AUTOMATED VALIDATION OF DEFINITIONAL INTERPRETERS

Validating Definitional Interpreters Using Property-Based Testing

PROBLEM

```
data Expr = Num Int | Add Expr Expr
eval :: Expr -> Int
eval (Num i) = i
eval (Add e1 e2) = interp e1 + interp e2

evil :: Expr -> Int
evil (Num i) = i
evil (Add e1 e2) = interp e1 + interp e1
```

We want to detect that **eval** and **evil** are **NOT** equivalent with:

- A uniform generator
- QuickCheck
- SmallCheck

```
QuickCheck:
*** Failed! Falsified (after 5 tests):
2 + 2 + 4 + (-4) + 0 + (-1) + 1 + 1
```

```
SmallCheck:
Failed test no. 4.
there exists 1 + 0 such that
  condition is false
```

```
Uniform generation:
*** Failed! Falsified (after 1 test):
0 + 1 + 0
```

Reported counter-examples for each generation method

METHOD

1.Define Algebraic Data Types (ADTs)

Arithmetic ADT (Add, Sub, Mul, Div)
Boolean ADT (And, Or, Not)
Conditional ADT (Arithmetic, boolean, conditional & lambda operations)

2.Write Test Data Generators

Write QuickCheck generators
Write SmallCheck generators
Define spaces for the ADTs and write the respective uniform generators

3.Check Interpreter Equivalence

Use each generation method to check the following property:

```
prop_interp :: Expr -> Bool
prop_interp e = interp1 e == interp2 e
```

RESULTS

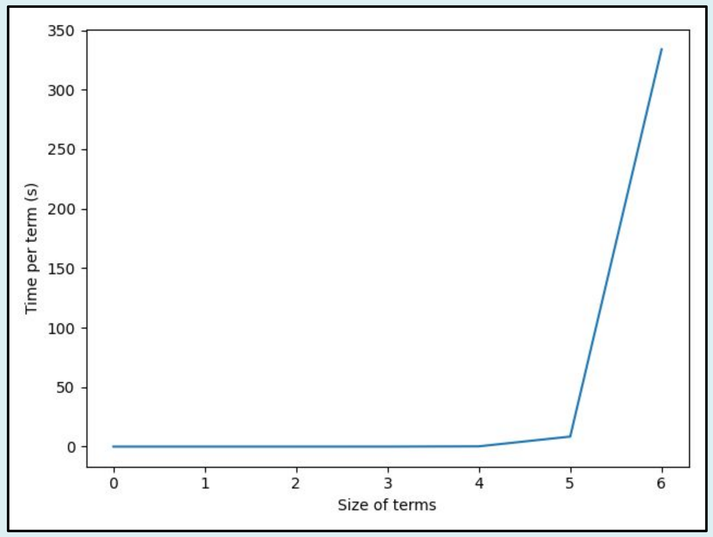
AVG number of tests to detect faults in each pattern match case

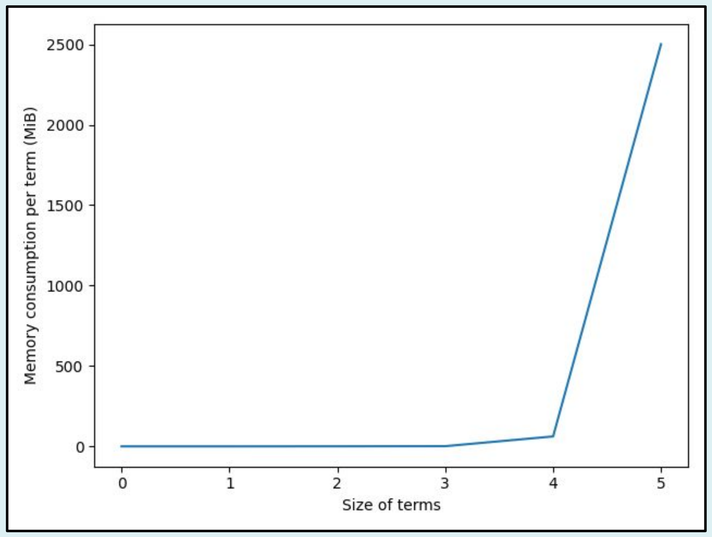
Method	Add	Sub	Mul	Div
QC	3	3	4	3
SC	12	19	33	37
UG	4	5	6	3

AVG = Average number of tests out of 10 runs of 100 tests

Method	And	Or	Not
QC	5	7	5
SC	9	16	9
UG	3	2	2

QC = **QuickChek**,
SC = **SmallCheck**,
UG = **Uniform generation**





Time & **Space** consumption for the conditional ADT (**UG Method**)

CONCLUSION

The **uniform generation** method outperforms the **QuickCheck** & **SmallCheck** naive generators when it comes to well-typed data.

Time needed for the uniform generator grows **exponentially** as size of terms increases. To that end, an improvement is needed.

While fast, **QuickCheck** & **SmallCheck** require more complex generation methods to work on a wide variety of grammars.

TU Delft Delft University of Technology

Alexandru Radu Moraru
CSE3000 02/07/2021

Casper Bach Poulsen
Cas van der Rest