Symbolic Execution Tools for Software Testing

0368-3526

Noam Rinetzky

Sundays* 0900-1100

Kaplun 324

Preliminaries

- Students will group in teams of 2-3 students.
- Each group will do one of the projects presented.

Administration

- Workshop meetings will take place **only** on Sundays 09-11
  - No classes during other hours

- **Attendance** in all meetings is **mandatory**

- Grading: 100% of grade will be given after final project submission.
- Projects will be graded based on:
  - Code correctness and functionality
  - Original and innovative ideas
  - Level of technical difficulty of solution

Administration

- Workshop staff (me) should be contacted by email.
  - Noam Rinetzky - maon@cs.tau.ac.il
- Follow updates on the workshop website:
  http://www.cs.tau.ac.il/~maon/teaching/workshop.html

Tentative Schedule

- Meeting 1 29/10/2017 (today)
  - Introduction to symbolic execution
  - Home assignment: Tutorial 1-3 in http://klee.github.io/tutorials/
- Meeting 2 12/11/2017
  - Projects description
  - Project suggestions
- Meeting 3, 26/11/2017
  - Each group presents its project & plan
- Meeting 4, 24/12/2017
  - Progress report
- Meeting 5, 21/1/2018
  - First phase submission

- Submission: 4/3/2018
- Presentation: 15/3/2018
  - Each group separately

Your Ideas

Suggested Project 1: Search

- New search heuristics
  - Study existing
    - E.g., random, BFS, DFS, directed
  - Come up with new ones

- Steering Symbolic Execution to Less Traveled Paths [Li et al. OOPSLA'13]

Suggested Project 2: EVM

- Smart contracts run on a VM
- Most popular: EVM (Ethereum)
- Build an SE engine for EVM

- Making Smart Contracts Smarter [Luu et al. CCS'16]

Suggested Project 3: Multisolver

- KLEE works with multiple solvers
  - STP or Z3
- Extend KLEE to work with multiple solvers simultaneously
  - Competition mode
  - Smart partitioning of work

- metaSMT: Focus On Your Application Not On Solver Integration [Haedicke et al. IJSTTT'17]

Suggested Project 4: Foresight

- Search extends prefixes
- May explore irrelevant paths
  - E.g., if we look for particular bugs

- Combine preliminary static analysis to pre-prune paths
  - Pointer analysis (Finds: x and y may never alias)
  - Numerical analysis (Finds: x > y)

Suggested Project 5: Interactive SE

- KLEE explores paths automatically
  - Search heuristics
- Add human into the loop
  - Guide the exploration
  - Guide the concretization