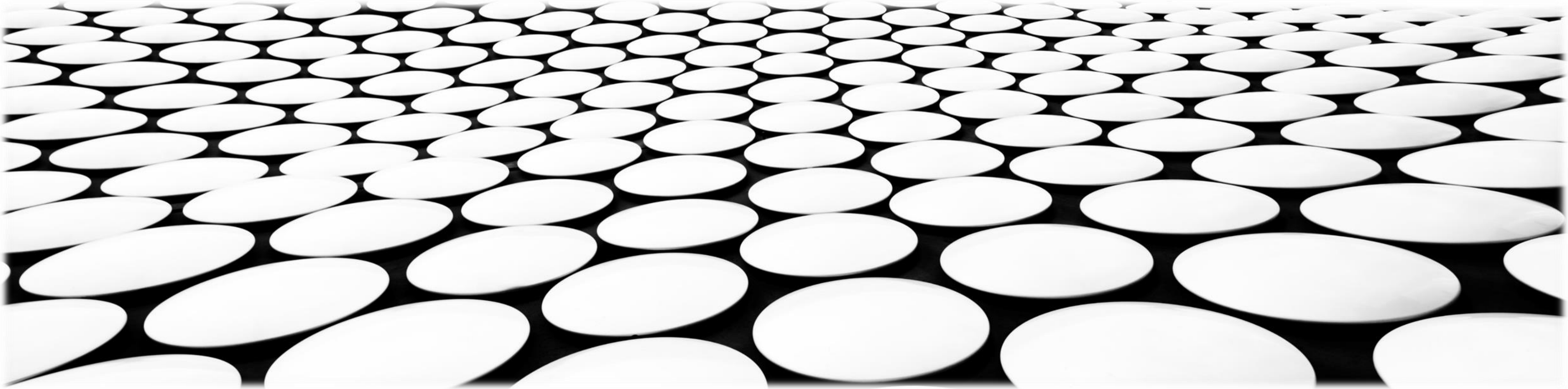
FUZZING TVM RELAY

STEVEN LYUBOMIRSKY, MICHAEL FLANDERS, AND EDWARD MISBACK



PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

PLSE

sampl

Edward Misback
misback@cs.washington.edu

Michael Flanders
mkf727@cs.washington.edu

WHY DON'T WE HAVE MORE TVM TESTS?

	Lines of code (KSLOC)
Implementation (tvm/src, tvm/python)	233
Tests (tvm/tests)	141

- Test cases are program fragments
  - Tedious to write by hand
  - Complex interactions between features
  - Shapes need to match up
- Fuzzing could help

