Software Testing Technique
Chapter 6

Blackbox Testing

Zan Wang（王赞）

Office: 55-A413

Email: wangzan@tju.edu.cn

2022

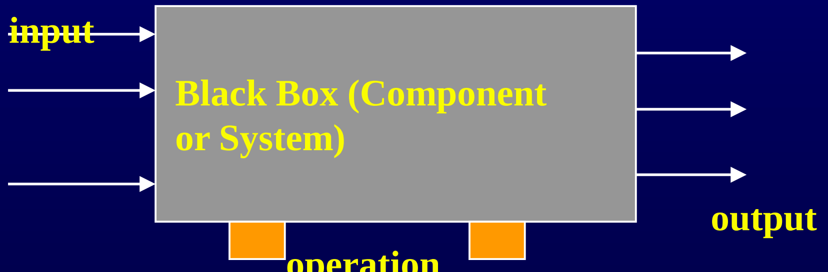
Outline

- Introduction to Blackbox Testing
- Single input parameter blackbox testing
  - Equivalence class partitioning (ECP，等价类)
  - Boundary value analysis (BVA，边界值)
- Combination of input parameters blackbox testing
  - Cause-effect graph(CEG，因果图)
  - Decision table (DT，决策表)
  - Pairwise

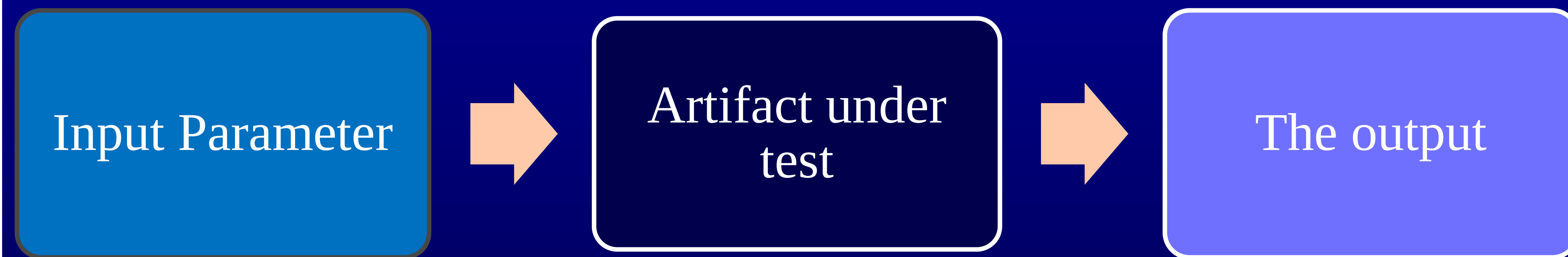
INTRODUCTION TO BLACKBOX TESTING

Blackbox Testing

- **What is black box testing?**
  - **The goal is to exercise the behaviors of the units-under-test based on its requirement, *without considering its implementation detail*. Such an artifact can be a method, a class, a sub-system, or the whole programs.**
  - **Our test suite is considered *adequate* if it covers representative choices of input to the artifact-under-test.**
- **The major testing focuses:**
  - **specification-based function errors**
  - **specification-based component/system behavior errors**
  - **specification-based performance errors**
  - **user-oriented usage errors**
  - **black box interface errors**

