tcs Research

Inventing for impact

VeriFuzz 1.4: Good Seeds For Fuzzing

Ravindra Metta r.metta@tcs.com
Dr. Raveendra Medicherla
Dr. Hrishikesh Karmarkar

tcs Research
Technische Universität München

24 April 2023

VeriFuzz

Fuzzing centric – Modified AFL fuzzer

Co-operative – BMC for initial seeds

Program analysis/Transformation

Classification – Categorization of programs using Decision trees

BMC vs Fuzz : Problem Illustration

Efficient (shorter time)	**BMC** ✓	**CGF** ✗
Effective (better coverage)	**BMC** ✗	**CGF** ✓

BMC vs Fuzz : Problem Illustration

Efficient (shorter time)	**BMC** ✓	**CGF** ✗
Effective (better coverage)	**BMC** ✗	**CGF** ✓

```
int32 x=input(), y=input();
if(y == 0x0123ABCD){ // hard for fuzzer
 while(*){ // unknown #iterations
  if(x % 3 == 1) ... // easy for both
  else ...
 }                   // hard for BMC
int32 z=input();
if(z % 3 == 2) ...  // easy for both
```

INPUTS : *x* and *y*

INPUT : *z*

BMC vs Fuzz : Problem Illustration

Efficient (shorter time)	**BMC** ✓	**CGF** ✗
Effective (better coverage)	**BMC** ✗	**CGF** ✓

```
int32 x=input(), y=input();
if(y == 0x0123ABCD){ // hard for fuzzer
 while(*){ // unknown #iterations
  if(x % 3 == 1) ... // easy for both
  else ...
 }                    // hard for BMC
int32 z=input();
if(z % 3 == 2) ...  // easy for both
```

INPUTS : *x* and *y*

CGF gets stuck

INPUT : *z*

BMC vs Fuzz : Problem Illustration

Efficient (shorter time)	**BMC** ✓	**CGF** ✗
Effective (better coverage)	**BMC** ✗	**CGF** ✓

```
int32 x=input(), y=input();
if(y == 0x0123ABCD){ // hard for fuzzer
 while(*){ // unknown #iterations
  if(x % 3 == 1) ... // easy for both
  else ...
 }                    // hard for BMC
int32 z=input();
if(z % 3 == 2) ...   // easy for both
```

INPUTS : *x* and *y*	
CGF gets stuck	
BMC gets stuck	
BMC doesn't scale	
INPUT : *z*	
Neither reaches	

BMC Short Unwind + Fuzz : This Problem Solved

Efficient (shorter time)	**BMC** ✓	**CGF** ✗	**BMC+CGF** ✓
Effective (better coverage)	**BMC** ✗	**CGF** ✓	**BMC+CGF** ✓

```
int32 x=input(), y=input();
if(y == 0x0123ABCD){ // hard for fuzzer
 while(*){ // unknown #iterations
  if(x % 3 == 1) ... // easy for both
  else ...
 }                    // hard for BMC
int32 z=input();
if(z % 3 == 2) ...  // easy for both
```

