**Ki Yung Ahn** Research Fellow in the School of Computer Science & Engineering at Nanyang Technological University, Singapore

## Contact Information

Cyber Security Lab (Blk. N4-B2C-06)
50 Nanyang Avenue
Singapore 639798

kyagrd@gmail.com
+65 8589 1226
http://kyagrd.github.io/

## Research Interests

Security protocol verification using process calcului,
Executable relational specifications of polymorphic type systems using logic programming,
Language design to support both convenient programming and logically consistent reasoning via the Curry–Howard correspondence,
Extending the Hindley–Milner (HM) type inference for languages with Mendler-style recursion schemes and GADTs with true term indices, and
Interfacing with solvers (e.g, SAT, SMT) in automated testing/verification frameworks.

## Education

**Portland State University**, Portland, OR, USA
Ph.D. (Advisor: Tim Sheard), Computer Science, Dec 2014
*The Nax language: unifying functional programming and logical reasoning in a language based on Mendler-style recursion schemes and term-indexed types*

**KAIST**, Daejeon, Republic of Korea
B.S., Computer Science (major) and Mathematics (sub-major), Mar 2002

## Research Experience and Academic Visits

**Research Fellow** (current position) Jul 2016 - current
School of Computer Science & Engineering, Nanyang Technological University, Singapore
PI: Alwen Tiu (Assistant Professor)

**Gratuitous Visit** (Talk info: http://talks.cam.ac.uk/talk/index/60589) Sep 2015
Programming Principles and Tools group, Microsoft Research, Cambridge, UK
Host: Claudio Russo (Senior Research Software Development Engineer)

**Academic Visit** (Talk info: http://slides.com/kyagrd/tiperdundee) Aug 2015
Programming Languages, Semantics and Logic group, University of Dundee, UK
Host: Ekaterina Komendantskaya (Reader)

**Visiting Student** [1] (Talk info: http://talks.cam.ac.uk/talk/index/33917) Sep - Dec 2011
Computer Laboratory, University of Cambridge, Cambridge, UK
Hosts: Andrew M. Pitts (Professor), Marcelo Fiore (Professor)

**NASA Ames MCT Internship** [2, 3] Jun - Sep 2009
Mission Control Technologies at NASA Ames Research Center, CA, USA
Supervisor: Ewen Denney (Senior Computer Scientist)

**Research Assistant (Graduate Student)** [4, 5, 6, 7] Sep 2007 - Sep 2013

## Software Projects

TIPER - Type Inference Prototyping Engines from Executable Relational specifcations
https://github.com/kyagrd/tiper

Mininax - prototype implementation of the Nax language
https://github.com/kyagrd/mininax

Haskell programming interface to the Yices SMT solver (through UNIX pipe)
http://hackage.haskell.org/package/yices

## Awards

Bronze medal in the ACM Student Research Competition (SRC) at ICFP 2012

## Industry Experience

**Formal Verification Software Engineer (Intern)** Sep 2013 - Mar 2014
Refactored parts of the Forte system libraries written in FL (a reflective functional language for HW design and theorem proving) and also implemented specification search by using term rewriting
Formal Verification Center of Expertise (DTS/FVCoE), Intel, Hillsboro, OR, USA
Supervisors: John W. O'Leary, Roope Kaivola (Principal Engineers)

**Quantitative Summer Institute (QSI) Associate (Intern)** Jun - Aug 2008
Global Modelling and Analytics Group, Credit Suisse, New York, NY, USA
Supervisor: Howard Mansell (Quantitative Strategist)

**Internet Storage Service Server Developer** Mar 2002 - May 2005
PopFolder: revenue over 10 million USD, over a million users in 2002
Gretech, Seoul, Republic of Korea
Supervisor: Keunho Bae (Director) Skills: C/C++, TCP/IP, UNIX, Berkley DB, PostgreSQL

## Teaching Experience

**Full-time Lecturer** Spring 2016
Electronics and Information Engineering, Korea University, Sejong City, Korea

- EIEN233(02) Data Structures (lecture in Korean)
- EIEN363(03) Computer Architecture (lecture in Korean)
- EIEN215(02) Engineering Mathematics I (lecture in English)

