Intro to software testing

CSC 236

[Created from slides by K. Anderson and D. Lasher]

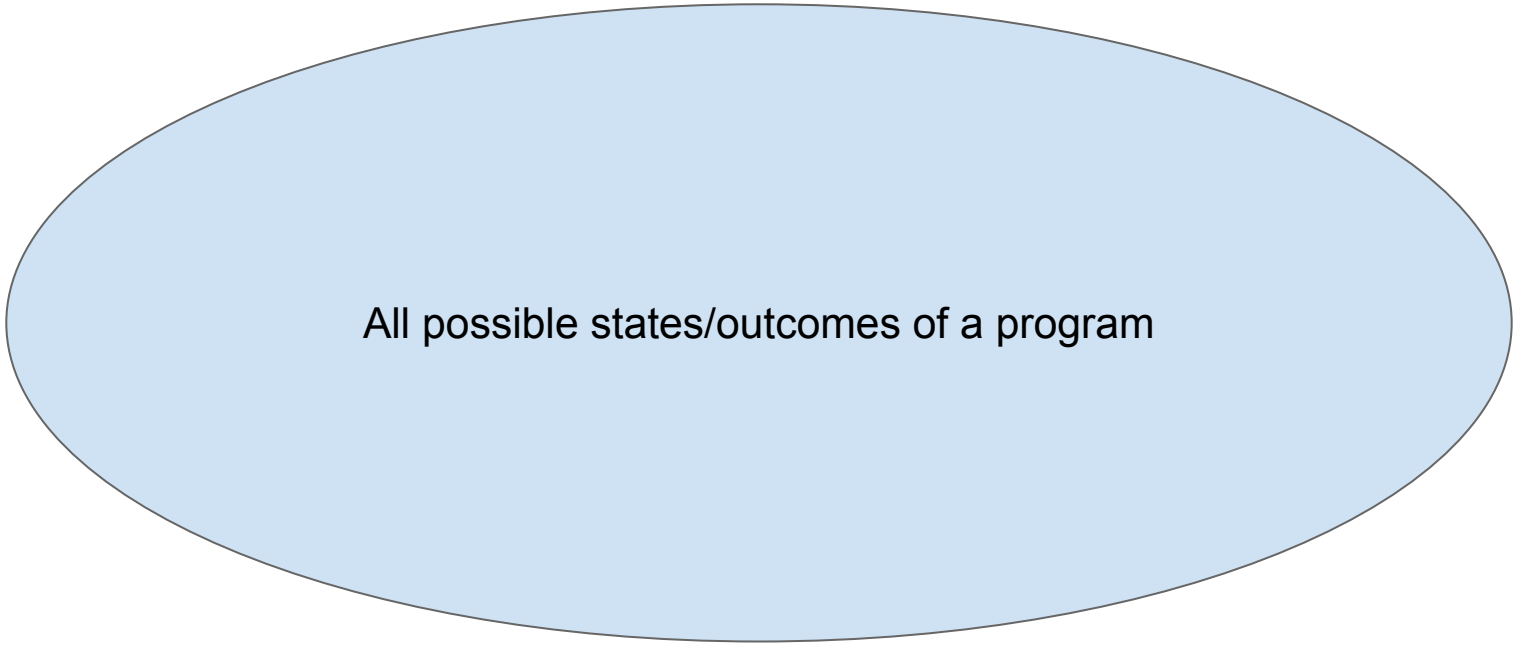
Fundamental concepts of software testing

- Terminology
- Testing of Systems
  - unit tests, integration tests, system tests, acceptance tests
- Testing of Code
  - Black Box
  - Gray Box
  - White Box
  - Code Coverage

