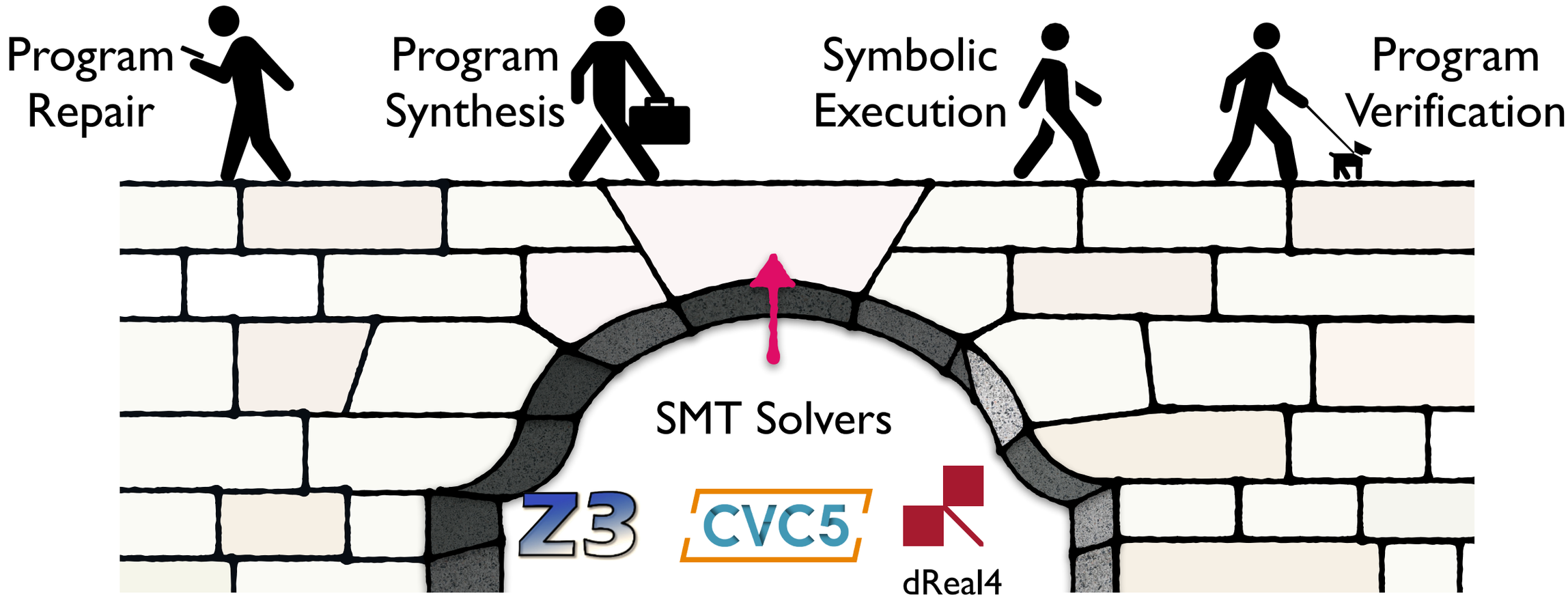
Semantic Metamorphic Testing for Finding Bugs in SMT Solvers

