

# Dr. Gergő Érdi — Curriculum Vitæ

## Personal information

---

*Name:* Dr. Érdi, Gergő  
*Address:* Nagyszombat u. 1/C I/4, Budapest H-1036, Hungary  
*Telephone:* +36 (30) 9319-447  
*E-mail:* [gergo@erdi.hu](mailto:gergo@erdi.hu)  
*Home page:* <http://gergo.erdi.hu/>



## Education

---

2003–2011 M.Sc. in Computer Science, Eötvös Loránd University, Hungary  
1996–2006 M.D., Semmelweis University of Medicine, Hungary

## Previous work

---

2005– Software developer, Domain Workbench/Structural Editor, [Intentional Software Corporation](#)  
2008–2010 Software developer, Emacs integration of [RefactorErl](#), a refactoring tool for Erlang: Emacs Lisp project, Eötvös Loránd University  
1999–2005 Software developer, various open source projects, mostly related to the [GNOME](#) desktop environment

## Functional programming

---

*Alef* Type checker, interpreter (using graph rewriting) and compiler for a lazy functional language with Hindley-Milner type system. Written in Common Lisp.  
*Tandoori* Compositional type checker for (a subset of) Haskell 98, supporting type class polymorphism. Written in Haskell. M.Sc. thesis at Eötvös Loránd University.  
*MetaFun* Compiler for a Haskell-like functional language into C++ compile-time template metaprograms. Written in Haskell.

## Desktop applications

---

*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).  
*Guikachu* High-level graphical editor for the RCP resource description language of the PalmOS handheld platform. Large-scale C++ project.  
*GNOME desktop* Contributions to open source projects including the [Evolution](#) groupware suite and the [Gnumeric](#) spreadsheet.

## Compiling and bridging

---

*MetaFun* (see above, in *Functional programming*)

*GTKmm* C++ wrapper around the C API of the GTK+ GUI toolkit. GTKmm presents the developer with a native C++ interface, using features of C++ such as class inheritance and templates to maximize productivity, and using compile-time type safety to help applications become more robust.

## Programming languages

---

- Functional: Haskell, Lisp, XSLT
- Imperative/Object-oriented: C/C++, C#/Java, Python, CLOS
- Formal methods: Rudimentary Agda and Atelier B

## Languages

---

- Hungarian: Mother tongue
- English: Fluent
- German: Intermediate