Stephanie **Weirich**

School of Engineering and Science, University of Pennsylvania
Levine 510, 3330 Walnut St, Philadelphia, PA 19104
215-573-2821 • sweirich@seas.upenn.edu • August 25, 2025

Academic Positions

University of Pennsylvania — **Philadelphia, Pennsylvania**
ENIAC President's Distinguished Professor — *September 2019-present*

University of Pennsylvania — **Philadelphia, Pennsylvania**
Professor — *July 2015-August 2019*

University of Pennsylvania — **Philadelphia, Pennsylvania**
Associate Professor — *July 2008-June 2015*

University of Cambridge — **Cambridge, UK**
Visitor — *August 2009-July 2010*

University of Pennsylvania — **Philadelphia, Pennsylvania**
Assistant Professor — *July 2002-July 2008*

Cornell University — **Ithaca, New York**
Instructor, Research Assistant and Teaching Assistant — *August 1996-July 2002*

Industry Positions

Epic Games — **Philadelphia, Pennsylvania**
Principal Language Engineer — *March 2022-September 2023*

Galois, Inc — **Portland, Oregon**
Visiting Scientist — *June 2018-August 2019*

Microsoft Research — **Cambridge, UK**
Visiting Researcher — *September-November 2009*

Lucent Technologies — **Murray Hill, New Jersey**
Graduate Intern — *June-July 1999*

Education

Cornell University — **Ithaca, NY**
Ph.D., Computer Science — *2002*

Cornell University — **Ithaca, NY**
M.S., Computer Science — *2000*

Rice University — **Houston, TX**
B.A., Computer Science, magnum cum laude — *1996*

Honors

- Milner Lecture, University of Edinburgh, 2022
- SIGPLAN Robin Milner Young Researcher award, 2016
- Most Influential ICFP 2006 Paper, awarded in 2016
- Microsoft Outstanding Collaborator, 2016
- Penn Engineering Fellow, University of Pennsylvania, 2014
- Institute for Defense Analyses Computer Science Study Panel, 2007
- National Science Foundation CAREER Award, 2003
- Intel Graduate Student Fellowship, 2000–2001
- National Science Foundation Graduate Research Fellowship, 1996–1999
- CRA-W Distributed Mentorship Project Award, 1996
- Microsoft Technical Scholar, 1995–1996

Students

Postdoc supervision

- Joachim Breitner, Aug 2016-2018.

Dissertation supervision (current students)

- Yiyun Liu, Aug 2021-
- Jonathan Chan, Aug 2022-
- Cassia Torczon (co-advised with Benjamin Pierce), Aug 2022-
- Daniel Sainati (co-advised with Benjamin Pierce), Aug 2024-

Dissertation supervision (graduated students)

- Pritam Choudhury, 2023. *Dependency and Linearity Analyses in Pure Type Systems.*
- Yao Li, August 2022. *Mechanized Reasoning About "How" Using Functional Programs and Embeddings.* Current employment: Portland State University
- Antal Spector-Zabusky, May 2021. *Don't mind the formalization gap: The Design and Usage of hs-to-coq.*
- Richard Eisenberg, December 2016. *Dependently-Typed Haskell.* Current position: Jane Street. *Co-winner of Morris and Dorothy Rubinoff award.*
- Vilhelm Sjöberg, May 2015. *A Dependently Typed Language with Nontermination.* Current position: CertiK. *Co-winner of SIGPLAN 2016 John C. Reynolds Doctoral Dissertation award.*
- Brent Yorgey, December 2014. *Combinatorial Species and Labelled Structures.* Current position: Associate Professor, Hendrix College.
- Chris Casinghino, December 2014. *Combining Proofs and Programs.* Current position: Jane Street.
- Dimitrios Vytiniotis, August 2008. *Practical type inference for first-class polymorphism.* Current position: Google DeepMind, London, UK.
- Geoffrey Washburn, December 2007. *Principia Narcissus: How to avoid being caught by your reflection.* Current position: Amazon Redshift.

Dissertation committee member (external)

- Peio Borthelle, Université Savoie Mont Blanc in Chambéry, France. March 2025.
- Riccardo Bianchini, Univ. di Genova, April 2024.

- Steven Keuchel, University of Ghent, June 2018.
- William Bowman, Northeastern University, November 2018.
- Harley Eades III, University of Iowa, May 2014.
- Julien Cretin, INRIA / University Paris 7, January 2014.
- Jean-Philippe Bernardy, Chalmers ("Faculty Opponent"), Gothenburg, Sweden, June 2011.
- Boris Yakobowski, INRIA / University Paris 7, December 2008.
- Dan Dantas, Princeton University, August 2007.
- Joeseph Vanderwaart, Carnegie Mellon University, August 2006.

Dissertation committee member (Penn)

- Joe Cutler, Penn.
- Calvin Beck, Penn.
- Lef Ioannidis, Penn, June 2025.
- Nicholas Rioux, Penn, June 2025.
- Lawrence Dunn, Penn, April 2025.
- Paul He, Penn, July 2024.
- Harrison Goldstein, Penn. May 2024.
- Lucas Silver, Penn, July 2023.
- Irene Yoon, Penn, November 2023.
- Li-Yao Xia, Penn, June 2022.
- Teng Zheng, Penn, October 2021.
- Robert Rand, Penn, November 2018.
- Jennifer Paykin, Penn, May 2018.
- Leonidas Lampropolos, Penn, May 2018.
- Arthur Azevedo de Amorim, Penn, September 2017.
- Peter Michael Osera, Penn, July 2015.
- Daniel Wagner, Penn, June 2014.
- Michael Greenberg, Penn, December 2013.
- Hongbo Zhang, Penn, Master's thesis, December 2013.
- Klara Mazurak, Penn, May 2013.
- Jianzhao Zhao, Penn, April 2013.
- Aaron Bohannon, Penn, February 2012.
- Jeffrey Vaughan, Penn, December 2009.
- Stephen Tse, Penn, August 2007.
- Wahnhong Nam, Penn, December 2006.
- Vladimir Gapayev, Penn, January 2006.

Visiting PhD student supervison

- Pedro Henrique Azevedo de Amorim, Mar-Aug 2016.
- Antoine Voizard, École Normale Supérieure, Paris. Mar-Aug 2014.
- Steven Keuchel, University of Ghent, Sep 2013-Mar 2014.
- Arthur Charguéraud (co-supervised with Benjamin Pierce), INRIA, 2007.

Independent study

- Doctoral: Daniel Sainati, Fall 2024-Spring 2025. Noé De Santo, Fall 2024-Spring 2025. Francis Rinaldi, Fall 2024-Spring 2025. Jessica Shi, Fall 2021. Irene Yoon, Spring 2020. Hengchu Zhang, Yao Li, Spring 2017. Antoine Voizard, Kenny Foner, Fall 2015. Antal Spector-Zabusky, Spring

2016, Spring 2013. Jennifer Paykin, Fall 2012. Richard Eisenberg, Justin Hsu, Spring 2012. Richard Eisenberg, Hongbo Zhang. Fall 2011. Brent Yorgey, Peter-Michael Osera, Vilhelm Sjöberg. Fall 2008-Spring 2009. Chris Casinghino, Spring 2008. Andrew Hilton (co-advised), Karl Mazurak, Jeff Vaughan, Fall 2004. Liang Huang, Spring 2004.

- Masters: Emmanuel Suarez, Fall 2022-Spring 2024. Eric Giovanni, Spring 2020. Dominik Bollman, Spring 2016. Simon Wimmer, Summer 2015.
- Undergraduate Senior Design Project: Memoria Matters and Lauren Leung, 2016-2017 (Honoable Mention). Charles Du 2017, Max McCarthy 2016. Lewis Ellis, Max Scheiber, Ashutosh Goel, and Jeff Grimes (Honorable Mention). Tiernan Garsys, Taylor Mandel, Lucas Peña, and Noam Zilberstein (Third place). 2014-2015. Kaycee Anderson, Juan Jose Lopez, Caroline Ho, and Johanna Martens (Honorable Mention), 2013-2014.
- Undergraduate Research: Liz Austell, Elliot Brobow, Kevin Diggs, Apol Medrano, Elon Roth, Summer 2024 (REPL). Shubh Agrawal, Maite Kramarz, Annabel Baniak, Summer 2023 (REPL). Vikram Singh (DeepSpec REU), Summer 2021. Daniel Lee, 2021. Joshua Cohen, 2018. Emmanuel Suarez, 2017-2018. Anastasiya Kravchuk-Kirilyuk, 2017. Matthew Weaver 2015-2016. Leondra Morse, Summer 2015 (CRA-DREU). Mitchell Stern, Spring 2014. Hamidhasan Ahmed, Spring 2014, Summer 2013. Sneha Popley, Summer 2008 (CRA-DREU). Stephanie Simon, Summer 2008. David Gorski, Fall 2006. Parshant Mittal, Atish Davda, Fall 2005. Neal Parikh, Summer 2004.

Funding

Current

1. *SHF: Small: SMALL:Dependency Tracking and Dependent Types.* Weirich, NSF CCF-2327738. $540,000. 12/2023-11/2026.