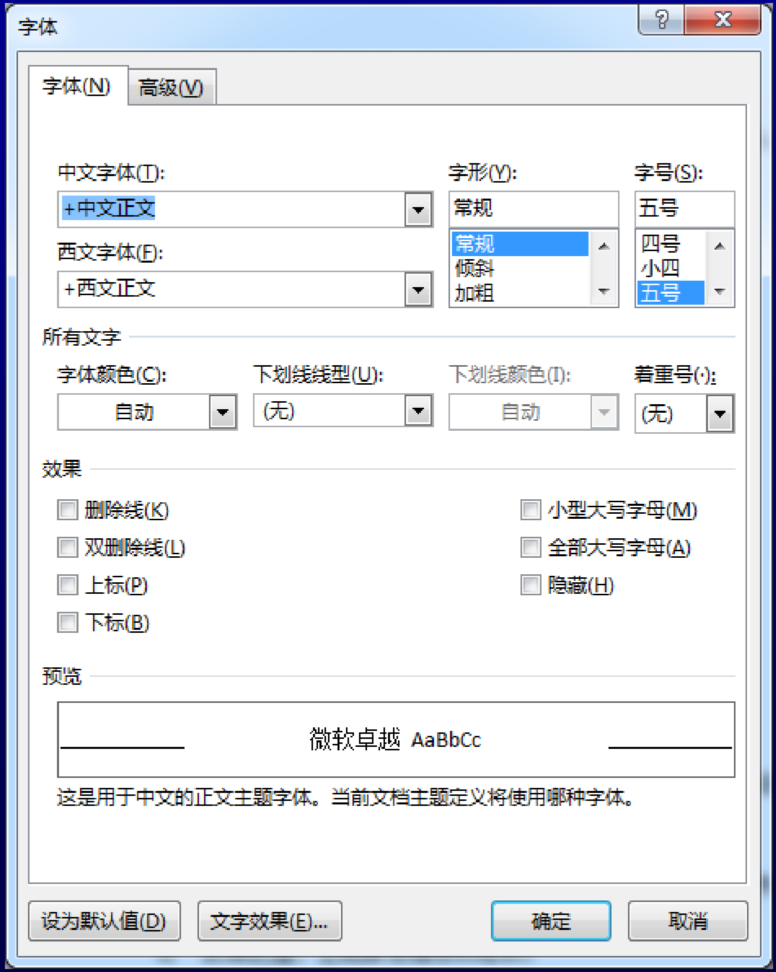


Blackbox Testing



Two kinds of Blackbox testing

- Single Input Parameter
  - Cover the representative choices for one single input parameter.
  - e.g. the font type. (宋体 or 黑体 or Arial or ….)
- Combinations of Input Parameters
  - Cover the representative combinations of multiple parameters.
  - e.g. settings in the whole font dialog. (宋体-加粗-11号-红色 or 宋体-普通-小四-黑色 or 黑体-加粗-12号-黄色 or …)



Single or Combination?

- **When the input involves multiple elements, single or multiple?**
- **One single parameter if:**
  - **Elements are handled independently in the same way. e.g. a list of user names to grant write-access.**
  - **The order of the elements matters. e.g. a list of number to sort.**
  - **The elements are tightly coupled. e.g. Coordinate (x, y).**
- **Multiple parameters if…**
  - **Elements are loosely coupled. e.g. user name vs. server address**

SINGLE INPUT PARAMETER

Single Input Techniques

- Equivalence class partitioning （ECP, 等价类划分）
  - Partition the input domain of the parameter into equivalence classes
  - Adequacy criterion: <u>cover each partition at least once</u>
- Boundary value analysis （BVA, 边界值分析）
  - Analyze the boundary cases for each equivalence class in ECP.
  - Adequacy criterion: cover each boundary case at least once.

Equivalence Class Partitioning (ECP 等价类划分)

Equivalence Class Partitioning (ECP)

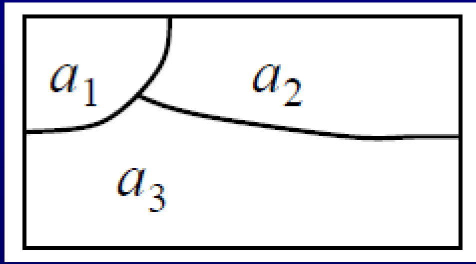
- **Test a method that implements the absolute value function for integers.**

public int abs(int x)

- **Number of test cases for exhaust testing:**
  - $2^{32}-1$
  - （$2^{32}-1$）/1000*60*60*24 : about 49 days
  - **Not practical**
- **Whether we can partition the input domain into *equivalence class*?**
  - **A set or range of input domain values can be considered to be an *equivalence class* if they can reasonably be expected to cause similar responses from the program.**
  - **Need to consider both valid classes (合法输入) and invalid classes (非法输入).**

Partition

- A partitioning of a set A is to divide A into subsets *a*1, *a2*, ..., *an* such that:
  - $a_1 \cup a_2 \cup \ldots \cup a_n = A$ (*completeness*)
  - for any *i* and *j*, $a_i \cap a_j = \emptyset$ (*disjoint*)
- Defined by a reflexive symmetric, transitive relation between input data points(equivalence relation)



- Rule to partition an input domain: group inputs that ***cause similar responses*** from the program as described in the requirement.
  - Drive the program to execute the same paths

Equivalence Class Partition(ECP)

[Integer.MIN_VALUE, -1] [0] [1,Integer.MAX_VALUE]

