Program Analysis
Symbolic and Concolic Execution
(Part 2)
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
Problems of Symbolic Execution

- **Loops and recursion**: Infinite execution trees
- **Path explosion**: Number of paths is exponential in the number of conditionals
- **Environment modeling**: Dealing with native/system/library calls
- **Solver limitations**: Dealing with complex path conditions
- **Heap modeling**: Symbolic representation of data structures and pointers
Problems of Symbolic Execution

- **Loops and recursion**: Infinite execution trees
- **Path explosion**: Number of paths is exponential in the number of conditionals
- **Environment modeling**: Dealing with native/system/library calls
- **Solver limitations**: Dealing with complex path conditions
- **Heap modeling**: Symbolic representation of data structures and pointers
function f(a) {
    var x = a;
    while (x > 0) {
        x --;
    }
}

\[
\begin{align*}
    a_0 \geq 0 \\
    a_0 - 1 \geq 0 \\
    a_0 - 2 \geq 0 \\
\end{align*}
\]
Dealing with Large Execution Trees

**Heuristically select which branch to explore next**

- Select at *random*
- Select based on *coverage*
- Prioritize based on distance to "interesting" program locations
- Interleaving symbolic execution with *random testing*
Problems of Symbolic Execution

- Loops and recursion: Infinite execution trees
- Path explosion: Number of paths is exponential in the number of conditionals
- Environment modeling: Dealing with native/system/library calls
- Solver limitations: Dealing with complex path conditions
- Heap modeling: Symbolic representation of data structures and pointers
Problems of Symbolic Execution

- **Loops and recursion**: Infinite execution trees
- **Path explosion**: Number of paths is exponential in the number of conditionals
- **Environment modeling**: Dealing with native/system/library calls
- **Solver limitations**: Dealing with complex path conditions
- **Heap modeling**: Symbolic representation of data structures and pointers
Modeling the Environment

- Program behavior may depend on **parts of system not analyzed** by symbolic execution
- E.g., native APIs, interaction with network, file system accesses

```javascript
var fs = require("fs");
var content = fs.readFileSync("/tmp/foo.txt");
if (content === "bar") {
  ...
}
```
Solution implemented by KLEE

- If all arguments are concrete, forward to OS
- Otherwise, provide models that can handle symbolic files
  - Goal: Explore all possible legal interactions with the environment

```javascript
var fs = {
    readFileSync: function(file) {
      // doesn’t read actual file system, but
      // models its effects for symbolic file name
    }
};
```
Problems of Symbolic Execution

- **Loops and recursion**: Infinite execution trees
- **Path explosion**: Number of paths is exponential in the number of conditionals
- **Environment modeling**: Dealing with native/system/library calls
- **Solver limitations**: Dealing with complex path conditions
- **Heap modeling**: Symbolic representation of data structures and pointers
Problems of Symbolic Execution

- **Loops and recursion**: Infinite execution trees
- **Path explosion**: Number of paths is exponential in the number of conditionals
- **Environment modeling**: Dealing with native/system/library calls
- **Solver limitations**: Dealing with complex path conditions
- **Heap modeling**: Symbolic representation of data structures and pointers

**One approach**: Mix symbolic with concrete execution