2. Research Experience for Undergraduates in Programming Languages (REPL). Zdancewic (PI), Weirich, Pierce. NSF CNS-2244494 $322,095. 2023-2027.

Completed

1. Epic Games. Unrestricted gift $74,400.00. 3/2024.

2. VERSE: Verification Engineering for Real-World Software Engineers. Pierce (PI), Head, Weirich. DARPA. 2024-2025.

3. 1. *SHF: Small: Mechanized reasoning for functional programs.* Weirich, NSF CCF-2006535. $450,000, 10/2020-09/2024.

4. *SHF: Medium: Collaborative Research: The Theory and Practice of Dependent Types in Haskell.* Weirich, Eisenberg (Bryn Mawr), NSF 1703835, $814,453 (Penn), 7/2017-6/2022.

5. *Collaborative Research: Expeditions in Computing: The Science of Deep Specification.* Weirich (PI), Pierce, Zdancewic (Penn), Appel (Princeton), Shao (Yale), Chlipala (MIT). NSF 1521539, $10 million total, 1/2016-10/2021.

6. *Collaborative Research: Expeditions in Computing: The Science of Deep Specification: REU Supplement.* Weirich, Pierce, Zdancewic (Penn). $24,000, 1/2017-10/2021.

7. *CIF: Small: Rich-Type Inference for Functional Programming.* Weirich (PI). NSF 1319880, $450,000, 9/2013-8/2018.

8. *SPARCS: Synthesis of Platform-aware Attack-Resilient Control Systems* Lee (PI), Sokolsky, Pappas, Michael, Mangharam, Weirich, Alur, Tabuada. DARPA, $5.5 million total, 8/2012-8/2017.

9. *CCF-SHF Small: Beyond Algebraic Data Types: Combinatorial Species and Mathematically-Structured Programming* Weirich (PI). NSF 1218002, $325,840, 8/2012-8/2017.

10. *SHF: Small: Dependently-Typed Haskell* Weirich (PI). NSF 1116620, $496,785, 8/2011-8/2016.

11. *Student travel support for ICFP 2015.* Weirich. NSF $20,000.

12. *CIF: Small: Rich-Type Inference for Functional Programming REU* Weirich. NSF, $7,000.

13. *SHF: Small: Dependently-Typed Haskell REU* Weirich (PI). NSF 1116620, $6,000.

14. *SHF: Large: Collaborative Research: TRELLYS:Community-Based Design and Implementation of a Dependently Typed Programming Language* Weirich (Penn), Stump (University of Iowa), Sheard (Portland State University). NSF 0910786, $2.1 million total. 2009-2014.

15. *Student Travel Support for Programming languages Mentoring Workshop (PLMW 2012)* Weirich (PI). NSF $15,900, 11/2011.

16. *Networks Opposing Botnets* Smith (PI), Pierce, Zdancewic, Loo, Weirich, Felton, Rexford, Walker, Morrisett, Welsh. ONR, $400,000 (Penn), 2009-2012.

17. *Computer Science Study Panel, Phase II* Weirich (PI), Zdancewic. DARPA, $500,000, 2008-2010.

18. *Collaborative Research: CT-T: Manifest Security* University of Pennsylvania. Pierce (PI), Weirich, Zdancewic. Carnegie Mellon University. Pfenning (PI), Harper, Crary. NSF $1 million total, 2007-2011.

19. *A Practical Dependently-Typed Functional Programming Language* Weirich (PI). NSF, $200,000. 2007-2009.

20. *Computer Science Study Panel, Phase I* Weirich. DARPA, $99,411. 2007-2008.

21. *CRI: Machine Assistance for Programming Languages Research* Weirich (PI), Pierce, Zdancewic. NSF $200,000, 2006-2008.

22. *CAREER: Type-Directed Programming in Object-Oriented Languages* Weirich (PI), NSF CCF-0347289: $400,000, 2003-2008.

Refereed Publications

[1] Jonathan Chan and Stephanie Weirich. Stratified type theory. In Viktor Vafeiadis, editor, *European Symposium on Programming Languages and Systems, ESOP 2025*, May 2025. doi: 10.1007/978-3-031-91118-7_10.

[2] Yiyun Liu, Jonathan Chan, and Stephanie Weirich. Consistency of a dependent calculus of indistinguishability. *Proc. ACM Program. Lang.*, 9(POPL), January 2025.

[3] Cassia Torczon, Emmanuel Suárez Acevedo, Shubh Agrawal, Joey Velez-Ginorio, and Stephanie Weirich. Effects and coeffects in call-by-push-value. *Proc. ACM Program. Lang.*, 8 (OOPSLA), October 2024. doi: 10.1145/3689750.

[4] Li-Yao Xia, Laura Israel, Maite Kramarz, Nathan Coltharp, Koen Claessen, Stephanie Weirich, and Yao Li. Story of your lazy function's life: A bidirectional demand semantics for mechanized cost analysis of lazy programs. *Proc. ACM Program. Lang.*, 8(ICFP), August 2024. doi: 10.1145/3674626.

[5] Yiyun Liu, Jonathan Chan, Jessica Shi, and Stephanie Weirich. Internalizing indistinguishability with dependent types. *Proc. ACM Program. Lang.*, 8(POPL), January 2024. doi: 10.1145/3632886.

[6] Yiyun Liu and Stephanie Weirich. Dependently-typed programming with logical equality reflection. *Proc. ACM Program. Lang.*, 7(ICFP), August 2023. doi: 10.1145/3607852.

[7] Yao Li and Stephanie Weirich. Program adverbs and Tlön embeddings. *Proc. ACM Program. Lang.*, 6(ICFP), September 2022. doi: 10.1145/3547632. Distinguished Paper Award. Artifact available.

[8] Pritam Choudhury, Harley Eades III, and Stephanie Weirich. A dependent dependency calculus. In Ilya Sergey, editor, *Programming Languages and Systems, ESOP 2022*, volume 13240 of *Lecture Notes in Computer Science*, pages 403–430, Cham, 2022. Springer International Publishing. ISBN 978-3-030-99336-8. doi: 10.1007/978-3-030-99336-8_15. Artifact available.

[9] Richard A. Eisenberg, Guillaume Duboc, Stephanie Weirich, and Daniel Lee. An existential crisis resolved: Type inference for first-class existential types. *Proc. ACM Program. Lang.*, 5 (ICFP), August 2021. doi: 10.1145/3473569. Distinguished Paper Award.

[10] Yao Li, Li-yao Xia, and Stephanie Weirich. Reasoning about the garden of forking paths. *Proc. ACM Program. Lang.*, 5(ICFP), August 2021. doi: 10.1145/3473585.

[11] Joachim Breitner, Antal Spector-Zabusky, Yao Li, Christine Rizkallah, John Wiegley, Joshua Cohen, and Stephanie Weirich. Ready, set, verify! Applying hs-to-coq to real-world Haskell code. *Journal of Functional Programming*, 31:1–40, February 2021. doi: 10.1017/S0956796820000283.

[12] Pritam Choudhury, Harley D. Eades III, Richard A. Eisenberg, and Stephanie Weirich. A graded dependent type system with a usage-aware semantics. *Proc. ACM Program. Lang.*, 5 (POPL), January 2021. doi: 10.1145/3434331. Artifact available.

[13] Anastasiya Kravchuk-Kirilyuk, Antoine Voizard, and Stephanie Weirich. Eta-equivalence in core dependent haskell. In Marc Bezem and Assia Mahboubi, editors, *Post-proceedings of the 25th International Conference on Types for Proofs and Programs (TYPES 2019)*, volume 175 of *Leibniz International Proceedings in Informatics*, pages 7:1–7:32, Dagstuhl, Germany, sep 2020. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. doi: 10.4230/LIPIcs.TYPES.2019.7.

[14] Stephanie Weirich, Pritam Choudhury, Antoine Voizard, and Richard Eisenberg. A role for dependent types in Haskell. *Proc. ACM Program. Lang.*, 3(ICFP), 2019. doi: 10.1145/3341705.

[15] Joachim Breitner, Antal Spector-Zabusky, Yao Li, Christine Rizkallah, John Wiegley, and Stephanie Weirich. Ready, set, verify! Applying hs-to-coq to real-world Haskell code (Experience report). *Proc. ACM Program. Lang.*, 2(ICFP):89:1–89:16, July 2018. ISSN 2475-1421. doi: 10.1145/3236784. Artifact Available.

[16] Antal Spector-Zabusky, Joachim Breitner, Christine Rizkallah, and Stephanie Weirich. Total Haskell is reasonable Coq. In *Proceedings of 7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP'18)*. ACM, 2018. doi: 10.1145/3167092. New York, NY, USA.

[17] Stephanie Weirich, Antoine Voizard, Pedro Henrique Avezedo de Amorim, and Richard A. Eisenberg. A specification for dependent types in Haskell. *Proc. ACM Program. Lang.*, 1 (ICFP):31:1–31:29, August 2017. ISSN 2475-1421. doi: 10.1145/3110275. Artifact evaluated as "Available" and Reusable".

[18] Andrew W. Appel, Lennart Beringer, Adam Chlipala, Benjamin C. Pierce, Zhong Shao, Stephanie Weirich, and Steve Zdancewic. Position paper: the science of deep specification. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 375(2104), 2017. ISSN 1364-503X. doi: 10.1098/rsta.2016.0331.

