Announcements

- Opportunity for bonus points if you want to do a tutorial on a tool for next week (M,W). Pick a tool from the following categories:
  - Code coverage tools
  - Test generation tools
  - Other?
- You can do it individually or in teams of 2-3
- You have to validate the tool with me first! (Modified FIFO: people that need more points will be given priority)

Cpt S 422: Software Engineering Principles II

White-box testing – Part 2

Dr. Venera Arnaoudova

WASHINGTON STATE UNIVERSITY

Outline

- ✓ Control flow coverage
  - ✓ Statement, Edge, Condition, Path coverage
- ❑ Data flow coverage
  - ➢ Definitions-Usages of data
- ❑ Analyzing coverage data
- ❑ Integration testing
  - ➢ Coupling-based criteria
- ❑ Conclusions
  - ➢ Generating test data, Marick's Recommendations

Data Flow Analysis

- CFG paths that are significant for the data flow in the program
- Focuses on the assignment of values to variables and their uses
- Analyze occurrences of variables
- **Definition occurrence**: a value is bound to a variable
- **Use occurrence**: a value of a variable is referred
  - **Predicate use**: variable used to decide if a predicate is true
  - **Computational use**: compute a value for defining other variables or output value

FACTORIAL Example

```
1.public int factorial(int n){
2.   int i, result = 1;
3.   for (i=2; i<=n; i++) {
4.        result = result * i;
5.     }
6.   return result;
7.}
```

- Identify lines where `n` and `result` are defined and used.
- Identify pairs of lines where the definition can impact the use

Variable	Definition line	Use line
n	1	3
result	2	4
result	2	6
result	4	4
result	4	6

Data Flow Definitions on CFG

- Node n in CFG(P) is a **defining node** of the variable v in V, written as **DEF(v,n)**, iff the value of a variable v is defined in the statement corresponding to node n

- Node n in CFG(P) is a **usage node** of the variable v in V, written as **USE(v,n)**, iff the value of a variable v is used in the statement corresponding to node n

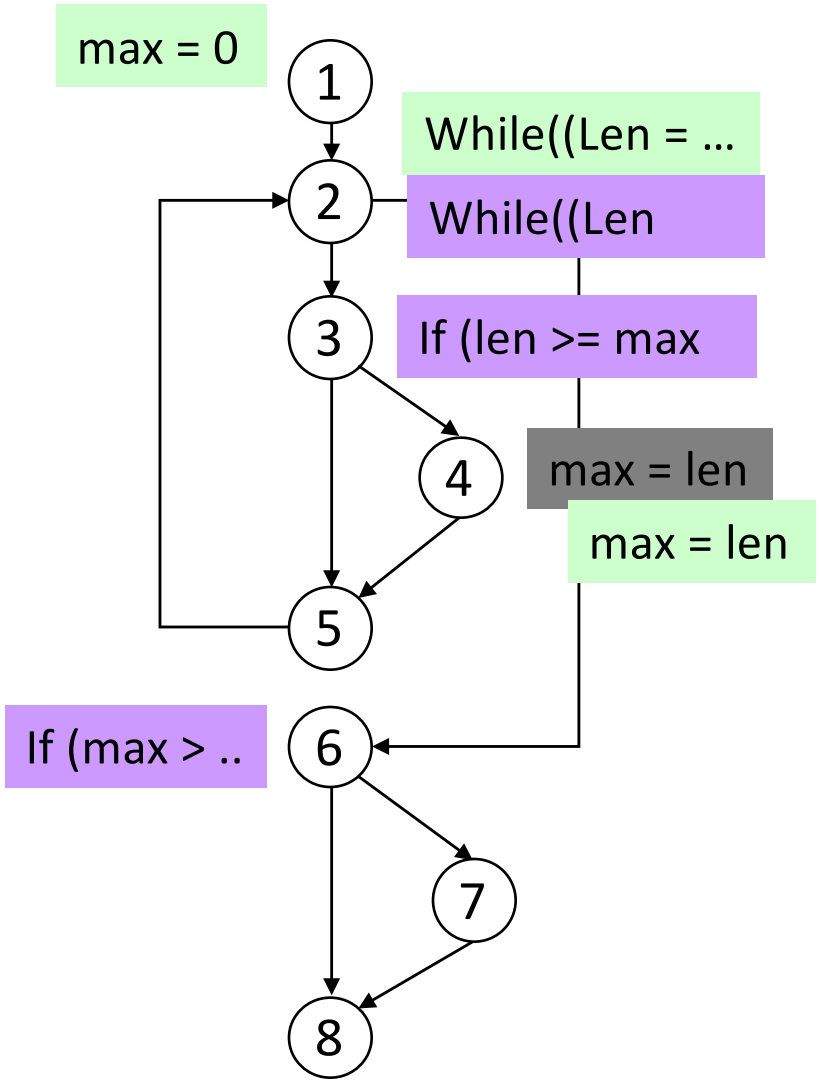
- A usage node USE(v,n) is a **predicate use** (denoted as **P-use**) iff the statement n is a predicate statement, otherwise USE(v,n) is a **computation use** (denoted as **C-use**)

