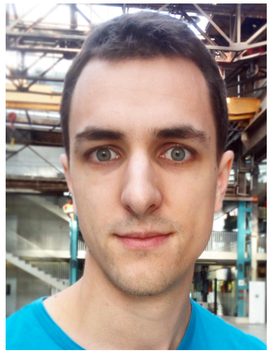
# On the Unusual Effectiveness of Type-aware Operator Mutations for Testing SMT Solvers

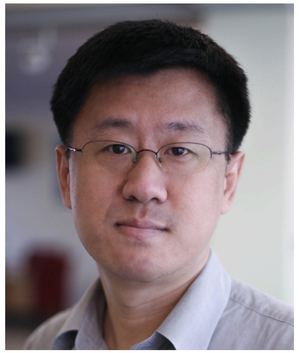

Dominik Winterer*
ETH Zurich, Switzerland

Chengyu Zhang*
East China Normal University, China

Zhendong Su
ETH Zurich, Switzerland

(*Equal contributions)




OOPSLA, 17 Nov, 2020, Online

# SMT Problem

$$\varphi : x > 0 \wedge x < 0$$

# SMT Problem

$$\varphi : x > 0 \wedge x < 0$$

**UNSAT**

# SMT Problem

$$\varphi : x > 0 \wedge x < 1$$

# SMT Problem

$$\varphi : x > 0 \wedge x < 1$$

**SAT**

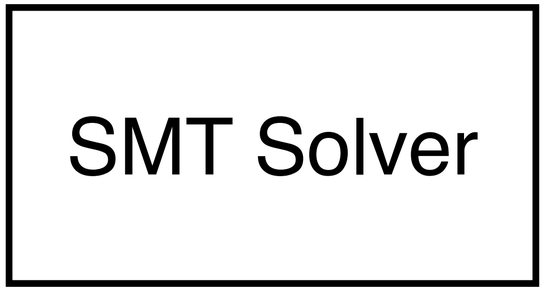
# SMT Problem

$$x = 0.5$$

$$\varphi : x > 0 \wedge x < 1$$

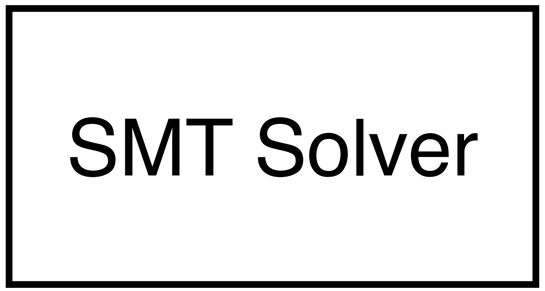
**SAT**

# SMT Solver

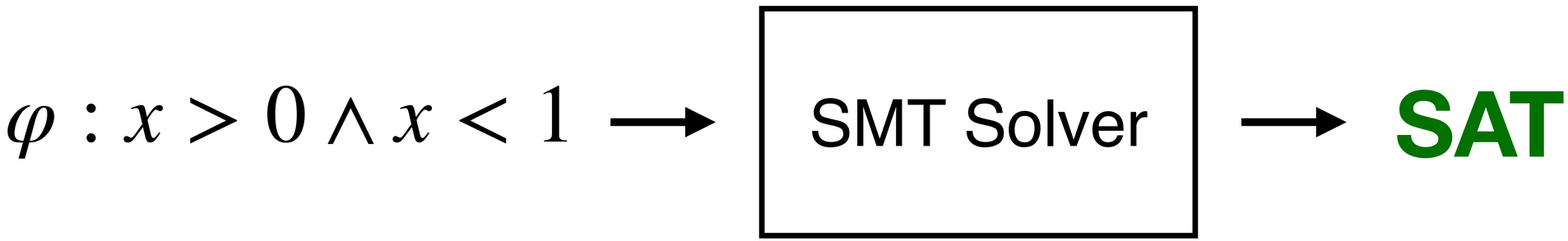


# SMT Solver

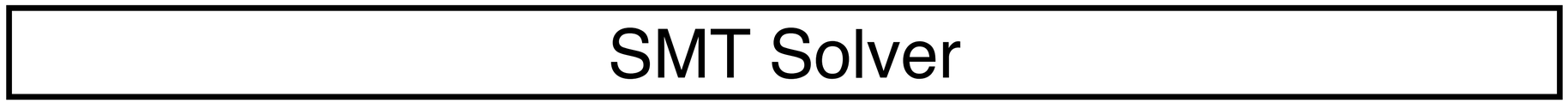
$$\varphi : x > 0 \wedge x < 1 \longrightarrow$$



