Structural Testing

Dr. Paul West

Department of Computer Science
College of Charleston

February 27, 2014

Condition Testing

- Branch coverage exposes faults in how a computation has been decomposed into cases
  - intuitively attractive: check the programmer's case analysis
  - but only roughly: groups cases with the same outcome
- Condition coverage considers case analysis in more detail
  - also individual conditions in a compound Boolean expression
    - e.g., both parts of digit_high == 1 || digit_low == -1

Basic Condition Testing

- Adequacy criterion: each basic condition must be executed at least once
- Coverage: $\frac{\#\textit{truth values taken by all basic conditions}}{2*\#\textit{basic conditions}}$

Basic Conditions vs Branches

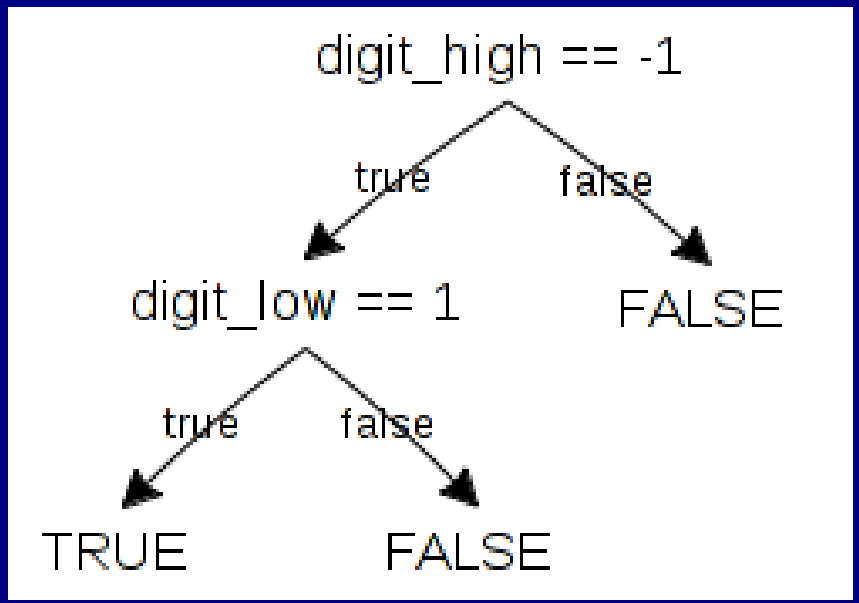
- Basic condition adequacy criterion can be satisfied without satisfying branch coverage

```
T4 = {"first+test%9Ktest%K9"}
```

- satisfies basic condition adequacy
- does not satisfy branch condition adequacy
- Branch and basic condition are not comparable (neither implies the other)

Covering Branches And Conditions

- Branch and condition adequacy:
  - cover all conditions and all decisions
- Compound condition adequacy:
  - Cover all possible evaluations of compound conditions
  - Cover all branches of a decision tree



Compound Conditions: Exponential Complexity

$(((a||b)\&\&c)||d)\&\&e$

Test Case	a	b	c	d	e
(1)	T	-	T	-	T
(2)	F	T	T	-	T
(3)	T	-	F	T	T
(4)	F	T	F	T	T
(5)	F	F	-	T	T
(6)	T	-	T	-	F
(7)	F	T	T	-	F
(8)	T	-	F	T	F
(9)	F	T	F	T	F
(10)	F	F	-	T	F
(11)	T	-	F	F	-
(12)	F	T	F	F	-
(13)	F	F	-	F	-

Modified condition/decision (MC/DC)

- Motivation: Effectively test important combinations of conditions, without exponential blowup in test suite size
  - “Important” combinations means: Each basic condition shown to independently affect the outcome of each decision
- Requires:
  - For each basic condition C, two test cases,
  - values of all evaluated conditions except C are the same
  - compound condition as a whole evaluates to true for one and false for the other