Example

```
main() { /* find longest line */
  int len;
  extern int max;
  extern char save[];
  max = 0;
  while ( (len = getline()) > 0 ) {
    if (len >= max) {
      max = len;
      copy();
    }
  }
  if (max > 0) /* there was a line */
    printf("%s", save);
}
```

- Create a CFG
- For `max` and `len` identify all:
  - DEFs
  - P-Uses
  - C-Uses

Simple Example (CFG)



v = …	Definition DEF(v, n)
v >= …	P-use USE(v,n)
… = …v …	C-use USE(v,n)

- DEF(max, 1)
- DEF(len, 2)
- DEF(max, 4)
- C-USE(len, 4)
- P-USE(len, 2)
- P-USE(len, 3)
- P-USE(max, 3)
- P-USE(max, 6)

Data Flow Definitions on CFG (cont.)

- A **definition-use (sub)path** with respect to a variable v (denoted **du-path**) is a (sub)path in PATHS(P) such that, for some v in V, there are define and usage nodes DEF(v, m) and USE(v, n) such that m and n are initial and final nodes of the (sub)path.

- A **definition-clear (sub)path** with respect to a variable v (denoted **dc-path**) is a du-path in PATH(P) with initial and final nodes DEF(v, m) and USE(v, n) such that no other node in the path is a defining node of v.

Criteria Formal Definitions I

- The set T satisfies the **all-Definitions** criterion for the program P iff for every variable v in V, T contains a definition clear path from every defining node of v to a use (p-use or c-use) of v.

- The set T satisfies the **all-Uses** criterion for the program P iff for every variable v in V, T contains a definition clear path from every defining node of v to every use (p-use and c-use) of v.

- The set T satisfies the **all-P-Uses/Some C-Uses** criterion for the program P iff for every variable v in V, T contains a definition clear path from every defining node of v to every predicate use of v, and if a definition of v has no P-Uses, there is a definition-clear path to at least one computation use.

Criteria Formal Definitions II

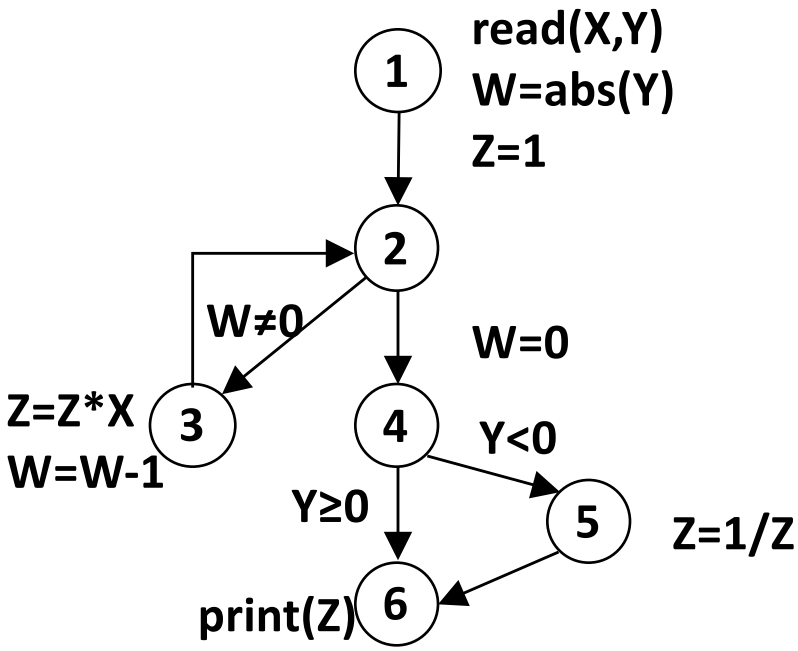
- The set T satisfies the **all-C-Uses/Some P-Uses** criterion for the program P iff for every variable v in V, T contains at least one definition-clear path from every defining node of v to every computation use of v, and if a definition of v has no C-Uses, there is a definition-clear path to at least one predicate use.

- The set T satisfies the **all-DU-Paths** criterion for the program P iff for every variable v in V, T contains all definition-clear paths from every defining node of v to every reachable use of v, and that these paths are either single loops traversals, or they are cycle free.

