FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Review report of a final thesis

Reviewer:	doc. Ing. Filip Křikava, Ph.D.
Student:	Bc. Jan Sliacký
Thesis title:	Implementation of a statically typed, lazy, pure functional programming language
Branch / specialization:	System Programming
Created on:	22 August 2022

Evaluation criteria

1. Fulfillment of the assignment

▸ [1] **assignment fulfilled**
[2] assignment fulfilled with minor objections
[3] assignment fulfilled with major objections
[4] assignment not fulfilled

The thesis was fulfilled and both the written part and especially the implementation goes beyond what was expected.

2. Main written part

95 / 100 (A)

The thesis is well written, easy to follow and engaging. The text is mostly to the point and while not overly formal, it is precise and self-contained (though some knowledge of Haskell is most likely required to be able to follow the thesis).
It is not easy to talk about programming language, type systems and type checking and the text does it pretty well.

The following are just a few points I would add / change:

- The language draws a lot from Haskell. It would be nice to have some brief comparison of the similarities with Haskell - feature-wise / syntax-wise. Currently, it is mostly in the thesis outline.
- It would be great to have a discussion about which type system features were considered, perhaps as some kind of a diagram what depends on what, to provide a bit of an overview of the exploration.
- It would be also nice to have some sort of overview of the "related work", i.e., what other functional programming language implementation tutorials are related to the thesis.
- The implementation chapter could have a better connection to the code.
- A few subsections in the Introduction are missing examples (1.6.5.3) while other shows rather straightforward ones (1.6.4.1).

- Sections "2.6 Interpretation" and "2.7 Interpretation" have confusingly the same name.
- The 4-level of nested sections with named paragraphs are perhaps a bit too dense.
- Finally (I know I'm stretching it a bit), would be great to do some comparison of the different implementation strategies, e.g., how is it done in Haskell.

3. Non-written part, attachments 100/100 (A)

The thesis is implementation heavy. In fact, the implementation is the main contribution, answering to the question how do you combine several advanced type-system features into a practical, working programming language.
While the features were independently described in the relevant papers, turning that into a working implementation is by far not an easy task. In the papers, there are often missing pieces, or the language on which the feature is demonstrated / formalized is way too simplistic.
Moreover, integrating them together is the main challenge.

4. Evaluation of results, publication outputs and awards 100/100 (A)

The main contribution is in the implementation making it a great resource for someone who would like to enter the implementation of statically-typed functional programming languages.
As far as I'm aware, there is no such an implementation of advanced type system feature written from scratch which is fairly complete, yet still hackable, i.e., it is not part of an industrial compiler whose code base it large and the concepts are intertwined with much other concerns.
I would love to see this work extended and published in some form.

The overall evaluation 100/100 (A)

The topic of this thesis is in many ways a dream one - you get to dive into design and implement advance type system features in your own programming language (and do that in Haskell). It does not get much better than that!
The result is a working language with a number of advanced features packed in a form that can be fairly easy to explore and hack which could be a great source for people interested in this topic.

Questions for the defense

1. General impredicative polymorphism is not implemented yet in GHC, but there is a support (via the ImpredicativeTypes extension which came out of retirement) for some restricted types, e.g. when the arrow type constructor. For example, it is possible to do this:

```
f :: (forall a. a -> a) -> (a, b) -> (a, b)
f g (x, y) = (g x, g y)
```

Would you consider that?

2. The Glask's operators allow for somewhat more flexibility than Haskell's operators. This extension is however perhaps less practical. How about adding mixfix operators from Agda which seems to be both practical and flexible.

3. How did you pick the type-system features, what did you leave out, what would you like to have in?

Instructions

Fulfillment of the assignment

Assess whether the submitted FT defines the objectives sufficiently and in line with the assignment; whether the objectives are formulated correctly and fulfilled sufficiently. In the comment, specify the points of the assignment that have not been met, assess the severity, impact, and, if appropriate, also the cause of the deficiencies. If the assignment differs substantially from the standards for the FT or if the student has developed the FT beyond the assignment, describe the way it got reflected on the quality of the assignment's fulfilment and the way it affected your final evaluation.

Main written part

Evaluate whether the extent of the FT is adequate to its content and scope: are all the parts of the FT contentful and necessary? Next, consider whether the submitted FT is actually correct – are there factual errors or inaccuracies?

Evaluate the logical structure of the FT, the thematic flow between chapters and whether the text is comprehensible to the reader. Assess whether the formal notations in the FT are used correctly. Assess the typographic and language aspects of the FT, follow the Dean's Directive No. 52/2021, Art. 3.

Evaluate whether the relevant sources are properly used, quoted and cited. Verify that all quotes are properly distinguished from the results achieved in the FT, thus, that the citation ethics has not been violated and that the citations are complete and in accordance with citation practices and standards. Finally, evaluate whether the software and other copyrighted works have been used in accordance with their license terms.

Non-written part, attachments

Depending on the nature of the FT, comment on the non-written part of the thesis. For example: SW work – the overall quality of the program. Is the technology used (from the development to deployment) suitable and adequate? HW – functional sample. Evaluate the technology and tools used. Research and experimental work – repeatability of the experiment.

Evaluation of results, publication outputs and awards

Depending on the nature of the thesis, estimate whether the thesis results could be deployed in practice; alternatively, evaluate whether the results of the FT extend the already published/known results or whether they bring in completely new findings.

The overall evaluation

Summarize which of the aspects of the FT affected your grading process the most. The overall grade does not need to be an arithmetic mean (or other value) calculated from the evaluation in the previous criteria. Generally, a well-fulfilled assignment is assessed by grade A.