White-Box Testing

Spring, 2023

Yi Xiang xiangyi@scut.edu.cn

Contents

- Data Flow Testing
  - DU-Pair Testing

Dataflow Testing --- Motivation

- Testing All-Nodes and All-Edges in a control flow graph may miss significant test cases!

- Testing All-Paths in a control flow graph is often too time consuming !

- Can we select a subset of these paths that will reveal the most faults?!

- **Dataflow Testing**

  focuses on the points at which variables receive values and the points at which these values are used!

Dataflow Testing --- Motivation

- A program accepts inputs, performs computations, assigns new values to variables, and returns results.

- One can visualize of "flow" of data values from one statement to another. A data value produced in one statement is expected to be used later.

- Motivations of data flow testing
  - The memory location for a variable is accessed in a "desirable" way.
  - Verify the correctness of data values "defined" (i.e. generated) -observe that all the "uses" of the value produce the desired results.
  - Find data flow anomalies

Dataflow Analysis

- Data flow analysis is in part based concordance analysis such as that shown below – the result is a variable cross reference Table.

```
 18  beta ← 2
 25  alpha ← 3 × gamma + 1
 51  gamma ← gamma + alpha - beta
123  beta ← beta + 2 × alpha
124  beta ← gamma + beta + 1
```

	Defined	Used
alpha	25	51, 123
beta	18, 123, 124	51, 123, 124
gamma	51	25, 51, 124

Dataflow Analysis

- Can reveal interesting bugs （data flow anomalies）！

1. A variable that is defined but never used
2. A variable that is used but never defined
3. A variable that is defined twice before it is used
4. Sending a modifier message to an object more than once between accesses
5. Deallocating a variable before it used
   - Container problem – deallocating container loses references to items in the container, memory leak

Dataflow Testing

- (Static Analysis) The bugs can be found from a cross-reference table.
- (Dynamic Testing) Paths from the definition of a variable to its use are more likely to contain bugs.
  - Generate test data that follows the pattern of data definition & use through the program.
  - The objective is to identify and classify all occurrences of variables in a program and for each variable generate test data so that all definitions and uses are exercised.

