Lecture 9 — Coverage Testing I

David J. Pearce

School of Engineering and Computer Science
Victoria University of Wellington

Software Testing

“Testing is indispensible, and no software system can be regarded as dependable if it has not been extensively tested, even if its correctness has been proven mathematically.

–Software for Dependable Systems

Code Coverage

Test coverage is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs. –Wikipedia

- **Intuition**: *higher test coverage implies more of the program has been tested.*
- **Example**: IEC61508 requires **100%** MC/DC coverage

"Test coverage is of little use as a numeric statement of how good your tests are.... If you make a certain level of coverage a target, people will try to attain it. The trouble is that high coverage numbers are too easy to reach with low quality testing." –Martin Fowler

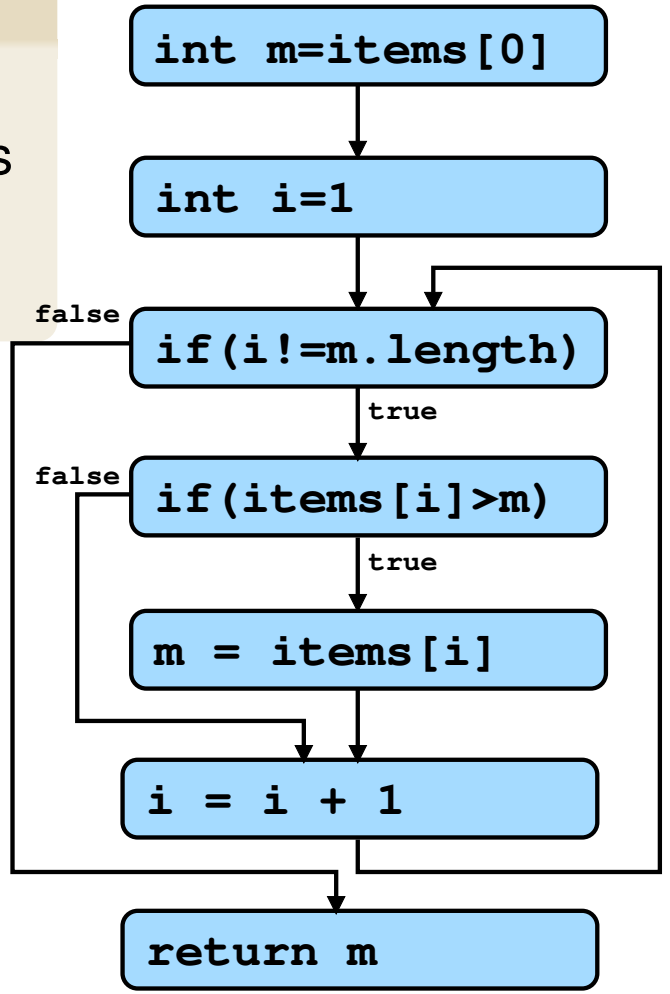
See: `https://martinfowler.com/bliki/TestCoverage.html`

Control-Flow Graph

Control-Flow Graph (CFG)

A directed graph representation of a program where each node typically represents a *basic block* and edges represent *realisable flows* between blocks. CFG's have unique *entry* nodes.

```
int max(int[] items) {
  int m = items[0];
  int i = 1;
  while(i != m.length) {
    if(items[i] > m) { m = items[i]; }
    i = i + 1;
  }
  return m;
}
```



