MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

RUHR
UNIVERSITÄT
BOCHUM

RUB

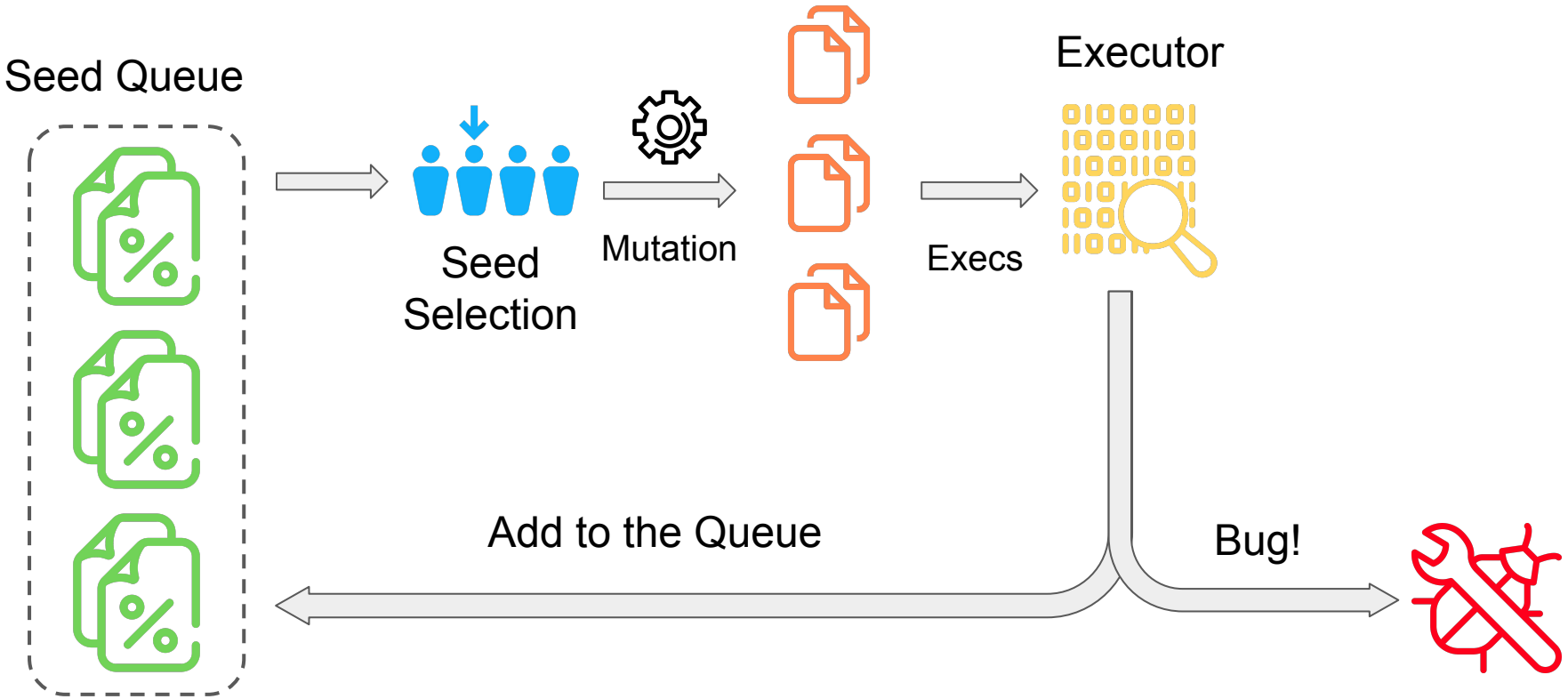
MendelFuzz: the Return of the Deterministic Stage

