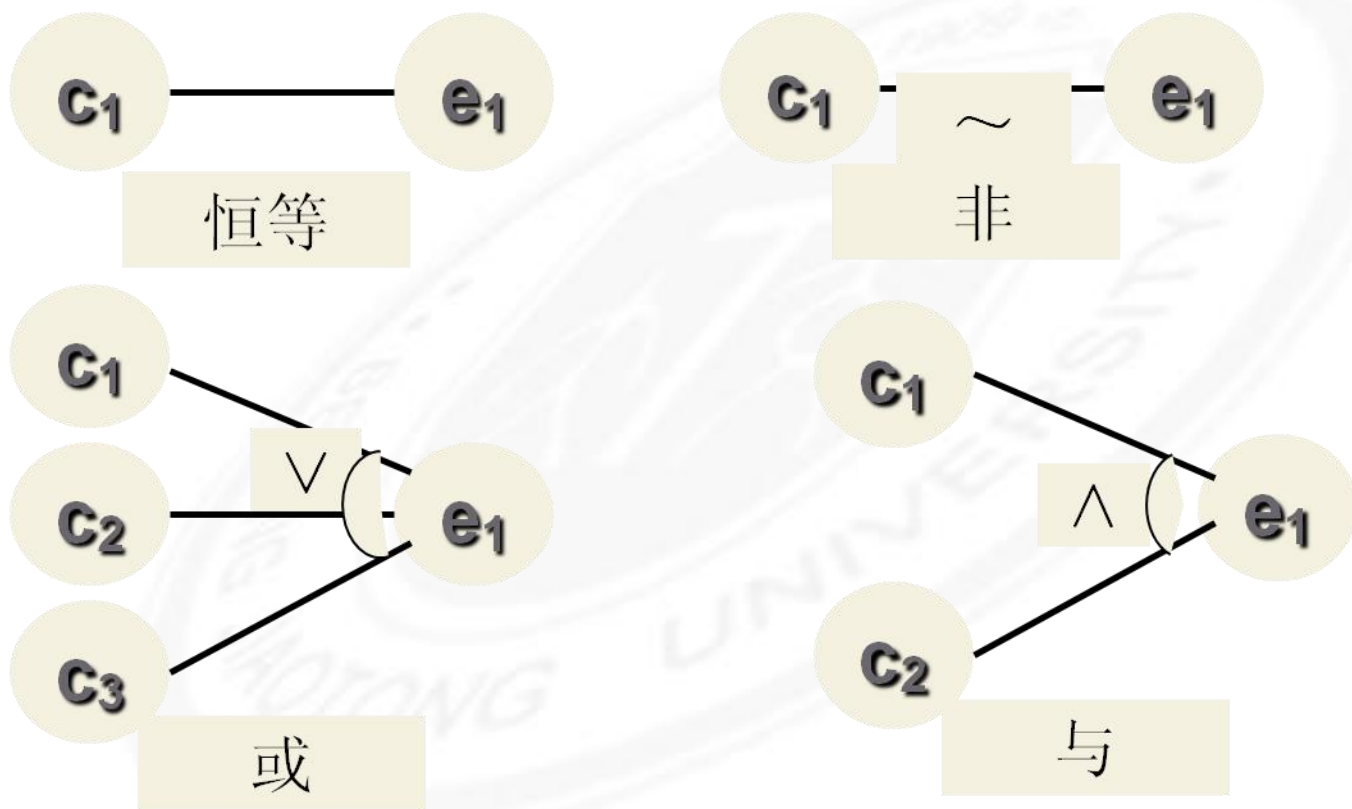
④ 使用因果图法的优点：

- （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、因果图

- 因果图中用来表示4种因果关系的基本符号：





因果图（续）



因果图中的4种基本关系

在因果图的基本符号中，图中的左结点 c_i 表示输入状态（或称原因），右结点 e_i 表示输出状态（或称结果）。 c_i 与 e_i 取值0或1，0表示某状态不出现，1则表示某状态出现。

- **恒等**：若 c_1 是1，则 e_1 也为1，否则 e_1 为0。
- **非**：若 c_1 是1，则 e_1 为0，否则 e_1 为1。
- **或**：若 c_1 或 c_2 或 c_3 是1，则 e_1 为1，否则 e_1 为0。
- **与**：若 c_1 和 c_2 都是1，则 e_1 为1，否则 e_1 为0。



