Haskell: A quantitative evaluation

By Adi Maini and Anton Nekhai

Quick Recap on Haskell

- Purely functional language (No OOP)
- Referentially transparent
    - Makes unit testing code much easier
- Support for simple parallel implementations
- Strong typing
    - Has a search engine called Hoogle where you can search for functions based on their typing (inputs and outputs)
    - Good type system means if code compiles, it’s correct most of the time
- Broad support for generic code through typeclasses, algebraic data types, optional data types, and monads

Evaluation Criteria

- Accessibility
- Package Ecosystem
- Employability
- Academic Use
- Industry Adoption
- Performance
- Simplicity

Accessibility

- Number of textbooks is a measurement of learning resources available
- Number of subreddit members of a language is a measurement of community
- Number of GitHub repositories is a measurement of community use of language

	Haskell	C	C++	JAVA	Python	LISP	Clojure	Scala
# of textbooks on Amazon	160	~20,000	~20,000	~3,000	~3,000	225	116	163
# of subreddit members	56k	83k	142k	190k	602k	28k	23k	34k
# of GitHub repositories	34k	478k+	558K+	1M+	961k+	8k+	35k+	59k+

Accessibility - Stack Overflow usage statistics

- Measurement of community size and difficulty of learning the language by the number of questions people ask on Stack Overflow
- Measured using the [language] tags
- We also report statistics about questions asked in the last 30 days and the percent unanswered in the last 30 days (note these numbers fluctuate) to judge how active the communities are

Accessibility - Stack Overflow results

	# total questions	% total unanswered	# recent questions	% recent unanswered	Total users watching
Haskell	44,179	7.4	249	24.1	18.5k
C	333,634	16.6	3,266	40	456.6k
C++	678,318	19.3	7,252	41.7	609.6k
Java	1,685,933	28.6	14,945	55.4	1.4m
Python	1,465,409	27.9	35,719	51.3	1.1m
LISP	6,194	5	23	17.4	5.5k
Clojure	16,390	9.7	57	21.1	8.5k
Scala	98,643	20.8	959	41.5	38.9k

Package Ecosystems

	Haskell (Hackage)	JAVA (Maven)	Python (PyPi)	Clojure (Clojar)
# of packages in ecosystem	14,960	304,586	241,916	26,462

Employability

- Measuring the number of jobs where Haskell language is a skill compared to other languages
- Median salary statistics from Stack Overflow's developer 2020 survey for the US

	Haskell	Java	Python	Clojure	Scala	LISP
Indeed	208	33,097	35,035	149	2,922	75
LinkedIn	349	93,673	86,809	371	9,262	148
Median Salary	121k	120k	120k	N/A	150k	N/A

Academic Use

	# of academic papers on IEEE	# of academic papers on Google Scholar
Haskell	156	94,200
C	22,665	3,820,000
C++	17,221	58,800
Java	7,794	2,330,000
Python	1,388	335,000
LISP	318	126,000
Clojure	16	4,370
Scala	109	127,000

Industry Adoption

	Google	Microsoft	Amazon	Apple	Facebook	Intel	Total (from stackshare)
Haskell	Used on a small number of internal projects and IT infrastructure	Used for product serialization, supports language development	N/A	N/A	Some internal tools	Developed a compiler for parallelism research	99
Clojure	No	Yes (swiftkey)	Yes	Yes	Yes	No	229
Scala	No, uses Groovy instead	Yes	Yes	Yes	No	Yes, for big data	843

Speed

- Measured using the computer language benchmarks game:
  - https://benchmarksgame-team.pages.debian.net/benchmarksgame/

Program / Exec time (in sec)	Haskell w/ GHC	Java	C with Clang	OCaml
regex-redux	1.68	10.27	1.41	25.23
fasta	1.40	2.22	1.44	6.00
mandlebrot	5.06	6.84	4.70	14.02
binary-trees	13.41	8.28	4.81	9.89
n-body	21.87	21.85	6.11	21.67
k-nucleotide	36.99	9.14	9.49	21.23

Simplicity

- Measured the length of the code used to implement the benchmarks on the last slide (lines with comments have been removed)

Program / Lines of code	Haskell w/ GHC	Java	C with Clang	OCaml
regex-redux	245	61	165	46
fasta	152	325	295	105
mandlebrot	237	66	65	67
binary-trees	50	84	98	58
n-body	186	170	165	105
k-nucleotide	167	174	190	223

Conclusions

- **Should you learn Haskell?** *Probably not (based on our metrics)*
- The language is not widely used among companies, though there are some high paying jobs for Haskell devs out there
- The language is surprisingly fast at certain tasks, but trades brevity for this speed
- The barrier to entry for a beginner is high as there are limited resources to learn, and community is smaller than other languages.

Who *should* learn Haskell?

- People with an interest in purely functional paradigms
- People who want to experiment with easy parallel implementations (like Intel!)
- Masochists
- Mathematicians who took Algebra 1, learned about about Category theory is, and found it humorous that it can be applied to a programming language
    - Functors, Applicative functors, Semigroups, Monoids, Monads, oh my!
- Further consider:
    - Skills learned in Haskell do carry over to other languages, such as Optionals, Generics, and Lambda functions
    - Functional Code has advantages that other languages are trying to utilize, Haskell provides context for why these features exist and matter
    - Hearing about functional programming and understanding how to think functionally are very different, Haskell can help you with the latter

Project Proposal: Haskell MNIST Neural Net

- Objective
  - Learn how to write Haskell code and about the patterns of functional programming
  - Understand the backpropagation algorithm by implementing it ourselves
  - To implement a neural network from the ground up in Haskell and train it with backpropagation and stochastic or batch gradient descent
  - Use the built neural network to train on images of handwritten digits (0-9) using the MNIST dataset, and predict on a set of test images with an accuracy > 85%.
  - Add on: make a front-end site where user can draw a digit which is predicted on
- Constraints (Scope)
  - Do not implement a convolutional neural network as it would require convolutional layers, pooling layers, and additional complexity that would extend the timeline of this project
  - Do not implement a model that would require a GPU or distributed computing to train in a reasonable time
  - Haskell does not have developed neural network, ML, front-end focussed libraries.

Project Proposal: Haskell MNIST Neural Net

- Features
  - Develop a scalable pipeline
  - Use a feed-forward neural network with 2 layers
  - Include orthogonality to include different kinds of layers
  - A simple frontend to draw a number on a grid and then classify it
- Technologies used
  - MNIST data set
  - AD symbolic differentiation package
  - Accel parallel acceleration package
  - Github for version control
  - VS Code and extensions for syntax highlighting, linter functions, etc.
  - GHC, Stack, and Cabal to compile, build, and manage dependencies respectively
  - Python for a simple frontend
- Where this project can be applied
  - This project can be used as a shell to create a neural network model for multi-class classification problems. The number of layers and neurons at each layer, including the output layer can be changed to fit the problem at hand.

Questions?