Han Zheng, Flavio Toffalini, Marcel Böhme, Mathias Payer

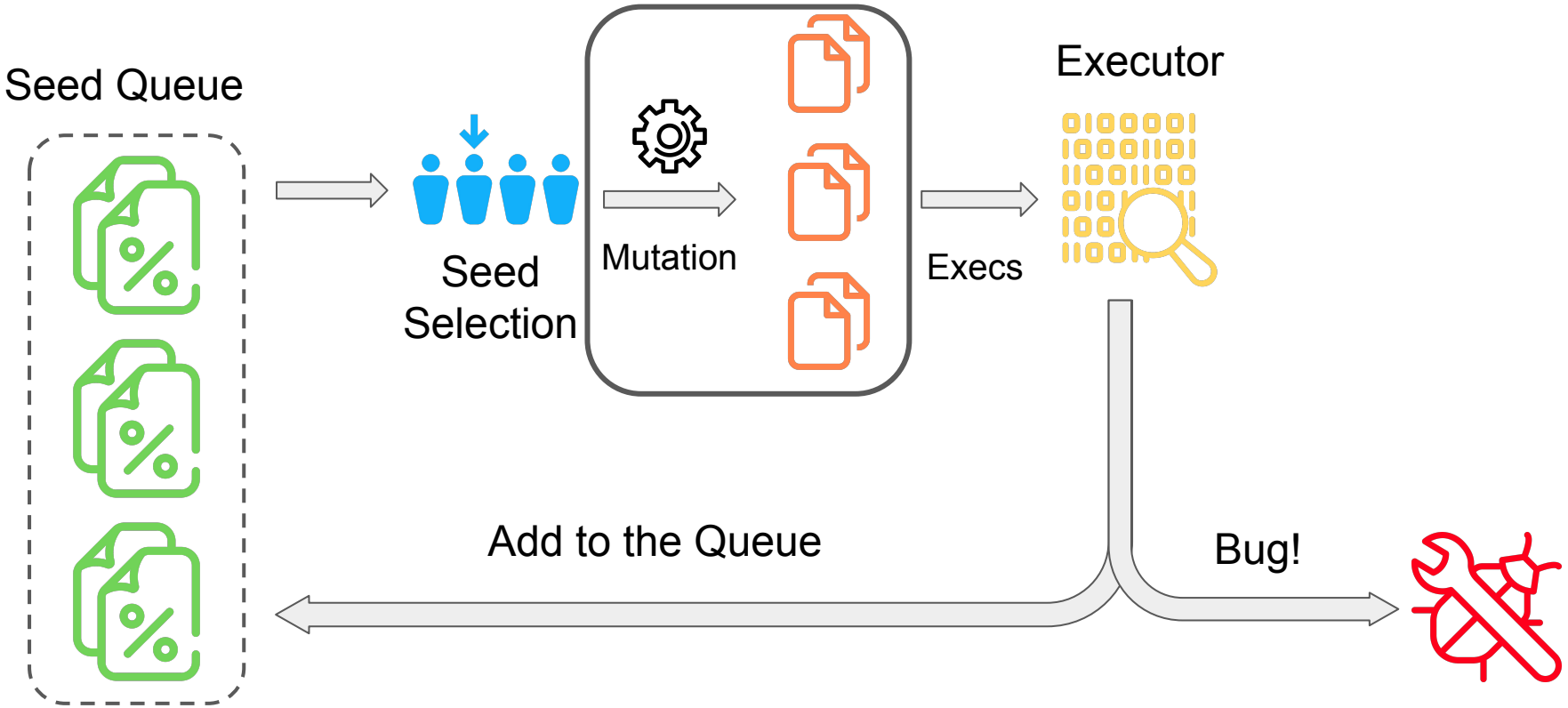
EPFL

hexhive

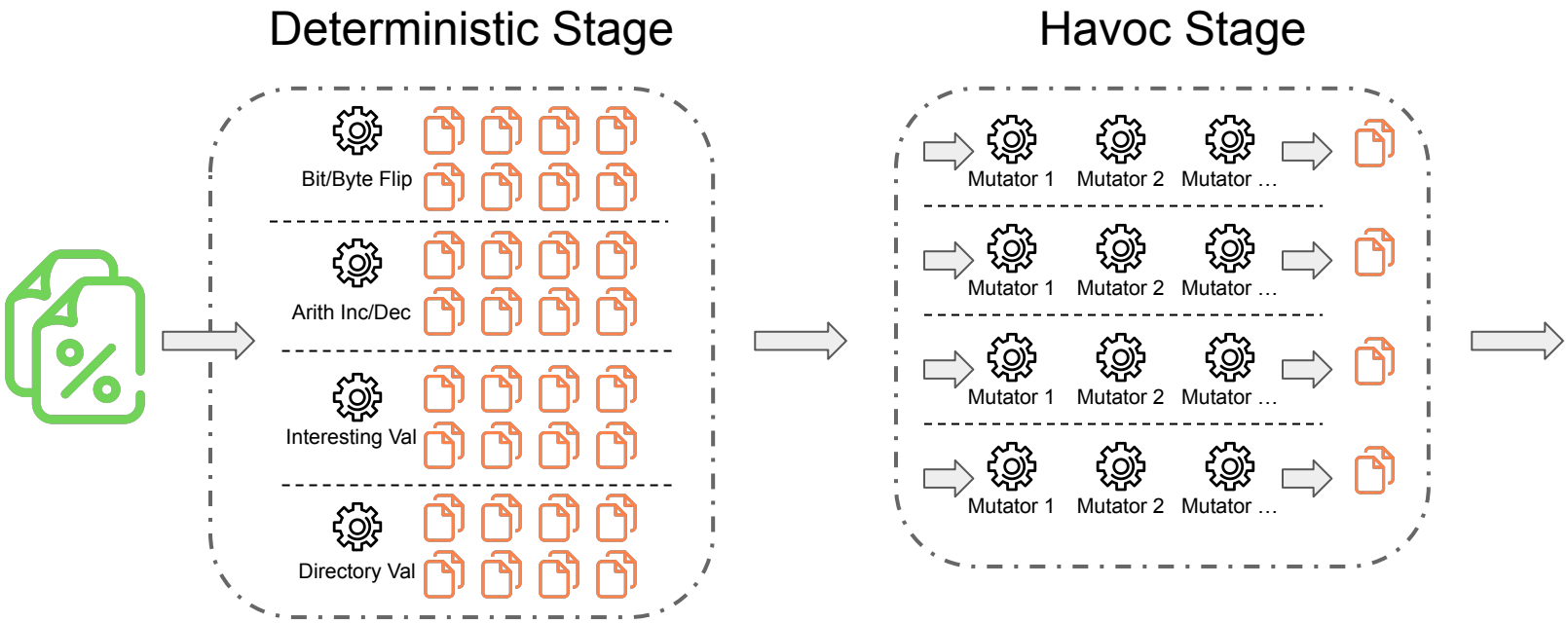
Introduction: Fuzzing Workflow



Introduction: Fuzzing Workflow



Introduction: Deterministic Stage Takes Much Longer



Up to **Millions**
execution per seed

Less than **128K**
execution per seed

Introduction: SoTAs Disable the Deterministic Stage

…, **skipping the deterministic stage** caused AFL to perform ***statistically significantly better*** than AFL with the deterministic stage – FuzzBench

Version 3.00c release:
deterministic fuzzing is now disabled by default

AFL++

No deterministic stage implementation

LibFuzzer

Motivation: Havoc is NOT all we need

benchmark	AFL		AFL++		Honggfuzz	
	det	havoc	det	havoc	det	havoc
systemd_fuzz-link-parser	1244	640	1256	640	1283	639
woff2-2016-05-06	2306	1859	2316	1872	2318	1893
