Data Flow Coverage 1

Stuart Anderson

School of informatics

Why Consider Data Flow?

- **Control Flow:**
  - Statement and branch coverage criteria are weak.
  - Condition coverage and path coverage are more costly and can become infeasible.
- **Data Flow:**
  - Base the coverage criterion on how variables are defined and used in the program.
  - Coverage is based on the idea that in principle for each statement in the program we should consider all possible ways of defining the variables used in the statement.
- **Data Flow Analysis** arose in the study of compiling –¿ as well as suggesting coverage criteria it can also provide a means of statically checking variables are defined before use.

Terminology

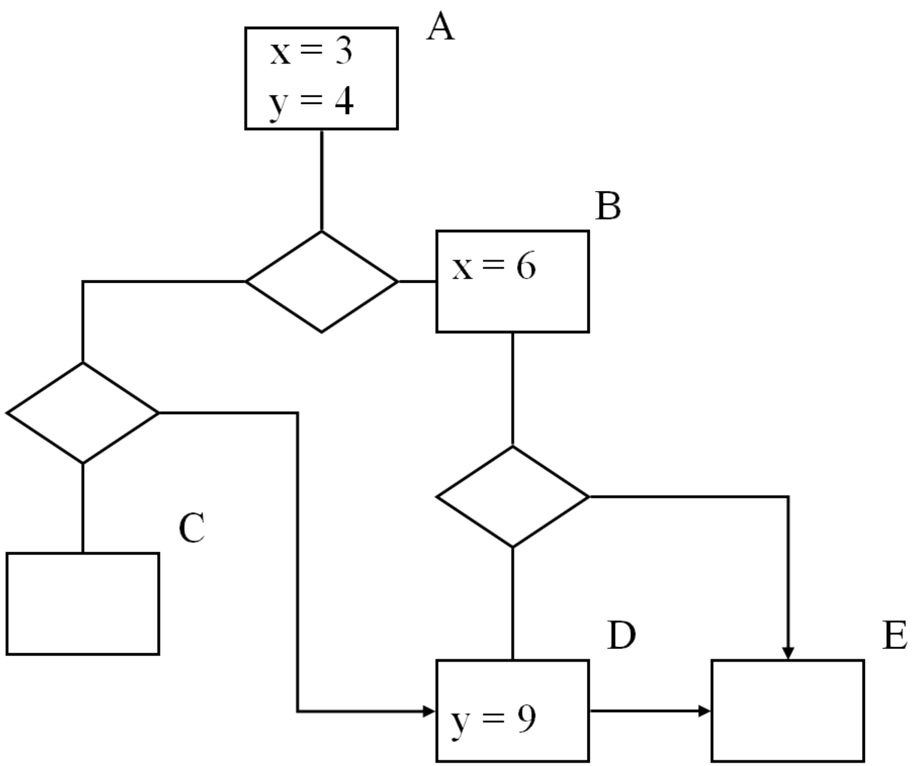
We introduce some standard naming conventions:

- P –¿ code under test.
- $G(P)$ –¿ control flow graph of P, $G(P) = (V, E, s, f)$ (Vertices, Edges, start node, finish node)
- Path is a sequence of vertices: v_0, v_1,..., v_k where for each i $(1 < i < k)$: (v_{i-1}, v_i) is a member of E.
- x is a variable of P

Terminology

- If v is a vertex of the flow graph we define:
  - $defs(v)$: the set of all variables that are defined at v (i.e. are on the left-hand side of an assignment or similar)
  - $undef(v)$: the set of all variables whose value is undefined after executing the code corresponding to v.
  - $c\text{-}use(v)$: (c for computation) all variables that are used to define other variables in the code corresponding to v
  - $p\text{-}use(v, v')$: (p for predicate) all variables used in taking the (v, v') branch out of vertex v.
  - v_0, v_1,..., v_k is a def-clear path for x, if x is not in $defs(v_i)$ for $0 < i < k$

Def-Clear Path



- A,D,E is def-clear for x but not for y
- A,B,E is def-clear for y but not for x

Refinement

- We call a c-use of x **global**, if it is not preceded by a definition of x in the same basic block.