Jongwook Kim, Sunbeom So, and Hakjoo Oh
Korea University

June 13, 2025
Dagstuhl Seminar 25242

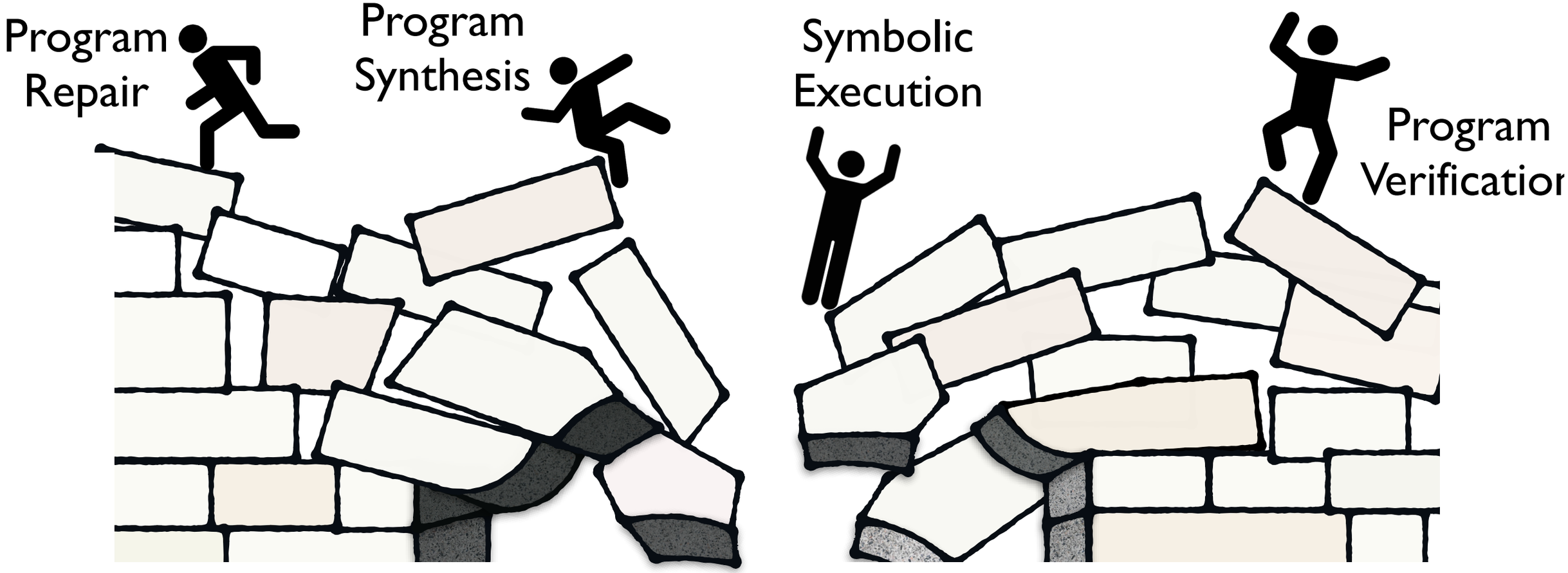
Correctness of SMT Solvers is Critical

- SMT solvers are the keystone of many SE applications
- Bugs in SMT solvers break the correctness of these SE tools



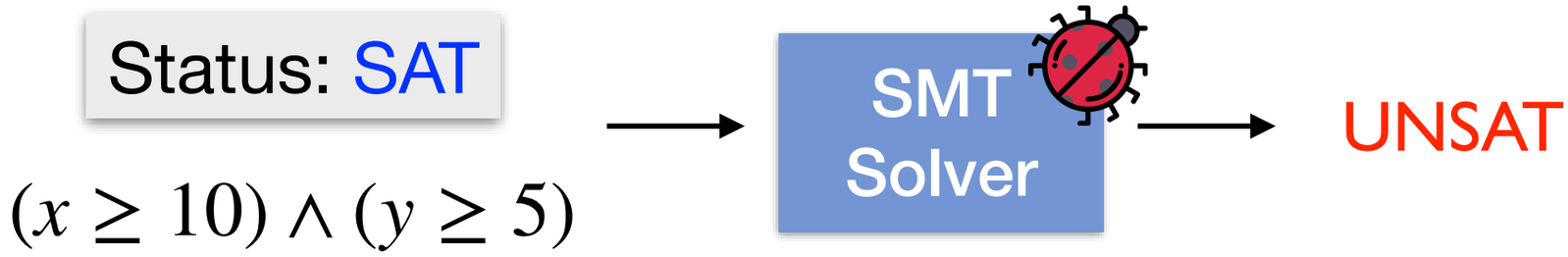
Correctness of SMT Solvers is Critical

- SMT solvers are the keystone of many SE applications
- Bugs in SMT solvers break the correctness of these SE tools