re2-2014-12-09	4152	3527	4121	3517	4032	3505
jsoncpp_jsoncpp_fuzzer	665	638	665	638	626	640

Fuzzer's edge coverage **w/** and **w/o** deterministic stage.

In 4 / 23 targets, fuzzers with deterministic stage perform better[1,2]

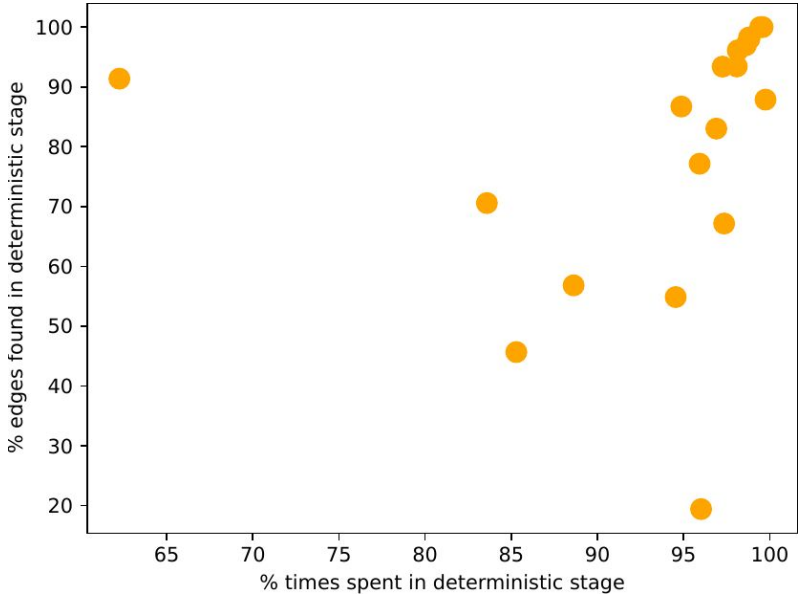
[1] https://www.fuzzbench.com/reports/paper/AFL%20Deterministic%20Experiment/index.html
[2] https://www.fuzzbench.com/reports/paper/Main%20Experiment/index.html

Study: Effectiveness of Deterministic Stage

Leveraging MAGMA, we assess the following:

- Study 1: Contribution of the deterministic / havoc stages
- Study 2: Contribution of individual seeds to coverage
- Study 3: Contribution of individual bytes to coverage

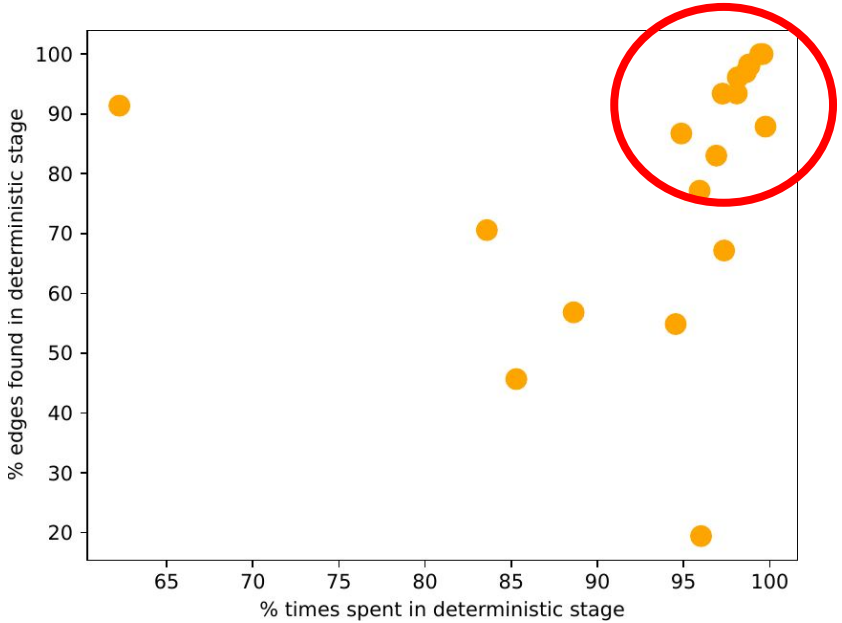
Study 1: Contribution of the Deterministic / Havoc Stages



We run 23 programs in MAGMA using AFL++ with both stages enabled for 24h.

