测试与静态分析
复习

严俊

University of Chinese Academy of Sciences

Institute of Software, Chinese Academy of Sciences

Effective measures

- Documentation
- Coding guidelines
- Formal verification
- Reviewing
- Static analysis
- Testing
- …

Testing and Debugging

- Testing tries to determine if a program has any errors.　测试

- Debugging tries to determine the cause of the errors.　调试

- Program under test; System under test (SUT) 被测程序/系统
- Test case （测试案例） ➔ Test suite / Test set
- Test Data （测试数据）
- Test Oracle （测试预言？）

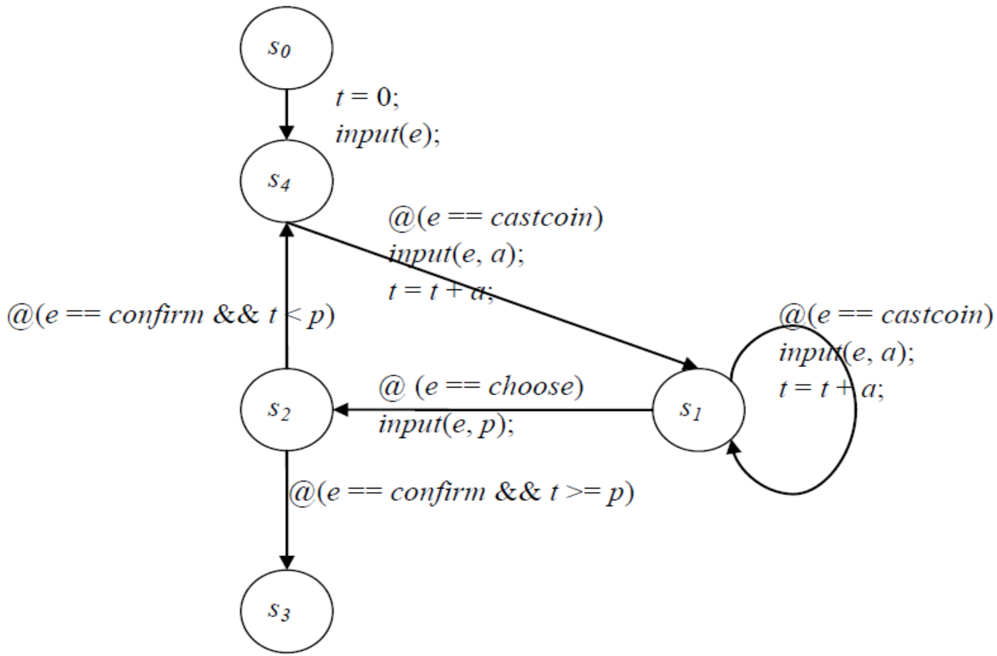
Test Cases

- A test case is a pair consisting of test data to be input to the program and the expected output. The test data is a set of values, one for each input variable.

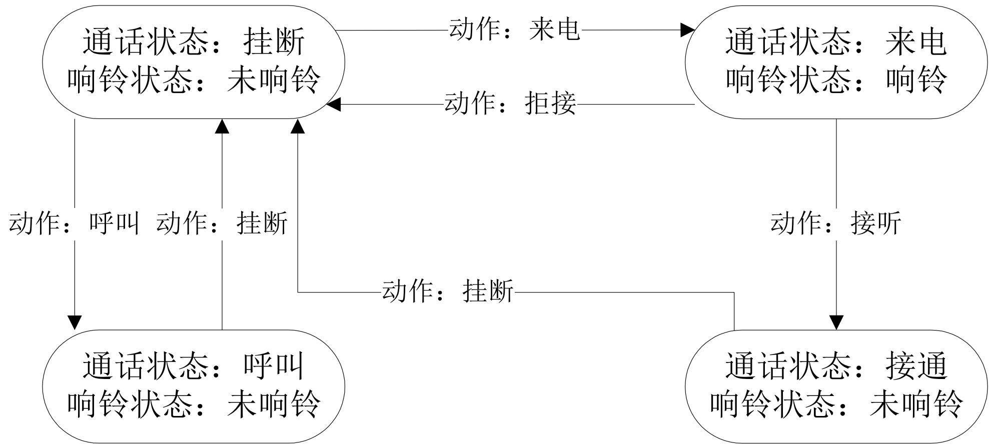
- A test set/suite is a collection of zero or more test cases.