# SMT Solver

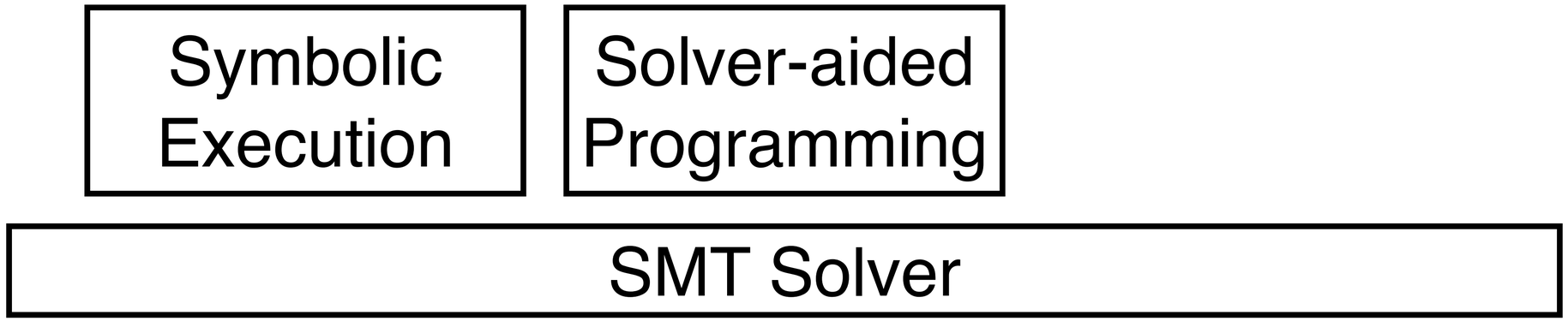


# SMT Solver

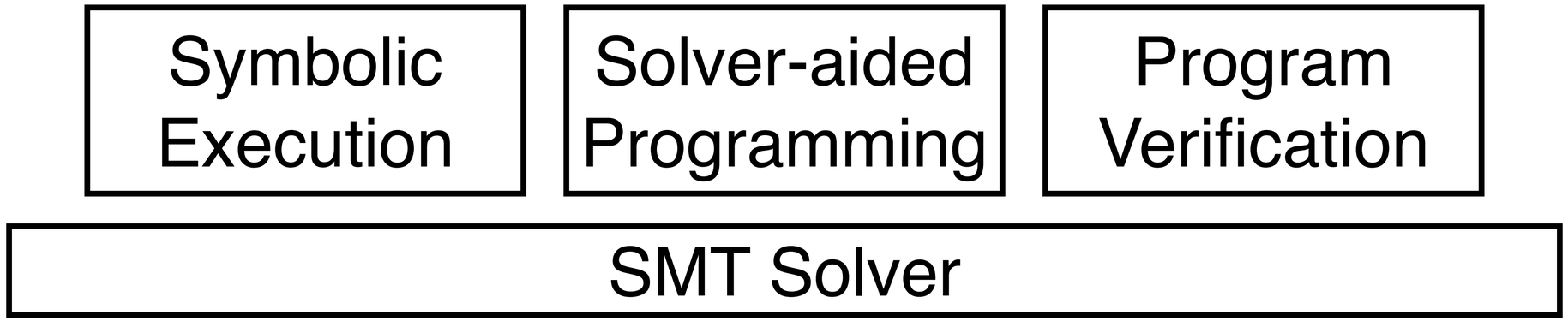


# SMT Solver

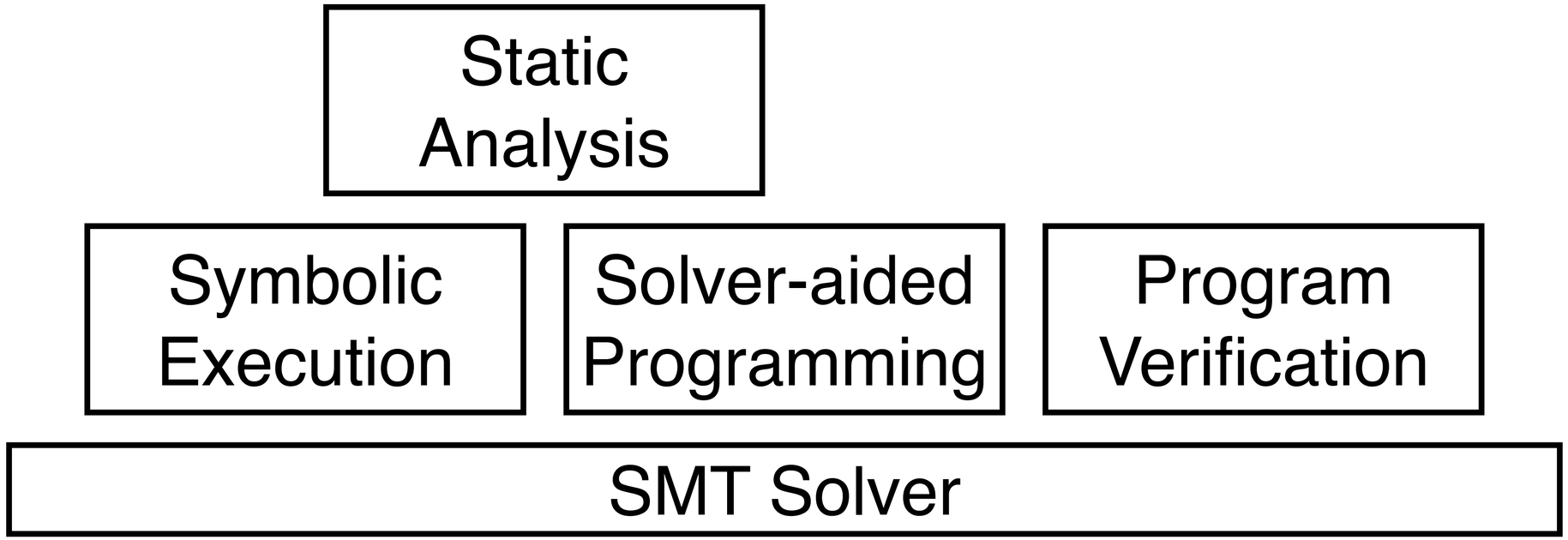
# SMT Solver



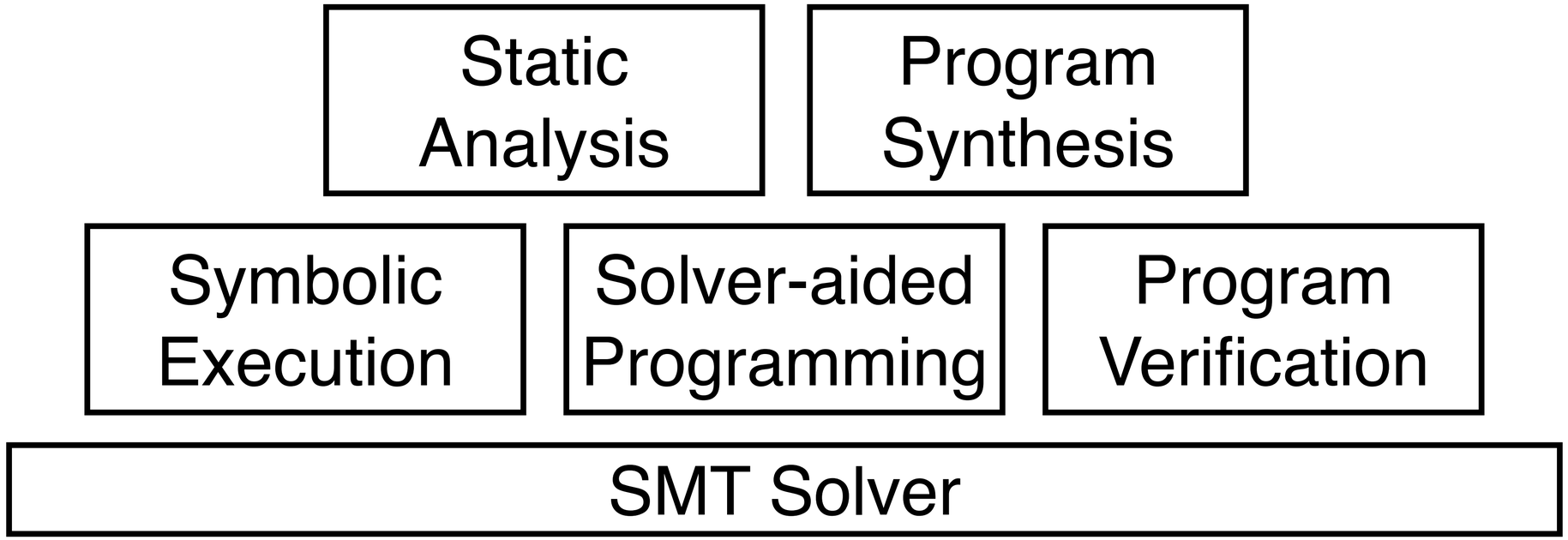