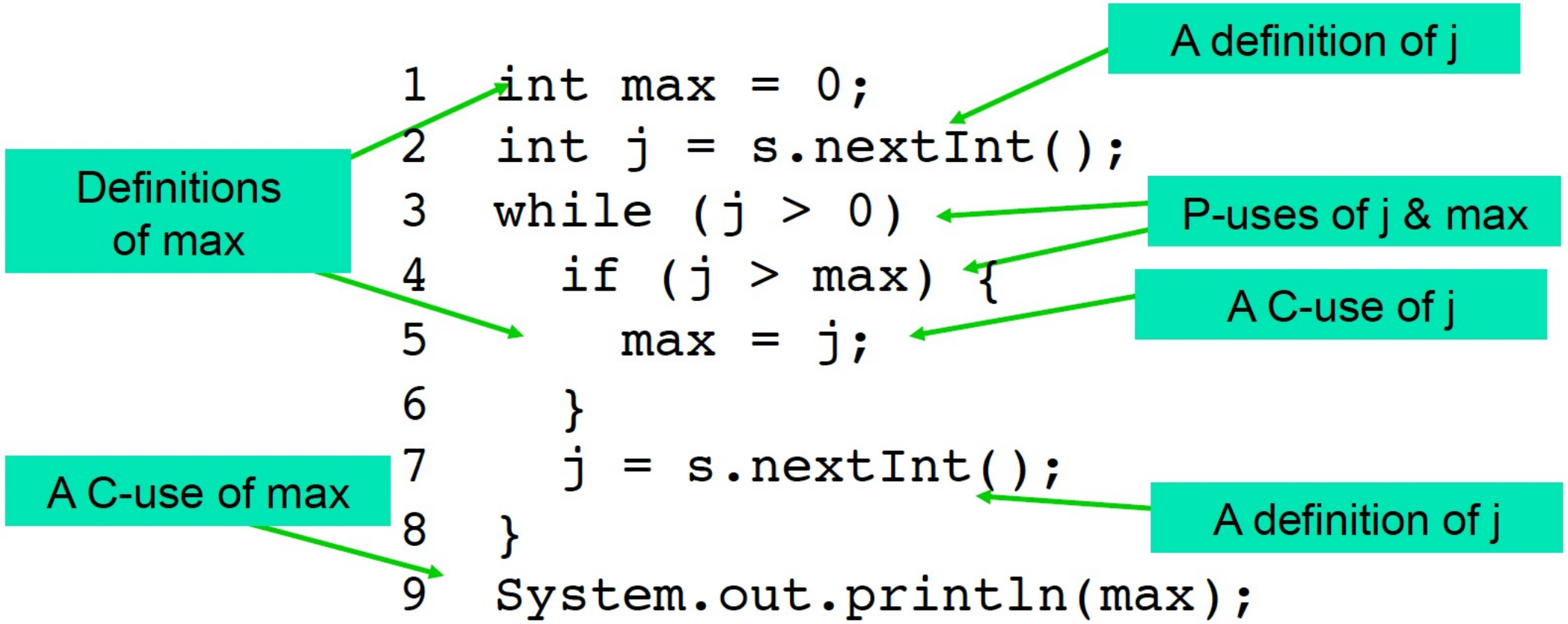
Dataflow Testing --- Definitions

- DEF(v, n) – if the value of v is defined at the statement n (or node n)
  - Input, assignment, procedure calls

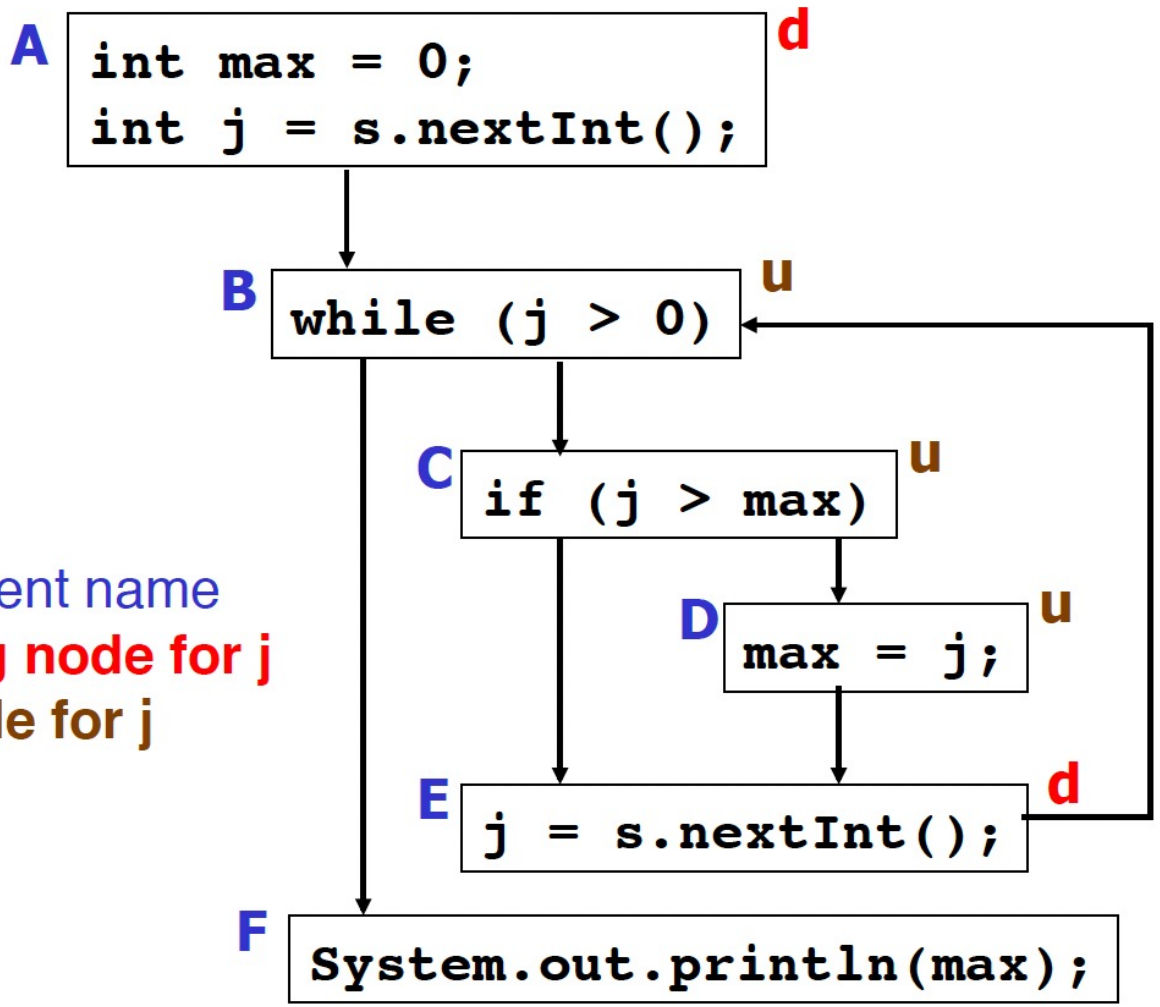
- USE(v, n) – if the value of v is used at the statement n (or node n)
  - Output, assignment, conditionals
  - P-use, if variable v appears in a predicate expression
  - C-use, if variable v appears in a computation

- A definition-use path, du-path, with respect to a variable v
  - A sub-path from a defining statement(node) for v to a usage statement(node) for v and the path is definition clear with no other defining statement(node) for v .

