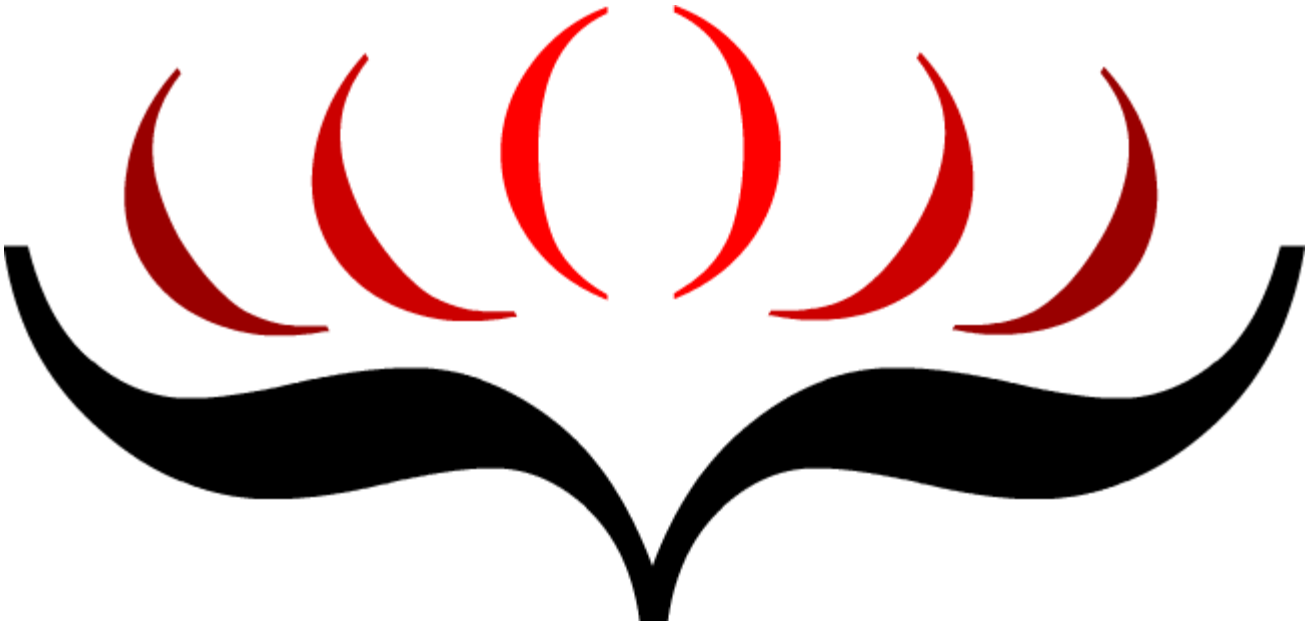
The philosophy and history of Haskell



Vladimir Alekseichenko

Agenda

- History of Haskell
- Language philosophy
- Goals and principles
- Haskell in industry
- Haskell community