Each point represents a program, highlighting the discovery from both the deterministic and havoc stage.

Study 1: Contribution of the Deterministic / Havoc Stages



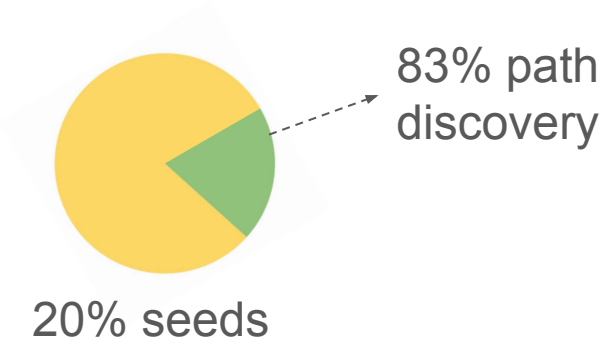
We run 23 programs in MAGMA using AFL++ with both stages enabled for 24h.

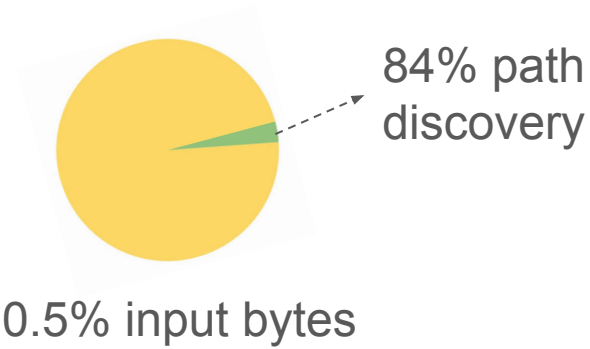
Each point represents a program, highlighting the discovery from both the deterministic and havoc stage.

Deterministic stage takes time, but may bring new coverage

Study 2 / 3: Contribution of Seeds / Bytes to Coverage

We study the contribution of each individual bytes / seeds by collect the paths discovery of each bytes / seeds.



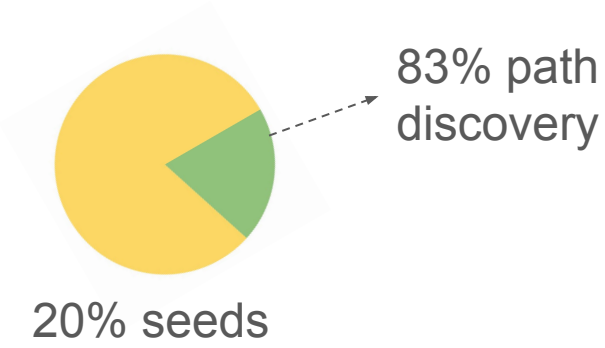


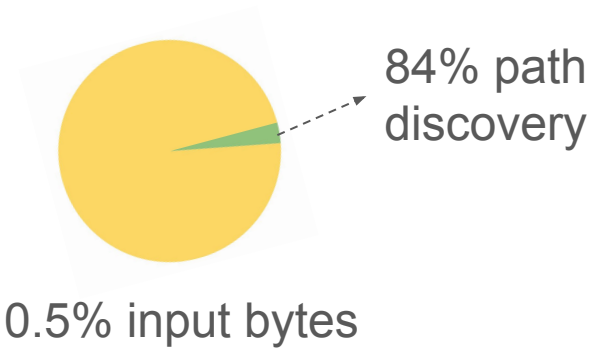
Study 2: 20% of the seeds contribute to 83% of path discovery

