# Hiromi Ishii

*Curriculum Vitae*

## Education

2016–2019 **Ph.D. in Math**, *University of Tsukuba*, Tsukuba-city, Ibaraki, Japan.
*Research Area*: Mathematical Logic (Set Theory and Formal Logic), Computer Algebra, and Functional Programming.
Thesis: *Bidirectional Interplay between Mathematics and Computer Science: Safety and Extensibility in Computer Algebra and Haskell* [3].

2014–2016 **M.S. in Math**, *University of Tsukuba*, Tsukuba-city, Ibaraki, Japan.
Thesis: *On Regularity Properties of Sets of Reals and Inaccessible Cardinals* [5].

2010–2014 **B.A. in Math**, *Waseda University*, Shinjuku-ku, Tokyo, Japan, Summa cum laude.

## Academic and Vocational Carrer

2024–present **Full-time Researcher and Developer**, *Jij, Inc.*, Tokyo, Japan

✱ *Born: 1992-01-18*
*konn.jinro@gmail.com* • *https://konn-san.com*
*konn* • *Haskell Discourse: @konn*

2019–Oct 2024 **Full-time Researcher and Developer**, *DeepFlow, Inc.*, Tokyo, Japan

Developped the massively-parallel numerical simulation software in Haskell from the ground up. Also designed and implemented simulation workflow tools for private company and national institutes. Main achievements:

- Dependently-typed EDSL for massively-parallel solver in Haskell
  - Designed both for equations and solver/plugin abstractions.
  - Custom anonymous and extensible record library, which can be labelled with any kind and has a dedicated type-checker plugin to solve type-level constraints.
  - For details, see my slides "Pragmatic Haskell in Large-Scale Numerical Computation" [6] and some of techniques developed is described in my preprint "Witness Me" [4].
- Haskell binding to Gmsh, an open-source mesher library.
  - Using Higher-Kinded Datatypes to provide a type-safe interface for element extrusion.
- Simulation GitOps workflow tools for industrial and academic institutes.
  - Executes simulation in parallel either by colaborating with pre-installed job queue or customly implemented scheduler.
  - Each indivisual job is specified by a Shake Build rule.
  - Jobs can be scheduled via GitHub Pull Request, and the history can be tracked there.
  - Results will be visualised by ParaView and HTML report is hosted either on local storage or R2.

2019–2020 **Visiting Researcher**, *Institute of Statistical Mathematics*, Tokyo, Japan

Worked on

2018–2019 **Part-time Developer**, *Clear Code, Inc.*, Tokyo, Japan.

2014 **Google Summer of Code 2014**

*Theme*: An Efficient Computational Algebra and Symbolic Linear Algebra Library in Haskell

2010–2014 **Internship student and Part-Time Developer**, *Preferred Infrastructure, Inc.*, Tokyo, Japan

## Fellowships, Awards, and Special Note

2017–2019 **Research Fellowship for Young Scientists (DC2)**, *Japan Society for the Promotion of Science.*

*Theme*: Properties of sets of reals and its applications to Computer

2016 **14th Meikei Prize**, *University of Tsukuba*, Tsukuba-city, Ibaraki prefecture, Tokyo

2014 **Highest Award of Deans' Prize of School of Fundamental Science and Engineering**, *Waseda University*, Tokyo, Japan

✱ *Born: 1992-01-18*
*konn.jinro@gmail.com* • *https://konn-san.com*
*konn* • *Haskell Discourse: @konn*

## Natural Languages

| | | |
|---|---|---|
| Japanese | Native | |
| English | Fluent | *(Expired Record: TOEIC 920 (2012))* |

## Programming Languages

| | | |
|---|---|---|
| Haskell | Expert | *Having been using for more than 15 years, and used as the main language in production for 5 years.* |
| Python | Moderate | *Using in production for Machine Learning.* |
| Agda | Hobbyist | *Can write easy algebraic proofs, and have been played around with Cubical Type Theory for a while.* |
| JavaScript | Moderate | *Used for automation and via WASM FFI* |
| TypeScript | Newbie | *Used for automation.* |
| Rust | Newbie | *Very interested in.* |
| C | Undergraduate level | *Mainly called via C FFI from Haskell.* |

## Technical Skills

I have been using advanced type-level machineries of GHC, including GADTs, Data Kinds, Type Families, Linear Types, amongst others. Especially, I have expertise in desigining statically/dependently-tyepd EDLs in Haskell for scientific and/or mathematical computations. In doing so, I have developed several libraries and type-checker plugins for GHC, and have been using in production.