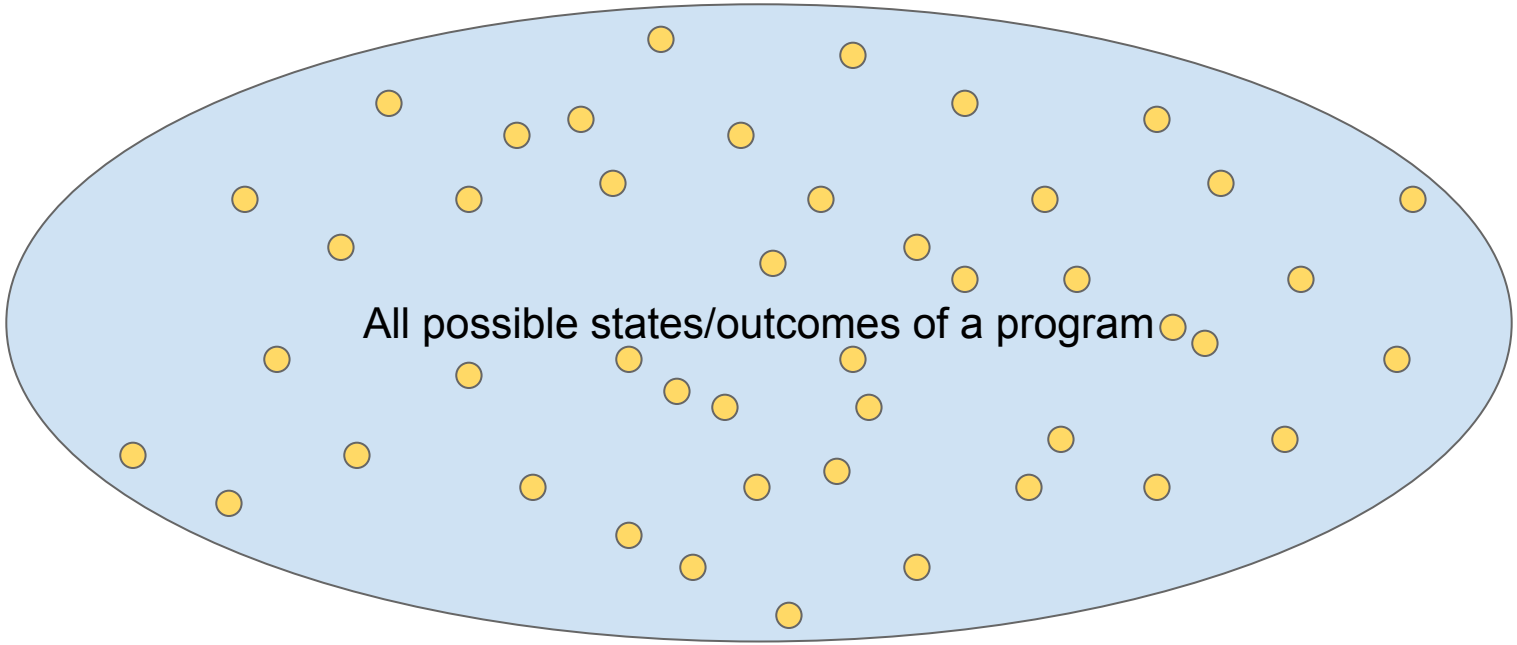
Testing

- Critical element of software development life cycles
  - Software quality control or software quality assurance
- Goals: validation and verification
- Validation:
  - Are we building the right product?
- Verification:
  - Does “X” meet its specification?
  - Where “X” can be code, a model, a design diagram, a requirement, ...
- At each stage, we need to verify that the thing we produce accurately represents its specification

