SYSC 4101 / 5105

Graph Criteria (Part II)

Graph Testing—Two Families of Criteria

- Control Flow Criteria
  - Only consider the flow of nodes and edges
  - Seven criteria

- Data Flow Criteria
  - Considers the definitions and usages of data along paths
  - Three criteria

Graph Testing—Data Flow Criteria

- Assumption: to test a program adequately, we should focus on the flows of data values.
  - To ensure that the values created at one point in the program are created and used correctly later on in the program.
  - Focus on definitions and uses of data values.

- Definition
  - A definition (def) is a location where a value for a variable is stored in memory (assignment, input, …)
- Use
  - A use (use) is a location where a variable value is accessed.
  - Complementary definitions (<u>not used in this course</u>)
    - P-use: a use in a predicate
    - C-use: a use in a computation
- DU-pair
  - A DU-pair is a pair (def, use) for a variable.

Data Flow Principles

- V is the set of all program variables involved in graph model M.
  - def(n) (or def(e)) is the subset of V defined at node n (or edge e)
  - use(n) (or use(e)) is the subset of V used at node n (or edge e)

- Def-clear path
  - Given a variable $v \in V$, and two locations l_i and l_j such that $v \in def(l_i)$ and $v \in use(l_j)$:
    - v is defined in l_i and used in l_j.
  - A path p from l_i to l_j is def-clear with respect to v if and only if for every location l_k on the path $(k \neq i, k \neq j)$, $v \notin def(l_k)$:
    - there is no re-definition of v between (the definition in) l_i and (the use in) l_j.
- DU-path
  - A du-path with respect to variable v is a def-clear (with respect to v), simple path.

    (May not be def-clear for another variable.)

Data Flow Principles

- Def-path: du(n_i,v)
  - du(n_i,v) is the set of du-paths with respect to variable v that start at node n_i.
- Def-pair: du(n_i,l_j,v)
  - du(n_i,l_j,v) is the set of du-paths with respect to variable v that start at node n_i and end at location l_j.
  - All the simple ways to get from a definition of v at n_i to a use of v at l_j without further (re)definitions of v.
- Notes:
  - A definition clear path with respect to variable v is not necessarily definition clear with respect to other variables
  - A definition clear path with respect to variable v is not necessarily use clear with respect to variable v!
    - There might be other uses of v between the definition and the use.
  - $du(n_i, v) = \bigcup_{l_j} du(n_i, l_j, v)$
  - It is also possible (but difficult, and not used a lot) to generalize those definitions and define du(l_i,v) and du(l_i,l_j,v), i.e., def-paths and def-pairs that start at any location (nodes and edges).