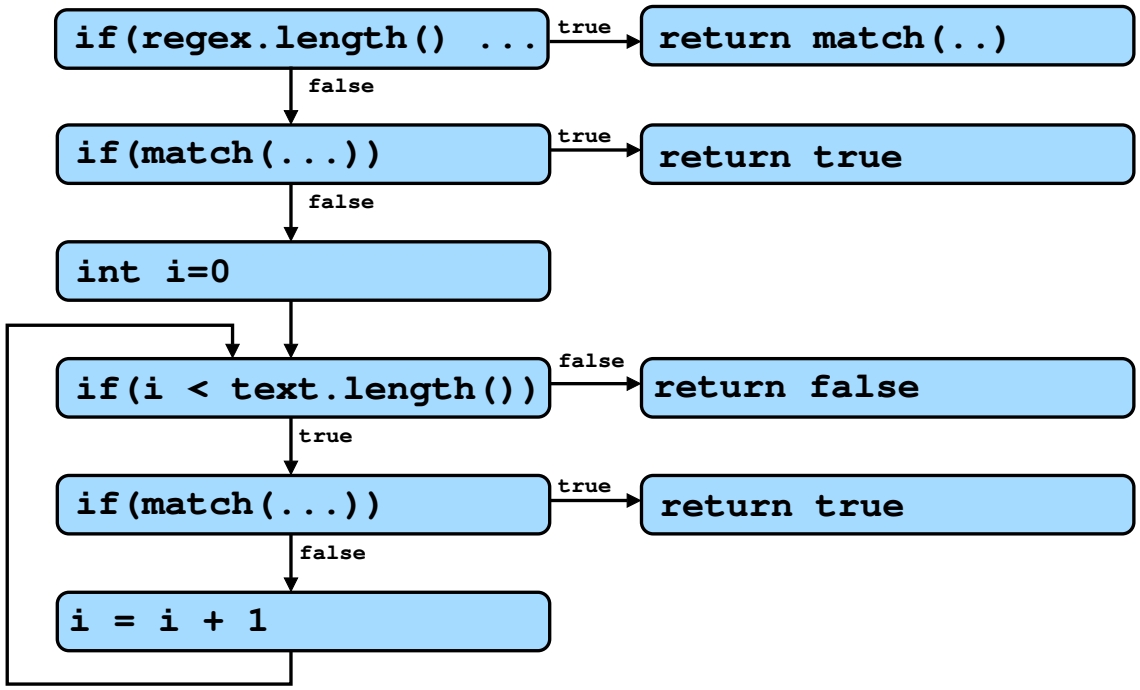
- **Control statements** are deconstructed in a CFG!

Regex Example

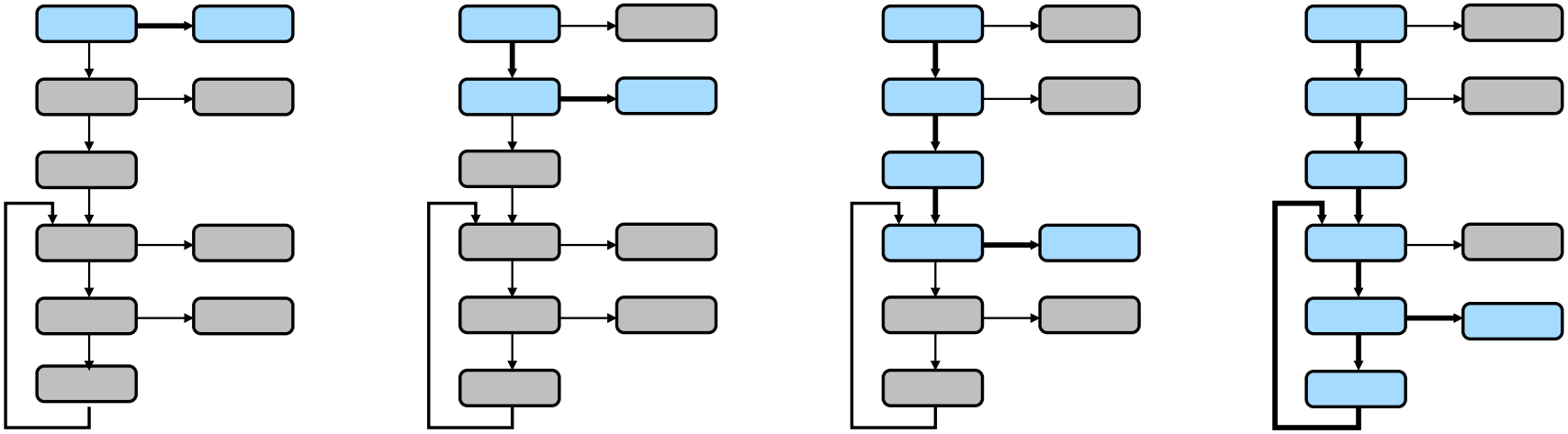
```
boolean match(String regex, String text) {
  if(regex.length() > 0 && regex.charAt(0) == '^'){
    return match(regex,1,text,0);
  } else if(match(regex,0,text,0)) {
    return true;
  }
  int i = 0;
  while(i < text.length()) {
    if(match(regex,0,text,i)) { return true; }
    else { i = i + 1; }
  }
  return false;
}
```

