Data flow graph coverage criteria: Applied to test code

.

Graph coverage criteria: Overview

- Model software artifacts as graphs and look at coverage criteria over graphs.
  - Structural coverage criteria.
  - Data flow coverage criteria.
  - Coverage criteria over call graphs.
- Focus of this lecture: Understand the notion of data flow coverage in graphs.

Outline

- We consider CFGs augmented with data and understand how the three data flow criteria come to use for testing code.
  - Data could be defined and used in nodes in the CFG.
  - Data is used in the edges of the CFG.
- Data flow criteria test if every definition reaches its use.

Recap: Def-use paths

- A du-path with respect to a variable v is a simple path that is def-clear from a def of v to a use of v.
  - du-paths are parameterized by v.
  - They need to be simple paths.
  - There may be intervening uses on the path.
- $du(n_i, n_j, v)$: The set of du-paths from n_i to n_j for variable v.
- $du(n_i, v)$: The set of du-paths that start at n_i for variable v.

Recap: Data flow criteria

There are three common data flow criteria:

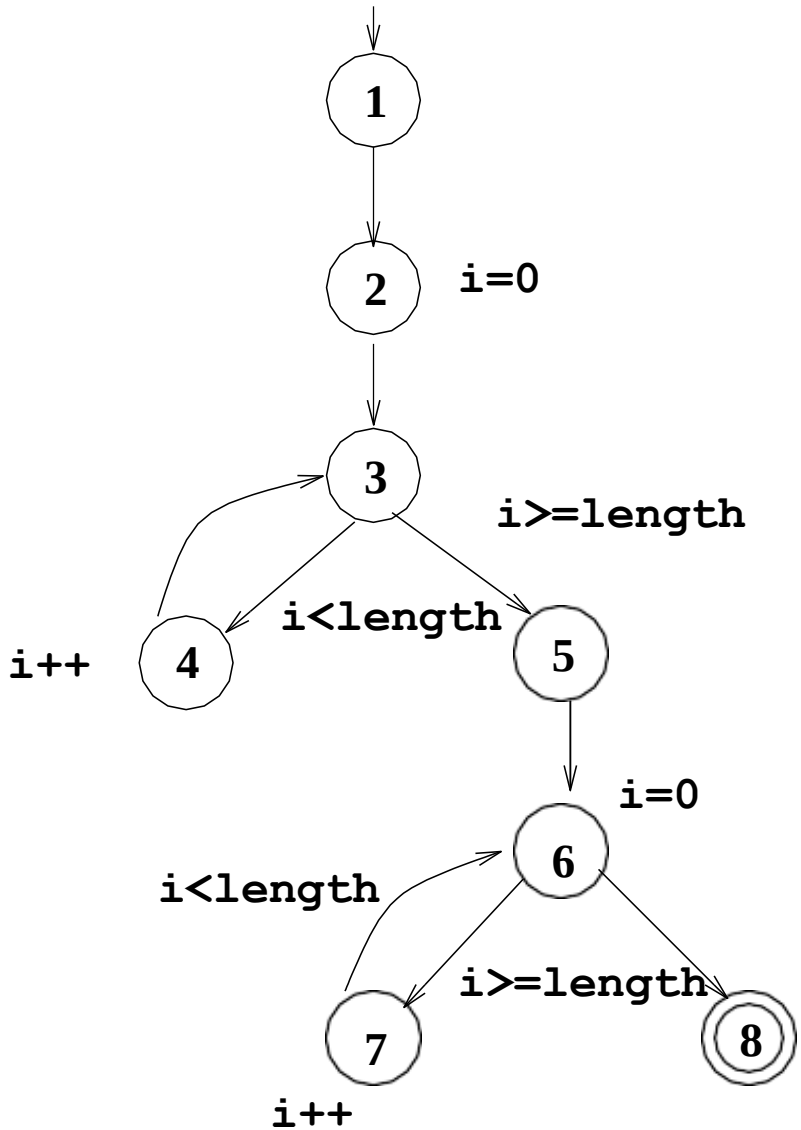
- All defs coverage: Each def reaches *at least one use*.
- All uses coverage: Each def reaches *all possible uses*.
- All du-paths coverage: Each def reaches all possible uses through *all possible du-paths*.

To get test paths to satisfy these criteria, we can assume best effort touring, i.e., side trips are allowed as long as they are def-clear.

Re-visiting Example program: Statistics

```
public static void computeStats (int [] numbers)
{  int length = numbers.length;
   double med, var, sd, mean, sum, varsum;
   sum = 0.0;
   for(int i=0; i<length; i++)
   { sum += numbers[i]; }
   med=numbers[length/2];
   mean=sum/(double)length;
   varsum = 0.0;
   for(int i=0; i<length; i++)
   {  varsum = varsum+((numbers[i]-mean)*(numbers[i]-mean)); }
   var = varsum/(length-1);
   sd = Math.sqrt(var);
   System.out.println ("mean:" + mean);
   System.out.println ("median:" + med);
   System.out.println ("variance:" + var);
   System.out.println ("standard deviation:" + sd);
}
```

Re-visiting: CFG for Statistics program



Statistics program: Definitions and uses at nodes

Node	Defs	Uses
1	{numbers,sum,length}	{numbers}
2	{i}	
3		
4	{sum,i}	{numbers,i,sum}
5	{med,mean,varsum,i}	{numbers,length,sum}
6		
7	{varsum,i}	{varsum,numbers,i,mean}
8	{var,sd}	{varsum,length,var,mean,med,sd}

Statistics program: Uses at edges

Edge	Use
(1,2)	
(2,3)	
(3,4)	{i,length}
(4,3)	
(3,5)	{i,length}
(5,6)	
(6,7)	{i,length}
(7,6)	
(6,8)	{i,length}

Statistics program: du-pairs

Variable	du pairs
numbers	(1,4),(1,5),(1,7)
length	(1,5),(1,8),(1,(3,4)),(1,(3,5)), (1,(6,7)),(1,(6,8))
med	(5,8)
var	(8,8)
sd	(8,8)
mean	(5,7),(5,8)

Statistics program: du-pairs, contd.