Model-based Testing

- Specify the system with a Model
- Extended Finite State Machine (EFSM)

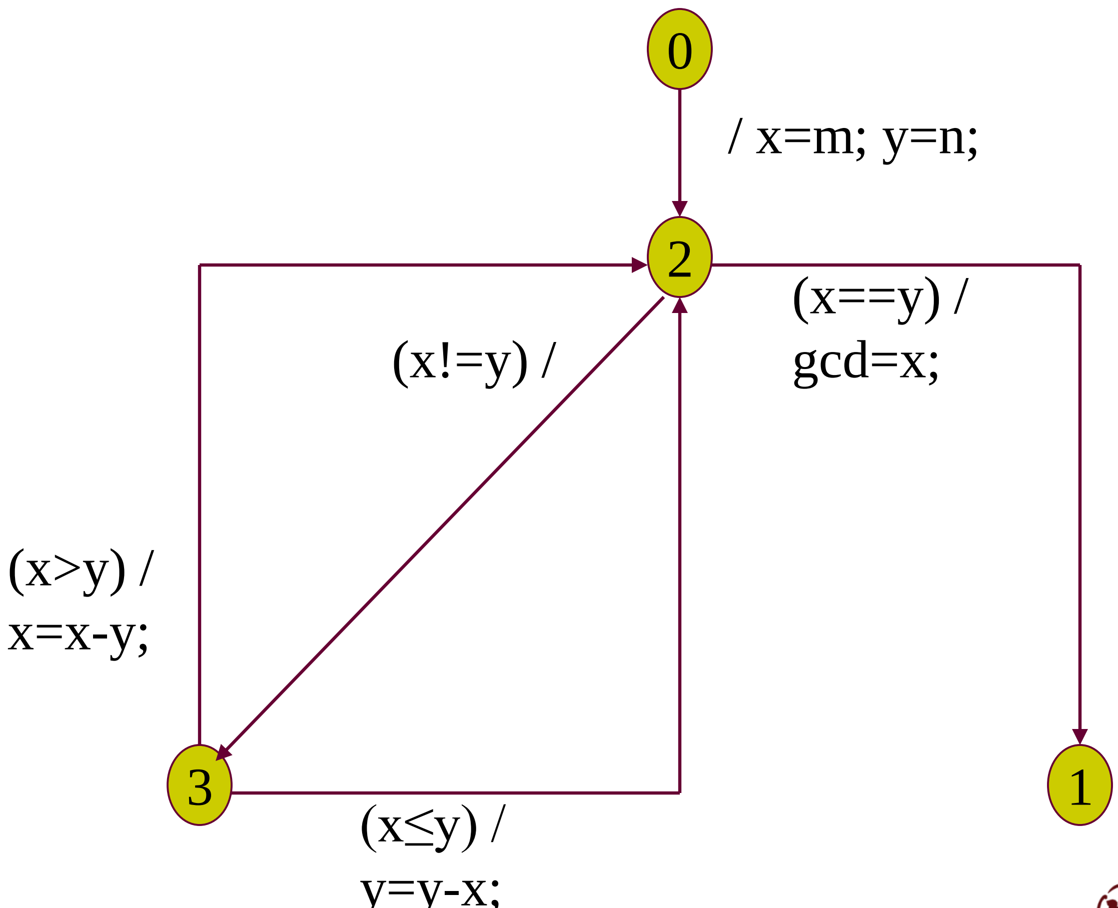


电话系统模型



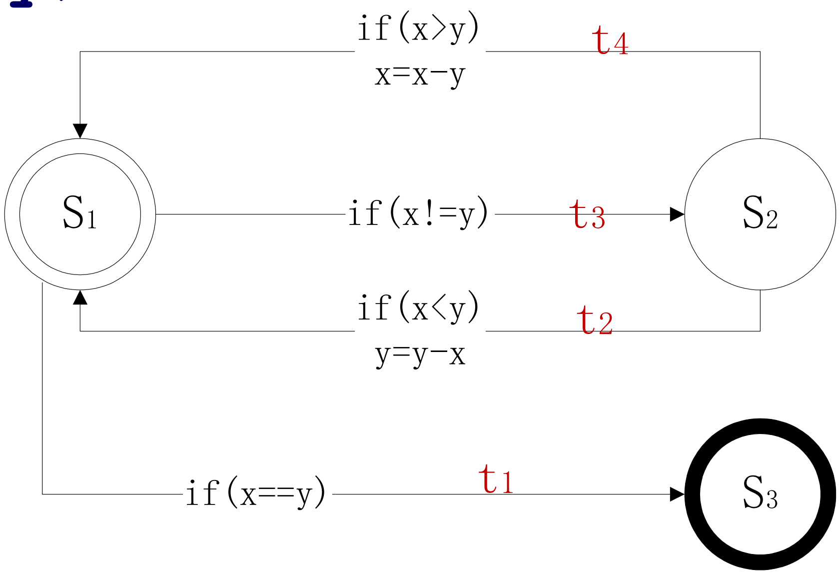
通话状态：挂断
响铃状态：未响铃
动作：来电
通话状态：来电
响铃状态：响铃
动作：拒接
动作：呼叫
动作：挂断
动作：接听
动作：挂断
通话状态：呼叫
响铃状态：未响铃
通话状态：接通
响铃状态：未响铃

EFSM for gcd()



用EFSM描述程序

```
int GCD(int x, int y)
{
    while (x != y) {
      if (x > y) x = x - y;
      else y = y - x;
    }