MC/DC: linear complexity

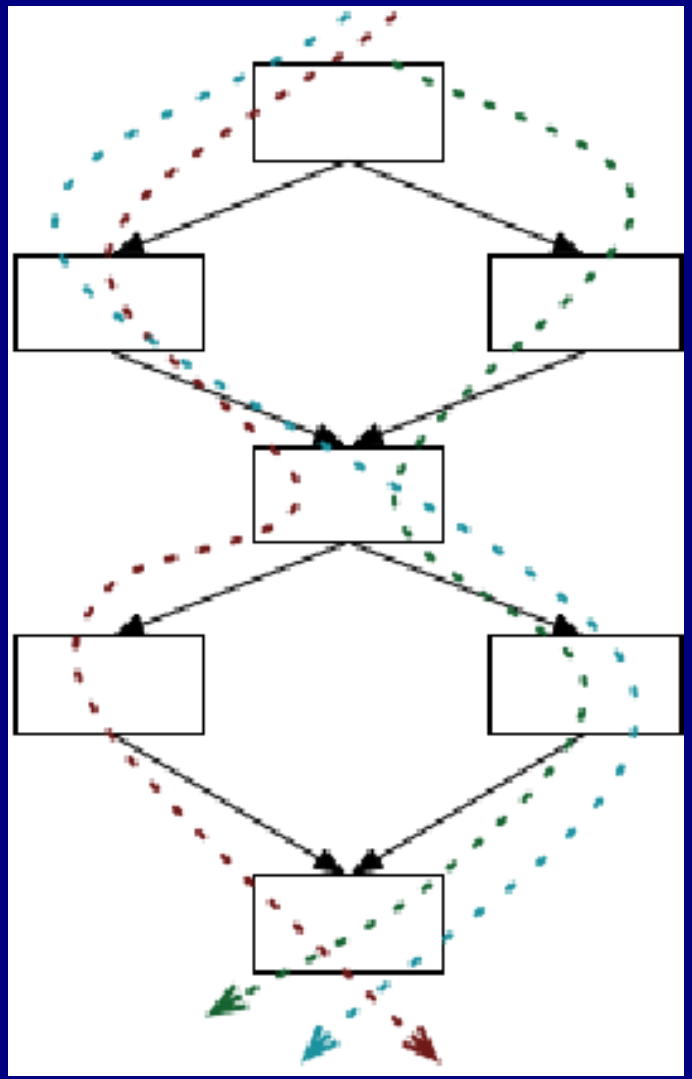
- N+1 test cases for N basic conditions
  (((a || b) && c) || d) && e

Test Case	a	b	c	d	e	outcome
(1)	<u>true</u>	–	<u>true</u>	–	<u>true</u>	true
(2)	false	<u>true</u>	true	–	true	true
(3)	true	–	false	<u>true</u>	true	true
(6)	true	–	true	–	<u>false</u>	false
(11)	true	–	<u>false</u>	<u>false</u>	–	false
(13)	<u>false</u>	<u>false</u>	–	false	–	false

- Underlined values independently affect the output of the decision
- Required by the RTCA/DO-178B standard

Comments on MC/DC

- MC/DC is
  - basic condition coverage (C)
  - branch coverage (DC)
  - plus one additional condition (M):
    every condition must independently affect the decision's output
- It is subsumed by compound conditions and subsumes all other criteria discussed so far
  - stronger than statement and branch coverage
- A good balance of thoroughness and test size (and therefore widely used)



- Should we explore sequences of branches (paths) in the control flow?
- Many more paths than branches
- A pragmatic compromise will be needed

Path adequacy

- Decision and condition adequacy criteria consider individual program decisions
- Path testing focuses consider combinations of decisions along paths
- Adequacy criterion: each path must be executed at least once
- Coverage: $\frac{\#executed\ paths}{\#paths}$

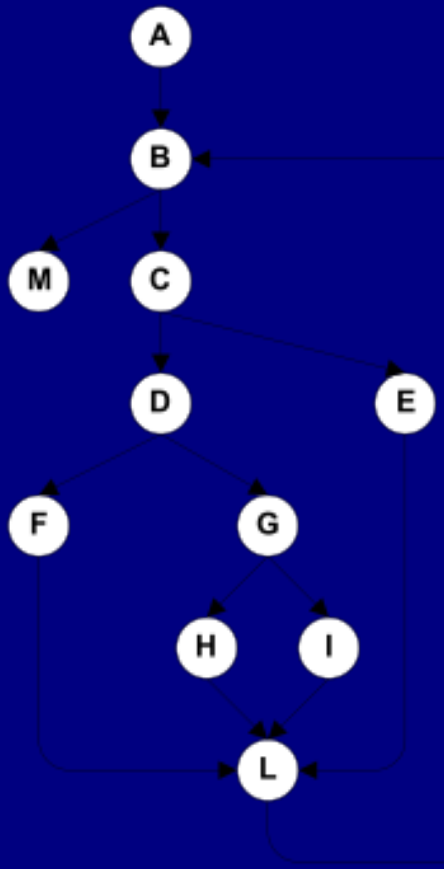
Practical path coverage criteria

- The number of paths in a program with loops is unbounded
  - the simple criterion is usually impossible to satisfy
- For a feasible criterion: Partition infinite set of paths into a finite number of classes
- Useful criteria can be obtained by limiting
  - the number of traversals of loops
  - the length of the paths to be traversed
  - the dependencies among selected paths

