Tango: Extracting Higher-Order Feedback through State Inference

Ahmad Hazimeh (EPFL; BugScale), Duo Xu (EPFL),
Qiang Liu (EPFL), Yan Wang (Huawei),
Mathias Payer (EPFL)

EPFL

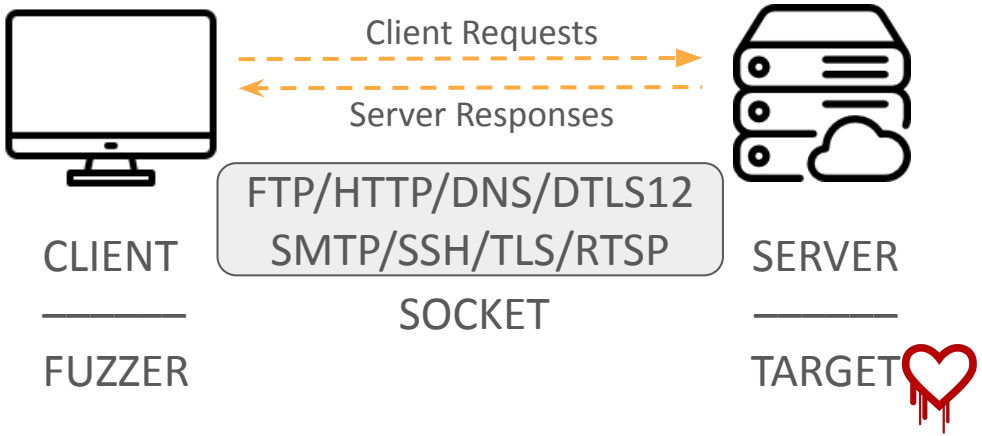
hexhive

Network protocol fuzzing based on the client-server architecture

Setup connection

Send and receive messages

Crash the target

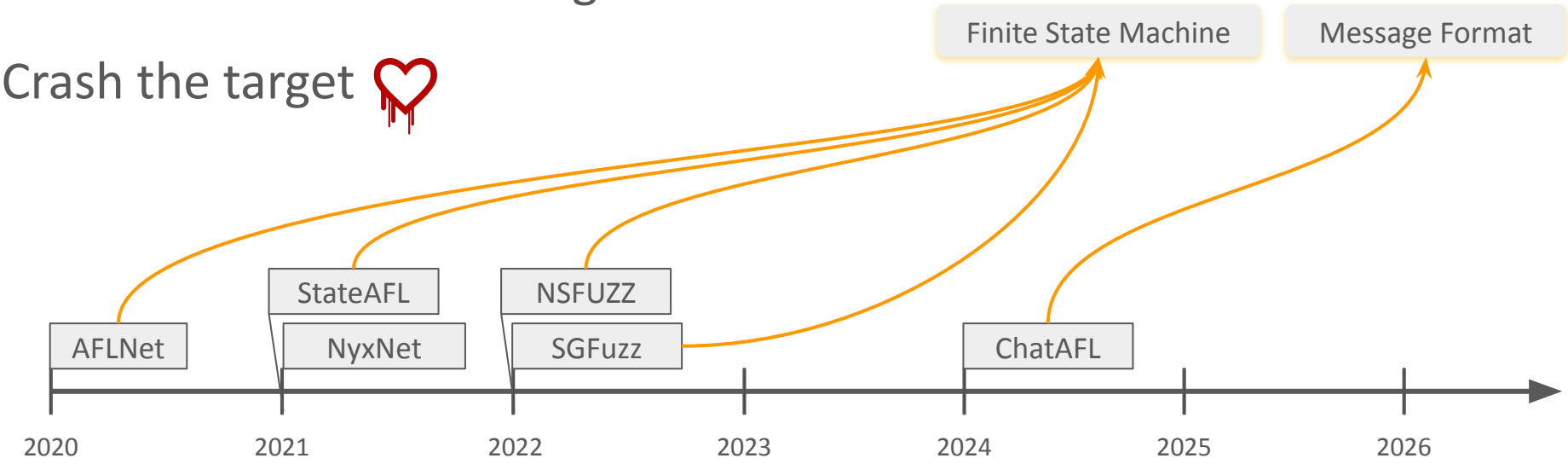


Network protocol fuzzing based on the client-server architecture

Setup connection

Generate and send messages

Crash the target

