Automated System-level Testing from Unit Testing through Composition of Function Summaries : FOCAL++ (FOcused CompositionAL testing)

Moonzoo Kim
SWTV group, KAIST

Joint work with
Yunho Kim @ Hanyang Univ, Shin Hong@ Handong Global Univ.
and Ahcheong Lee @ KAIST

15년 간 System-level 및 Unit-level 자동 테스팅 산학연구 수행

V+Lab

CROWN 2.0

’10 ~14

SAMSUNG

삼성전자

산학과제

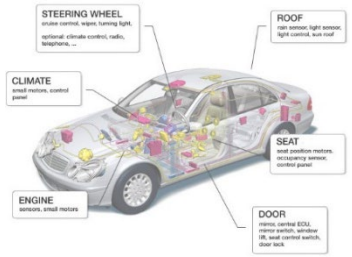
- 통신 모듈 펌웨어에서 수십건의 crash 오류 검출

’15 ~20

HYUNDAI MOBIS

현대자동차/ 모비스

산학과제



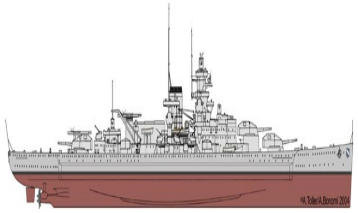
- 자동 테스팅 기술로 분기 커버리지 90% 자동 달성 테스팅 인건비 70% 감소