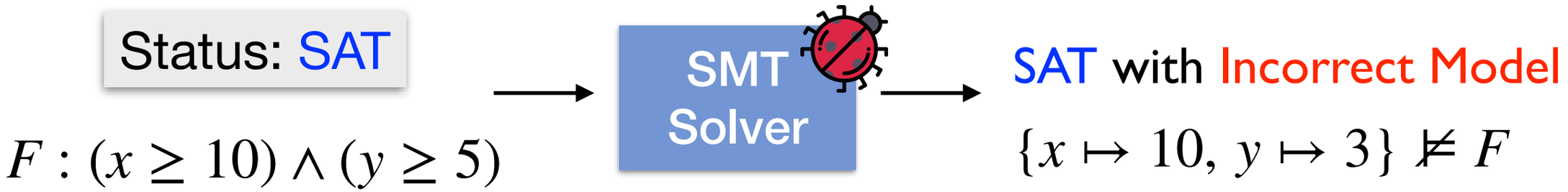


Goal: Finding Bugs in SMT Solvers

- (Refutational) Soundness bug

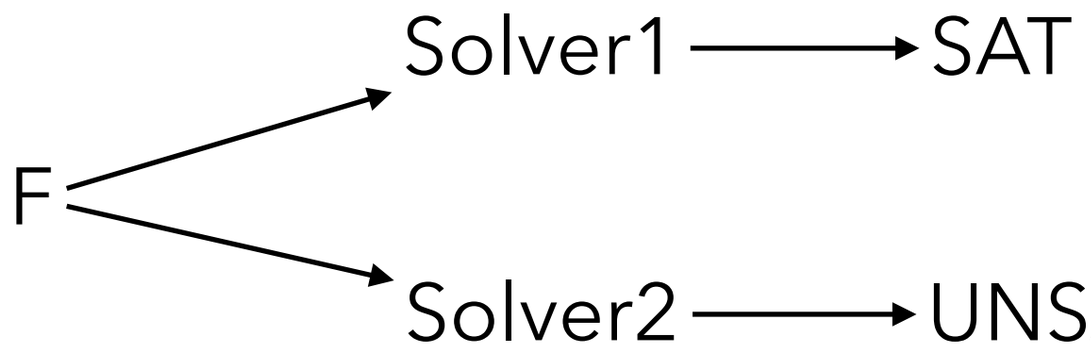


- Invalid model bug

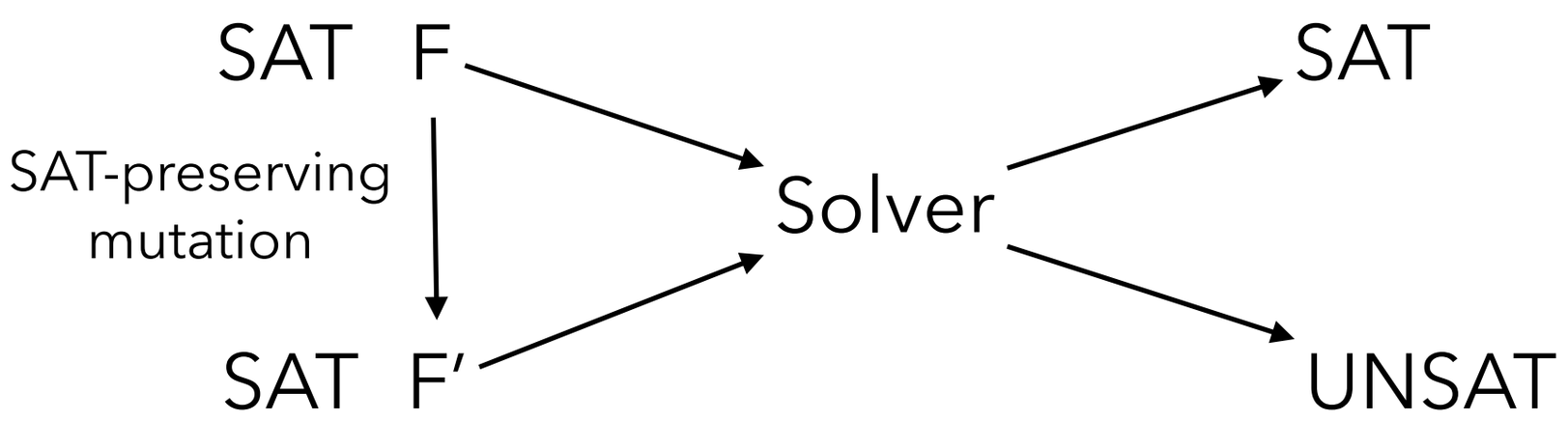


Existing Approaches

- Differential testing: TypeFuzz [OOPSLA'21], OpFuzz [OOPSLA'21]



- Metamorphic testing: Storm [FSE'20], Fusion [PLDI'20]