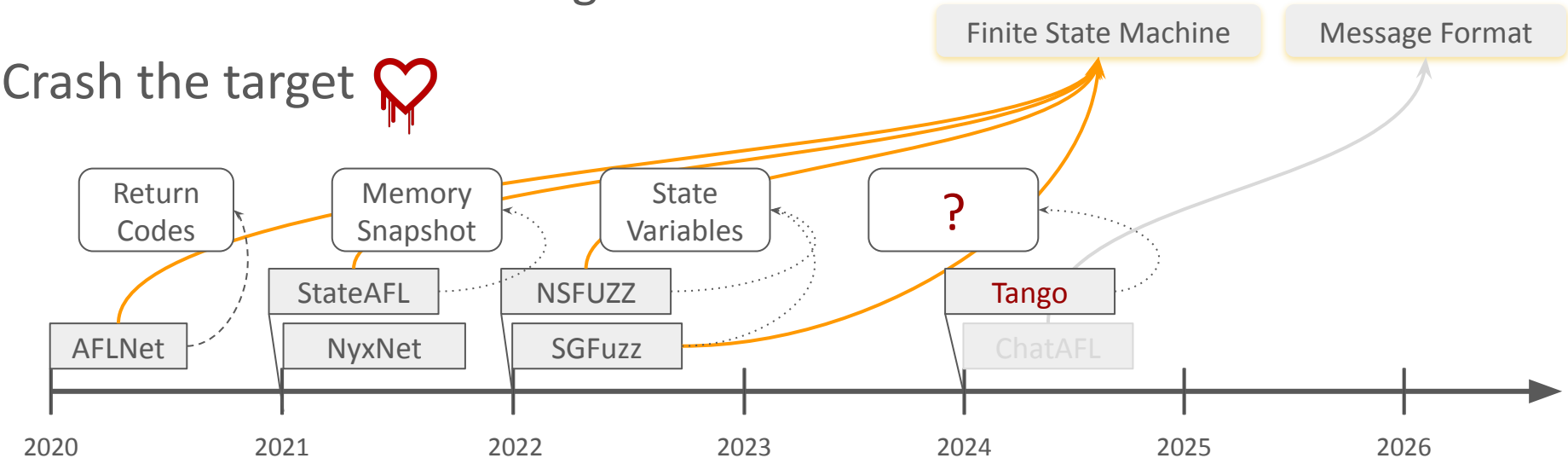


Network protocol fuzzing based on the client-server architecture

Setup connection

Generate and send messages

Crash the target



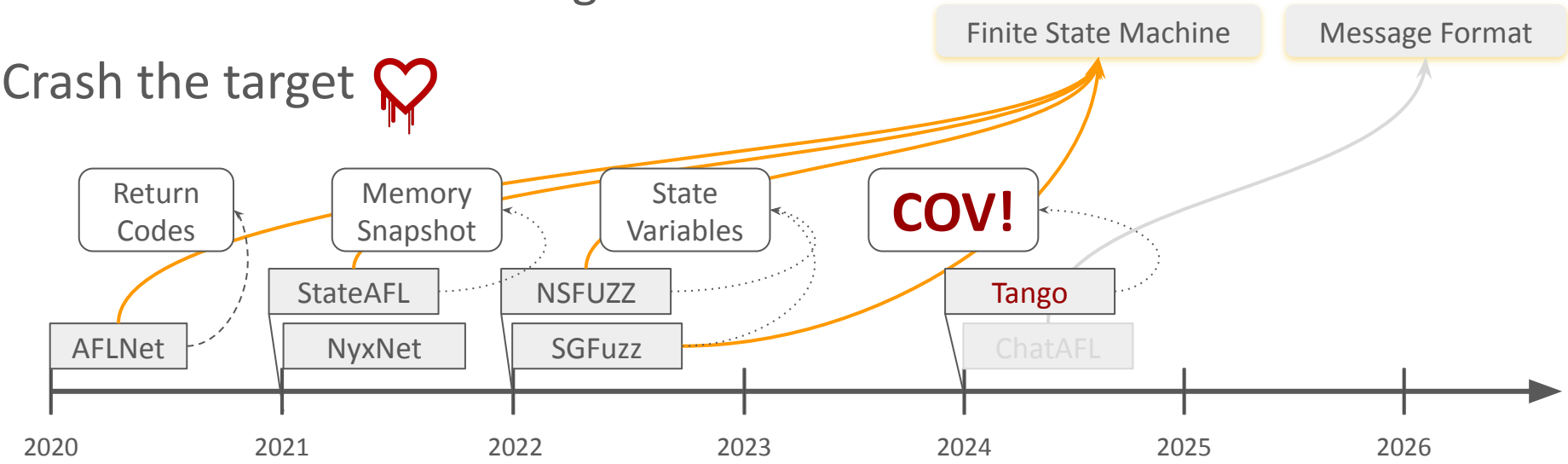
How can we extract the states in a generic way?

Network protocol fuzzing based on the client-server architecture

Setup connection

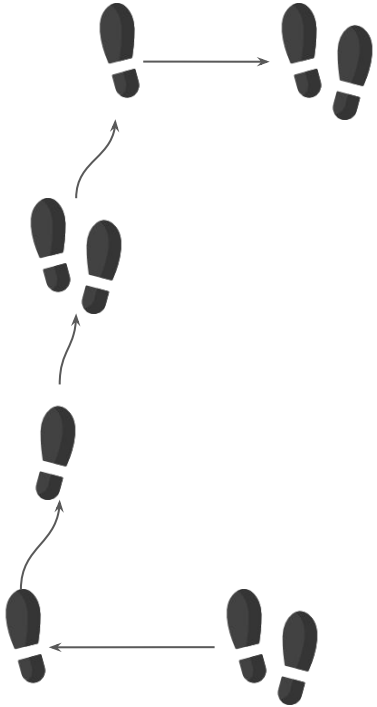
Generate and send messages

Crash the target



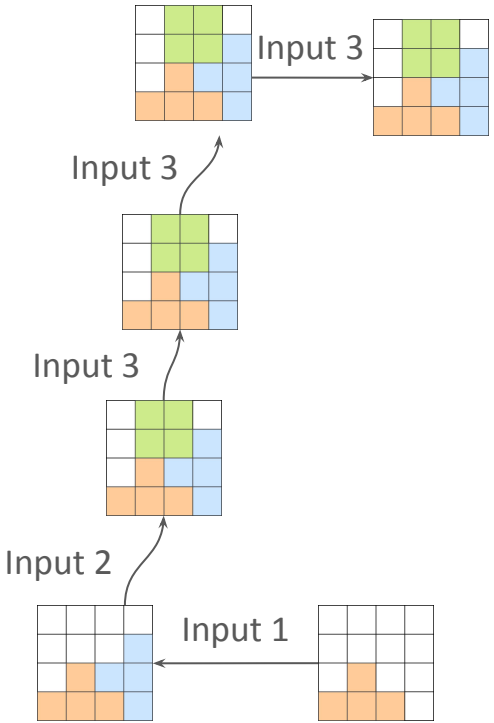
How can we extract the states in a generic way?

