Project Report

Concepts of Program Design

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

1 Problem

1.1 Experiment

1.2 Janus - Reversible Computation

2 Methodology

2.1 Expected outcome

2.2 Planning

3 Results

3.1 Achievements

3.2 Goal/planning adjustments

4 Reflection

4.1 Good/bad surprises

4.2 Problems along the way

5 Future work

The Hanus project can be extended or improved in multiple ways. Some ideas for future work on Hanus are:

1. Implementing proper field and array indexers, i.e. at the moment, multiple concatenated field and array indexers are not supported.

2. Better error messages that don't look like Haskell errors. This is not a bug per se, but when a programmer writes complicated Hanus code, they will quickly find out that the error messages are very difficult to debug, partially because the error messages concern the generated Haskell code, but more so because of the fact that

the structure of certain Hanus expressions such as loops is rigorously changed in the compilation step to Haskell, making it very hard for the programmer to find out what is wrong in the actual Hanus program.

3. Template Haskell bug fixes. During the development of Hanus, multiple Template Haskell bugs have been found that are most probably caused by template Haskell, e.g. function declarations that Haskell cannot find when they are declared at the top level, but that Haskell *can* find when they are defined in a were clause of the callee, or variables that should get a unique name from TemlateHaskell, but still cause “multiple declarations of x” errors in Haskell.

4. Finding and fixing parser loop bugs. Although the parser has been tested rigorously, there are rare cases where the parser gets stuck in an endless loop. These bugs are very hard to find, so the chance that they were *all* caught thanks to the many parser tests is still slim.

6 Appendix

6.1 Repo link

6.2 Code navigation