‘18

LIG 넥스원

LIGnex1

산학과제



- 함정 전투체계에 사용되는 10개 프로그램에서 다수의 SW 결함 발견

’20

NSR 국가보안기술연구소 National Security Research Institute

국가보안기술 연구소

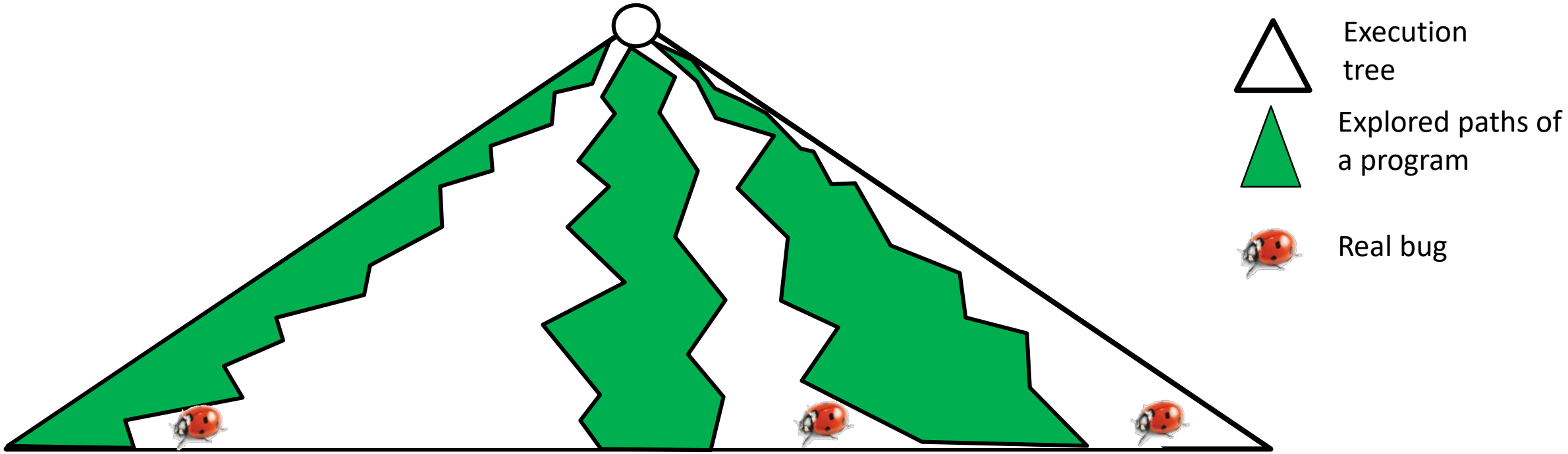
SW 테스팅

- 보안장비 프로그램에서 기존에 발견하지 못했던 신규 SW 결함 검출

Pros and Cons of Auto. Test Gen. at **System-level**

› Pros: **No false alarms**

› Cons: Low bug detection power due to **large search space**

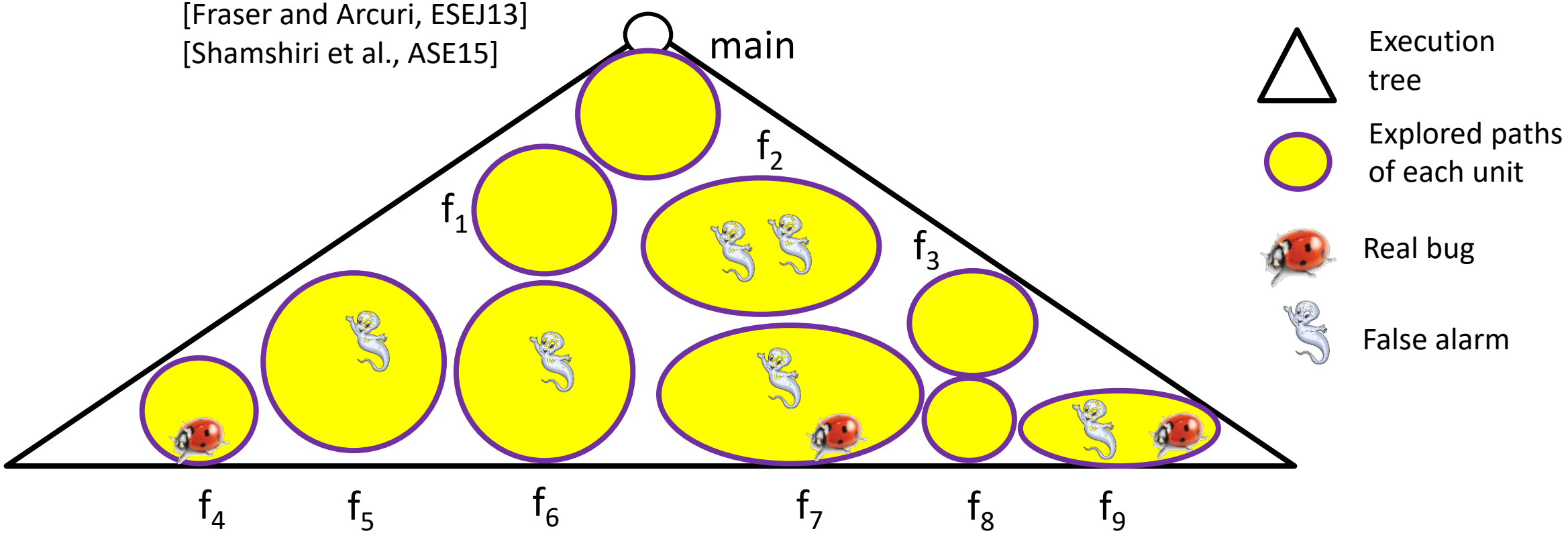


Pros and Cons of Auto. Test Gen. at **Unit-level**

› Pros: High bug detection power for **small search space**

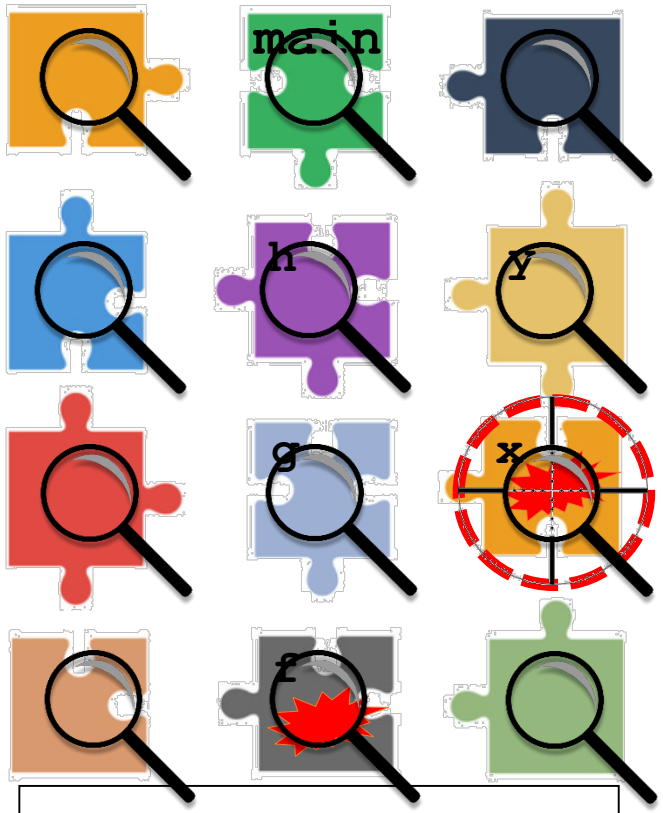
› Cons: **Many false alarms** due to over-approximated context of a unit

[Gross et al., ISSTA12]
[Fraser and Arcuri, ESEJ13]
[Shamshiri et al., ASE15]