Our Tango has two steps



Our Tango has two steps

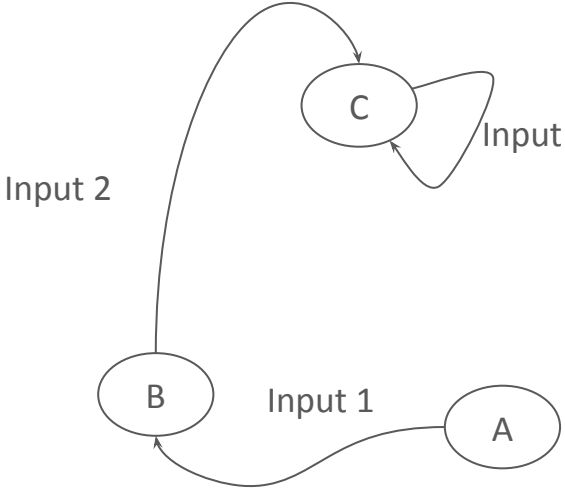
Step 1: for every input, snapshot the coverage map

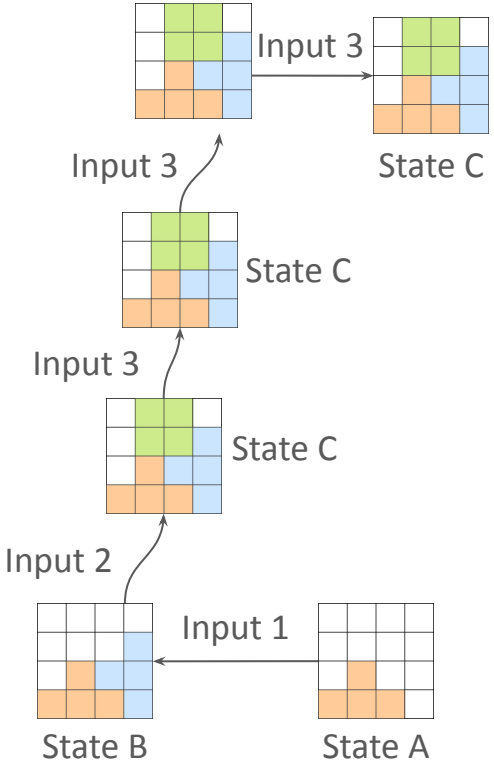


Our Tango has two steps

Step 1: for every input, snapshot the coverage map

Step 2: for every M snapshots, infer the states



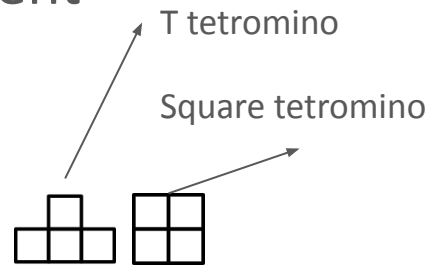


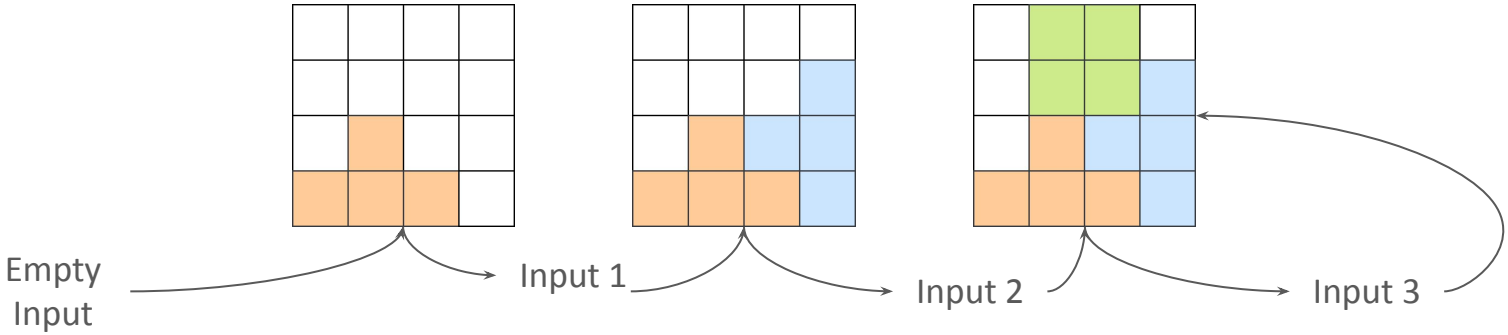
Step 1: Code coverage as snapshot indicator

A snapshot is a set of program's status at a certain moment

An interesting snapshot

- is found due to the relatively new response pattern
- is reachable by recording previous inputs





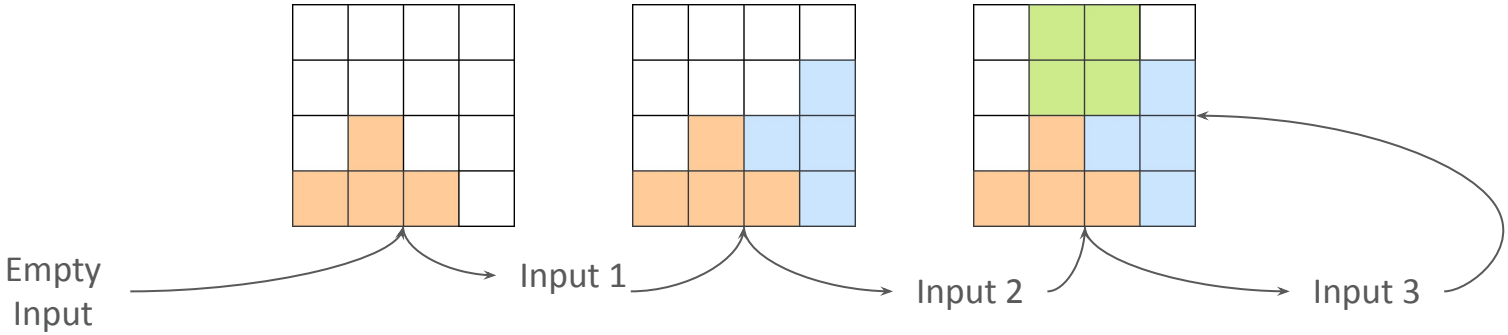
Tetromino: https://en.wikipedia.org/wiki/Tetromino