   return x;
}
```



迁移

t2 = (s2, s1, x<y, y=y-x)

一条测试路径

- t3 -> t2 -> t3 -> t4 -> t1
- 执行序列

  @ (x != y) /* t3 */

  @ (x < y) /* t2 */

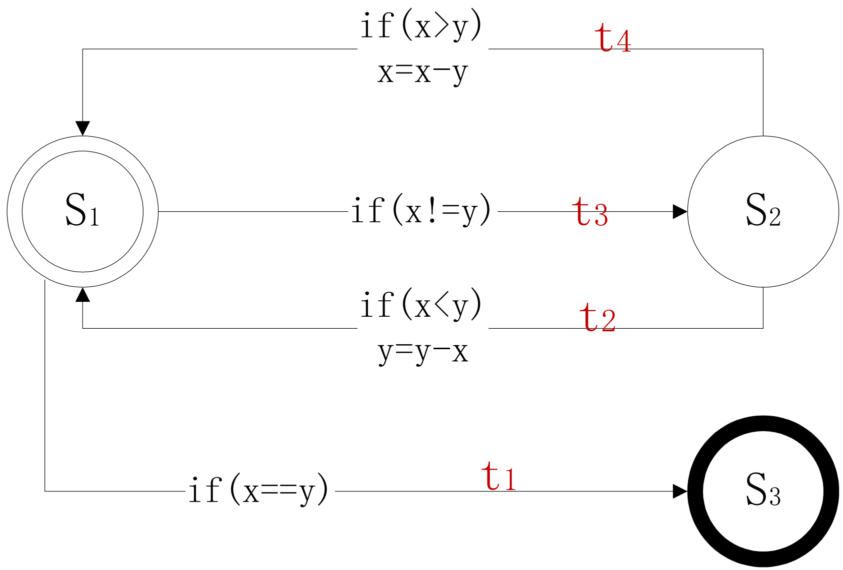
  y = y - x

  @ (x ! = y) /* t3 */

  @ (x > y) /* t4 */

  x = x – y

  @ (x == y) /* t1 */



- 测试输入(x, y) = (2, 3)

组合测试

- ## 系统配置

Client	Web Server	Payment	Database
Firefox IE Opera	WebSphere Apache .NET	MasterCard Visa AmEx	DB/2 Oracle Access

