Mutually Enhancing Test Generation and Specification Inference

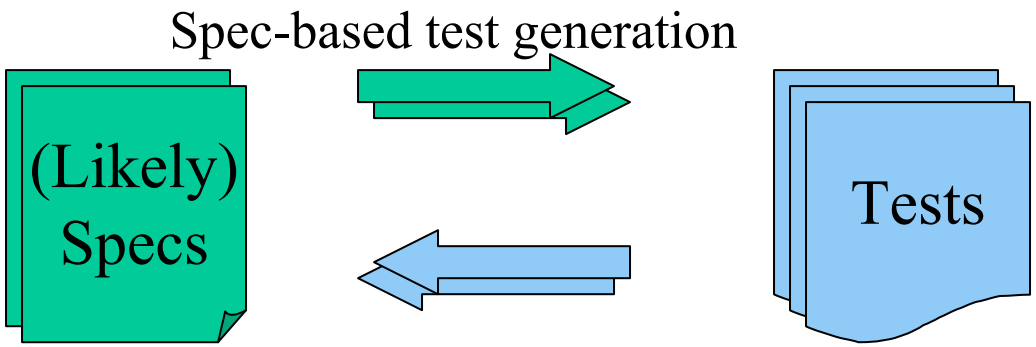
Tao Xie David Notkin

Department of Computer Science & Engineering
University of Washington, Seattle, WA

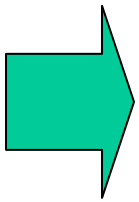
Oct. 6th, 2003
FATES 2003, Montreal, Canada

Synopsis

- Need specs for (many kinds of) test generation
- Need tests for dynamic spec inference
- We have applied feedback loop between these approaches that
  - aids in test generation
    - improving specs and helping in producing oracles
  - aids in spec inference
    - improving the underlying test suites



Outline



- Motivation
- Feedback Loop between Test Generation and Spec Inference
- Related Work
- Conclusion

Test Generation

- White-box test generation
  + Cover structural entities, e.g. statement, branch
  – **Test oracle problem**
    - Rely on uncaught runtime exceptions or program crashes
- Black-box test generation
  + Use specs to guide test generation
  + Use specs to produce test oracles
  – **Lack of specs problem**

Dynamic Spec Inference

- A formal specification: desired behavior and written before code
- An operational abstraction: observed behavior and induced dynamically from program executions [Harder et al. 03]
- Operational abstraction $\overset{\text{in form}}{\approx}$ formal specification

Dynamic spec inference ≈ operational abstraction generation

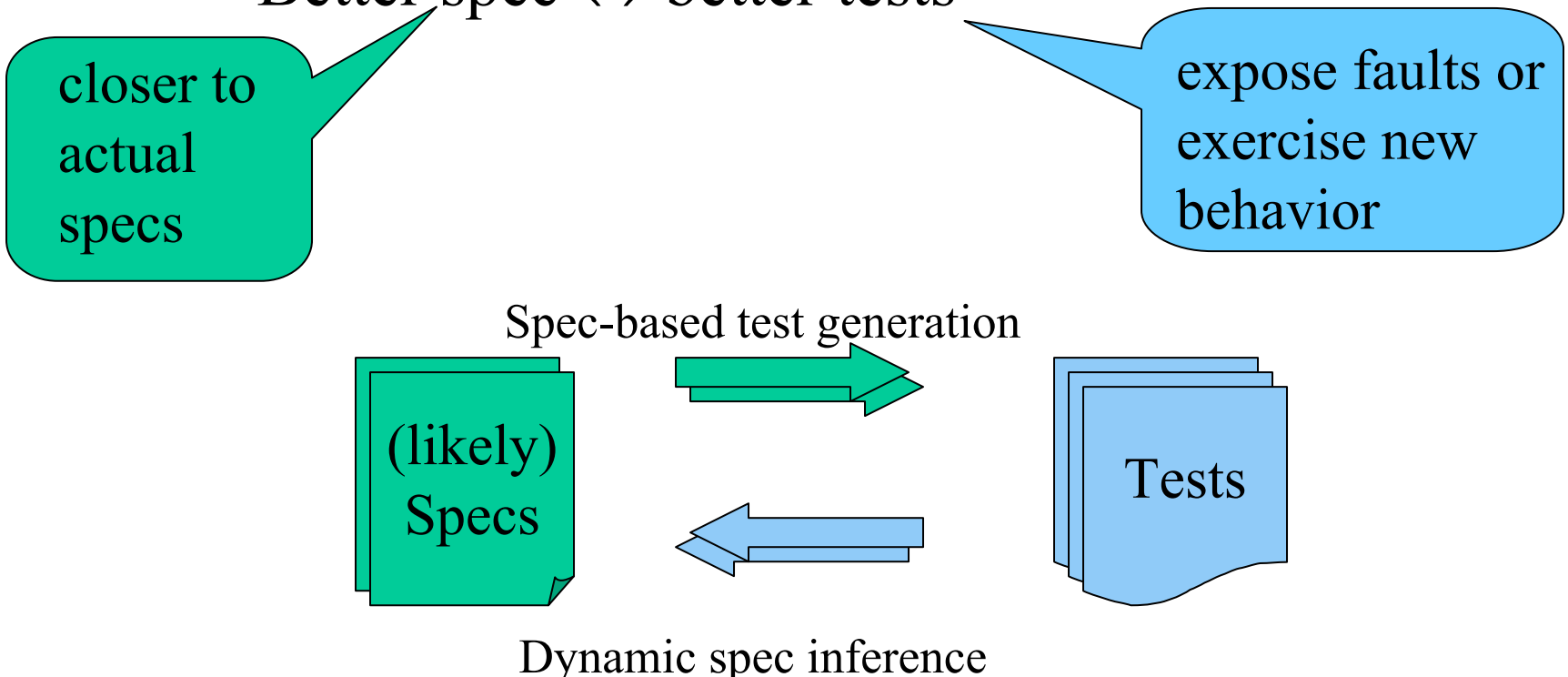
Running insufficient tests produce low-quality specs

Dynamic Spec Inference (cont.)