[19] Steven Keuchel, Stephanie Weirich, and Thomas Tom Schrijvers. Needle and Knot: Binder boilerplate tied up. In *European Symposium on Programming (ESOP)*, pages 419–445, April 2016. doi: 10.1007/978-3-662-49498-1_17.

[20] Richard A. Eisenberg, Stephanie Weirich, and Hamidhasan G. Ahmed. Visible type application. In *European Symposium on Programming (ESOP)*, pages 229–254, April 2016. doi: 10.1007/978-3-662-49498-1_10.

[21] Joachim Breitner, Richard A. Eisenberg, Simon Peyton Jones, and Stephanie Weirich. Safe zero-cost coercions for Haskell. *Journal of Functional Programming*, 26, 2016. doi: 10.1017/S0956796816000150.

[22] Simon Peyton Jones, Stephanie Weirich, Richard A. Eisenberg, and Dimitrios Vytiniotis. A reflection on types. In Sam Lindley, Conor McBride, Phil Trinder, and Don Sannella, editors, *WadlerFest 2016: A list of successes that can change the world*, LNCS, pages 292–317. Springer, 2016. doi: 10.1007/978-3-319-30936-1_16.

[23] Wenrui Meng, Junkil Park, Oleg Sokolsky, Stephanie Weirich, and Insup Lee. Verified ros-based deployment of platform-independent control systems. In *Seventh NASA Formal Methods Symposium*, pages 248–262, Pasadena, CA, 2015.

[24] Vilhelm Sjöberg and Stephanie Weirich. Programming up to congruence. In *POPL 2015: 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 369–382, Mumbai, India, January 2015. doi: 10.1145/2676726.2676974. Artifact Evaluated.

[25] Joachim Breitner, Richard A. Eisenberg, Simon Peyton Jones, and Stephanie Weirich. Safe zero-cost coercions for haskell. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ICFP '14, page 189–202, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328739. doi: 10.1145/2628136.2628141.

[26] Richard A. Eisenberg, Dimitrios Vytiniotis, Simon Peyton Jones, and Stephanie Weirich. Closed type families with overlapping equations. In *POPL 2014: 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 671–683, San Diego, CA, USA, January 2014.

[27] Chris Casinghino, Vilhelm Sjöberg, and Stephanie Weirich. Combining proofs and programs in a dependently typed language. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, page 33–45, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325448. doi: 10.1145/2535838.2535883.

[28] Zhengjiang Hu, Shin-Cheng Mu, and Stephanie Weirich. Guest editorial: Advanced programming techniques for construction of robust, generic and evolutionary programs. *Progress in Informatics*, (10):1–2, March 2013. doi: 10.2201/NiiPi.2013.10.1.

[29] Garrin Kimmel, Aaron Stump, Harley D. Eades, Peng Fu, Tim Sheard, Stephanie Weirich, Chris Casinghino, Vilhelm Sjöberg, Nathin Collins, and Ki Yunh Anh. Equational reasoning about programs with general recursion and call-by-value semantics. *Progress in Informatics*, (10):19–48, March 2013.

[30] Stephanie Weirich, Justin Hsu, and Richard A. Eisenberg. System fc with explicit kind equality. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*, ICFP '13, page 275–286, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450323260. doi: 10.1145/2500365.2500599.

[31] Miroslav Pajic, Nicola Bezzo, James Weimer, Rajeev Alur, Rahul Mangharam, Nathan Michael, George J. Pappas, Oleg Sokolsky, Paulo Tabuada, Stephanie Weirich, and Insup Lee. Towards synthesis of platform-aware attack-resilient control systems: extended abstract. In *HiCoNS '13: Proceedings of the 2nd ACM international conference on High confidence networked systems*, pages 75–76, New York, NY, USA, 2013. ISBN 978-1-4503-1961-4.

[32] Richard A. Eisenberg and Stephanie Weirich. Dependently typed programming with singletons. In *Haskell Symposium*, pages 117–130, Copenhagen, Denmark, September 2012.

[33] Michael Greenberg, Benjamin C. Pierce, and Stephanie Weirich. Contracts made manifest. *Journal of Functional Programming*, 22(3):225–274, May 2012.

[34] Benjamin C. Pierce and Stephanie Weirich. Preface to special issue on the POPLMark challenge. *J. Autom. Reasoning*, 49(3):301–302, 2012.

[35] Umut A. Acar, James Cheney, and Stephanie Weirich. Editorial - special issue dedicated to ICFP 2010. *J. Funct. Program.*, 22(4-5):379–381, 2012. doi: 10.1017/S0956796812000287.

[36] Chris Casinghino, Vilhelm Sjöberg, and Stephanie Weirich. Step-indexed normalization for a language with general recursion. In *Fourth workshop on Mathematically Structured Functional Programming (MSFP '12)*, pages 25–39, 2012.

[37] 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, heterogenous equality, and call-by-value dependent type systems. In *Fourth workshop on Mathematically Structured Functional Programming (MSFP '12)*, pages 112–162, 2012.

[38] 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 *Sixth ACM SIGPLAN Workshop Programming Languages meets Program Verification (PLPV '12)*, pages 15–26, 2012.

[39] Brent A. Yorgey, Stephanie Weirich, Julien Cretin, Simon Peyton Jones, Dimitrios Vytiniotis, and José Pedro Magalhaẽs. Giving Haskell a promotion. In *Seventh ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI '12)*, pages 53–66, 2012.

[40] Stephanie Weirich and Chris Casinghino. Generic programming with dependent types. In Jeremy Gibbons, editor, *Generic and Indexed Programming*, number 7470 in Lecture Notes in Computer Science, pages 217–258. Springer-Verlag Berlin Heidelberg, 2012.

[41] Stephanie Weirich, Brent A. Yorgey, and Tim Sheard. Binders unbound. In *Proceeding of the 16th ACM SIGPLAN International Conference on Functional Programming*, ICFP '11, pages 333–345, New York, NY, USA, 2011. ISBN 978-1-4503-0865-6.

[42] Stephanie Weirich, Dimitrios Vytiniotis, Simon Peyton Jones, and Steve Zdancewic. Generative type abstraction and type-level computation. In *POPL 11: 38th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, January 26–28, 2011. Austin, TX, USA.*, pages 227–240, January 2011.

[43] Tim Sheard, Aaron Stump, and Stephanie Weirich. Language-based verification will change the world. In *2010 FSE/SDP Workshop on the Future of Software Engineering Research*, pages 343–348, November 2010. Position paper.

[44] Aaron Stump, Vilhelm Sjöberg, and Stephanie Weirich. Termination casts: A flexible approach to termination with general recursion. In *Workshop on Partiality and Recursion in Interactive Theorem Provers*, pages 76–93, Edinburgh, Scotland, July 2010.

[45] Dimitrios Vytiniotis and Stephanie Weirich. Parametricity, type equality and higher-order polymorphism. *Journal of Functional Programming*, 20(2):175–210, March 2010.

[46] Michael Greenberg, Benjamin Pierce, and Stephanie Weirich. Contracts made manifest. In *37th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 353–364, Madrid, Spain, January 2010.

[47] Limin Jia, Jianzhou Zhao, Vilhem Sjöberg, and Stephanie Weirich. Dependent types and program equivalence. In *37th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 275–286, Madrid, Spain, January 2010.

[48] Stephanie Weirich and Chris Casinghino. Arity-generic type-generic programming. In *ACM SIGPLAN Workshop on Programming Languages Meets Program Verification (PLPV)*, pages 15–26, January 2010.

[49] Aaron Bohannon, Benjamin C. Pierce, Vilhelm Sjöberg, Stephanie Weirich, and Steve Zdancewic. Reactive noninterference. In *16th ACM Conference on Computer and Communications Security*, pages 79–90, November 2009.

[50] Dimitrios Vytiniotis, Stephanie Weirich, and Simon Peyton Jones. FPH: first-class polymorphism for Haskell. In *ICFP 2008: The 13th ACM SIGPLAN International Conference on Functional Programming*, pages 295–306, Victoria, BC, Canada, September 2008.

[51] Daniel S. Dantas, David Walker, Geoffrey Washburn, and Stephanie Weirich. AspectML: A polymorphic aspect-oriented functional programming language. *ACM Transactions on Programming Languages*, 30(3):1–60, May 2008. ISSN 0164-0925.

[52] Brian Aydemir, Arthur Charguéraud, Benjamin C. Pierce, Randy Pollack, and Stephanie Weirich. Engineering formal metatheory. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 3–15, January 2008.

[53] Geoffrey Washburn and Stephanie Weirich. Boxes go bananas: Encoding higher-order abstract syntax with parametric polymorphism. *Journal of Functional Programming*, 18(1):87–140, January 2008.

[54] Dimitrios Vytiniotis and Stephanie Weirich. Dependent types: Easy as PIE. In Marco T. Morazán and Henrik Nilsson, editors, *Draft Proceedings of the 8th Symposium on Trends in Functional Programming*, pages XVII–1—XVII–15. Dept. of Math and Computer Science, Seton Hall University, April 2007. TR-SHU-CS-2007-04-1.