# SMT Solver


Symbolic Execution
Solver-aided Programming
Program Verification
SMT Solver

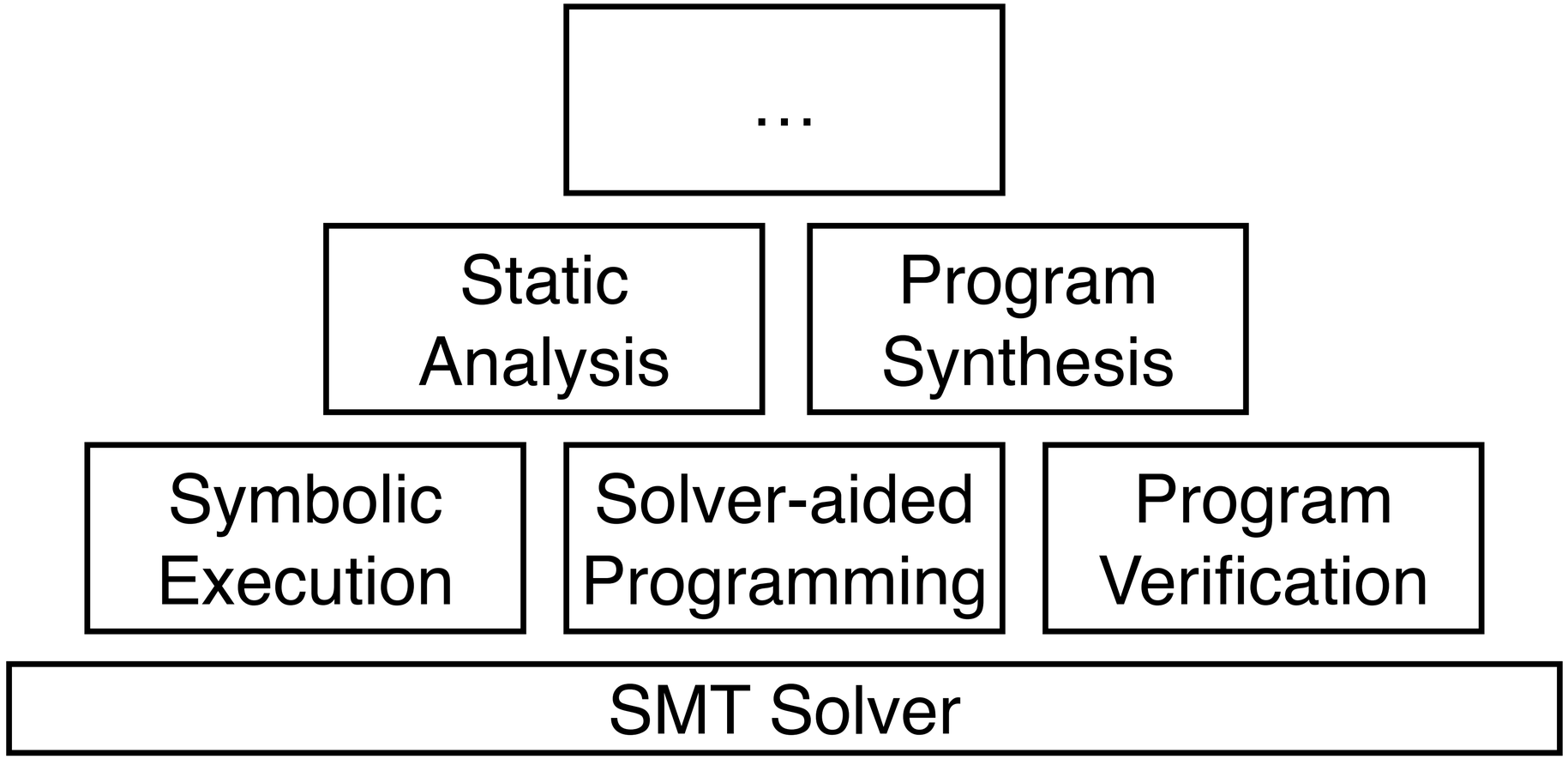
# SMT Solver


Static
Analysis
Symbolic
Execution
Solver-aided
Programming
Program
Verification
SMT Solver