- Taken from tool for matching **regular expressions**.

Regex Example CFG



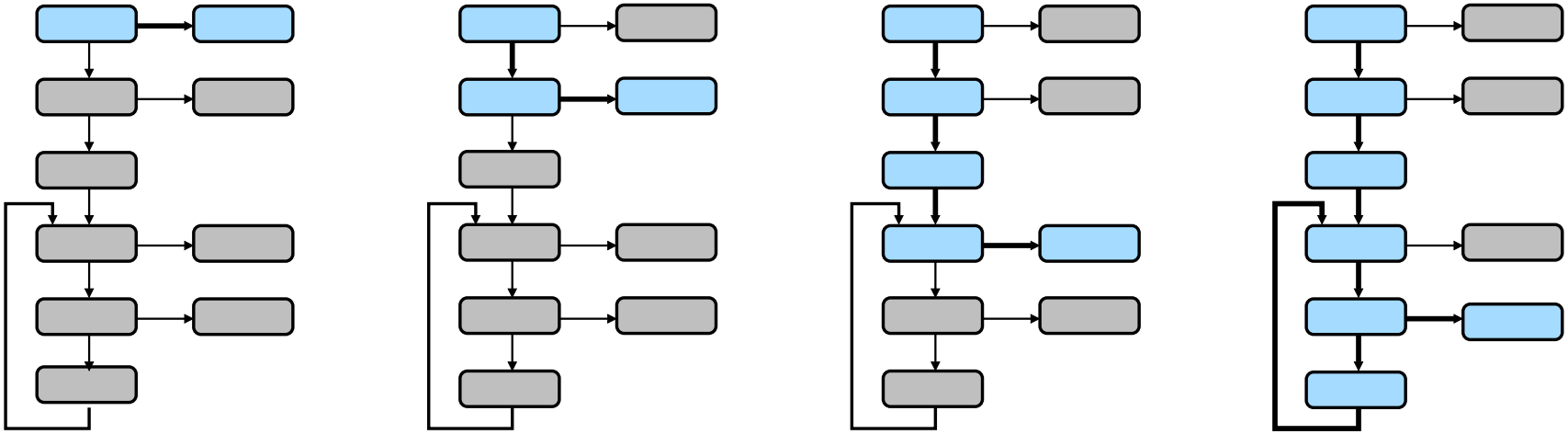
Statement Coverage



- **Statement Coverage**. *The proportion of statements executed in a program.*

- Need **four** test cases for 100% coverage above.

Branch Coverage



- **Branch Coverage**. *The proportion of conditions where both branches taken in a program.*

- Need same **four** test cases for 100% coverage above!

Modified Condition/Decision Coverage (MC/DC)

In software testing, the **modified condition/decision coverage (MC/DC)** is a code coverage criterion that requires all of the below during testing:

- Each **entry** and **exit** point is invoked
- Each **decision** takes every possible outcome
- Each **condition** in a decision takes every possible outcome
- Each condition in a decision is shown to **independently** affect the outcome of the decision.

