Functional Programming in the Real World

Antti-Pekka Liedes <apl@iki.fi>

6th April 2000

“You’ve scrutinized functional languages. You’ve admired the elegance of lambda calculus, checked the benchmarks from the compilers, noted the security provided by strong typing. Now you want to know if they have been used to serious purpose. Mathematical elegance is well and good, but will it run that mission–critical system?”

Why isn’t functional programming used?

“Compared to users of C, “no one” is a tolerably accurate count of the users of functional languages.”

Compatibility, foreign language interface

- Most operating system interfaces in C/C++
- CORBA, COM, etc. support interaction of any languages
- e.g., a COM interface has been implemented for Haskell

Libraries

- Behind much of the success of Tcl/Tk and Java
- GUI library essential
- Good module system helps
- “Just needs to be done”

Portability

- C strong in portability
- For example, researchers at Lucent would have preferred to build the PRL database language using SML, but chose C++, because SML was not available on the Amdahl mainframe they were required to use.
- Virtual machines make functional languages (or any language) more portable

Availability

- For example, a typical response from a user of Glasgow Haskell is that installing it was an "adventure".
- `./configure && make install`

Packability

- ability to develop compact standalone applications is essential
- shared libraries help
- how to minimize heap size?

Training

- "Curried food and curried functions are both acquired tastes."
- imperative solutions to problems more readily available
- Software AG found they could train industrial programmers to use Natural Expert in a one–week course that included lazy evaluation, polymorphic types, and higher–order functions. Typically, students were miffed when the compiler would repeatedly reject programs for type errors, but pleasantly surprised when their programs finally passed the type checker and ran correctly on the first try.

Tools

- debugger and profiler essential
- integration, ease of use
- integrated development environment?

Popularity

- If a manager chooses to use a functional language for a project and the project fails, then he or she will certainly be fired. If a manager chooses C++ and the project fails, then he or she has the defense that the same thing has happened to everyone else.
- on the other hand, switching to a functional programming language may save a project in trouble by improving productivity
- functional world requires widely recognized "break–throughs"
- Haskell doesn't look as good as Java in your CV

Performance?

- Not really true — roughly twice as slow as C.
- Is it required for success? Tcl/Tk, Perl, Visual Basic, and Java are (were) not very performant either.
- In the future, functional programming may offer great chances for optimizations, e.g., much better *automatic* parallelism.

"They just don't get it"?

- People *do* get the point of functional programming, but the other reasons (lack of support) are against using functional languages.

Why is functional programming used?

“Advocates of functional languages claim they produce an order of magnitude improvement in productivity. Experiments don’t always verify that figure — sometimes they show an improvement of only a factor of four.”

Examples ...

Theorem provers

LCF

- About 1975
- The first good theorem prover with interaction
- ML developed as the metalanguage for LCF
- Exploited higher–order functions and types

HOL

- HOL88 (Lisp) and HOL90 (SML)
- Used in both industry and academia
- hardware design and verification, reasoning about security, proofs about real–time systems, compiler verification, program correctness, modelling concurrency, program refinement
- Open platform for general theorem–proving research

Erlang

- From 1987
- Used for implementing very large scale, real–time, concurrent, industrial applications
- 300 users within Ericsson, “many” elsewhere
- Several telephone switching products, hundreds of kSLOCs each
- Full support: tools, training, consulting, ...

Hewlett–Packard Event Correlation Services

- A system to handle storms of event messages in a telecommunication network
- Expert systems available, but too slow
- C implementation fast but cumbersome and difficult to maintain
- HP's solution falls in between these two — easy to maintain, yet fast enough to sustain event storms
- A virtual engine, an SML–like functional language (Event Correlation Description Language), and a GUI for designing and simulating a circuit
- Real–time under heavy loads, distributed

CPL/Kleisli

- In April 1993, a workshop organised by the US Department of Energy considered the database requirements of the Human Genome Project. An appendix of the workshop report listed twelve queries that would be difficult or impossible to answer with current database systems, because they require combining information from two or more databases in disparate formats.
- All twelve of these queries have been answered using CPL/Kleisli.
- Collection Programming Language (CPL) and implementation (Kleisli).
- CPL is a functional language and Kleisli implemented in SML.
- CPL queries transformed into SQL queries by Kleisli.
- Lazy evaluation and concurrency allow SQL computation at the

database sites and CPL processing at the query site to overlap.

BlurFit

- “models laboratory measurements of blur images obtained with a focal plane imaging array using a series of Zernike polynomials to represent the wavefront distortion in the exit pupil of an optical system.”, whatever that means
- The use of a functional technique limits the spread of nasty side effects.
- The elegant data structures available in an ML language makes the process of data entry and reporting much easier.
- “The speed of the system is roughly 70% of an equivalent written entirely in C, and twice as fast as one written in RSI/IDL, but the code is inestimably more maintainable in its functional form.”
- OCAML, about 600 SLOC + libraries (5k SLOC)
- For Windows

Education

- Scheme (HUT, MIT)
- Haskell (Chalmers)
- ML (TUT)
- Teaching material available, for example
  - MiraCalc, a tool for supporting the teaching of functional programming in Miranda
  - Several lecture notes and presentations
  - Books, like Structure and Interpretation of Computer Programs

Feedback

- Need performant, real–time runtimes
- Need a solid, full development environment
- Need libraries, GUI essential
- Don't mess with the syntax, just use `f(x, y)` instead of `f x y`
- Provide industrial level support
- The need for a Killer App
- The need for a critical mass
- The need to demonstrate the benefits of functional programming

References

- `http://cm.bell-labs.com/cm/cs/who/wadler/realworld/`
- `http://www.cs.kun.nl/fple/`