CMPT 473
Software Quality Assurance

Data Flow Criteria

Nick Sumner
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows

\[ x = \ldots \]

Definition (Def)
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows

\[
\text{Definition (Def)} : \quad y = x
\]

\[
\text{Use : } \quad y = x
\]

\[
\text{Use : } \quad x = \ldots
\]
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows

\[ x = \ldots \]  \hspace{2cm} \text{Definition (Def)}

\[ y = x \]  \hspace{2cm} \text{Use}

Def-Use Pair
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows

- New goal?
  - Try to test all of the ways that a Def may flow to its varied uses
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows
- New goal?
  - Try to test all of the ways that a Def may flow to its varied uses
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows
- New goal?
  - Try to test all of the ways that a Def may flow to its varied uses

What are the defs?
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows
- New goal?
  - Try to test all of the ways that a Def may flow to its varied uses

What are the defs?
What are the uses?
Focus on Data

- Programs manipulate data
  - Focus on testing the ways that data moves/flows

- New goal?
  - Try to test all of the ways that a Def may flow to its varied uses

What are the defs?
What are the uses?
What may be interesting to test?
Reachability

1. x = ...
2. print(x)
3. x = ...
4. print(x)
Reachability

What are the def-use pairs?
Reachability

1. \(x=\ldots\)
2. \(\text{print}(x)\)
3. \(x=\ldots\)
4. \(\text{print}(x)\)

What are the def-use pairs?

What is interesting to test?
Reachability

1. x = ...
2. print(x)
3. x = ...
4. print(x)

What are the use pairs?
What is interesting to test?

- The def at 1 is *killed* by the def at 3,
Reachability

What are the use pairs?

What is interesting to test?

- The def at 1 is killed by the def at 3, so it does not reach 4
Possible Criteria

- **AllDefs Coverage**
  - Every Def is covered by at least one test of a use
Possible Criteria

- **AllDefs Coverage**
  - Every Def is covered by at least one test of a use

- **All Uses Coverage**
  - Every Use is covered through at least one def
Possible Criteria

- **AllDefs Coverage**
  - Every Def is covered by at least one test of a use

- **AllUses Coverage**
  - Every Use is covered through at least one def

- **AllDef-Use Pairs Coverage**
  - All def-use pairs are covered
Possible Criteria

- **All Defs Coverage**
  - Every Def is covered by at least one test of a use

- **All Uses Coverage**
  - Every Use is covered through at least one def

- **All Def-Use Pairs Coverage**
  - All def-use pairs are covered

- **All Def-Use Paths Coverage**
  - All simple paths between def-use pairs are covered
Possible Criteria

- **AllDefs Coverage**
  - Every Def is covered by at least one test of a use
- **All Uses Coverage**
  - Every Use is covered through at least one def
- **All Def-Use Pairs Coverage**
  - All def-use pairs are covered
- **All Def-Use Paths Coverage**
  - All simple paths between def-use pairs are covered

How do these compare to edge coverage?
Possible Criteria

- **All Defs Coverage**
  - Every Def is covered by at least one test of a use
- **All Uses Coverage**
  - Every Use is covered through at least one def
- **All Def-Use Pairs Coverage**
  - All def-use pairs are covered
- **All Def-Use Paths Coverage**
  - All simple paths between def-use pairs are covered

How do these compare to edge coverage?
How do these compare to prime paths?
A Brief Example

- What should be tested for the different criteria?
Another Example

- What should be tested for the different criteria?

```
x = ...
print(x)
```
Moving On...

- Where else might we see graphs when thinking about program design?
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs

How does graph coverage translate?
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs

How does graph coverage translate?

How might it be useful?
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
  - inheritance
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
  - inheritance

How does graph coverage translate?
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
  - inheritance

How does graph coverage translate?

How might it be useful?
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
  - inheritance
  - finite state machines
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
  - inheritance
  - finite state machines

How does graph coverage translate?
Moving On...

- Where else might we see graphs when thinking about program design?
  - call graphs
  - inheritance
  - finite state machines

How does graph coverage translate?

How might it be useful?
Moving On...

• Where else might we see graphs when thinking about program design?
  – call graphs
  – inheritance
  – finite state machines
  – ...

Graph coverage is a powerful & general concept. You can apply it to many varied features of programs.
No One Clear Winner

- Is there a case where input space partitioning is weaker than CFG coverage?
No One Clear Winner

- Is there a case where input space partitioning is weaker than CFG coverage?
- Is there a case where CFG coverage is weaker than input space partitioning?
No One Clear Winner

- Is there a case where input space partitioning is weaker than CFG coverage?
- Is there a case where CFG coverage is weaker than input space partitioning?
- Using just one approach may not be enough
  - But maybe there are other ways to evaluate adequacy...