- We call a def of x **global**, if it is used in some other vertex in the flow graph.

- We refine our definitions only to take account of global uses and definitions (e.g. $c\text{-}use(v)$ is the global c-uses in vertex v)

Definition and Use

```
public int Segment(int t[], int l, int u) {
  // Assumes t is in ascending order, and l < u,
  // counts the length of the segment
  // of t with each element l < t[i] < u
  int k = 0;

  for(int i = 0; i < t.length && t[i] < u; i++) {
    if(t[i] > l) {
      k++;
    }
  }

  return k;
}
```

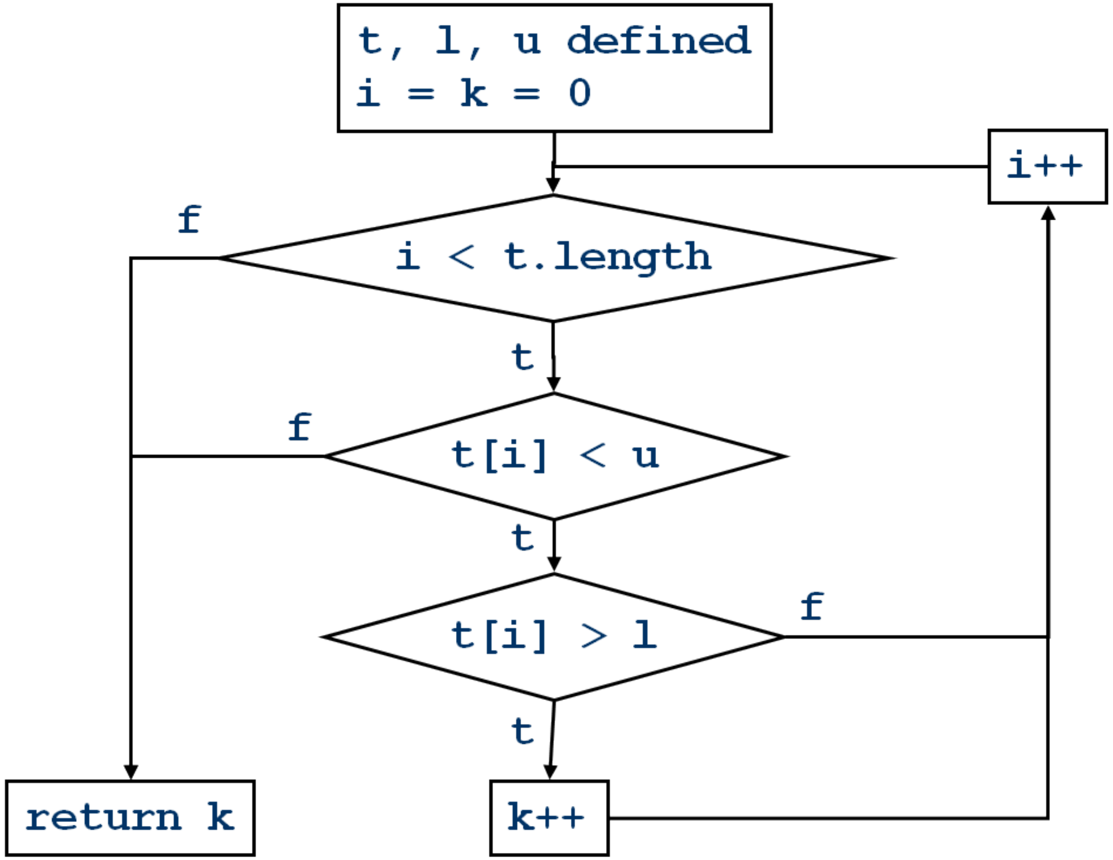
Defn of k

p-use of i

c-use and definition of k

c-use of k

Corresponding Flow Graph



Data-flow Terminology

- $dcu(x, v) = \{v'$ in $V \mid x$ is in $c\text{-}use(v')$ and there is a def-clear path for x from v to $v'\}$

  This is the set of vertices with c-uses of x that can potentially be influenced by the definition of x at v

- $dpu(x, v) = \{(v', v'')$ in $E \mid x$ is in $p\text{-}use(v', v'')$ and there is a def clear path for x from v to $(v', v'')\}$

  This is the set of edges with p-uses of x that can potentially be influenced by the definition of x at v.

