Program Analysis

Symbolic and Concolic Execution (Part 4)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2020/2021

Overview

1. Classical Symbolic Execution
2. Challenges of Symbolic Execution
3. Concolic Testing
4. Large-Scale Application in Practice ←

Mostly based on these papers:

- *DART: directed automated random testing*, Godefroid et al., PLDI'05
- *KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs*, Cadar et al., OSDI'08
- *Automated Whitebox Fuzz Testing*, Godefroid et al., NDSS'08

Large-Scale Concolic Testing

- SAGE: Concolic testing tool developed at Microsoft Research
- Test robustness against unexpected inputs read from files, e.g.,
  - Audio files read by media player
  - Office documents read by MS Office
- Start with known input files and handle bytes read from files as symbolic input
- Use concolic execution to compute variants of these files

Large-Scale Concolic Testing (2)

- Applied to hundreds of applications
- Over 400 machine years of computation from 2007 to 2012
- Found hundreds of bugs, including many security vulnerabilties
  - One third of all the bugs discovered by file fuzzing during the development of Microsoft's Windows 7

Details: Bounimova et al., ICSE 2013

Summary: Symbolic & Concolic Testing

Solver-supported, whitebox testing

- Reason symbolically about (parts of) inputs
- Create new inputs that cover not yet explored paths
- More systematic but also more expensive than random and fuzz testing
- Open challenges
  - Effective exploration of huge search space
  - Other applications of constraint-based program analysis, e.g., debugging and automated program repair