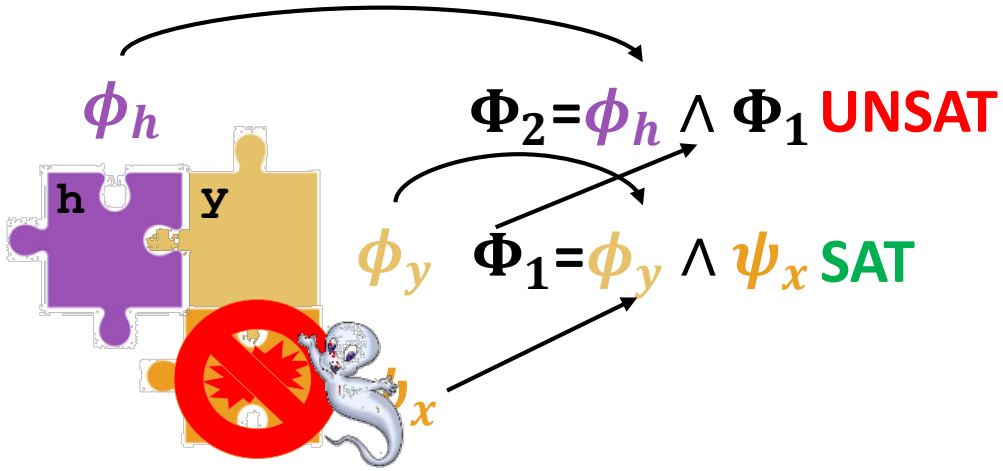


Main Idea of FOCAL++:

Unit-level failure identification



Composing system-level paths using function summaries



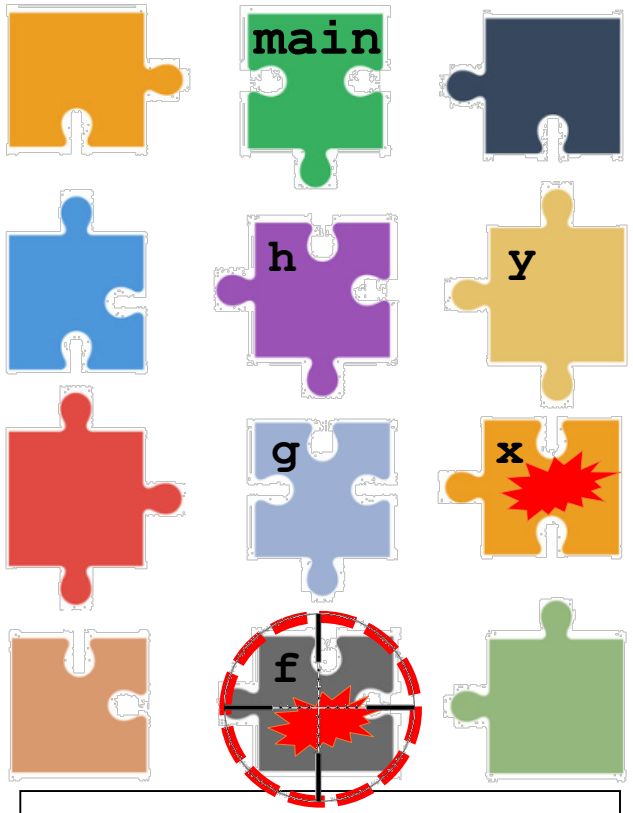
High bug detection of *unit testing*

+

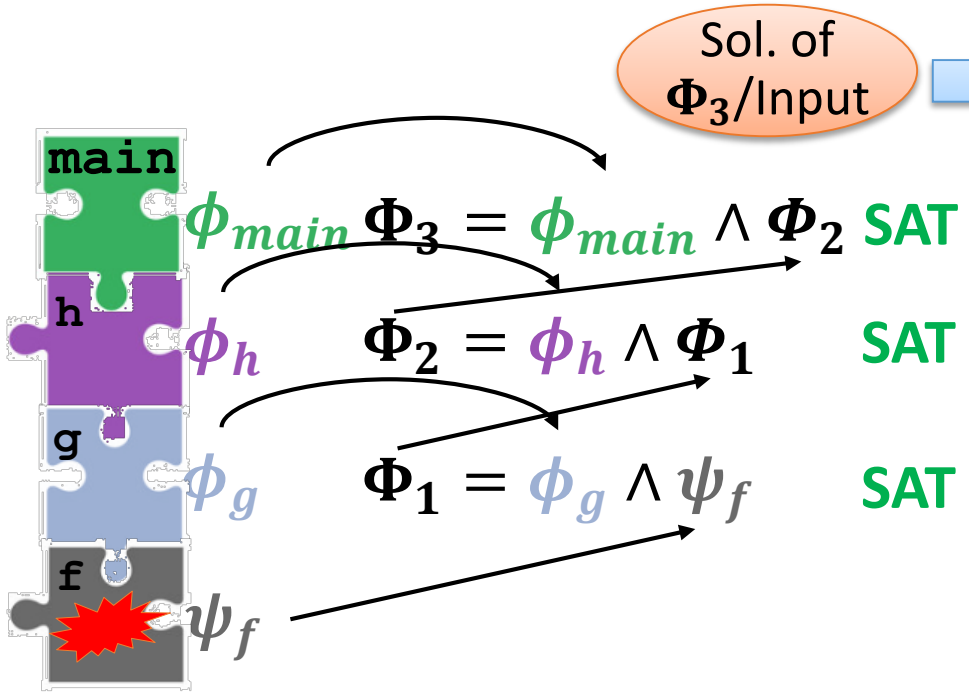
False alarm filtering through **unit context** in *system-level*