Study 3: 0.5% of the input bytes contribute to 84% of path discovery

Study 2 / 3: Contribution of Seeds / Bytes to Coverage

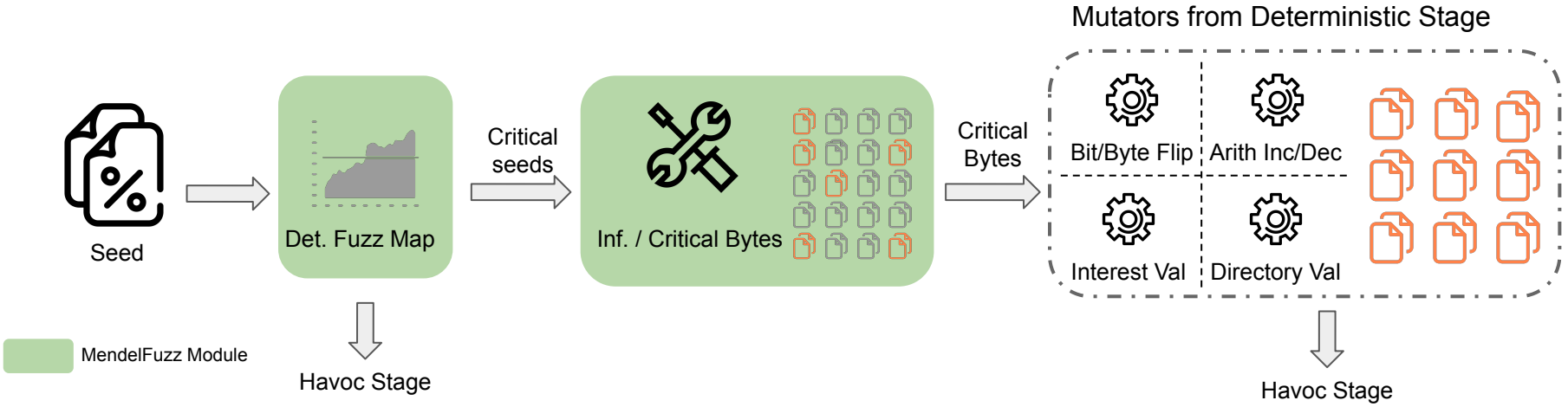
We study the contribution of each individual bytes / seeds by collect the paths discovery of each bytes / seeds.





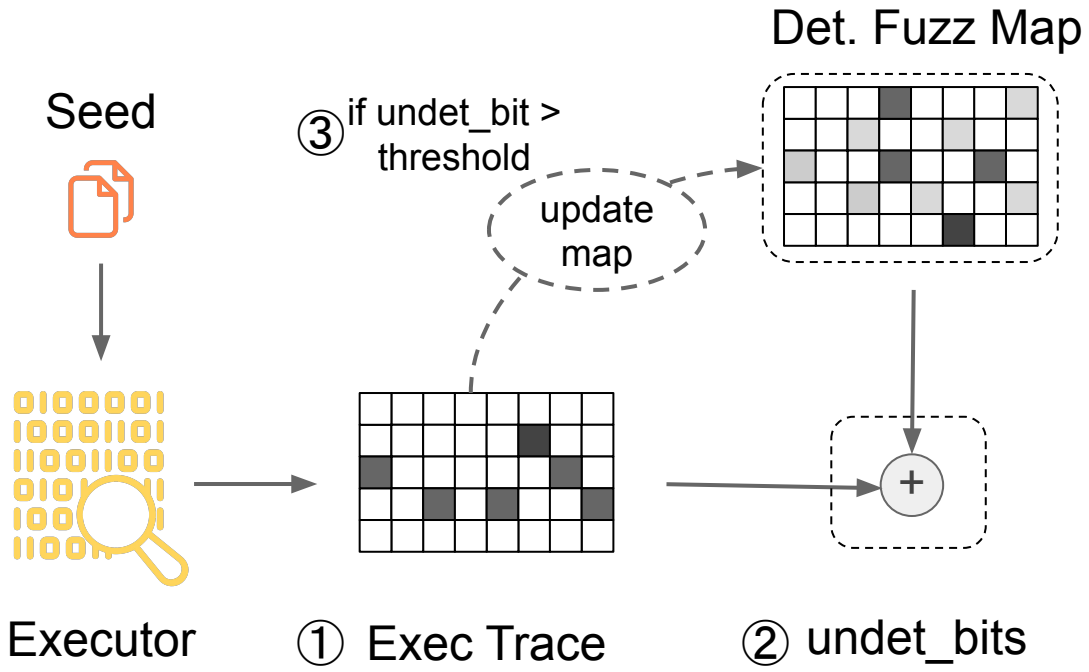
Small fraction of bytes / seeds contribute to most new path findings

Design of MendelFuzz



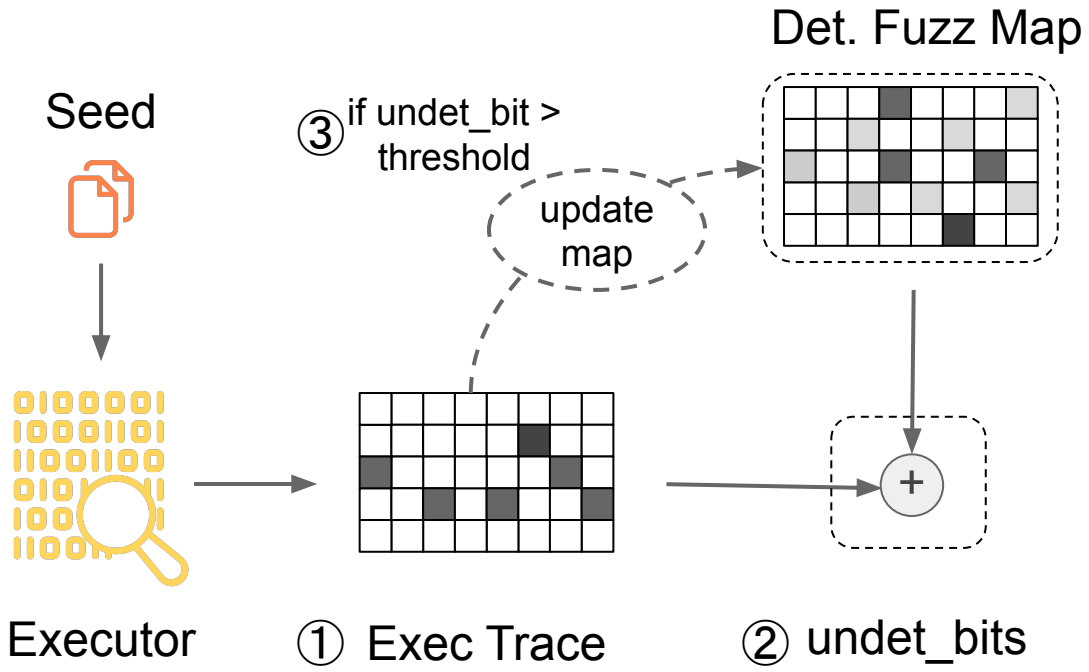
Based on our observations, we introduce **Deterministic Fuzz Map** and **Critical Bytes** to reduce number of bytes and seeds being deterministically fuzzed, finally implement our prototype **MendelFuzz.**