Measurements courtesy of David A. Wheeler's SLOCCount tool for Linux

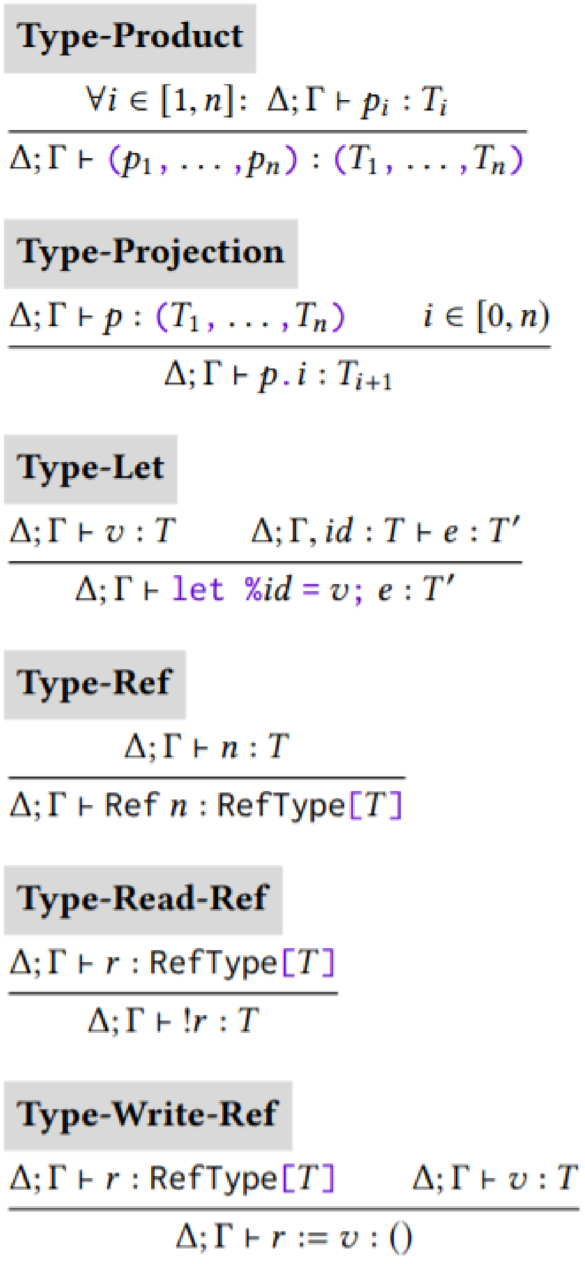
RELAY FUZZING APPROACH

- How do we generate Relay programs we know are valid?
- Use typing information: Given a type, generates expression fulfilling it
- We have a prototype! Only ~2000 lines of Python
- Supports most statically typed Relay constructs, ~20 operators

BASIC CASES IN FUZZING RELAY

- Most of Relay’s type system plays nice
- Set goal type and work backwards
  - All types have a literal for a base case*
  - Connectives (let bindings, etc.) combine existing terms
- Ensuring termination: Fall back to a literal!

*This can get tricky with arbitrary ADTs.



Type-Product

$$\frac{\forall i \in [1, n]:\ \Delta;\Gamma \vdash p_i : T_i}{\Delta;\Gamma \vdash (p_1, \ldots, p_n) : (T_1, \ldots, T_n)}$$

Type-Projection

$$\frac{\Delta;\Gamma \vdash p : (T_1, \ldots, T_n) \qquad i \in [0, n)}{\Delta;\Gamma \vdash p.i : T_{i+1}}$$

Type-Let

$$\frac{\Delta;\Gamma \vdash v : T \qquad \Delta;\Gamma, id : T \vdash e : T'}{\Delta;\Gamma \vdash \texttt{let}\ \%id = v;\ e : T'}$$

Type-Ref

$$\frac{\Delta;\Gamma \vdash n : T}{\Delta;\Gamma \vdash \mathsf{Ref}\ n : \mathsf{RefType}[T]}$$

Type-Read-Ref

$$\frac{\Delta;\Gamma \vdash r : \mathsf{RefType}[T]}{\Delta;\Gamma \vdash !r : T}$$

Type-Write-Ref

$$\frac{\Delta;\Gamma \vdash r : \mathsf{RefType}[T] \qquad \Delta;\Gamma \vdash v : T}{\Delta;\Gamma \vdash r := v : ()}$$

THE TOUGH PART: SOLVING TYPE RELATIONS

- Type system includes constraints on tensor shapes!
- Argument types (shapes) affect result type (shape)
- Every single op has a relation!
- Hardest part: Implemented *imperatively* in C++

DEALING WITH TYPE RELATIONS: SOLVER-BASED APPROACH

- Encode type relations in a solver domain (e.g., ILP)
- Given return type, use solver to generate valid argument types
- Pro: Only one solver query at a time, easily composable
- Cons:
  - The solver is a dependency
  - Need to formalize the type relations in the solver domain