Main Idea of FOCAL:

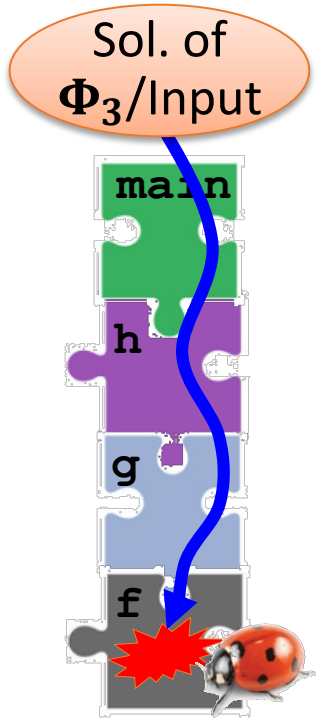
Unit-level failure identification



Composing system-level paths using function summaries



Bug confirmed via system testing



High bug detection of *unit testing*

+

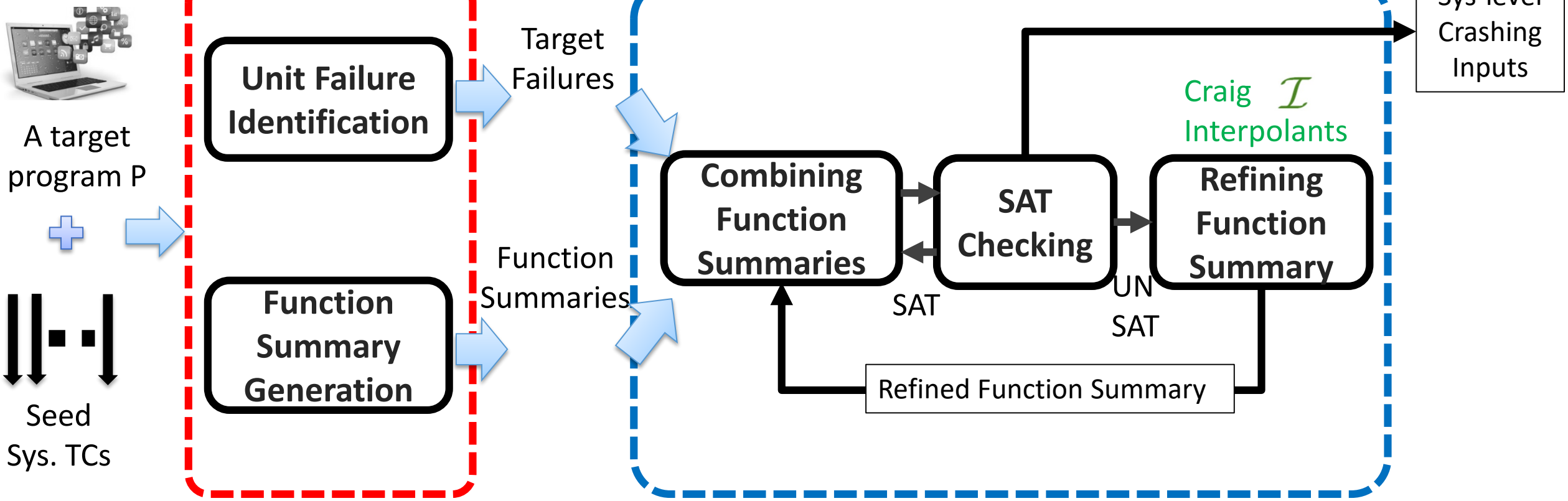
False alarm filtering through **unit context** in *system-level*

→

Detecting many bugs with **no false alarm**

Overview of FOCAL (**FO**cused **C**omposition**AL** concolic testing)

: published at FSE 2019



Overview of FOCAL++

(**FO**cused **C**omposition**AL** concolic testing)

A target program P

Seed Sys. TCs

Reducing False Alarms

Carving Realistic Unit Context

Unit-level Analysis (Fuzzing, Symbolic Exec, CBMC, Static Annalysis)

Unit Failure Identification

Function Summary Generation

Target Failures

Function Summaries

System-level Backward (and Forward) Compositional Symbolic Analysis

Refined Function Summary

Comb. Func. Sum.

SAT Checking

Refining Func. Sum.