- ## 测试集

Client	Web Server	Payment	Database
Firefox	WebSphere	MasterCard	DB/2
Firefox	.NET	AmEx	Oracle
Firefox	Apache	Visa	Access
IE	WebSphere	AmEx	Access
IE	Apache	MasterCard	Oracle
IE	.NET	Visa	DB/2
Opera	WebSphere	Visa	Oracle
Opera	.NET	MasterCard	Access
Opera	Apache	AmEx	DB/2

Test Oracle

- The oracle problem is often much harder than it seems, and involves solving problems related to controllability and observability.
- Method postconditions are commonly used as automated oracles in automated class testing.
- Common oracles include:
  - specifications and documentation
  - other products
  - a heuristic oracle that provides approximate results or exact results for a set of a few test inputs
  - a human oracle (i.e. the correctness of the system under test is determined by manual analysis)

程序执行结果的检查

- 无法直接判断实际运行结果的正确性
- 程序多次执行结果之间是有关系的
- 例如，对于函数 sin(*x*)的测试
  - $\sin(x) = \sin(2\pi + x)$
  - $\sin(x) = -\sin(\pi + x)$
  - $\sin^2(x) + \sin^2(\pi/2 - x) = 1.0$
  - ……

蜕变测试

- Metamorphic Testing
- 利用多次执行结果的关系构造蜕变关系（Metamorphic Relation）
- 判断大量的随机测试结果是否满足蜕变关系

差分测试 (Differential Testing)

- 同样的测试输入，对于不同的系统，应该有一致的结果
  - 同一系统的不同版本（纵向）
  - 不同的系统（横向）

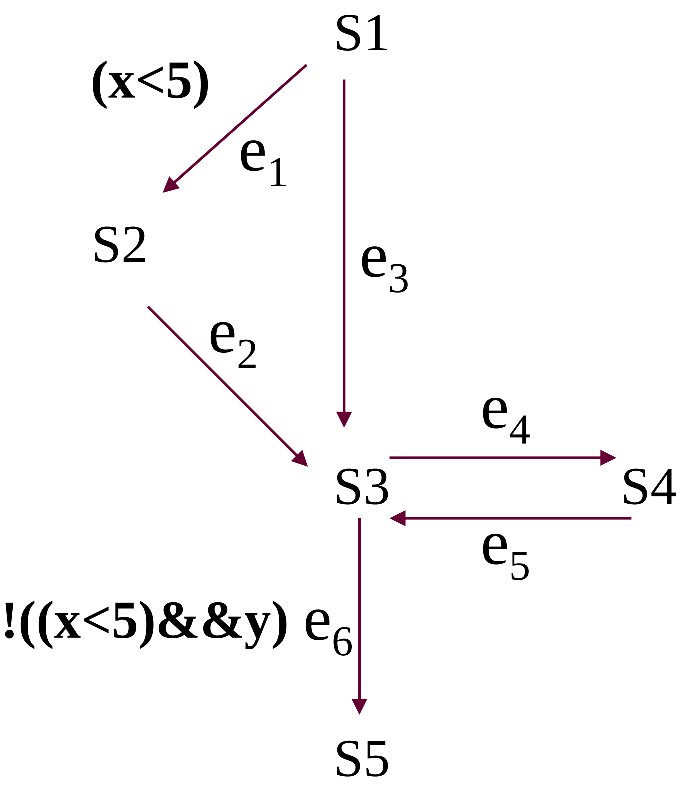
- 测试应用
  - 编译器测试
  - 浏览器测试
  - ……

- 白盒测试基本概念

Example. A program and its flow graph

```
 int x, y;

/*S1*/  if (x < 5)
/*S2*/    y = 2;
/*S3*/  while ((x < 5) && y)
/*S4*/    { x++; y--; }
/*S5*/  return;
```



