Project Proposal

Concepts of Program Design

Joris ten Tusscher, Joris Burgers, Ivo-Gabe de Wolff, Cas van der Rest, Orestis Melkonian

1 Background

- Reversible Computation

- other languages

- a bit of theory

2 Problem

2.1 Experiment

2.2 Janus

We want to implement the reversible programming language Janus as an DSL in Haskell. The syntax and semantics given by Lutz en Derby [**?**] will be used. There will however be a few changes. There will be no input and output in the DSL. Therefore, the READ and WRITE statements will not be valid statements. Besides this limitation, the DSL will allow arbitrary Haskell expressions to the right-hand side of Janus statements, enabling programmers to reuse Haskell code. This freedom of accepting Haskell code will not limit the reversibility of the DSL, since the semantics for Janus require statements to be reversible, but not right-hand expressions. For instance, the statement $x+ = f$ where f is any valid Haskell expression. As long as f does not depend on x, the statement can be inverted by the statement $x- = f$.

2.3 Extensions

- Data structures (trees/sets/maps/ADTs)

- Prelude

- Syntactic sugar

2.4   Applications

- Path finding algorithms

- Encoding-Decoding

- Low-level bit manipulation (hamming error correction)

- [Optional] Reversible Debugger

2.5   Benchmarking

- Janus vs rFun

- LOC metrics

2.6   Formal verification

- Reversibility

- Verifying pre/post-conditions

- r-Turing Completeness

3   Methodology

- Template Haskell [1]

  - Embedding Janus
  - Compile-time guarantees
    * Type-checking
    * Variable usage

- GHC Profiling

  - Criterion package [2]

- Liquid Haskell [3]

  - Theorem proving

---

[1] https://wiki.haskell.org/Template_Haskell
[2] http://hackage.haskell.org/package/criterion)
[3] https://ucsd-progsys.github.io/liquidhaskell-blog/)

4 Planning

- Milestones

- Division of labour