[55] Dimitrios Vytiniotis and Stephanie Weirich. Free theorems and runtime type representations. In *Mathematical Foundations of Programming Semantics (MFPS XXIII)*, pages 357–373, New Orleans, LA, USA, April 2007.

[56] Simon L. Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Mark Shields. Practical type inference for arbitrary-rank types. *Journal of Functional Programming*, 17(1):1–82, January 2007.

[57] Stephanie Weirich. Type-safe run-time polytypic programming. *Journal of Functional Programming*, 16(10):681–710, November 2006.

[58] Stephanie Weirich. RepLib: A library for derivable type classes. In *Haskell Workshop*, pages 1–12, Portland, OR, USA, September 2006.

[59] Geoffrey Washburn and Stephanie Weirich. Good advice for type-directed programming: Aspect-oriented programming and extensible generic functions. In *Workshop on Generic Programming (WGP)*, pages 33–44, Portland, OR, USA, September 2006.

[60] Dimitrios Vytiniotis, Stephanie Weirich, and Simon L. Peyton Jones. Boxy type inference for higher-rank types and impredicativity. In *International Conference on Functional Programming (ICFP)*, pages 251–262, Portland, OR, USA, September 2006.

[61] Simon L. Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Geoffrey Washburn. Simple unification-based type inference for GADTs. In *International Conference on Functional Programming (ICFP)*, pages 50–61, Portland, OR, USA, September 2006.

[62] Brian Aydemir, Aaron Bohannon, and Stephanie Weirich. Nominal reasoning techniques in Coq. In *International Workshop on Logical Frameworks and Meta-Languages:Theory and Practice (LFMTP)*, pages 60–69, Seattle, WA, USA, August 2006.

[63] Benjamin C. Pierce, Peter Sewell, Stephanie Weirich, and Steve Zdancewic. It is time to mechanize programming language metatheory. In *Verified Software: Theories, Tools, Experiments (VS:TTE)*, pages 26–30, Zürich, Switzerland, October 2005.

[64] Daniel S. Dantas, David Walker, Geoffrey Washburn, and Stephanie Weirich. PolyAML: A polymorphic aspect-oriented functional programmming language. In *ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 306–319, Tallinn, Estonia, September 2005.

[65] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. Mechanized metatheory for the masses: The POPLmark challenge. In *The 18th International Conference on Theorem Proving in Higher Order Logics (TPHOLs)*, pages 50–65, Oxford, UK, August 2005.

[66] Geoffrey Washburn and Stephanie Weirich. Generalizing parametricity using information flow. In *Twentieth Annual IEEE Symposium on. Logic in Computer Science (LICS 2005)*, pages 62–71, Chicago, IL, USA, June 2005.

[67] Dimtrios Vytiniotis, Geoffrey Washburn, and Stephanie Weirich. An open and shut typecase. In *ACM SIGPLAN Workshop on Types in Language Design and Implementation*, pages 13–24, Long Beach, CA, USA, January 2005.

[68] Stephanie Weirich. Type-safe cast. *Journal of Functional Programming*, 14(6):681–695, November 2004.

[69] Stephanie Weirich and Liang Huang. A design for type-directed Java. In Viviana Bono, editor, *Workshop on Object-Oriented Developments (WOOD)*, ENTCS, pages 117–136, August 2004.

[70] Geoffrey Washburn and Stephanie Weirich. Boxes go bananas: Encoding higher-order abstract syntax with parametric polymorphism. In *ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 249–262, Uppsala, Sweden, August 2003.

[71] Karl Crary, Stephanie Weirich, and Greg Morrisett. Intensional polymorphism in type erasure semantics. *Journal of Functional Programming*, 12(6):567–600, November 2002.

[72] Stephanie Weirich. Higher-order intensional type analysis. In Daniel Le Métayer, editor, *11th European Symposium on Programming (ESOP)*, pages 98–114, Grenoble, France, April 2002.

[73] Stephanie Weirich. Encoding intensional type analysis. In D. Sands, editor, *10th European Symposium on Programming (ESOP)*, pages 92–106, Genova, Italy, April 2001.

[74] Michael Hicks, Stephanie Weirich, and Karl Crary. Safe and flexible dynamic linking of native code. In R. Harper, editor, *Types in Compilation: Third International Workshop, TIC 2000; Montreal, Canada, September 21, 2000; Revised Selected Papers*, volume 2071 of *Lecture Notes in Computer Science*, pages 147–176. Springer, 2001.

[75] Stephanie Weirich. Type-safe cast: Functional pearl. In *Proceedings of the fifth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 58–67, Montreal, Canada, September 2000.

[76] Karl Crary and Stephanie Weirich. Resource bound certification. In *The Twenty-Seventh ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 184–198, Boston, MA, USA, January 2000.

[77] Karl Crary and Stephanie Weirich. Flexible type analysis. In *Proceedings of the fourth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 233–248, Paris, France, September 1999.

[78] Greg Morrisett, Karl Crary, Neal Glew, Dan Grossman, Richard Samuels, Frederick Smith, David Walker, Stephanie Weirich, and Steve Zdancewic. TALx86: A realistic typed assembly language. In *Second ACM SIGPLAN Workshop on Compiler Support for System Software*, pages 25–35, Atlanta, GA, USA, May 1999. Published as INRIA research report number 0228, March 1999.

[79] Karl Crary, Stephanie Weirich, and Greg Morrisett. Intensional polymorphism in type erasure semantics. In *Proceedings of the third ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 301–313, Baltimore, MD, USA, September 1998.

[80] Cormac Flanagan, Matthew Flatt, Shriram Krishnamurthi, Stephanie Weirich, and Matthias Felleisen. Catching bugs in the web of program invariants. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 23–32, 1996.

Thesis

[81] Stephanie Weirich. *Programming With Types*. PhD thesis, Cornell University, August 2002.

Artifacts and Technical Reports

[82] Yiyun Liu, Jonathan Chan, and Stephanie Weirich. Artifact associated with consistency of a dependent calculus of indistinguishability. Technical report, November 2024. URL `https://doi.org/10.5281/zenodo.14252132`. Awarded "Available", and "Evaluated & Reusable" badges by POPL 2025 Artifact Evaluation Committee.

[83] Cassia Torczon, Emmanuel Suárez Acevedo, Shubh Agrawal, Joey Velez-Ginorio, and Stephanie Weirich. Artifact associated with "effects and coeffects in call-by-push-value". Technical report, July 2024. URL `https://zenodo.org/records/12654518`. Awarded "Available", "Evaluated & Reusable" and "Results Reproduced" badges by OOPSLA 2024 Artifact Evaluation Committee.

[84] Li-yao Xia, Laura Israel, Maite Kramarz, Nicholas Coltharp, Koen Claessen, Stephanie Weirich, and Yao Li. Story of Your Lazy Function's Life: A Bidirectional Demand Semantics for Mechanized Cost Analysis of Lazy Programs (Artifact). Technical report, June 2024. URL `https://doi.org/10.5281/zenodo.11493754`. Awarded "Available" and "Evaluated and Functional" badges by ICFP 2025 Artifact Evaluation Committee.

[85] Yiyun Liu, Jonathan Chan, Jessica Shi, and Stephanie Weirich. Artifact associated with Internalizing Indistinguishability with Dependent Types. Technical report, October 2023.

URL https://doi.org/10.5281/zenodo.8423073. Awarded "Available" and "Evaluated & Reusable" badges by POPL 2024 Artifact Evaluation Committee.

[86] Yiyun Liu and Stephanie Weirich. Artifact associated with dependently-typed programming with logical equality reflection. Technical report, 2023. URL https://doi.org/10.1145/3580401. Awarded "Available" and "Evaluated & Reusable" badges by ICFP 2023 Artifact Evaluation Committee.

[87] Yao Li and Stephanie Weirich. Program adverbs and tlön embeddings (artifact). Technical report, June 2022. Awarded "Available" and "Evaluated & Reusable" badges by ICFP 2023 Artifact Evaluation Committee.

[88] Pritam Choudhury, Harley Eades III, and Stephanie Weirich. Artifact associated with a dependent dependency calculus. Technical report, January 2022. Awarded "Reusable" and "Available" badges by ESOP 2022 Artifact Evaluation Committee.

[89] Yao Li, Li yao Xia, and Stephanie Weirich. Reasoning about the garden of forking paths (artifact). Technical report, May 2021. Awarded "Functional" and "Available" badges by ICFP 2021 Artifact Evaluation Committee.

[90] Yao Li, Li-yao Xia, and Stephanie Weirich. Reasoning about the garden of forking paths. Technical report, March 2021. URL https://arxiv.org/abs/2103.07543.

[91] Pritam Choudhury, Harley Eades III, Richard A. Eisenberg, and Stephanie Weirich. Artifact for "a graded dependent type system with a usage-aware semantics". Technical report, December 2020. Awarded "Reusable", "Functional" and "Available" badges by POPL 2021 Artifact Evaluation Committee.

[92] Pritam Choudhury, Harley D. Eades III, Richard A. Eisenberg, and Stephanie Weirich. A graded dependent type system with a usage-aware semantics (extended version). Technical report, 2020. URL https://arxiv.org/abs/2011.04070.

