A quick overview of the Haskell tooling

Julien Dehos

Fosdem 2023
Who am I?

- assistant professor in Computer Science at ULCO, France
- using Haskell since 2015 (for teaching FP + small projects)
50 shades of Haskell tooling

- compiler: GHC
- build tools: cabal, stack
- editors (+ plugins): vscode, vim, emacs
- LSP implementation: HLS
- setup tools: ghcup, nix

→ in this talk: cabal, vscode + haskell.haskell
Hackage

- community package archive
- https://hackage.haskell.org
vector: Efficient Arrays

[ bsd3, data, data-structures, library ] [ Propose Tags ]

An efficient implementation of Int-indexed arrays (both mutable and immutable), with a powerful loop optimisation framework.

It is structured as follows:

Data.Vector
   Boxed vectors of arbitrary types.

Data.Vector.Unboxed
   Unboxed vectors with an adaptive representation based on data type families.

Data.Vector.Storable
   Unboxed vectors of Storable types.

Data.Vector.Primitive
   Unboxed vectors of primitive types as defined by the primitive package. Data.Vector.Unboxed is more flexible at no performance cost.
instance Monad Vector where
  {-# INLINE return #-}
  return = Applicative.pure

  {-# INLINE (>>=) #-}
  (>>=) = flip concatMap

#if !(MIN_VERSION base)
  {-# INLINE fail #-}
  fail = Fail.fail
#endif

-- | @since 0.12.1.0
instance Fail.MonadFail Vector where
  {-# INLINE fail #-}
  fail _ = empty

instance MonadPlus Vector where
  {-# INLINE mzero #-}
  mzero = empty

  {-# INLINE mplus #-}
  mplus = (++)

(a -> Vector b) -> Vector a -> Vector b
forall a b. (a -> Vector b) -> Vector a -> Vector b
Hoogle

▶ “Haskell google”

▶ https://hoogle.haskell.org
search by function name or by type signature:

- search by function name or by type signature:

  - (a -> b) -> [a] -> [b]

  **map**: (a -> b) -> [a] -> [b]

  - base Prelude Data.List GHC.Base GHC.List GHC.Ord.List, ghc GHC.Prelude, haskell-gi-base
  - Data.Gi.Base.ShortPrelude, ihaskell IhaskellPrelude, ghc-lib-parser GHC.Prelude, rebase

  map f xs is the list obtained by applying f to each element of xs, i.e.,

  - strictMap**: (a -> b) -> [a] -> [b]

  - ghc GHC.Util.Misc, ghc-lib-parser GHC.Util.Misc

  **map**: (a -> b) -> [a] -> [b]

  - base Prelude, rio RIO.List RIO.Prelude, numeric-prelude NumericPrelude
  - NumericPrelude.Base, relude Prelude.List, reexport, dimensional Numeric.Units.Dimensional.Prelude,

  map f xs is the list obtained by applying f to each element of xs, i.e.,
Cabal

- system for building and packaging Haskell libraries and programs

- write a cabal file:

```plaintext
myproject.cabal

1  cabal-version: 3.4
2  name: myproject
3  version: 0.1.0.0
4  description:
5      This is myproject.
6
7  common shared-properties
8      default-language: Haskell2010
9      ghc-options: -Wall
10
11 library
12      import: shared-properties
13      build-depends: base
14      hs-source-dirs: src
15      exposed-modules: Tree
16
17 executable myproject
18      import: shared-properties
19      build-depends: base, myproject
20      hs-source-dirs: app
21      main-is: Main.hs
```
run the cabal tool:

```
[nix-shell:~/code/myproject]$ cabal build
Build profile: -w ghc-9.0.2 -O1
In order, the following will be built (use -v for more details):
  - myproject-0.1.0.0 (exe:myproject) (file app/Main.hs changed)
Preprocessing executable 'myproject' for myproject-0.1.0.0..
Building executable 'myproject' for myproject-0.1.0.0..
[1 of 1] Compiling Main
    ( app/Main.hs, /home/julien/code/myproject/dist-newstyle/build/x86_64-linux/ghc-9.0.2/myproject-0.1.0.0/x/myproject/build/myproject/myproject-tmp/Main.o )
Linking /home/julien/code/myproject/dist-newstyle/build/x86_64-linux/ghc-9.0.2/myproject-0.1.0.0/x/myproject/build/myproject/myproject ...

[nix-shell:~/code/myproject]$ cabal run myproject data/example1.txt
Up to date
list: [1,2,2,3,3,8,23,34,43,52,123,432,432,541,893]
sum: 2592
length: 15
```
REPL

Read-Eval-Print Loop

[nix-shell:-/code/myproject]$ cabal repl
Build profile: -w ghc-9.0.2 -O1
In order, the following will be built (use -v for more details):
- myproject-0.1.0.0 (lib) (ephemeral targets)
Preprocessing library for myproject-0.1.0.0..
GHCi, version 9.0.2: https://www.haskell.org/ghc/ :? for help
[1 of 1] Compiling Tree
   ( src/Tree.hs, interpreted )
Ok, one module loaded.
ghci>
ghci> import Data.Foldable
ghci>
ghci> :info Tree
   type Tree = * -> *
data Tree v = Leaf | Node (Tree v) v (Tree v)
   -- Defined at src/Tree.hs:10:1
instance [safe] Eq v => Eq (Tree v) -- Defined at src/Tree.hs:13:15
instance [safe] Foldable Tree -- Defined at src/Tree.hs:13:19
instance [safe] Show v => Show (Tree v)
   -- Defined at src/Tree.hs:13:29
ghci>
ghci> mytree = fromList [2, 1, 4, 3]
ghci>
ghci> toList mytree :: [Int]
[1,2,3,4]
ghci>
ghci>
Visual Studio Code (+ HLS + haskell.haskell)