DEALING WITH TYPE RELATIONS: STOCHASTIC APPROACH

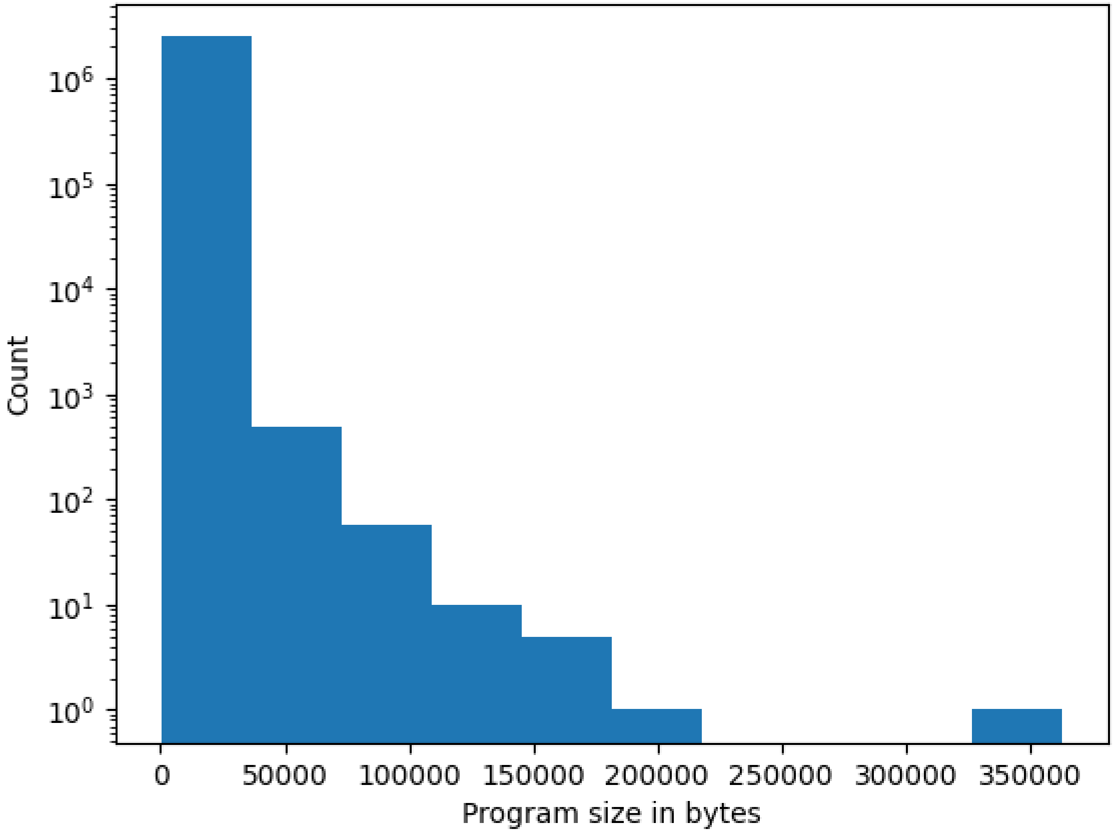
- Sample possible inputs, check which solutions work, keep a cache
- Use argument type–return type pairs to guide type generation
- Pro: Can reuse existing type relation implementations, no solver
- Con: Not as flexible as solver-based approach

BUGS FOUND

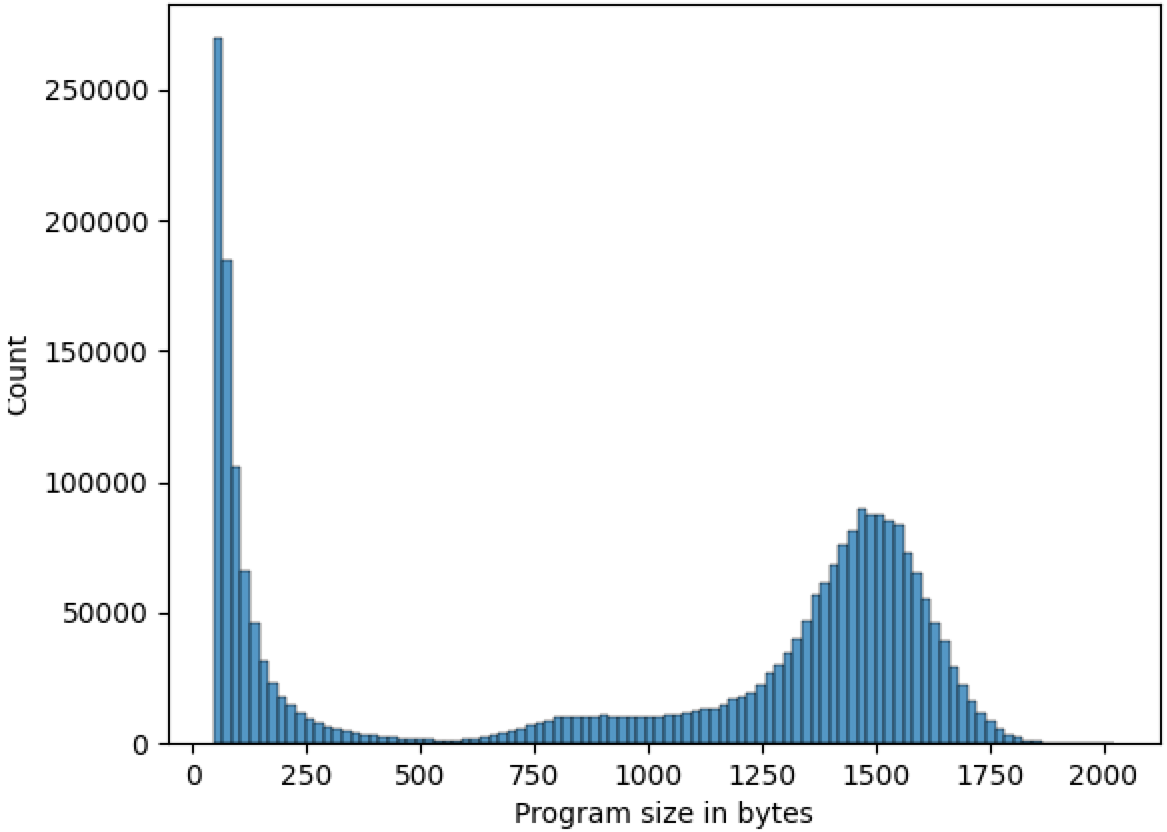
- Match exhaustion bug:
  - Found by fuzzer very quickly in small-scale test runs
  - Fix merged https://github.com/apache/tvm/pull/7459
- Missing bounds check in bias add specification:
  - Found manually while formalizing the type relation
  - Fix merged https://github.com/apache/tvm/pull/7554
- Also found a bug parsing refs of refs (fix not yet PR’d)