Design: Deterministic Fuzz Map



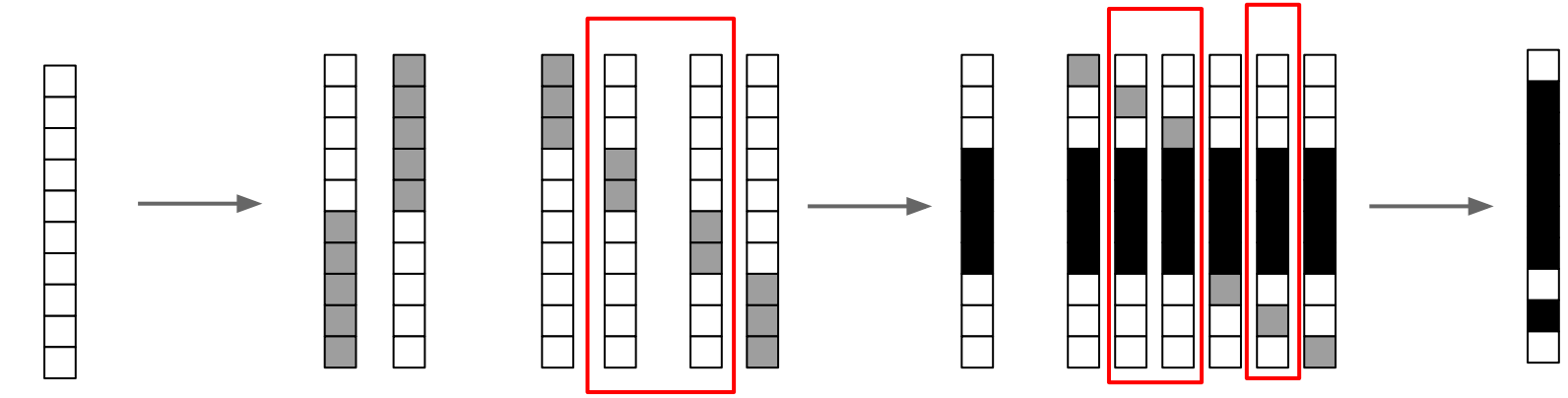
MendelFuzz ① run each seed for its **Execution Trace**, then ② diff between **Det. Fuzz Map** and current **Exection Trace** to get undet_bits. Finally, ③ if undet_bits is bigger than threshold, we fuzz current Seed deterministically.