–Wikipedia

- **Terminology:**
  - *Condition:* An atomic boolean expression.
  - *Decision:* A boolean expression composed of two or more conditions.

MC/DC Example

```
if((A && B) || C) { ... }
```

A	B	C	(A&&B)||C
false	false	false	false
false	false	true	true
false	true	false	false
false	true	true	true
true	false	false	false
true	false	true	true
true	true	false	true
true	true	true	true

- How many test cases for **100% MC/DC coverage**?

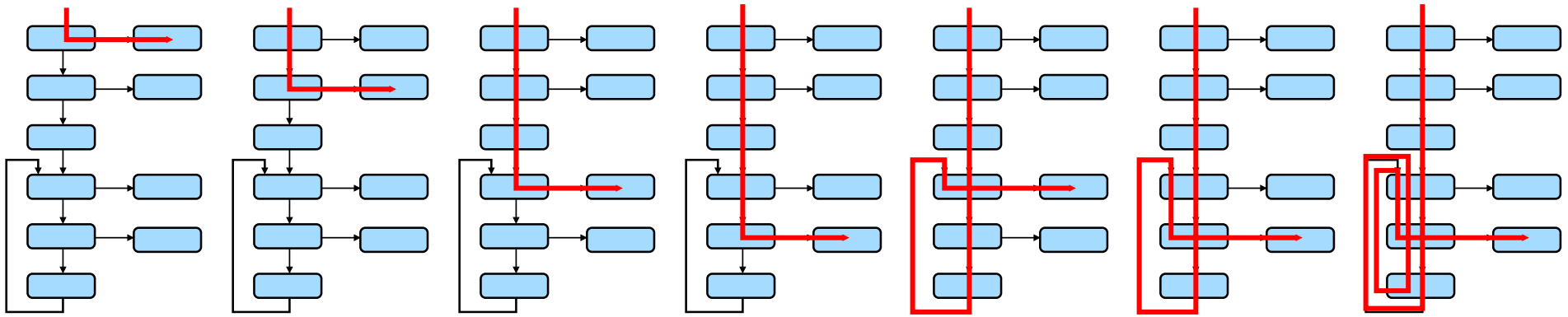
MC/DC Example (Cont'd)

A	B	C	(A&&B)||C
false	false	false	false
false	false	true	true
false	true	false	false
false	true	true	true
true	false	false	false
true	false	true	true
true	true	false	true
true	true	true	true

- Above shows condition `C` **independently** affecting decision.

- Above shows condition `B` **independently** affecting decision.

