AUTOBAHN : Using Genetic Algorithms to Infer Strictness Annotations

Yisu Remy Wang, Diogenes Nunez, Kathleen Fisher

Tufts University, Medford MA, USA

September 1, 2016

Overview

- Intro.
- The Problem: Adding Strictness Annotations
- Background: Strictness Annotations & Genetic Algorithms
- The Algorithm
- Soundness
- Evaluation & Case Study
- Related Work

Once Upon a Time, 2 Eager Programmers with a Lazy Program . . .

- Example code & explanation here
- Annotated code & explanation, explain laziness / bangs informally in passing
- ?? What kind of example do we want?

The Problem

- Too much laziness **slows down** programs
  - Runtime, allocation, GC work ...
- **Difficult to add** strictness annotations
  - Add **helpful** annotations, guarantee **soundness**

Background: Laziness & Strictness Annotations

- What's a thunk, what's WHNF
- Different kinds of annotations and what do they do (mention StrictHaskell)

Background: Genetic Algorithms

- Introduce GA

Background: GA for Strictness Annotations

- Why is it good?
- Avoid local optima
- Works like a desperate Haskeller trying all kinds of different bangs
- Works great if bangs work with each other in a simple way (corpus analysis)

The Algorithm: Representation

- Genes & chromosomes
- Fitness Functions
- Parameters
- 1st Generation
- New Generations
- Determining a Winner
- Pulling it All Together
- Discussion

The Algorithm: Optimization

- Parameters
- 1st Generation
- New Generations
- Determining a Winner

The Algorithm: Pulling it All Together

The Algorithm: Discussion

Soundness

Evaluation

- Introduce benchmarks / setup

Evaluation: nofib benchmarks

Evaluation: strict Haskell

- ?? Did we ever try StrictHaskell on aeson/gcSim?

Evaluation - Case Study: gcSimulator

Evaluation - Case Study: Aeson

Evaluation: 10-fold Cross-validation

Evaluation: Autobahn Performance

Related Work

- static analysis
- including dynamic information
- other approaches

Future Work

Conclusion

Acknowledgments