EFSM与CFG

	边	节点
EFSM	迁移条件、动作（事件）	状态
CFG	迁移条件	基本块（顺序执行的语句）

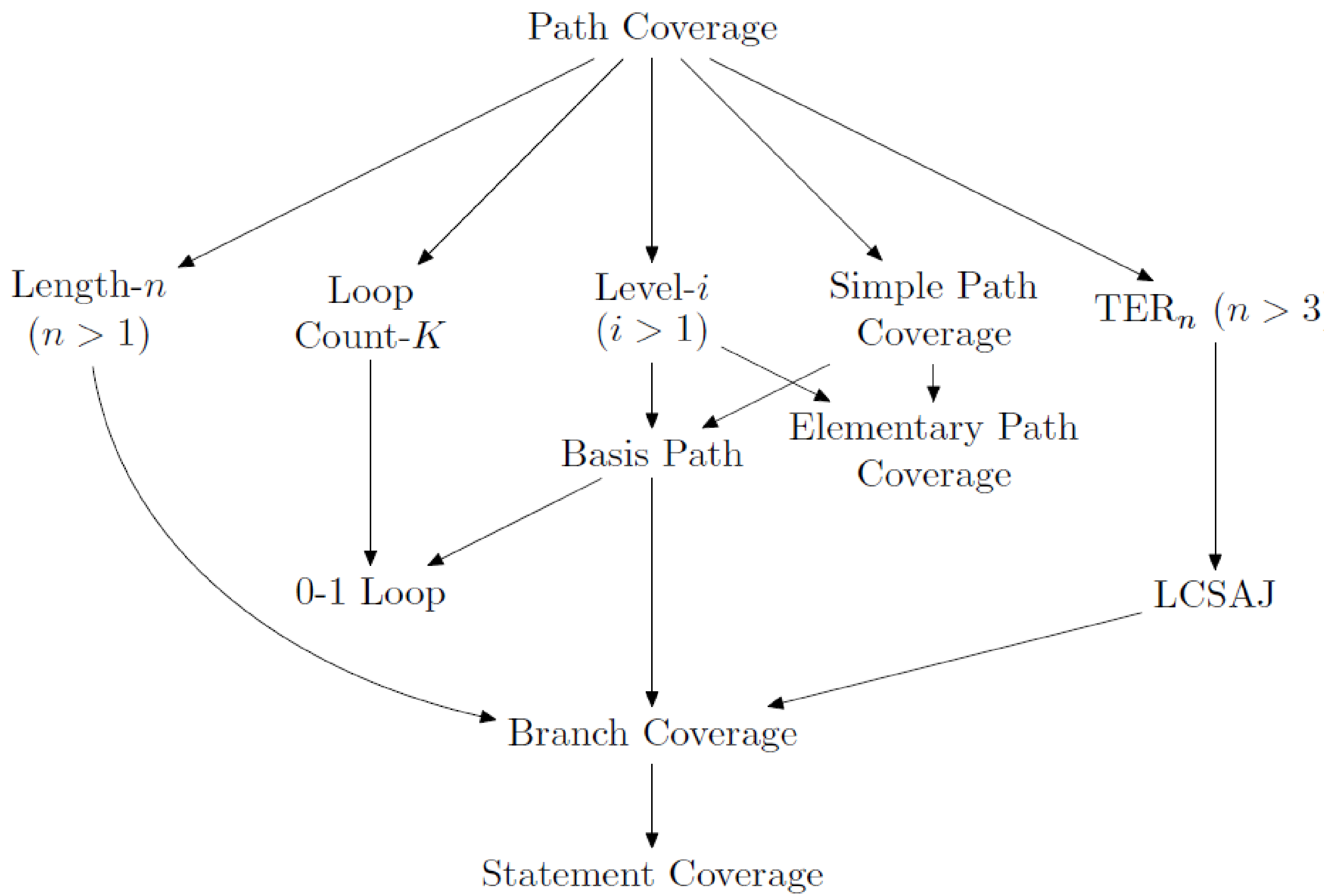
覆盖标准

- 控制流
  - Statement, branch, path (basis path)
  - decision, condition, MC/DC
- 数据流
  - (def, use)
  - All-defs, All-uses, All-DU-paths

覆盖准则

- 语句覆盖（Statement Coverage）
  - 程序中的所有语句都被执行
  - 覆盖控制流图中所有的节点
- 分支覆盖（Branch Coverage）
  - 控制转移都得到了覆盖
  - 测试路径覆盖了控制流图中所有的边
  - 对应于迁移条件表达式的判定覆盖
- 路径覆盖（Path Coverage）
  - 覆盖控制流图中所有的完整路径
  - 进一步定义可行路径覆盖(Feasible Path Coverage)
  - （可行）路径数量可能是无限的