Frankl and Weyuker's data-flow coverage criteria

1. **All-defs** requires that for each definition of a variable x in P, the set of paths Π executed by the test set T contains a def-clear path from the definition to at least one $c\text{-}use$ or one $p\text{-}use$ of x.
   ☞ *all definitions get used.*
2. **All-c-uses** requires that for each definition of a variable x in P, and each $c\text{-}use$ of x reachable from the definition (see definition of $dcu(x, v)$), Π contains a def-clear path from the definition to the c-use.
   ☞ *all computations affected by each definition are exercised.*
3. **All-p-uses** requires that for each definition of a variable x in P, and each $p\text{-}use$ of x reachable from the definition (see definition of $dpu(x, v)$), Π contains a def-clear path from the definition to the $p\text{-}use$.
   ☞ *all branches affected by each definition are exercised.*

Frankl and Weyuker's data-flow coverage criteria

4. **All-c-uses/some-p-uses**, for each definition of x in P at v:
   - If $dcu(x, v)$ is not empty, the paths Π executed by the test set T contains a def-clear path from v to each member of $dcu(x, v)$;
   - otherwise, the paths Π executed by the test set T contains a def-clear path from v to an edge in $dpu(x, v)$.

   ☞ *all definitions get used, and if they affect computations then all affected computations are exercised.*
5. **All-p-uses/some-c-uses**, for each definition of x in P at v:
   - If $dpu(x, v)$ is not empty, the paths Π executed by the test set T contains a def-clear path from v to each edge in $dpu(x, v)$;
   - otherwise, the paths Π executed by the test set T contains a def-clear path from v to a member of $dcu(x, v)$.

   ☞ *all definitions get used, and if they affect control flow then all affected branches are exercised.*

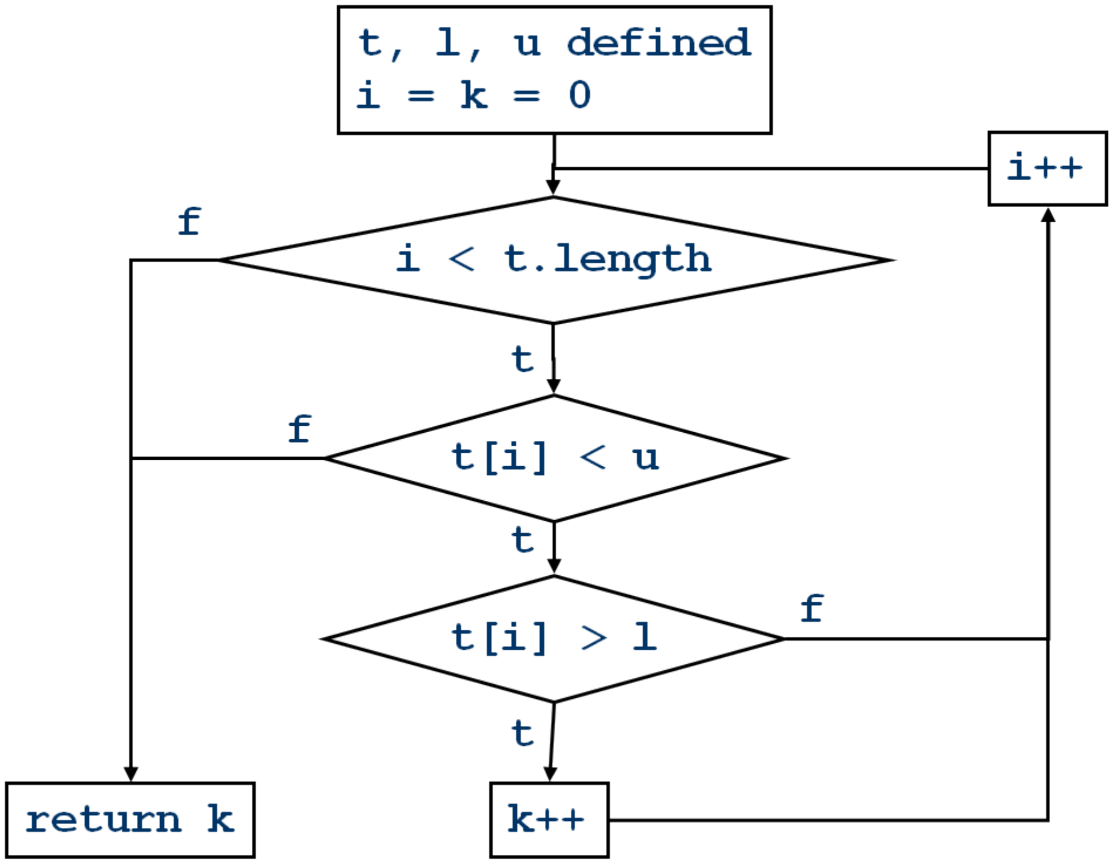
Frankl and Weyuker's data-flow coverage criteria