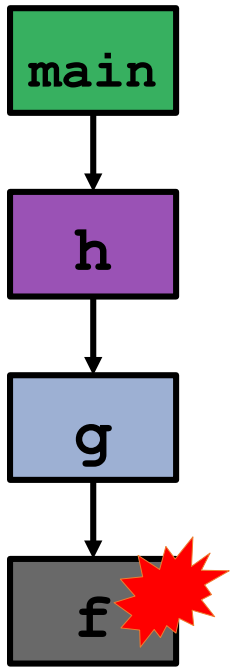
SAT

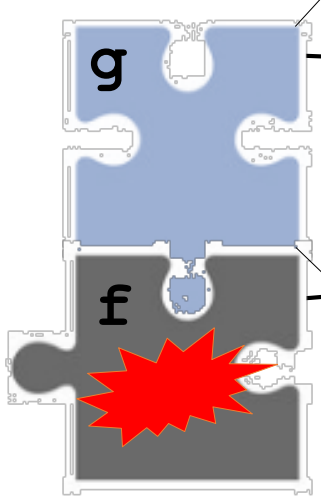
UN SAT

Sys-level Crashing Inputs

Combining Function Summaries

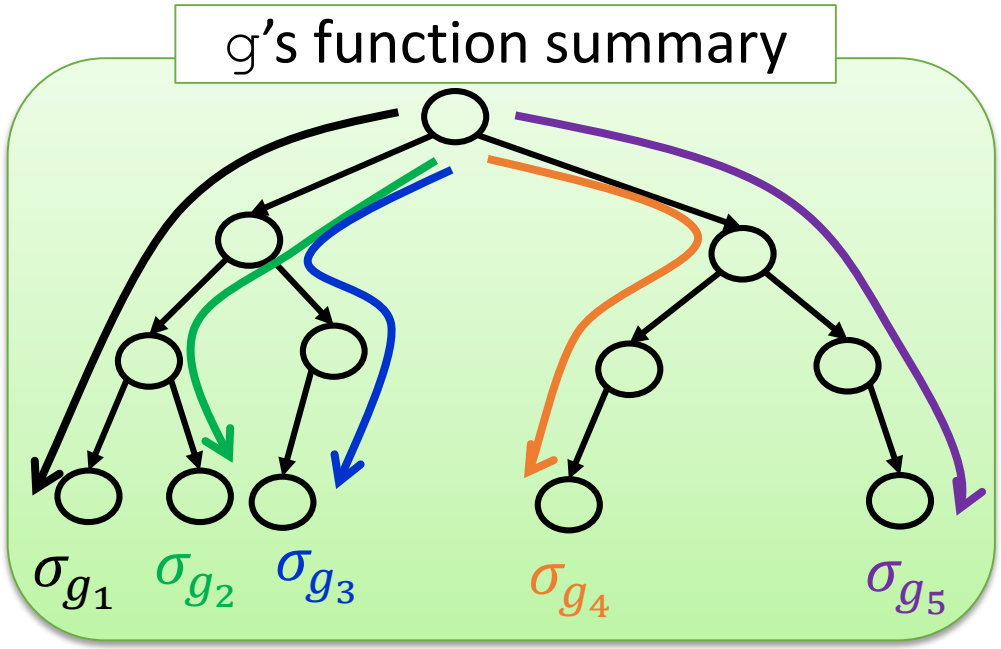
Combining function summaries from error-revealing function `f` to `main`





$\Phi_1 = \phi_g \wedge \psi_f$ **SAT**

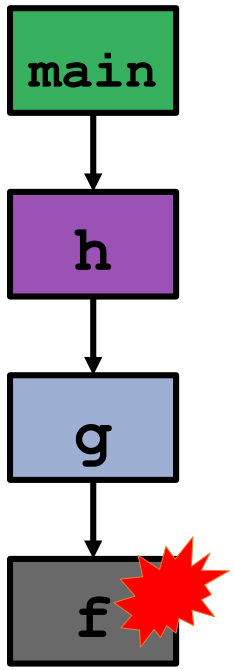
› Ex1. Function summary: a disjunction of symbolic path formulas explored

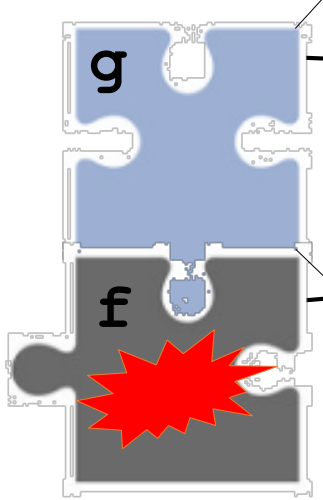


$\phi_g := \sigma_{g_1} \vee \sigma_{g_2} \vee \sigma_{g_3} \vee \sigma_{g_4} \vee \sigma_{g_5}$