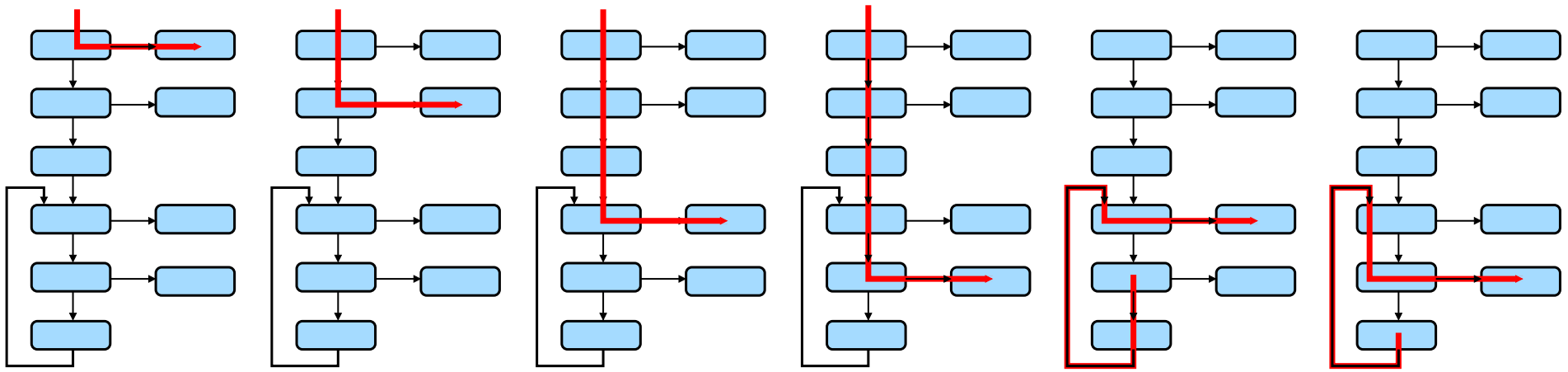
Path Coverage



- **Path Coverage**: every distinct execution path through program taken.

- **Question**: *how many distinct paths for this example?*

Prime Path Coverage



- **Simple Path**: Except for first and last nodes, no two nodes are identical.
- **Prime Path**: A simple path which is not proper subpath of any other simple path.
- **Question**: *which prime paths are missing from above?*

Unrealisable Paths

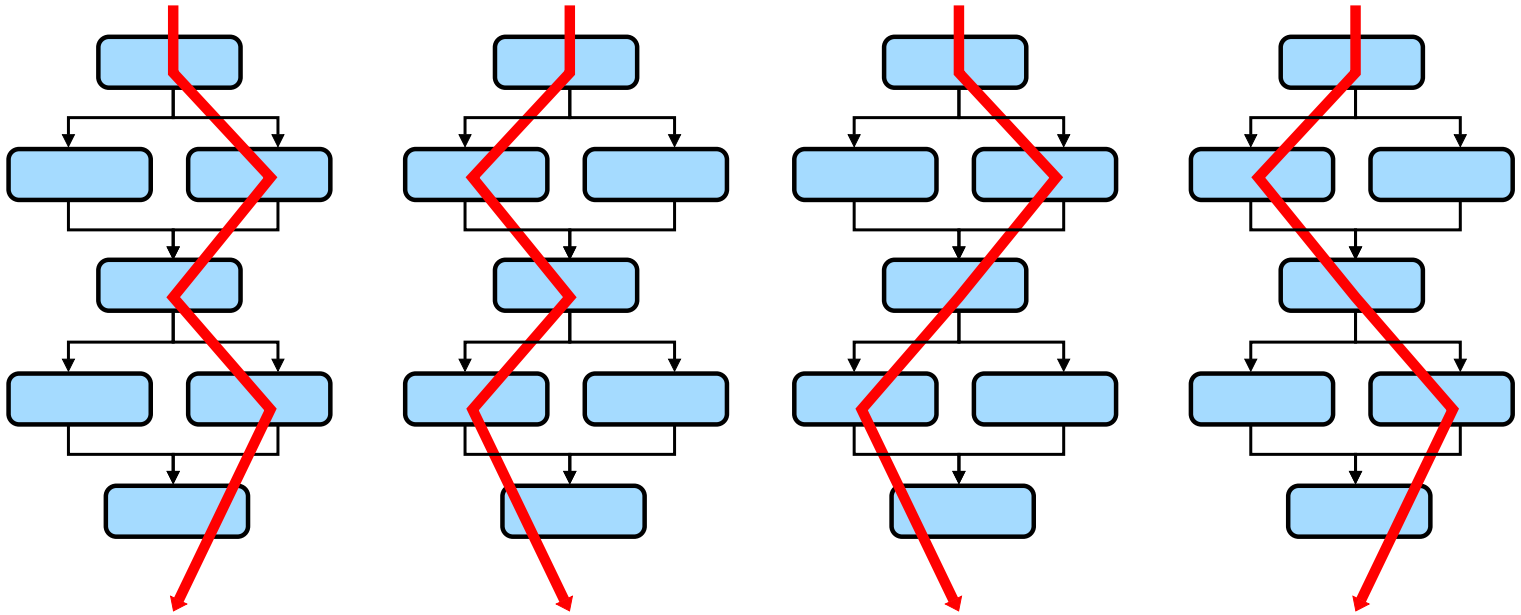
```
if(...) {

} else {


}
if(...) {

} else {


}
```



- For two conditions in sequence, have exactly **four** paths.

