Hi Haddock

...or how to get Haddock docstrings into .hi files

Student: Simon Jakobi

Synopsis

A long-standing issue with Haskell's documentation tool Haddock is that it needs to effectively re-perform a large part of the parse/template-haskell/typecheck compilation pipeline in order to extract the necessary information from Haskell source for generating rendered Haddock documentation. This makes Haddock generation a costly operation, and makes for a poor developer experience.

An equally long-standing suggestion to address this issue (c.f. "Haddock strings in .hi files" email thread) is to have GHC include enough information in the generated `.hi` interface files in order to avoid Haddock having to duplicate that work. This would pave the way for following use-cases and/or have the following benefits:

Benefits to the Community

- Significantly speed up Haddock generation by avoiding redundant work.
- On-the-fly/lazy after-the-fact Haddock generation in `cabal new-haddock` and `stack haddock` for already built/installed Cabal library packages.
- Add native support for a `:doc` command in GHCi's REPL and editor tooling (ghc-mod/HIE) similar to the one available in other languages (c.f. the Idris REPL or the Python REPL)
- Allow downstream tooling like Hoogle or Hayoo! to index documentation right from interface files.
- Simplify Haddock's code base.

Deliverables

- Make GHC include all Haddock annotations in the generated `.hi` files.

- Add a `:doc` command to GHCi that reads the documentation from the `.hi` files. (If there is a decision to do so I will alternatively adapt the existing `:info` command to output the docs.)
- Stretch/optional goal: Teach GHCi to render the documentation and not just print the raw comment strings.

Implementation strategy for the changes to GHC

1. Teach GHC to lex, parse and rename haddock comments again. `haddock-library` can take care of lexing and parsing, renaming can be adapted from `rename` in `haddock-api`. Switch `DocDecl` and other documentation types in GHC to use String as this is what haddock-library currently uses
2. Add two new fields to GHC's `ModIface` and `ModGuts` resembling `ifaceDocMap` and `ifaceArgMap` from Haddock's interface files.
3. Populate `ModGuts` with the documentation for declarations (taken from the `ParsedSource`). This can be modeled after haddock-api's `mkMaps` and `topDecls` functions and integrated into `hscDesugar'`.
4. Teach `MkIface` to serialise the collected documentation in its parsed and renamed AST structure.

In the implementation I will make sure to handle the documentation as lazily as possible in order not to incur a performance hit in the common case.

Timeline

April 23–May 13 (before the coding period)

- Finish ongoing work on GHC.
- Familiarize myself with the GHCi codebase.

May 14–20 (start of coding period)

- Introduce dependency on `haddock-library`.
- Add back lexing, parsing and renaming of haddocks. (Step 1)

May 21–May 27

- Serialise the module headers to .hi files, thereby getting a first impression of the layers

involved in steps 2–4.
- Set up first tests that use `--show-iface` to test the serialised `.hi` files.

May 28–June10

- Fully implement steps 2 and 3, i.e. populate `ModGuts` with the documentation.

June 11–15: First evaluation period

June 11–July 8

- Fully implement the serialisation to `.hi` files (Step 4).
- Add the `:doc` command to GHCi that uses the `.hi` files.

July 9–13: Second evaluation period

July 9–22

- Wrap up the implementation, write documentation.
- If so decided, merge the new GHCi `:doc` command into the existing `:info` command

July 23–August 6

- Buffer for any unpredictable delay.

The following stretch goals may be attempted if the buffer is not needed:

- Switch the implementation and haddock-library to use `ByteString` instead of `String` in order to increase performance.
- Teach GHCi to format/render the documentation.

August 6–14: Final evaluations

About me

I'm a student at Leipzig University wrapping up my B.Sc. in Computer Science.
I have been contributing to the Haskell open source community since 2015. Much of that

work was with the `stack` project.
In February 2018 I have started to contribute to GHC directly.

Contact information

- Email: simon.jakobi@gmail.com
- GitHub: https://github.com/sjakobi