Static and dynamic analysis: synergy and duality

Michael Ernst

MIT Lab for Computer Science

http://pag.lcs.mit.edu/~mernst/

Goals and outline

- Theme: static and dynamic analyses are less different than many people believe
- Goal: encourage blending of the two techniques and communities
- Outline
  - Review of static and dynamic analysis
  - Combining them: aggregation, analogies, hybrids
  - Observation: both examine a subset of executions

Static analysis

- Examine program text (only), reason over possible behaviors by building a model of program state
- Example: compiler optimizations

- Slow: models of state are large, so use abstraction
- Conservative: account for abstracted-away state
- Sound: (weak) properties are guaranteed to be true

Dynamic analysis

- Execute program, observe executions
- Examples: testing, profiling

- Fast: as quick as execution (over a test suite)
  - Example: aliasing
- Precise: no abstraction or approximation
- Unsound: results may not generalize to future executions

Static analysis	Dynamic analysis
Slow	Fast
use abstraction	simple execution
Conservative	Precise
due to abstraction	no approximation
Sound	Unsound
due to conservatism	does not generalize

Research agendas

- Static analysis: choose good abstractions
  - Less useful for applications that require precision
- Dynamic analysis: choose good tests
  - Less useful for applications that require correctness
  - Many domains do not require correctness!

Combining static and dynamic analysis

1. Aggregation: pre- or post-processing
   - Profile-directed compilation
   - Reduce instrumentation requirements
2. Inspiring analogous analyses
3. Hybrid analyses that blend both approaches

Analogous analyses

- Static and dynamic slicing
- Memory checking
  - Purify [Hastings 92]: run-time tagged memory; each instruction checks/updates the tags
  - LCLint [Evans 96]: compile-time dataflow analysis; each transfer function checks/updates the state
  - Essentially identical analyses!

More analogous analyses

- Specification checking
  - Statically: theorem-proving
  - Dynamically: `assert` statement
- Specification generation
  - Statically: by hand or abstract interpretation [Cousot 77]
  - Dynamically: by invariant detection [Ernst 99], reporting unfalsified properties
- Lesson: look for more gaps with no analogous analyses!

Hybrid analyses

Combine static and dynamic analyses

- Not mere aggregation, but a new analysis
- Disciplined trade-off between precision and soundness

Possible starting points

- Analyses that trade off run-time and precision
- Ignore some available information
  - Example: examine only some paths
- Merge based on observation that both examine only a subset of executions (next section of talk)
  - Problem: optimistic vs. pessimistic treatment

Examples: bounded model checking, security analyses, delta debugging, etc.

Sound dynamic analysis

- Observe every possible execution!
- Problem: infinite number of executions
- Solution: test case selection and generation
  - Efficiency tweaks to an algorithm that works perfectly in theory but exhausts resources in practice

Precise static analysis

- Reason over full program state!
- Problem: infinite number of executions
- Solution: data or execution abstraction
  - Efficiency tweaks to an algorithm that works perfectly in theory [Cousot 77] but exhausts resources in practice

Subsets of executions

- Dynamic analysis: executions in the test suite
  - Easy to enumerate, characterizes program use
- Static analysis: executions that induce particular data structures or control flow
  - Characterizes what program parts are exercised
  - Example: *k*-limiting [Jones 81]
- Each subset/characterization is better for certain uses
  - Characterize with respect to code or input/execution
- Combine them to notice analogies and to produce new analyses

Why this won’t work

- Analogies between analyses
  - What applications tolerate imprecision?
  - No more low-hanging fruit
  - Approaches too different
- Hybrid analyses
  - How to measure/trade-off precision and soundness
  - Optimistic vs. pessimistic treatment of unseen executions
- Subset characterization
  - How to characterize program executions
  - What is “partial soundness”? What is in between?

Why this might work

- Analogous analyses
    - Success in various domains
- Hybrid analyses
    - Existing analyses increasingly look like points in this continuum
- Subset characterization