Data Flow Criteria – An Example

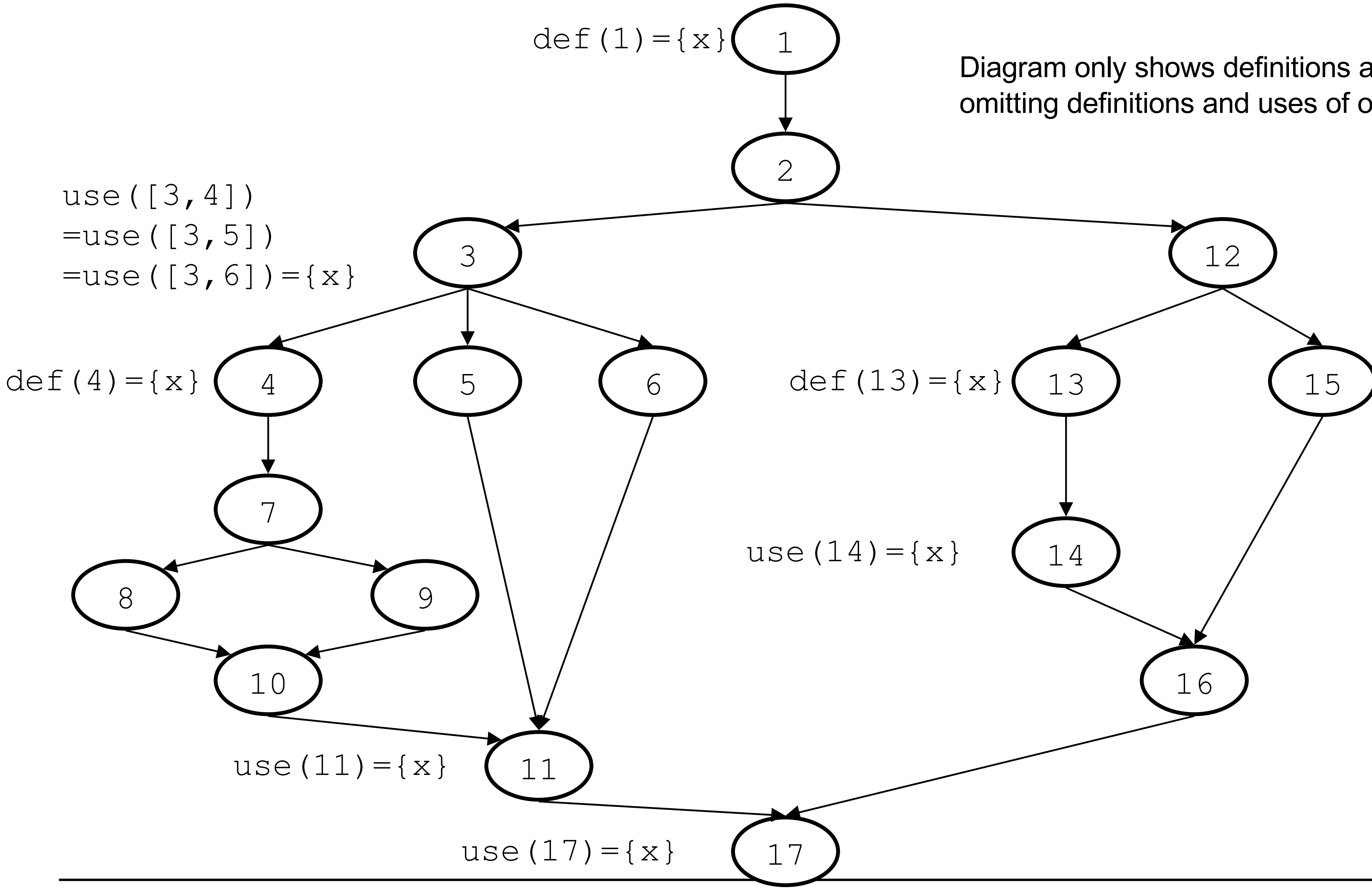


Diagram only shows definitions and uses of x, omitting definitions and uses of other variables.

Data Flow Criteria – An Example (cont.)

- du(1,[3,4],x)={[1,2,3,4]}
- du(1,[3,5],x)={[1,2,3,5]}
- du(1,[3,6],x)={[1,2,3,6]}
- du(1,11,x)={[1,2,3,5,11], [1,2,3,6,11]}
- du(1,14,x)=∅
- du(1,17,x)={[1,2,3,5,11,17], [1,2,3,6,11,17]. [1,2,12,15,16,17]}
- du(1,x)={[1,2,3,4], [1,2,3,5], [1,2,3,6], [1,2,3,5,11], [1,2,3,6,11], [1,2,3,5,11,17], [1,2,3,6,11,17]. [1,2,12,15,16,17]}
- du(4,11,x)={[4.7.8.10.11], [4.7.9.10.11]}
- du(4,17,x)={[4.7.8.10.11,17], [4.7.9.10.11,17]}
- du(4,x)={[4.7.8.10.11], [4.7.9.10.11], [4.7.8.10.11,17], [4.7.9.10.11,17]}
- du(13,14,x)={[13,14]}
- du(13,17,x)={[13,14,16,17]}
- du(13,x)={[13,14], [13,14,16,17]}

Data Flow Criteria

- All-Defs Criterion (ADC)
  - For each def-path set S=du(n,v), TR contains at least one path d in S.
- All-Uses Criterion (AUC)
  - For each def-pair set S=du(n_i,n_j,v), TR contains at least one path d in S.
- All-DU-Paths Criterion (ADUPC)
  - For each def-pair set S=du(n_i,n_j,v), TR contains every path d in S.

Data Flow Criteria – An Example (cont.)