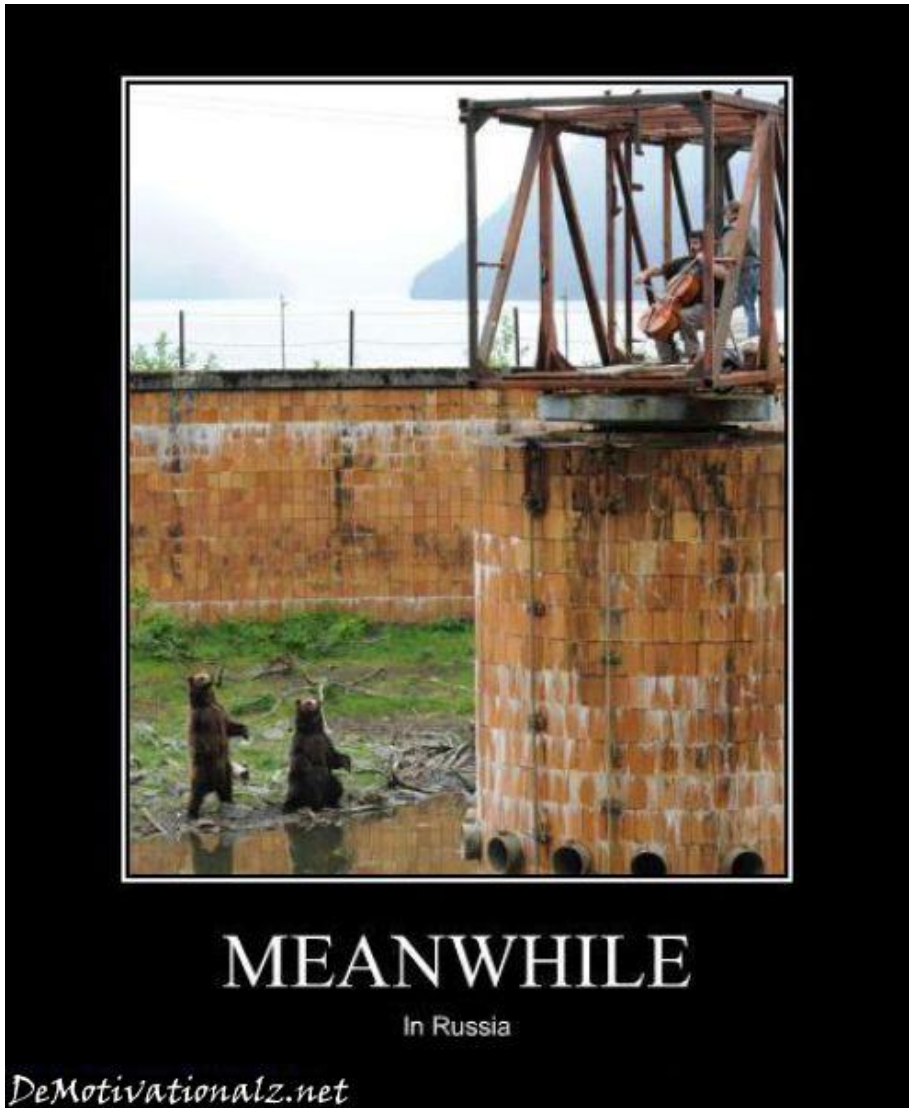

MADNESS?
THIS IS HASKELL
memegenerator.net


HIPSTER PROGRAMMER
USE HASKELL AND TYPE THEORY
memegenerator.net

1980s

http://en.wikipedia.org/wiki/1980s

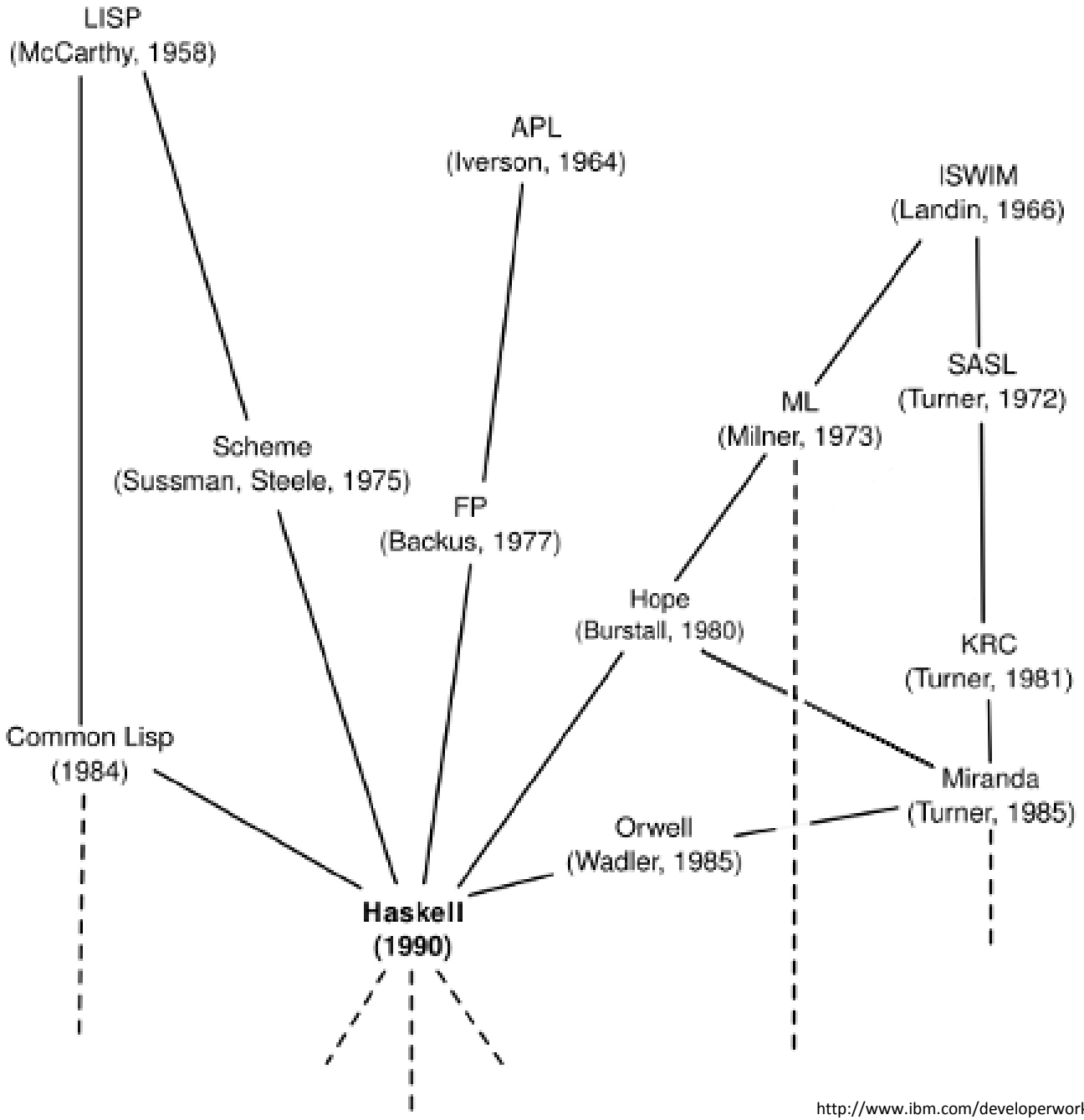
Meanwhile, in … functional programming



Each group develops its own language

Significant impact on Haskell

- ML (1973) – the first language with typed Hindley-Milner.
- SASL, KRC, Miranda (1972–1985) – one of the first lazy languages.
- Hope (1980) – one of the first languages with algebraic data types.



http://www.ibm.com/developerworks/ru/library/l-haskell/

The birth of Haskell

The FPCA meeting (1987) thus marked the beginning of the Haskell design process.

Although we had no name for the language and very few technical discussions or design decisions occurred.

The Yale meeting

- The first physical meeting (after the impromptu FPCA meeting) was held at Yale, January 9–12, 1988,
- The first order of business was to establish the following goals for the language.

Goals for the language

The Yale meeting

1. It should be suitable for teaching, research, and applications, including building large systems.
2. It should be completely described via the publication of a formal syntax and semantics.
3. It should be freely available.
4. It should be usable as a basis for further language research.
5. It should be based on ideas that enjoy a wide consensus.
6. It should reduce unnecessary diversity in functional programming languages.

Choosing a name

Semla, **Haskell**, Vivaldi, Mozart, CFL (Common Functional Language), Funl 88, Semlor, Candle (Common Applicative Notation for Denoting Lambda Expressions), Fun, David, Nice, Light, ML Nouveau (or Miranda Nouveau, or LML Nouveau, or ..), Mirabelle, Concord, LL, Slim, Meet, Leval, Curry, Frege, Peano, Ease, Portland, Haskell B Curry.

Haskell B. Curry

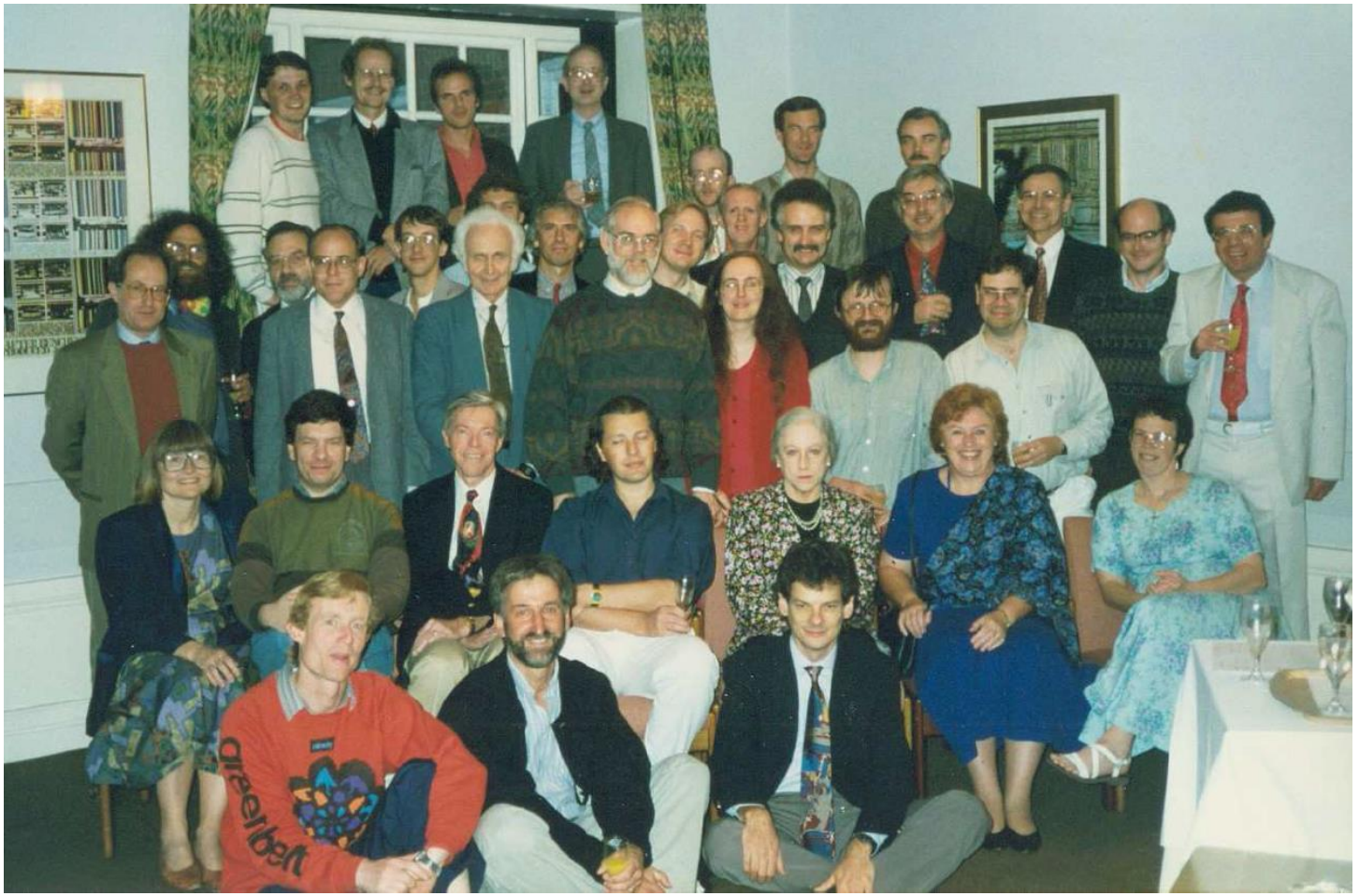
http://www.haskell.org/haskellwiki/Haskell_Brooks_Curry

The Glasgow meeting

- That meeting was held April 6–9, 1988 at the

University of Glasgow, whose Functional programming group was beginning a period of rapid growth.