Combining Function Summaries

Combining function summaries from error-revealing function `f` to `main`



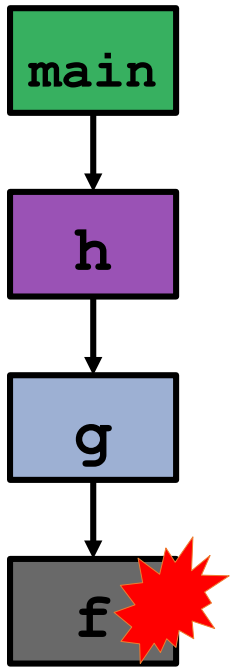


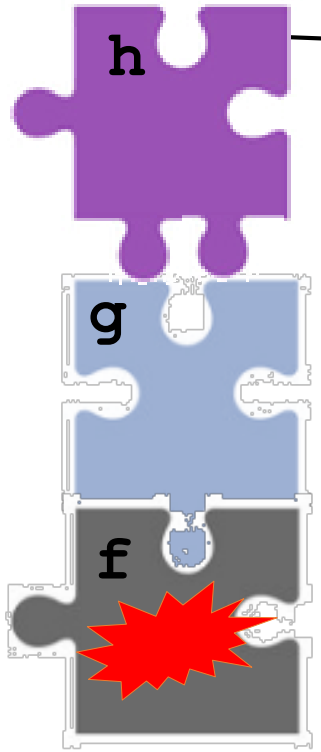
$\Phi_1 = \phi_g \wedge \psi_f$ **SAT**

- Ex2. Disjunction of symbolic formulas obtained from fuzzed paths (for fast summary gen.)
- Ex3. SMT/SAT formula of a target unit obtained from CBMC

- Ex4. Model learned from input/output of a **target unit**
- Ex5. Model learned from input/output of **interacting units** of a target unit

Ex. Function Summary Refinement using Craig Interpolants

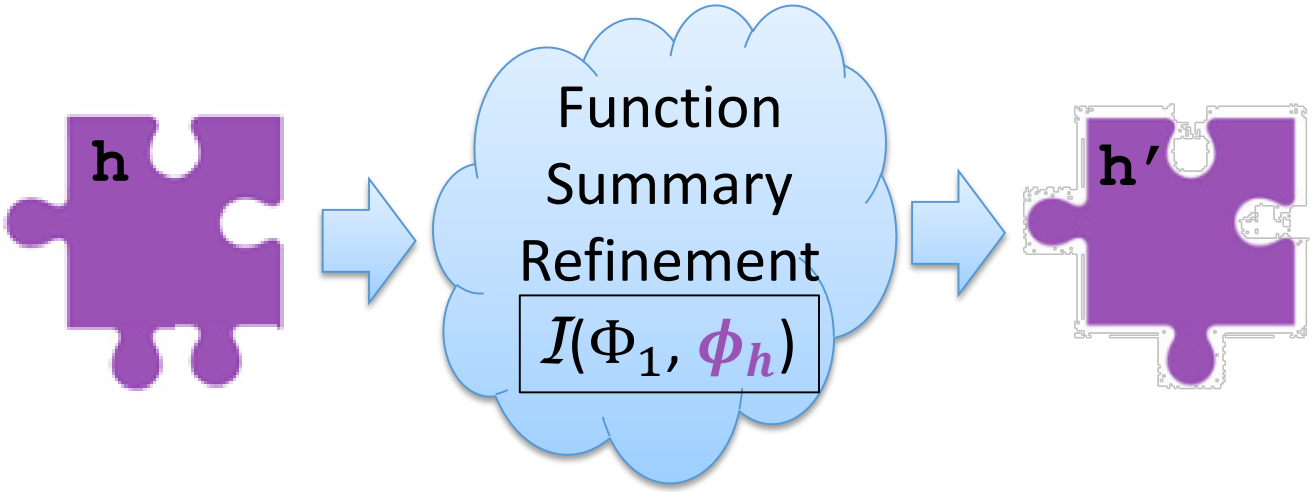




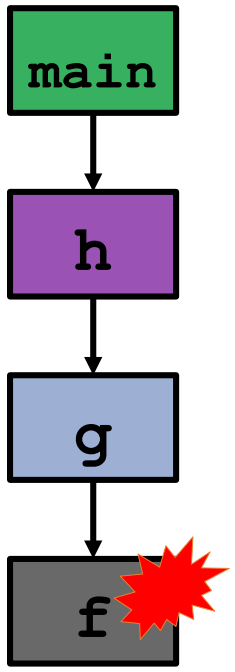
$\Phi_2 = \phi_h \wedge \Phi_1$ **UNSAT**