# SMT Solver


Static Analysis
Program Synthesis
Symbolic Execution
Solver-aided Programming
Program Verification
SMT Solver

# SMT Solver


...
Static Analysis
Program Synthesis
Symbolic Execution
Solver-aided Programming
Program Verification
SMT Solver

# SMT-LIB language

$$\varphi : x > 0 \wedge x < 1$$

# SMT-LIB language

$$\varphi : x > 0 \wedge x < 1 \longrightarrow$$

```
(declare-fun x () Real)
(assert (and (> x 0)(< x 1)))
(check-sat)
```

# SMT-LIB language

$$\varphi : x > 0 \wedge x < 1 \longrightarrow$$

```
(declare-fun x () Real)
(assert (and (> x 0)(< x 1)))
(check-sat)
```

# SMT-LIB language

$$\varphi : x > 0 \wedge x < 1 \longrightarrow$$

```
(declare-fun x () Real)
(assert (and (> x 0)(< x 1)))
(check-sat)
```

# SMT-LIB language

$$\varphi : x > 0 \wedge x < 1 \longrightarrow$$

```
(declare-fun x () Real)
(assert (and (> x 0)(< x 1)))
(check-sat)
```

# SMT Solver Bug

```
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)
```

# SMT Solver Bug

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔
```

# SMT Solver Bug

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat   ✘
```

# SMT Solver Bug

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat   ✘
```

https://github.com/Z3Prover/z3/issues/3973

# Related Work

| | Confirmed Bugs in Z3 | | Confirmed Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz [1] | 0 | 0 | - | - |
| BanditFuzz [2] | 0 | 0 | - | - |
| Bugariu et al. [3] | 1 | 3 | 0 | 0 |
| YinYang [4] | 24 | 36 | 5 | 8 |
| STORM [5] | 17 | 21 | 0 | 0 |


[1] Dmitry Blotsky, Federico Mora, Murphy Berzish, Yunhui Zheng, Ifaz Kabir, and Vijay Ganesh. StringFuzz: A fuzzer for string solvers. In CAV, 2018.
[2] Joseph Scott, Federico Mora, and Vijay Ganesh. BanditFuzz: Fuzzing SMT solvers with reinforcement learning. In CAV, 2020.
[3] Alexandra Bugariu and Peter Müller. Automatically testing string solvers. In ICSE, 2020.
[4] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating SMT solvers via semantic fusion. In PLDI, 2020
[5] Muhammad Numair Mansur, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Detecting critical bugs in SMT colvers using blackbox mutational fuzzing. In FSE, 2020.

# Related Work

| | Confirmed Bugs in Z3 | | Confirmed Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz [1] | 0 | 0 | - | - |
| BanditFuzz [2] | 0 | 0 | - | - |
| Bugariu et al. [3] | 1 | 3 | 0 | 0 |
| YinYang [4] | 24 | 36 | 5 | 8 |
| STORM [5] | 17 | 21 | 0 | 0 |


[1] Dmitry Blotsky, Federico Mora, Murphy Berzish, Yunhui Zheng, Ifaz Kabir, and Vijay Ganesh. StringFuzz: A fuzzer for string solvers. In CAV, 2018.
[2] Joseph Scott, Federico Mora, and Vijay Ganesh. BanditFuzz: Fuzzing SMT solvers with reinforcement learning. In CAV, 2020.
[3] Alexandra Bugariu and Peter Müller. Automatically testing string solvers. In ICSE, 2020.
[4] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating SMT solvers via semantic fusion. In PLDI, 2020
[5] Muhammad Numair Mansur, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Detecting critical bugs in SMT colvers using blackbox mutational fuzzing. In FSE, 2020.

# Related Work

| | Confirmed Bugs in Z3 | | Confirmed Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz [1] | 0 | 0 | - | - |
| BanditFuzz [2] | 0 | 0 | - | - |
| Bugariu et al. [3] | 1 | 3 | 0 | 0 |
| YinYang [4] | 24 | 36 | 5 | 8 |
| STORM [5] | 17 | 21 | 0 | 0 |


[1] Dmitry Blotsky, Federico Mora, Murphy Berzish, Yunhui Zheng, Ifaz Kabir, and Vijay Ganesh. StringFuzz: A fuzzer for string solvers. In CAV, 2018.
[2] Joseph Scott, Federico Mora, and Vijay Ganesh. BanditFuzz: Fuzzing SMT solvers with reinforcement learning. In CAV, 2020.
[3] Alexandra Bugariu and Peter Müller. Automatically testing string solvers. In ICSE, 2020.
[4] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating SMT solvers via semantic fusion. In PLDI, 2020
[5] Muhammad Numair Mansur, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Detecting critical bugs in SMT colvers using blackbox mutational fuzzing. In FSE, 2020.

# Related Work

| | Confirmed Bugs in Z3 | | Confirmed Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz [1] | 0 | 0 | - | - |
| BanditFuzz [2] | 0 | 0 | - | - |
| Bugariu et al. [3] | 1 | 3 | 0 | 0 |
| YinYang [4] | 24 | 36 | 5 | 8 |
| STORM [5] | 17 | 21 | 0 | 0 |
| Type-aware operator mutation | 157 | 578 | 27 | 241 |


[1] Dmitry Blotsky, Federico Mora, Murphy Berzish, Yunhui Zheng, Ifaz Kabir, and Vijay Ganesh. StringFuzz: A fuzzer for string solvers. In CAV, 2018.
[2] Joseph Scott, Federico Mora, and Vijay Ganesh. BanditFuzz: Fuzzing SMT solvers with reinforcement learning. In CAV, 2020.
[3] Alexandra Bugariu and Peter Müller. Automatically testing string solvers. In ICSE, 2020.
[4] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating SMT solvers via semantic fusion. In PLDI, 2020
[5] Muhammad Numair Mansur, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Detecting critical bugs in SMT colvers using blackbox mutational fuzzing. In FSE, 2020.

# Related Work

| | Confirmed Bugs in Z3 | | Confirmed Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz [1] | 0 | 0 | - | - |
| BanditFuzz [2] | 0 | 0 | - | - |
| Bugariu et al. [3] | 1 | 3 | 0 | 0 |
| YinYang [4] | 24 | 36 | 5 | 8 |
| STORM [5] | 17 | 21 | 0 | 0 |
| Type-aware operator mutation | 157 | 578 | 27 | 241 |

SIMPLE BUT UNUSUAL EFFECTIVE APPROACH


[1] Dmitry Blotsky, Federico Mora, Murphy Berzish, Yunhui Zheng, Ifaz Kabir, and Vijay Ganesh. StringFuzz: A fuzzer for string solvers. In CAV, 2018.
[2] Joseph Scott, Federico Mora, and Vijay Ganesh. BanditFuzz: Fuzzing SMT solvers with reinforcement learning. In CAV, 2020.
[3] Alexandra Bugariu and Peter Müller. Automatically testing string solvers. In ICSE, 2020.
[4] Dominik Winterer, Chengyu Zhang, and Zhendong Su. Validating SMT solvers via semantic fusion. In PLDI, 2020
[5] Muhammad Numair Mansur, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. Detecting critical bugs in SMT colvers using blackbox mutational fuzzing. In FSE, 2020.

# Type-aware operator mutation