控制流测试准则之间的关系



条件表达式

- （布尔）逻辑公式
- Decision 判定/判断
- Condition 条件

- 例： $S = (x1 \vee x2) \wedge (x3 \vee x4)$

  四个条件

Decision and Condition coverage

- **Decision coverage (DC) requires two test cases for each decision: one for a true outcome and another for a false outcome.**

- **Condition coverage (CC) requires that each condition in a decision take on all possible outcomes at least once.**

条件表达式测试

- **判定覆盖DC**
  - 对于每个判断，至少有两个测试数据使其在运行中分别取真以及假
  - 两个测试用例
- **条件覆盖CC**
  - 每个判断中的每个条件，至少有两个测试数据使其在运行中分别取真、假值
  - 两个测试用例

S = (x1∨x2)∧(x3∨x4)

判定覆盖测试集

x1	x2	x3	x4	S
0	0	0	0	0
1	0	1	0	1

条件覆盖测试集

x1	x2	x3	x4	S
0	1	0	1	1
1	0	1	0	1

MC/DC

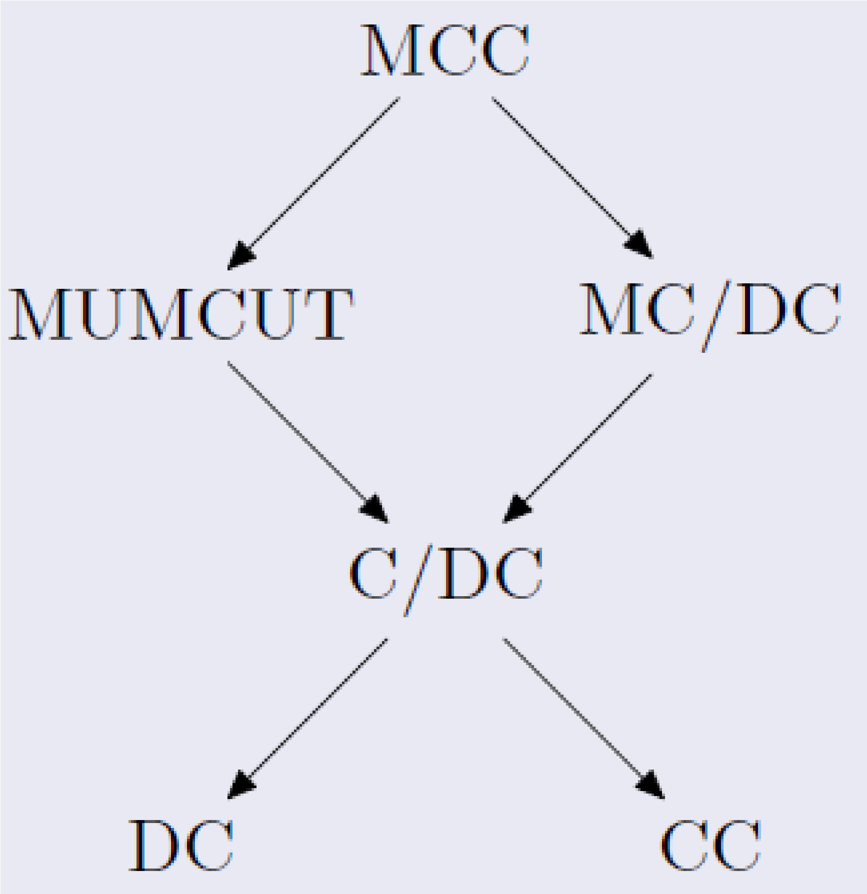
- It requires that each condition be shown to independently affect the outcome of the decision.
- Example. For the formula (A or B), we have test cases: (TF), (FT), and (FF).