Unrealisable Paths (Cont'd)

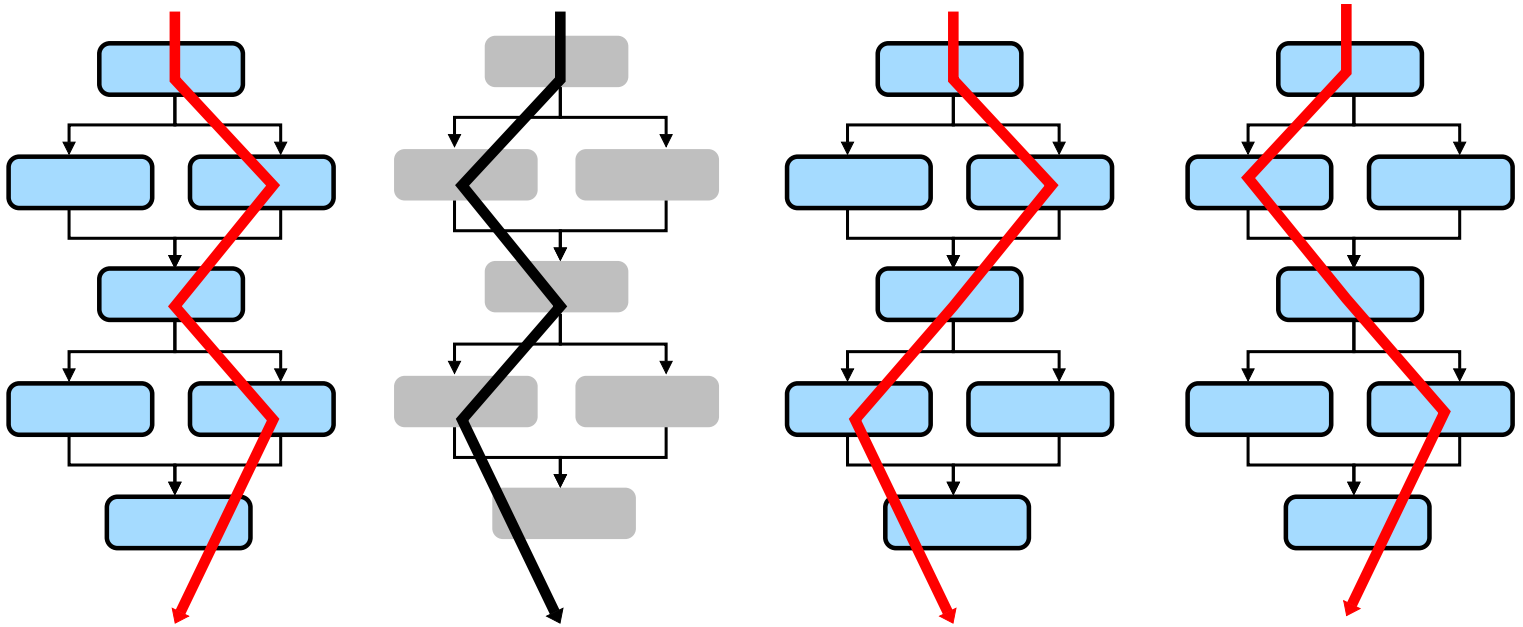
```
if(x>=0)  {

} else {


}
if(x<=0)  {

} else {


}
```



- In this case, **`false-false`** impossible!

Exceptions

```
byte[] readFile(String filename) {
  try {
    FileReader reader = new FileReader(filename);
    return reader.read();
  } catch(IOException e) {
    return null;
  }
}
```

- **Question)** which metrics require `return null` executed?