```
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)
```

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

```
(assert (forall ((a Int))
        (exist  ((b Int))
        (   (* 2 b) a))))
(check-sat)
```

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

```
(assert (forall ((a Int))
        (exist  ((b Int))
        (   (* 2 b) a))))
(check-sat)
```

Int Int

Int Int

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

Int Int

```
(assert (forall ((a Int))
        (exist  ((b Int))
        (   (* 2 b) a))))
(check-sat)
```

Int Int

Operator candidates:

=
>
<
>=
<=

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

Int Int

```
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)
```

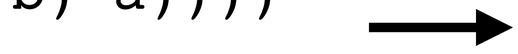
Int Int

Operator candidates:

- =
- >
- <
- >=
- <=

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```



```
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)
```

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat

$ z3 bug.smt2
sat
```

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat   ✘
```

# Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat   ✘
```

https://github.com/Z3Prover/z3/issues/3973

# Type-aware operator mutation chain

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (> (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (> (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (> (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (> (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (> (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (/ a a ) 1))
(check-sat)
```

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (> (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (* a a ) 1))
(check-sat)
```

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (/ a a ) 1))
(check-sat)
```

...

# Type-aware operator mutation chain

```
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (/ a a ) 1))
(check-sat)
```

# Type-aware operator mutation chain

```
$ cat > bug.smt2
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (/ a a ) 1))
(check-sat)

$ cvc4 bug.smt2
sat

$ z3 bug.smt2
unsat
```

# Type-aware operator mutation chain

```
$ cat > bug.smt2
(declare-fun a () Real)
(assert (= (/ (* 2 a) a) (/ a a ) 1))
(check-sat)

$ cvc4 bug.smt2
sat    ✔

$ z3 bug.smt2
unsat  ✘
```

https://github.com/Z3Prover/z3/issues/2715

# Empirical Evaluation

# Empirical Evaluation

- Tool: OpFuzz

# Empirical Evaluation

- Tool: OpFuzz
- Bug hunting: Sep 2019 - Oct 2020

# Empirical Evaluation

- Tool: OpFuzz
- Bug hunting: Sep 2019 - Oct 2020
- Testing targets: Z3 and CVC4

# Empirical Evaluation

- Tool: OpFuzz
- Bug hunting: Sep 2019 - Oct 2020
- Testing targets: Z3 and CVC4
- Seeds: SMT-LIB benchmarks (300k+ formulas)

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 685 |
| Duplicate | 85 | 18 | 106 |

Bug status as of 30 Oct 2020

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 685 |
| Duplicate | 85 | 18 | 106 |

Bug status as of 30 Oct 2020

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 685 |
| Duplicate | 85 | 18 | 106 |

Bug status as of 30 Oct 2020

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 685 |
| Duplicate | 85 | 18 | 106 |

Bug status as of 30 Oct 2020

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Soundness | 157 | 27 | 184 |
| Invalid model | 83 | 19 | 102 |
| Crash | 316 | 185 | 501 |
| Others | 22 | 10 | 32 |

Types of confirmed bugs

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Soundness | 157 | 27 | 184 |
| Invalid model | 83 | 19 | 102 |
| Crash | 316 | 185 | 501 |
| Others | 22 | 10 | 32 |

Types of confirmed bugs

# Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Soundness | 157 | 27 | 184 |
| Invalid model | 83 | 19 | 102 |
| Crash | 316 | 185 | 501 |
| Others | 22 | 10 | 32 |

Types of confirmed bugs

# Bug Findings

| Logic | Z3 | CVC4 | Total |
|---|---|---|---|
| Soundness | 157 | 27 | 184 |
| Invalid model | 83 | 19 | 102 |
| Crash | 316 | 185 | 501 |
| Others | 22 | 10 | 32 |

Types of confirmed bugs

# Bug Findings

| #Options | Z3 | CVC4 | Total |
|---|---|---|---|
| Default | 388 | 101 | 489 |
| 1 | 109 | 67 | 176 |
| 2 | 45 | 31 | 76 |
| >= 3 | 36 | 42 | 78 |

Number of options for triggering the bugs

# Bug Findings

| #Options | Z3 | CVC4 | Total |
|---|---|---|---|
| Default | 388 | 101 | 489 |
| 1 | 109 | 67 | 176 |
| 2 | 45 | 31 | 76 |
| >= 3 | 36 | 42 | 78 |

Number of options for triggering the bugs

# Bug Findings

| #Options | Z3 | CVC4 | Total |
|---|---|---|---|
| Default | 388 | 101 | 489 |
| 1 | 109 | 67 | 176 |
| 2 | 45 | 31 | 76 |
| >= 3 | 36 | 42 | 78 |

Number of options for triggering the bugs

# Comparison to Previous Approaches

| | Bugs in Z3 | | Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz | 0 | 0 | - | - |
| BanditFuzz | 0 | 0 | - | - |
| Bugariu et al. | 1 | 3 | 0 | 0 |
| YinYang | 24 | 36 | 5 | 8 |
| STORM | 17 | 21 | 0 | 0 |
| OpFuzz | 114 | 316 | 11 | 185 |

Confirmed bugs in the default modes of the solvers

# Comparison to Previous Approaches

| | Bugs in Z3 | | Bugs in CVC4 | |
|---|---|---|---|---|
| **Approach** | soundness | all | soundness | all |
| StringFuzz | 0 | 0 | - | - |
| BanditFuzz | 0 | 0 | - | - |
| Bugariu et al. | 1 | 3 | 0 | 0 |
| YinYang | 24 | 36 | 5 | 8 |
| STORM | 17 | 21 | 0 | 0 |
| OpFuzz | 114 | 316 | 11 | 185 |

Confirmed bugs in the default modes of the solvers

F

| Approach | Bugs in Z3 | | Bugs in CVC4 | |
|---|---|---|---|---|
| | soundness | all | soundness | all |
| StringFuzz | 0 | 0 | - | - |
| BanditFuzz | 0 | 0 | - | - |
| Bugariu et al. | 1 | 3 | 0 | 0 |
| YinYang | 24 | 36 | 5 | 8 |
| STORM | 17 | 21 | 0 | 0 |
| OpFuzz | 114 | 316 | 11 | 185 |

**OpFuzz found many more soundness bugs in Z3 and CVC4's default modes than all previous approaches**

# Z3 Soundness Bug #2832

