Is Haskell ready for everyday computing?

An informal experience report.

Jeff Polakow
CUFP 2008
Talk Overview

- My background
- Job background
- System description
- Points of interest
- Conclusions
About Me

- Theoretical PL research
- Insecurity about utility of my work
- Desire to spread use of “good” languages
About My Job

- Small credit trading group
- Credit markets are opaque
- Information management is main task
- Quantitative analysis less important
Why Haskell?

- Because I can
  - My chance to put theory into practice
  - Curious to know how Haskell fares

- Easiest way for me to be productive
  - Usual typed, higher-order reasons
  - Nicer syntax than OCaml
System Overview

- Database and Web system
- Scheduler to spawn autonomous tasks
- Several communicating pieces
- Distributed over several computers
First Version

- GHC 6.6.1 (started with GHC 6.4)
- HappS 0.8.4, HDBC 1.0.1 (using ODBC)
- All XP

- All process (not thread) based
  - Issues with -threaded

- Somewhat primitive, but stable
Current Version

- GHC 6.8.3
- HAppS 0.9.2, HDBC 1.1.5 (using ODBC)
- XP and Linux
- Threads (where possible) and processes
- Nice machinery for logical processes and servers
- More autonomous pieces talking to each other
Novelties

- Statically typed tables with mini SQL DSL
  - Manipulate tables in memory
  - Generates SQL queries to create a table in memory

- Automatic generation of RPC wrappers

- Proc monad for logical process machinery

- Abstract (socket-based) server machinery
The Good

• Usual stuff
  – Types & type classes for static guarantees
  – First class (higher-order) functions for code reuse

• Purity
  – Able to upgrade old (poorly documented) code with relative ease

• Performance not an issue (for our purposes)
The Bad

• Upgrading to 6.8.3 was painful
  – Some libraries don't like XP
  – Some libraries don't like cabal-install

• Errors / inadequacies of some libraries

• Most library documentation is poor
What is everyday computing?

My very subjective criteria.

- **Database access tools**
  - HDBC, Takusen, etc...

- **Web tools**
  - HAppS, powerful but difficult to install and learn
  - HSP, WASH, etc...
  - Curl bindings, FTP lib work pretty well

- **Ability to write stable server-like programs**
  - Great lightweight threads support
  - Good socket interface
What is everyday computing?

More very subjective criteria.

- **Scripting**
  - ghci as a shell, HSH
  - Good string processing machinery

- **Foreign library interaction**
  - FFI, plus helper tools, are good
  - No easy way to use .NET or Java libs

- **Development Environment**
  - GHC is easy to install & low maintenance
  - Libraries are not always easy to install
  - Available IDEs not adequate for everyone
Is Haskell ready for everyday computing?

Yes

- if you are
  - a seasoned Haskell programmer
  - comfortable with laziness/strictness trade offs
  - comfortable reading library source code
  - capable of understanding and fixing linker errors
  - ...

- and, if in a corporate environment, you are
  - free to try drastically new things
  - capable of functioning without IT dept support