- **Choose a "representative" value from each class to cover each equivalence**
  - **[Integer.MIN_VALUE, -1]** **-100**
  - **[0]** **0**
  - **[1,Integer.MAX_VALUE]** **789**

Ways to Find Classes

- **From the requirement:**
  - **For a range [a, b]**
    - **Within range [a, b]: a valid class**
    - **Too large (b, ∞] : an invalid class**
    - **Too small[- ∞, a) : another invalid class**
  - **For a set of values**
    - **A valid equivalence class for each value in the enumeration.**
    - **An invalid equivalence class: all values not in the set.**
  - **To satisfy some conditions**
    - **A valid class for satisfying the conditions**
    - **An invalid class for violating each condition.**
  - **For a list of values**
    - **Create one valid class for the correct number of values**
    - **Two invalid classes: fewer values and more values**

Some additional classes

- They might not be explicitly mentioned in the requirement. But it is still necessary to test them:
  - Empty: empty string “”, empty set, empty list.
  - Null: values does not exist or is not allocated, null pointer
  - Very long: the length of the input is extremely long.
  - Special value: Feb. 29

The steps of designing test cases

- **Find out equivalence classes.**
- **Set up a table and give all equivalence classes.**
- **Every equivalence class has a unique No.**
- **Design a new test case, try to make it include those exclusive equivalence classes which doesn't be included now, repeat this step until all valid equivalence classes are included.**
- **Design a new test case, try to make it include a invalid equivalence class only which doesn't be included now, repeat this step until all invalid equivalence classes are included.**

ECP-Example 1： Triangle problem

- It receive there integers a, b, c as input. If a, b, c satisfy these conditions below:

C1.	$1 \le a \le 200$	C4.	$a < b + c$
C2.	$1 \le b \le 200$	C5.	$b < a + c$
C3.	$1 \le c \le 200$	C6.	$c < a + b$

- Four possible outputs: **Not a Triangle**, **Scalene**, **Isosceles**, and **Equilateral**.
- If no input value is satisfied then there will be a message. Example , c is out of range
- If a, b ,c satisfy the condition C1, C2,C3 then we can get four output exclusively:
  - If three sides have the same value then output "equilateral"
  - If only two sides have the same value then output "isosceles"
  - If any two sides have different value then output "scalene"
  - If one condition among C4, C5 and C6 isn't be satisfied than output "not a triangle"

ECP-Example 1：Triangle problem

- **Step1: find out equivalence classes**
  - EC1={<a，b，c>：the triangle with sides a ,b, and c is equilateral}
  - EC2={<a，b，c>：the triangle with sides a, b, and c is isosceles}
  - EC3={<a，b，c>：the triangle with sides a, b, and c is scalene}
  - EC4={<a，b，c>： the triangle with sides a, b, and c not form a triangle}

- **Step2 select test cases**

Test cases	a	b	c	Expected output
VTC1	5	5	5	Equilateral
VTC2	2	2	3	Isosceles
VTC3	3	4	5	Scalene
VTC4	4	1	2	Not a triangle
ITC1	-1	-1	5	a、b out of range
ITC2	5	-1	-1	b、c out of range
ITC3	-1	5	-1	a、c out of range
ITC4	-1	-1	-1	a、b、c out of range

Boundary Value Analysis（BVA 边界值分析)

Boundary Value Analysis

- **Definition:**
  - Analyze the boundary cases for each equivalence class in ECP
- **Motivation:**
  - It has been found that most errors are caught at the boundary of the equivalence classes.
  - E.g. checking X is in range [a, b):
    - If (X>=a && $\boxed{\text{X<=b)}}$
- **Criterion:**
  - Cover each boundary case at least once.
  - Boundary value analysis requires that these boundary cases being covered by the test suite.
  - It is an extension and refinement of ECP.