Looking for faults

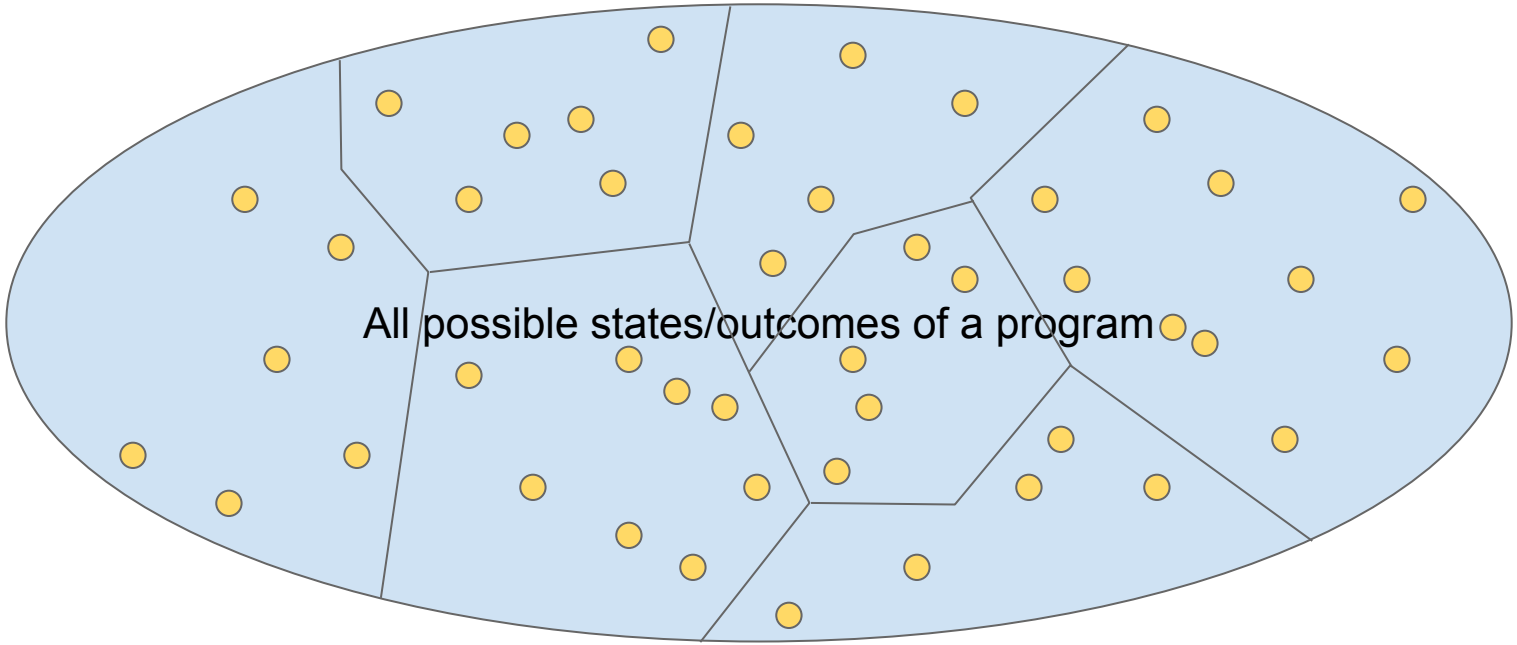


Looking for faults



Tests sample behavior; check results

Looking for faults



Fold space into equivalent behaviors; sample each partition

Example

int gcd(int x, int y)

- Function has an infinite* number of test cases
- Fold the space
  - Common case: x=6 y=9, returns 3
  - X is GCD: x=2 y=4, returns 2
  - Primes: x=3 y=5, returns 1
  - Zero: x=9 y=0, returns ?
  - Negative: x=-3 y=9, returns ?

*** Not really infinite for any actual int type, but we can still think of it as having a really larger domain.**

Completeness

- **Completely** testing a system is (generally) not feasible
- Settle for methodology
  - Fold the space into different functional partitions
  - Create tests that sample the behavior/output for each functional partition
- Coverage
  - Statements — straightforward
  - Loops — how many tests
  - Paths — hard, critical
  - Guide to quality of testing

Continuous Testing

- Performed in all stages software development
- During requirements gathering, for instance, we must continually query the user, “Did we get this right?”
- Iteration:
  - Throughout a development life cycle
  - At the end of each iteration
  - Check to see solution meets requirements (specification)

Testing the System

- Unit Tests
  - Unit = module, class, component, ...
  - Tests each unit independently
  - Tests each function / method of each unit
- Integration Tests
  - Check that modules work together in combination
  - Most projects are on schedule until they hit this point (MMM, Brooks)
  - All sorts of hidden assumptions are surfaced when code written by different developers is integrated

Mars polar lander

- "... embarrassing failure to convert English units ... to metric."
- "... mistake somehow escaped what is supposed to be a rigorous error-checking process."
- "The bad numbers had been used ever since the spacecraft's launch [9 months ago], but the effect was so small that is went unnoticed."

Metric-English mix-up doomed Mars probe

THE ASSOCIATED PRESS

LOS ANGELES — The $125 million spacecraft that was destroyed on a mission to Mars was probably doomed by the embarrassing failure to convert English units of measurement to metric ones, NASA said Thursday.

The Mars Climate Orbiter flew too close to Mars and is believed to have broken apart or burned up in the atmosphere.

NASA said the English-vs.-metric mixup apparently caused the navigation error. The company that built the spacecraft acknowledged it made the error.

The mistake was particularly embarrassing because the spacecraft had successfully flown 416 million miles over 9½ months before its disappearance Sept. 23 just as it was about to go into orbit around the Red Planet.

Agency officials said the mistake somehow escaped what is supposed to be a rigorous error-checking process. A report is expected in mid-November, which would be soon enough to fix any possible similar problems with a spacecraft already en route to Mars.

"It does not make us feel good that this happened," said Tom Gavin of NASA's Jet Propulsion Laboratory. "This mix-up has caused us to look at our entire end-to-end process. We will get to the bottom of this."