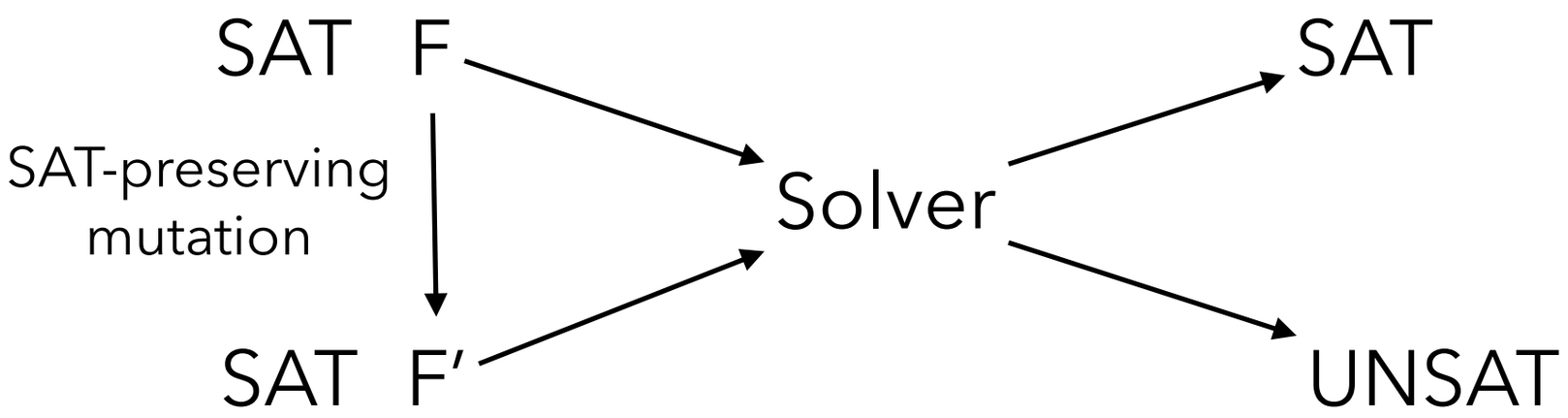


Existing Approaches

- Differential testing: TypeFuzz [OOPSLA'21], OpFuzz [OOPSLA'21]

  - Strength: Unrestricted random mutations → diverse test inputs
  - Weakness: Limited to testing features shared by multiple solvers

- Metamorphic testing: Storm [FSE'20], Fusion [PLDI'20]



Existing Approaches

- Differential testing: TypeFuzz [OOPSLA'21], OpFuzz [OOPSLA'21]

  - Strength: Unrestricted random mutations → diverse test inputs
  - Weakness: Limited to testing features shared by multiple solvers

- Metamorphic testing: Storm [FSE'20], Fusion [PLDI'20]

  - Strength: Can be applied when multiple solvers are unavailable
  - Weakness: Restricted, SAT-preserving mutations (e.g., x → x+0)

Existing Approaches

- Differential testing: TypeFuzz [OOPSLA'21], OpFuzz [OOPSLA'21]

  - Strength: Unrestricted random mutations → diverse test inputs
  - Weakness: Limited to testing features shared by multiple solvers

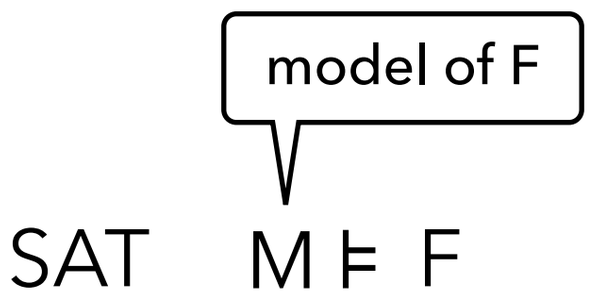
- Metamorphic testing: Storm [FSE'20], Fusion [PLDI'20]

  - Strength: Can be applied when multiple solvers are unavailable
  - Weakness: Restricted, SAT-preserving mutations (e.g., x → x+0)