Boundary Value Analysis (BVA)

- **For each boundary between classes, cover:**
  - **The boundary**
  - **Just above the boundary by one min_unit**
  - **Just below the boundary by one min_unit**
- **Test cases of BVA are supplementary for ECP.**

Ways to find boundary cases

- **If the parameter shall be within a range, [X … Y]**
  - Four values should be tested:
    - Valid: X and Y
    - Invalid: Just below X (X – min_unit)
    - Invalid: Just above Y (Y + min_unit)
  - E.g., [3 … 20], and the min_unit is 1
    - Test Data: {2, 3, 20, 21}
- **Similar for open interval (X … Y), i.e., X and Y are not inclusive.**
  - Four values should be tested:
    - Invalid: X and Y
    - Valid: Just above X (e.g., X + min_unit)
    - Valid: Just below Y (e.g., Y – min_unit)
  - e.g., (100 … 200) and the min_unit is 1
    - Test Data: {100, 101, 199, 200}

- **If the parameter is one from a list of values:**
  - Four values should be tested:
    - Valid: min, max
    - Invalid: Just below min (e.g., min – min_unit)
    - Invalid: Just above max (e.g., max + min_unit)
  - E.g., the input parameter shall be one in {2, 4, 6, 8} and min_unit is 1.
    - Test Data: {1, 2, 8, 9}
- **If the parameter is a data structure:**
  - Define test data to exercise the data structure at the prescribed boundaries.
  - E.g.,
    - Array : Empty Array, Array with 1 element, Full Array
    - Set: Empty set, set with 1 element
- **Boundary value analysis is not applicable for data with no meaningful boundary, e.g., the set color {Red, Green, Yellow}.**

BVA-Example 1: Triangle Problem

- **Three inputs: a、b、c**
  - $1 \le a \le 200$
  - $1 \le b \le 200$
  - $1 \le c \le 200$

  - **a = {0, 1，*100*， 200, 201}**
  - **b = {0, 1，*100*， 200, 201}**
  - **c = {0, 1，*100*， 200, 201}**

BVA-Example 1: Triangle Problem

Triangle problem test case

Test case	A	B	C	Expected output
1	100	100	1	Isosceles
2	100	100	0	Not Valid
3	100	100	100	Equilateral
4	100	100	201	Not Valid
5	100	100	200	Not a triangle
6	100	1	100	Isosceles
7	100	0	100	Not Valid
8	100	100	100	Equilateral
9	100	201	100	Not Valid
10	100	200	100	Not a triangle
11	1	100	100	Isosceles
12	0	100	100	Not Valid
13	100	100	100	Equilateral
14	201	100	100	Not Valid
15	200	100	100	Not a triangle

BVA-Example2: Charges of Mailing

- **A mail order company charges $2.95 postage for deliveries if the package weighs less than 2 kg, $3.95 if the package weighs 2 kg or more but less than 5 kg, and $5 for packages weighing 5 kg or more.**
  - **What are the classes for the parameter "package weighs"?**
  - **What are the boundary cases that are required to be covered by BVA? (assume the min_unit is 1 gram).**

- **Classes:**
  - **C1: 0 (invalid class)**
  - **C2: (0, 2kg)**
  - **C3: [2kg, 5kg)**
  - **C4: [5kg, ∞)**
- **Boundary:**
  - **C1: 0kg**
  - **C2: 0.001kg, 1.999kg, 2kg**
  - **C3: 4.999kg, 5kg**
  - **C4: 50kg**

Some Problems of ECP and BVA

- **Problem 1:** ECP asks us to partition the domain of the input parameter into different classes by the different ways the program handle it. But how do we know these ways?
- **Problem 2:** When the input is subject to complicated conditions and their combinations, how do we design the equivalence classes?
- **The idea: Partition the input domain by how the program responds (generate output) to different input conditions.**
  - **Establish the relation between input and output by reading the requirement.**

MULTIPLE INPUT PARAMETER

Multiple Input Techniques

- **Cause-effect graph and decision table（因果图和决策表）**
  - **Analyze the causal relation between input and output as edges.**
  - **Adequacy criterion: cover each edge at least once.**
- **Pairwise**

Decision Table(判定表方法)

Multiple Input

- Sign In
  - Username
  - Password
- 保险客户升级
  - 连续在本公司投保超过两年
  - 上一年度没有出险
  - 今年保费超过3500
  - VIP升级

Decision Table

- Of all the functional testing methods, those based on decision tables are the most rigorous, because decision tables themselves enforce logical rigor.
- They are ideal for describing situations in which a lot of combinations of actions are taken under varying sets of conditions.