Design: Deterministic Fuzz Map



Det. Fuzz Map reduce the number of seeds for deterministic stage

Design: Inf. / Critical Bytes



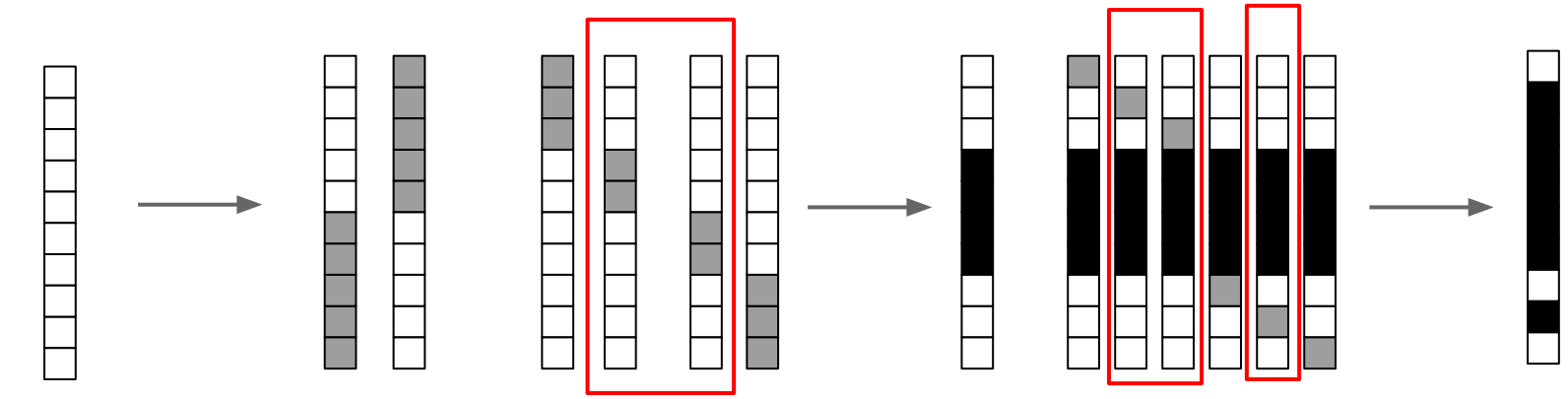
Seed

Inference stage:
Binary Search by Flipping

Critical Bytes Stage:
Flipping each byte

MendelFuzz rapidly scan bytes that does not contribute by binary searching, then flipping remaining bytes one by one, finally find the critical bytes and goes through whole deterministic stage.

Design: Inf. / Critical Bytes



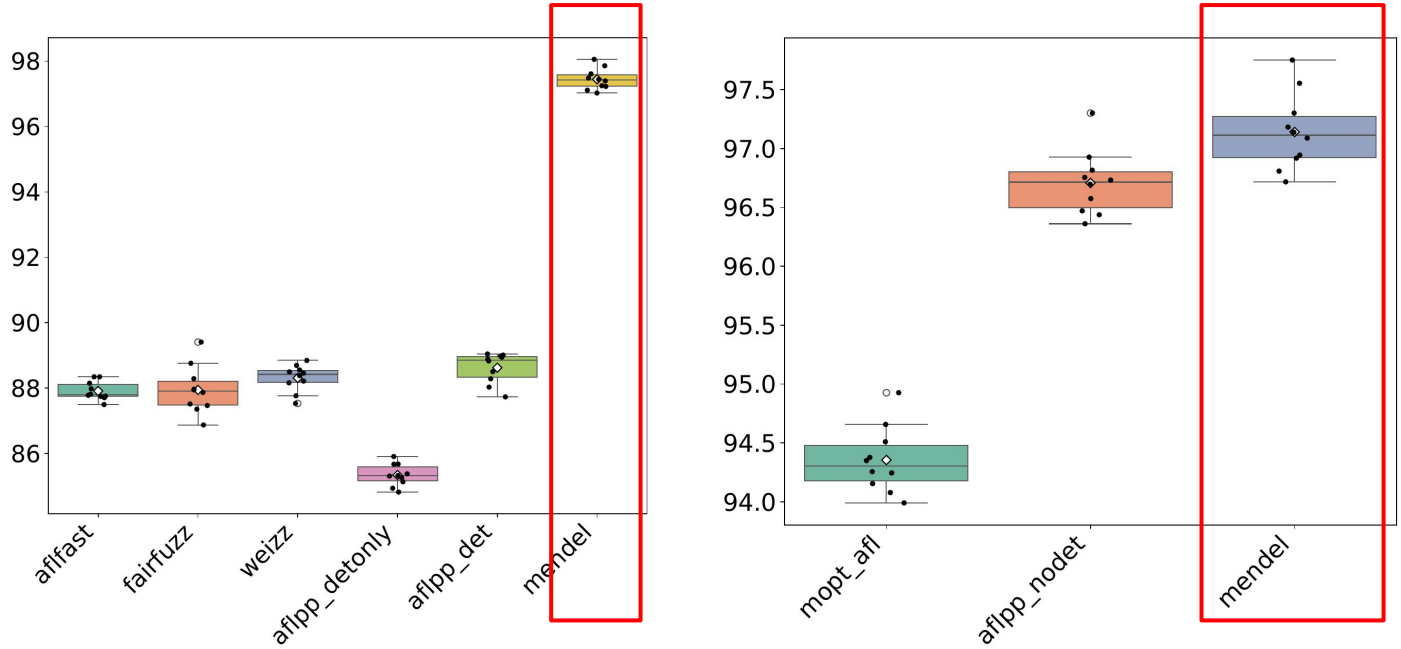
Seed

Inference stage:
Binary Search by Flipping

Critical Bytes Stage:
Flipping each byte

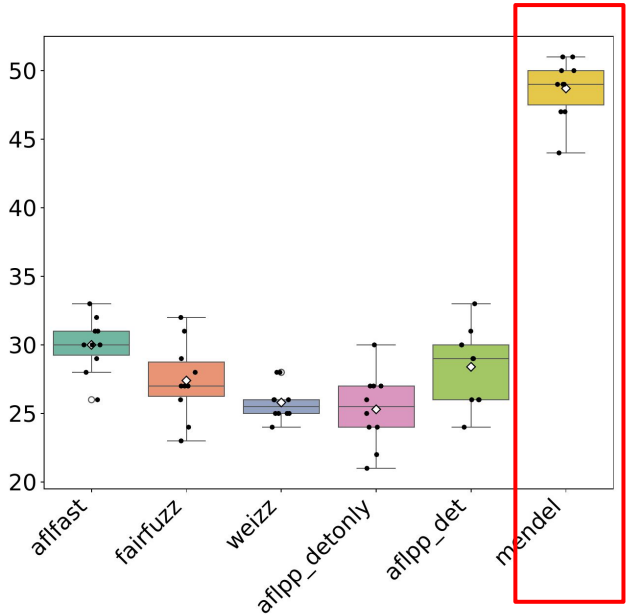
Inference / Critical Bytes reduce the # bytes for deterministic stage