因果图（续）

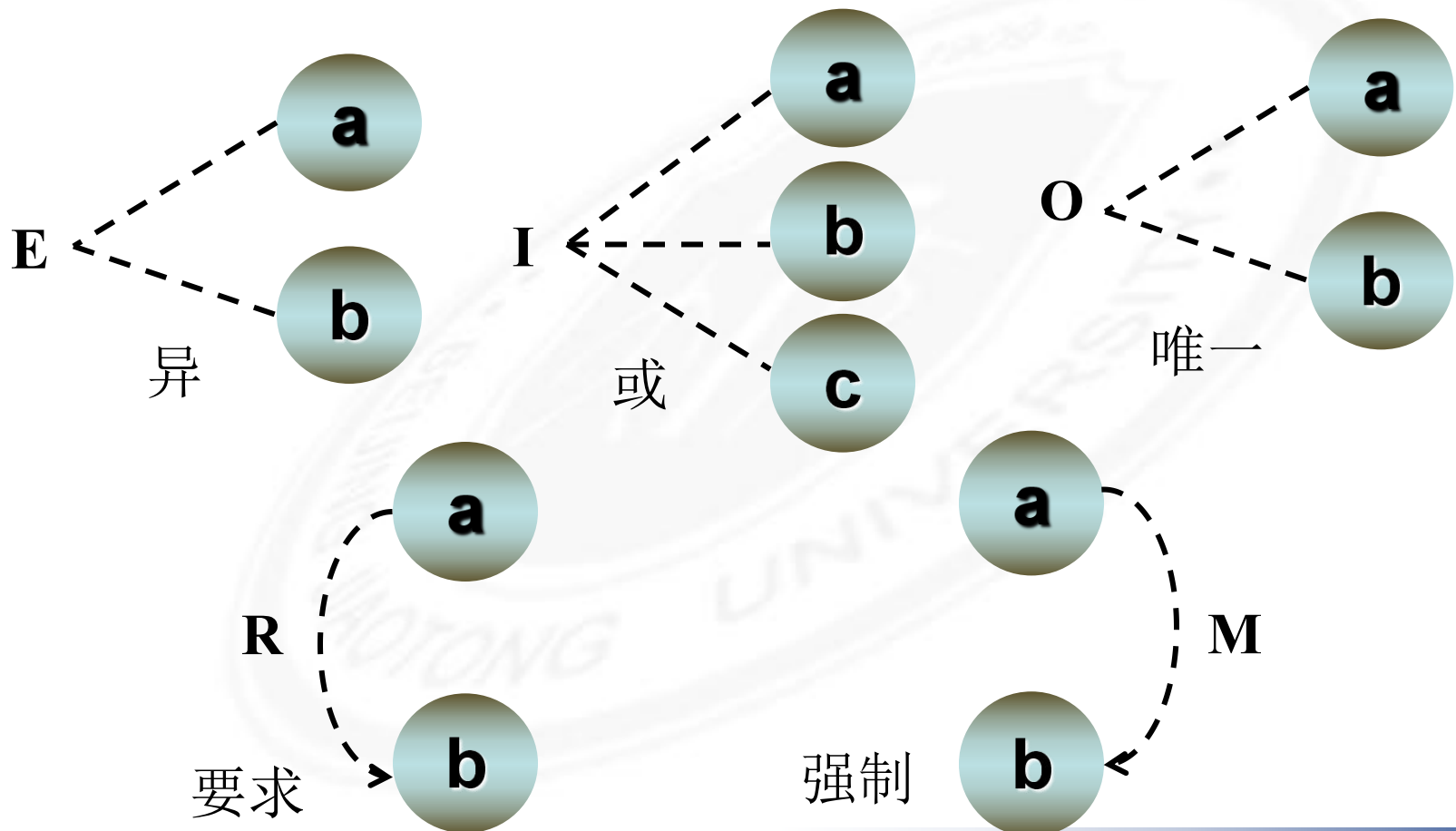
因果图中的约束

在实际问题中输入状态相互之间、输出状态相互之间可能存在某些依赖关系，称为“约束”。对于输入条件的约束有E、I、O、R四种约束，对于输出条件的约束只有M约束。

- E约束(异)：a和b中最多有一个可能为1，即a和b不能同时为1。
- 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。
- O约束(唯一)：a和b必须有一个且仅有一个为1。
- R约束(要求)：a是1时，b必须是1，即a为1时，b不能为0。
- M约束(强制)：若结果a为1，则结果b强制为0。

因果图 (续)

- 因果图中用来表示约束关系的约束符号:





