

Preface

This volume is dedicated to Professor Phil Wadler, and the collection of papers form a Festschrift for Phil. The contributions are made by some of the many who know Phil and have been influenced by him. The research papers included here represent some of the areas in which Phil has been active, and the editors thank their colleagues for agreeing to contribute to this Festschrift. Each paper received at least two reviews, and we thank the many external reviewers who helped to make this feasible. These proceedings were prepared with the aid of EasyChair.

In this preface, we attempt to summarize Phil Wadler's scientific achievements. In addition, we describe the personal style and enthusiasm that Phil has brought to the subject.

Introduction

Throughout his career Phil has made fundamental contributions to programming language design and formal systems for capturing the essence of programming language features. In his research he fruitfully combines theory with practice. Many of his contributions have opened up entire fields of investigation, for example, the "Theorems for Free!" paper [82] inspired much research on parametricity. At other times Phil has worked to bring a technique into the mainstream, for example, popularizing the use of monads for describing effectful computations in functional languages [92].

He has contributed to widely used programming languages and important type systems. His contributions to popular languages include many aspects of the design and implementation of Haskell [31], a formal account of the XQuery declarative query language [103], and generic types for Java [6]. His contributions to important type systems include his work on effects [118], linearity [88], and blame [113].

He has also developed experimental programming languages. Orwell was an early experimental lazy functional programming language, and a precursor to Haskell, and Links [14] is a current experimental functional programming language for the Web.

Phil stimulates fellow researchers. He asks hard and unexpected questions, and often provides valuable critical insights. He takes a personal interest in the work of his research students, providing timely and creative feedback.

Phil sets high standards. He will not accept a technical explanation until it is crisp, clear, and concise. As a result, interacting with Phil can sometimes be painful, particularly if your idea has not yet fully crystallized. However, if you persist and carefully address each of Phil's concerns then you often end up with a much deeper understanding of the original problem.

Scientific Contribution

Rather than giving a detailed review of Phil’s research, we provide a few highlights. Much of Phil’s work is collaborative with other research workers and with his research students and post-docs, often with long-standing collaborations. Our abridged summary does not mention all of the many collaborators in his work, but the list of co-authors in the bibliography below documents these collaborations.

In addition to specific technical contributions outlined below, Phil has provided leadership in the community, for example, serving as chair of the ACM Special Interest Group on Programming Languages (SIGPLAN) between 2009 and 2012.

Programming Languages Phil has made major contributions to programming language design, in particular contributing to functional programming and the theory underpinning functional languages.

Phil’s PhD thesis, entitled “Listlessness Is Better Than Laziness,” made an early contribution to the field of deforestation, and the title, like many of his research paper titles, contains an apposite wordplay. (Phil’s paper titles even appear as the subject of a “spotlight on style” in a book by Helen Sword, *Stylish Academic Writing*.)

In 1984 he developed the Orwell lazy functional language, a precursor of Haskell. In the late 1980s he added ad hoc polymorphism to Haskell in the form of type classes [115]. Type classes became influential, for example, laying the foundation for concepts in C++ and inspiring the design of implicits and traits in Scala.

Phil proposed views [79] as a way of reconciling pattern matching with abstract data types. His work on views was influential in the design of pattern matching for dependently typed programming languages such as Agda and Idris. Abstractions similar to views are now part of F# and Haskell.

Another important piece of work was to develop the theory of, and evangelize the use of, monads for structuring functional programs. Monads were originally proposed by Eugenio Moggi as a basis for reasoning about program equivalence, by making a distinction between values and computations, and allowing the notion of computation to vary. Phil showed how they could be used for a variety of purposes, for instance, expressing queries over collection classes and stateful computations in a functional language. Monads were initially adopted in Haskell, and subsequently in other languages, such as OCaml, F#, and Scala. Phil proposed monad comprehensions [86], which have recently made a comeback in Microsoft’s LINQ, F#, and Haskell.

His work on type erasure for Java [6] directly inspired the current design and implementation of generics in Java [53].

His work on the XQuery typed functional language for querying XML [103] is another example of Phil applying theory to improve a programming language technology. He encouraged the development of a formal semantics for XQuery by showing that the natural language semantics was flawed.

In 2005 Phil began working on the Links functional web programming language [14]. From a single source file, Links generates target code for all three tiers of a typical

Web application (client, server, and database). Ideas developed in Links have had significant impact in industry. For instance, formlets [15], a canonical abstraction for constructing Web forms, have become an integral part of IntelliFactory’s WebSharper framework for Web programming in F#. Similarly, the Links approach to language-integrated query has influenced the design of Microsoft’s LINQ.

Links remains an active research vehicle. Phil currently heads a research project entitled “From Data Types to Session Types—A Basis for Concurrency and Distribution” (or simply “ABCD”). An important part of the ABCD project involves adding session types to Links. The design is partly inspired by Phil’s theoretical work connecting classical linear logic and session types [111].

Type Systems Phil has made major contributions to type systems for programming languages and logic.

