# Haskell

Logo created with Haskell script:

https://wiki.haskell.org/TW-Logo-Haskell

# Intro

In September 1987 a meeting was held at the conference on Functional Programming Languages and Computer Architecture in Portland, Oregon, to discuss an unfortunate situation in the functional programming community…
This document describes the result of that commitee's: a purely functional programming language called Haskell, named after logician Haskell B. Curry whose work provides the logical basis for much of ours.

**Report on the Programming Language Haskell, A Non-strict Purely Functional Language. Version 1.0, 1 April 1990**

Latest release of the Glasgow Haskell Compilation System is 8.6.5 / 23 April 2019

https://gitlab.haskell.org/ghc/ghc

One more compilation system: Hugs, latests release September 2006

# Intro

Non-strict means Haskell is not lazy. But it's lazy by default.

-- Just like me.

Purely Functional means you can use first-class functions and higher-order functions and you can't use side effects and input/output.

-- At least outside monads.

# How to start (hint: it's easy)

MacOS:

brew install haskell-stack

https://www.haskell.org/platform/mac.html

Windows:

https://www.haskell.org/platform/windows.html

https://github.com/lambdaheart/Haskell-Guide/blob/master/DevelopmentEnvironment.md

Linux:

https://www.haskell.org/platform/linux.html

If you have a lot of time you can build it from source. But GHC is written in Haskell.

So to build GHC you need GHC.

# What we have now

- GHC allows you to compile projects
- Cabal to work with modules
- Stack to do it even better
- Set of common modules and packages
- GHCi is GHC's interactive environment, in which Haskell expressions can be interactively evaluated and programs can be interpreted

  Let's try it:

```
GHCi, version 8.6.3: http://www.haskell.org/ghc/  :? for help
λ> 2+2
4
λ> reverse "abrakadabra"
"arbadakarba"
λ> take 20 . zipWith const [1..] $ drop 100 [1..]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
```

# Stack, or Cabal: that is the question

More information here:

https://stackoverflow.com/questions/30913145/what-is-the-difference-between-cabal-and-stack

http://www.scs.stanford.edu/16wi-cs240h/labs/stack.html

Or maybe Nix?

# Stack

```
$ stack new awesome-project
Downloading template "new-template" to create project "awesome-project" in awesome-project/ ...
... some operations here ...
Writing configuration to file: awesome-project/stack.yaml
All done.
$ cd ./awesome-project/
$ stack build
Building all executables for `awesome-project' once. After a successful build of all of them, only specified executables will be rebuilt.
awesome-project-0.1.0.0: configure (lib + exe)
Configuring awesome-project-0.1.0.0...
awesome-project-0.1.0.0: build (lib + exe)
... some operations here ...
Registering library for awesome-project-0.1.0.0..
$ stack exec -- awesome-project-exe
someFunc
$
```

# Hello World!

```
module Main where
import System.IO

main :: IO ()
main = putStrLn "Hello Haskell!"


-- That's it.
```

# Fibonacci numbers

```
fib :: Int -> Integer
fib 0 = 0
fib 1 = 1
fib n = fib (n-2) + fib (n-1)

main :: IO ()
main = show . fib . read . head <$> getArgs >>= putStrLn


-- That's it.
```

# Fibonacci memoized

```
fib :: Int -> Integer
fib = ([fib' n | n <- [0..]] !!)
    where fib' 0 = 0
          fib' 1 = 1
          fib' n = fib (n-2) + fib (n-1)

main :: IO ()
main = show . fib . read . head <$> getArgs >>= putStrLn
-- That's it.
```

Read more about memoization here:

https://wiki.haskell.org/Memoization

# Replace URLs

## Problem:

- Huge amount of documents (9878). Each document has various length and different structure.
- Huge amount of URLs to replace (48474). Long list of strings like: /home/kitchen/best-air-fryers/ /best-air-fryers
- Only one human life.

# What to do?

# Alfred V. Aho and Margaret J. Corasick

We will use Aho-Corasick algorithm to solve it.

Aho-Corasick algorithm is generalization of Knuth-Morris-Pratt search algorithm to a set of patterns.

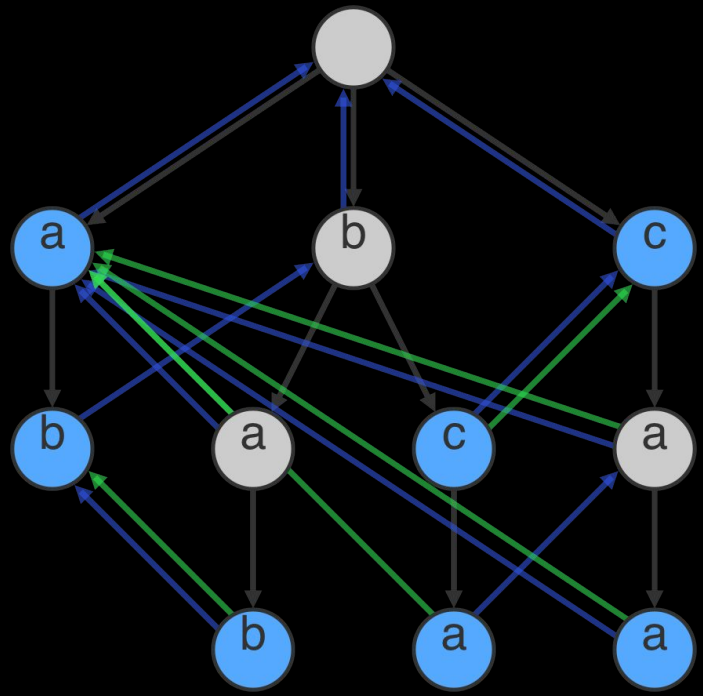
More information:

http://www.cbcb.umd.edu/confcour/Spring2010/CMSC858W-materials/Lecture4.pdf

https://en.wikipedia.org/wiki/Aho-Corasick_algorithm

https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm

I'm too lazy to build it!