Five portions of a decision table

- **Four portions of decision table:**
  - **Condition Stub** portion
  - **Action Stub** portion
  - **Condition Entry** portion
  - **Action Entry** portion
- **Rules indicate which actions are taken for the conditional circumstances indicated in the condition portion of the rule.**

Steps of constructing a decision table

- **Determine the rules number.**
  - **if there are *n* conditions, there must be 2^n rules.（True/False, Yes/No, 0/1 ）**
- **List all condition subs and action subs**
- **Enter condition entry**
- **Enter action entry, To obtain initial decision table**
- **Simplify decision table**
  - **If there are two more rules have *the same action*, and the conditions are very similar, it could be merged.**
  - **Don’t Care entry has two major interpretations: the condition is irrelevant, or the condition does not apply.**

DT-Example 1: Triangle

c1: a、b、c get a triangle?	N	Y	Y	Y	Y	Y	Y	Y	Y
c2: a=b?	—	Y	Y	Y	Y	N	N	N	N
c3: a=c?	—	Y	Y	N	N	Y	Y	N	N
c4: c=b?	—	Y	N	Y	N	Y	N	Y	N
a1: not a triangle	×								
a2: scalene									×
a3: isosceles					×		×	×	
a4: equilateral		×							
a5:impossible			×	×		×			

DT-Example 1: Triangle

c1: a <b + c?	F	T	T	T	T	T	T	T	T	T	T
c2: b <a + c?	—	F	T	T	T	T	T	T	T	T	T
c3: c <b + a?	—	—	F	T	T	T	T	T	T	T	T
c4: a = b?	—	—	—	T	T	T	T	F	F	F	F
c5: a = c?	—	—	—	T	T	F	F	T	T	F	F
c6: b = c?	—	—	—	T	F	T	F	T	F	T	F
a1: not a triangle	×	×	×								
a2: scalene											×
a3: isosceles							×		×	×	
a4: equilateral				×							
a5:impossible					×	×		×			

DT-Example 2

- 条件、动作

ID	项目名称
C1	此商品在经营范围
C2	此商品可以发货
C3	此客户没有拖欠过付款
ID	**项目名称**
A1	货到后允许客户转账
A2	货到客户必须立即付款
A3	重新组织货源
A4	电话通知
A5	书面通知

判定表方法-根据条件组合给出动作

	ID	项目名称	1	2	3	4	5	6	7	8
条件	C1	此商品在经营范围	N	N	N	N	Y	Y	Y	Y
	C2	此商品可以发货	Y	Y	N	N	Y	Y	N	N
	C3	此客户没有拖延过付款	Y	N	Y	N	Y	N	Y	N
活动	A1	货到后允许客户转账					1			
	A2	货到客户必须立即付款						1		
	A3	重新组织货源							1	1
	A4	电话通知							1	
	A5	书面通知	1	1	1	1				1

判定表方法-优化

	ID	项目名称	1	2	3	4	5	6	7	8
条件	C1	此商品在经营范围	N	N	N	N	Y	Y	Y	Y
	C2	此商品可以发货	-	-	-	-	Y	Y	N	N
	C3	此客户没有拖延过付款	-	-	-	-	Y	N	Y	N
活动	A1	货到后允许客户转账					1			
	A2	货到客户必须立即付款						1		
	A3	重新组织货源							1	1
	A4	电话通知							1	
	A5	书面通知	1	1	1	1				1

判定表方法-合并

	ID	项目名称	1	2	3	4	5
条件	C1	此商品在经营范围	N	Y	Y	Y	Y
	C2	此商品可以发货	-	Y	Y	N	N
	C3	此客户没有拖延过付款	-	Y	N	Y	N
活动	A1	货到后允许客户转账		1			
	A2	货到客户必须立即付款			1		
	A3	重新组织货源				1	1
	A4	电话通知				1	
	A5	书面通知	1				1

Cause-Effect Graph(CEG 因果图法)

Cause-Effect Graph

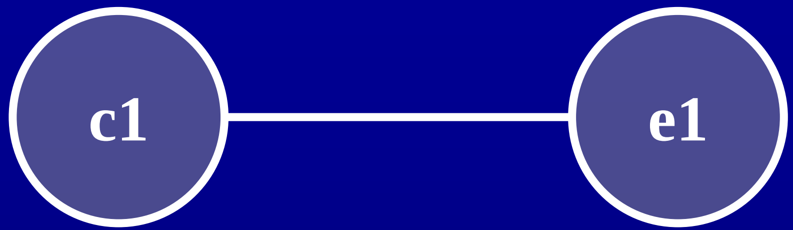
- **Cause-effect graph models dependency between program input conditions (known as causes) and output condition (known as effects).**
  - **A cause is any condition in the requirements that may affect the program output.**
  - **An effect is the response of the program to some combination of input conditions.**
- **Example: the input parameter is a list of integers:**
  - **Display message A if input contains 1 and 2.**
  - **Display message B if input contains 2 and 3.**
  - **Display message C if input contains 1 or 3.**
  - **Display message D if input contains 1, 2, and 4.**
  - **One and only one of {3, 4} must be in the input.**

