

Re: Haskell: functional languages vs Lisp

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-05/2422.html>

From: mikel (mikel_at_evins.net)

Date: 05/28/04

Date: Fri, 28 May 2004 21:05:44 GMT

Nelson Marcelino wrote:

> *I am curious to know what advantages does Lisp have over Haskell.*

It's been around a long time and its features reflect a lot of what people need and want to do with a programming language. It's especially easy to edit, and easy to make tools to support editing, refactoring, and other code-construction tasks. It supports live updates of running applications better than any other language or runtime except perhaps Smalltalk. Also along with Smalltalk, it provides exceptional support for catching and repairing errors and other deficiencies while an application is running.

> *I would like to know what people think of newer languages such as Lisp*

> *OCAML SML*

> *and how they compare to Lisp.*

I like ML generally, and especially Ocaml, a lot. I use it almost as much as I use Lisp. Basically, when I need to do something complicated that must run very fast and is best delivered as a single, self-contained command-line utility, I write it in Ocaml. For interactive things (or long-running processes with interactive monitors) I prefer to use Lisp.

It so happens that over the past few months I've conducted a sort of informal experiment, using Common Lisp, Scheme, and Ocaml to write and rewrite the same nontrivial application. The application is a command-line code-groveling tools that has to process about 600MB of data on each run and do a bunch of data-extraction and restructuring, and write a report. It's for construction of part of a product that customers pay a lot of money for, so it has to be reliable and good, but I have a lot of freedom to build it any way I like (as long as it keeps doing its job quickly and reliably). Without any particular agenda (I tried all three languages because they're the ones I like most), I have evolved a process that involves prototyping new features in Common Lisp and then delivering them in Ocaml. I find it easier to explore the problem space interactively in Lisp, and easier to deliver a program whose source is concise, and whose executable is small, simple, and fast in Ocaml. It's pretty easy to translate ideas from Lisp to Ocaml.