Algorithms: Data Flow Graph Coverage Criteria

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: Data flow coverage criteria over graphs.

Data flow coverage criteria

- Data flow coverage criteria will be defined as sets of du-paths.
- Such du-paths will be *grouped* to define the data flow coverage criteria.
- Data flow coverage criteria defined using du-paths will check for definitions of variables reaching their uses in different ways.

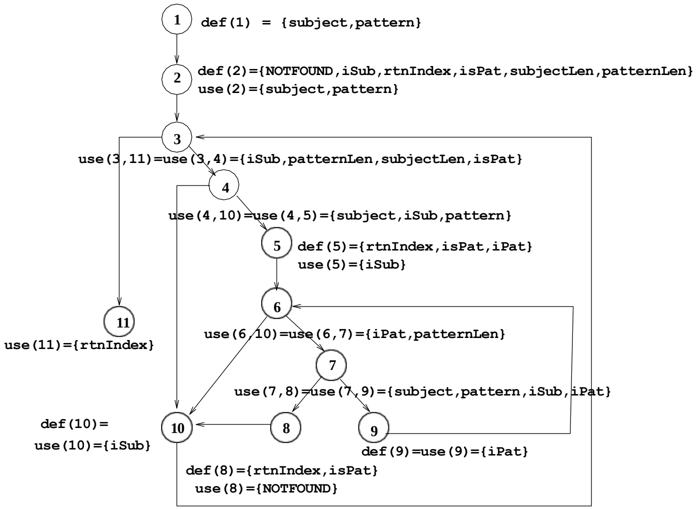
Grouping du-paths: As per definitions

- Grouping according to definitions: Consider all du-paths with respect to a given variable defined in a given node.
- The def-path set $du(n_i, v)$ is the set of du-paths with respect to variable v that start at node n_i.
- For large programs, the number of def-path sets can be large.
- We *do not* group du-paths by uses.

Grouping du-paths: As per definitions and uses

- Grouping du-paths as per definitions and uses allows definitions to flow to uses.
- A def-pair set, $du(n_i, n_j, v)$ is the set of du-paths with respect to variable v that start at node n_i and end at node n_j.
- A def-pair set collects together all the (simple) ways to get from a given definition to a given use.
- A def-pair set for a def at node n_i is the union of all the def-path sets for that def. $du(n_i, v) = \cup_{n_j} du(n_i, n_j, v)$.

Pattern Matching example


1
def(1) = {subject,pattern}
2
def(2)={NOTFOUND,iSub,rtnIndex,isPat,subjectLen,patternLen}
use(2)={subject,pattern}
3
use(3,11)=use(3,4)={iSub,patternLen,subjectLen,isPat}
4
use(4,10)=use(4,5)={subject,iSub,pattern}
5
def(5)={rtnIndex,isPat,iPat}
use(5)={iSub}
6
11
use(11)={rtnIndex}
use(6,10)=use(6,7)={iPat,patternLen}
7
use(7,8)=use(7,9)={subject,pattern,iSub,iPat}
def(10)=
use(10)={iSub}
10
8
9
def(9)=use(9)={iPat}
def(8)={rtnIndex,isPat}
use(8)={NOTFOUND}

Def-paths and def-pairs: Example

- In the pattern matching example, there is a definition of `iSub` at node 10.
- The following is the du-path set with respect to `iSub` at node 10: *du*(10, `iSub`) = {[10,3,4],[10,3,4,5],[10,3,4,5,6,7,8], [10,3,4,5,6,7,9],[10,3,4,5,6,10],[10,3,4,5,6,7,8,10],[10,3,4,10]}
- The above def-path set can be split into the following def-pair sets:
  - *du*(10,4,iSub)={[10,3,4]}.
  - *du*(10,5,iSub)={[10,3,4,5]}.
  - *du*(10,8,iSub)={[10,3,4,5,6,7,8]}.
  - *du*(10,9,iSub)={[10,3,4,5,6,7,9]}.
  - *du*(10,10,iSub)={[10,3,4,5,6,10],[10,3,4,5,6,7,8,10], [10,3,4,10]}.

Towards defining data-flow coverage criteria

- Like structural coverage criteria, we need to consider tours and side-trips for data flow coverage criteria too.
- This is to make the coverage criteria feasible.
- A test path p is said to du tour a sub-path d with respect to v if p tours d and the portion of p to which d corresponds is def-clear with respect to v.
- We can allow def-clear side trips with respect to v while touring a du-path, if needed.

Data flow criteria

There are three common data flow criteria:

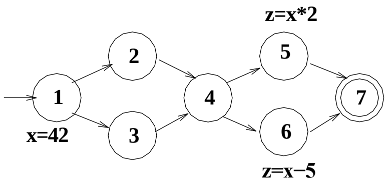
- TR: Each def reaches *at least one use*.
- TR: Each def reaches *all possible uses*.
- TR: Each def reaches all possible uses through *all possible du-paths*.

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.

Data flow criteria

- All-Defs Coverage: For each def-path set $S = du(n, v)$, TR contains at least one path d in S.
- All-Uses Coverage: For each def-pair set $S = du(n_i, n_j, v)$, TR contains at least one path d in S.
  Since a def-pair set $du(n_i, n_j, v)$ represents all def-clear simple paths from a def of v at n_i to a use of v at n_j, all-uses requires us to tour at least one path for every def-use pair.
- All-du-Paths Coverage: For each def-pair set $S = du(n_i, n_j, v)$, TR contains every path d in S.