6. **All-uses** requires that for each definition of x at v in P, the set of paths Π executed by the test set T contains a def-clear path from v to both $dcu(x, v)$ and $dpu(x, v)$.
   ☞ *every computation and branch directly affected by a definition is exercised.*

7. **All-du-paths** requires that for each definition of x at v in P, the set of all paths Π executed by the test set T contains all def-clear paths from v to both $dcu(x, v)$ and $dpu(x, v)$, such that each path is loop free, or contains at most one loop of any loop on the path.
   ☞ *all-uses, but requires exercise of all def-use paths, modulo looping.*

8. **All-paths** requires that all paths through the program be executed.

Flow Graph



*What is the **point** of all these distinctions?*

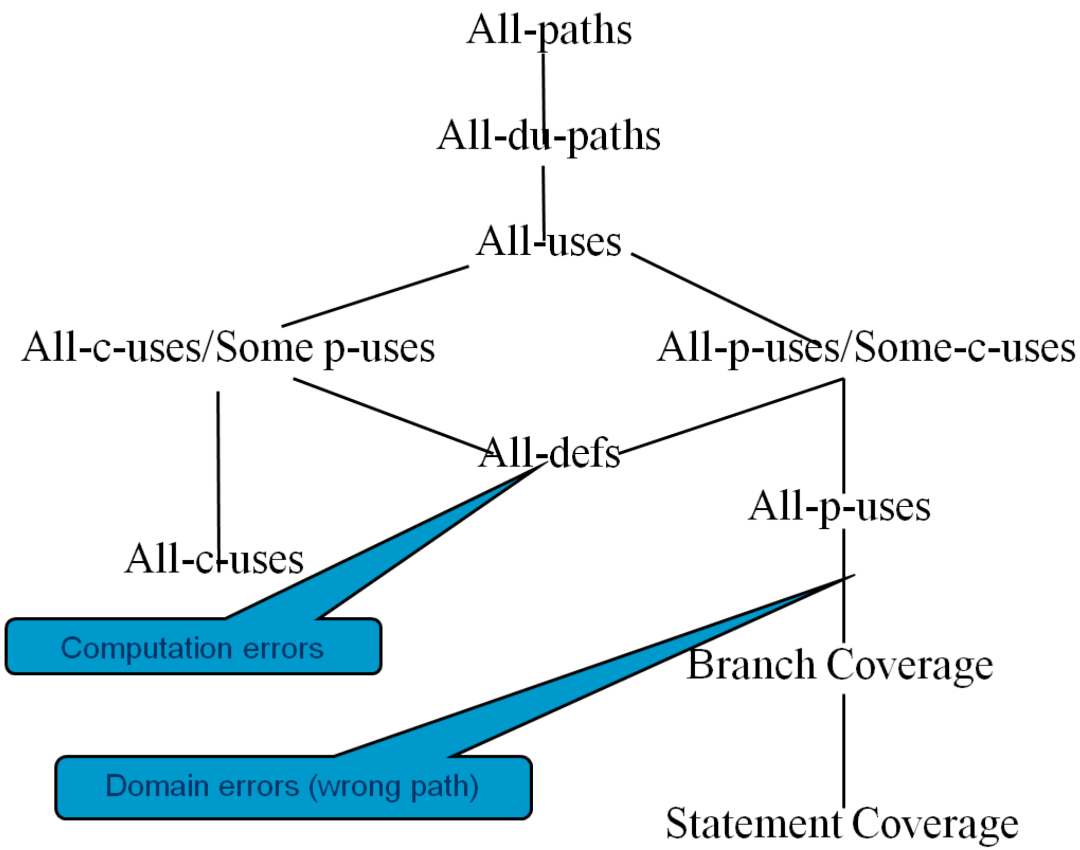
Subsumption

- We say that test coverage criterion A **subsumes** test coverage criterion B if and only if, for every program P, every test set satisfying A with respect to P also satisfies B with respect to P.

- i.e. if any test set satisfying criterion A will (provably) always also satisfy B, then *"A subsumes B"*.

- Example: branch coverage subsumes statement coverage.

Subsumption Relationships



Uses of Data Flow analysis

- We can use the analysis of definition and use to calculate optimistic and pessimistic estimates of whether variables are defined or not at particular vertices in the flow graph.

- We can use these to discover potential faults in the program.

- For example:
  – If a definition is only followed by definitions of the same variable - is it useful?
  – If we use a variable and it is not always preceded by a definition we might use it when it is undefined.

Summary