Our approach aims to combine both strengths

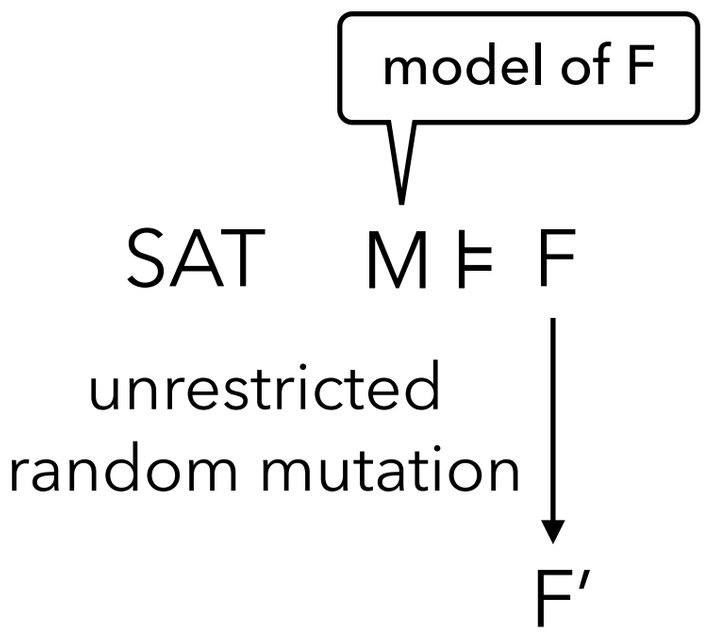
Our Approach

- “Semantic” model-based metamorphic testing without syntactic mutation rules



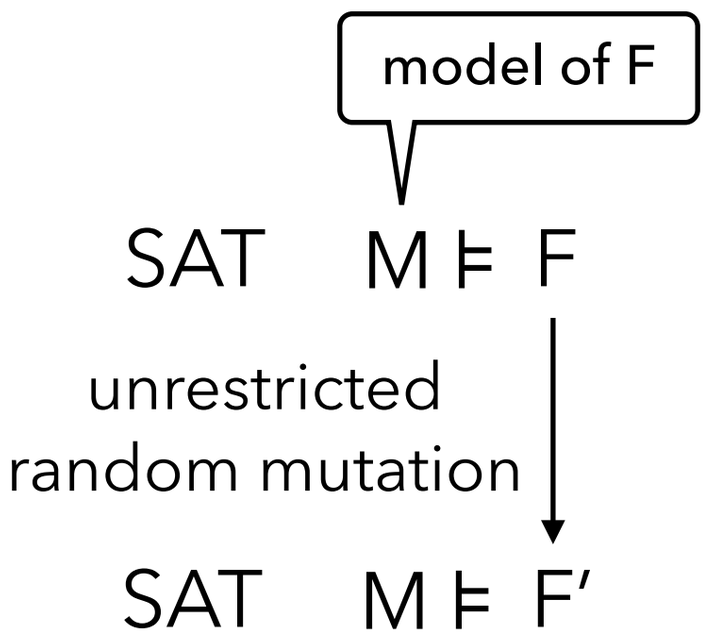
Our Approach

- “Semantic” model-based metamorphic testing without syntactic mutation rules



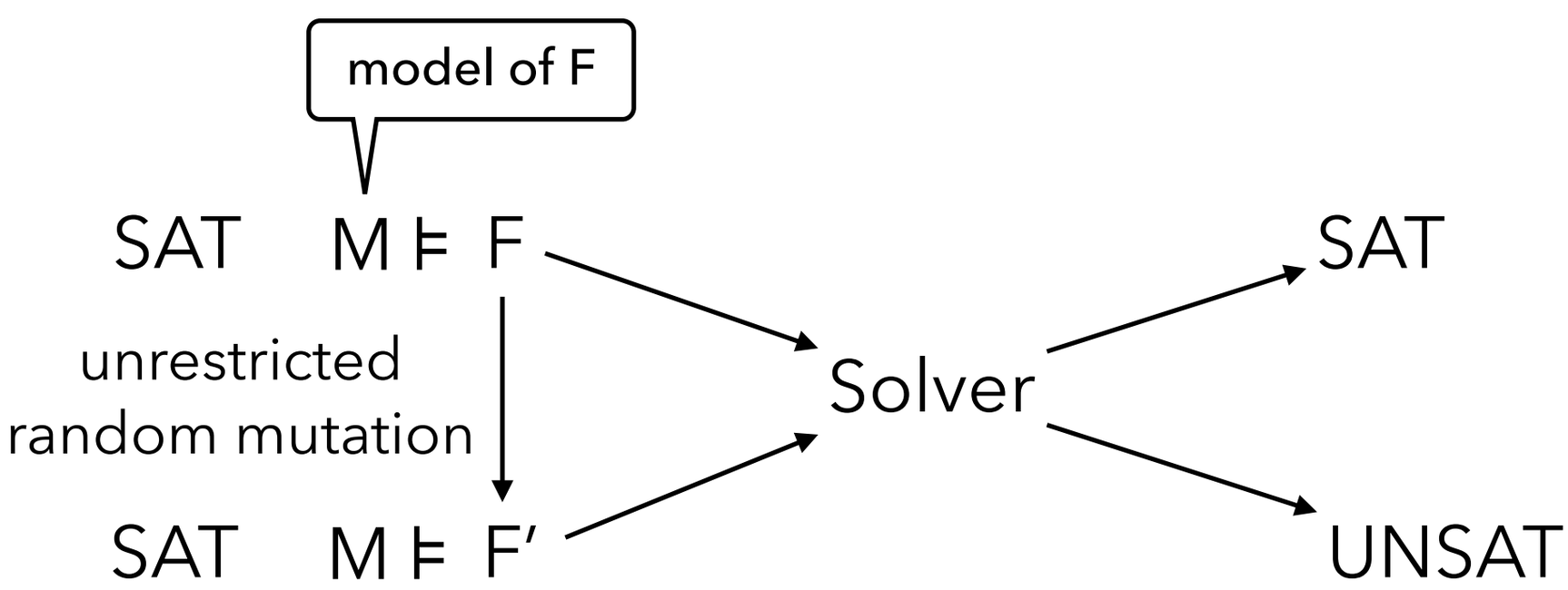
Our Approach

- “Semantic” model-based metamorphic testing without syntactic mutation rules



Our Approach

- “Semantic” model-based metamorphic testing without syntactic mutation rules



Example: A Soundness Bug in CVC5

- The original and mutated formulas are satisfiable
- CVC5 reports the mutant as unsatisfiable

```
1  (set-logic QF_SLIA)
2  (declare-fun t () String)
3  (assert (str.prefixof "-" (str.substr t 0 1)))
4  (assert (> (str.len (str.substr t 0 2)) 1))
5  (-)(assert (not (= (- 1) (str.to_int (str.substr t 1 1)))))
5' (+)(assert (not (xor (str.< (str.update "-0" 0 t) "-0") false)))
6  (-)(assert (>= (+ 0 2) (str.len t)))
6' (+)(assert (str.suffixof (str.replace t "-0" "-") "-"))
7 (check-sat)
```

Example: A Soundness Bug in CVC5

- The original and mutated formulas are satisfiable
- CVC5 reports the mutant as unsatisfiable

```
1  (set-logic QF_SLIA)
2  (declare-fun t () String)
3  (assert (str.prefixof "-" (str.substr t 0 1)))
4  (assert (> (str.len (str.substr t 0 2)) 1))
5  (-)(assert (not (= (- 1) (str.to_int (str.substr t 1 1)))))
5' (+)(assert (not (xor (str.< (str.update "-0" 0 t) "-0") false)))
6  (-)(assert (>= (+ 0 2) (str.len t)))
6' (+)(assert (str.suffixof (str.replace t "-0" "-") "-"))