IFIP WG2.8 meetings



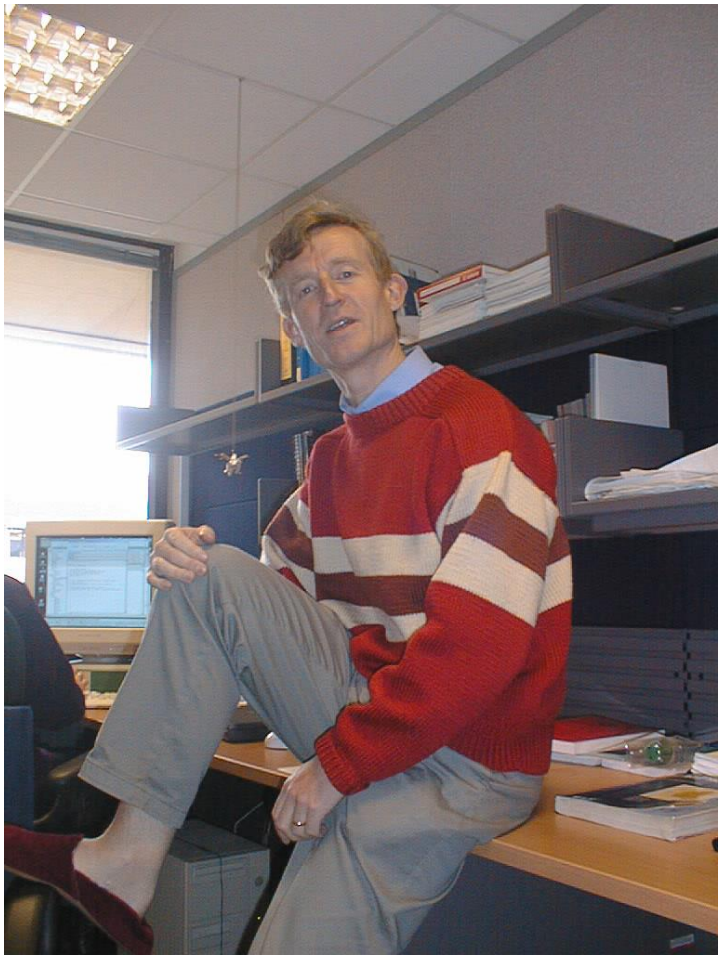
Paul Hudak

http://www.cs.rutgers.edu/news/distinguished/

Philip Wadler

http://homepages.inf.ed.ac.uk/wadler/

Simon Peyton Jones



http://research.microsoft.com/en-us/um/people/simonpj/Pictures/

A short history (1)

- 1 April 1990
  The Haskell version 1.0

- August 1991
  The Haskell version 1.1

- March 1992
  The Haskell version 1.2

A short history (2)

- 1994.

  Haskell gained Internet presence when John Peterson registered the haskell.org domain name and set up a server and web-site at Yale.

- May 1996.

  The Haskell version 1.3

- April 1997.

  The Haskell version 1.4

A short history (3)