- Data-flow coverage criteria are claimed to provide a better measure of coverage than control flow because they track dependencies between variables in the flow graph.

- Frankl and Weyuker have done some empirical work on this (see references) and there is some justification for believing data- flow coverage is a good approach to structural testing.

- There are the usual issues of the computability of the exact relationships between definition and use but we are usually satisfied with approximations.

Required Readings

- **Textbook (Pezzè and Young):** Chapter 6, Dependence and Data Flow Models
- **Textbook (Pezzè and Young):** Chapter 13, Data Flow Testing
- P. G. Frankl and E. J. Weyuker. 1988. An Applicable Family of Data Flow Testing Criteria. IEEE Trans. Softw. Eng. 14, 10 (October 1988), 1483-1498. http://dx.doi.org/10.1109/32.6194

Suggested Readings

- L. A. Clarke, A. Podgurski, D. J. Richardson and Steven J. Zeil, A Formal Evaluation of Data Flow Path Selection Criteria, IEEE Transactions on Software Engineering, 15 (11), November 1989, pp. 1318-1332. http://dx.doi.org/10.1109/32.41326

- Lori A. Clarke, Andy Podgurski, Debra J. Richardson, and Steven J. Zeil. 1985. A comparison of data flow path selection criteria. In Proceedings of the 8th international conference on Software engineering (ICSE '85). IEEE Computer Society Press, Los Alamitos, CA, USA, 244-251. http://portal.acm.org/citation.cfm?id=319568.319646

- S.C. Ntafos. 1988. A Comparison of Some Structural Testing Strategies. IEEE Trans. Softw. Eng. 14, 6 (June 1988), 868-874. http://dx.doi.org/10.1109/32.6165

- Sandra Rapps and Elaine J. Weyuker. 1982. 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, USA, 272-278. http://portal.acm.org/citation.cfm?id=800254.807769