因果图（续）

- ① 因果图法最终生成的是决策表。利用因果图生成测试用例的基本步骤如下：
 - (1) 分析软件规格说明中哪些是原因（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) 分析软件规格说明中的语义，找出原因与结果之间、原因与原因之间对应的关系，根据这些关系画出因果图。
 - (3) 由于语法或环境的限制，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表明约束或限制条件。
 - (4) 把因果图转换为决策表。
 - (5) 根据决策表中的每一列设计一个测试用例。



3、因果图法测试举例

❖ 实例 采用因果图法测试以下程序。

程序的规格说明要求：①输入的第一个字符必须是#或*，第二个字符必须是一个数字，此情况下进行文件的修改；②如果第一个字符不是#或*，则给出信息N，如果第二个字符不是数字，则给出信息M。

➤ 解题分析：

- (1) 分析程序的规格说明，列出原因和结果。
- (2) 找出原因与结果之间的因果关系、原因与原因之间的约束关系，画出因果图。
- (3) 将因果图转换成决策表。
- (4) 根据(3)中的决策表，设计测试用例的输入数据和预期输出。



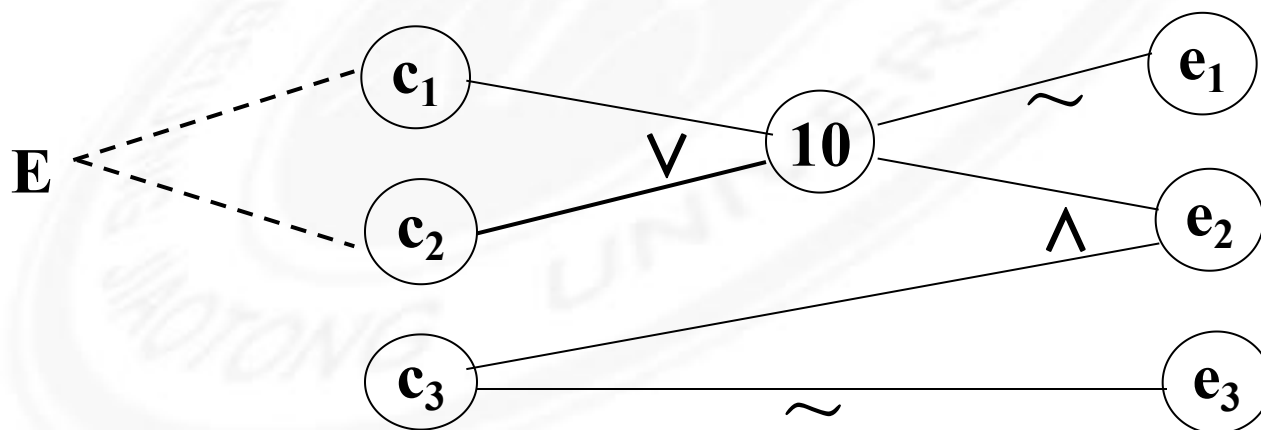
因果图法测试举例（续）

（1）分析程序规格说明中的原因和结果。

原因
c1: 第一个字符是#
c2: 第一个字符是*
c3: 第二个字符是一个数字

结果
e1: 给出信息N
e2: 修改文件
e3: 给出信息M

（2）画出因果图（编号为10的中间结点是导出结果的进一步原因）。





因果图法测试举例（续）

（3）将因果图转换成如下所示的决策表：

选项 \ 规则	1	2	3	4	5	6	7	8
条件:								
C1	1	1	1	1	0	0	0	0
C2	1	1	0	0	1	1	0	0
C3	1	0	1	0	1	0	1	0
10			1	1	1	1	0	0
动作:								
e1							√	√
e2			√		√			
e3				√		√		√
不可能	√	√						
测试用例			#3	#A	*6	*B	A1	GT