Objectives (sub-paths)

- All-Defs:
  - [1,2,3,4]
  - [4,7,8,10,11]
  - [13,14]
- All-Uses:
  - [1,2,3,4]
  - [1,2,3,5]
  - [1,2,3,6]
  - [1,2,3,5,11]
  - [1,2,3,5,11,17]
  - [4,7,8,10,11]
  - [4,7,8,10,11,17]
  - [13,14]
  - [13,14,16,17]

Test cases (test paths)

- All-Defs:
  - [1,2,3,4,7,8,10,11,17]
  - [1,2,12,13,14,16,17]
- All-Uses:
  - [1,2,3,5,11,17]
  - [1,2,3,6,11,17]
  - [1,2,3,4,7,8,10,11,17]
  - [1,2,12,13,14,16,17]

Data Flow Criteria – An Example (cont.)

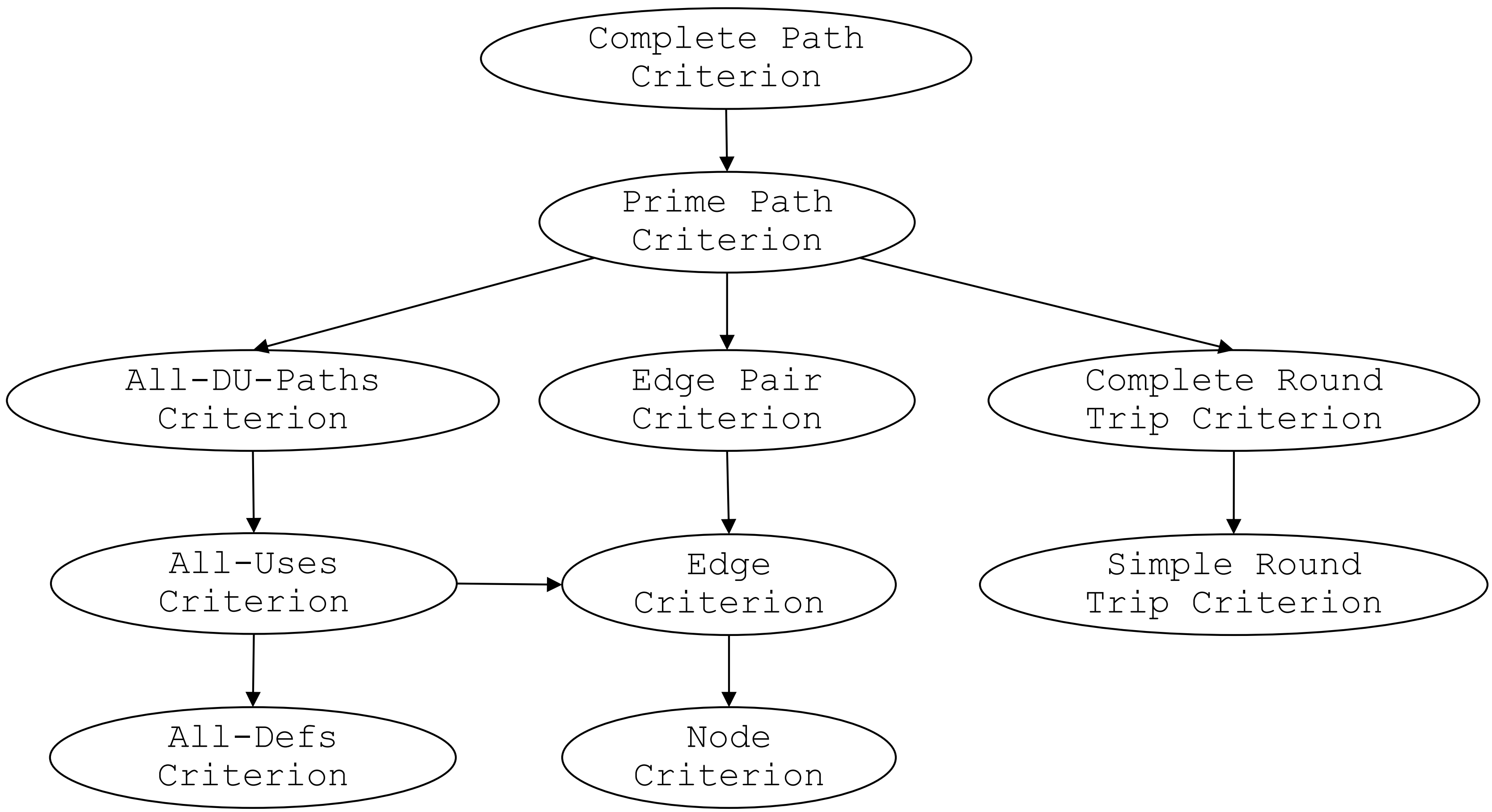
Objectives (sub-paths)