JPL said that its preliminary findings showed that Lockheed Martin Astronautics in Colorado submitted acceleration data in English units of pounds of force instead of the metric unit called newtons. At JPL, the numbers were entered into a computer that assumed metric measurements.

"In our previous Mars missions, we have always used metric," Gavin said.

The numbers were used in figuring the force of thruster firings used by the spacecraft to adjust its position.

The bad numbers had been used ever since the spacecraft's launch last December, but the effect was so small that it went unnoticed. The difference added up over the months as the spacecraft journeyed toward Mars.

Gavin said he does not expect the error to affect NASA's relationship with Lockheed Martin Astronautics, which has built several probes for the space agency.

Lockheed Martin acknowledged the mistake.

"We should have had them in metric units," said Noel Hinners, vice president of flight systems for Lockheed Martin Astronautics in Denver.

The Mars Climate Orbiter was on a mission to study the Red Planet's weather and look for signs of water — information key to understanding whether life ever existed or can exist there. It carried cameras along with equipment for measuring temperature, dust, water vapor and clouds.

The orbiter's sibling spacecraft, Mars Polar Lander, is set to arrive Dec. 3. Gavin said investigators are trying to determine whether NASA made the same mistake with that spacecraft. The problem could be fixed if it did occur.

The Mars Polar Lander will study Mars' climate history and weather with the goal of finding what happened to water on the planet. It is equipped with a robotic arm that will collect samples for testing inside the spacecraft.

THE NEWS & OBSERVER
FRIDAY, OCTOBER 1, 1999

Tesla

- First death in self-driving mode
- Driver was watching Harry Potter
- Rural Texas highway
- At intersection
  - Tractor trailer made a left turn into oncoming lane
  - The Telsa crested hill and “saw” trailer in lane
  - Went under trailer @ >70mph; removed roof
  - Didn’t hit apply brakes until off the road
- Was not able to identify trailer
  - Could have “looked” like the sky or
  - Billboard

Types of testing

- Black Box Testing
  - Does not examine code
  - Compare behavior to specification
- Grey Box Testing
  - Some insight into code
  - Code not “examined” in detail
  - Compare behavior to specification
- White Box Testing
  - Complete* access to code
  - Tests written based on code (as well as specification)
  - Coverage

* **In-house code. May not have code for libraries.**

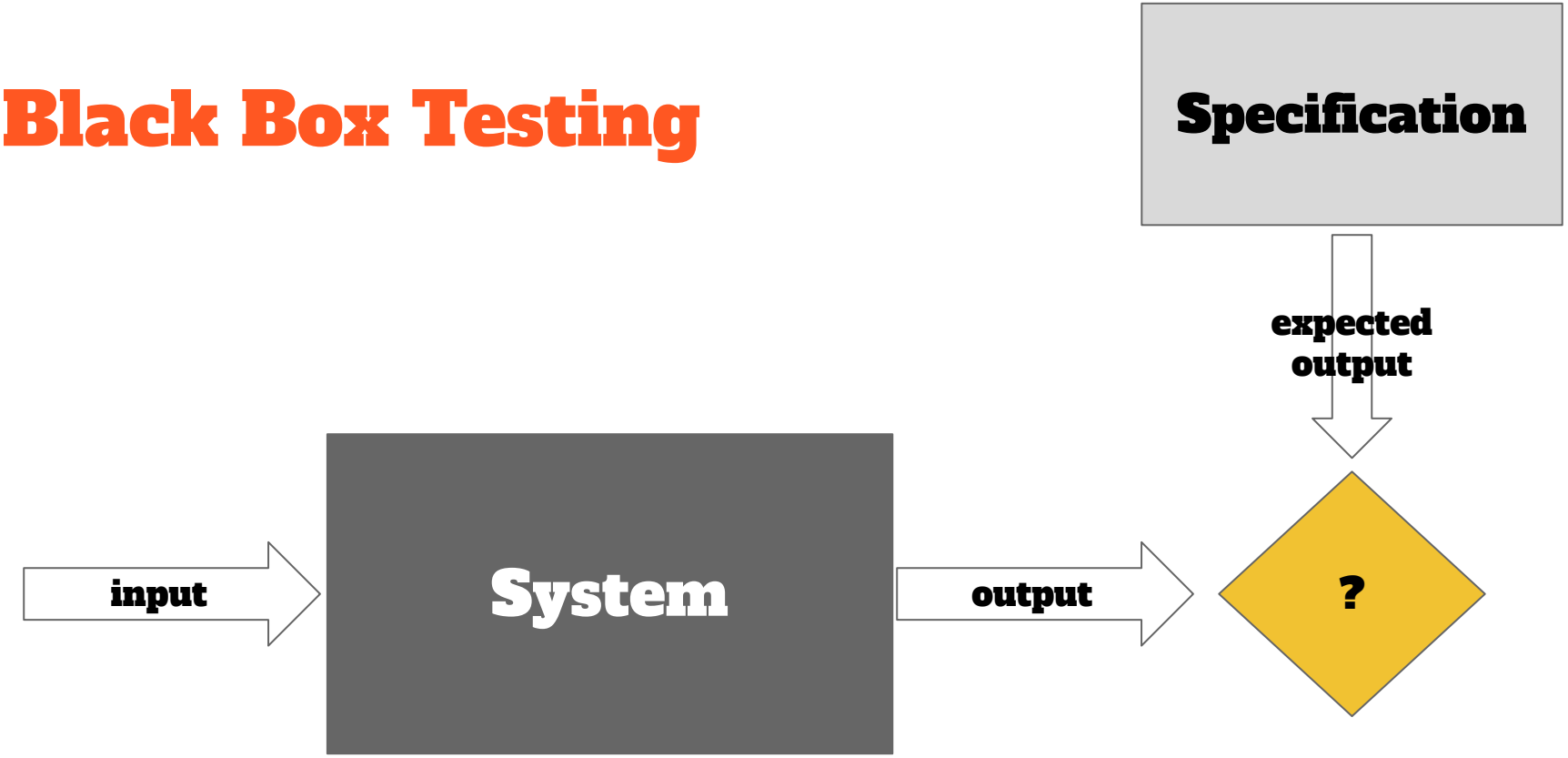
What is a test case?