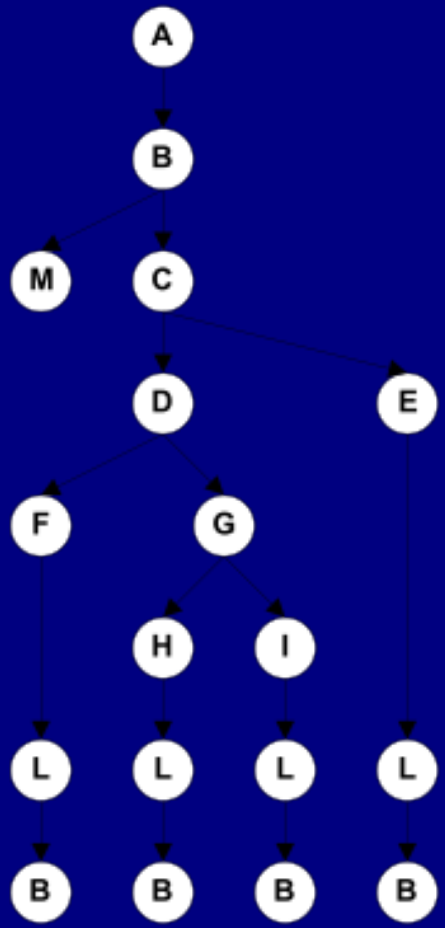
Boundary interior path testing

- Group together paths that differ only in the subpath they follow when repeating the body of a loop
  - Follow each path in the control flow graph up to the first repeated node
  - The set of paths from the root of the tree to each leaf is the required set of subpaths for boundary/interior coverage

Boundary interior adequacy for cgidecode



(i)



(ii)

Limitations of boundary interior adequacy

- The number of paths can still grow exponentially

```
if (a) {
    S1;
}
if (b) {
    S2;
}
if (c) {
    S3;
}
...
if (x) {
    Sn;
}
```

- The subpaths through this control flow can include or exclude each of the statements Si, so that in total N branches result in 2^N paths that must be traversed
- Choosing input data to force execution of one particular path may be very difficult, or even impossible if the conditions are not independent

Loop boundary adequacy

- Variant of the boundary/interior criterion that treats loop boundaries similarly but is less stringent with respect to other differences among paths
- Criterion: A test suite satisfies the loop boundary adequacy criterion iff for every loop:
  - In at least one test case, the loop body is iterated zero times
  - In at least one test case, the loop body is iterated once
  - In at least one test case, the loop body is iterated more than once
- Corresponds to the cases that would be considered in a formal correctness proof for the loop

LCSAJ adequacy

- Linear Code Sequence And Jumps: How much testing is enough?
  We use a sequential subpath in the CFG starting and ending in a branch
  - TER1 = statement coverage
  - TER2 = branch coverage
  - TERn+2 = coverage of n consecutive LCSAJs
- TER = Test Effectiveness Ratio

Cyclomatic adequacy

- Cyclomatic number:
  number of independent paths in the CFG
- A path is representable as a bit vector, where each component of the vector represents an edge
- “Dependence” is ordinary linear dependence between (bit) vectors
- If e = #edges, n = #nodes, c = #connected components of a graph, it is:
- e - n + c for an arbitrary graph
- e - n + 2 for a CFG
- Cyclomatic coverage counts the number of independent paths that have been exercised, relative to cyclomatic complexity