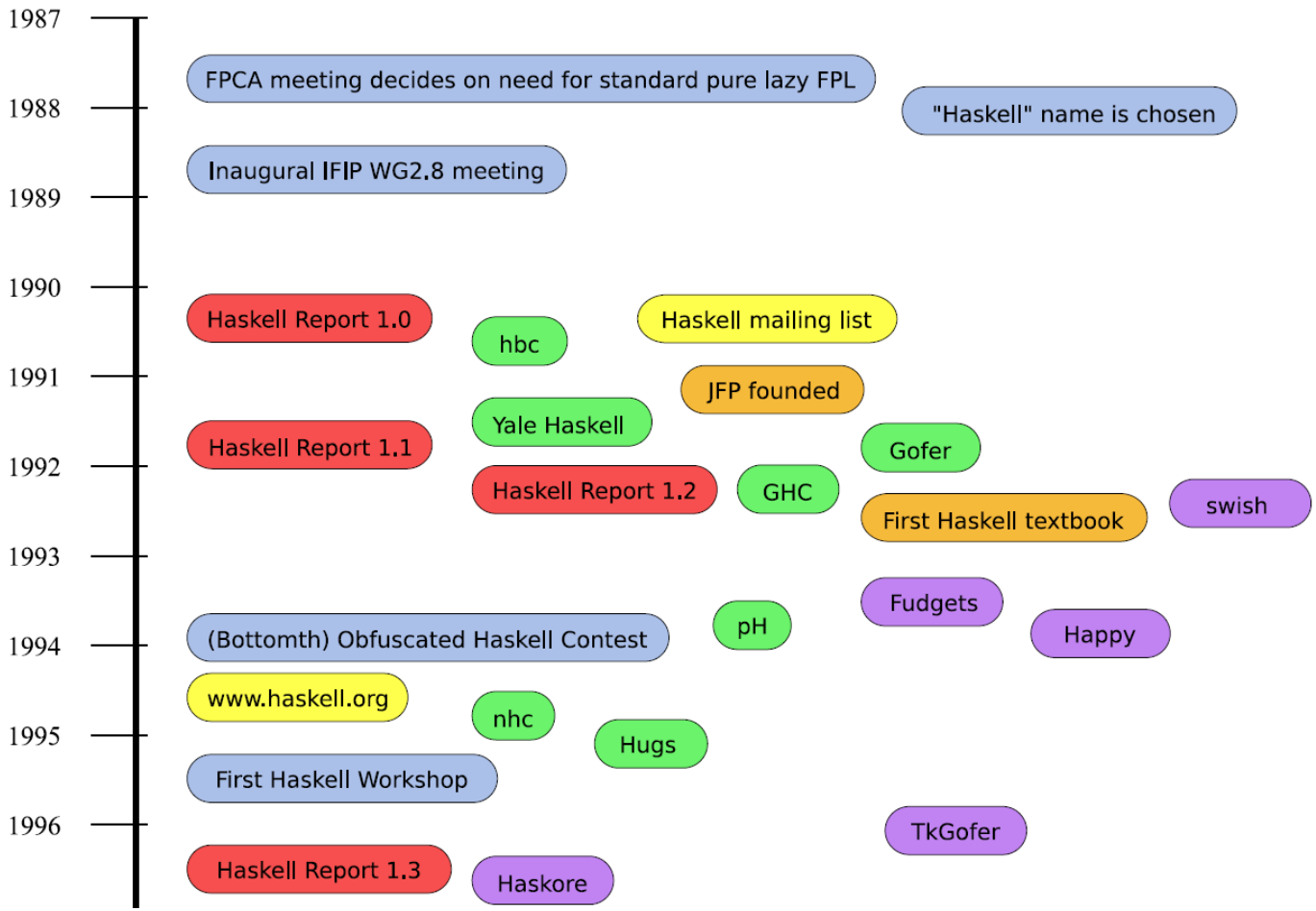
- February 1999

  The Haskell 98

- July 2010

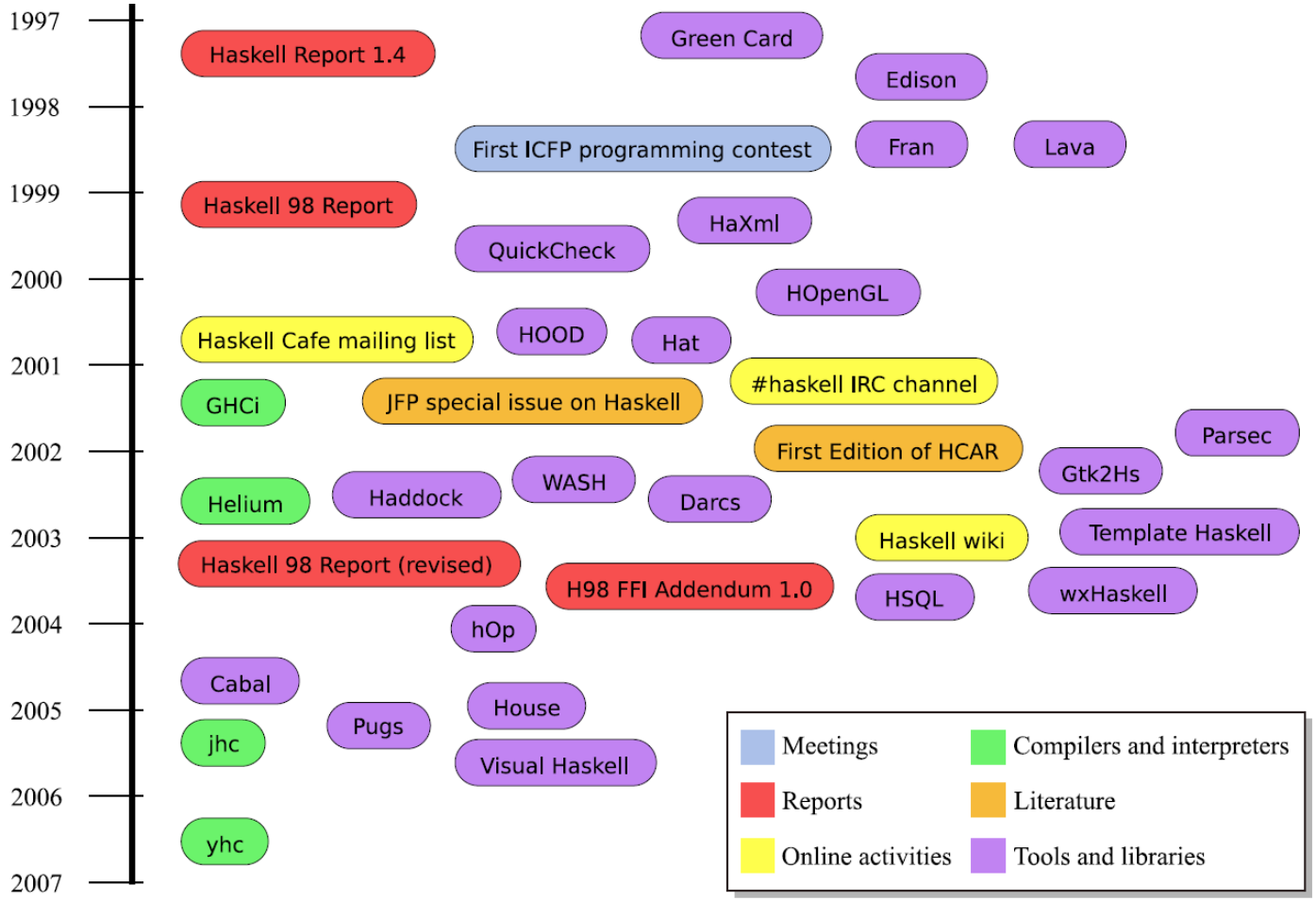
  The Haskell 2010

Haskell timeline (1)



A History of Haskell: Being LazyWith Class

Haskell timeline (2)



A History of Haskell: Being LazyWith Class

Haskell is a committee language

Haskell is a language designed by committee, and conventional wisdom would say that a committee language will be full of warts and awkward compromises.

Properties of Haskell

- Lazy evaluation
- Non-side effect
- GC and automatic allocation of data included
- Strong typing

http://www.slideshare.net/romanandreg/why-haskell-matters

Lazy evaluation

Properties of Haskell

Functions won't perform sequentially, instead they will evaluate expressions when needed.

http://www.slideshare.net/romanandreg/why-haskell-matters

Non-side effect

Properties of Haskell

There is a controlled environment for side effect functions but by default functions are pure.

http://www.slideshare.net/romanandreg/why-haskell-matters

GC

Properties of Haskell

Management of resources is abstracted completely from the developer.

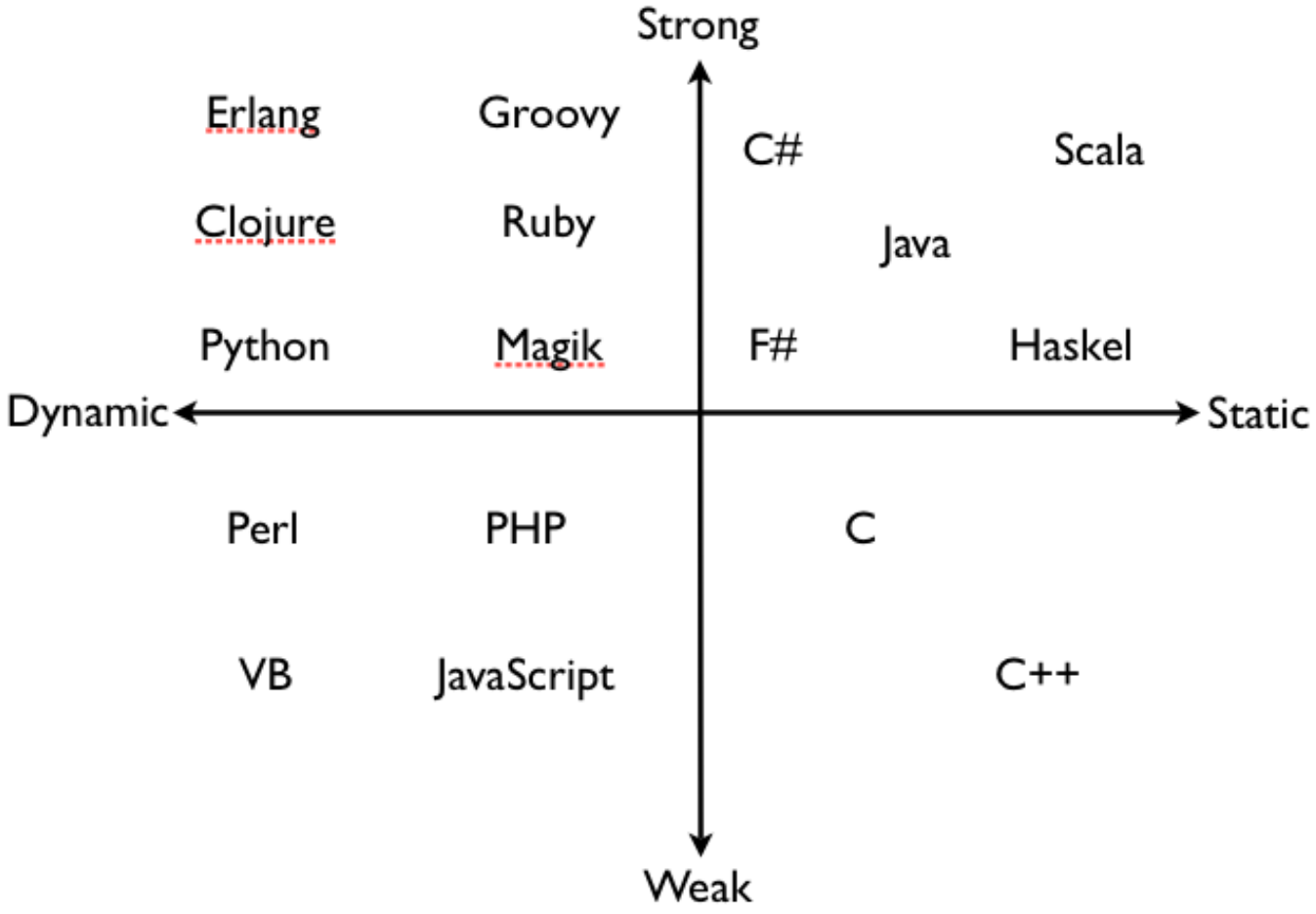
http://www.slideshare.net/romanandreg/why-haskell-matters