# Hoogle

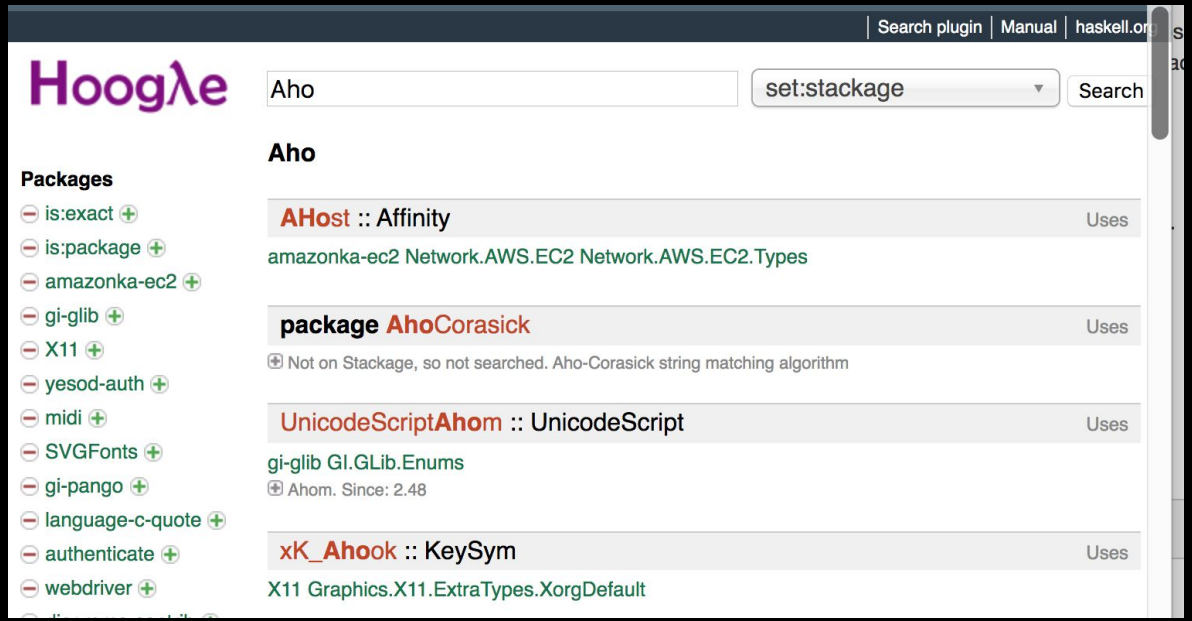
Everything is easier with a proper tool.

https://hoogle.haskell.org/?hoogle=Aho

http://hackage.haskell.org/package/AhoCorasick

Easy to use. Just try to find:

```
Ord a => [a] -> [a]
```



# We did it!

```
$ stack build
stack exec -- future-haskell-demo +RTS -s -RTS ./data/rewrite.txt ./data/in ./data/out

It works!

9878 documents
48474 rewrite rules
80.223s elapsed
```

## Can it be faster?

# Concurrent programming

An easy task for Control.Concurrent and Control.Concurrent.STM modules!

forkIO, forkFinally to create threads

TMVar to wait for threads to be finished

That's it.

# We did it again!

```
$ stack build
$ stack exec -- future-haskell-demo-threaded +RTS -s -N4 -RTS ./data/rewrite.txt ./data/in
./data/out
```

It works!

9878 documents

48474 rewrite rules

**39.375s** elapsed

**Can it be faster? Maybe. Do it.**

## And here's where I keep assorted lengths of wire.

How I downloaded all the articles from TTR.

Library:

Proof.IO

- Proof.IO.Articles
- Proof.IO.Tags
- Proof.IO.Images
- Proof.IO.Authors

Set of tools.

https://gitlab.futurenet.com/dkurilo/proof-bulk-edit

# Join me!

# Further reading

1. Programming in Haskell, Graham Hutton, Second Edition
   https://www.amazon.com/Programming-Haskell-Graham-Hutton/dp/
2. https://simonmar.github.io/pages/pcph.html
3. https://www.haskell.org/
4. https://www.haskell.org/documentation/ (Tutorials)
5. https://github.com/hmemcpy/milewski-ctfp-pdf

   Repos:

1. https://gitlab.futurenet.com/dkurilo/proof-bulk-edit
2. https://gitlab.futurenet.com/dkurilo/haskell-presentation

# How to draw an Owl.

*"A fun and creative guide for beginners"*

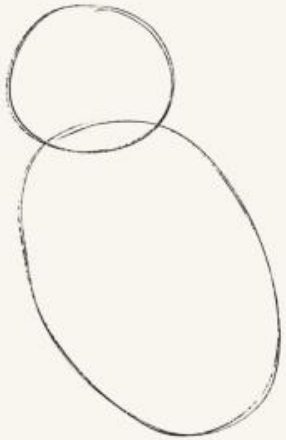


Fig 1. Draw two circles

Fig 2. Draw the rest of the damn Owl

# Questions?

Just in case someone forgot.

My name: Dima Kurilo

EMail address: dmitry.kurilo@futurenet.com

dkurilo@gmail.com