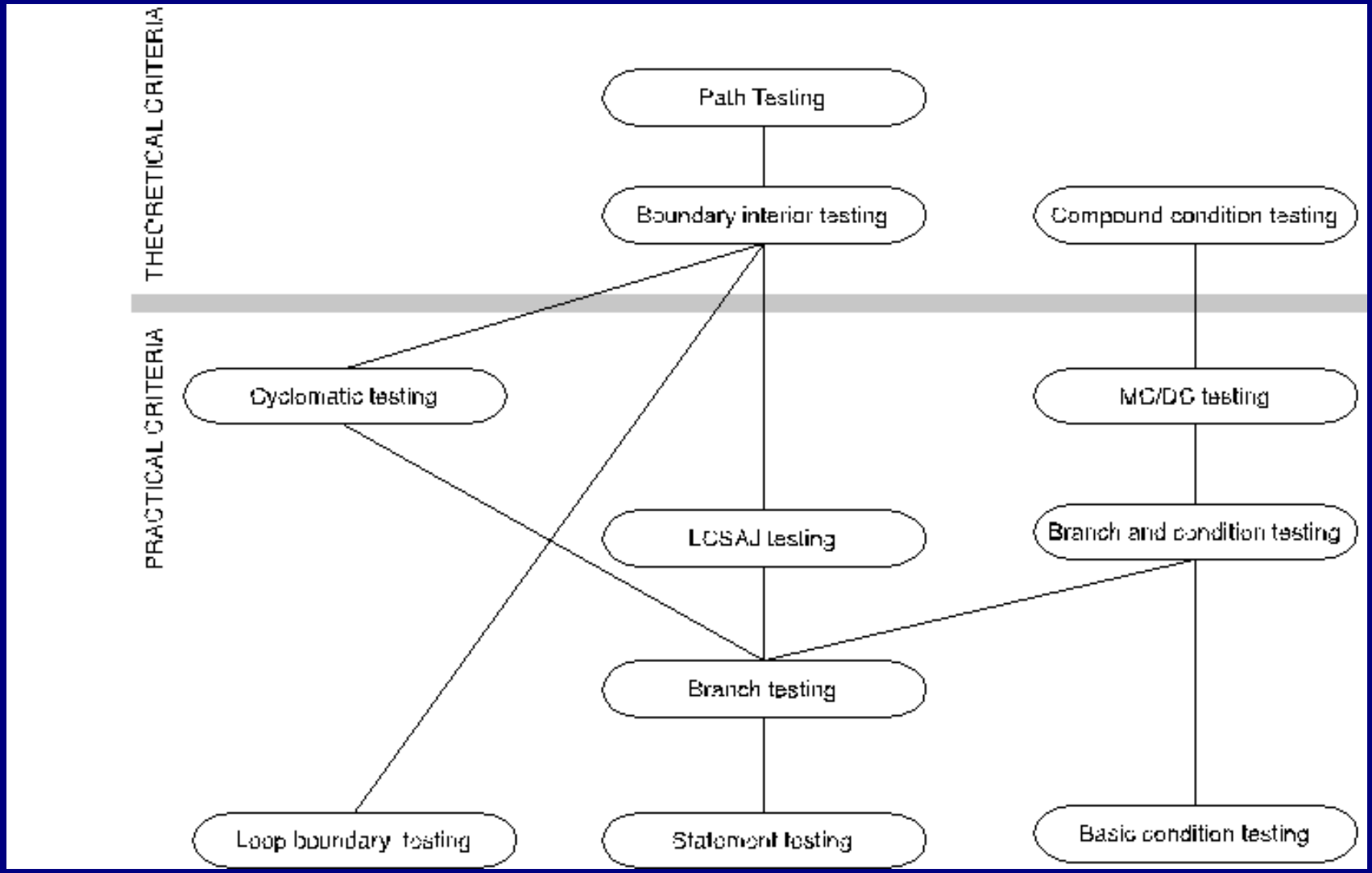
Towards procedure call testing

- The criteria considered to this point measure coverage of control flow within individual procedures.
  - not well suited to integration or system testing
- Choose a coverage granularity commensurate with the granularity of testing
  - if unit testing has been effective, then faults that remain to be found in integration testing will be primarily interface faults, and testing effort should focus on interfaces between units rather than their internal details

Procedure call testing

- Procedure entry and exit testing
  - procedure may have multiple entry points (e.g., Fortran) and multiple exit points
- Call coverage
  - The same entry point may be called from many points

Subsumption relation



Satisfying structural criteria

- Sometimes criteria may not be satisfiable
- The criterion requires execution of
  - statements that cannot be executed as a result of
    - defensive programming
    - code reuse (reusing code that is more general than strictly required for the application)
  - conditions that cannot be satisfied as a result of
    - interdependent conditions
  - paths that cannot be executed as a result of
    - interdependent decisions

Satisfying structural criteria

- Large amounts of fossil code may indicate serious maintainability problems
  - But some unreachable code is common even in well-designed, well-maintained systems
- Solutions:
  - make allowances by setting a coverage goal less than 100%
  - require justification of elements left uncovered
    - RTCA-DO-178B and EUROCAE ED-12B for modified MC/DC

Summary

- We defined a number of adequacy criteria
  - NOT test design techniques!
- Different criteria address different classes of errors
- Full coverage is usually unattainable
  - Remember that attainability is an undecidable problem!
- ... and when attainable, “inversion” is usually hard
  - How do I find program inputs allowing to cover something buried deeply in the CFG?
  - Automated support (e.g., symbolic execution) may be necessary
- Therefore, rather than requiring full adequacy, the “degree of adequacy” of a test suite is estimated by coverage measures
  - May drive test improvement

Book Assignment

Choose and complete any two Chapter 12 Exercises (pg 233-234)
Due in the dropbox by March 12, 2014 2359
Note that is a Wednesday!

Reading Assignment

Chapter 13 & 14