- Input

  and

- Expected output

Black Box Testing



black box test passes input to a system; compares output to expected output

Black box test categories

- User input validation (based off specification)
- Output results
- State transitions
  - Are there clear states in the system in which the system is supposed to behave differently based on the state?
- Boundary cases and off-by-one errors

Gray box testing

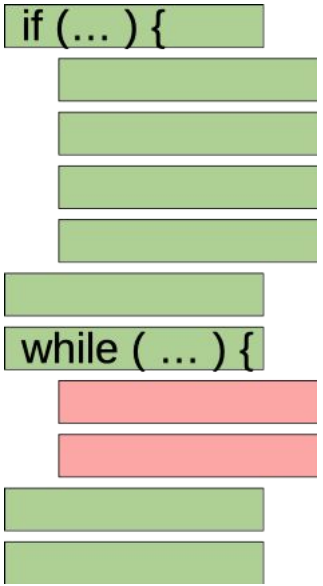
- Knowledge of code ⇒ more complete set of black box tests
- Verifying auditing and logging information
  - Check if internal state is correct
- System-added information (timestamps, checksums, etc.)
- Is the system correctly cleaning up after itself
  - temporary files, memory leaks, data duplication/deletion

White Box Testing

- Write test cases with complete knowledge of code
  - Format is for a test is the same: input, expected output, actual output
- Includes
  - Code coverage
  - Proper error handling
  - Works as documented (ie, is method thread safe?)
  - Proper handling of resources
    - How does the software behave when resources become constrained?