因果图法测试举例（续）

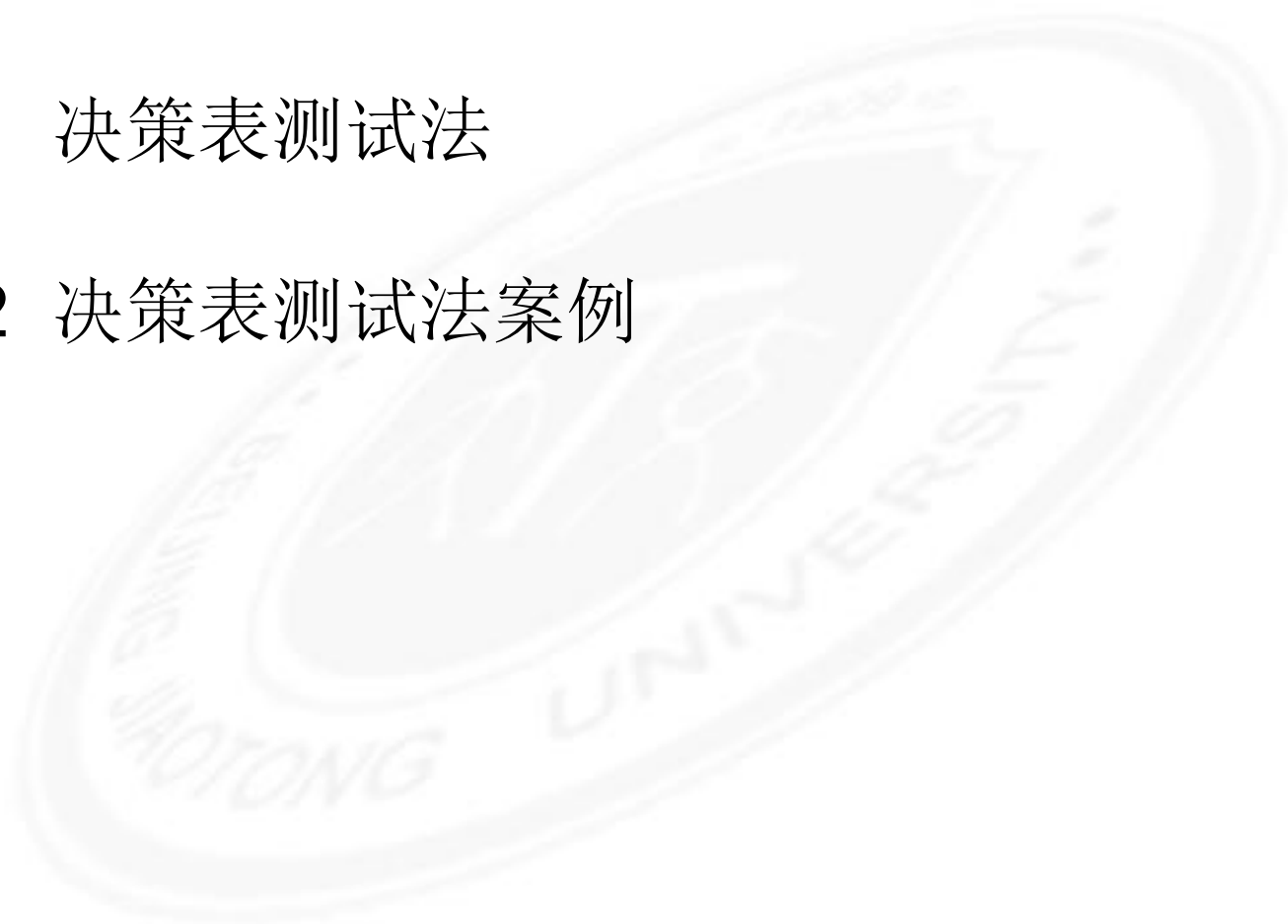
（4）根据决策表中的每一列设计测试用例：

测试用例 编号	输入数据	预期输出
1	#3	修改文件
2	#A	给出信息M
3	*6	修改文件
4	*B	给出信息M
5	A1	给出信息N
6	GT	给出信息N和信息 M



3.1.5 决策表法

- 1 决策表测试法
- 2 决策表测试法案例



1 决策表测试法

- 前面因果图方法中已经用到了判定表。判定表（Decision Table）主要用于分析和表达多逻辑条件下执行不同操作。
- 在程序设计发展的初期，判定表就已被当作编写程序的辅助工具了，因为它可以把复杂的逻辑关系和多种条件组合的情况表达得既具体又明确。



1、决策表测试法

- ❶ 在所有的黑盒测试方法中，基于决策表（也称判定表）的测试是最为严格、最具有逻辑性的测试方法。
- ❷ **决策表的定义：决策表是分析和表达多逻辑条件下执行不同操作的情况的工具。**
- ❸ 决策表的优点：能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用决策表能够设计出完整的测试用例集合。
- ❹ 在一些数据处理问题当中，某些操作的实施依赖于多个逻辑条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。决策表很适合于处理这类问题。