Bonus 1: Reliable Reproduction (Time Stone)

To reach  : replay Input 1

To reach  : replay Input 1, Input 2, Input 3, Input 3

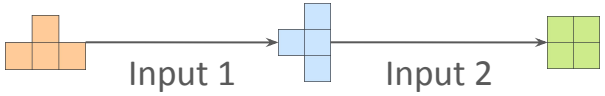
To reach  : replay Input 1, Input 2 (shortest) ✅



Step 2: Group equivalent snapshots to have the same state label

Two snapshots are equivalent

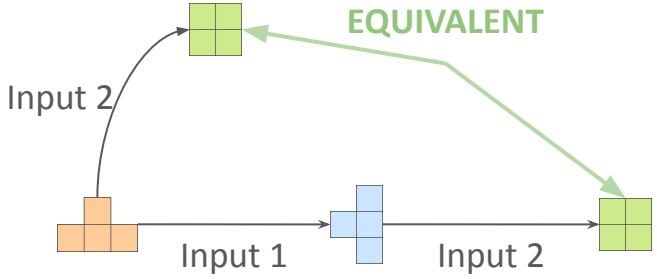
- if they have the same response pattern by the same input



Step 2: Group equivalent snapshots to have the same state label

Two snapshots are equivalent

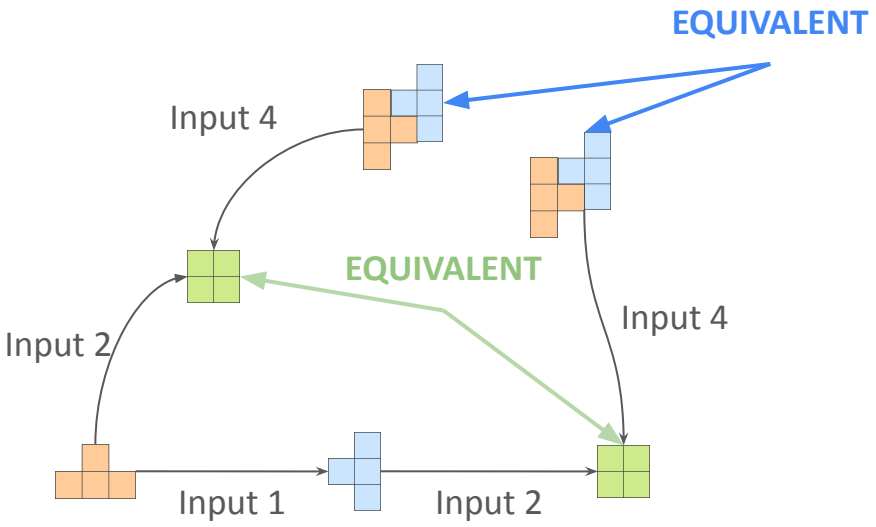
- if they have the same response pattern by the same input



Step 2: Group equivalent snapshots to have the same state label

Two snapshots are equivalent

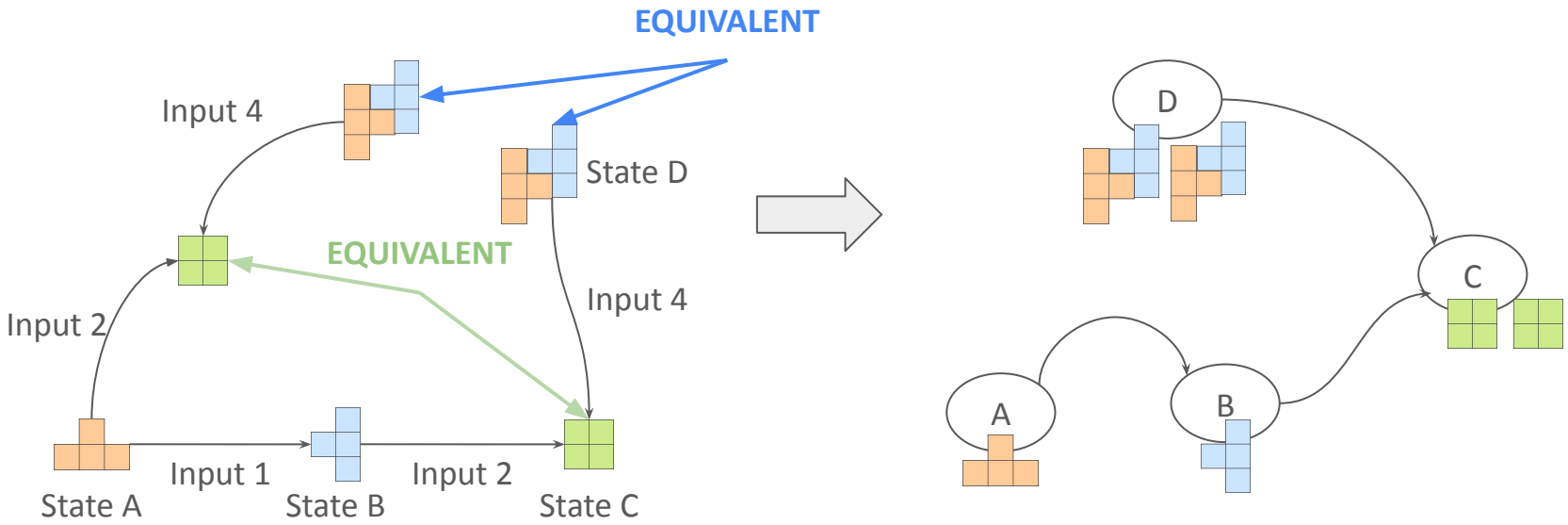
- if they have the same response pattern by the same input
- if the snapshots they can directly reach are equivalent



