Focaccia

Execution State Comparison for Emulators using Symbolic Execution

Nicola Crivellin
Advisor: Sebastian Reimers & Theofilos Augoustis
Chair of Computer Systems
https://dse.in.tum.de/

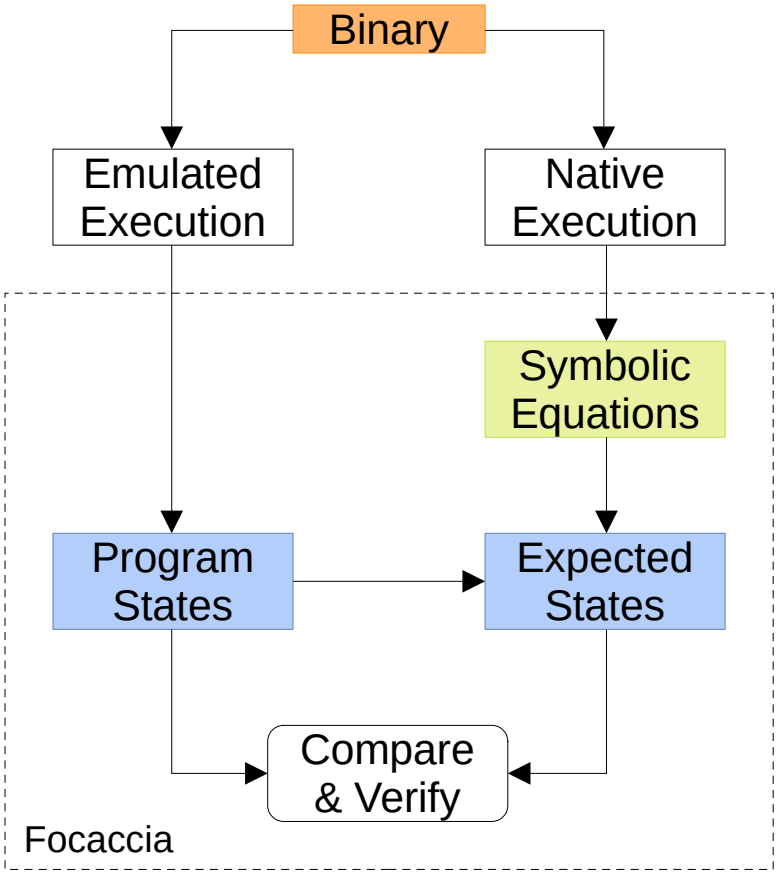
15.11.2023 – 15.05.2024

Introduction

- Developing emulators is difficult
- We want to make it easier to test them
  - Systematic
  - Automatic
  - Simple
- Premise: Emulators output code

