

Combining Verification Approaches

<http://d3s.mff.cuni.cz>

Department of
Distributed and
Dependable
Systems



Pavel Parízek



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Verification approaches

- Model checking programs
 - Explicit state (Java Pathfinder)
 - Abstraction-based (CEGAR, ...)
- Symbolic execution (concolic testing)
- Deductive methods (Spec#/Boogie)
- Static analysis (data-flow, pointers)
- Abstract interpretation
- Dynamic analysis (runtime)

- Classical testing (e.g., JUnit)

Evaluation

- Advantages
 - Model checking
 - path-sensitive, very precise, does not scale well (state explosion)
 - Static analysis
 - explores all program behaviors, limited precision, highly scalable
- Limitations
 - Abstraction-based model checking and deductive methods
 - Problem with concurrency (limited support for threads)
 - Very good at checking properties related to data values
 - Explicit state model checking
 - Supports threads well (detecting concurrency errors)
 - Does not handle data non-determinism very well

Categories

- Search for errors
 - testing, symbolic execution, dynamic analysis
- Search for proofs
 - program model checking, deductive methods

Search for errors

- Program executed concretely on many inputs
 - Finds only real errors
 - Achieves small coverage
- Abstract execution tracking only some facts
 - Covers all the program paths
 - Reports many false positives
- Intermediate solutions
 - Example: directed concolic testing

Search for proofs

- Goal: find the safe over-approximation
- Model checking: reachable state space
- Deductive methods: inductive invariant
- Limitations
 - Verification procedure might not terminate
 - State explosion (many thread interleavings)
- Recent solutions: CEGAR

Bonus topics

- Combining tests and program verification
- Detecting some bugs in web applications
- Program termination and checking liveness
- Program synthesis: overview, current state

Combining tests and verification

- Search for errors and proofs at the same time
- Using results of one search also in the other
- Example: **SYNERGY**
 - B.S. Gulavani, T.A. Henzinger, Y. Kannan, A.V. Nori, and S.K. Rajamani. **SYNERGY: A New Algorithm for Property Checking**. SIGSOFT FSE 2006, ACM.

Example program

```
x = 0;
```

```
while (x < 1000) {  
    x = x + 1;  
}
```

```
assert (x > 1000);
```

Checking dynamic web applications

- Dynamic programming languages
 - Features: dynamically typed programs, `eval()`
- Implicit input parameters (GET, POST)
- Persistent state (database, cookies)
- Complex patterns of user interactions
- On-the-fly generating of source code
- Control flows through the HTML pages
 - forms, buttons, input events (keyboard, mouse)

Checking dynamic web applications

- Example: **Apollo**
 - S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A.M. Paradkar, and M.D. Ernst. **Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking**. IEEE Transactions on Software Engineering, 36(4), 2010.

Example program

```
<?php
    if (!isset($_GET['step'])) $step = 1;
    else $step = $_GET['step'];
    if ($_GET["login"] == 1) validateAuth();
    switch ($step) {
        case 1: require('login.php'); break;
        case 2: require('news.php'); break;
        case 3: require('inbox.php'); break;
        default: die("wrong input!");
    }
?>
```

Convergence

- Classic model checking
 - Program model: abstract reachability tree
 - Path-sensitive: never joins different paths
- Static program analysis
 - Program model: control flow graph (inter-proc)
 - Path-insensitive: losing precision at join points

Generalization

- Abstract domain
 - Transfer functions
 - Merge operator
 - Termination check
-
- Based on this research paper
 - D. Beyer, T. A. Henzinger, and G. Theoduloz.
Configurable Software Verification: Concretizing the Convergence of Model Checking and Program Analysis. CAV 2007, LNCS 4590.