POWER Example

node i	def(i)	c-use(i)	edge(i,j)	p-use(i,j)
1	X, Y, W, Z	Y	(1,2)	
2			(2,3) (2,4)	W W
3	W, Z	X, W, Z	(3,2)	
4			(4,5) (4,6)	Y Y
5	Z	Z	(5,6)	
6		Z		

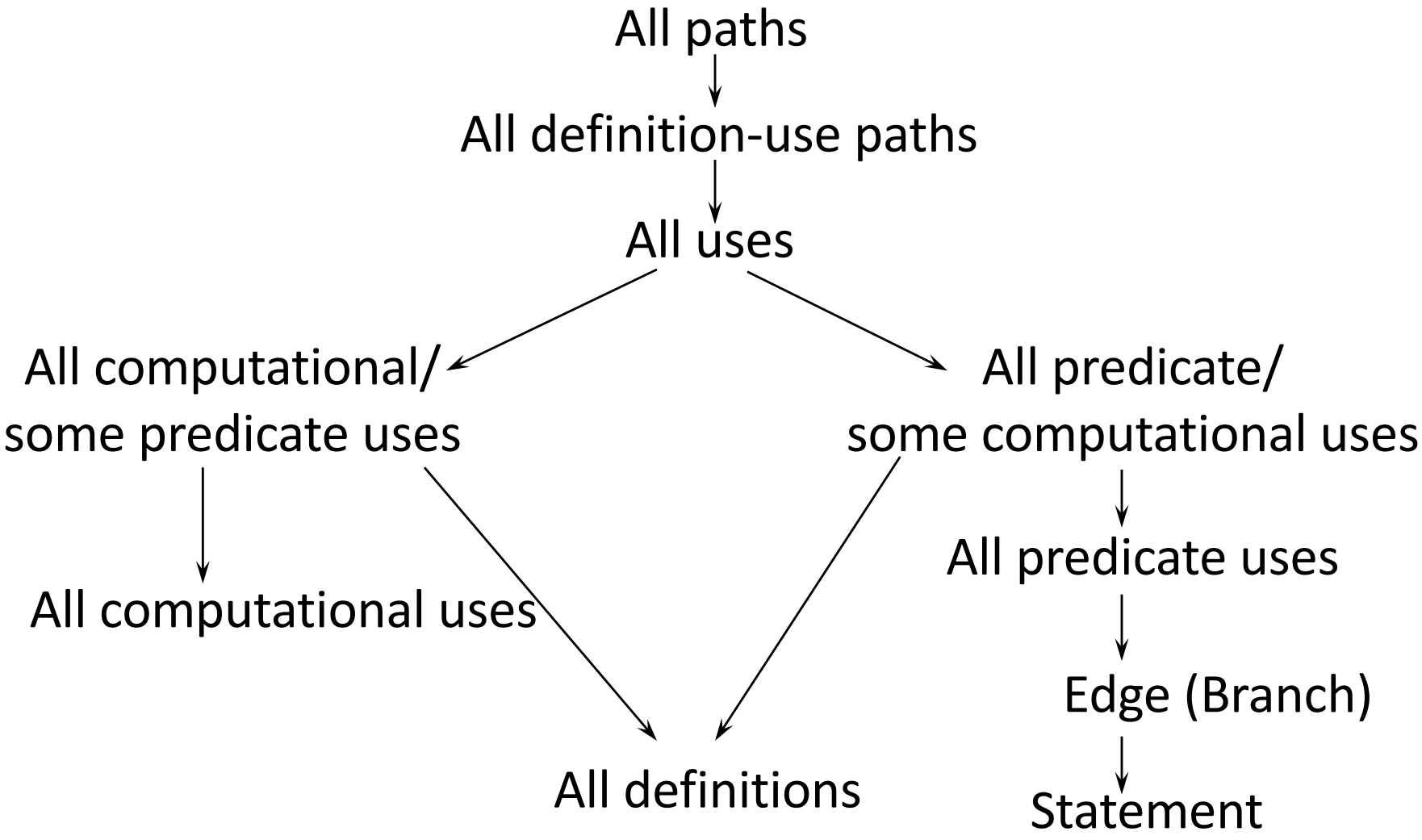
node i	dcu(v,i)	dpu(v,i)
1	dcu(X,1) = {3} dcu(Z,1) = {3,6} dcu(W,1) = {3}	dpu(Y,1) = {(4,5),(4,6)} dpu(W,1) = {(2,3),(2,4)}
3	dcu(W,3) = {3} dcu(Z,3) = {3,5,6}	dpu(W,3) = {(2,3),(2,4)}
5	dcu(Z,5) = {6}	



Discussion

- Generates test data according to the way data is manipulated in the program
- Help define intermediary criteria between all-edges testing (possibly too weak) and all-paths testing (often impossible)
- Needs effective tool support

Subsumption hierarchy



Measuring Code Coverage

- One advantage of structural criteria is that their coverage can be measured *automatically*
- To control testing progress
- To help fix targets for testers
- High coverage is not a guarantee of fault-free software, just an element of information to increase our confidence