Step 2: Group equivalent snapshots to have the same state label

Two snapshots are equivalent

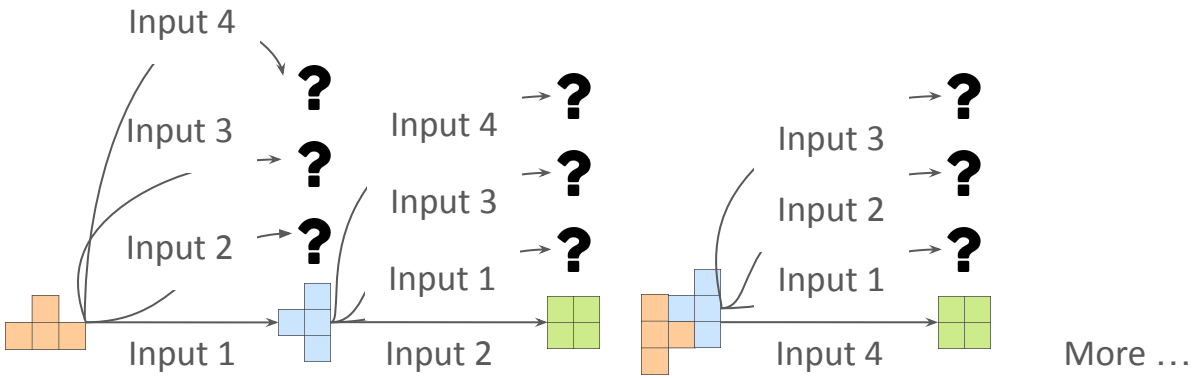
- if they have the same response pattern by the same input
- if the snapshots they can directly reach are equivalent



Step 2: Group equivalent snapshots to have the same state label

How to obtain a complete set of equivalent snapshots?

- Cross-pollination: apply every input to every existing snapshot
- During fuzz testing, cross-pollination becomes less and less, so the overhead starts high and then drops



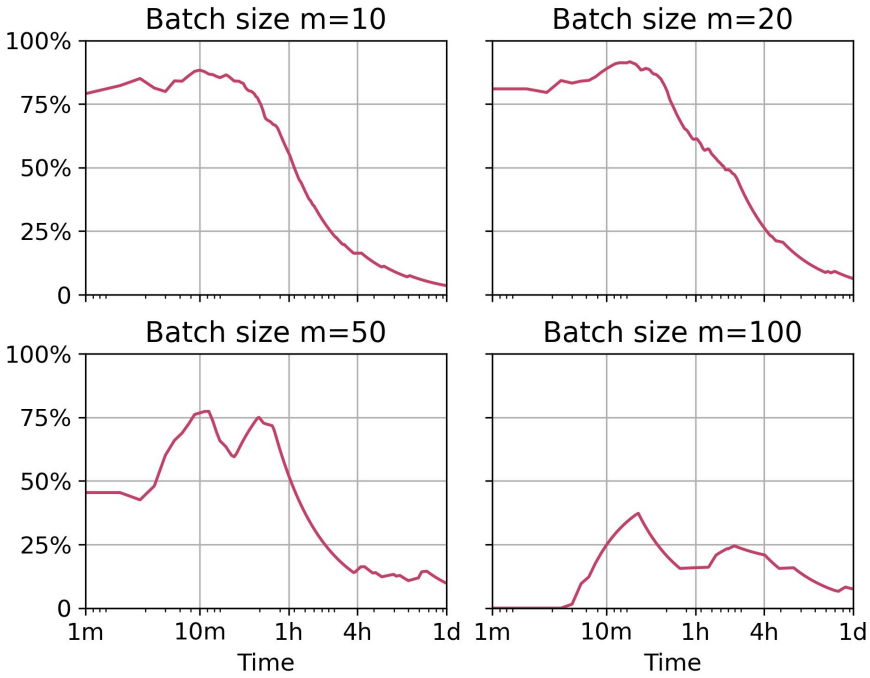
Time spent on state inference (Y-axis)

High overhead first

Going down gradually

The larger the batch size, the less time it takes to infer the state

More in paper

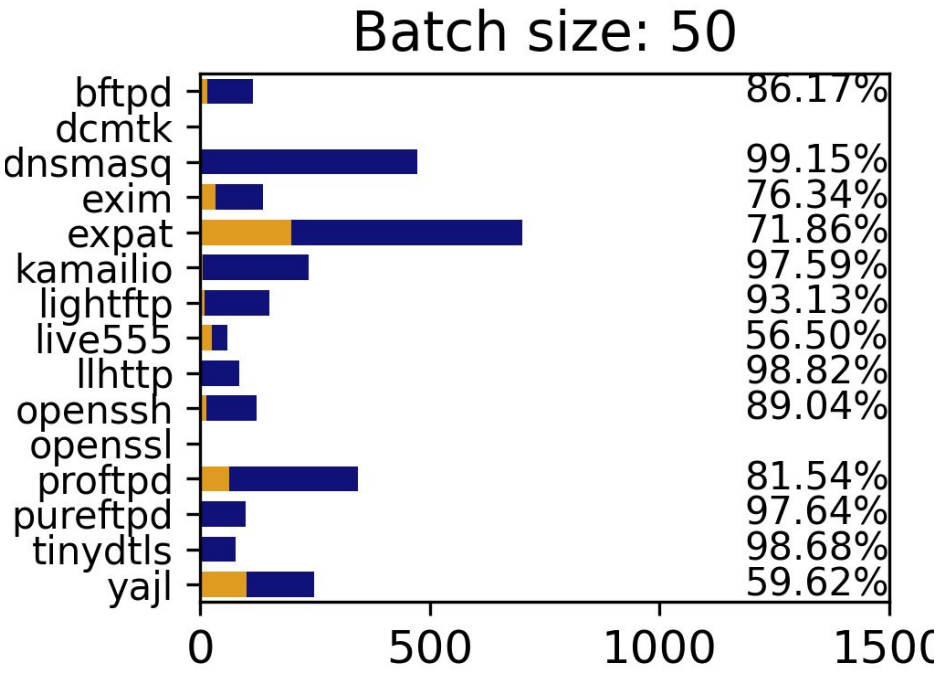


Bonus 2: Efficient seed scheduling at the state level

The queue size is reduced by more than 85%

The larger the batch, the less likely it is to perform state inference, so not all targets will benefit