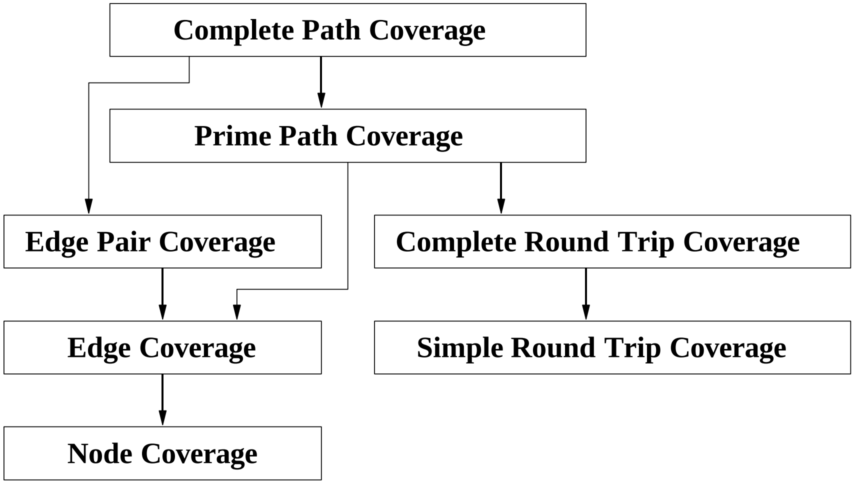
Data flow coverage criteria: A simple example



- All defs for x: Test path is [1,2,4,6,7]
- All uses for x: Test paths are [1,2,4,5,7] and [1,2,4,6,7].
- All du-paths for x: Test paths are [1,2,4,5,7], [1,3,4,5,7], [1,2,4,6,7] and [1,3,4,6,7].

Graph structural coverage criteria: subsumption

Recap: Subsumption of structural graph coverage criteria



Data flow coverage criteria subsumption

Subsumption results for data flow coverage criteria are based on three assumptions:

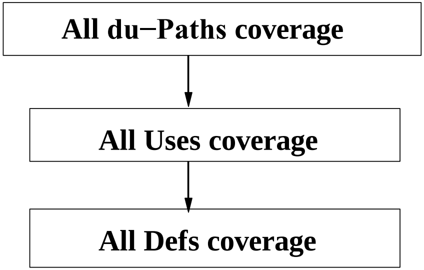
- Every use is preceded by a def.
- Every def reaches at least one use.
- For every node with multiple out-going edges, at least one variable is used on each out edge, and the same variables are used on each out edge.

Data flow coverage criteria subsumption

- If we satisfy all-uses criteria, due to our assumption, we would have ensured that every def was used.
- If we satisfy all-du paths criteria, we would have ensured that every def reaches every possible use.
    - Hence, all uses is also satisfied.

Graph data flow coverage criteria: subsumption

Subsumption of data flow graph coverage criteria

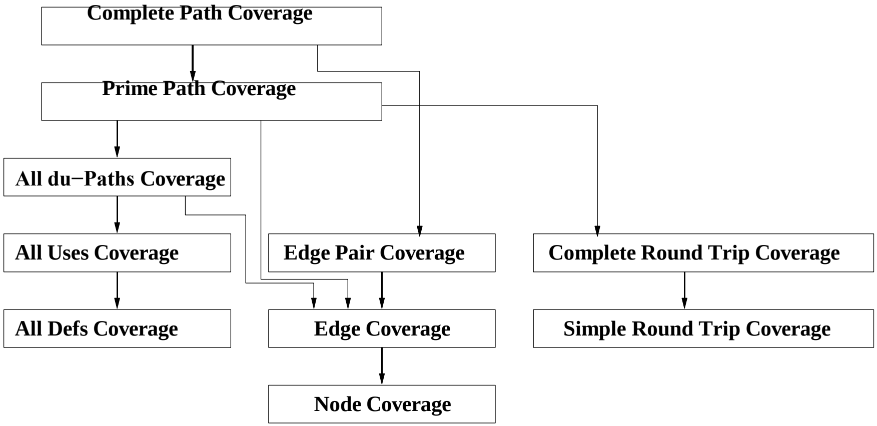


Graph criteria: subsumption

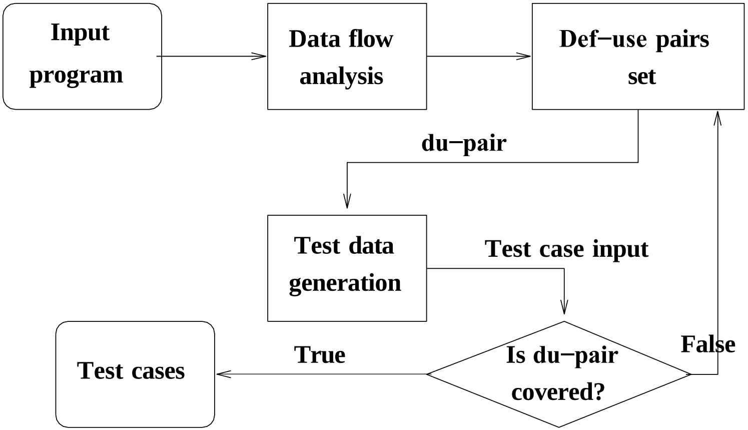
- Each edge of the graph is based on the satisfaction of some predicate. So, each edge has at least one use.
- All-du-paths coverage will subsume edge coverage.
- Each du-path is a simple path, so prime path coverage subsumes all-du-paths coverage.

Graph criteria: subsumption

Graph coverage criteria subsumption



Data flow testing: Process



Data flow testing: Algorithms

- There are several algorithms for data flow testing that cover each of the stages outlined in the previous slide.
- Identifying du-pairs use program analysis tools, there are several research papers in this area.
  - The number of du-pairs can be very large.
  - Some of the du-pairs can be infeasible. Identifying infeasible du-pairs is usually undecidable.
- Several approaches for test data generation exist:
  - Explicit search, random testing, symbolic execution, model checking etc.

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