```
$ cat bug.smt2
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const c (_ BitVec 8))
(assert (= (bvxnor a b c)
           (bvxnor (bvxnor a b) c)))
(check-sat)

$ cvc4 bug.smt2
sat     ✔

$ z3 bug.smt2
unsat   ✘
```

https://github.com/Z3Prover/z3/issues/2832

# Z3 Soundness Bug #2832

```
$ cat bug.smt2
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const c (_ BitVec 8))
(assert (= (bvxnor a b c)
           (bvxnor (bvxnor a b) c)))
(check-sat)

$ cvc4 bug.smt2
sat     ✔

$ z3 bug.smt2
unsat   ✘
```

https://github.com/Z3Prover/z3/issues/2832

# Z3 Soundness Bug #2832

```
(bvxnor a b c) ≠ (not (bvxor a b c))
```

# Z3 Soundness Bug #2832

**(bvxnor a b c) ≠ (not (bvxor a b c))**

```
(bvxnor true true true) = (not (bvxor true true true))
```

# Z3 Soundness Bug #2832

**(bvxnor a b c) ≠ (not (bvxor a b c))**

```
(bvxnor true true true) = (not (bvxor true true true))
(bvnxor (bvnxor true true) true)) = (not (bvxor (bvxor true true) true))
```

# Z3 Soundness Bug #2832

**(bvxnor a b c) ≠ (not (bvxor a b c))**

```
                (bvxnor true true true) = (not (bvxor true true true))
(bvnxor (bvnxor true true) true)) = (not (bvxor (bvxor true true) true))
                    (bvnxor true true) = (not (bvxor false true))
```

# Z3 Soundness Bug #2832

**(bvxnor a b c) ≠ (not (bvxor a b c))**

```
                (bvxnor true true true) = (not (bvxor true true true))
(bvnxor (bvnxor true true) true)) = (not (bvxor (bvxor true true) true))
                    (bvnxor true true) = (not (bvxor false true))
                                  true ≠ false
```

# Z3 Soundness Bug #2832

```
$ cat bug.smt2
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const c (_ BitVec 8))
(assert (= (bvxnor a b c)
           (bvxnor (bvxnor a b) c)))
(check-sat)

$ cvc4 bug.smt2
sat     ✔

$ z3 bug.smt2
unsat   ✘
```

https://github.com/Z3Prover/z3/issues/2832

**Root Cause: Unsound rewriter**

# Z3 Soundness Bug #2830

```
$ cat bug.smt2
(declare-fun a () Int)
(declare-fun b (Int) Bool)
(assert (b 0))
(push)
(assert (distinct true (= a 0) (not (b 0))))
(check-sat)

$ cvc4 bug.smt2
unsat   ✔

$ z3 bug.smt2
sat     ✘
```

https://github.com/Z3Prover/z3/issues/2830

# Z3 Soundness Bug #2830

```
$ cat bug.smt2
(declare-fun a () Int)
(declare-fun b (Int) Bool)
(assert (b 0))
(push)
(assert (distinct true (= a 0) (not (b 0))))
(check-sat)

$ cvc4 bug.smt2
unsat   ✔

$ z3 bug.smt2
sat     ✘
```

https://github.com/Z3Prover/z3/issues/2830

**Root Cause: Last argument omitted in distinct operator**

# CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat     ✔

$ cvc4 bug.smt2
unsat   ✘
```

https://github.com/CVC4/CVC4/issues/3497

# CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat     ✔

$ cvc4 bug.smt2
unsat   ✘
```

https://github.com/CVC4/CVC4/issues/3497

# CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat     ✔

$ cvc4 bug.smt2
unsat   ✘
```

```
(str.indexof x y 1) <= (str.len x)
```

https://github.com/CVC4/CVC4/issues/3497

# CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat      ✔

$ cvc4 bug.smt2
unsat    ✘
```

https://github.com/CVC4/CVC4/issues/3497

```
(str.indexof x y 1) < (str.len x)
```

**Root Cause: Missed boundary condition**

# CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat     ✔

$ cvc4 bug.smt2
unsat   ✘
```