Variable	du pairs
sum	(1,4),(1,5),(4,4),(4,5)
varsum	(5,7),(5,8),(7,7),(7,8)
i	(2,4),(2,(3,4)),(2,(3,5)),(2,7),(2,(6,7)),(2,(6,8))
	(4,4),(4,(3,4)),(4,(3,5)),(4,7),(4,(6,7)),(4,(6,8))
	(5,7),(5,(6,7)),(5,(6,8))
	(7,7),(7,(6,7)),(7,(6,8))

Statistics program: DU paths

Variable	DU pairs	DU paths
number	(1,4),(1,5),(1,7)	[1,2,3,4],[1,2,3,5],[1,2,3,5,6,7]
length	(1,5),(1,8)	[1,2,3,5],[1,2,3,5,6,8]
	(1,(3,4)),(1,(3,5)	[1,2,3,4],[1,2,3,5]
	(1,(6,7)),(1,(6,8))	[1,2,3,5,6,7],[1,2,3,5,6,8]
med	(5,8)	[5,6,8]
var	(8,8)	No path needed
sd	(8,8)	No path needed
mean	(1,4),(1,5)	[1,2,3,4],[1,2,3,5]
	(4,4),(4,5)	[4,3,4],[4,3,5]

Statistics program: DU paths

Variable	DU pairs	DU paths
mean	(5,7),(5,8)	[5,6,7],[5,6,8]
varsum	(5,7),(5,8)	[5,6,7],5,6,8]
	(7,7),(7,8)	[7,6,7],[7,6,8]
i	(2,4),(2,(3,4)),(2,(3,5))	[2,3,4],[2,3,4],[2,3,5]
	(4,4),(4,(3,4)),(4,(3,5))	[4,3,4],[4,3,4],[4,3,5]
	(5,7),(5,(6,7)),(5,(6,8))	[5,6,7],[5,6,7],[5,6,8]
	(7,7),(7,(6,7)),(7,(6,8))	[7,6,7],[7,6,7],[7,6,8]

Statistics program: Du-paths without duplicates

There are 38 du-paths for Stats, but only 12 of them are unique.

- Paths that skip a loop:
  - Four paths: [1,2,3,5], [1,2,3,5,6,8], [2,3,5], [5,6,8]
- Paths that require at least one iteration of a loop:
  - Six paths: [1,2,3,4], [1,2,3,4,6,7], [4,3,4], [7,6,7], [4,3,5], [7,6,8]
- Paths that require at least two iterations of a loop:
  - Two paths: [4,3,4], [7,6,7]

Statistics program: Test case #1

- Test case: numbers = (44), length = 1.
- Test path: [1,2,3,4,3,5,6,7,6,8].
- Additional DU paths covered (without side trips): [1,2,3,4], [2,3,4], [4,3,5], [5,6,7], [7,6,8].
- Note: All these require at least one iteration of the two loops.

Statistics program: Test case #2

- Test case: numbers = (2,10,15), length = 3.
- Test path: [1,2,3,4,3,4,3,4,3,5,6,7,6,7,6,7,6,8].
- Additional DU paths covered (without side trips): [4,3,4], [7,6,7].
- Note: All these require at least two iterations of both the loops.

Statistics program: Test case #3

- Only loop coverage criteria pending needs to skip the loops.
- Need test case with array of length 0.
- But, the method fails with index out of bound exception: A fault is found.

Data flow coverage criteria

- Measuring actual coverage achieved by the various data flow coverage criteria is an undecidable problem as we have to work with graphs that contain data information.
- Several studies exist for measuring data flow coverage.
- It has been reported that the number of bugs detected by putting the criteria of 90% data coverage were twice as high as those detected by 90% branch coverage criteria.

Reference material

- A latest survey on data flow testing techniques. Covers several topics not introduced in these lectures also.
  T. Su, K. Wu, W. Miao, G. Pu, J. He, Y. Chen and Z. Su, A Survey on Data-Flow Testing, ACM Computing Surveys, 50(1), April 2017.
- Data flow testing criteria as discussed in these lectures.
  - S. Rapps and E. J. Weyuker, Data flow analysis techniques for test data selection. In Proceedings of the 6th International Conference on Software Engineering (ICSE'82). IEEE Computer Society Press, Los Alamitos, CA, 272-278.
  - S. Rapps and E. J. Weyuker, Selecting software test data using data flow information. IEEE Transactions Software Engg. 11 (4), 367-375, 1985.

Credits

Part of the material used in these slides are derived from the presentations of the book Introduction to Software Testing, by Paul Ammann and Jeff Offutt.

COURTESY:MEENAKSHI D’SOUZA,IIIT ,BANGLORE