判定表测试方法

下表是一张关于科技书阅读指南的判定驱动表，包含3个问题8种情况。

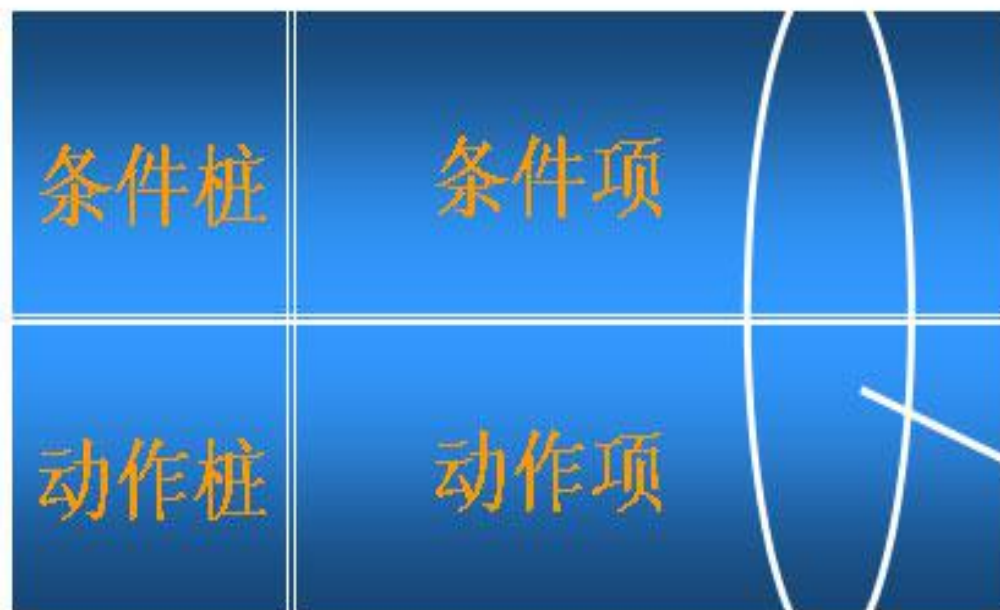
		1	2	3	4	5	6	7	8
问题	你觉得疲倦吗?	Y	Y	Y	Y	N	N	N	N
	你对内容感兴趣吗?	Y	Y	N	N	Y	Y	N	N
	书中内容使你糊涂吗?	Y	N	Y	N	Y	N	Y	N
建议	请回到本章开头重读	x				x			
	继续读下去		x				x		
	跳到下一章去读							x	x
	停止阅读, 请休息			x	x				

“读书指南”判定表

判定表组成

④ 判定表通常由四个部分组成：

- 条件桩
- 动作桩
- 条件项
- 动作项



规则

决策表的组成

- ❶ **决策表**通常由四个部分组成：
 - 条件桩（Condition Stub）：列出了问题的所有条件，通常认为列出的条件的**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。

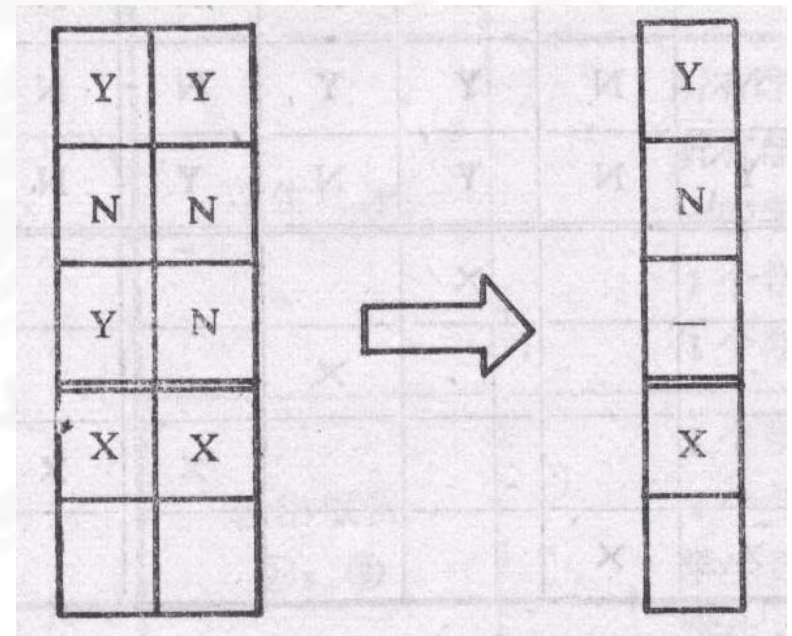
决策表的组成

- ❶ **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称为规则。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。
- ❷ **化简:** 是指合并具有相同的动作的两条或多条规则, 并且其条件项之间存在着极为相似的关系。