- Axiomatic specification inference
  - Daikon [Ernst et al. 01]
- Algebraic specification inference
  - [Henkel & Diwan 03]
- Protocol specification inference
  - Strauss [Ammons et al. 02], Hastings [Whaley et al. 02], [Hagerer et al. 02]

Circular Dependency

- Circular dependency: spec-based test generation and dynamic spec inference
- Solution: win-win feedback loop
  - Better spec ⇔ better tests



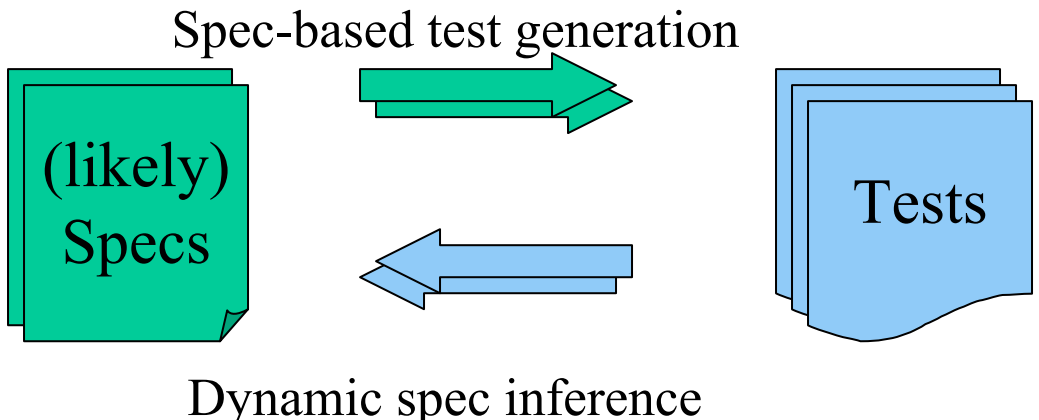
Feedback Loop

- Inferred Specs → Test Generation
  - Reduce the scope of analysis
- Generated Tests → Spec Inference
  - Verify/refine the inferred specs
- Spec-violating Tests → Test Selection
  - Inspection and test augmentation

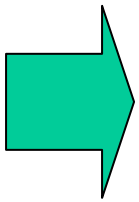
Lack of specs problem

Insufficient test problem

Test oracle problem



Outline



- Background
- Feedback Loop between Test Generation and Spec Inference
- Related Work
- Conclusion

Feedback Loop Framework

The existing test suite (manual tests)

Selected tests

Automatically generated test inputs

Test Selection

Test Generation

Program

Automatically generated test inputs

Likely specs

Data trace

Trace Collection

Spec Inference

Test Generation Based on Inferred Specs

- Generate test inputs
  - preconditions (axiomatic specs)
  - transitions (protocol specs)
  - axioms (algebraic specs)
- Over-constrained specs
  - leave (maybe important) legal inputs untested
- Solutions:
  - remove precondition guards (axiomatic specs)
  - complement-transitions (protocol specs)
  - other method call pairs (algebraic specs)

Test Selection - I

- Execution of a test input:
  - no specification violations/exceptions
  - throw uncaught exceptions
    - A fault or illegal input
  - violate over-constrained postconditions/axioms
    - Existing test inputs are insufficient
  - violate true postconditions/axioms
    - A fault or illegal input

Test Selection - II

- Human inspection:
  - illegal input:
    - Add preconditions or adopt defensive programming
  - legal input:
    - Fault exposure: fix bug and augment regression test suite
    - Normal, but new feature being exercised: augment regression test suite
- Complementary technique: select tests exercising new structural entities
- We observed that the number of selected test was relatively small

Iterations

- Iterates until
  - no spec violations or inferred specs remain the same
  - user-specified max # iteration has been reached

- In the subsequent iteration, spec inference is based on:
  - the existing test suite augmented by
    - selected violating tests
    - all generated tests

Outline

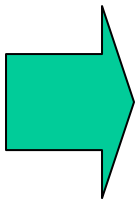
- Motivation
- Feedback Loop between Test Generation and Spec Inference
- Related Work
- Conclusion

Related Work

- Feedback loop between static analyses
  - Program abstraction and model checking [Ball et al. 01]
  - Annotation guess and theorem proving [Flanagan et al. 01]
  - Assumption generation and model checking [Giannakopoulou et al. 02, 03]
- Feedback loop between static and dynamic analyses
  - Finite state verification and testing [Naumovich et al. 00]
- Other feedback loops:
  - Test data generation and branch predicate constraint solving [Gupta et al. 98]
  - Automata learning and testing/model checking [Peled et al. 99, 02]

Outline

- Motivation
- Feedback Loop between Test Generation and Spec Inference
- Related Work
- Conclusion



Conclusion

- Feedback loop between test generation and spec inference
  - Axiomatic specs (integration of Daikon and Jtest) [ASE 03]
  - Algebraic/protocol specs (ongoing work)
- Aids in test generation
  - improving specs and helping in producing oracles
- Aids in spec inference
  - improving the underlying test suites

Questions?