Haskell's type system

There are three interesting aspects to types in Haskell:

- they are *strong*,
- they are *static*,
- and they can be automatically *inferred*.

Strong vs weak
dynamic vs static



http://blogs.agilefaqs.com/2011/07/11/dynamic-typing-is-not-weak-typing/

Strong types

- The type system guarantees that a program cannot contain certain kinds of errors.
- These errors come from trying to write expressions that don’t make sense, such as using an integer as a function.

Strong types

Another aspect of Haskell's view of strong typing is that it will not automatically coerce values from one type to another.

Coercion is also known as casting or conversion.

Real World Haskell by *Bryan O'Sullivan, Don Stewart, and John Goerzen*

Implicitly convert in C#

```
int intValue = 1;
uint uintValue = 1;
long longValue = 1;
ulong ulongValue = 1;
```

C# is strongly a typed language, but it has a different meaning than in Haskell.

```
float floatValueFromInt = intValue;
float floatValueFromUint = uintValue;
float floatValueFromLong = ulongValue;
float floatValueFromUlong = longValue;

double doubleValueFromLong = longValue;
double doubleValueFromUlong = ulongValue;
```

Static types

The compiler knows the type of every value and expression at compile time, before any code is executed.

Real World Haskell by *Bryan O'Sullivan, Don Stewart, and John Goerzen*

Static types

```
Prelude> True && 0

<interactive>:15:9:
    No instance for (Num Bool) arising from the literal `0'
    Possible fix: add an instance declaration for (Num Bool)
    In the second argument of `(&&)', namely `0'
    In the expression: True && 0
    In an equation for `it': it = True && 0
Prelude> True && "False"

<interactive>:16:9:
    Couldn't match expected type `Bool' with actual type `[Char]'
    In the second argument of `(&&)', namely `"False"'
    In the expression: True && "False"
    In an equation for `it': it = True && "False"
```

Static types

- Static typing can occasionally make it difficult to write some useful kinds of code.

- Fortunately, Haskell's system of *typeclasses*, provides almost all of the benefits of dynamic typing, in a safe and convenient form.

What are *typeclasses*?

- *Typeclasses* define a set of functions that can have different implementations depending on the type of data they are given.

- *Typeclasses* may look like the objects of object-oriented programming, but they are truly quite different.

Real World Haskell by *Bryan O'Sullivan, Don Stewart, and John Goerzen*

A Haskell program that compiles will not suffer from type errors when it runs.

Haskell's combination of strong and static typing makes it impossible for type errors to occur at runtime.

Real World Haskell by *Bryan O'Sullivan, Don Stewart, and John Goerzen*

Jigsaw puzzle

http://thamesford-united-church.com/ministries/children-and-youth/

Type inference

A Haskell compiler can automatically deduce the types of almost all expressions in a program.

Real World Haskell by *Bryan O'Sullivan, Don Stewart, and John Goerzen*

Implementations

Haskell is a language with many implementations, two of which are widely used:

- GHC
- Hugs

The Glasgow haskell compiler

- Probably the most fully featured Haskell compiler today is the Glasgow Haskell Compiler (GHC), an open-source project with a liberal BSD-style licence.

GHC has three main components

- ghc – an optimizing compiler that generates fast native code
- ghci – an interactive interpreter and debugger
- runghc – a program for running Haskell programs as scripts, without needing to compile them first

Haskell User's Gofer System

Hugs

It’s a bytecode interpreter for the functional programming language Haskell.

GHC vs Hugs

Compared to Hugs, GHC is more suited to “real work”: it compiles to native code, supports parallel execution, and provides useful performance analysis and debugging tools.

Good For Equational Reasoning

Gofer

It’s an implementation of the programming language Haskell intended for educational purposes and supporting a language based on version 1.2 of the Haskell report.

Yale Haskell

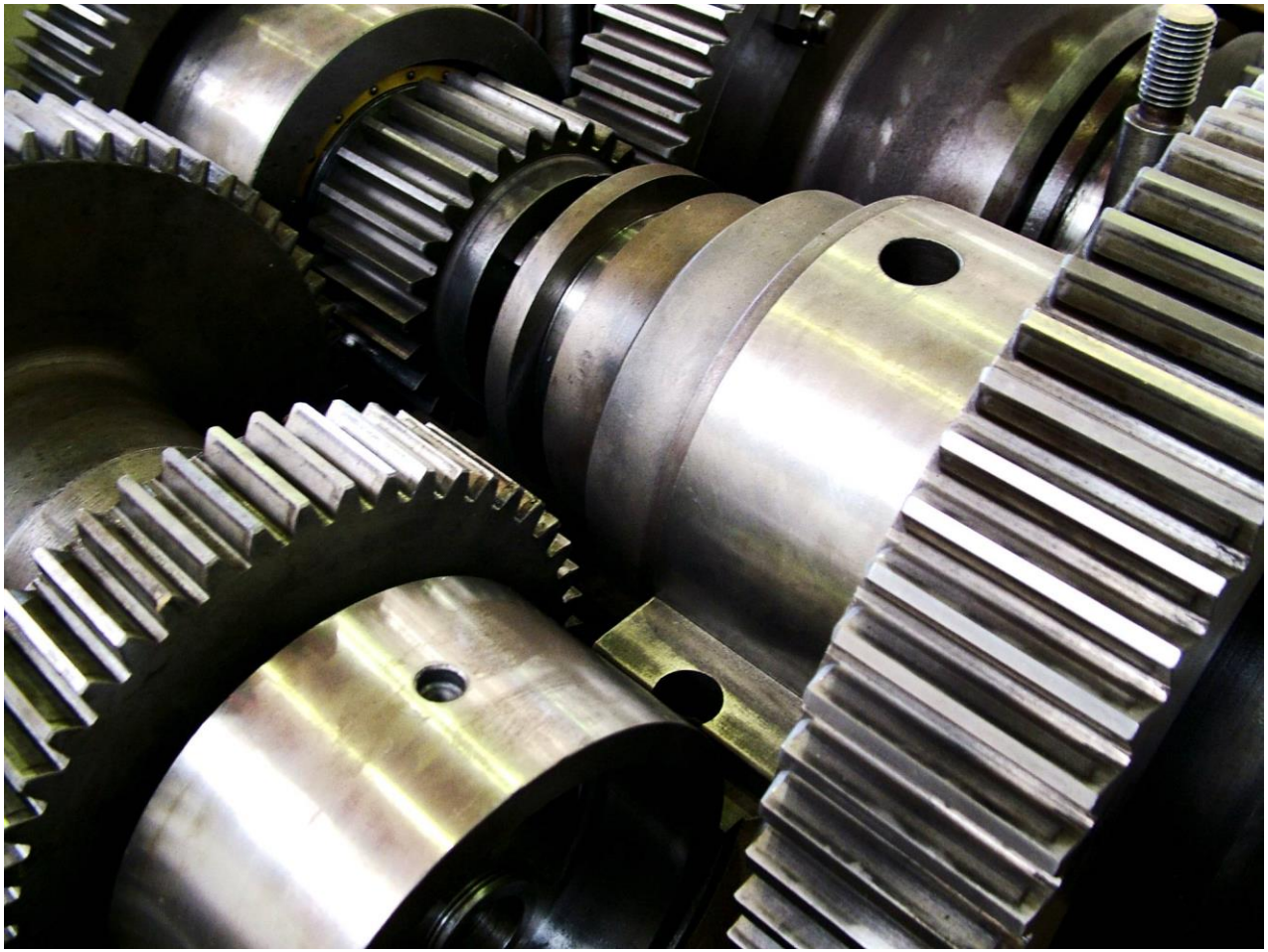
An early implementation of Haskell in Lisp

