Software Testing, Quality Assurance & Maintenance—Lecture 36

Patrick Lam

April 6, 2015

Summary: Useful Terms

- software faults, errors and failures;
- tests and test requirements; and
- coverage criteria and subsumption.

Summary: Theory

Key Coverage Criteria:

- graph coverage;
- logic coverage;
- syntax-based coverage
  (and mutation and fuzzing); and
- input-space coverage.

Evaluate test suites against these criteria.
(guides test suite construction.)

Summary: Practice

We gained experience with:

- using tools for unit testing (JUnit) &
  bug detection (Valgrind)
- test design:
  testability, badly designed tests, self-checking tests;
- writing an invariant detection and bug detection tool (using LLVM and the idea behind Coverity);
- using the state-of-the-art tool Coverity for bug finding;
- writing good bug reports.

and learned about automated testing and bug detection tools, regression testing, testing for concurrency (Helgrind), and state-of-the-art techniques (Daikon, Coverity, iComment).

About the Final

- Tuesday, April 21, 2015, 12:30-3:00, PAC.
- Open-book, open note exam.
- You may consult any printed material (books, slides, notes, etc).
- No electronic devices.

Review: Input Space Partitioning

- Can’t feed all inputs to the program—
  test a representative set.
- Input space partitioning makes this idea more formal: test one input from each partition.
- Two properties for partitions:

  Completeness

  Disjointness

Review: Input Domain Models

Require creativity and analysis to formulate.

Two general approaches:

- Interface-based, using the input space directly, or
- Functionality-based, using a functional or behavioural view of the program.

Review: Cross-checking program beliefs (MUST)

- MUST beliefs: inferred from acts that imply beliefs code **must** have.

  ```
  x = *p / z;
  // MUST belief: p not null
  // MUST: z != 0
  unlock(l);
  // MUST: l acquired
  x++;
  // MUST: x not protected by l
  ```

  Check using internal consistency:
  infer beliefs at different locations, then cross-check for contradiction.

Review: Cross-checking program beliefs (MAY)

- MAY beliefs: Could be coincidental.
  Inferred from acts that imply beliefs code **may** have.

```
A(); A(); A(); A();
...  ...  ...  ...
B(); B(); B(); B();
// MAY: A() and B()
// must be paired
```

Check as MUST beliefs; rank errors by belief confidence.

Review: Static vs Dynamic Analysis

Static analysis:

- Conceptually, can find everything.
- Problems: pointer aliasing, false positives.

Dynamic analysis:

- Imposes run-time overhead.
- Depends on the quality of inputs/environments.
- May report false negatives.

Review: Race conditions and Deadlocks

Race condition:
two concurrent accesses to the same state,
at least one of which is a write.

Deadlock:
cyclic structure between lock acquisitions
that will never succeed.

Review: Testing Concurrent Programs

Some options:

- Run them multiple times.
- Add noise: sleep, background load, ...
- Use tools: Helgrind, etc.
- Force different scheduling.

Some of these options use the following technologies:

- lock-set;
- happens-before;
- other state-of-art techniques.

Review: Regression Tests

- Gotta automate them.
- Have enough regression tests:
  - Too few: you'll miss bugs.
  - Too many: maintenance overhead, and if badly designed, too slow to run.
- Keep them up-to-date.

Review: Output validation

How do you know your test outputs are correct?

- Hope for the best.
- Manual effort.
- Compute multiple ways.
- Checksums, redundant data, etc.

Review: Syntax-based Testing

	Program-based	Input Space
Grammar	Programming language	Input languages / XML
Summary	Mutates programs / tests integration	Input space testing
Use Ground String?	Yes (compare outputs)	No
Use Valid Strings Only?	Yes (mutants must compile)	Invalid only
Tests	Mutants are not tests	Mutants are tests
Killing	Generate tests by killing	Not applicable

Strong and weak mutants.

Sample Questions

- Give an example of a prime path.
- Exhibit a case where a clause always determines a predicate.
- Exhibit a case where a clause never determines a predicate.
- Why might GACC be infeasible?
- What causes node coverage to be infeasible?
- Give an example of a *du*-path.
- Given a function, draw the CFG. Identify the def-set and use-set of each node, and list the TR for ADC and AUC.
- The exercise questions seen in class.

More Sample Questions

- Read given program, give an input that strongly kills this mutant.
- Enumerate requirements for Prime Path Coverage on given example. If PPC is feasible, provide a test set which achieves it. Otherwise, provide the test set that you feel comes closest to achieving PPC.

Input Space Partitioning: Sample Questions

- Propose an input space partition for this method (use a functionality-based input domain model):

```
static boolean isPrime (int n) {
  if (n<=1) return false;
  for (int i = 2; i <= (int)(Math.sqrt(n))
     ; i++)
    if (n % i == 0)
      return false;
  return true;
}
```

Best of luck

~~after Convocation~~

on your work term.