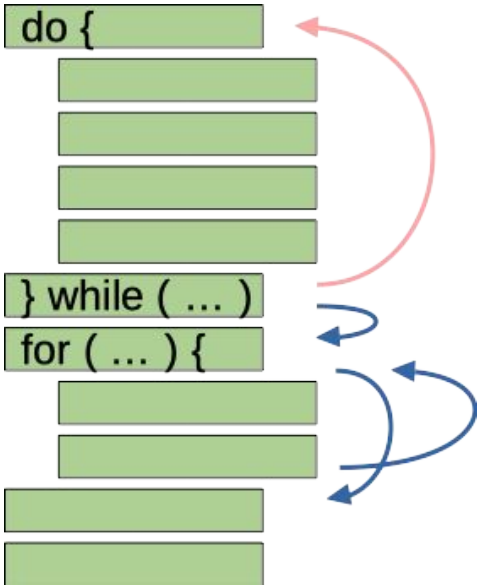
Code Coverage

- Statement coverage
  - All statements have been executed at least once



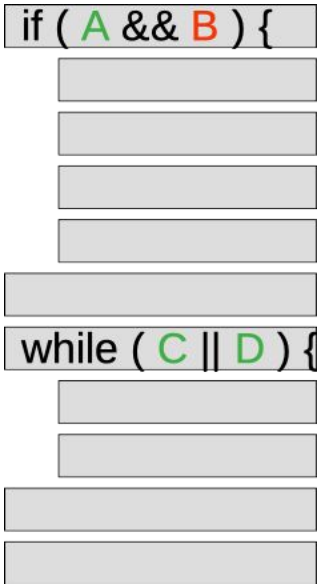
Code Coverage

- Branch coverage (a.k.a., edge coverage)
  - Each edge in control flow graph has been taken at least once



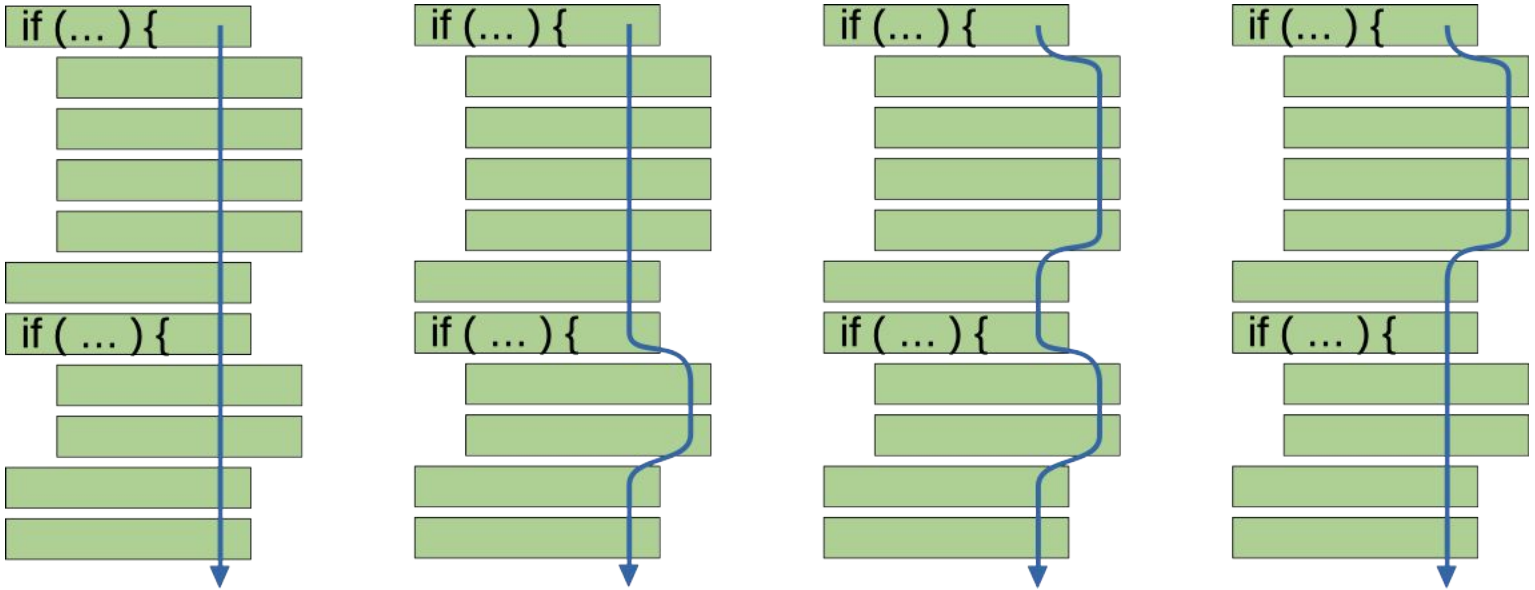
Code Coverage

- Condition coverage
  - Like branch coverage but looking inside each conditional
  - Ensure that all parts of a conditional have been executed (short-circuiting might make this non-trivial)
  - For a compound conditional, ensure that all subexpressions have yielded true and false.



Code Coverage

- Path coverage
  - Test all paths in control flow graph



Purpose of testing

- To find errors
- The **intent** is errors
- Cannot just use the program
- Must stress program

Performance testing

- Previously, discussed correct result
- Execution time can be a requirement
- Test succeeds iff
  - Result is correct **and**
  - On time
- As features are added, code slows down
  - Test that was successful, now fails
  - Sometimes called “regression”
  - Major effort in mature, evolving code
  - One reason why test plans should be reusable