INPUTS : *x* and *y*

CGF gets stuck

BMC gets stuck

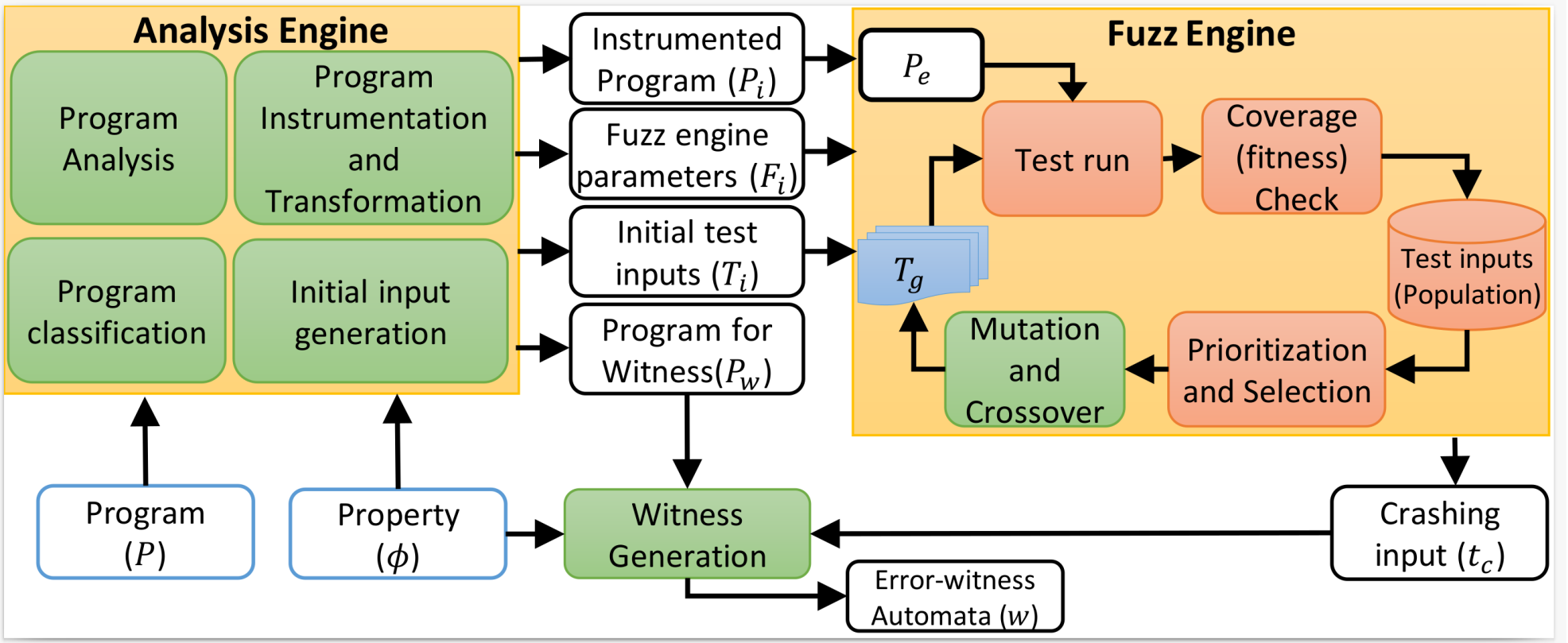
BMC doesn't scale

INPUT : *z*

Neither reaches

Earlier VeriFuzz (1.0) : BMC only for sequentialized programs

- **AFL-based, with CBMC for *complete seeds* (test inputs) for sequentialized programs**
  (Competition Contribution in TACAS'19)



Solution Part 1: Generate Incomplete inputs, and extend them

- Generate *incomplete seeds*: BMC for *short* bound of 2

Test 1: {x = 7, y=0x0123ABCD, z=?}

Test 2: {x = 12, y=0x0123ABCD, z=?}

- Complete z randomly
  - Say, z = 0
  - {x = 7, y=0x0123ABCD, z=0}
  - {x = 12, y=0x0123ABCD, z=0}

```
x=input(); y=input();
if (*){    // unwinding#1 of the loop
    if(x % 3 == 1) ...
    else ...
} else goto loop_end;
if (*){    // unwinding#2 of the loop
    if(x % 3 == 1) ...
    else ...
} else goto loop_end;
...    // #unwinding = ?    (termination)
...    // 2^#unwinding paths (explosion)
loop_end:
int32 z=input();
if(z % 3 == 2) ...   // easy for both
```

Solution Part 2: Remedial Strategies for BMC

- ***When*** **does BMC fail for short unwindings?**
  - **Complex features**
  - **Once BMC fails, tools give up**
- ***Why*** **does BMC fail?**
  - **Evaluated a variety of benchmarks**
  - **Failures in different translation phases**
- ***Can*** **this be remediated?**
  - **In some cases, "yes", and in some cases "not yet".**

Remedial Strategy 1

- ***Why***: **large loops, large loop bodies, recursive calls**
- ***Remedy***
  - **Re-run with shorter unwinding**
    - **CBMC option: --unwind k**
  - **Still stuck? Ignore loops (*--unwind 1)***
  - **Still stuck? Loop abstraction *(under investigation)***
    - **abstraction (DATE'15), shrink (TACAS'18), or just havoc?**

Remedial Strategy 2

- ***Why***: **array size is too big**
  - **copy-3-n-u.c: int a[1000][1500][1800]**
  - **Too many Boolean variables due to bit blasting**
- ***Remedy***
  - **Translate arrays as *uninterpreted functions***
    - **CBMC option: *--arrays-uf-always***
  - **Use a SMT solver for backend, like Z3**
    - **Support for array theory**