**ajreynol** commented on Nov 26, 2019 — Member

Thanks a lot for the bug report, this is fixed in #3499.

https://github.com/CVC4/CVC4/issues/3497

# CVC4 Soundness Bug #4469

```
$ cat bug.smt2
(set-logic QF_AUFBVLIA)
(declare-fun a () Int)
(declare-fun b (Int) Int)
(assert (distinct (b a)
   (b (b a))))
(check-sat)

$ z3 bug.smt2
sat     ✔

$ cvc4 bug.smt2
unsat   ✘
```

https://github.com/CVC4/CVC4/issues/4469

# CVC4 Soundness Bug #4469

```
$ cat bug.smt2
(set-logic QF_AUFBVLIA)
(declare-fun a () Int)
(declare-fun b (Int) Int)
(assert (distinct (b a)
   (b (b a))))
(check-sat)

$ z3 bug.smt2
sat     ✔

$ cvc4 bug.smt2
unsat   ✘
```

https://github.com/CVC4/CVC4/issues/4469

# CVC4 Soundness Bug #3475

```
$ cat bug.smt2
(set-logic ALL)
(declare-fun x () Real)
(assert (< x 0))
(assert (not (= (/ (sqrt x) (sqrt x)) x)))
(check-sat)

$ z3 bug.smt2
sat   ✔

$ cvc4 bug.smt2
unsat ✘
```

https://github.com/CVC4/CVC4/issues/3475

# CVC4 Soundness Bug #3475

```
$ cat bug.smt2
(set-logic ALL)
(declare-fun x () Real)
(assert (< x 0))
(assert (not (= (/ (sqrt x) (sqrt x)) x)))
(check-sat)

$ z3 bug.smt2
sat   ✔

$ cvc4 bug.smt2
unsat ✘
```

```
Formula is satisfiable
negative x = -1
(/ sqrt(-1) sqrt(-1)) = 1
```

https://github.com/CVC4/CVC4/issues/3475

# CVC4 Soundness Bug #3475

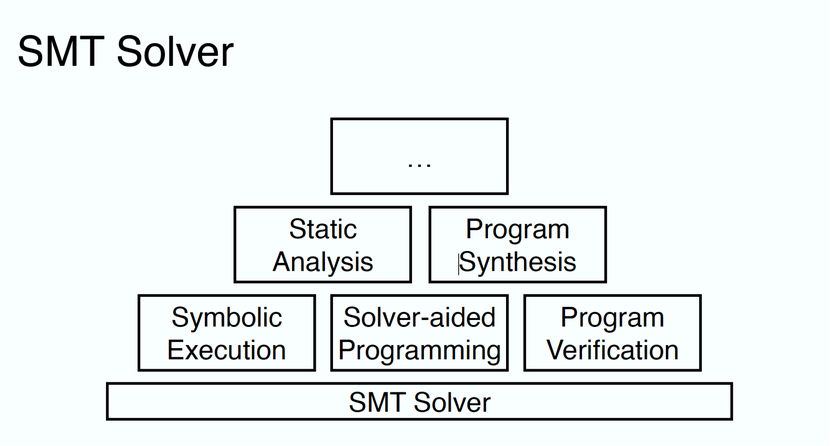
```
Simplification:
(sqrt x) = choice real y s.t. x*x = y
```

# CVC4 Soundness Bug #3475

```
Simplification:
(sqrt x) = choice real y s.t. x*x = y
```

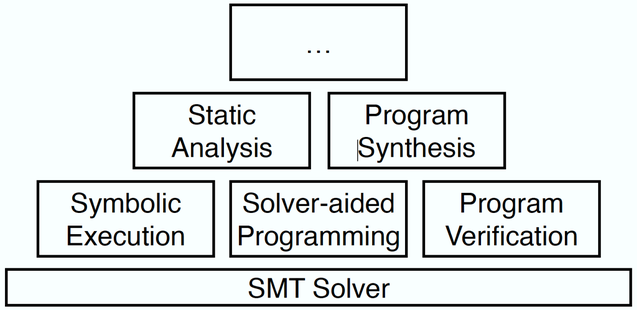
Problem: Inadmissible for negative y (since no real x exists)

# Summary


SMT Solver
…
Static Analysis
Program Synthesis
Symbolic Execution
Solver-aided Programming
Program Verification
SMT Solver

# Summary

## SMT Solver




## Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

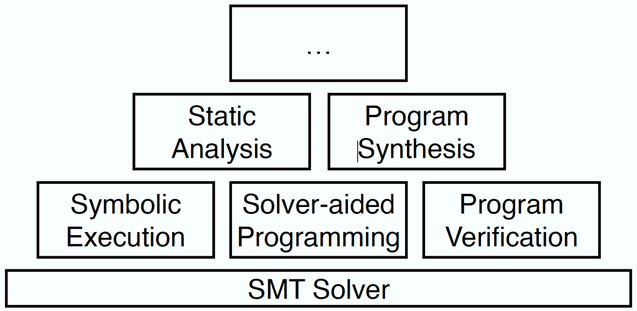
→

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat    ✘
```

https://github.com/Z3Prover/z3/issues/3973

# Summary

## SMT Solver




## Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

→

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat    ✘
```

https://github.com/Z3Prover/z3/issues/3973

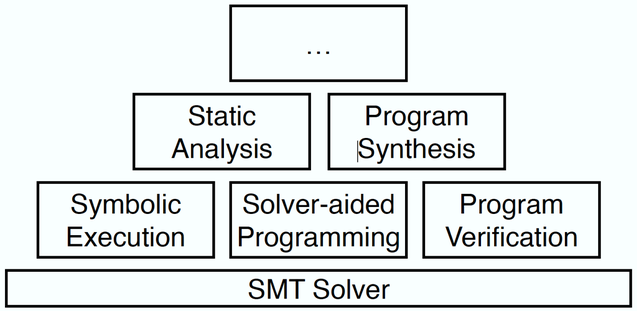
## Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 106 |
| Duplicate | 85 | 18 | 79 |