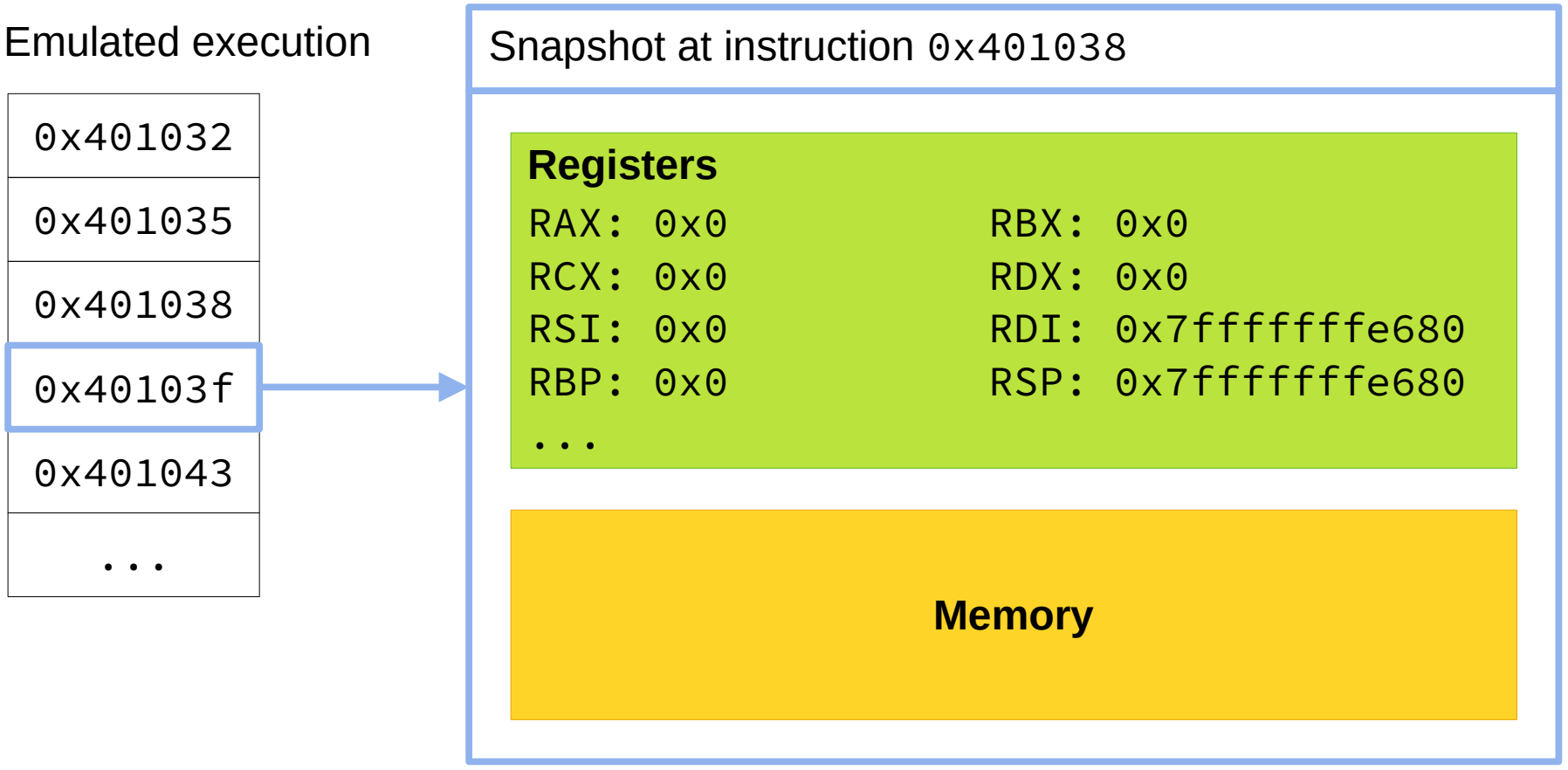
Can we use symbolic execution to verify an emulator's output?

The system: Overview



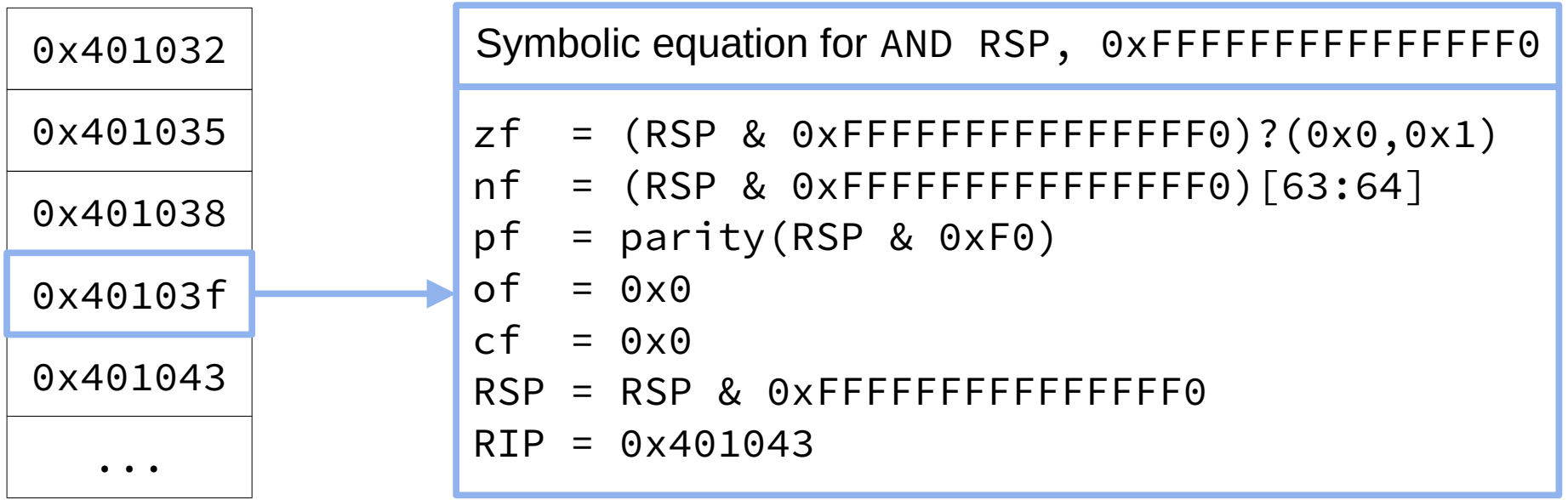
- Automatic testing system for emulators (currently for x86 guests)
- Calculates test truths via oracle
  - ➔ Uses symbolic execution
- Compares truth states to emulated states
  - ➔ Figures out whether instructions modify state correctly

