Author: Julius Gvozdiovas
Contact: j.gvozdiovas@student.tudelft.nl
Supervisors: Jesper Cockx, Leonhard Applis

Not all extensions are equal

Taxonomy of Haskell language extensions based on function and usage

RQ1: What are the community built language extensions for Haskell?

RQ2: How can language extensions be classified into useful categories?

RQ3: How widespread is the use of language extensions in Haskell projects?

RQ4: What type of projects use which type of extensions?

RQ5: Do developers stick to language extensions they have used before?

What are Haskell language extensions?

“Language extensions are used to enable language features in Haskell that may seem useful in certain cases. They can be used to loosen restrictions in the type system or add completely new language constructs to Haskell” [0]

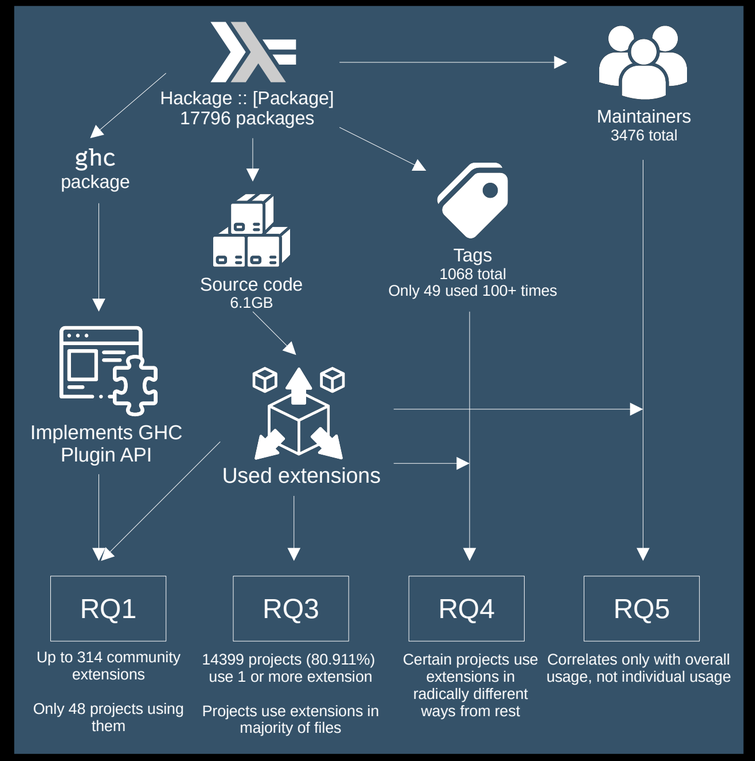
Using GHC extensions in Haskell projects:
- `default-extensions` in .cabal files
- `LANGUAGE` pragma. For example:
```
{-# LANGUAGE OverloadedStrings #-}
```
- `OPTIONS_GHC` pragma. For example:
```
{-# OPTIONS_GHC -XOverloadedStrings #-}
```

Using community built extensions (implemented as GHC plugins) is done using `OPTIONS_GHC` pragma together with `-fplugin` flag. Example:
```
{-# OPTIONS_GHC -fplugin Control.Super.Monad.Plugin #-}
```

Background

Many papers detailing individual extensions [1, 2, 3], no papers examine the overall ecosystem of Haskell language extensions.

Prior work for examining usage of language extensions has been used to decide which extensions are part of language standards GHC2021 and GHC2024 [4].



Discussion & Conclusion

Great proportion of projects using extensions indicates Haskell’s versitility, but perhaps also a need for standartization.
Extension taxonomies can be made based on functionality (e.g. syntax vs types) and usage (e.g. one-off quality-of-life vs project-wide overhaul)
Community builds extensions, but does not use them.

