Dr. Gergő Érdi — Curriculum Vitæ

Personal information

Name:	Dr. Érdi, Gergő
Location:	Singapore
Telephone:	+65 9186-8336
E-mail:	gergo@erdi.hu
Home page:	http://gergo.erdi.hu/

Education

2003–2011	M.Sc. in Computer Science, Eötvös Loránd University, Hungary. Master's thesis: A compositional type checker for (most of) Haskell 98
1999–2005	M.D., Semmelweis University of Medicine, Hungary

Previous work

2014–	Quantitative developer, Standard Chartered Bank. CVA pricing of vanilla and exotic financial instruments via compiling a declarative contract description to Monte-Carlo pricers
2011–2014	Strat developer, Standard Chartered Bank. Rapid development of end-user GUI applications for traders and financial structurers, including building the infrastructure for delivering applications over the web
2008–2010	Software developer, Emacs integration of RefactorErl, a refactoring tool for Erlang: Emacs Lisp project, Eötvös Loránd University
2005–2011	Software developer, Intentional Software Corporation, Domain Workbench/Structural Editor. Research and implementation of a DSL workbench for language-oriented development
1999–2005	Software developer, various Free Software/Open Source projects including the GNOME desktop environment, Evolution groupware suite, Guikachu GUI development tool and the GTKmm C/C++ language interoperability bridging solution

Functional and formal programming methods

Functional languages at SCB	At Standard Chartered Bank, we use a software stack written in part in Haskell, and exposed to developers of end-user applications as a scripting language that is an in-house, strict dialect of Haskell called Mu. My day-to-day work is 95+% development of software in Mu.
Glasgow Haskell Compiler	Added the new language feature of pattern synonyms to GHC. Implementing this cross-cutting feature required changes to the type checker and the code generator parts of GHC.
Dependent types	Familiar with Agda and Idris. Contributed to the Agda standard library. Gave a talk on Agda at a Singapore Functional Programmers' Meetup.

Compilers & DSLs

CVA on financial contracts	At Standard Chartered, I implemented a compiler for our declarative DSL of financial contracts, to implement CVA (credit valuation adjustment) calculation *à la* Longstaff & Schwartz, on top of our in-house Monte-Carlo engine. This enables computing CVA on all kinds of vanilla and exotic financial derivatives without having to write any per-instrument ad-hoc code. Written in Haskell and Mu.
Haskell to Javascript	To turn the existing and growing codebase of in-house GUI applications into web applications, I wrote a new backend for the compiler of our Haskell dialect that emits Javascript. With some hand-written Javascript runtime for building and manipulating DOM trees, we managed to compile the same code base to native Windows applications and web applications with the editing logic running on the client side and expensive calculations (like pricing of trades) on the server side; runtime written in Javascript, compiler written in Haskell
MetaFun	Compiler for a Haskell-like functional language into C++ compile-time template metaprograms; written in Haskell
Alef	Type checker, interpreter and compiler for a lazy functional language with Hindley-Milner type system; written in Common Lisp
Financial desktop applications	Development of interactive GUI applications (Windows and web front-ends) for traders and financial structurers. Highlights include developing a DSL for high-level description of structured product editors and creating a domain-specific spreadsheet editor with bi-directional affine constraints.
Intentional Domain Workbench	Structural editor tool to create schema, editable projections, a type system, semantic validators, and compilers for domain-specific languages (DSLs). Large-scale .NET project written in C♯ and in-house languages (some of them functional).

Selected blogposts

- *Static analysis with Applicatives*
- *Arrow's place in the Applicative/Monad hierarchy*
- *Typed embedding of STLC into Haskell*
- *Write Yourself a Haskell... in Lisp*
- *Mod-n counters in Agda*
- *The case for compositional type checking*
- *Initial version of my Commodore PET*