QuickCheck to Check the Type Check

Casper Bach Poulsen

Sára Juhošová
s.juhosova@student.tudelft.nl

Cas van der Rest

1 Research Question

How effective is property-based testing (PBT) on type checkers for definitional interpreters?

- How frequently can PBT catch a bug in a type checker?
- How quickly can PBT find a counterexample to a buggy implementation of a type checker?

2 Languages

```
data Type = TInt
  | TBool
  | TFun Type Type

data Expr = Id String
  | Lambda (String, Type) Expr
  | App Expr Expr
```

↑ Simply-Typed Lambda Calculus
↓ Programming Computable Functions

```
data Type = TInt | TBool
  | TList Type
  | TFun Type Type

data Expr = TrueE | FalseE
  | And Expr Expr | Or Expr Expr
  | Not Expr
  | NumE Int
  | Add Expr Expr | Mul Expr Expr
  | Eq Expr Expr | Lt Expr Expr
  | Nil Type | Cons Expr Expr
  | Head Expr | Tail Expr
  | IsNil Expr | IsList Expr
  | Id String
  | Lam String (String, Type) Type Expr
  | App Expr Expr
  | Let (String, Expr) Expr
  | If Expr Expr Expr
```

3 Type Checker

```
type TEnvironment = [(String, Type)]

typeOf :: Expr -> TEnvironment
            -> Either Error Type
typeOf = ...
```

4 Properties

(GEN) `typeOfBuggy (typed t) nv == t`

(CMP) `typeOfBuggy e nv == typeOf e nv`

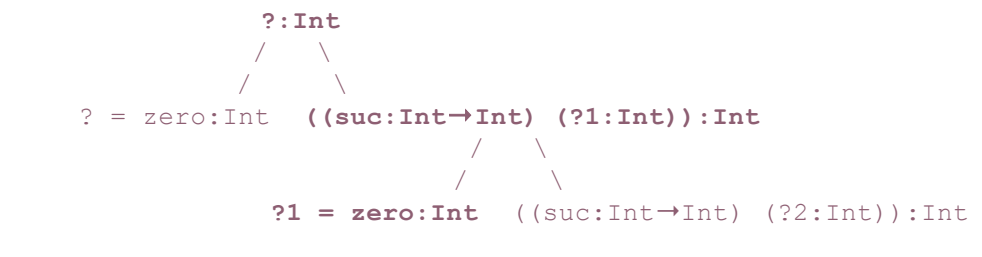
(APP)
```
if typeOfBuggy lam nv == TFun p b
   and typeOfBuggy arg nv == p then
typeOfBuggy (App lam arg) nv == b
```

↑ well-typed input
↓ ill-typed input

(ILL) `typeOfBuggy e nv == typeOf e nv`

5 Generators

1. A generator for well-typed expressions using a build-up approach[1]:



?:Int = **(suc zero):Int** where *suc:(Int→Int)* and *zero:Int*

2. A naïve generator for ill-typed expressions using fully arbitrary generation.

6 Results

The Error-Catching Frequency (in %):

	missing case match	incorrect binding	incorrect type returned	no check for sub-expression
GEN	100-100	60-99	100-100	0-0
CMP	100-100	50-96	100-100	0-0
APP	100-100	100-100	100-100	0-0
ILL	100-100	0-0	12-95	100-100

↑ Simply-Typed Lambda Calculus
↓ Programming Computable Functions

	missing case match	incorrect binding	incorrect type returned	no check for sub-expression
GEN	100-100	100-100	100-100	0-0
CMP	100-100	100-100	100-100	0-0
APP	100-100	100-100	100-100	0-0
ILL	100-100	0-0	0-0	4-100

[1]Michał H. Pałka, Koen Claessen, Alejandro Russo, and John Hughes. 2011. Testing an optimising compiler by generating random lambda terms. In *Proceedings of the 6th International Workshop on Automation of Software Test (AST '11)*. Association for Computing Machinery, New York, NY, USA, 91-97. DOI:https://doi.org/10.1145/1982595.1982615