The Yale Haskell compiler is slower than the Glasgow and Chalmers Haskell compilers.

Other Haskell compilers

- Helium
- UHC and EHC
- Jhc
- The York Haskell Compiler

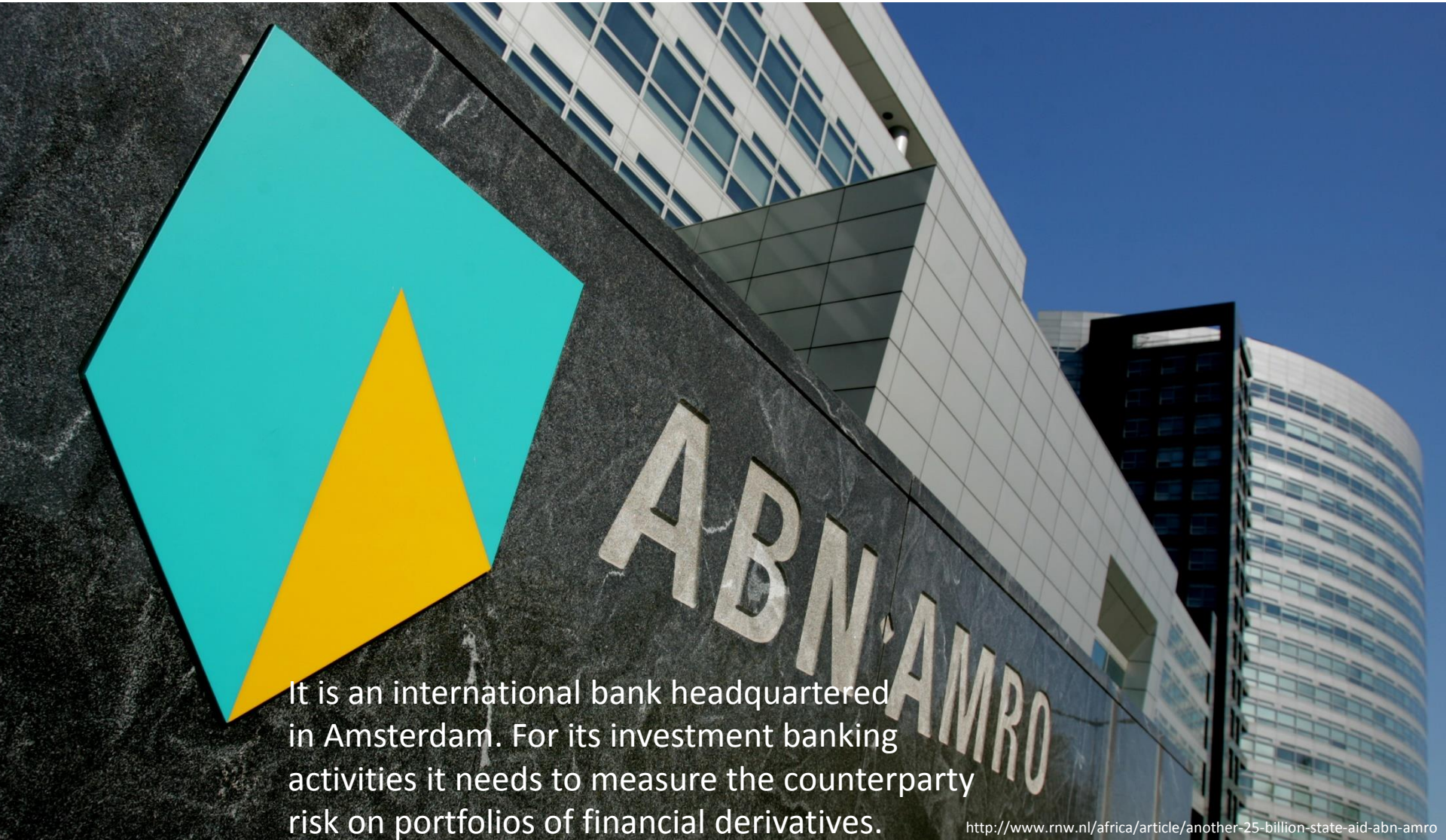
Haskell in industry



http://laboracademy.org/five-industries-for-innovation/

ABN AMRO

Amsterdam, The Netherlands



It is an international bank headquartered in Amsterdam. For its investment banking activities it needs to measure the counterparty risk on portfolios of financial derivatives.

http://www.rnw.nl/africa/article/another-25-billion-state-aid-abn-amro

A consortium of groups, including Alcatel-Lucent, have used Haskell

to prototype narrowband software radio systems, running in (soft) real-time.

http://www.haskell.org/haskellwiki/Haskell_in_industry
http://www.telepresenceoptions.com/2013/02/alcatel-lucent_high_definition/