NyxNet + Tango and AFL++ + Tango result in competitive coverage and two crashes



Tango: Extracting Higher-Order Feedback through State Inference

We propose a generic and automated state inference approach

facilitating reliable crash reproduction and efficient seed scheduling

By extending NyxNet and AFL++ with Tango, we achieve competitive coverage and get two crashes

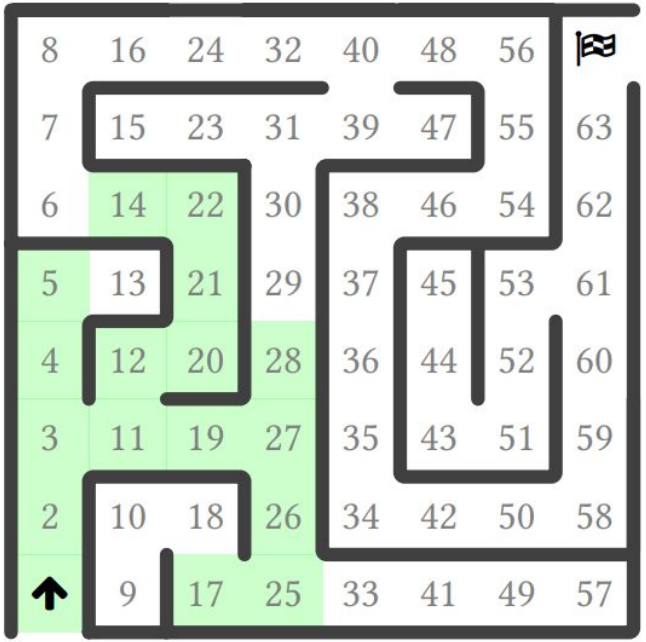
Repo link: https://github.com/HexHive/Tango

EPFL

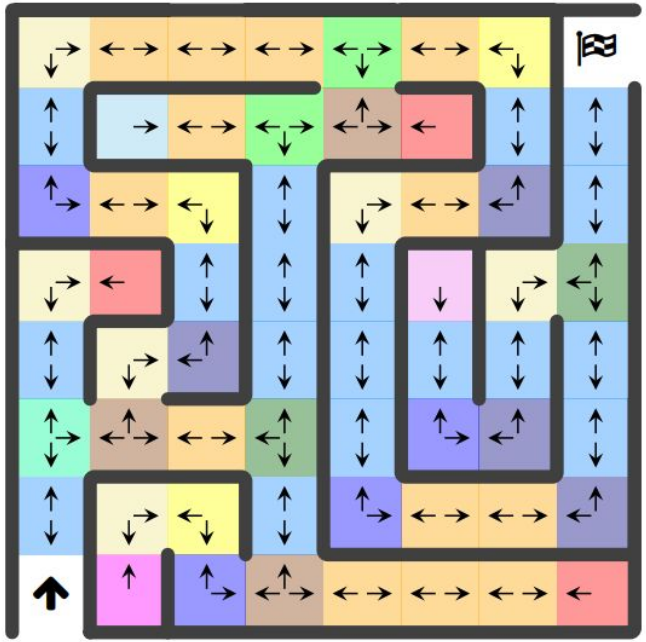
hexhive

Backup Slides

What are the states?



(a) W/ labeled cells, a fuzzer can systematically explore the maze.



(b) W/o labels, a fuzzer can indentify cells by their surroundings.

