Practical performance enhancements to the evaluation model of the Hazel programming environment

Jonathan Lam[1] Prof. Fred Fontaine, Advisor[1]
Prof. Robert Marano, Co-advisor[1] Prof. Cyrus Omar[2]

[1]Electrical Engineering
The Cooper Union for the Advancement of Science and Art

[2]Electrical Engineering and Computer Science
Future of Programming Lab (FPLab), University of Michigan

2022/04/29

Table of Contents

A programming language is a specification

[TODO: semantics]
[TODO: syntax]

A brief primer on the λ-calculus

Table of Contents

The Hazel programming environment

Hazelnut: A bidirectionally-typed static semantics

Hazelnut Live: A bidirectionally-typed dynamic semantics

Table of Contents

Evaluation using environments vs. substitution

Updated evaluation rules

Handling recursion

Matching the result from evaluation using substitution

Memoizing by environments for substitution and equality checking

Generalized closures

Table of Contents

Motivating example

Hole instances vs. hole closures/instantiations

Hole instance parent vs. hole closure parents

The hole numbering algorithm

A unified postprocessing algorithm

Table of Contents

Motivating example

The FAR process

1-step vs. *n*-step FAR

Detecting a valid fill operation

The fill operation

The resume operation

The postprocessing operation

Table of Contents

Evaluation with environments

Hole numbering motivating example

FAR motivating example

Table of Contents

Generalized closures

Unique hole closures

FAR as a generalization of evaluation

Table of Contents

Completion of n-step FAR

Generalized memoization

Formal evaluation of metatheory

Conclusions

References

[TODO: include metatheory]