The system: Recording a concrete trace

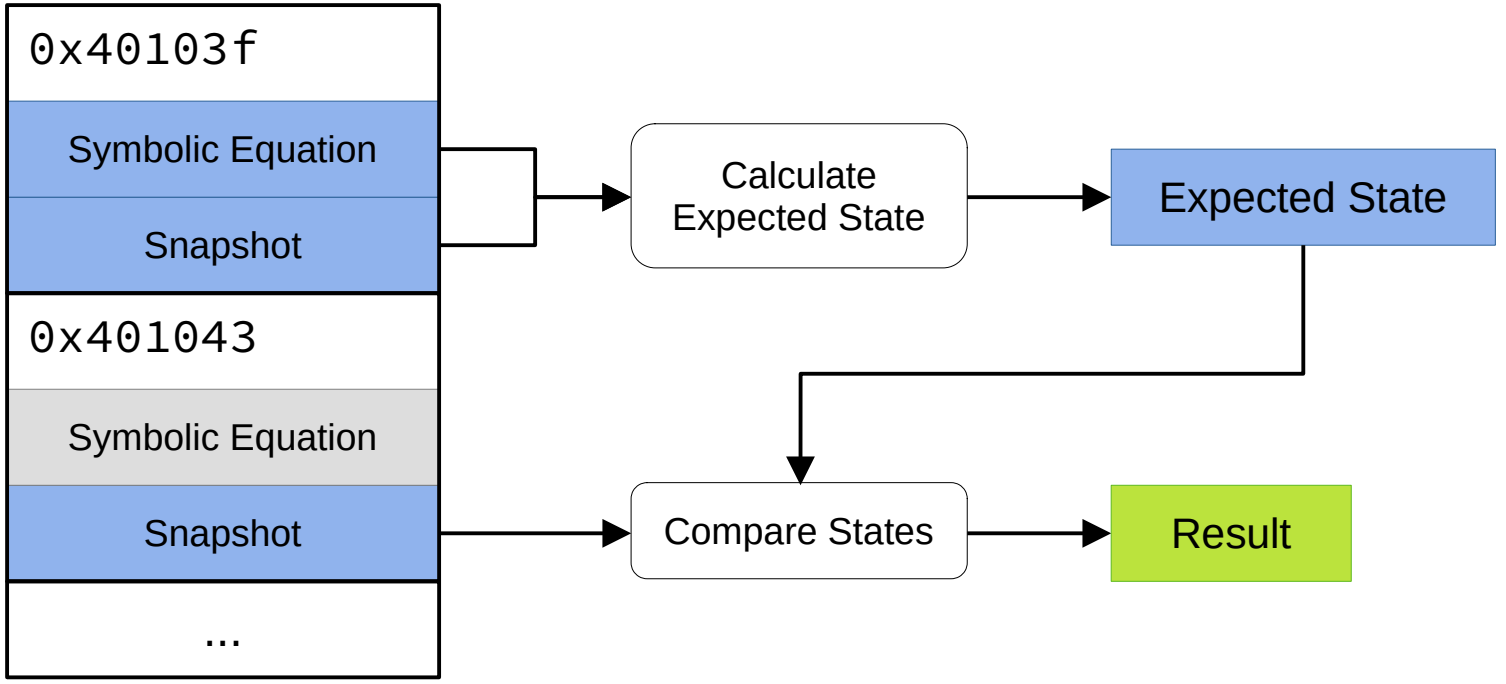


The system: Generating a symbolic trace

Native execution



The system: Predicting outcomes



Example

```
$ qemu-x86_64 -g 12345 bug.out &
$ ./capture_transforms.py -o symbols.trace bug.out
$ ./verify_qemu.py --symb-trace symbols.trace localhost 12345
```

```
[...]
--------------------------------------------------------------------------------
For PC=0x40116a
--------------------------------------------------------------------------------
  1. [ERROR] Content of register RAX is false. Expected value: 0x1234567812345678, actual value in the translation: 0x12345678.

Expected transformation: Symbolic state transformation 0x40116a -> 0x40116e:
  [Symbols]
    ZF = (-@32[RBP + 0xFFFFFFFFFFFFFFEC]) == (-RAX[0:32])
    RAX = ((-@32[RBP + 0xFFFFFFFFFFFFFFEC]) == (-RAX[0:32]))?(RAX,{@32[RBP + 0xFFFFFFFFFFFFFFEC] 0 32, 0x0 32 64})
    RIP = 0x40116E
    @32[RBP + 0xFFFFFFFFFFFFFFEC] = ((-@32[RBP + 0xFFFFFFFFFFFFFFEC]) == (-RAX[0:32]))?(RDI[0:32],@32[RBP + 0xFFFFFFFFFFFFFFEC])
  [Instructions]
    CMPXCHG DWORD PTR [RBP + 0xFFFFFFFFFFFFFFEC], EDI
Actual difference: Snapshot (x86_64): {'RIP': '0x4', 'RBP': '0x0', 'RAX': '-0x1234567800000000'}
```

Example

```
$ qemu-x86_64 -g 12345 bug.out &
$ ./capture_transforms.py -o symbols.trace bug.out
$ ./verify_qemu.py --symb-trace symbols.trace localhost 12345
```

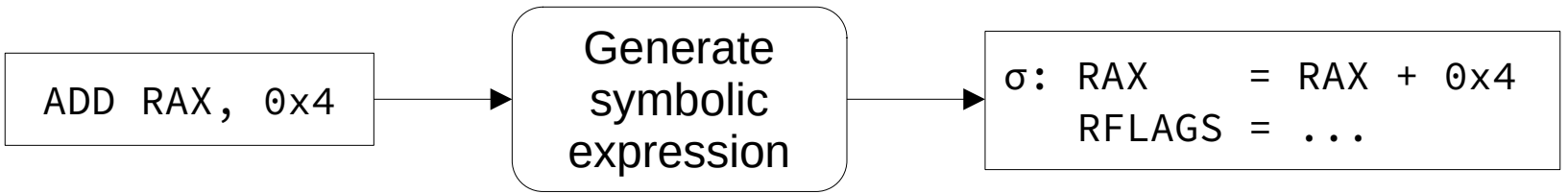
```
[...]
--------------------------------------------------------------------------------
For PC=0x40116a