Cause-Effect Graphing

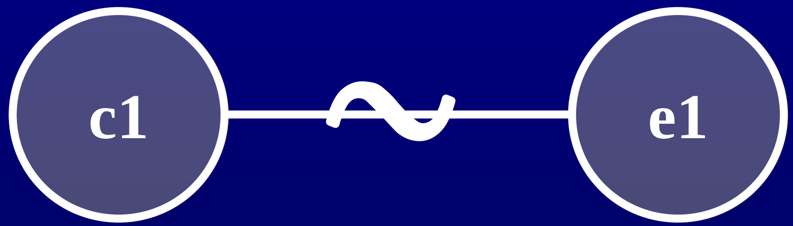
- CEG is based on such a method:
  - Some functions of a program can be displayed by judgment table, and provisions the operations base on the combinations of input conditions.
  - Therefore, we can design test case for every column of judgment table to test whether we can get the correct output by using the combination of input conditions.
- Generally speaking: CEG find out cause (input conditions) and effect (output or program state change) relations from specification. Then change CEG into judgment table, and design test cases for every columns.
- CEG method considers all kinds of combinations of input and constraint relations among outputs .

Cause-effect relationship

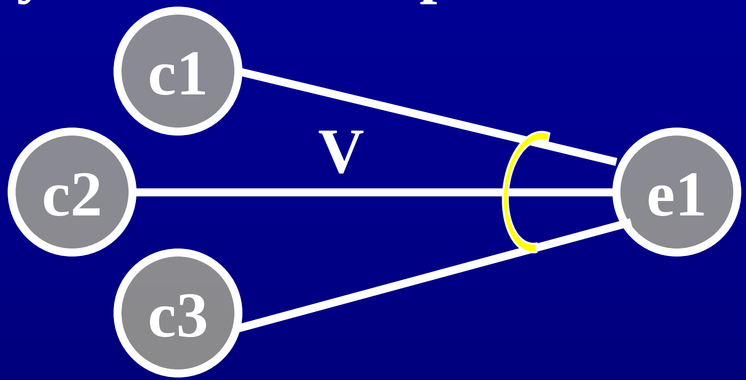
- （1）Identical： if c1 is 1， then e1 is 1 also， else e1 is 0.



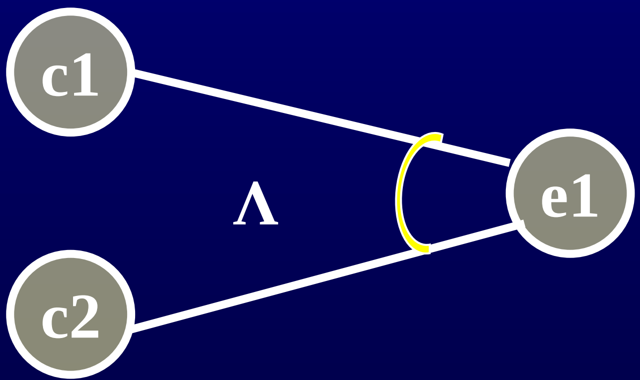
- （2）Not： if c1 is 1， then e1 is 0， else e1 is 1.



（3）**Or**： if c1 or c2 or c3 is 1， then e1 is 1， else e1 is 0； “Or” may have arbitrary number of inputs.



（4）**And**： if both c1 and c2 are 1， then e1 is 1， else e1 is 0； “And” may have arbitrary number of input.