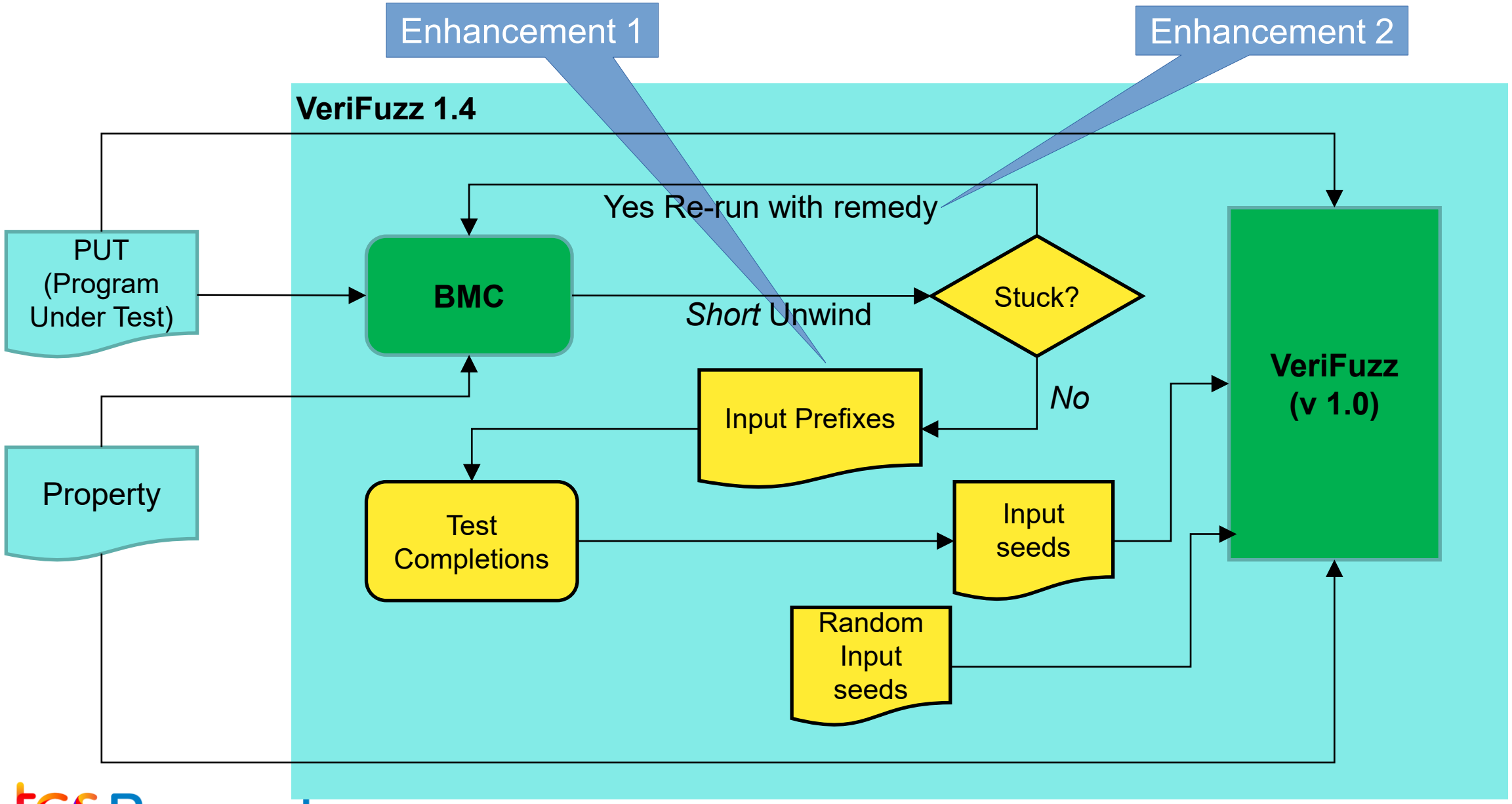
Remedial Strategy 3

- ***Why***: **access of arrays through pointer offsets**
  - **Specially in low level code, e.g. device drivers**
  - **Ackermann constraints for functional consistency**
  - **Quadratic –- 40000^2 (1.5 billion+ constraints)**
- **Remedies?**
  - **No working idea yet.**
    - **Abstractions: havoc?**

Remedial Strategy 4

- ***Why***
  - **one SAT call per coverage goal**
  - **Some call might take a lot of time**
- **Remedies**
  - **Timeout trap and output the tests of goals covered so far**
  - **In FuSeBMC (TAP'21): smart time management per goal**
  - **Slicing per path?**

VeriFuzz v1.4



VeriFuzz 1.4 Results

Track	VeriFuzz 1.1 in 2020	VeriFuzz 1.2 in 2022	VeriFuzz 1.4 in 2023
Cover-Error	636/699 in 63k s	623/776 in 13k sec	964/1173 in 16K s
Cover-Bracnhes	1577/2531 in 2.1M s	2075/3460 in 3.1M s	1650/2933 in 2.6M s

- **About 13.6 seconds per error**
- **Faster than other top tools in Cover-Error 2022**
- **Detected errors in industrial code**

Tool Issues

- **Floating point divergence issues**
  - **cdaudio.c**
- **Tool bugs**
  - **Lost score in Busy-Box Memsafety**

Future

- **Research → Concolic Generalization**
  - **Bidirectional cooperation between BMC and Fuzzing**
- **Tooling – Parameter driven AFL++, LLVM migration**
- **For academic evaluation : VeriFuzz.Tool@tcs.com**

Thank you

Copyright © 2021 Tata Consultancy Services Limited