规则及规则合并举例

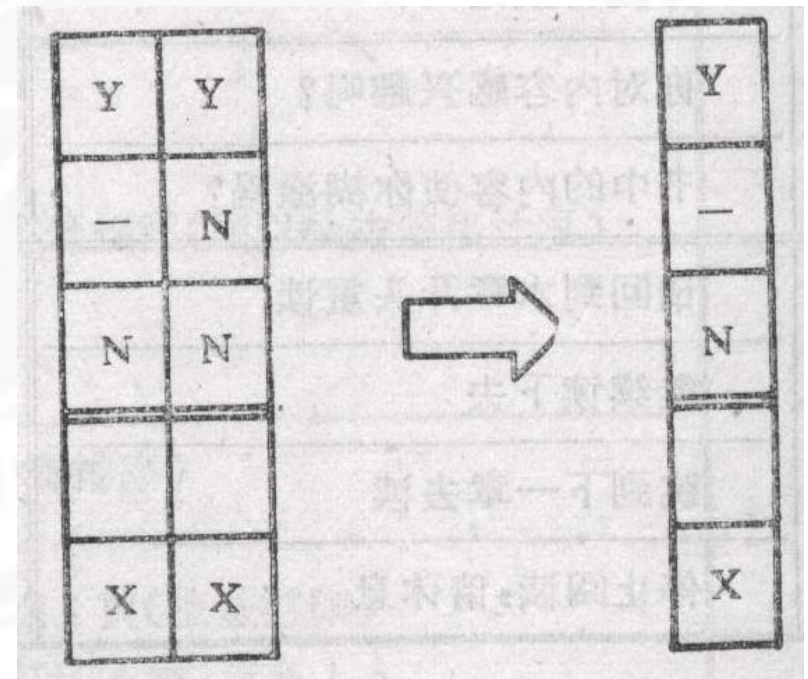
如右图左端，两规则动作项一样，条件项类似，在1、2条件项分别取Y、N时，无论条件3取何值，都执行同一操作。即要执行的动作与条件3无关。于是可合并。

“—” 表示与取值无关。



规则及规则合并举例

例：与上图类似，右图中，无关条件项“-”可包含其它条件项取值，具有相同动作的规则可合并。





		1	2	3	4	5	6	7	8
问题	你觉得疲倦吗?	Y	Y	Y	Y	N	N	N	N
	你对内容感兴趣吗?	Y	Y	N	N	Y	Y	N	N
	书中内容使你糊涂吗?	Y	N	Y	N	Y	N	Y	N
建议	请回到本章开头重读	x				x			
	继续读下去		x				x		
	跳到下一章去读							x	x
	停止阅读, 请休息			x	x				

”读书指南”判定表

		1	2	3	4
问题	你觉得疲倦吗?	-	-	Y	N
	你对内容感兴趣吗?	Y	Y	N	N
	书中内容使你糊涂吗?	Y	N	-	-
建议	请回到本章开头重读	x			
	继续读下去		X		
	跳到下一章去读				x
	停止阅读, 请休息			x	

化减后的”读书指南”判定表



决策表的生成

① 构造决策表生成的5个步骤：

(1) 确定规则的个数。

➤ 有 n 个条件的决策表有 2^n 个规则（每个条件取真、假值）。

(2) 列出所有的条件桩和动作桩。

(3) 填入条件项。

(4) 填入动作项，得到初始决策表。

(5) 简化决策表，合并相似规则。

➤ 若表中有两条以上规则具有相同的动作，并且在条件项之间存在极为相似的关系，便可以合并。

➤ 合并后的条件项用符号“-”表示，说明执行的动作与该条件的取值无关，称为无关条件。

建立决策表举例*

例：问题要求：“……对功率大于50马力的机器、维修记录不全或已运行10年以上的机器，应给予优先的维修处理……”。这里假定，“维修记录不全”和“优先维修处理”均已在别处有更严格的定义。请建立判定表。



建立判定表举例

● 解答：

①确定规则的个数：这里有3个条件，每个条件有两个取值，故应有 $2*2*2=8$ 种规则。

②列出所有的条件桩和动作桩：

条 件	功率大于 50 马力吗？
	维修记录不全吗？
	运行超过 10 年吗？
动 作	进行优先处理
	作其他处理



建立判定表举例

- ③填入条件项。可从**最后1行条件**项开始，逐行向上填满。如第三行是：
Y N Y N Y N Y N
第二行是：
Y Y N N Y Y N N
等等。
- ④填入动作桩和动作项。这样便得到如图的初始判定表。

		1	2	3	4	5	6	7	8
条 件	功率大于 50 马力吗?	Y	Y	Y	Y	N	N	N	N
	维修记录不全吗?	Y	Y	N	N	Y	Y	N	N
	运行超过 10 年吗?	Y	N	Y	N	Y	N	Y	N
动 作	进行优先处理	x	x	X		X		X	
	作其他处理				X		X		X