7 (check-sat)
```

These mutations are beyond the reach of existing metamorphic testing

Example: A Soundness Bug in CVC5

- The original and mutated formulas are satisfiable
- CVC5 reports the mutant as unsatisfiable

```
1  (set-logic QF_SLIA)
2  (declare-fun t () String)
3  (assert (str.prefixof "-" (str.substr t 0 1)))
4  (assert (> (str.len (str.substr t 0 2)) 1))
5  (-)(assert (not (= (- 1) (str.to_int (str.substr t 1 1)))))
5' (+)(assert (not (xor (str.< (str.update "-0" 0 t) "-0") false)))
6  (-)(assert (>= (+ 0 2) (str.len t)))
6' (+)(assert (str.suffixof (str.replace t "-0" "-") "-"))
7 (check-sat)
```

```
$ cvc5 mutant.smt2
unsat
```

```
$ z3 mutant.smt2
Error:Unsupported Function
```

Differential testing is not applicable

Effectiveness

- Found 25 new bugs in Z3, CVC5, and dReal

Bug Type	Z3	CVC5	dReal	Total
Soundness	6	4	2	12
Invalid-Model	2	7	0	9
Crash	0	4	0	4

- Most of these bugs were hard to detect by existing methods
  - Seven involved solver-specific features
  - Most bug-triggering mutants were substantially different from the original formulas

Summary

- Proposed a domain-specific, model-based metamorphic testing technique for SMT solvers

- For more information, see our paper “Diver: Oracle-Guided SMT Solver Testing with Unrestricted Random Mutations. ICSE 2023”

  - In particular, purely random mutations hardly succeed to satisfy the original seed’s model

  - In the paper, we proposed a method for weighted mutation to increase this probability

- Curious whether such “semantic” metamorphic testing can be applied to other domains

Thank you!