[93] Stephanie Weirich, Pritam Choudhury, Antoine Voizard, and Richard A. Eisenberg. Replication package for article: A role for dependent types in haskell. Technical report, July 2019. Awarded "Functional" and "Available" badges by ICFP 2019 Artifact Evaluation Committee.

[94] Stephanie Weirich, Pritam Choudhury, Antoine Voizard, and Richard Eisenberg. A role for dependent types in Haskell (extended version). Technical report, 2019. URL https://arxiv.org/abs/1905.13706.

[95] Joachim Breitner, Antal Spector-Zabusky, Yao Li, Christine Rizkallah, John Wiegley, Joshua Cohen, and Stephanie Weirich. The hs-to-coq tool with examples. Technical report, July 2018. Awarded "Artifacts Available" and "Artifacts Evaluated - Functional" badges by ICFP 2018 Artifact Evaluation Committee.

[96] Richard A. Eisenberg, Stephanie Weirich, and Hamidhasan G. Ahmed. Visible type application (extended version). Technical report, January 2016.

[97] Vilhelm Sjöberg and Stephanie Weirich. Programming up to congruence (extended version). Technical Report MS-CIS-14-10, University of Pennsylvania, October 2014.

[98] Joachim Breitner, Richard A. Eisenberg, Simon Peyton Jones, and Stephanie Weirich. Safe zero-cost coercions for Haskell (extended version). Technical Report MS-CIS-14-07, Univ. of Pennsylvania, April 2014.

[99] Chris Casinghino, Vilhelm Sjöberg, and Stephanie Weirich. Combining proofs and programs in a dependently typed language (with technical appendix). Technical Report MS-CIS-13-08, University of Pennsylvania, November 2013.

[100] Richard A. Eisenberg, Dimitrios Vytiniotis, Simon Peyton Jones, and Stephanie Weirich. Closed type families with overlapping equations (extended version). Technical Report MS-CIS-13-10, University of Pennsylvania, November 2013.

[101] Stephanie Weirich, Justin Hsu, and Richard A. Eisenberg. System FC with explicit kind equality (extended version). Technical report, September 2013.

[102] Stephanie Weirich, Dimitrios Vytiniotis, Simon Peyton Jones, and Steve Zdancewic. Generative type abstraction and type-level computation (extended version). Technical report, November 2010.

[103] Aaron Stump, Vilhelm Sjöberg, and Stephanie Weirich. Termination casts: A flexible approach to termination with general recursion (technical appendix). Technical Report MS-CIS-10-21, University of Pennsylvania Department of Computer and Information Science, 2010.

[104] Dimitrios Vytiniotis, Stephanie Weirich, and Simon L. Peyton Jones. Boxy type inference for higher-rank types and impredicativity, Technical Appendix. Technical Report MS-CIS-05-23, University of Pennsylvania, April 2006.

[105] Dimitrios Vytiniotis, Stephanie Weirich, and Simon L. Peyton Jones. Simple unification-based type inference for GADTs, Technical Appendix. Technical Report MS-CIS-05-22, University of Pennsylvania, April 2006.

[106] Simon L. Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Mark Shields. Practical type inference for arbitrary-rank types (technical appendix). Technical Report MIS-CIS-05-14, University of Pennsylvania, July 2005.

[107] Geoffrey Washburn and Stephanie Weirich. Generalizing parametricity using information flow (extended version). Technical Report MS-CIS-05-04, Computer and Information Science, University of Pennsylvania, July 2005.

[108] Daniel S. Dantas, David Walker, Geoffrey Washburn, and Stephanie Weirich. PolyAML: a polymorphic aspect-oriented functional programming language (extended version). Technical Report MS-CIS-05-07, University of Pennsylvania, Department of Computer and Information Science, 2005.

[109] Dan S. Dantas, David Walker, Geoffrey Washburn, and Stephanie Weirich. Analyzing polymorphic advice. Technical Report TR-717-04, Princeton University Computer Science, December 2004.

[110] Liang Huang and Stephanie Weirich. A design for type-directed programming in Java (extended version). Technical Report MS-CIS-04-11, University of Pennsylvania, Computer and Information Science, October 2004.

[111] Dimtrios Vytiniotis, Geoffrey Washburn, and Stephanie Weirich. An open and shut typecase (extended version). Technical Report MS-CIS-04-26, University of Pennsylvania, Computer and Information Science, October 2004.

[112] Simon L. Peyton Jones, Geoffrey Washburn, and Stephanie Weirich. Wobbly types: Practical type inference for generalised algebraic dataypes. Technical Report MS-CIS-05-26, University of Pennsylvania, Computer and Information Science Department, Levine Hall, 3330 Walnut Street, Philadelphia, Pennsylvania, 19104-6389, July 2004.

[113] Geoffrey Washburn and Stephanie Weirich. Unifying nominal and structural ad-hoc polymorphism (extended version). Technical report, University of Pennsylvania, March 2004.

[114] Geoffrey Washburn and Stephanie Weirich. Boxes go bananas: Encoding higher-order abstract syntax with parametric polymorphism (extended version). Technical Report MS-CIS-03-26, University of Pennsylvania, Computer and Information Science, September 2003.

[115] Michael Hicks and Stephanie Weirich. A calculus for dynamic loading. Technical Report MS-CIS-00-07, University of Pennsylvania, April 2000.

[116] Karl Crary, Stephanie Weirich, and Greg Morrisett. Intensional polymorphism in type erasure semantics (extended version). Technical Report TR98-1721, Cornell University, Computer Science, November 1998.

Unrefereed Reports

[117] Stephanie Weirich. Tracking how dependently-typed functions use their arguments, 2024. Invited keynote given at LICS/ICALP/FSCD.

[118] Emmanuel Suárez Acevedo and Stephanie Weirich. Making logical relations more relatable (proof pearl), September 2023.

[119] Stephanie Weirich. Implementing dependent types in pi-forall, 2023. URL `https://arxiv.org/abs/2207.02129`. Lecture notes for the Oregon Programming Languages Summer School.

[120] Stephanie Weirich and Benjamin Pierce. Icfp 2020 post-conference report, 2021. URL `https://arxiv.org/abs/2104.01239`.

[121] Antal Spector-Zabusky, Joachim Breitner, Yao Li, and Stephanie Weirich. Embracing a mechanized formalization gap, October 2019.

[122] Anastasiya Kravchuk-Kirilyuk Stephanie Weirich, Antoine Voizard. Locally nameless at scale. The Fourth International Workshop on Coq for Programming Languages, January 2018.

[123] Stephanie Weirich. The influence of dependent types, January 2017. Invited keynote given at POPL 2017.

[124] Stephanie Weirich. Depending on types, 2014. Invited keynote given at ICFP 2014.

[125] Stephanie Weirich. Designing dependently-typed programming languages. Lectures given at the Summer School on Logic and Theorem Proving in Programming Languages, Eugene OR, USA. July 2013, July 2013.

[126] Stephanie Weirich. Dependently typed programming in ghc, May 2012. Invited keynote given at FLOPS 2012.

[127] Kathleen Fisher, Ronald Garcia, and Stephanie Weirich. Nourishing the future of the field: the programming language mentoring workshop 2012, April 2012.

[128] Stephanie Weirich. Combining proofs and programs, June 2011. Joint invited speaker for Rewriting Techniques and Applications (RTA 2011) and Typed Lambda Calculi and Applications (TLCA 2011).

[129] Stephanie Weirich. Icfp 2010 pc chair's report, September 2010.

[130] Brian Aydemir and Stephanie Weirich. Lngen: Tool support for locally nameless representations, June 2010.

[131] Stephanie Weirich. Haskell 2009 pc chair's report, August 2009.

[132] Brian Aydemir, Steve Zdancewic, and Stephanie Weirich. Abstracting syntax, March 2009.

[133] Stephanie Weirich and Brian Aydemir. Coq for programming language metatheory. Lectures given at the Summer School on Logic and Theorem Proving in Programming Languages, Eugene OR, USA. July 22-25, 2008, 2008. Tutorial materials available at `http://www.cis.upenn.edu/~plclub/oregon08/`.

[134] Brian Aydemir, Aaron Bohannon, Benjamin Pierce, Jeffrey Vaughan, Dimitrios Vytiniotis, Stephanie Weirich, and Steve Zdancewic. Using proof assistants for programming language research or, how to write your next popl paper in coq. Tutorial session co-located with POPL 2008, San Francisco, CA, January 2008. Tutorial materials available at `http://www.cis.upenn.edu/~plclub/popl08-tutorial/`.

[135] Karl Crary, Robert Harper, Frank Pfenning, Benjamin C. Pierce, Stephanie Weirich, and Stephan Zdancewic. Manifest security, January 2007. White paper.

Keynotes

1. *Tracking how dependently-typed functions use their arguments.* LICS/ICALP/FSCD Joint Invited Speaker. Talinn, Estonia. 10 July, 2024.

2. *What are Dependent Types and What are they Good for?* Milner lecture, Laboratory for Computer Science, Edinburgh. June 15, 2022.

3. *How to implement the Lambda Calculus, Quickly.* Keynote presentations at IFL, The 33rd Symposium on Implementation and Application of Functional Languages. September 3, 2021.

