Exploiting Synergy Between Testing and Inferred Partial Specifications

Tao Xie       David Notkin

Department of Computer Science & Engineering
University of Washington
May 9, 2003
Workshop on Dynamic Analysis (WODA 2003)
Outline

- Background
- Synergy issues
- Application
- Why it will fail
- Why it will succeed
Background

- **Test case generation**, e.g. Korat [BKM 02], Jtest [ParaSoft], AsmL [MSR]
- **Test oracle generation**, e.g. Korat, Jtest, JML+JUnit [CL 01]
- **Test selection/coverage criteria**, e.g. ADLscope [CR 99], UMLTest [OA 99]
- **Likely spec Inference based on test executions**, e.g. Daikon operational abstraction [ECGN 01], Strauss [ABL 02], Hastings [WML 02]
Synergy Issue: Chicken-and-Egg I

Spec-based test generation

(likely) Specs → Tests

Dynamic likely spec inference

- Win-win feedback loop: better spec ↔ better tests?
- Chicken and egg problem?
Synergy Issue: Chicken-and-Egg II

Spec-based test generation

(likely) Specs \(\rightarrow\) Tests

Dynamic likely spec inference

- **Initial tests** \(T\) (manually written tests, automatically generated tests w/o specs, etc.)
- Likely specs \(S\) inferred from \(T\)
- Tests \(T'\) generated based on \(S\)
- Executions of \(T'\) \(\rightarrow select\) a subset of \(T'\)
  
  \[\text{Test augmentation: } T = T \cup \text{the subset of } T'\]

  **Better tests**

- Likely specs \(S\) inferred from \(T\)

  **Better specs**
Executions of Tests Generated From Likely Specifications -I

Universal

Input domain

Legal inputs

Stronger inferred pre

Stronger inferred post

Inferred precondition constrained domain

Method Execution

Output domain

Legal outputs

Inferred postcondition constrained domain

Postcondition violation (exercise a new feature)

Postcondition violation (expose a fault)
Executions of Tests Generated From Likely Specifications -II

Method Execution

Input domain
- Inferred precondition
  - Constrained domain
- Weaker inferred pre
- Stronger inferred post

Output domain
- Inferred postcondition
  - Constrained domain
- Postcondition violation
  - (exercise a new feature)
- Postcondition violation
  - (narrow down precondition)
- Postcondition violation
  - (expose a fault)
Executions of Tests Generated From Likely Specifications -III

- Legal inputs
- Legal outputs
- Inferred precondition
- Inferred postcondition
- Stronger inferred precondition
- Weaker inferred postcondition
- Method Execution
- Inferred precondition constrained domain
- Inferred postcondition constrained domain
- Postcondition violation (expose a fault)
Executions of Tests Generated From Likely Specifications -IV

- Legal inputs
- Input domain
- Method Execution
- Weaker inferred pre
- Weaker inferred post
- Output domain
- Inferred precondition constrained domain
- Inferred postcondition constrained domain
- Postcondition violation (narrow down precondition)
- Postcondition violation (expose a fault)
Handling Synergy Issues

- **Precondition guard removal**
  - Too restrictive preconditions may leave (maybe important) legal unit inputs untested

- **Iterations until reaching a fixed point**
  - Add new violating tests (legal inputs) to the existing test suite for spec inference in next cycle
  - Add stronger preconditions manually
Application: Spec-Violation Approach to Unit Test Data Selection

- Problem
  - Insufficiency of the manually maintained unit test suite A (small number)
  - Oracle unavailability of the automatically generated unit test suite B (large number)
- Goal: Selectively augment A with a small (most valuable) subset of B
- Related work: Operational Difference [HME 03], DIDUCE [HL 02]
Why it will fail

- Not enough inferred postconditions to violate
  - Improved inference techniques can help
- Precondition guard removal might induce false positives
  - Precondition guard relaxation can help
- Postcondition violations are due to limited test data value range uninteresting to testers
- Manually commenting out violated specs is tedious
  - Improved Jtest to support it can help
Why it will succeed

- Without a priori specification, there are few effective black box unit test data selection approaches.
- Violating tests can guarantee to exercise a new program feature.
- The violated specs for the corresponding violating tests can help developers to make selection decision easily.
- The approach can be largely automated.