Dataflow Testing --- Max Program



Dataflow Testing --- Max Program



Legend
A..F Segment name
d defining node for j
u use node for j

du-paths j

A B
A B C
A B C D
E B
E B C
E B C D

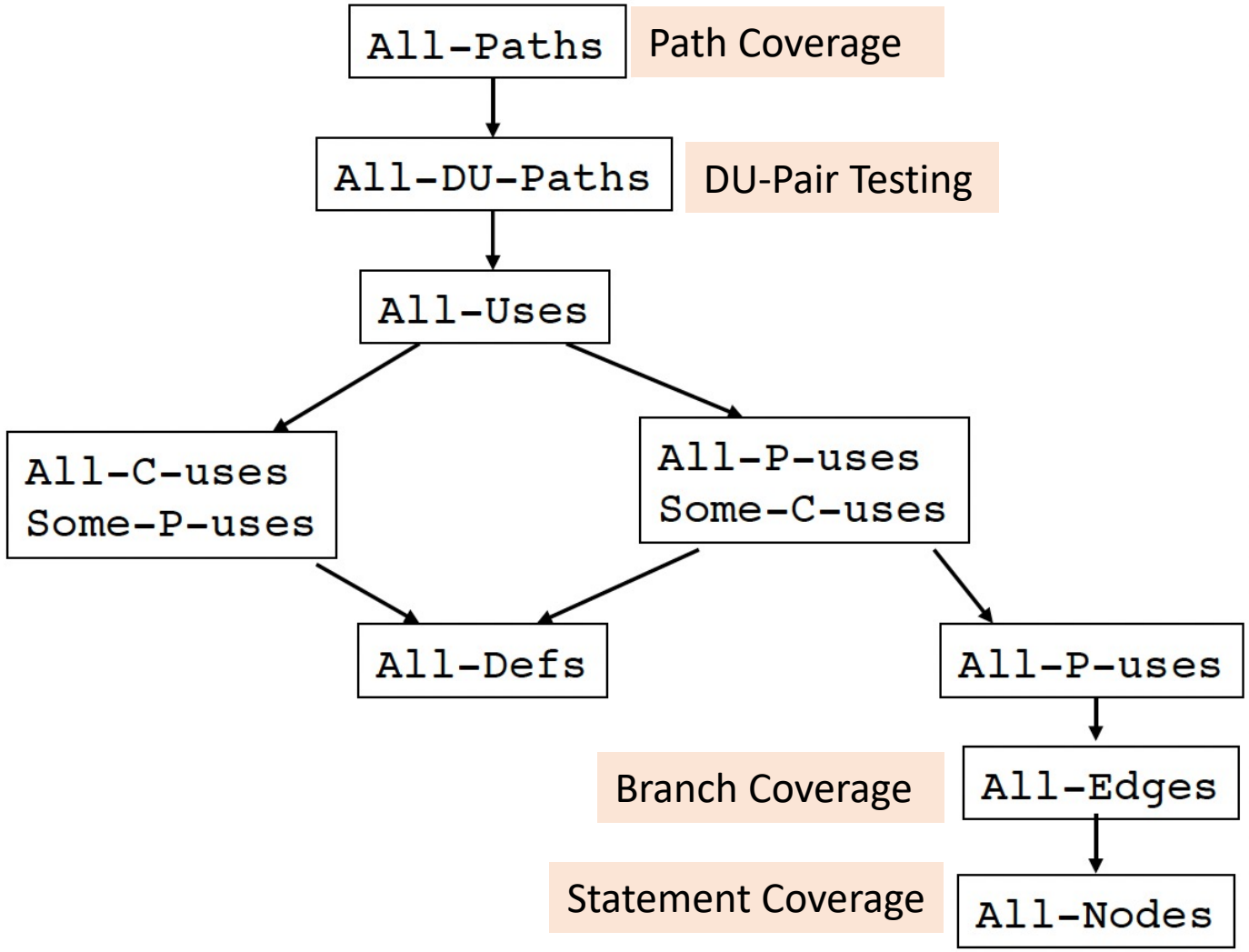
du-paths max

A B F
A B C
D E B C
D E B F

Dataflow Testing --- Coverage Criteria

- ADUP – (All-DU-Paths)
  - One of the strongest data-flow testing strategies.