初始判定表



⑤化简。合并相似规则后得到图。

		1	2	3	4	5	6	7	8
条件	功率大于 50 马力吗?	Y	Y	Y	Y	N	N	N	N
	维修记录不全吗?	Y	Y	N	N	Y	Y	N	N
	运行超过 10 年吗?	Y	N	Y	N	Y	N	Y	N
动作	进行优先处理	x	x	X		X		X	
	作其他处理				X		x		x

初始判定表

		1	2	3	4	5
条件	功率大于 50 马力吗?	Y	Y	Y	N	N
	维修记录不全吗?	Y	N	N	-	-
	运行超过 10 年吗?	-	Y	N	Y	N
动作	进行优先处理	x	x		X	
	作其他处理			x		x

化减后的判定表

NextData函数的精简决策表*

NextData函数的精简决策表

- ① $M_1 = \{\text{月份, 每月有30天}\}$
- ② $M_2 = \{\text{月份, 每月有31天}\}$
- ③ $M_3 = \{\text{月份, 2月}\}$
- ④ $D_1 = \{\text{日期, 1} \sim 28\}$
- ⑤ $D_2 = \{\text{日期, 29}\}$
- ⑥ $D_3 = \{\text{日期, 30}\}$
- ⑦ $D_4 = \{\text{日期, 31}\}$
- ⑧ $Y_1 = \{\text{年: 年是闰年}\}$
- ⑨ $Y_2 = \{\text{年: 年不是闰年}\}$

分析:

- (1) 有 $2^9 = 512$ 条规则;
- (2) 12月末31日和其它月份的31日处理不同;
- (3) 平年2月28日处理不同于2月27日。

改进为：

- ① $M_1 = \{\text{月份: 每月有30天}\}$
- ② $M_2 = \{\text{月份: 每月有31天, 12月除外}\}$
- ③ $M_3 = \{\text{月份: 12月}\}$
- ④ $M_4 = \{\text{月份: 2月}\}$
- ⑤ $D_1 = \{\text{日期: 1} \sim 27\}$
- ⑥ $D_2 = \{\text{日期: 28}\}$
- ⑦ $D_3 = \{\text{日期: 29}\}$
- ⑧ $D_4 = \{\text{日期: 30}\}$
- ⑨ $D_5 = \{\text{日期: 31}\}$
- ⑩ $Y_1 = \{\text{年: 年是闰年}\}$
- ⑪ $Y_2 = \{\text{年: 年不是闰年}\}$



	处理M1（小月）					处理M2（大月）					处理M3（12月）					处理M4（2月）						
c1:month在:	M 1	M 1	M 1	M 1	M 1	M 2	M 2	M 2	M 2	M 2	M 3	M 3	M 3	M 3	M 3	M 4	M 4	M 4	M 4	M 4	M 4	M 4
c2:day在:	D 1	D 2	D 3	D 4	D 5	D 1	D 2	D 3	D 4	D 5	D 1	D 2	D 3	D 4	D 5	D 1	D 2	D 2	D 3	D 3	D 4	D 5
c3:year在:	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y 1	Y 2	Y 1	Y 2	—	—
a1:不可能					√															√	√	√
a2:day+1	√	√	√			√	√	√	√		√	√	√	√		√	√					
a3:day=1				√						√					√			√	√			
a4:month+1				√						√												
a5:month=1															√							
a6:year+1															√							

决策表在功能测试中的应用

- ④ 有些软件的功能需求可用判定表表达得非常清楚，在检验程序的功能时判定表也就成为一个不错的工具。

如果一个软件的规格说明指出：

- (1)当条件1和条件2满足，并且条件3和条件4不满足，或者当条件1、3和条件4满足时，要执行操作1。
- (2)在任一个条件都不满足时，要执行操作2。
- (3)在条件1不满足，而条件4被满足时，要执行操作3。



决策表在功能测试中的应用

根据规格说明得到如下判定表

	规则 1	规则2	规则3	规则4
条件 1	Y	Y	N	N
条件 2	Y	-	N	-
条件 3	N	Y	N	-
条件 4	N	Y	N	Y
操作 1	x	x		
操作 2			x	
操作 3				x

根据规则说明得到的判定表



决策表在功能测试中的应用

这里，决策表只给出了16种规则中的8种。事实上，除这8条以外的一些规则是指当不能满足指定的条件时，要执行1个默许的操作。在没必要时，判定表通常可略去这些规则。但如果用判定表来设计测试用例，就**必须列出这些默许规则（如下表）**。