Future work

Only newest versions on Hackage were analysed
- Analysing past versions
- Use different source code forges, e.g. GitHub

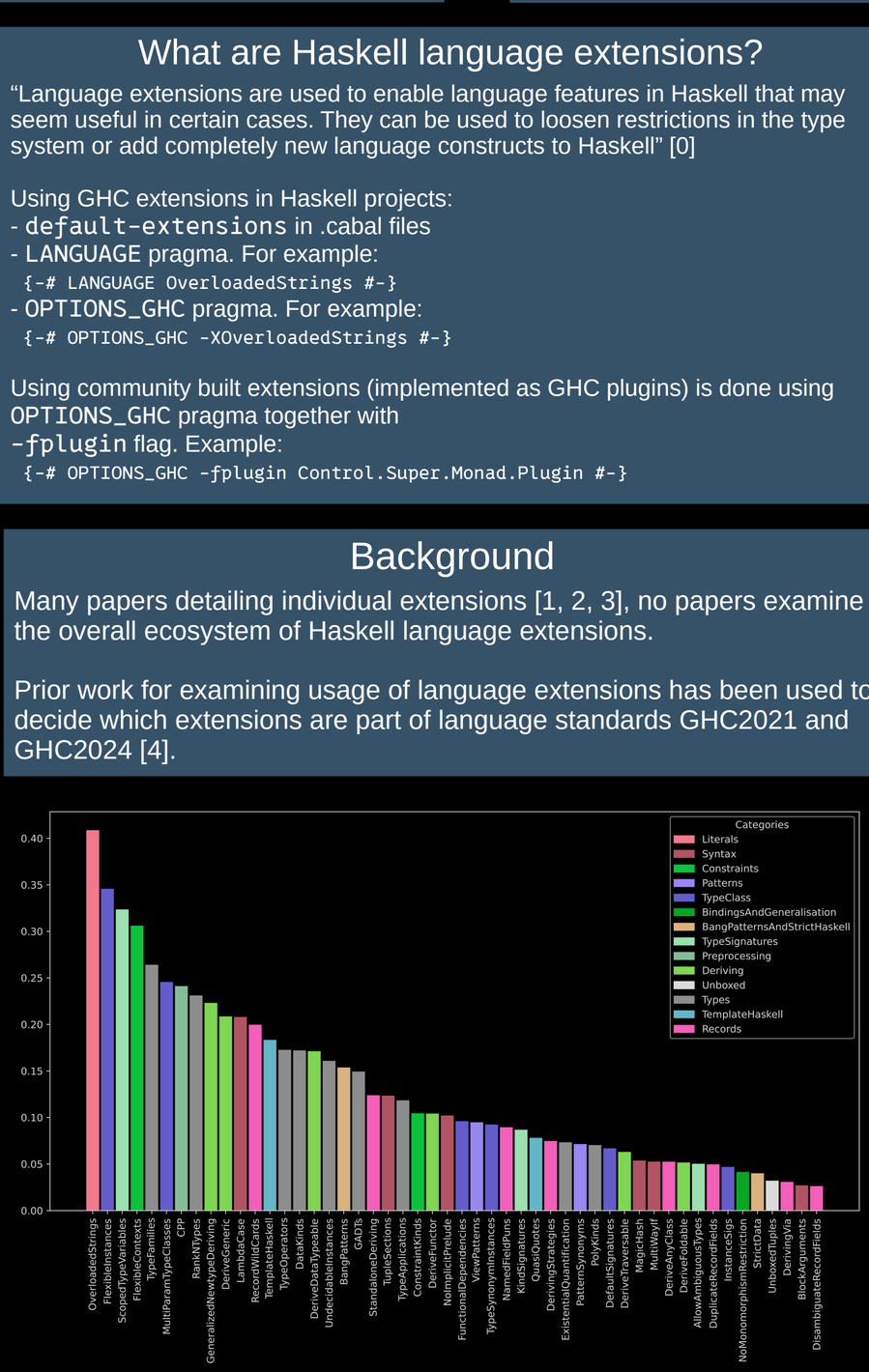


Figure 1. 50 most-used GHC extensions, plotted by portion of projects using them. Functionality-based categorization is shown.