例

- S = (x1∨x2) ∧ (x3∨x4)
- 5个测试用例
- {t1, t3} 覆盖条件x1
  {t1, t2} 覆盖条件x2
  {t4, t5} 覆盖条件x3
  {t2, t4} 覆盖条件x4

用例	x1	x2	x3	x4	S
t1	0	0	0	1	0
t2	0	1	0	1	1
t3	1	0	0	1	1
t4	0	1	0	0	0
t5	0	1	1	0	1

条件表达式覆盖准则之间的关系



圈复杂度 (Cyclomatic Complexity)

- 欧拉公式：G是连通平面图，那么
  $\varphi = e - n + 2$
  φ 是图中不同区域的个数 – 圈
  e 是G中的边数
  n 是G中的节点数
- 图的圈复杂度由$V(G) = e - n + 2p$给出，其中
  e 是G中的边数
  n 是G中的节点数
  p 是G中的连接组件数

圈复杂度 (Cyclomatic Complexity)

- 程序的控制流图
  - 增加一个分支，控制流图的区域数+1
  - 如果所有分支都是二叉的，那么 *V*(*G*)是判定节点数+1
- 圈复杂度代表了程序的逻辑复杂度
- 经验表明，程序中可能存在的Bug数和圈复杂度有着很大的相关性
- 测试应该和圈复杂度关联

基本路径测试

- 路径向量$A = \langle a_1, a_2, \ldots, a_k \rangle$， a_i 表示编号为 i 的有向边在路径 A 中出现的次数
- 路径B称为路径 $A_1, A_2, \ldots, A_n$ 的线性组合，如果存在常数 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 使得 $B = \lambda_1 A_1 + \lambda_2 A_2 + \ldots + \lambda_n A_n$
- 对于一个测试路径集合来说，如果流图中的任意一个完整路径都是测试集合中路径的线性组合，那么这个测试集合满足基本路径覆盖（Basis Path Coverage）。

Path Feasibility – Ex.

```
int  i, j;
bool good;
if ((i > 2) && (j > 3))
{
    j = j-1;
    if (i+2j < 5)
       good = FALSE;
    else  good = TRUE;
}
```

Two paths

- **Path 1:**

@((i > 2) && (j > 3))

j = j-1;

@(i+2j < 5)

good = FALSE;

- **Path 2:**

@((i > 2) && (j > 3))

j = j-1;

@! (i+2j < 5)

good = TRUE;

Another example

```
int x, j;

    x = 3;
    j = 2*x + 1;
  @(j < 4);
```

```
int x, j;

   // x = 3;
   j = 2*x + 1;
  @(j < 4);
```

符号执行（Symbolic Execution）

用符号（而不是具体的数值）作为输入来“执行”程序。

Ex. After the assignment $x = a+b$, the variable x gets the symbolic value a_0+b_0, rather than some concrete value such as 3+5=8.

[Boyer et al. 1975] [King 1976] [Clarke 1976]

A path in bubble-sort

```
 i = n-1;
@ i > 0;
 indx = 0;
 j = 0;
@ j < i;
@ a[j+1] < a[j];
 temp = a[j];
 a[j] = a[j+1];
 a[j+1] = temp;
 indx = j;   j = j+1;
@ j >= i;
 i = indx;
@ i <= 0;
```

- **Path condition:**

  n-1 > 0

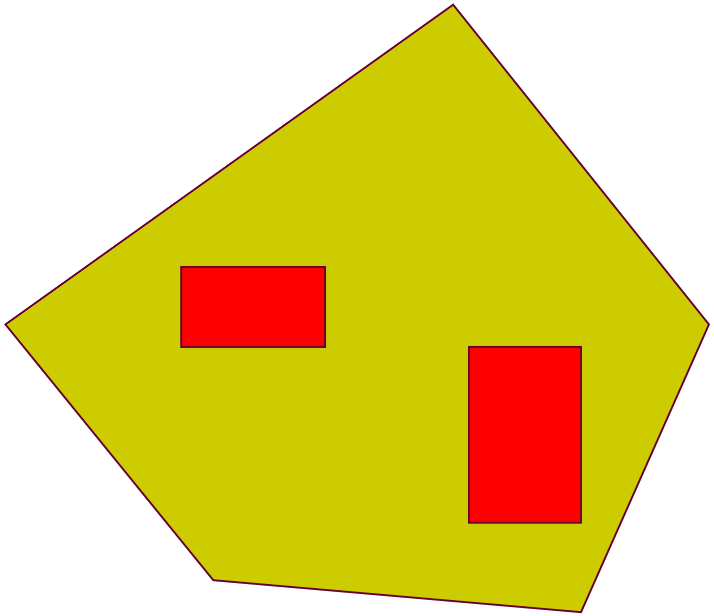
  a[1] < a[0]

  n-1 <= 1

具体输入：n = 2

a: { 3, 2 } 或 { 2, 1} …

Symbolic Execution + Constraint Solving



Checking areas (rather than points) in the input space.
What are the sizes of the areas? How much is covered?