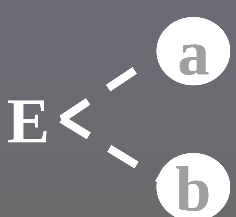
CEG：Constraint

- **Constraint**
  - In practical issues, the input states also exist certain dependencies, called constraint.
  - There are constraints among outputs states .
  - In Cause effect graph, we will use some specific symbols to show these constraints

E(互斥)

a and *b* can not be 1

at the same time



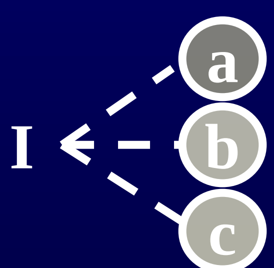
I(包含)

One of *a*、*b* and *c*

must be 1 at least

They can't be 0

at the same time



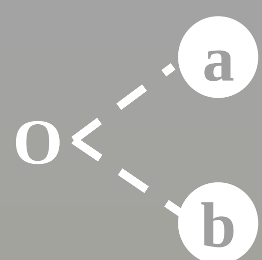
O(唯一)

Only one of

a and *b* is1

and must

one of them is 1

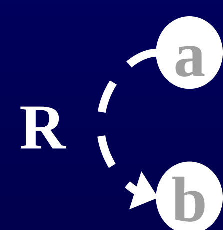


M(强制)

If the result of *a*

is 1 then

b must be 0



R(要求)

If *a* is 1

then *b* must be 1

The steps of designing a cause-effect graph

- According to the description of specification to analysis and determine “Cause” and “Effect”.
  - Usually causes are input conditions or equivalence classes of input conditions,
  - effects are “CEG” among every output condition and output result.
- Analysis the semantic content
  - find out correspondences among causes.
  - find out correspondences between causes and effects.
  - link into CEG
- Due to the restrictions of grammar or the environment, certain combinations of causes or causes and effects are impossible, we can use some marks to indicate these restrictions or limitations.
- Change CEG into judgment table
- Design test case for every column of judgment table.

CEG-Example 1: 中国象棋中马的走法

1. 如果落点在棋盘外，则不移动棋子；
2. 如果落点与起点不构成日字型，则不移动棋子；
3. 如果落点处有自己方棋子，则不移动棋子；
4. 如果在落点方向的邻近交叉点有棋子（绊马腿），则不移动棋子；
5. 如果不属于1-4条，且落点处无棋子，则移动棋子；
6. 如果不属于1-4条，且落点处为对方棋子 (非老将)，则移动棋子并除去对方棋子；
7. 如果不属于1-4条，且落点处为对方老将，则移动棋子，并提示战胜对方，游戏结束。