Motivation of cross-pollination

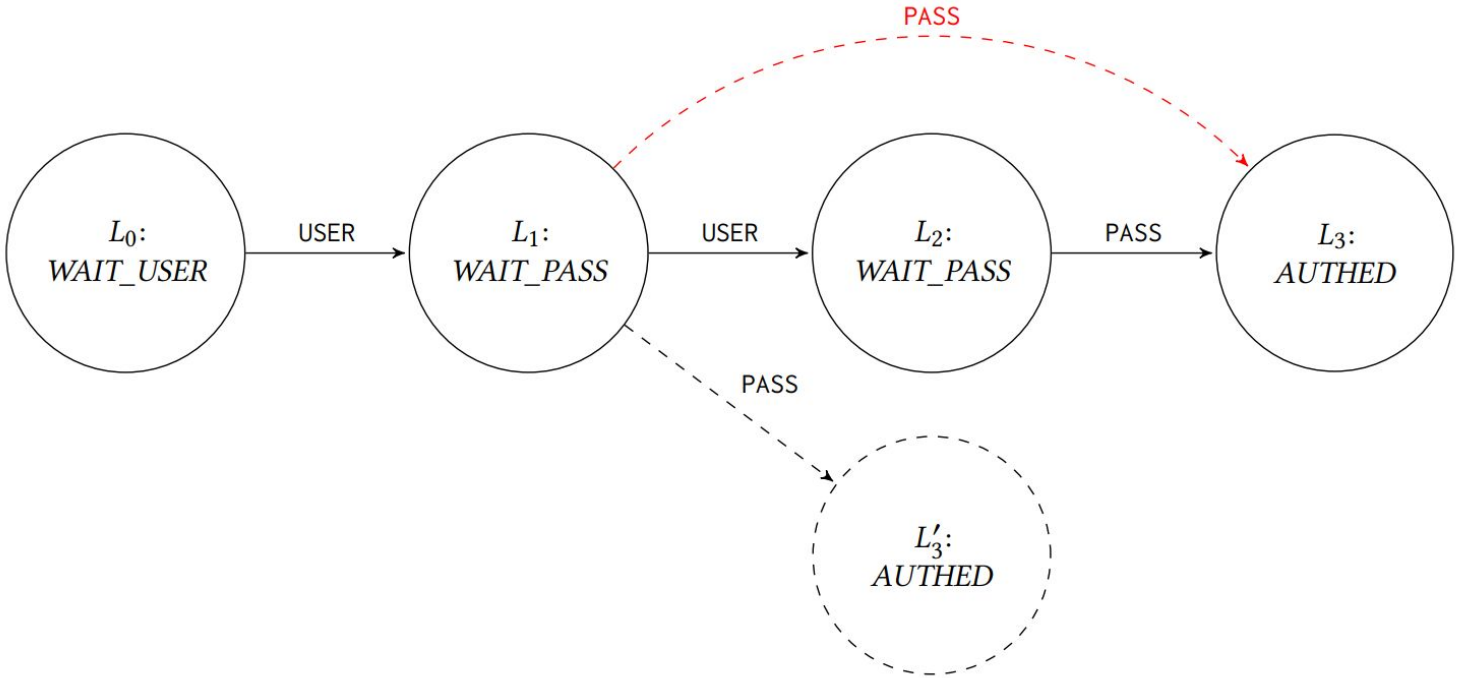
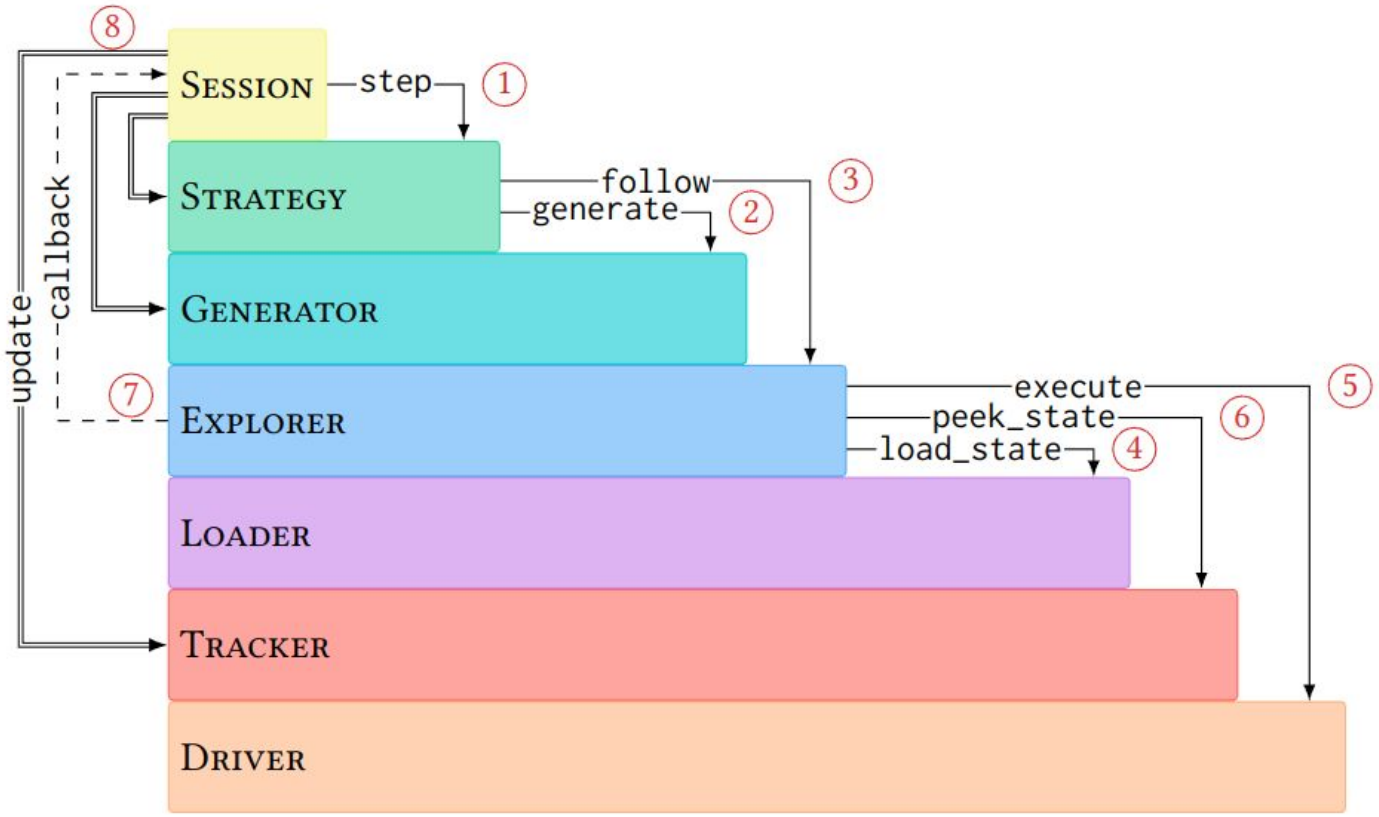
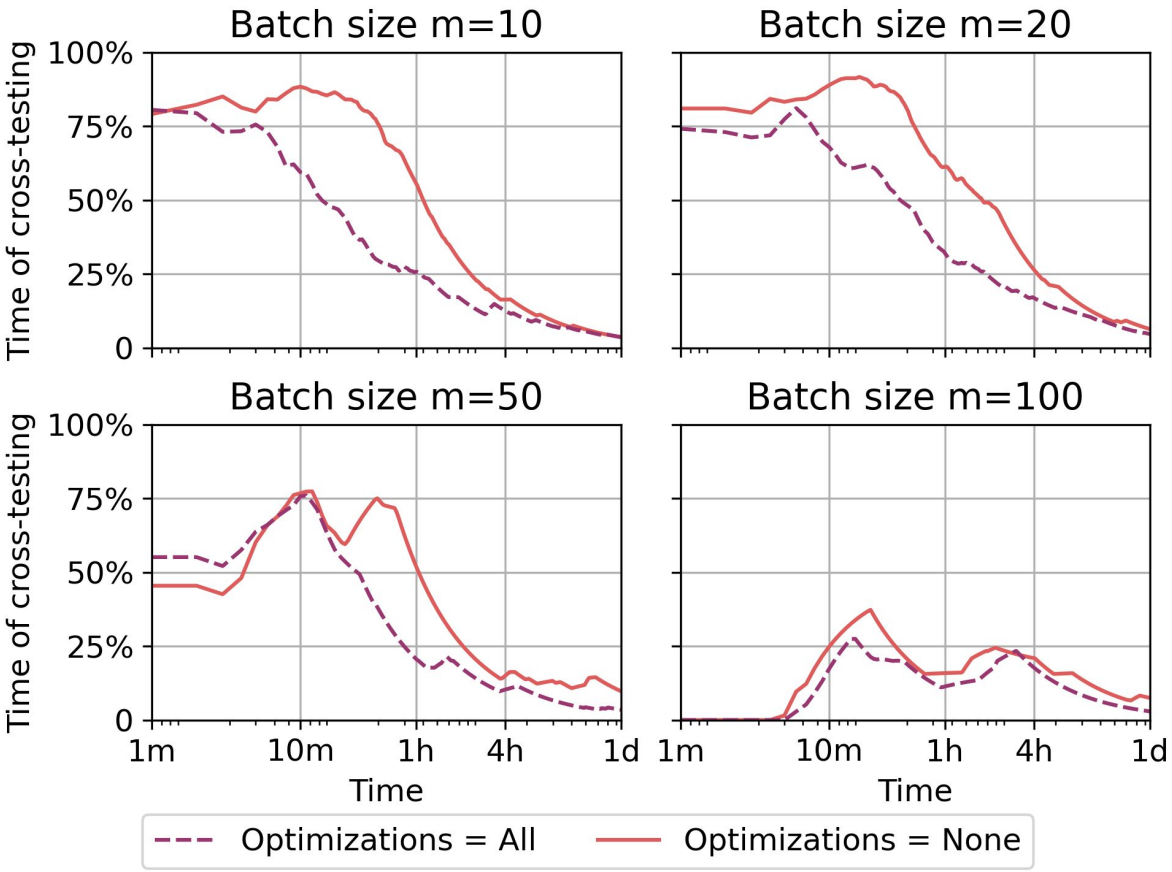