To implement efficient applications, I have been using the following technologies for many years:

- Concurrency: Threads, STM, and `async`-like libraries.
- Distributed Computation: OpenMPI and Cloud Haskell.
- Streaming: Currently, I am using `streaming` family for streaming processing, but have a lengthy experience with `conduit` formerly.
- Linear Types: mainly used for array allocation and pure mutation.
- FFI: having been using to implement binding libraries. Both via GHC's FFI functionality and/or `inline-c`.

I am using `rio` and/or `effectful` to build enterprise-level applications.

I have been using ParaView and its Python interface for scientific visualization.

## Selected Open Source Contributions

Any other contributions can be found on my GitHub profile: `https://github.com/konn`.

Guardian - a dependency constraint enforcer for Haskell monorepo

URL: `https://github.com/deepflowinc/guardian#readme`

Guardian is a tool to enforce dependency constraints on a Haskell monorepo. When maitaining a huge monorepo, it is crucial to keep decoupling between package. Otherwise, small changes in one package can cause an essentially redundant recompilation. This tool provides a way to group packages into several domains, and specify which group can depend on which. In this way, one can prevent bad dependency and keep the entire repository loosely coupled.
The tool is developed as part of my work at DeepFlow, Inc. and now open-sourced.
For details, see the README on GitHub: `https://github.com/deepflowinc/guardian#readme`.

### Haskell on Cloudflare Worker Library

URL: `https://github.com/konn/ghc-wasm-earthly`
Using the recent WASM backend of GHC, I have developed a library to run Haskell on Cloudflare Worker. It consists of the following:

- `ghc-wasm-jsobjects`: A library to simulate JavaScript Object System in Haskell, wrapping `JSVal` inside.
- `webidl-codegen-wasm`: A library to generate Haskell FFI bindings for APIs defined by WebIDL.
- `cloudflare-worker`: A raw binding to Cloudflare Worker API.

I am currently interested in extending servant to run on Cloudflare Worker and usage of Linear Types to ensure the safety of the code.
For details, see my post [7] on Discourse.

### Type-Level Programming Utilities

I have been developping the several type-level programming utilities for Haskell and many of them are used in production. Here is the table:

- `ghc-typelits-presburger`: A GHC plugin to solve Presburger Arithmetic (i.e. theory of integer programming with quantifiers) at type-level. This can solve linear (in-)euqalities at type-level.
- `type-natural`: A library providing singletons for type-level naturals and proof of their properties.

### Missing Utilities for Linear Haskell

URL: `https://github.com/konn/linear-extra`
This monorepo includes missing utilities for Linear Haskell, which are mainly intended to be used with linear-base. It includes reference counting, token-base resource management as a poor man's alternative of linear constraints, generic vector builder, and so on.

### Some contributions to Haskell Language Server

I am one of the contributors of Haskell Language Server. I contributed the following features:

- *Splice Plugin* to expand Template Haskell splices in-place.

- *Import Disambiguation* mechanism to hide conflicting imports.
- Improvements to *Eval Plugin*, including the refactoring of its parser and adding `:type` and `:kind` commands.

Computer Algebra System `computational-algebra`

Web Site: `https://konn-san.com/computational-algebra`

**Keywords**: *Google Summer of Code 2014, CASC 2018, Computer Algebra, Type-level programming, Parallel Computation, Formal Method.*

A safe and reliable computational algebra system implemented as an EDSL in Haskell, powered by the progressive type-system of Glasgow Haskell Compiler. It enjoys the following advantages:

**Type-Safety** Encoding fields and arity as a type, it prevents wrong operations;

**Extensibility** Providing type-classes for polynomials and algebraic systems, users can enjoy various algorithms regardless to implementation detail;

**Intuitive** The library automatically gives the injection mapping between two distinct rings such as $\mathbb{Q}[x, y, z]$ and $\mathbb{Q}[x, z, y, w]$, based on their types. One can also write polynomials much alike to ordinary Math, for example: `#x ^ 2 + #z * #x - 2`.

**Static Verification** Using QuickCheck, correctness of algorithms are statically verified.

In *Google Summer of Code 2014*, I tried to implement two algorithms $F_4$ and $F_5$, which are known to be the fastest state-of-the-art algorithms for Gröbner basis computation. To that end, I implemented a framework to treat multiple matrix libraries uniformly. Also, other algorithms, such as polynomial factorisation and algebraic real computation, are provided. Large part of these implementation is verified with the method of property-based testing.

To achieve type-safety in a full spectrum, I implemented two libraries: the `type-natural` package and the `ghc-typelits-presburger` compiler plugin. The former provides the abstraction over various type-level naturals, and the latter augments the GHC's type-checker with Presburger Arithmetic solver. Based on them, I implemented the `sized` package, which provides a functionality to turn arbitrary sequential data-types into fixed-size container, indexed with type-level naturals, used for efficient representation of monomials.