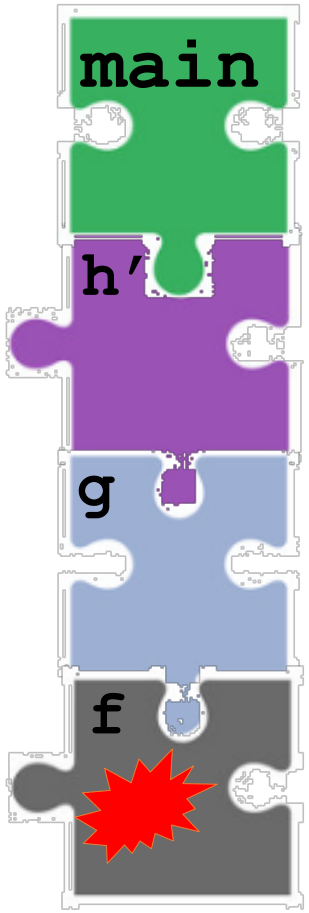
$\Phi_1 = \phi_g \wedge \psi_f$ **SAT**

Refinement using Craig Interpolant $\mathcal{I}(\Phi_1, \phi_h)$



Continue Compositional Analysis with Refined Summaries



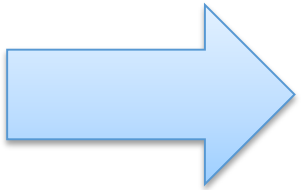


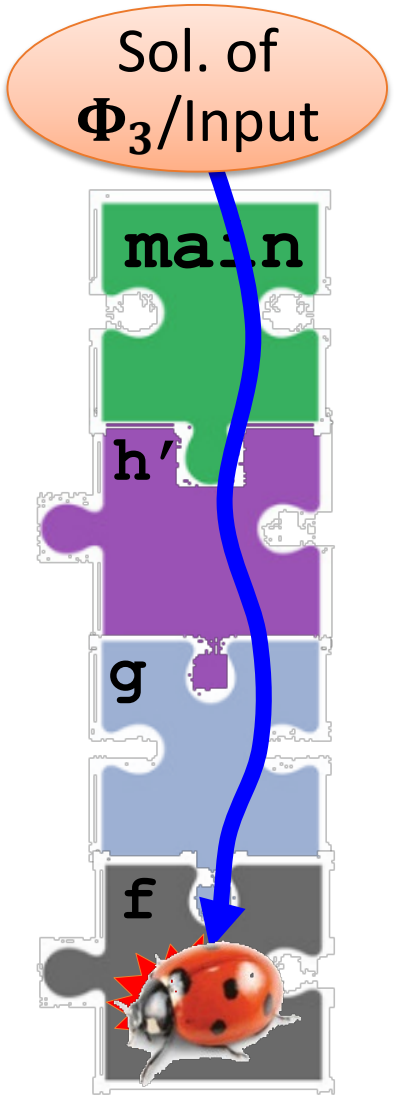
$\Phi_3 = \phi_{main} \wedge \Phi'_2$ **SAT**

$\Phi'_2 = \phi'_h \wedge \Phi_1$ **SAT**

$\Phi_1 = \phi_g \wedge \psi_f$ **SAT**

Constructing
a system-level TC





Research Questions

› RQ1, RQ2: Bug detection power of FOCAL

RQ1: **How many target crash bugs does FOCAL detect?**

7 Programs (148 KLOC)

BASH, GNU flex, GNU Make, GZIP, Vim, GNU Grep, GNU SED

RQ2: **How many new crash bugs does FOCAL detect?**

13 Programs (213 KLOC)

{JSON}
jsmn
json-c
...

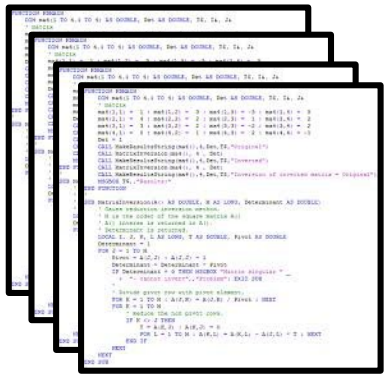
RegEx
gnulib regex
sxmlc

XML
expat
libxml2

FOCAL Crash Bug Benchmark

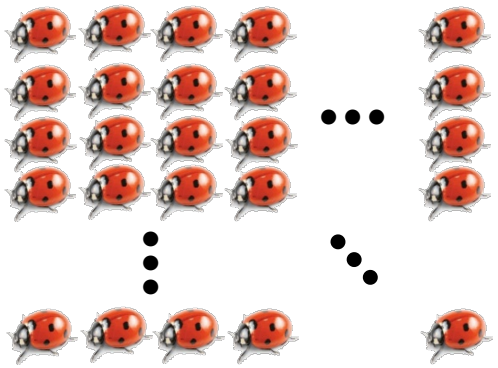
7 Programs(148 KLOC)