Figure 3: A snapshot tree constructed for an FTP server.

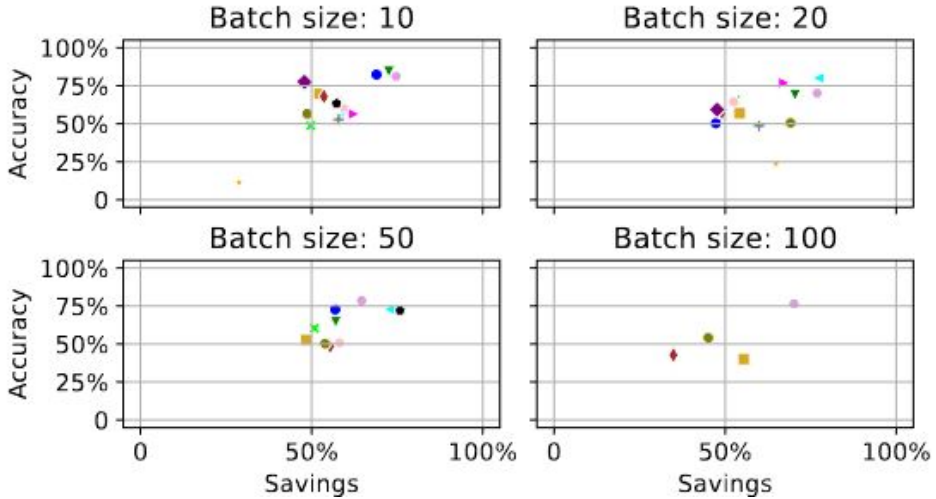
The general workflow of the Tango framework



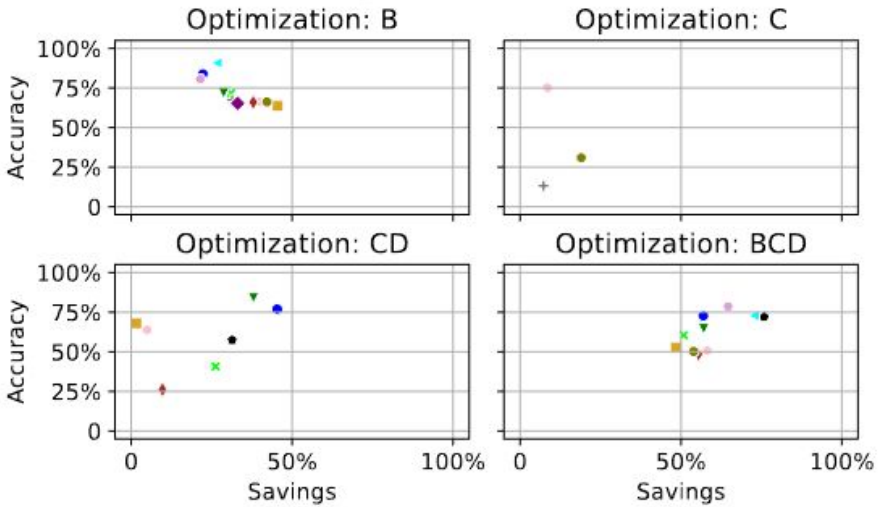
Time spent on state inference during the fuzzing campaign



The hit accuracy (the percentage of correctly labeled snapshots) of optimizations as a function of introduced savings (the skipped tests) based the ground truth collected



Optimizations: All



Batch size m = 50

Edge coverage collected from Nyx-Net (for network servers) and AFL++ (for parsers) when running without (solid lines) and with (dotted lines) the state inference extension