Bug status as of 30 Oct 2020

# Summary

## SMT Solver




## Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

→

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat   ✘
```

https://github.com/Z3Prover/z3/issues/3973

## Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 106 |
| Duplicate | 85 | 18 | 79 |

Bug status as of 30 Oct 2020

### Z3 Soundness Bug #2832

```
$ cat bug.smt2
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const c (_ BitVec 8))
(assert (= (bvxnor a b c)
           (bvxnor (bvxnor a b) c)))
(check-sat)

$ cvc4 bug.smt2
sat    ✔

$ z3 bug.smt2
unsat  ✘
```

https://github.com/Z3Prover/z3/issues/2832

### CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat    ✔

$ cvc4 bug.smt2
unsat  ✘
```

(str.indexof x y 1) < (str.len x)

Root Cause: Missed boundary condition

https://github.com/CVC4/CVC4/issues/3497

### Z3 Soundness Bug #2830

```
$ cat bug.smt2
(declare-fun a () Int)
(declare-fun b (Int) Bool)
(assert (b 0))
(push)
(assert (distinct true (= a 0) (not (b 0))))
(check-sat)

$ cvc4 bug.smt2
unsat  ✔

$ z3 bug.smt2
sat    ✘
```

Root Cause: Last argument omitted in distinct operator

https://github.com/Z3Prover/z3/issues/2830

### CVC4 Soundness Bug #4469

```
$ cat bug.smt2
(set-logic QF_AUFBVLIA)
(declare-fun a () Int)
(declare-fun b (Int) Int)
(assert (distinct (b a)
   (b (b a))))
(check-sat)

$ z3 bug.smt2
sat    ✔

$ cvc4 bug.smt2
unsat  ✘
```

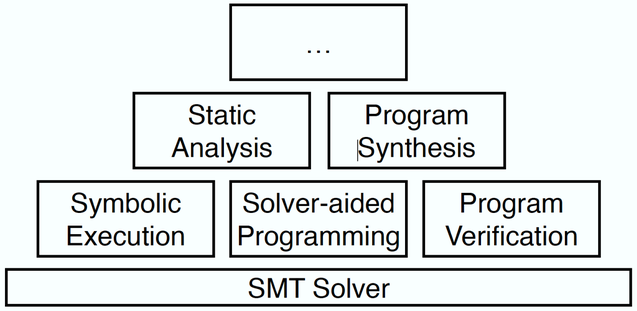
Labels: bug, major

https://github.com/CVC4/CVC4/issues/4469

# Summary

testsmt.github.io @testsmt

## SMT Solver




## Type-aware operator mutation

```
$ cat > formula.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (distinct (* 2 b) a))))
(check-sat)

$ cvc4 formula.smt2
sat

$ z3 formula.smt2
sat
```

→

```
$ cat > bug.smt2
(assert (forall ((a Int))
        (exist  ((b Int))
        (= (* 2 b) a))))
(check-sat)

$ cvc4 bug.smt2
unsat ✔

$ z3 bug.smt2
sat   ✘
```

https://github.com/Z3Prover/z3/issues/3973

## Bug Findings

| Status | Z3 | CVC4 | Total |
|---|---|---|---|
| Reported | 811 | 281 | 1,092 |
| Confirmed | 578 | 241 | 819 |
| Fixed | 521 | 164 | 106 |
| Duplicate | 85 | 18 | 79 |

Bug status as of 30 Oct 2020

### Z3 Soundness Bug #2832

```
$ cat bug.smt2
(declare-const a (_ BitVec 8))
(declare-const b (_ BitVec 8))
(declare-const c (_ BitVec 8))
(assert (= (bvxnor a b c)
           (bvxnor (bvxnor a b) c)))
(check-sat)

$ cvc4 bug.smt2
sat    ✔

$ z3 bug.smt2
unsat  ✘
```

https://github.com/Z3Prover/z3/issues/2832

### CVC4 Soundness Bug #3497

```
$ cat bug.smt2
(declare-fun x () String)
(declare-fun y () String)
(assert (= (str.indexof x y 1)
           (str.len x)))
(assert (str.contains x y))
(check-sat)

$ z3 bug.smt2
sat    ✔

$ cvc4 bug.smt2
unsat  ✘
```

(str.indexof x y 1) < (str.len x)

Root Cause: Missed boundary condition

https://github.com/CVC4/CVC4/issues/3497

### Z3 Soundness Bug #2830

```
$ cat bug.smt2
(declare-fun a () Int)
(declare-fun b (Int) Bool)
(assert (b 0))
(push)
(assert (distinct true (= a 0) (not (b 0))))
(check-sat)

$ cvc4 bug.smt2
unsat  ✔

$ z3 bug.smt2
sat    ✘
```

Root Cause: Last argument omitted in distinct operator

https://github.com/Z3Prover/z3/issues/2830

### CVC4 Soundness Bug #4469

```
$ cat bug.smt2
(set-logic QF_AUFBVLIA)
(declare-fun a () Int)
(declare-fun b (Int) Int)
(assert (distinct (b a)
   (b (b a))))
(check-sat)

$ z3 bug.smt2
sat    ✔

$ cvc4 bug.smt2
unsat  ✘
```

Labels: bug, major

https://github.com/CVC4/CVC4/issues/4469

# Yin-Yang Umbrella Release

OpFuzz + Semantic Fusion [PLDI ’20]

github.com/testsmt