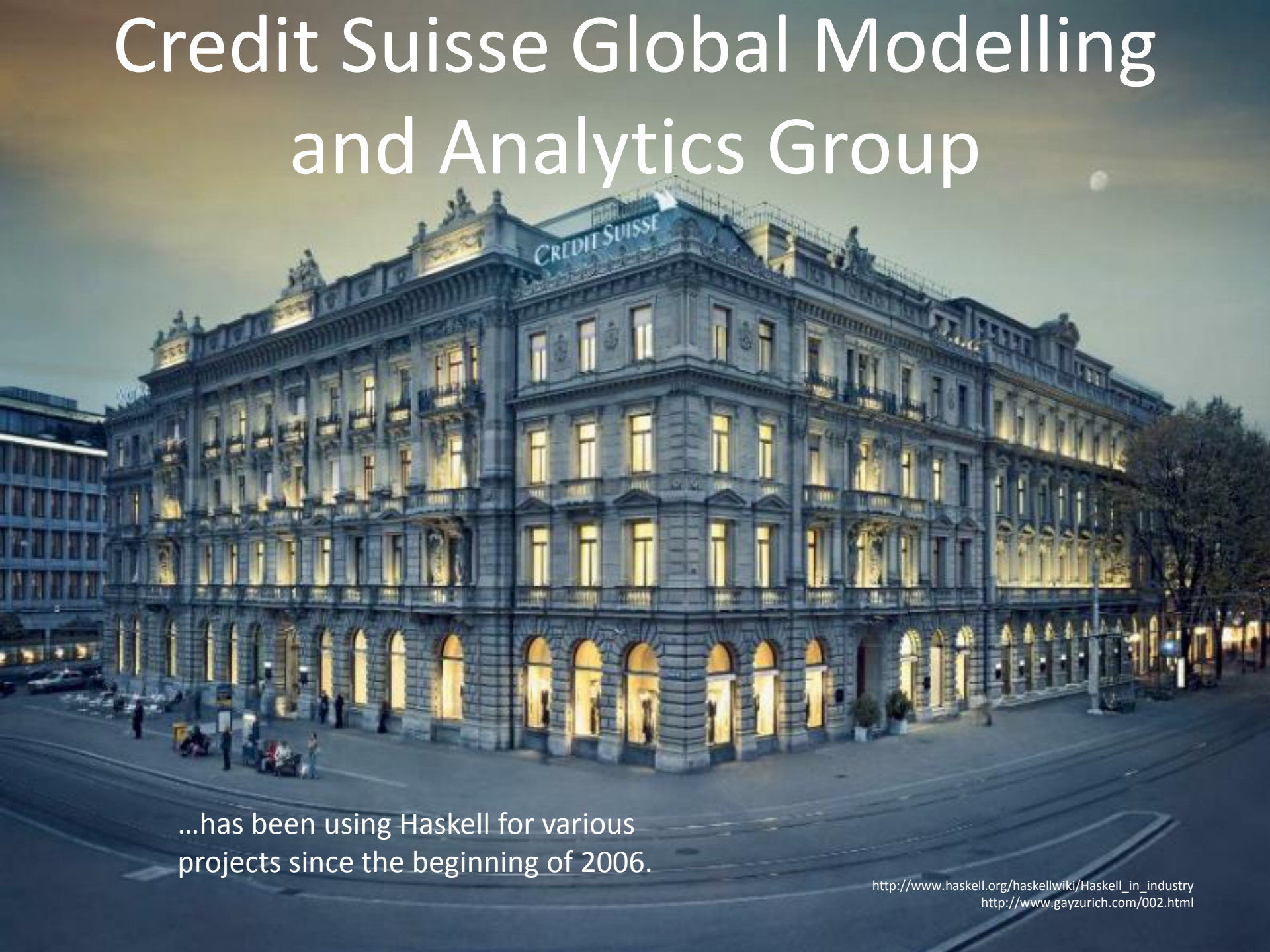
It is a premier high frequency market maker in over 40 financial exchanges, in 20 countries, and in nearly every conceivable product class.

http://www.haskell.org/haskellwiki/Haskell_in_industry
http://career-fair.mit.edu/company_all.php

Haskell is being used for backend data transformation and loading.

http://www.haskell.org/haskellwiki/Haskell_in_industry
http://www.nydailynews.com/2.1353/bank-america-turn-documents-merrill-lynch-takeover-sec-investigators-article-1.381396

Credit Suisse Global Modelling and Analytics Group



…has been using Haskell for various projects since the beginning of 2006.

http://www.haskell.org/haskellwiki/Haskell_in_industry
http://www.gayzurich.com/002.html

Google

Haskell is used on a small number of internal projects in Google, for internal IT infrastructure support.

http://www.haskell.org/haskellwiki/Haskell_in_industry
http://designcollector.net/googles-newest-office-london/google-office-7/

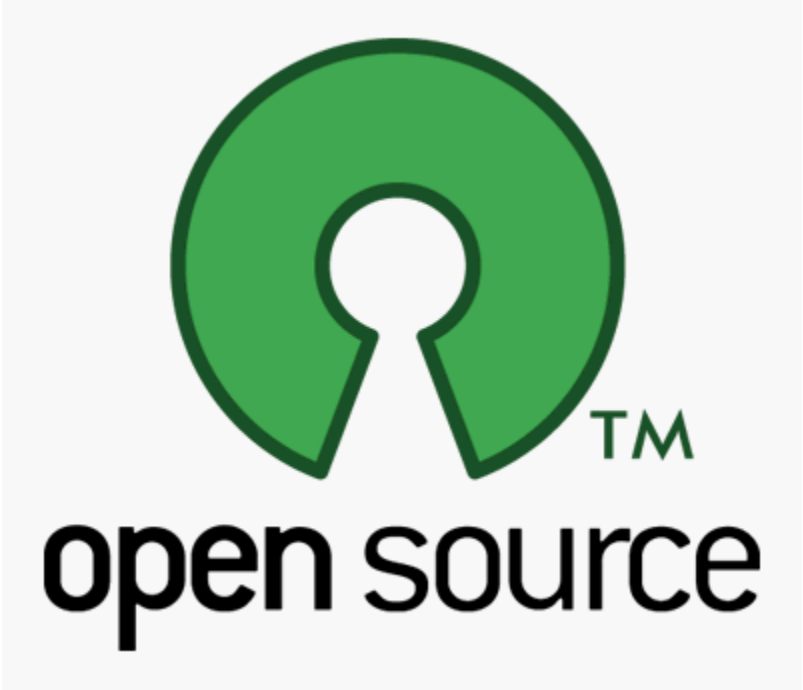
Facebook uses some Haskell internally for tools

http://www.haskell.org/haskellwiki/Haskell_in_industry
http://mashable.com/2012/06/21/facebook-edit-comments/

http://www.haskell.org/haskellwiki/Haskell_in_industry

...

Haskell in open source



http://www.biglever.com/ecosystem/bridges/opensource.html

A databases

- Haskelldb
- HDBC
- Takusen
- Library to work with BerkeleyDB, CouchDB, MongoDB, Redis, TokyoCabinet, TokyoTyrant, SimpleDB, SQLite

http://ru.wikipedia.org/wiki/Haskell

A graphics

- Blobs
- Diagrams
- FieldTrip
- Glome
- GLUT and OpenGL
- GPipe
- Grapefruit
- Haven
- HaskellCharts
- SDL
- Yampa
- …

http://ru.wikipedia.org/wiki/Haskell

Graphical user interface

- Gtk2Hs
- Grapefruit and wxFruit
- qtHaskell
- wxHaskell

http://ru.wikipedia.org/wiki/Haskell

Games

- bloxorz
- Frag
- monadius
- Raincat

http://ru.wikipedia.org/wiki/Haskell

Internet

- gitit
- happstack
- Twidge

http://ru.wikipedia.org/wiki/Haskell

Word processing

- HaXml
- HXT
- Leksah
- Pandoc
- Parsec
- The Grammatical Framework
- Yi

Parallel, multi-tasking and multi-threaded programming

- CHP — Communicating Haskell Processes
- Data Parallel Haskell
- parallel
- STM

http://ru.wikipedia.org/wiki/Haskell

http://ru.wikipedia.org/wiki/Haskell

Development

- alex
- cabal-install
- happy
- haddock
- HUnit
- QuickCheck

http://ru.wikipedia.org/wiki/Haskell

System software

- Darcs
- House
- Xmonad
- Himerge
- FreeArc

http://ru.wikipedia.org/wiki/Haskell

Languages and compilers

- Agda
- Curry
- cpphs
- Epigram
- Flapjax
- Language.Python
- Language.C
- Lava
- llvm
- Pugs
- WebBits

http://ru.wikipedia.org/wiki/Haskell

Compilation, debugging, and performance analysis

- For practical work, almost as important as a language itself is the ecosystem of libraries and tools around it.
- Haskell has a strong showing in this area.