	规则 5	规则 6	规则 7	规则 8
条件 1	-	N	Y	Y
条件 2	-	Y	Y	N
条件 3	Y	N	N	N
条件 4	N	N	Y	-
默许操作	x	x	x	x

默许的规则

决策表在功能测试中的应用

判定表的优点和缺点


- 优点：

它能把复杂的问题按各种可能的情况一一列举出来，
简明而易于理解，也可避免遗漏。

- 缺点：

不能表达重复执行的动作，例如循环结构。

决策表在功能测试中的应用

- ④ B. Beizer 指出了 适合使用判定表设计测试用例的条件:
 - ①规格说明以判定表形式给出,或很容易转换成判定表。
 - ②条件的排列顺序不会也不影响执行哪些操作。
 - ③规则的排列顺序不会也不影响执行哪些操作。
 - ④每当某一规则的条件已经满足,并确定要执行的操作后,不必检验别的规则。
 - ⑤如果某一规则得到满足要执行多个操作,这些操作的执行顺序无关紧要。
- ④ B. Beizer提出这5个必要条件的目的是为了使操作的执行完全依赖于条件的组合。其实对于某些不满足这几条的判定表,同样可以借以设计测试用例, 只不过还需增加其它的测试用例。 



3.1.6 错误推测法

- ④ 错误推测法的概念：基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性的设计测试用例的方法。
- ④ 错误推测方法的基本思想：列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据它们选择测试用例。
- ④ 例如：
 - 在单元测试时曾列出的许多在模块中常见的错误、以前产品测试中曾经发现的错误等，这些就是经验的总结。
 - 还有，输入数据和输出数据为0的情况、输入表格为空格或输入表格只有一行等。这些都是容易发生错误的情况，可选择这些情况下的例子作为测试用例。

3.1.6 错误推测法

例如，测试一个对线性表（比如数组）进行排序的程序，可推测列出以下几项需要特别测试的情况：

- ① 输入的线性表为空表；
- ② 表中只含有一个元素；
- ③ 输入表中所有元素已排好序；
- ④ 输入表已按逆序排好；
- ⑤ 输入表中部分或全部元素相同。

3.1.6 错误推测法

- ❶ 错误推测法的基本思想是列举出程序中所有可能有的错误或缺陷和容易发生错误或缺陷的特殊情况，根据这些推测来设计测试用例。
- ❷ 错误推测法本身不是一种测试技术，而是一种可以应用到所有测试技术中产生更加有效测试的一种技能。



3.1.7 测试方法的选择

- ④ 通常，在确定测试方法时，应遵循以下原则：
- ① 根据程序的重要性的和一旦发生故障将造成的损失来确定测试等级和测试重点。
 - ② 认真选择测试策略，以便能尽可能少的使用测试用例，发现尽可能多的程序错误。因为一次完整的软件测试过后，如果程序中遗留的错误过多并且严重，则表明该次测试是不足的，而测试不足则意味着让用户承担隐藏错误带来的危险，但测试过度又会带来资源的浪费。因此测试需要找到一个平衡点。

测试方法的选择（续）

- ④ 通常在确定**测试策略**时，有以下5条参考原则：
 - (1) 在任何情况下都必须采用**边界值分析法**。这种方法设计出的测试用例发现程序错误的能力最强。
 - (2) **必要时**采用**等价类划分法**补充测试用例。
 - (3) 采用**错误推断法**再追加测试用例。
 - (4) 如果软件的功能说明中含有输入**条件的组合**情况，也就是输入变量之间有很强的**依赖关系**，则一开始就可选用**因果图法**或**决策表法**。但是不要忘记用边界值法或其它方法设计测试用例作**补充**。
 - (5) 对照程序逻辑，检查已设计出的测试用例的**逻辑覆盖程度**。如果没有达到要求的覆盖标准，则应当再**补充**更多的测试用例。☀



本节总结

- 黑盒测试方法的基本概念
- 黑盒测试的等价类划分法
- 黑盒测试的边界值分析法
- 黑盒测试的因果图测试法和决策表法
- 测试方法的选择



练习1

1、某城市电话号码由三部分组成，分别是：

地区码——空白或三位数字；

前 缀——非‘0’或‘1’开头的三位数字；

后 缀——4位数字。

假定被测程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的电话号码。要求：

- (1) 请选择适当的黑盒测试方法，写出选择该方法的原因，并使用该方法的步骤，给出测试用例表。
- (2) 如果所生成的测试用例不够全面，请考虑用别的测试方法生成一些补充的测试用例。



本节完

