Topics in Software Dynamic White-box Testing
Part 2: Data-flow Testing

[Reading assignment: Chapter 7, pp. 105-122 plus many things in slides that are not in the book …]

Data-Flow Testing

- **Data-flow testing** uses the control flowgraph to explore the unreasonable things that can happen to data (*i.e.*, anomalies).
- Consideration of data-flow anomalies leads to test path selection strategies that fill the gaps between complete path testing and branch or statement testing.

Data Object Categories

- (d) Defined, Created, Initialized
- (k) Killed, Undefined, Released
- (u) Used:
  - (c) Used in a calculation
  - (p) Used in a predicate

du Path Segments

- 定义-使用路径。
- 关于变量v的定义-使用路径（记做du-path）是PATHS(P) 中的路径，使得对某个v∈V,存在定义和使用节点DEF(v,m)和USE(v,n),使得m和n是该路径的最初和最终节点。

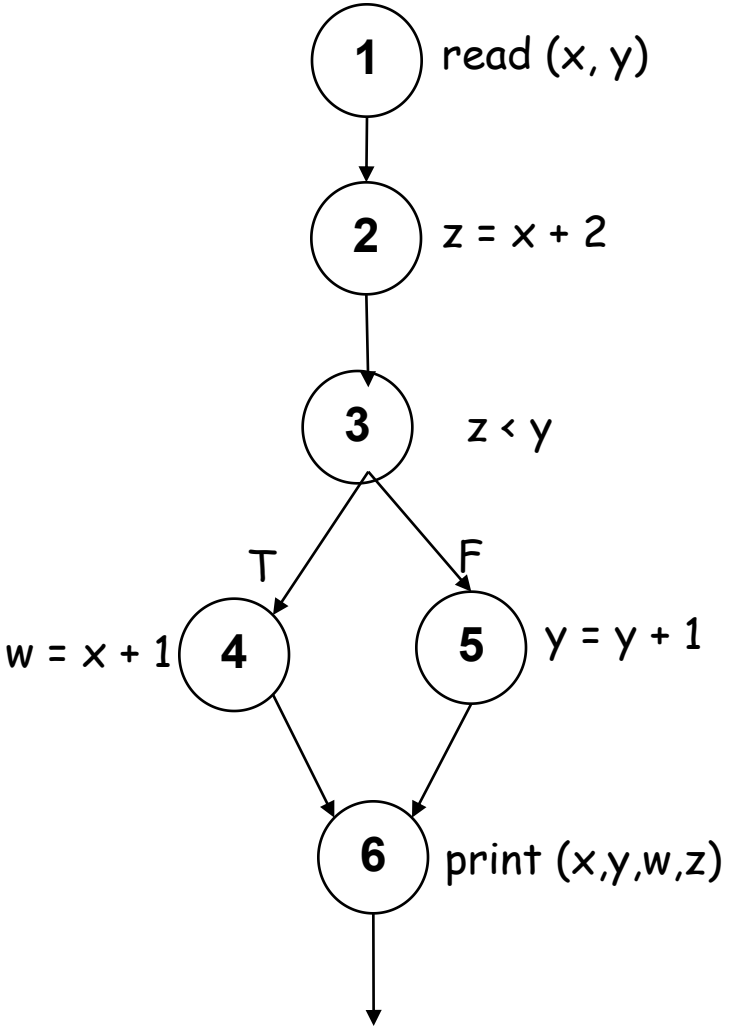
def-use Associations

- A def-use association is a triple (x, d, u,), where:

  x is a variable,
  d is a node containing a definition of x,
  u is either a statement or predicate node containing a use of x,

  and there is a sub-path in the flow graph from d to u with no other definition of x between d and u.