4. *A Dependently-Typed Core Calculus for GHC*. TYPES 2019 Conference Invited speaker. Oslo, Norway. June 2019.

5. *The Influence of Dependent Types.* Keynote address, ACM Symposium on Principles of Programming Languages (POPL '17) Paris, France, January 2017.

6. *Depending on Types.* Keynote address, International Conference on Functional Programming (ICFP). Gothenburg, Sweden, September 3, 2014

7. *Dependently-typed programming in GHC.* Invited speaker, FLOPS 2012: Eleventh International Symposium on Functional and Logic Programming, Kobe, Japan, May 25, 2012.

8. *Combining Proofs and Programs.* Joint invited speaker for Rewriting Techniques and Applications (RTA 2011) and Typed Lambda Calculi and Applications (TLCA 2011) Novi Sad, Serbia, June 1, 2011.

Industry Presentations

1. *How to implement the Lambda Calculus, Quickly.* Haskell Love, virtual developer conference. September 10, 2021.

2. *Strongly-Typed System F in GHC*. YOW! Lambda Jam Online 2020. July 2020.

3. *Dependent Types in Haskell.* BOBkonf invited speaker (research track). Berlin, Germany. August 21, 2019.

4. *Dependent Types in Haskell.* Haskell eXchange keynote, London, October 2018.

5. *Work-in-progress: Verifying the Glasgow Haskell Compiler Core language using the Coq proof assistant.* Intel Labs, Hillsboro, OR. August 2018.

6. *Dependent Types in Haskell.* Invited speaker, Comcast Labs Connect: Functional Programming. March 9, 2018.

7. *Dependent Types in Haskell.* StrangeLoop 2017, St. Louis, MO, September 2017.

8. *Depending on Types.* Typelevel Summit, Philadelphia, PA, March 2-3, 2016.

9. *Dynamic Typing in GHC*. Compose :: Conference, Brooklyn, NY, February 4-5, 2016.

10. *Visible Type Application.* Microsoft Research, Cambridge, UK, November 6, 2015.

11. *Depending on Types.* Code Mesh 2015, London, November 4, 2015.

12. *Pi-Forall: How to use and implement a dependently-typed language.* Technical Keynote, Compose Conference. New York, January 30, 2015.

13. *The Pleasure and Pain of Advanced Type Systems.* Facebook Faculty Summit. Menlo Park, CA, August 6, 2013.

14. *A Design for Type-Directed Java.* Microsoft Research Lab, Cambridge, UK. August 31, 2004.

15. *Resource Bound Certification.* IBM Research, Hawthorne, NY. June 2000.

Conference talks, Seminars, and other Technical presentations

1. *A tale of four lambda calculus interpreters.* WG 2.8 Meeting, York, Maine. May 2025.
2. *Tracking how dependently-typed functions use their arguments.* Chalmers Department Seminar, Gothenberg, Sweden. 7 March 2025.
3. *CBPV+effects, CBPV+coeffects.* WG 2.11 Meeting, Drexel University, Philadelphia, PA. March 2024.
4. *CBPV+effects, CBPV+coeffects.* WG 2.8 Meeting, Pembroke College, Cambridge UK. April 2023.
5. *Stratified Type Theory*, WG 2.8 Meeting, Pembroke College, Cambridge UK. April 2023.
6. *A Dependent Dependency Calculus.* Ediburgh LFCS Seminar, June 14, 2022.
7. *Programming Language Design: From Grace Hopper to Today.* ENIAC Day: 75th anniversary of ENIAC mini-symposium. February 15, 2021.
8. *Strongly-Typed System F in GHC*. Chalmers Functional Programming Seminar Series. June 2020.
9. *Adventures in Quantitative Type Theory.* IFIP WG 2.8 Meeting, Zion National Park. March 2020.
10. *A Dependently-Typed Core Calculus for GHC*. PurPL Fest invited speaker. Purdue University. West Lafayette, IN. September 23, 2019.
11. *Strongly-typed System F in GHC*. IFIP WG 2.8, Bordeaux, Fr. May 2019.
12. *Dependent Types in Haskell.* Cornell CS Colloquium, Ithaca, NY. November 2018.
13. *Work-in-progress: Towards a formal semantics for GHC Core.* DeepSpec 2018 Workshop, Philadelphia, PA. June 2018.
14. *Work-in-progress: Verifying the Glasgow Haskell Compiler Core language.* IFIP WG 2.8, Asilomar, CA. June 2018.
15. *Work-in-progress: Verifying the Glasgow Haskell Compiler Core language.* Dagstuhl Seminar 18201, "Secure Compilation" Wadern, Germany, May, 2018.
16. *Locally Nameless at Scale.* Joint presentation with Anastasiya Kravchuck-Kirilyuk. CoqPL workshop, Los Angeles, CA. January 2018.
17. *Dependent Types in Haskell.* University of Washington, PLSE Seminar Series, Seattle, WA, December 2017.
18. *Dependent Types in Haskell.* McMaster University Departmental Seminar, Hamilton Ontario, November 2017.

19. *Eta-equivalence in Core Dependent Haskell.* WG 2.8, Edinburgh, UK. June 2017.

20. *A Foundation for Dependently Typed Haskell.* WG 2.8, Lake Placid, NY, July 19, 2016.

21. *Visible Type Application.* University of Kent, November 5, 2015.

22. *From System F to Typed Assembly Language, by Morrisett, Walker, Crary, Glew.* Papers We Love, Philadelphia. Philadelphia, PA, October 6, 2015

23. *Towards Dependently Typed Haskell.* WG 2.8, Kefalonia, Greece, May 24, 2015

24. *Programming up-to Congruence.* ACM Symposium on Principles of Programming Languages (POPL '15). Mumbai, India, January 16, 2015

25. *Depending on Types.* Computer Science Colloquium Series, Indiana University. Bloomington, Indiana, October 17, 2014

26. *Programming Languages Panel.* Cornell CS 50th Anniversary Symposium. Ithaca, New York, October 2, 2014

27. *Programming Up-to Congruence, Again.* WG 2.8, Estes Park, Colorado, August 12, 2014

28. *Combining Proofs and Programs.* Certification of High-level and Low-level programs. Paris, France, July 7, 2014

29. *Why You Should Care About Dependent Types.* Programming Languages Mentoring Workshop. San Diego, CA, January 21, 2014

30. *Programming Up-to Congruence.* WG 2.8, Aussios, France, October 14, 2013

31. *Paradoxical Typecase.* WG 2.8, Anapolis, MD, November 7, 2012

32. *A POPLmark Retrospective: Using Proof Assistants in Programming Language Research.* Invited speaker, LFMTP 2012: 7th International Workshop on Logical Frameworks and Meta-languages: Theory and Practice, Copenhagen, Denmark, September 9, 2012.

33. *Binders Unbound.* The 16th ACM SIGPLAN International Conference on Functional Programming, ICFP 2012 Tokyo Japan, September 21, 2011

34. *Combining Proofs and Programs.* Dependently Typed Programming, Shonan Seminar 007, Shonan Village, Japan, September 16, 2011

35. *Combining Proofs and Programs in Trellys.* Plenary Address, MFPS 27. Pittsburgh, PA. May 26, 2011

36. *Generic Binding and Telescopes.* WG 2.8, Marble Falls, TX. March 11, 2011

37. *Generative Type Abstraction and Type-level Computation.* ACM Symposium on Principles of Programming Languages (POPL '11). Austin, TX, January 2011

38. *ICFP 2010 Program Chair's Report.* Baltimore, MD. September 27, 2010

39. *Dependent Types and Program Equivalence.* University of Strathclyde. Glasgow, Scotland. April 30, 2010

40. *Generic Programming with Dependent Types.* IFIP 2.11, St. Andrews, Scotland. March 1-3, 2010

41. *Dependent Types and Program Equivalence.* University of Nottingham. Nottingham, England. February 5, 2010

42. *Trellys Status Report.* PLPV Discussion. Madrid, Spain. January 19, 2010

43. *A POPLmark Retrospective: Using Proof Assistants in Programming Language Research.* University of Cambridge Computer Laboratory Wednesday Seminars. Cambridge, England. December 2, 2009

44. *Dependent Types and Program Equivalence.* Semantics Lunch, University of Cambridge Computer Laboratory. Cambridge, England. November 2, 2009

45. *Haskell Symposium 2009 Program Chair's report.* Edinburgh, Scotland. September 3, 2009

46. *Doing Dependent Types Wrong Without Going Wrong.* IFIP WG 2.8, Frauenchiemsee, Germany, June 2009

47. *Adventures in Dependently-Typed Metatheory.* IFIP WG 2.11, Mountain View CA. April 15, 2009

48. *Engineering Formal Metatheory* Computer Science Colloquium, City University of New York Graduate Center. New York, NY. February 2, 2009

49. *First-class Polymorphism for Haskell.* IFIP WG 2.8, Park City, UT. June 19, 2008

50. *Engineering Formal Metatheory.* Princeton University, Princeton NJ, USA. November 19, 2007

51. *Machine Assistance for Programming Language Research.* Cornell University, Ithaca, NY, USA. October 12, 2007

52. *Formal Reasoning About Programs and Programming Languages.* National Security Agency. Fort Meade, MD, USA. July 20, 2007