**Teaching Assistant** Spring and Summer 2007
CS 106: Computing Fundamentals II (Intro. to programming for non-CS majors)
Computer Science, Portland State University, Portland, OR, USA
Supervisor: Cynthia A. Brown (Emerita Professor)

## Translations

Korean translation (ISBN 9788972808183) of *Programming in Haskell* (ISBN 9780521692694) by Graham Hutton

## Reviewer (Referee)

Typed Lambda Calculus and Applications 2015
Trends in Functional Programming 2013
Higher-Order and Symbolic Computation (special issue for PEPM 2012)
NASA Formal Methods Symposium 2011

## Talks

A Prolog Specification of Extensible Records using Row Polymorphism (invited talk)
Workshop on Coalgebra, Horn Clause Logic Programming and Types (CoALP-Ty '16), Edinburgh, UK, 28-29 November 2016

## Publications


[1] Ki Yung Ahn, Tim Sheard, Marcelo Fiore, and Andrew M. Pitts. System $F_i$: a higher-order polymorphic $\lambda$-calculus with erasable term indices. In *Proceedings of the 11th international conference on Typed Lambda Calculi and Applications*. TLCA '13, volume 7941 of *LNCS*. Springer, 2013. doi:10.1007/978-3-642-38946-7_4.

[2] Ki Yung Ahn and Ewen Denney. A framework for testing first-order logic axioms in program verification. *Software Quality Journal*, 21(1):159–200, March 2013. ISSN 0963-9314. doi:10.1007/s11219-011-9168-1.

[3] Ki Yung Ahn and Ewen Denney. Testing first-order logic axioms in program verification. In *Proceedings of the 4th international conference on Tests and Proofs*, TAP'10, pages 22–37. Springer-Verlag, 2010. ISBN 3-642-13976-0, 978-3-642-13976-5.

[4] Ki Yung Ahn and Tim Sheard. A hierarchy of mendler style recursion combinators: taming inductive datatypes with negative occurrences. In *Proceedings of the 16th ACM SIGPLAN international conference on Functional programming*, ICFP '11, pages 234–246, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0865-6. doi:10.1145/2034773.2034807.

[5] Ki Yung Ahn and Tim Sheard. Shared subtypes: subtyping recursive parametrized algebraic data types. In *Proceedings of the first ACM SIGPLAN symposium on Haskell*, Haskell '08, pages 75–86, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-064-7. doi:10.1145/1411286.1411297.

[6] Garrin Kimmell, Aaron Stump, Harley D. Eades, III, Peng Fu, Tim Sheard, Stephanie Weirich, Chris Casinghino, Vilhelm Sjöberg, Nathan Collins, and Ki Yung Ahn. Equational reasoning about programs with general recursion and call-by-value semantics. In *Proceedings of the sixth workshop on Programming languages meets program verification*, PLPV '12, pages 15–26, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1125-0. doi:10.1145/2103776.2103780.

[7] Vilhelm Sjöberg, Chris Casinghino, Ki Yung Ahn, Nathan Collins, Harley D. Eades III, Peng Fu, Garrin Kimmell, Tim Sheard, Aaron Stump, and Stephanie Weirich. Irrelevance, heterogeneous equality, and call-by-value dependent type systems. In *MSFP*, pages 112–162, 2012. doi:10.4204/EPTCS.76.9.

[8] Ki Yung Ahn. *The Nax Language: Unifying Functional Programming and Logical Reasoning in a Language based on Mendler-style Recursion Schemes and Term-indexed Types*. PhD thesis, Portland State University, 2014. Dissertations and Theses. Paper 2088. http://pdxscholar.library.pdx.edu/open_access_etds/2088.

[9] Ki Yung Ahn and Andrea Vezzosi. Executable relational specifications of polymorphic type systems using Prolog. In *Proceedings of the 13th International Symposium on Functional and Logic Programming*, volume 9613 of *LNCS*, pages 109–125. Springer, March 2016. Draft available at https://www.sharelatex.com/project/557756cfdfb75ebd54bf5807.