Branch selection--Example

```
int x;
@ ((x <= 100) && (x > 20))
{
  x = x - 10;
  if (x > 30)
     ... //TRUE branch
  else
     ... //FALSE branch
}
```

Constraints:

- **TRUE branch**
  - (a <=100)&&(a > 20)
    (a-10 > 30)
  - 75% (3/4)
- **FALSE branch**
  - (a <=100)&&(a > 20)
    (a-10 <= 30)
  - 25% (1/4)

国科大 ISCAS

```
int x;
@ ((x <= 50) && (x > 20))
{
  x = x - 10;
  if (x > 30)
    ... //TRUE branch
  else
    ... //FALSE branch
}
```

- TRUE branch: 1/3
- FALSE branch: 2/3

Constraints:

- (a <= 50)&&(a > 20)
  (a-10 > 30)
- (a <= 50)&&(a > 20)
  (a-10 <= 30)

静态分析技术

- 数据流分析
- 静态规则检查
- 采用符号执行的静态分析
- 程序切片

数据流分析

- 在某一点重用之前的表达式→可用表达式→前向Must分析
- 追踪数据的定义和使用→到达定值→前向May分析
- 提前计算表达式的值并存储其值→非常忙表达式→后向Must分析
- 消除无用代码→活跃变量→后向May分析

分析敏感性

- 针对多种关注点
  - 流，路径，字段，上下文，对象
- 可有不同的分析精度
  - xx敏感 vs. xx不敏感
- 寻求精度和效率的平衡
  - 轻量级？ 高精度？

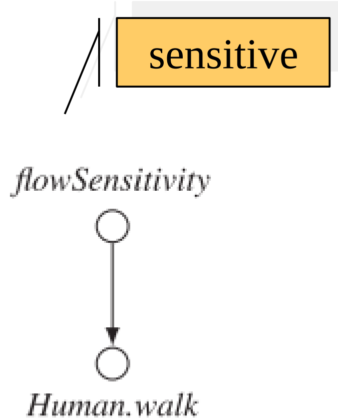
以函数调用图CG的构建为例

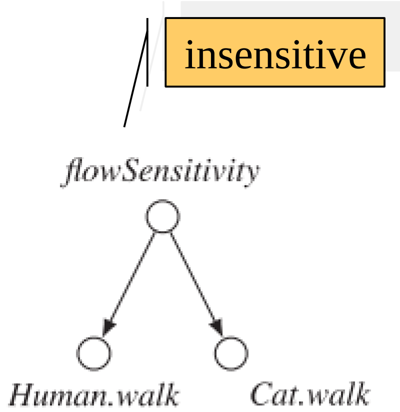
FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps.

流敏感 flow-sensitive

- 函数调用关系是怎样的？
- 考虑语句的执行顺序
- 过程内数据流分析--基础分析是流敏感的

```
public void flowSensitivity() {
   Animal a = new Human();
   a.walk();
   a = new Cat();
}
```





[1] Li Li, et al. **Static analysis of android apps: A systematic literature review.** IST 2017.

跨过程分析

- 代码内联（inlining）
- 跨过程数据流分析
- 函数摘要

切片技术

- **Static Slicing vs Dynamic Slicing**

- **The *backward slice* w.r.t variable *v* at program point p : The program subset that may influence the value of variable *v* at point *p*.**
- **The *forward slice* w.r.t variable *v* at program point *p* : The program subset that may be influenced by the value of variable *v* at point *p*.**

```c
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf("%d\n",sum);
    printf("%d\n",i);
}
```