- All-DU-paths:
  - [1,2,3,4]
  - [1,2,3,5]
  - [1,2,3,6]
  - [1,2,3,5,11]
  - [1,2,3,6,11]
  - [1,2,3,5,11,17]
  - [1,2,3,6,11,17]
  - [1,2,12,15,16,17]
  - [4,7,8,10,11]
  - [4,7,9,10,11]
  - [4,7,8,10,11,17]
  - [4,7,9,10,11,17]
  - [13,14]
  - [13,14,16,17]

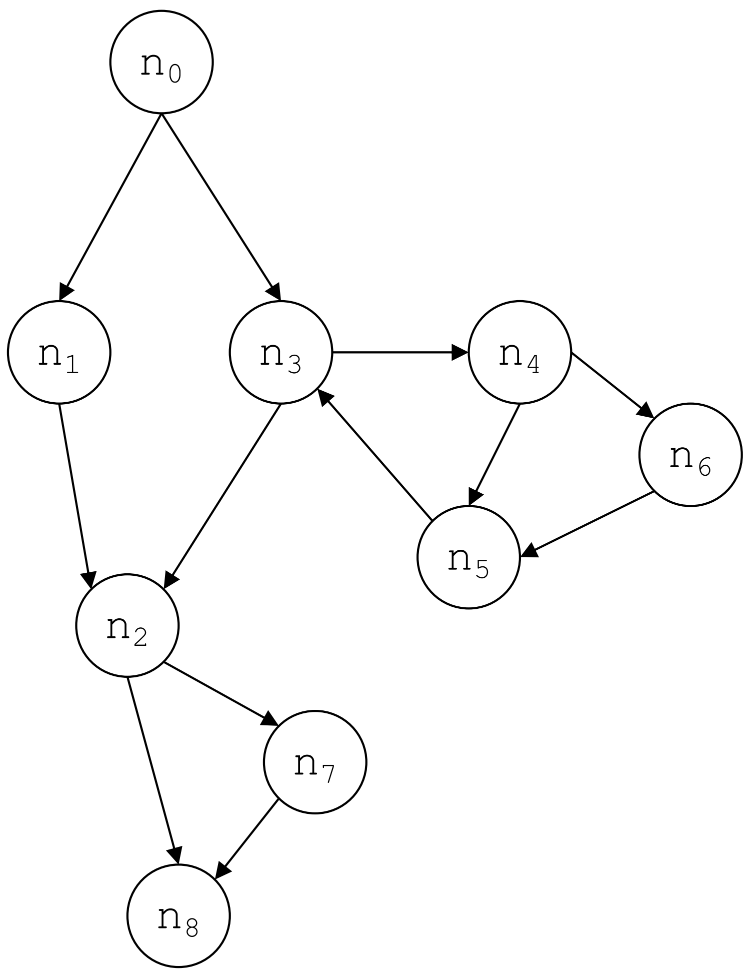
Test cases (test paths)

- All-DU-paths:
  - [1,2,3,5,11,17]
  - [1,2,3,6,11,17]
  - [1,2,3,4,7,8,10,11,17]
  - [1,2,3,4,7,9,10,11,17]
  - [1,2,12,15,16,17]
  - [1,2,12,13,14,16,17]

Subsumption



Comment on Feasibility



- We only used the syntax of the graph to determine test requirements and build test cases

  (we did not consider actual inputs)
- What if (n_2,n_7) can only execute after n_3?
  - path[t_1]=[n_0,n_1,n_2,n_7,n_8] was used for the All-Edges criterion (recall previous lecture)
  - t_1 is not feasible
- The test sets we built may not all be feasible.
  - Not necessarily because of unfeasible test requirements
  - But because we build test paths that are unfeasible
- Alternative test sets must be investigated.
  - To still satisfy all feasible test requirements for the selected criterion