53. *Engineering Aspects of Formal Metatheory.* Harvard University, Boston MA, USA. June 1, 2007

54. *Dependently-Typed Languages.* Working session summary. IFIP WG 2.11, Portland, OR, October 2006

55. *Simple Unification-Based Type Inference for GADTs.* International Conference on Functional Programming (ICFP). Portland, OR. September 2006

56. *RepLib: A Library for Derivable Type Classes.* Haskell Workshop. Portland, OR. September 2006

57. *Parametricity and GADTs.* IFIP Working Group 2.8 (Functional Programming). Boston, MA. July 2006

58. *Practical Type Inference for Advanced Type Systems.* International Federation for Information Processing (IFIP) Working Group 2.11, Dagstuhl, Wadern, Germany. January 2006

59. *Boxy Types: Inference for Higher-rank Types and Impredicativity.* International Federation for Information Processing (IFIP) Working Group 2.8, Kalvi Manor, Estonia. October 2005

60. *A Core Language for Generalised Algebraic Datatypes.* International Federation for Information Processing (IFIP) Working Group 2.8, West Point, USA. November 2004

61. *A Design for Type-directed Java.* Programming Languages Seminar, Yale University, New Haven, CT. October 1, 2004

62. *2004 ICFP Programming Contest Results.* (Presented jointly with Benjamin Pierce and Steve Zdancewic) International Conference on Functional Programming, Snowbird, UT. September 20, 2004

63. *A Core Language for Generalised Algebraic Datatypes.* Dagstuhl Seminar 04381: Dependently Typed Programming, Wadern, Germany. September 12, 2004

64. *A Design for Type-Directed Java.* Workshop on Object-Oriented Developments (WOOD '04). London, UK, August 2004

65. *Unifying Nominal and Structural Ad-hoc Polymorphism.* International Federation for Information Processing (IFIP) Working Group 2.8, Coffs Harbour, Australia. January 2003

66. *Unifying Nominal and Structural Ad-hoc Polymorphism.* Computer Science Colloquium, City University of New York Graduate Center. New York, NY. October 30, 2003

67. *Boxes Go Bananas: Parametric Higher-Order Abstract Syntax in System F.* Laboratory for Secure Systems Seminar, Stevens Institute of Technology. Hoboken, NJ. May 5, 2003

68. *Run-time type analysis in Haskell with an Awful Lot of Newtypes.* International Federation for Information Processing (IFIP) Working Group 2.8, Crans-Montana, Switzerland. January 2003

69. *Polytypic Programming and Intensional Type Analysis.* New Jersey Programming Languages Seminar. University of Pennsylvania, Philadelphia, PA. September 20, 2002

70. *Programming with Types.* OHSU/Oregon Graduate Institute, Beaverton, OR. February 11, 2002

71. *Programming with Types.* University of Oregon, Eugene, OR. February 15, 2002

72. *Programming with Types.* University of Pennsylvania, Philadelphia, PA. February 19, 2002

73. *Programming with Types.* University of Virginia, Charlottesville, VA. February 28, 2002

74. *Programming with Types.* University of Maryland, College Park, MD. March 4, 2002

75. *Programming with Types.* Northeastern University, Boston, MA. March 13, 2002

76. *Programming with Types.* University of California, San Diego, CA. March 15, 2002

77. *Programming with Types.* Purdue University, West Lafayette, IN. March 25, 2002

78. *Programming with Types.* University of Michigan, Ann Arbor, MI. March 27, 2002

79. *Programming with Types.* University of Texas, Austin, TX. April 2, 2002

80. *Higher-order Intensional Type Analysis.* European Symposium on Programming (ESOP '02). Grenoble, France, April 2002

81. *Programming with Types.* University of Colorado at Boulder, CO. April 16, 2002

82. *Programming with Types.* Pennsylvania State University, State College, PA. April 19, 2002

83. *Programming with Types.* Massachusetts Institute of Technology, Boston, MA. April 25, 2002

84. *Programming with Types.* Rice University, Houston TX. April 29, 2002

85. *Run-Time Type Analysis and Program Verification.* Research, Careers and Computer Science: A Maryland Symposium. University of Maryland, College Park, MD. November 2001

86. *Polytypic Programming and Intensional Type Constructor Analysis.* International Federation for Information Processing (IFIP) Working Group 2.8, Are, Sweden. April 2001

87. *Encoding Intensional Type Analysis.* European Symposium on Programming (ESOP '01). Genova, Italy. April 2001

88. *Resource Bound Certification.* Harvard University, Boston, MA. February 2001

89. *Functional Pearl: Type-Safe Cast.* International Conference on Functional Programming. Montreal, Canada. September 2000

90. *Resource Bound Certification.* ACM Symposium on Principles of Programming Languages (POPL '00). Boston, MA, USA. January 2000

91. *Flexible Type Analysis.* International Conference on Functional Programming (ICFP '99). Paris, France, September 1999

92. *Type Analysis and Typed Compilation.* Princeton University, Princeton, NJ. June 1999

93. *Intensional Polymorphism in Type-Erasure Semantics.* International conference on Functional Programming (ICFP '98). Baltimore, MD, USA, September 1998

Research Community Service

Conference and Symposium Leadership

- POPL 2026, associate chair.
- European Symposium on Programming (ESOP) 2024, program chair.
- International Conference on Functional Programming (ICFP) 2020, general chair.
- Principles of Programming Languages (POPL) 2019, program chair.
- International Conference on Functional Programming (ICFP) 2010, program chair.
- Haskell Symposium 2009, program chair.

Journal Leadership

- Editorial Board of TheoreTICs, 2021-2024.
- Associate Editor of ACM TOPLAS, 2019-2023.
- Editorial Board of Logical Methods in Computer Science, 2016-2024.

- Editor of Journal of Functional Programming, 2011-2017.
- Guest Editor (with Zhenjiang Hu, Shin-Cheng Mu), Progress in Informatics. Special Issue on Advanced Programming Techniques for Construction of Robust, General and Evolutionary Programs, March 2013.
- Guest Editor (with Benjamin Pierce), Journal of Automated Reasoning. Special Issue on the POPLmark Challenge. October 2012.
- Nomination committee, SIGPLAN CACM Research Highlights, 2009-2011.

Workshop Leadership

- DeepSpec Workshop, 2018. Co-located with PLDI, co-organizer.
- Dagstuhl seminar "Language Based Verification Tools for Functional Programs" (16131), April 2016, co-organizer.
- Dependently-Typed Programming Workshop (DTP) 2013, program chair and organizer.
- Shonan Village Dependently-Typed Programming, 2011, co-organizer.
- Types in Language Design and Implementation Workshop, 2011, general chair.
- Workshop on Mechanizing Metatheory, 2006-2009, co-organizer.
- Workshop on Mechanizing Metatheory, 2006, program chair.

Program Committee Membership (conference/symposium)

- International Conference on Functional Programming (ICFP) 2025.
- Certified Proofs and Programs (CPP) 2025.
- International Conference on Functional Programming (ICFP) 2022.
- Formal Structures for Computation and Deduction (FSCD) 2022.
- History of Programming Languages (HOPL IV) 2020.
- Symposium on Trends in Functional Programming (TFP) 2020.
- Principles of Programming Languages (POPL) 2018.
- Certified Proofs and Programs (CPP) 2018.
- European Symposium on Programming (ESOP) 2017.
- Principles and Practice of Declarative Programming (PPDP) 2016.
- Symposium on Trends in Functional Programming (TFP) 2016.
- International Conference on Functional Programming (ICFP) 2015.
- Certified Proofs and Programs (CPP) 2015.
- Principles of Programming Languages (POPL) 2014.
- Functional and Logic Programming (FLOPS) 2014.
- Typed Lambda Calculi and Applications (TLCA) 2013.
- Asian Symposium on Programming Languages and Systems (APLAS) 2012.
- International Symp. on Principles and Practice of Declarative Programming (PPDP) 2012.
- Certified Proofs and Programs (CPP) 2011.
- European Symposium on Programming (ESOP) 2011.
- Verified Software, Tools, Theory and Experiments (VSTTE) 2010.
- International Conference on Functional Programming (ICFP) 2007.
- International Conference on Aspect-Oriented Software Development (AOSD) 2007.
- Principles of Programming Languages (POPL) 2006.
- European Symposium on Programming (ESOP) 2006.
- Programming Language Design and Implementation (PLDI) 2004.
- International Conference on Functional Programming (ICFP) 2002.

Program Committee Membership (workshop)

- TYPES 2021.
- TyDe 2021.
- CoqPL Workshop 2020.
- Coq Workshop 2018.
- Haskell Implementor's Workshop (HiW) 2016.
- Higher-Order Programming with Effects (HOPE) 2016.
- Implementation of Functional Languages (IFL) 2015.
- Coq Workshop 2015.
- Logical Frameworks and Meta Languages Theory and Practice (LFMTP) 2013.
- Trends in Functional Programming in Education (TFPIE) 2013.
- Grace Hopper Conference, Panels, Workshops, and Presentations 2012.
- Programming Languages meets Program Verification Workshop (PLPV) 2010.
- IFIP TC2 Working Conference Domain Specific Languages 2009.
- Proof Carrying Code Workshop 2008.
- Haskell Workshop 2007.
- Workshop on Types in Language Design and Implementation (TLDI) 2007.
- ML Workshop 2006.
- MetaOCaml Workshop, 2005.
- Foundations of Object-Oriented Languages Workshop (FOOL) 2005.
- MetaOCaml Workshop, 2004.
- Foundations of Global Ubiquitous Computing Workshop (FGUC) 2004.
- IFIP TC2 Working Conference on Generic Programming 2002.
- Haskell Workshop 2001.