Example: Def-Use Associations



Some Def-Use Associations:

(x, 1, 2), (x, 1, 4), …

(y, 1, (3,t)), (y, 1, (3,f)), (y, 1, 5), …

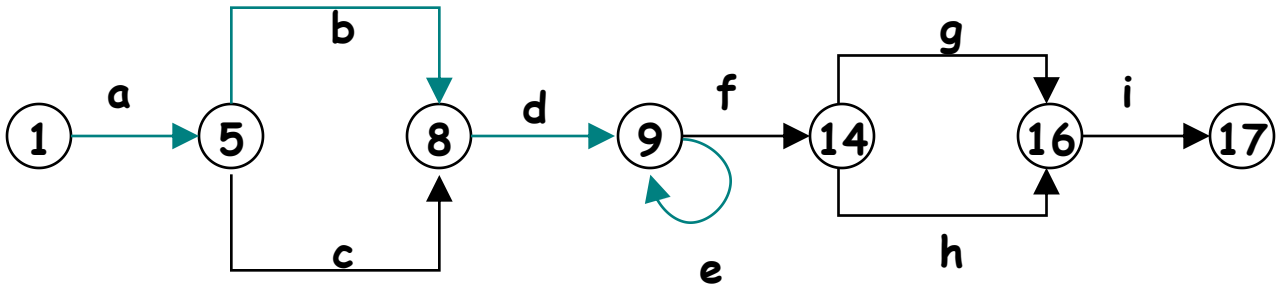
(z, 2, (3,t)),...

Data-Flow Testing Strategies

- All **du** Paths (ADUP)

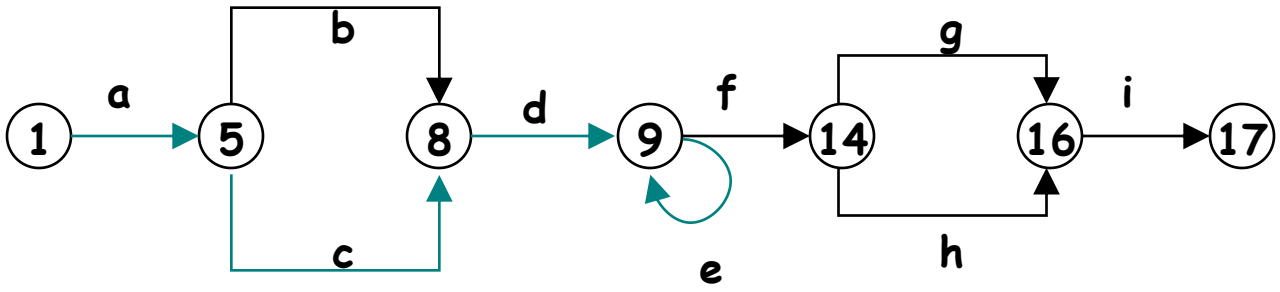
Example: pow(x,y) du-Path for Variable x

```
/* pow(x,y)
   This program computes x to the power of y, where x and y are integers.
    INPUT:     The x and y values.
    OUTPUT: x raised to the power of y is printed to stdout.
 */
1        void pow (int x, y)
2        {
3        float z;
4        int p;
5        if (y < 0)
6           p = 0 – y;
7        else p = y;
8        z = 1.0;
9        while (p != 0)
10          {
11          z = z * x;
12          p = p – 1;
13          }
14       if (y < 0)
15          z = 1.0 / z;
16       printf(z);
17       }
```



Example: pow(x,y) du-Path for Variable x

```
/* pow(x,y)
   This program computes x to the power of y, where x and y are integers.
    INPUT:     The x and y values.
    OUTPUT: x raised to the power of y is printed to stdout.
 */
1        void pow (int x, y)
2        {
3        float z;
4        int p;
5        if (y < 0)
6           p = 0 – y;
7        else p = y;
8        z = 1.0;
9        while (p != 0)
10          {
11          z = z * x;
12          p = p – 1;
13          }
14       if (y < 0)
15          z = 1.0 / z;
16       printf(z);
17       }
```



Summary

- Data are as important as code.
- Data-flow testing strategies span the gap between **all paths** and **branch** testing.填补路径和分支测试的缝隙