Compilation, debugging, and performance analysis

- Compiles to efficient native code on all major modern operating systems and CPU architectures.
- Easy deployment of compiled binaries, unencumbered by licensing restrictions.
- Code coverage analysis.
- Detailed profiling of performance and memory usage.

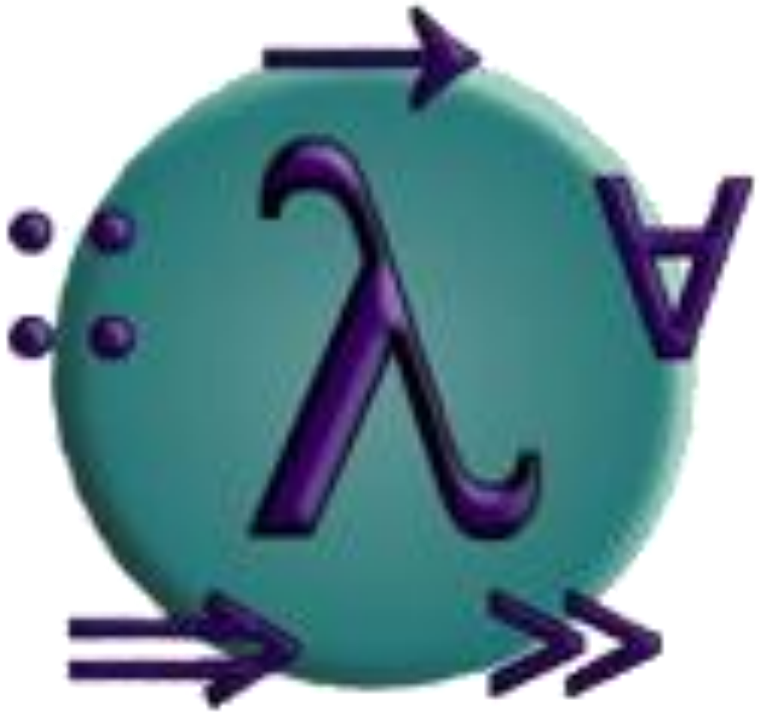
The Haskell community

http://ei.mhs.com/Community.aspx

The Haskell community

1. *http://www.haskell.org/*
2. *http://haskell.org/haskellwiki/Mailing_lists*
3. *http://haskell.org/haskellwiki/IRC_channel*
   named #haskell
4. *http://haskell.org/haskellwiki/User_groups*
5. *http://www.meetup.com/find/*
6. *http://sequence.complete.org/*
7. *http://www.haskell.org/haskellwiki/Haskell_Communities_and_Activities_Report*

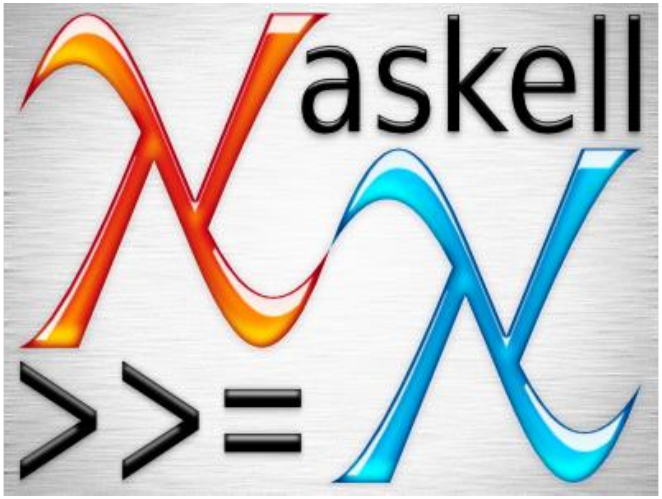
Previous Haskell.org logo



http://www.haskell.org/haskellwiki/Haskell_logos

The great 2009 Haskell logo contest

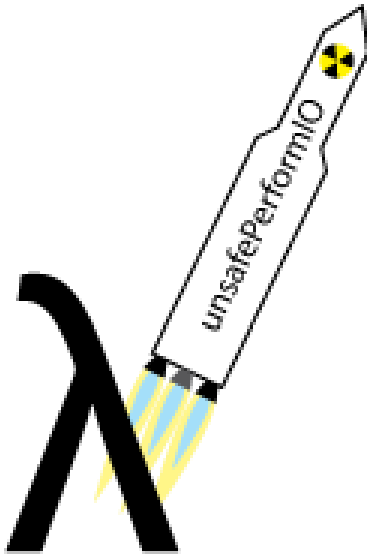
HASKELL
purely functional



Haskell
BIND THE REAL WORLD

http://www.haskell.org/haskellwiki/Haskell_logos/New_logo_ideas

The great 2009 Haskell logo contest



http://www.haskell.org/haskellwiki/Haskell_logos/New_logo_ideas

Current Haskell logo

The current Thompson-Wheeler logo was chosen in the logo competition in early 2009. It replaces the previous Haskell.org logo.

http://www.haskell.org/haskellwiki/Haskell_logos

When learning Haskell is a good idea?

http://imodulesinsider.wordpress.com/2012/05/04/its-friday-so-it-must-be-time-for-a-good-idea-5/

When learning Haskell is a good idea?

- If you are a experienced developer that wants to grow professionally by learning completely new and different approaches to solve problems
- If you need to build systems where correctness is critical (Banks, Safety-Critical Systems)

http://www.slideshare.net/romanandreg/why-haskell-matters

When learning Haskell is a good idea?

- Is well suited to implement Domain Specific Languages, so it could work very well in systems programming
- as Erlang, it is easy to implement parallel and concurrent programs safely due to the pureness of the language

http://www.slideshare.net/romanandreg/why-haskell-matters

Benefits of using Haskell

http://www.movebenefits.co.uk/

Benefits of using Haskell

- Algorithms are normally shorter and more concise than their iterative versions
- Due to the use of function composability, curryfication and high-order functions, code can be reused in really smart ways
- The kick-ass type system with type inference makes you develop code that would normally work at the first compilation

http://www.slideshare.net/romanandreg/why-haskell-matters

Why Haskell is not being used as much?

http://www.raywenderlich.com/20881/beginning-auto-layout-part-1-of-2/rage-why

Why Haskell is not being used as much?

- It's different, people is always afraid of what is different
- It's difficult, mostly related to the first point
- It's unpopular, mostly related to the first and second point

http://www.slideshare.net/romanandreg/why-haskell-matters

Why Haskell is not being used as much?

- It's risky (for employers), there are not many Haskell developers to count on, of course this is mostly related to the first, second and third point
- Libraries are broad, but there is no depth (Many incompatible experimental libraries that do the same thing)

http://www.slideshare.net/romanandreg/why-haskell-matters

Resources

Books and papers

- **A History of Haskell: Being LazyWith Class** by *Paul Hudak, John Hughes, Philip Wadler, Simon Peyton Jones*
- **Real World Haskell** by *Bryan O'Sullivan, Don Stewart, and John Goerzen*
- **Programming in Haskell** by *Graham Hutton*
- **Haskell The Craft of Functional Proaramming** by *Simon Thompson*