BASH
GNU flex
GNU Grep
GZIP
GNU Make
GNU SED
Vim



19,108 commits

19,108 commits
587 crash fix



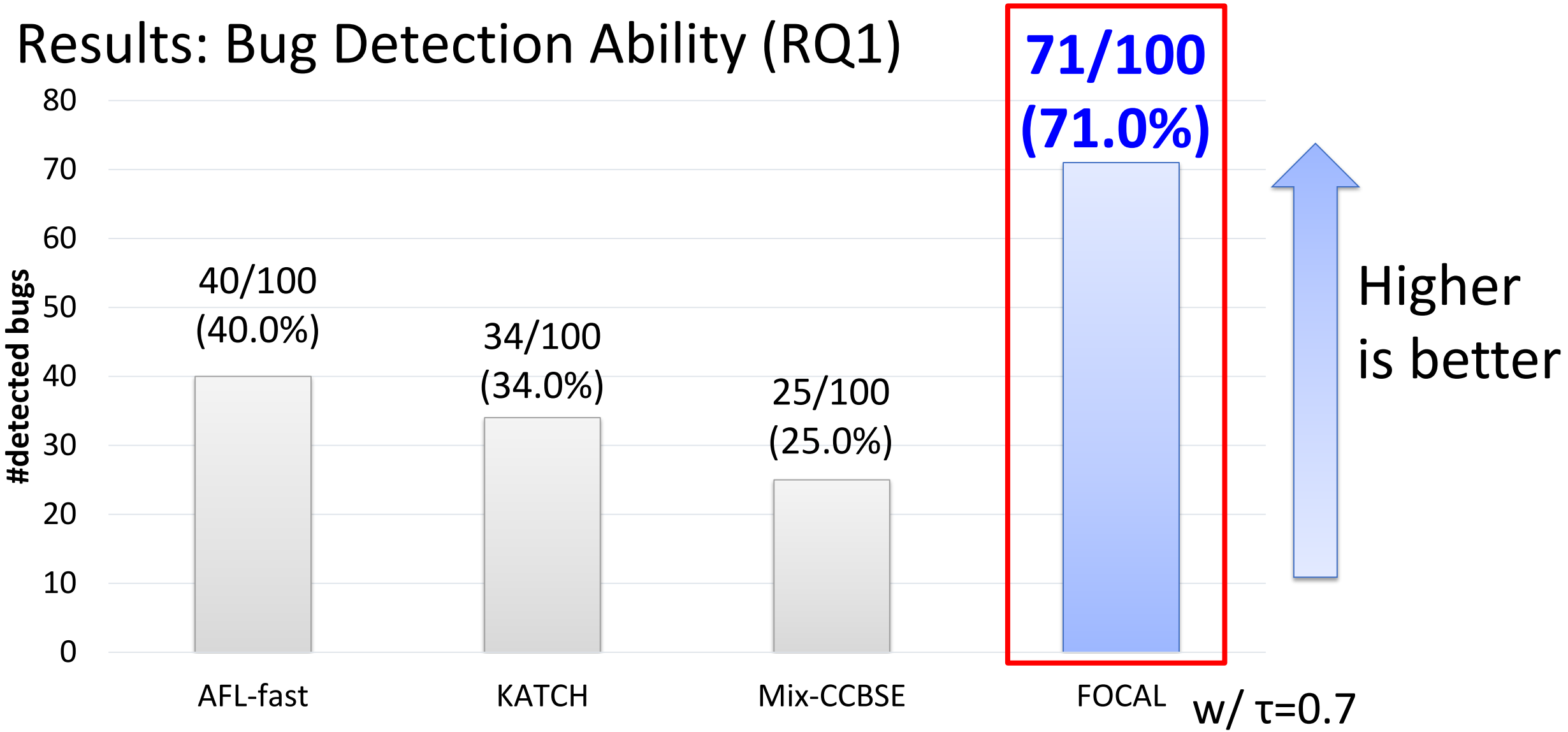
100 Real-world Target Crash Bug Benchmark

1993

Jul. 2018

https://sites.google.com/view/focal-fse19

Results: Bug Detection Ability (RQ1)



FOCAL spent **6.3 hours** on 100 machines

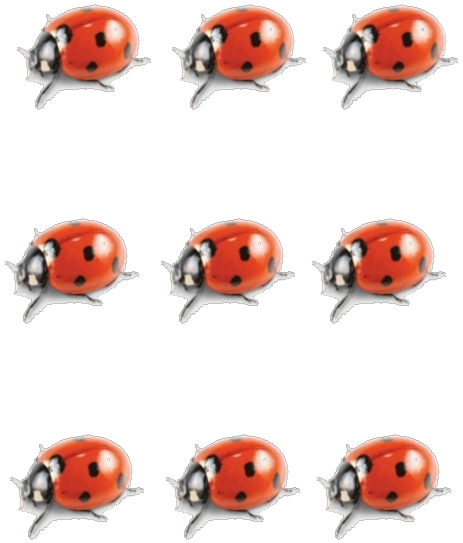
Results: New Bug Detection (RQ2)

FOCAL detected **13 new bugs in 12 text parsing programs (213 KLOC)**

{JSON}

9 bugs

jsmn
json-c
json-parser
jsonsl
libpcre
microjson
mjson
parson



RegEx

2 bugs

gnulib regex
sxmlc

XML

2 bugs

expat
libxml2