- ADUP requires that every du path from every definition of every variable to every use of that definition be exercised under some testcase.



Dataflow Testing Example --- Factorial (from textbook)

```
(1) int factorial(int n) {
(2)     int result=1;
(3)
(4)     for (i=2; i<=n; i++) {
(5)         result = result * i;
(6)     }
(7)     return result;
(8) }
```

DU-Pair
For Variable Result

d-u pair	d	u
1	2	5
2	2	7
3	5	5
4	5	7

To generate test data to exercise these pairs

- input variable n =3 would exercise pairs 1, 3 and 4
- input variable n =1 would exercise pair 2

Airline Seat reservation Example

Program Code:

```
(1) public static Boolean seatsAvailable(
        int freeSeats, int seatsRequired)
(2) {
(3)     boolean rv=false;
(4)     if ( (freeSets>=0) && (seatsRequired>=1) &&
             (seatsRequired<=freeSeats) )
(5)         rv=true
(6)     return rv;
(7) }
```

Variables

- freeSeats
- seatsRequired
- rv

- The principle in du-pair testing is to execute each path between the definition of the value in a variable and its subsequent use.

Airline Seat reservation Example

Program Code:

```
(1) public static Boolean seatsAvailable(
       int freeSeats, int seatsRequired)
(2) {
(3)    boolean rv=false;
(4)    if ( (freeSets>=0) && (seatsRequired>=1) &&
          (seatsRequired<=freeSeats) )
(5)       rv=true
(6)    return rv;
(7) }
```

du-pairs for seatsRequired

d-u pair	D	U
2	1	4

du-pairs for freeSeats

d-u pair	D	U
1	1	4

du-pairs for rv

d-u pair	D	U
3	3	6
4	5	6

Airline Seat reservation Example

Test Data

Test No.	Test Cases/Pairs Covered	Inputs		Expected Outputs
		freeSeats	*seatsRequired*	*Return Value*
1	1, 2, 4	50	50	True
2	1, 2, 3	50	75	False

Dataflow Testing Example --- Grade

```
1   public static String Grade (int exam, int
    course) {
2   String result="null";
3   long average;
4   average = Math.round((exam+course)/2);
5   if ( (exam<0) || (exam>100) || (course<0) ||
    (course>100) )
6   result="Marks out of range";
7   else {
8   if ( (exam<50) || (course<50)) {
9   result="Fail";
10  }
11  else if (exam < 60) {
12  result="Pass,C";
13  }
14  else if ( average >= 70) {
15  result="Pass,A";
16  }
17  else {
18  result="Pass,B";
19  }
20  }
21  return result;
22  }
```

Variables:

- exam
- course
- average
- result

Dataflow Testing Example --- Grade

```
1   public static String Grade (int exam, int course) {
2   String result="null";
3   long average;
4   average = Math.round((exam+course)/2);
5   if ( (exam<0) || (exam>100) || (course<0) || (course>100) )
6   result="Marks out of range";
7   else {
8   if ( (exam<50) || (course<50)) {
9   result="Fail";
10  }
11  else if (exam < 60) {
12  result="Pass,C";
13  }
14  else if ( average >= 70) {
15  result="Pass,A";
16  }
17  else {
18  result="Pass,B";
19  }
20  }
21  return result;
22  }
```

DU-Pairs for exam

DU-Pair	D	U
1	1	4
2	1	5
3	1	8
4	1	11

DU-Pairs for average

DU-Pair	D	U
8	4	14

DU-Pairs for course

DU-Pair	D	U
5	1	4
6	1	5
7	1	8

DU-Pairs for result

DU-Pair	D	U
9	6	21
10	9	21
11	12	21
12	15	21
13	18	21

Dataflow Testing Example --- Grade

Test Data

Test No.	Test Cases/DU-Pairs Covered	Inputs		Expected Outputs
		exam	*course*	*Result*
1	1, 2, 5, 6, 9	-1	1	Marks out of Range
2	1, 2, 3, 5, 6, 7, 10	40	50	Fail
3	1, 2, 3, 4, 5, 6, 7, 11	55	50	Pass, C
4	1, 2, 3, 4, 5, 6, 7, 8, 12	90	50	Pass, A,
5	1, 2, 3, 4, 5, 6, 7, 8, 13	80	50	Pass, B

Dataflow Testing --- Comment

- The principle in du-pair testing is to execute each path between the definition of the value in a variable and its subsequent use.
  - A definition is the assignment of a value to a variable, including assignment at function entry.
  - A use is the reading of the value from a variable.
  - Increment and decrement operations cause a use followed by a definition.

- DU-Pair testing provides comprehensive testing of all the Definition-Use paths in a program, but generating the test data can be a very time consuming exercise