--------------------------------------------------------------------------------
  1. [ERROR] Content of register RAX is false. Expected value: 0x1234567812345678, actual value in the translation: 0x12345678.

Expected transformation: Symbolic state transformation 0x40116a -> 0x40116e:
  [Symbols]
    ZF = (-@32[RBP + 0xFFFFFFFFFFFFFFEC]) == (-RAX[0:32])
    RAX = ((-@32[RBP + 0xFFFFFFFFFFFFFFEC]) == (-RAX[0:32]))?(RAX,{@32[RBP + 0xFFFFFFFFFFFFFFEC] 0 32, 0x0 32 64})
    RIP = 0x40116E
    @32[RBP + 0xFFFFFFFFFFFFFFEC] = ((-@32[RBP + 0xFFFFFFFFFFFFFFEC]) == (-RAX[0:32]))?(RDI[0:32],@32[RBP + 0xFFFFFFFFFFFFFFEC])
  [Instructions]
    CMPXCHG DWORD PTR [RBP + 0xFFFFFFFFFFFFFFEC], EDI
Actual difference: Snapshot (x86_64): {'RIP': '0x4', 'RBP': '0x0', 'RAX': '-0x1234567800000000'}
```

Evaluation

- Search QEMU's Gitlab for instruction-semantics bugs with x86 guest code
- Data: 9 bugs since 2021
- Procedure: Reproduce bugs, then run them through Focaccia
- Result: 1/9 bugs found successfully
  - Found: 1 bug
  - Symbolic execution backend: 6 bugs
  - QEMU crashes: 1 bug
  - Not reproducible: 1 bug

Discussion: Con

- Symbolic execution backend acts as a trusted emulator
- Focaccia uses one correct implementation to verify all other emulators
  - ➔ Software oracle vs. hardware oracle?
- Approach is not systematic: works on specific test cases



Discussion: Pro

- Gives useful results if the backend is good
- Resiliant to faults: backend, input data, emulator
- Tool is highly usable: minimally invasive, out-of-the-box paradigm, configurable