As a teaching assistant, I supervised the graduation work using this library in Mathematics Department in University of Tsukuba. For more details, see the recent paper [1].

## Publications

Amongst others, I coauthored the paper "Freer Monads, more extensible effects" [9] with O. Kiselyov.

[1] Hiromi Ishii. "A Purely Functional Computer Algebra System

Embedded in Haskell". In: *Computer Algebra in Scientific Computing* (Lille, France). Ed. by Vladimir P. Gerdt, Wolfram Koepf, and Werner M. Seiler. Vol. 11077. Lecture Notes in Computer Science. Springer, Cham, 2018, pp. 288–303. ISBN: 978-3-319-99638-7. DOI: 10.1007/978-3-319-99639-4_20. arXiv: 1807.01456.

[2] Hiromi Ishii. "Automatic Differentiation With Higher Infinitesimals, or Computational Smooth Infinitesimal Analysis in Weil Algebra". In: *Computer Algebra in Scientific Computing* (Sochi, Russia). Ed. by François Boulier et al. Vol. 11077. Lecture Notes in Computer Science. Springer, Cham, 2021, pp. 174–191. ISBN: 978-3-030-85164-4. DOI: 10.1007/978-3-030-85165-1_11. arXiv: 2106.14153.

[3] Hiromi Ishii. "Bidirectional Interplay between Mathematics and Computer Science: Safety and Extensibility in Computer Algebra and Haskell". PhD thesis. University of Tsukuba, 2019. DOI: 10.15068/00156548.

[4] Hiromi Ishii. *Functional Pearl: Witness Me – Constructive Arguments Must Be Guided with Concrete Witness*. preprint. 2021. DOI: 10.48550/arXiv.2103.11751. URL: https://arxiv.org/abs/2103.11751.

[5] Hiromi Ishii. "On Regularity Properties of Set of Reals and Inaccessible Cardinals". MA thesis. Tsukuba University, 2016. URL: http://hdl.handle.net/2241/00135591.

[6] Hiromi Ishii. *Pragmatic Haskell in Large-Scale Numerical Computation*. Talk at Haskell Day 2019, Tokyo, Japan. 2019. URL: https://speakerdeck.com/konn/da-gui-mo-shu-zhi-ji-suan-wozhi-eru-haskell-nil-nil-pragmatic-haskell-in-large-scale-numerical-computation-nil-nil.

[7] Hiromi Ishii. *Serverless Haskell with GHC WASM JSFFI + Cloudflare Workers*. 2024. URL: https://discourse.haskell.org/t/serverless-haskell-with-ghc-wasm-jsffi-cloudflare-workers.

[8] Hiromi Ishii. *Haskell Pragmatic Haskell in Large-Scale Numerical Computation*. Talk at Haskell Day 2019, Tokyo, Japan. 2019. URL: https://speakerdeck.com/konn/da-gui-mo-shu-zhi-ji-suan-wozhi-eru-haskell-nil-nil-pragmatic-haskell-in-large-scale-numerical-computation-nil-nil.

[9] Oleg Kiselyov and Hiromi Ishii. “Freer Monads, More Extensible Effects”. In: *Proceedings of the 2015 ACM SIGPLAN Symposium on Haskell*. Haskell '15. Vancouver, BC, Canada: ACM, 2015, pp. 94–105. ISBN: 978-1-4503-3808-0. DOI: 10.1145/2804302.2804319.

[10] konn. *2023 Linear Haskell FFT Haskell Rust* . 2023. URL: https://zenn.dev/konn/articles/2023-12-14-pure-parallel-fft-in-linear-haskell.

[11] konn. *ghc-typelits-presburger: Presburger Arithmetic Solver for GHC Type-level natural numbers*. 2015-2024. URL: https://hackage.haskell.org/package/ghc-typelits-presburger.

[12] konn. *Guardian Haskell* . 2021. URL: https://zenn.dev/deepflow_tech/articles/secure-haskell-monorepo-deps-with-guardian.

[13] konn. *Haskell Rust 2023 Linear Haskell* . 2023. URL: https://zenn.dev/konn/articles/2023-10-01-linear-haskell-in-2023.

[14] konn. *Serverless Haskell - GHC WASM Haskell Cloudflare Workers* . 2021. URL: https://zenn.dev/konn/articles/2024-06-22-serverless-haskell.

[15] konn. *sized: Sized sequence data-types*. 2014-2024. URL: https://hackage.haskell.org/package/sized.

[16] konn. *type-natural: Type-level natural and proofs of their properties*. 2013-2024. URL: https://hackage.haskell.org/package/type-natural.