GENERATED PROGRAM SIZES

Type-Directed

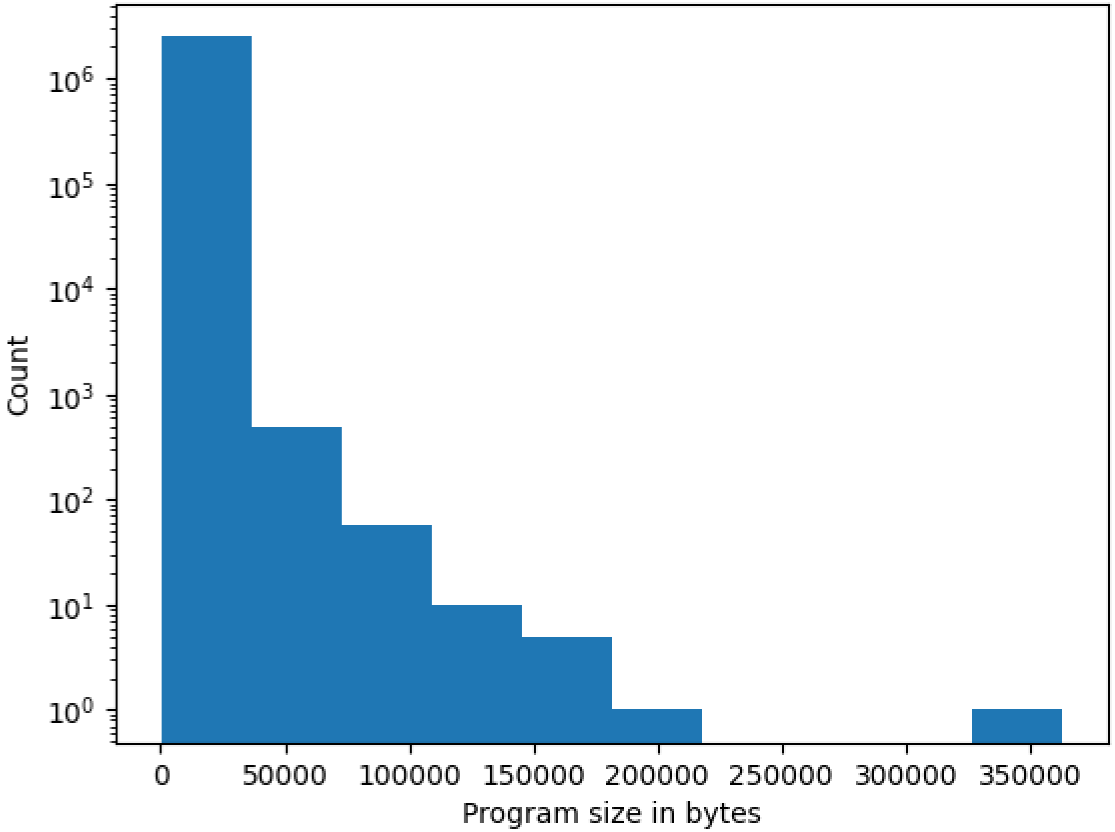


Grammar-Based

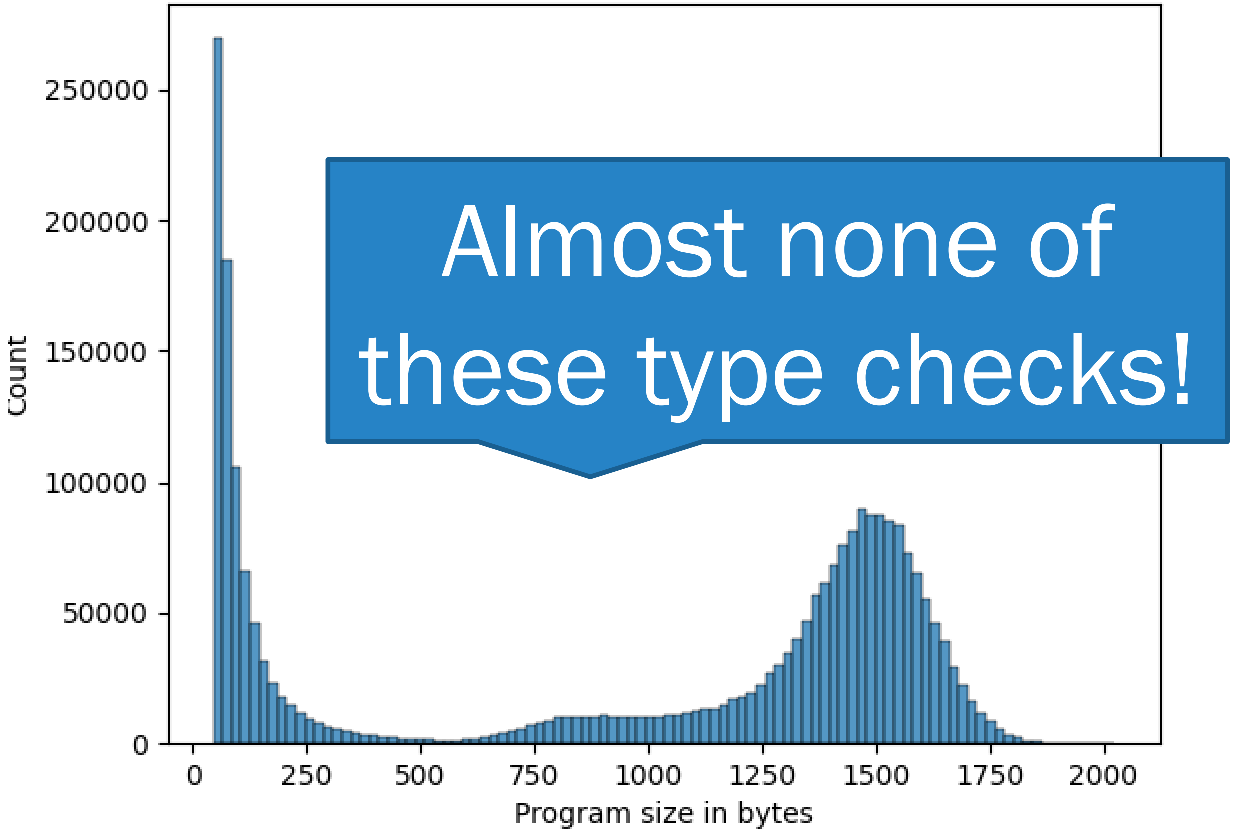


GENERATED PROGRAM SIZES

Type-Directed



Grammar-Based

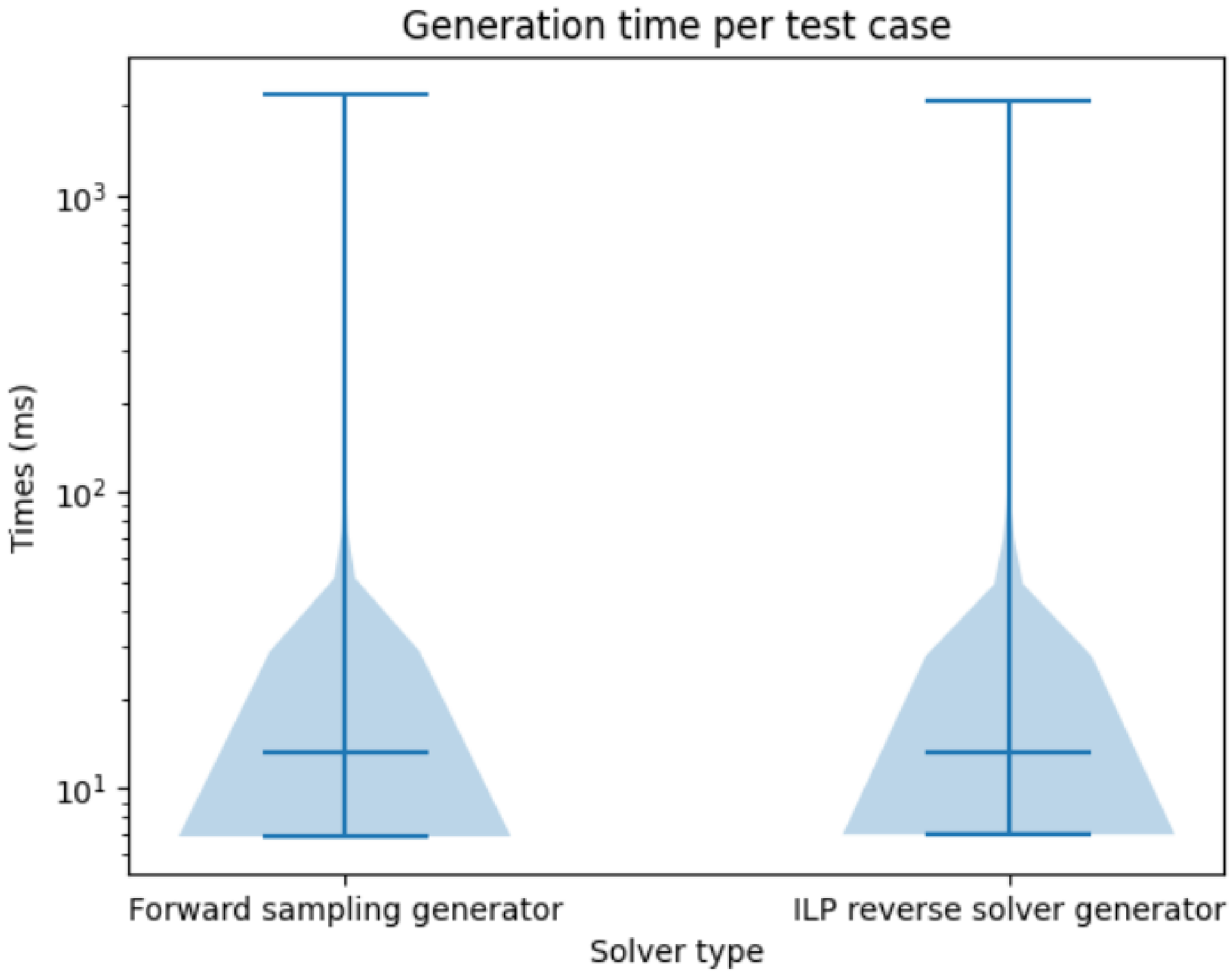


Almost none of these type checks!

(SOME OF THE TRIVIAL CASES THAT DID TYPE CHECK)

```
def @main() ->   () {   match? 75 {}}
def @main() ->   () {()}
def @main() ->   () {   match? (((),)) {}}
def @main() ->   () {  %793 =     8 ; ()}
def @main() ->   uint16 {   match? () {}}
```

GENERATION SPEED



THE FUTURE OF THE FUZZER

- Prototype available at: https://github.com/slyubomirsky/relay_fuzzer
- Will create a TVM RFC for discussing the future of fuzzing
- Questions for the future:
  - How can we support dynamic or parametric shapes?
  - What testing oracles make the most sense to use?
  - How should we express constraints on generated programs?
  - What about mutating and minimizing Relay programs for bug reports?

THANK YOU!