- file edition, syntax highlighting:
code navigation/documentation:
code completion:
- compilation, refactoring:
holes:
inline REPL:
Haddock

document the code:

```haskell
-- | This is the Tree module
module Tree where

import Data.List ( foldl' )

-- | A polymorphic binary Tree
data Tree v
    = Leaf
    | Node (Tree v) v (Tree v)
    deriving (Eq, Foldable, Show)

-- | Build a Tree from a List
--

-- >>> fromList [2,1,3] :: Tree Int
-- Node (Node Leaf 1 Leaf) 2 (Node Leaf 3 Leaf)
fromList :: Ord a => [a] -> Tree a
fromList = foldl' insert1 Leaf
    where
```

```
generate the documentation:

```
[nix-shell:~/code/myproject]$
cabal haddock
Build profile: -w ghc-9.0.2 -01
In order, the following will be built (use -v for more details):
- myproject-0.1.0.0 (lib) (file src/Tree.hs changed)
./myproject.cabal has been changed. Re-configuring with most recently used options. If this fails, please run configure manually.
Configuring library for myproject-0.1.0.0..
Preprocessing library for myproject-0.1.0.0..
Running Haddock on library for myproject-0.1.0.0..
83% ( 5 / 6) in 'Tree'
Missing documentation for:
  mysum (src/Tree.hs:47)
Documentation created:
/home/julien/code/myproject/dist-newstyle/build/x86_64-linux/ghc-9.0.2/myproject-0.1.0.0/doc/html/myproject/index.html
```
result:

```haskell
data Tree v
#
A polymorphic binary Tree

Constructors

Leaf
Node (Tree v) v (Tree v)

Instances

Foldable Tree
#
Show v => Show (Tree v)
#
Eq v => Eq (Tree v)
#

fromList :: Ord a => [a] -> Tree a
#
Build a Tree from a List

>>> fromList [2,1,3] :: Tree Int
Node (Node Leaf 1 Leaf) 2 (Node Leaf 3 Leaf)
```
Tests

- unit tests, with HSpec:

```hs
main :: IO ()
main = hspec spec

spec :: Spec
spec = do
  describe "fromList" $ do
    it "[1,2,3]" $ fromList [1,2,3::Int] `shouldBe` Node Leaf 1 (Node Leaf 2 (Node Leaf 3 Leaf))
    it "[2,1,3]" $ fromList [2,1,3::Int] `shouldBe` Node (Node Leaf 1 Leaf) 2 (Node Leaf 3 Leaf)
  describe "toList . fromList" $ do
    it "[1,2,3]" $ toList (fromList [1,2,3]) `shouldBe` [1,2,3]
    it "[2,1,3]" $ toList (fromList [2,1,3]) `shouldBe` [1,2,3]
```
results:

```bash
[nix-shell:~/code/myproject]$ cabal test --test-show-details=always
Build profile: -w ghc-9.0.2 -01
In order, the following will be built (use -v for more details):
  - myproject-0.1.0.0 (test:spec) (first run)
Preprocessing test suite 'spec' for myproject-0.1.0.0..
Building test suite 'spec' for myproject-0.1.0.0..
Running 1 test suites...
Test suite spec: RUNNING...

Tree
  fromList
    [1,2,3]
    [2,1,3]
  toList . fromList
    [1,2,3]
    [2,1,3]

Finished in 0.0001 seconds
4 examples, 0 failures

Test suite spec: PASS
Test suite logged to:
/home/julien/code/myproject/dist-newstyle/build/x86_64-linux/ghc-9.0.2/myproject-0.1.0.0/t/spec/test/myproject-0.1.0.0-spec.log
1 of 1 test suites (1 of 1 test cases) passed.
```
test properties, with QuickCheck:

```haskell
let
    prop_sort :: [Int] -> Bool
    prop_sort = isSorted . toList . fromList

describe "quickCheck" $ do
    it "prop_sort" $ property prop_sort
```
result:

```
Test suite spec: RUNNING...

Tree
d fromList
    [1,2,3]
    [2,1,3]
  toList . fromList
    [1,2,3]
    [2,1,3]
quickCheck
d prop_sort
    +++ OK, passed 100 tests.
```
Conclusion

- Haskell has some nice tools for many years (cabal, repl, QuickCheck.
- and intuitive tools since recently (vscode plugins, HLS.
- through a quite easy setup : ghcup + vscode + haskell.haskell
References

- slides & code: https://gitlab.com/juliendehos/talk-2023-fosdem
- 2022 State of Haskell Survey Results: https://taylor.fausak.me/2022/11/18/haskell-survey-results
- The Varieties of the Haskelling Experience: https://www.tweag.io/blog/2021-11-25-varieties-of-haskelling-experience
Thank you! Questions/discussion?