Phil’s work on free theorems [82] helped popularize relational parametricity, a fundamental notion of uniformity enjoyed by second-order logic and hence the polymorphic lambda calculus [106]. Relational parametricity underlies the design of generics in Java and C#. Free theorems and relational parametricity have been applied in areas ranging from dimension types to bidirectional programming to physics.

In the 1990s Phil wrote a series of papers on linear logic [3, 48, 49, 61, 68, 85, 88, 89], promoting its application to programming languages via linear type systems. Nowadays a number of research languages, including F*, Clean, Cyclone, Idris, and Links, make use of some form of linear typing. Mozilla’s language Rust, intended as a safe alternative to C, also incorporates a form of linearity. Furthermore, linear types are fundamental to session typing.

Having successfully popularized the idea of using monads to structure effectful programs, Phil later observed a deep correspondence between monads and effect type systems [118]. Recently, his work has inspired the investigation of an important refinement of monads known as graded monads.

Phil has long been interested in the Curry-Howard correspondence between propositions in logic and types in programming languages. His CACM article [114] and regular invited talks at industry conferences are helping to make the Curry-Howard correspondence much more widely appreciated. As well as the Curry-Howard correspondence between second-order intuitionistic logic and polymorphic lambda calculus [106], a notable correspondence Phil has worked on recently is that between classic linear logic and session types [111].

Within the last ten years Phil has developed an interest in gradual typing—systems that combine statically typed programs with untyped programs in a controlled fashion. He introduced the blame calculus [116] (along with the slogan “well-typed programs cannot be blamed”), a formal language for tracking which untyped part of a program causes a run-time error in gradually typed programming languages. He continues to develop the theory of blame, for instance, extending it to account for new features such as polymorphism [1].

Biographical Details

Philip Lee Wadler was born in 1956. He received a Bachelor of Science degree in Mathematics from Stanford University in 1977, and a Master of Science degree in Computer Science from Carnegie Mellon University in 1979. He completed his Doctor of Philosophy in Computer Science at Carnegie Mellon University in 1984. His thesis, “Listlessness Is Better Than Laziness,” was supervised by Nico Habermann.

Between 1983 and 1987 Phil worked as a Research Fellow at St. Cross College in Oxford, working in the Programming Research Group with Mary Sheeran, Richard Bird, John Hughes, and others. (Richard and Phil co-authored the influential book *Introduction to Functional Programming* [4]). In 1987 Phil moved to the University of Glasgow to join Mary Sheeran and John Hughes, and shortly thereafter they attracted Simon Peyton Jones to join them. At Glasgow Phil was appointed Reader in 1990, and Professor in 1993. In 1996 he joined Bell Labs, Lucent Technologies, as a researcher. Subsequently, his group was spun out as part of Avaya Labs. In 2003 Phil took up his current position as Professor of Theoretical Computer Science in the Laboratory for Foundations of Computer Science, in the School of Informatics at the University of Edinburgh. In 2006, Phil wrote the O’Reilly book *Java Generics and Collections* [53], co-authored with his friend Maurice Naftalin.

Phil maintains an impressive group of post-docs, PhD students, and visitors. He has held visiting positions in Chalmers University of Technology and the Universities of Sydney and Copenhagen. He is regularly invited to speak at conferences across the world, both academic and industrial. Phil also regularly performs computer science-oriented stand-up comedy.

The quality of Phil’s research has been recognized by a plethora of awards, and some of the notable ones are as follows. He was elected to the Royal Society of Edinburgh in 2005, and as an ACM Fellow in 2007. He held a Wolfson-Royal Society Research Merit award between 2004 and 2009. Together with Simon Peyton Jones he was awarded the Most Influential POPL Paper Award in 2003 (for 1993) for their paper “Imperative Functional Programming” [39].

Throughout his career Phil has been passionate about teaching. At Edinburgh he successfully established the first-year Haskell course—Haskell is now the first programming language undergraduate students are exposed to. He has often taught more courses than required, and inspires students with his passion for the subject. This enthusiasm has been recognized, for example, in the Edinburgh University Student Association (EUSA) Teaching Awards 2009.

At the end of 2014, Phil fell ill with what turned out to be a serious illness. Initially, his symptoms were mysterious to doctors, so they performed a full range of tests. He had a serious, but treatable, heart and liver infection. Additionally, but unrelated, a small tumor on one of his kidneys was detected. Phil was fortunate to have the infection when he did, because otherwise the kidney tumor would probably have gone undetected until much later. After the infection had cleared, Phil had the offending kidney removed, and is now fully recovered.

Phil was supported by his wife Catherine from the time they met in Oxford in 1985 until their amicable separation in 2015. Phil and Catherine are devoted parents to their

twin son and daughter, Adam and Leora. Phil is actively engaged in wider society, for example, maintaining a liberal Jewish faith, being active in the University and College Union, and the Scottish Green Party.

January 2016

Sam Lindley
Conor McBride
Don Sannella
Phil Trinder

A List of Successes That Can Change the World
Essays Dedicated to Philip Wadler on the Occasion of
His 60th Birthday

Lindley, S.; McBride, C.; Trinder, P.; Sannella, D. (Eds.)
2016, XXII, 479 p. 79 illus., 4 illus. in color., Softcover
ISBN: 978-3-319-30935-4