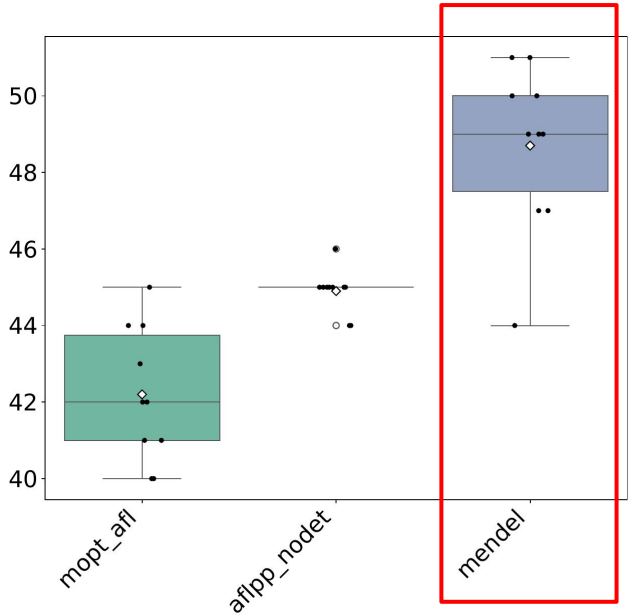
Evaluation: MendelFuzz Outperforms SoTA in Coverage



MendelFuzz outperform state-of-the-art (w/ and w/o deterministic stage) in coverage using MAGMA benchmark

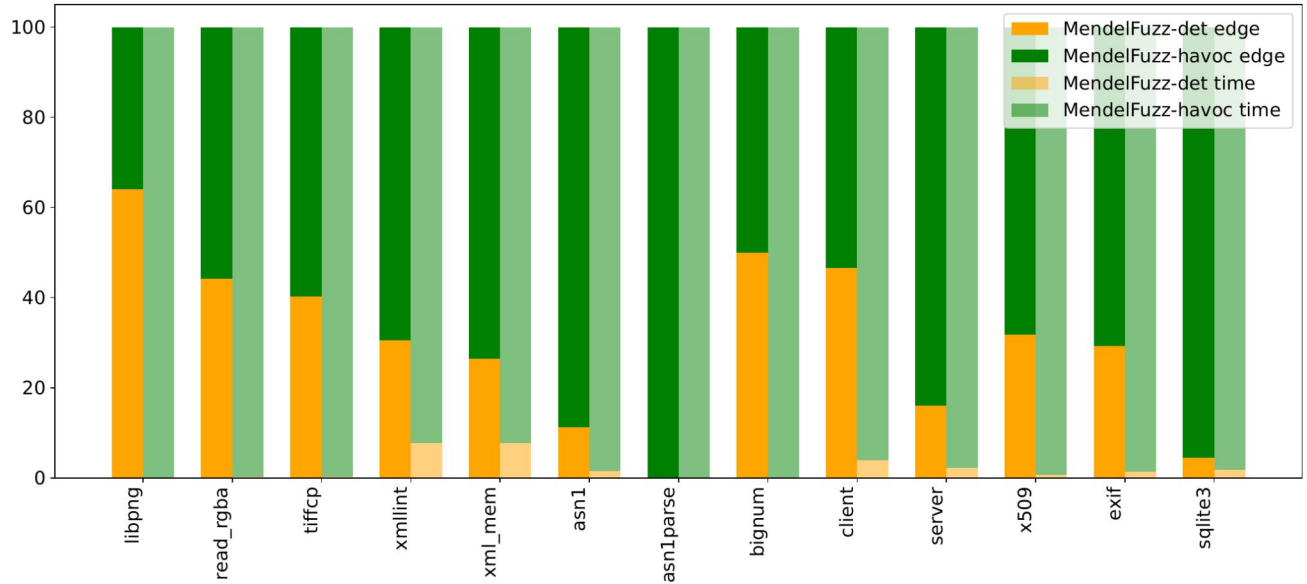
Evaluation: MendelFuzz Outperforms SoTA in Bug Finding





MendelFuzz outperform state-of-the-art (w/ and w/o deterministic stage) in unique bug discovery using MAGMA benchmark.

Evaluation: MendelFuzz Improve Deterministic Efficiency



We redo the study on MendelFuzz and notice that MendelFuzz's new deterministic stage has notably higher efficiency compared to havoc stage

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

RUHR
UNIVERSITÄT
BOCHUM
RUB

Deterministic stage is beneficial but needs tuning.

Most deterministic mutations are redundant.

MendelFuzz improves coverage and bug finding.

MendelFuzz became the default mode in AFL++!

EPFL

hexhive