## Upcoming Papers

[u1] Mendler-style recursion schemes for mixed-variant datatypes
Ki Yung Ahn, Tim Sheard, and Marcelo Fiore.
(slides in TYPES 2013 talk, draft)

[u2] An executable relational specification of extensible records using logic programming
Ki Yung Ahn and others. (An early draft plan available online)

## Summary of my Research Contributions

The contributions of my reseach can be categorized into three areas:

**Executable relational specifications of polymorhpic type systems.**

I (in collaboration with Vezzosi) provided an executable specification [9] of an advanced type system that supports parametric polymorphism in several dimensions including type polymorphism, type constructor polymorphism, and kind polymorphism (all in rank-1, for type inference). The specification is succinct, declarative, and it also servers as a reference implementation for type inference — key features of Nax including pattern matching and Mendler-style iteration could be specified in less than 100 lines of Prolog. The specification is not only short but also highly readable because of its structural resemblance to the typing rules typically used for describing type systems on

paper. In addition, this specification can execute queries of type inference by exploiting backward-chaining of Prolog as well as checking types. Recently, I was able to identify key ingredients to specify extensible records [u2].

I have initiated a project called TIPER in order to realize the idea from this line of research (i.e., logic programming for executable specification of type systems) into a practical tool that automates type system implementation in compiler construction and plug-ins for development tools. We already have tools (e.g. Yacc) for automatically generating parsers from declarative grammar rule specifications but in lack of tools for automatically generating type system implementations supporting polymorphic type inference from typing rule specifications.

**Mendler-style recursion schemes for mixed-variant datatypes.**

My contribution in Mendler-style recursion schemes for mixed-variant datatypes provides the basis for safely supporting mixed-variant datatype declarations in the Nax language and future language designs to be inspired by Nax. Mainstream dependently typed proof assistants disallow mixed-variant datatype declarations.

I (in collaboration with Sheard) discovered a Mendler-style version [4] of the catamorphism over datatypes with contravariant recursive occurrences[1] (Fegaras and Sheard 2009). Discovering Mendler-style counterparts is more than mere reformulation of already known recursion schemes. Unlike conventional recursion schemes, Mendler-style recursion schemes naturally generalize non-regular datatypes including nested datatypes and GADTs. In addition, previous studies on Mendler-style recursion schemes were mostly focused on datatypes with covariant recursive occurrences.

Recently, I (in collaboration with Fiore and Sheard) formulated a Mendler-style version [u1] of the parametric[2] variant of the (non-Mendler-style) Sheard–Fegaras catamorphsim (Bahr and Hvitved 2012). However, investigations on the termination properties of this new recursion scheme are still left for future work.

**Typed lambda calculi with erasable term indices.**

I (in collaboration with Sheard, Pitts, and Fiore) provided a baseline understanding of generalized algebraic datatypes (GADTs) with real term indices and their associated Mendler-style recursion schemes, by formulating System $\mathrm{F}_i$ [1], which extends System $\mathrm{F}_\omega$ with erasable term indices. System $\mathrm{F}_i$ is carefully designed to inherit the desirable properties of System $\mathrm{F}_\omega$ including type preservation and logical consistency. In my dissertation [8], I also show that the same extension can be applied to System $\mathrm{Fix}_\omega$ (Abel and Matthes 2004) designed for embedding the Mendler-style primitive recursion.

Polymorphic lambda calculi provide us a baseline understanding of the type systems of functional languages. For example, we know that extensions to HM such as higher-rank types will not break properties proven to hold in System F because higher-rank polymorphism is already part of System F. Datatypes and pattern matching can also be understood within a pure polymorphic lambda calculus via the Böhm-Berarducci encoding. Functional languages with more advanced type systems that support type constructor polymorphism (a.k.a., higher-kinded polymorphism) could be understood in terms of System $\mathrm{F}_\omega$. Due to my contribution of formulating System $\mathrm{F}_i$, our baseline understanding now extends over GADTs with real term indicies.

[1] Datatypes with both covariant and contravariant recursive occurrences are also known as *mixed-variant datatypes*.

[2] As in *parametric* higher-order absract syntax (PHOAS).

In addition to my contributions in the three major areas mentioned above, I have several other research interests and experiences including the following items:

- During my research internship at NASA Ames, I used QuickCheck to generate test cases for the first-order axioms for proving automatically generated verification conditions [3, 2]. The main challenge was to cut down the vacuously satisfiable tests caused by the variables inside the premise (i.e., left-hand side of the implication).
- I participated in the Trellys research group [6, 7], which was an inter-institutional project funded by the NSF in the USA. The goal of the project was to develop a dependently typed language that can be used for both logical reasoning and programming.