Steering Committee Membership

- ESOP, 2023-2025.
- ICFP, 2009-2012, 2017-2022. SC Chair, 2021-2022
- POPL, 2017-2021
- PLMW, 2012
- Haskell Symposium, 2008-2012, 2018-2021
- TLDI, 2010-2011
- PLPV, 2012-2014
- WGP, 2012-2015

Technical Society Membership

- Association for Computing Machinery, 1998-present
- ACM SIGPLAN, 1998-present
- ACM SIGLOG, 2014-present
- IFIP Working Group 2.8 (Functional Programming), 2003-present
- IFIP Working Group 2.11 (Program Generation), 2007-2012

Other

- SIGPLAN Reynold's dissertation award committee, 2021-2022, 2022-2023.
- NSF panel: January 2025, October 2022, March 2019, March 2018, February 2016, June 2014, October 2012, December 2011, March 2008, December 2004.
- Haskell' language standard committee, 2005-2010.

- TYPES forum moderator: 2003-2009.
- PLDI External Review Committee: 2013, 2011, 2009
- POPL External Review Committee: 2015, 2012.
- Ad hoc reviews: ACM Computing Surveys, JFP, HOSC, Acta Informatica, TOPLAS, SCP, ICFP, POPL, PLDI, ECOOP, LCTES, ICALP, FOOL, ML, Haskell, PEPM, NSF

Teaching Experience

University of Pennsylvania

- CIS 120 - Programming Languages and Techniques I
  Spring 2025, Spring 2024, Spring 2022, Spring 2021, Spring 2020, Spring 2018, Spring 2016, Spring 2015, Spring 2014, Spring 2013, Spring 2012, Spring 2011, Fall 2008, Spring 2008, Spring 2007
- CIS 5520/552 - Advanced Programming
  Fall 2023, Fall 2022, Fall 2021, Fall 2020, Fall 2019, Fall 2017, Spring 2017, Fall 2015, Fall 2013, Fall 2012, Fall 2011
- CIS 6700/670/700 - Advanced topics in Programming Languages
  Spring 2023, Fall 2016, Fall 2010, Spring 2009, Fall 2006, Spring 2006, Fall 2002
- CIS 500 - Software Foundations
  Fall 2014, Fall 2005, Fall 2004
- CIS 340 - Principles of Programming Languages
  Spring 2004, Spring 2003

Cornell University

- CS 212 - Java Practicum
- CS 213 - C++ Programming
- CS 214 - A Taste of UNIX and C

Department, School and University Service

- FPC, 2023-present.
- Faculty Council on Access and Academic Support, 2019-2023.
- University committee on the Facilities, 2022-2023, 2011-2014. Chair 2012-2014.
- CIS Lecturer Hiring committee, 2022-2023.
- CIS Tenure Track Hiring committee, 2021-2022, 2019-2020 (chair), 2012-2013.
- Provost's Faculty Advisory Commitee on Online Learning, 2020.
- Penn Engineering COVID-19 oversight committee, 2020.
- SEAS Undergraduate Curriculum Committee, 2017-2018.
- Undergraduate Chair, CIS, Sep 2014-Dec 2017.
- SEAS Diversity Committee, 2015-2016.
- Penn Forum for Women Faculty, 2015-2016.
- Faculty advisor to CommuniTech (Penn undergraduate service organization). 2012-present.
- Faculty advisor to AΩE International Engineering and Technical Science Sorority. 2012-present.
- Graduate student admissions chair, 2013-2014, 2012-2013.
- CIS seminar series coordinator, 2011-2012.
- Faculty Council, 2010-2012.

- Academic Performance Committee, 2004-2017.
- CIS seminar organizer, 2011-2012.
- CIS 120 reform, 2009-2010.
- Back-up Care Committee, 2009.
- Senior Design Project Judge, 2006.
- Freshman Advisor, 2003.
- Graduate Admissions Committee, 2003-2004.
- Curriculum Committee, 2002-2003.

Outreach

- SIGPLAN CARES member, January 2020-present
- Oregon Programming Languages Summer School, co-organizer, 2023
- Computing Connections Fellowship, selection committe, 2022-present
- Workshop organizer: (with Ron Garcia) ICFP Programming Languages Mentoring Workshop (ICFP-PLMW 2015) Vancouver, BC, September, 2015
- Workshop co-founder: (with Kathleen Fisher and Ron Garcia) Programming Language Mentoring Workshop (PLMW 2012) Philadelphia, PA, January 24, 2012
- SRC@ICFP student research competition, selection committee. 2015, 2014.
- Haskell Foundation, Interim Board member, 2020-2021
- Programming Contest co-organizer:
  - 2004, Seventh Annual ICFP programming contest
  - 2000, Third Annual ICFP Programming Contest
- Panelist/Speaker (External events):
  - Researcher Panel, PLMW @POPL, January 18, 2022
  - Panel on Advising and Research. PLMW @PLDI, June 22, 2021
  - "Dependent types—salvation or plague?", panel discussion at LambdaDays, Feb 4th, 2021
  - CS Curriculum Panel, 35th anniversary of the Computer Science Department at Rice, Houston, TX, October 11-13, 2019
  - Programming Languages Panel, Computer Science 50th Anniversary Symposium, Cornell University, Ithaca, NY, October 2, 2014
  - Teaching Haskell in Academia and Industry (panel). Haskell Symposium, September 2013
  - CRA-W/CDC Programming Languages Summer School, UT Austin, May 2007
  - Women in Science and Engineering Conference, Princeton, February 2006
- Panelist/Speaker (Penn events):
  - Grace Hopper Milestone Celebration, Tuesday, May 7, 2024.
  - Reflections by 50 Years of Women CIS Faculty, March 28, 2023.
  - Women in STEM at Penn, Research Symposium, *"A logical approach to programming language design and verification."* June 3, 2022.
  - SWE Ted Talks Event for high school students: Feb 2022, Dec 2020
  - AWE Faculty Panel: Aug 2022, Feb 2022, Oct 2021
  - Panel Moderator/Member at FemmeHacks: Feb 2018, Feb 2017, Feb 2016
  - The "Computers": Apr 2015
  - WICS high school day for girls: Apr 2017, Apr 2015, Apr 2014, Apr 2013, Apr 2012
  - Graduate Student Professional Seminars: Mar 2013, Mar 2012

  - SWE Graduate Section Inspiration Lunch Talk: April 20, 2012
  - Philadelphia Area Aspirations in Computing Award presentation: March 21, 2012
  - Penn AWE Pre-Orientation: Aug 2021, Aug 2017, Aug 2016, Aug 2011
- Podcast interviews:
  - CoRecursive w/ Adam Bell, June 2018.
  - Type Theory Podcast, "Episode 4: Stephanie Weirich on Zombie and Dependent Haskell", April 2015.

Tutorials

- *Implementing Dependent Types in pi-forall.* Oregon Programming Languages Summer School: Types, Logic, and Verification. Eugene OR, USA. June 2023
- *Implementing Dependent Types in pi-forall.* Oregon Programming Languages Summer School: Types, Logic, and Verification. Eugene OR, USA. June 2022
- *Dependent Types.* Programming Languages Mentoring Workshop. St. Louis, MO. September 2018
- *Formal Logic and Software Verification using Interactive Theorem Provers.* ACM Philadelphia Region Celebration of Women in Computing, April 21, 2018
- *Language Specification and Variable Binding.* The Science of Deep Specifications Summer School, July 2017
- *How to write a great research paper: Simon's seven easy steps.* Programming Languages Mentoring Workshop. Mumbai, India, 2015
- *How to give a good research talk.* Programming Languages Mentoring Workshop. Mumbai, India, 2015
- *Designing Dependently-Typed Programming Languages.* Oregon Programming Languages Summer School: Types, Logic, and Verification. Eugene OR, USA. June 2014
- *Designing Dependently-Typed Programming Languages.* Oregon Programming Languages Summer School: Types, Logic, and Verification. Eugene OR, USA. July 2013
- *Computational Flags.* Swarthmore CATALYST Conference for 7th/8th graders, April 2015, March 2012
- *Generic Programming with Dependent Types.* Spring School on Generic and Indexed Programming. Oxford, England. March 2010
- *Coq for Programming Language Metatheory.* Oregon Programming Languages Summer School on Logic and Theorem Proving in Programming Languages. University of Oregon, July 2008
- *Using Proof Assistants for Programming Language Research or, How to write your next POPL paper in Coq.* POPL Tutorial, Jan 2008
- *Getting started in PL design research.* CRA-W/CDC Programming Languages Summer School. UT Austin, May 2007
- *Career paths: How to get started in academia or industry.* CRA-W/CDC Programming Languages Summer School. UT Austin, May 2007