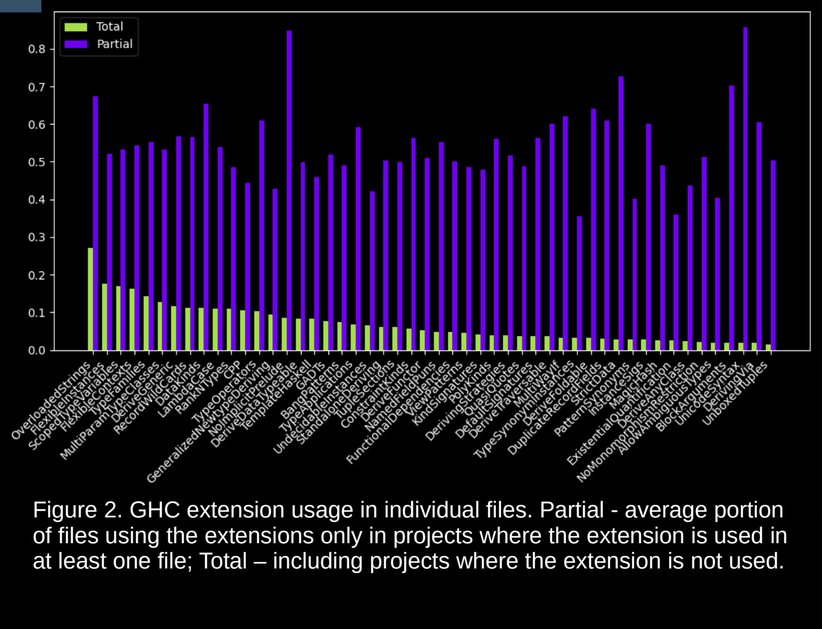


Figure 2. GHC extension usage in individual files. Partial - average portion of files using the extensions only in projects where the extension is used in at least one file; Total – including projects where the extension is not used.

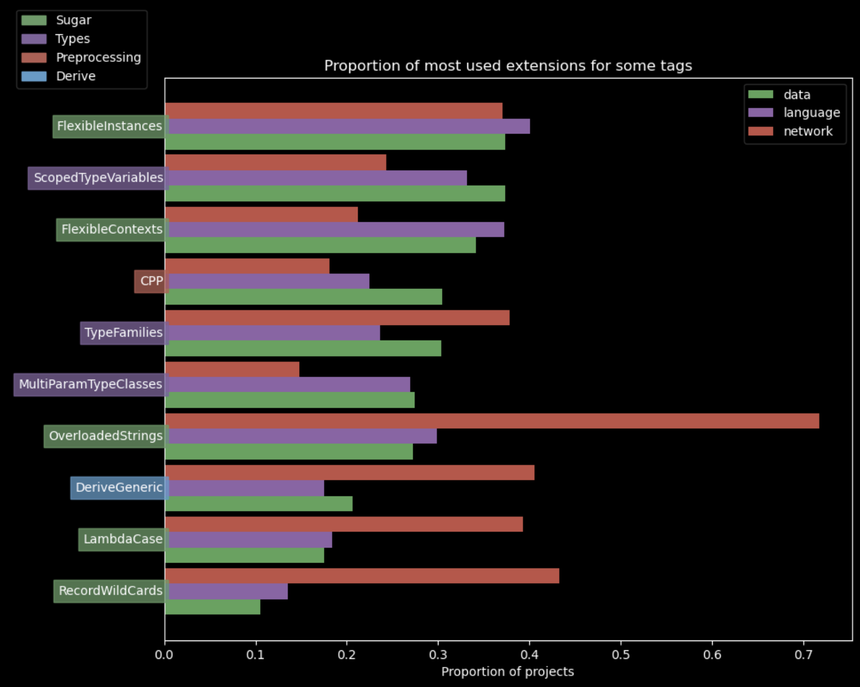


Figure 3. Usage of extensions, based on specific tags. We can observe that “data” and “language” tags tend to use similar extensions, but differ from “network”.

Icon image credit: Flaticon.com, Haskell
[0] Haskell Wiki. Use of language extensions. https://wiki.haskell.org/Language_extensions
[1] Jan Stolarek, Simon Peyton Jones, and Richard A. Eisenberg. “Injective type families for Haskell”. In: SIGPLAN Not. 50.12 (Aug. 2015), pp. 118–128. issn: 0362-1340. doi: 10.1145/2887747.2804314.
[2] Jeff Epstein, Andrew P. Black, and Simon Peyton-Jones. “Towards Haskell in the cloud”. In: SIGPLAN Not. 46.12 (Sept. 2011), pp. 118–129. issn: 0362-1340. doi: 10.1145/2096148.2034690.
[3] Jan Bracker and Henrik Nilsson. “Supermonads: one notion to bind them all”. In:SIGPLAN Not. 51.12 (Sept. 2016), pp. 158–169. issn: 0362-1340. doi: 10.1145/3241625.2976012.
[4] Joachim Breitner. GHC 2021 Proposal.https://github.com/ghc-proposals/ghc-proposals/blob/master/proposals/0380-ghc2021.rst. [Accessed 11-06-2024].2021.