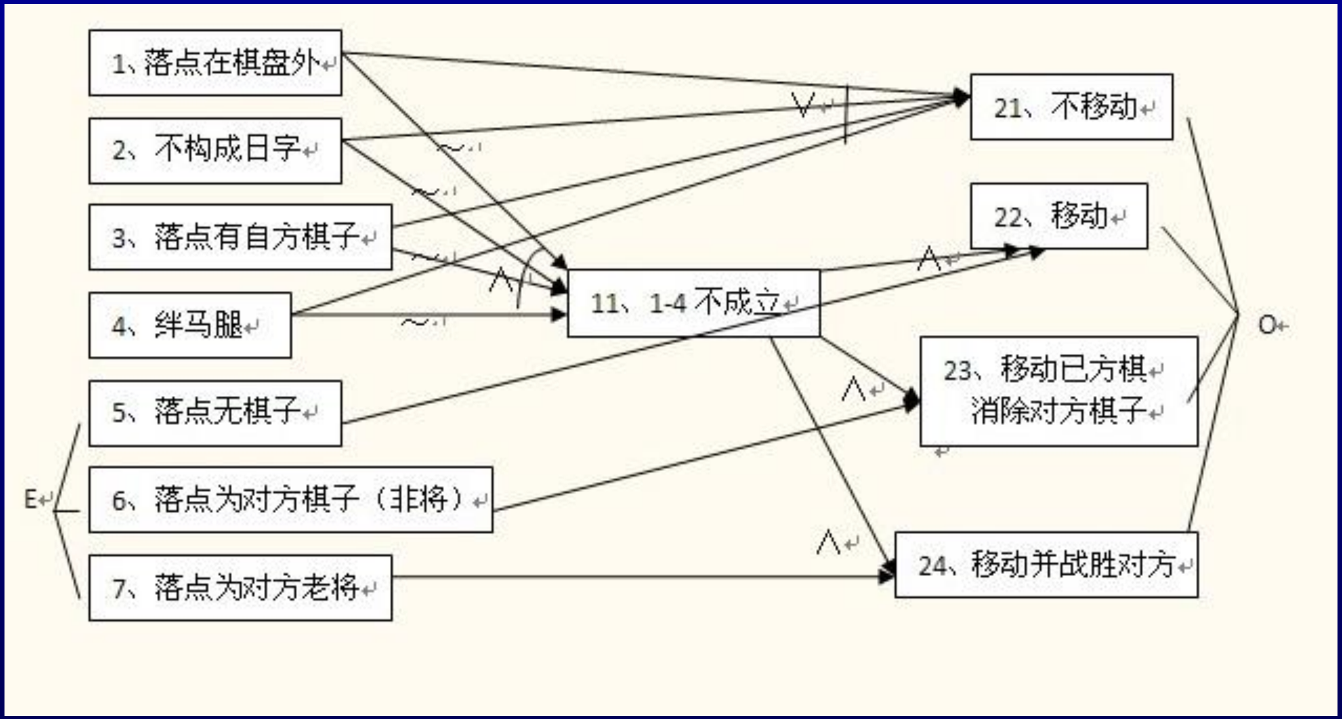
中国象棋中马的走法-原因

1. 落点在棋盘外；
2. 不构成日字；
3. 落点有自方棋子；
4. 绊马腿；
5. 落点无棋子；
6. 落点为对方棋子；
7. 落点为对方老将。

中国象棋中马的走法-结果

21. 不移动；
22. 移动；
23. 移动己方棋子消除对方棋子；
24. 移动并战胜对方。

将“因”和“果”表示成“因果图”,并标明约束条件;



CEG->DT

条件	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
	5	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0
	6	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0
	7	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	1	0
中间结果	11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
结果	21	1			1	1	1				1	1	1	1	0	0	0	
	22	0			0	0	0				0	0	0	0	1	0	0	
	23	0			0	0	0				0	0	0	0	0	1	0	
	24	0			0	0	0				0	0	0	0	0	0	1	

Pairwise

多因素组合测试

- 判定表和因果图，是1和0，两个取值
  - N个条件： 2的n次方
- 如果不是2个取值，而是多个，怎么办？
- 在线保险系统

保费计算

- 车辆行驶城市：
  - 一线、二线、三线、四线
- 车价：
  - 0-10万，10-25万，25-50万, 50-100万, 100万以上
- 车龄：
  - 1年以内，1-2年，2-6年，6年以上
- 驾龄：
  - 1年以下，1-3年，3年以上
- 交强险价格是浮动费率，与上年出险情况相关。
- 4*5*4*3*2=480*10分钟/测试=4800分钟=80小时

更复杂的保险组合？

- 当参数更多，容易出现组合爆炸
  - 6*7*5*3*2*5 = 6300 * 10 = 63000/60=1050小时/8=131工作日/5 =26周，半年

重温测试的目的

- 测试工作的目的
  - 最大限度的发现系统中存在的问题，即我们一般所称的bug
  - 测试管理所要解决的问题
    - 测试覆盖率
    - 测试效率
    - 测试工作量

先看缺陷统计数据/测试覆盖率

- NIST（美国国家标准研究所）
-

Vars	Medical Devices	Browser	Server	NASA GSFC	Network Security
1	66	29	42	68	20
2	97	76	70	93	65
3	99	95	89	98	90
4	100	97	96	100	98
5		99	96		100
6		100	100		

两两组合（Pairwise）方法

- 在不可能对所有测试产品功能点进行覆盖、遍历的情况下，如何用最少的工作量发现最多的缺陷，即测试性价比问题。
  - 绝大部分缺陷是在两个变量取值冲突的测试被发现的。

Defect Trigger	Cumulative % of Defects		
	Min	Max	Avg
Single Para	30%	70%	60%
2 paras	70%	95%	86%
3 paras	88%	99%	91%

  - 所以，测试所有的“Pairwise”就基本满足质量要求。

如何实现两两组合

- 因素A：A1，A2
- 因素B：B1，B2
- 因素C：C1，C2，C3
- 完全组合：2*2*3 = 12

A	B
A1	B1
A1	B2
A2	B1
A2	B2

A	B	C
A1	B1	C1
A1	B2	C2
A2	B1	C3
A2	B2	C1
A1	B2	C3
A2	B1	C2

再回到保险组合

- 用手工构造两两组合是很恐怖的
- 用工具构造：
  - PICT
  - CTS
  - DDA
  - …

Pairwise大大降低测试工作量