Dead extension elimination for Haskell modules

Péter Podlovics, Boldizsár Németh, Tamás Kozsik

Department of Programming Languages and Compilers
ELTE Eötvös Loránd University, Budapest

{anabra,nboldi,kto}@caesar.elte.hu

Haskell is a purely functional programming language with an expressive type system. In addition to the most recent standard [1], many compilers, such as GHC [2] and UHC [3] provide extensions [4] to the language. The functionality of these extensions can range from mere syntactic modifications to the grammar of Haskell, to introducing completely new semantic features.

Many extensions are compiler specific, hence using them can reduce the portability of the source code. The currently existing tools [5] are only capable of identifying unused syntactic extensions, but they provide no information on those that require semantical analysis.

In this paper, we present a method which helps programmers reason about the language extensions in their Haskell modules. With our approach, we can find and eliminate all unnecessary extensions from the modules, moreover, we can discover the exact locations of the language constructs that require a certain extension. The method has been implemented in the Haskell-Tools framework [6]. We also apply our technique to real-life Haskell packages and analyze the results.

References

[1] Haskell2010 Language Report:
https://www.haskell.org/onlinereport/haskell2010/

[2] Glasgow Haskell Compiler:
https://wiki.haskell.org/GHC

[3] Utrecht Haskell Compiler:
http://foswiki.cs.uu.nl/foswiki/UHC

[4] GHC specific extensions:
http://downloads.haskell.org/~ghc/latest/docs/html/users_guide/glasgow_exts.html

[5] **Neil Mitchell**: HLint package (Hackage), 2006–2018.
Available at: https://hackage.haskell.org/package/hlint

[6] **Boldizsár Németh**: Haskell-Tools framework (github), 2016–2018.
Available at: https://github.com/haskell-tools/haskell-tools