Exploring Program Equivalence as a Means of Comparing Definitional Interpreters

Research Project by
Ruben Backx

Responsible Professor & Supervisor
Casper Bach Poulsen

Supervisor
Cas van der Rest

1 Research Question

How effective is program equivalence for comparing definitional interpreters?

Two programs can be compared using an approach introduced by Clune et al. It has been tested on simple student submissions, but not on complex interpreters students write in a course like Concepts of Programming Languages. The ability to use program equivalence to verify student-written interpreters would reduce the need to manually write tests.

See right an example of two definitional interpreters.

```
data Expr = Val Float | Add Expr Expr | Sub Expr Expr
          | Mul Expr Expr | Div Expr Expr
```

```
interp :: Expr -> Float
interp (Val x) = x
interp (Add left right) =
    interp left + interp right
interp (Sub left right) =
    interp left - interp right
interp (Mul left right) =
    interp left * interp right
interp (Div left right) =
    interp left / interp right
```

```
interp :: Expr -> Float
interp (Val x) = x
interp (Add left right) =
    interp left + interp left
interp (Sub left right) =
    interp left - interp right
interp (Mul left right) =
    interp left * interp right
interp (Div left right) =
    interp left / interp right
```

Figure 1: An example of two definitional interpreters.

2 Method

Interpreters can be transformed to a logical formula that will be satisfiable only if the programs are equivalent. This approach is sound, i.e. two non-equivalent interpreters are never recognised as equivalent.

3 Contributions

- Support for strings and lists added
- Extra rules added for better equivalence checking between interpreters
- Pre- and post-processing added to improve equivalence checking

$$\frac{\Gamma \vdash e_l \overset{\sigma_1}{\Longleftrightarrow} e_l' : \tau_1 \dashv \Gamma_1 \quad \Gamma \vdash e_r \overset{\sigma_2}{\Longleftrightarrow} e_r' : \tau_2 \dashv \Gamma_2 \quad \Gamma \vdash e_l \overset{\sigma_3}{\Longleftrightarrow} e_r' : \tau_3 \dashv \Gamma_3 \quad \Gamma \vdash e_r \overset{\sigma_4}{\Longleftrightarrow} e_l' : \tau_4 \dashv \Gamma_4}{\Gamma \vdash e_l * e_r \overset{(\sigma_1 \wedge \sigma_2) \vee (\sigma_3 \wedge \sigma_4)}{\longleftrightarrow} e_l' * e_r' : \tau \dashv \Gamma_1 \Gamma_2 \Gamma_3 \Gamma_4}$$

Figure 2: One of the new rules. Specifically, this provides better results when interpreters have the left and right terms of a multiplication swapped.

4 Conclusion

Program equivalence is able to recognise about half of the tested modifications as equivalent. Because of its soundness it is suitable for verifying student submissions, but its effectiveness is yet to be tested on real student submissions.

In combination with other forms of grading, program equivalence can be also be used to give students feedback in batches and to build up a collection of common mistakes.

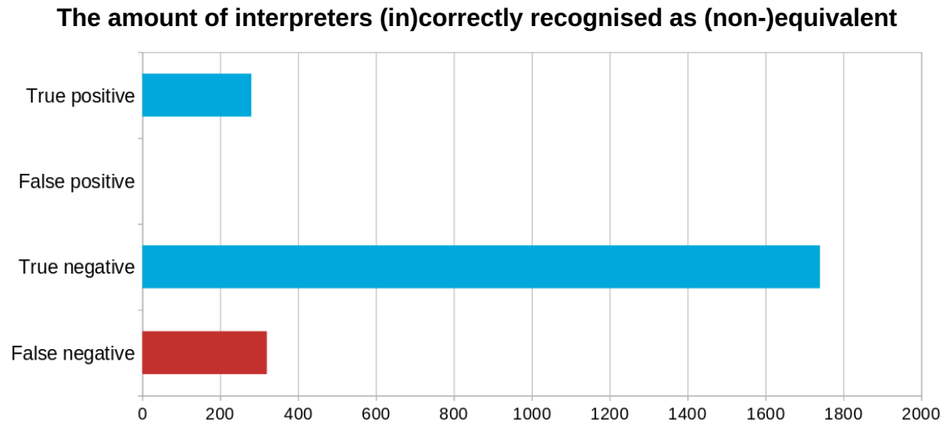

Figure 3: Experimental results for 3 sets of 40 interpreters.

r.w.backx@student.tudelft.nl