Linear Haskell

Practical Linearity in a Higher-Order Polymorphic Language

JEAN-PHILIPPE BERNARDY, University of Gothenburg, Sweden
MATHIEU BOESPFLUG, Tweag I/O, France
RYAN R. NEWTON, Indiana University, USA
SIMON PEYTON JONES, Microsoft Research, UK
ARNAUD SPIWACK, Tweag I/O, France

Linear type systems have a long and storied history, but not a clear path forward to integrate with existing languages such as OCaml or Haskell. In this paper, we study a linear type system designed with two crucial properties in mind: backwards-compatibility and code reuse across linear and non-linear users of a library. Only then can the benefits of linear types permeate conventional functional programming. Rather than bifurcate types into linear and non-linear counterparts, we instead attach linearity to *function arrows*. Linear functions can receive inputs from linearly-bound values, but can *also* operate over unrestricted, regular values.

To demonstrate the efficacy of our linear type system — both how easy it can be integrated in an existing language implementation and how streamlined it makes it to write programs with linear types — we implemented our type system in GHC, the leading Haskell compiler, and demonstrate two kinds of applications of linear types: mutable data with pure interfaces; and enforcing protocols in I/O-performing functions.

CCS Concepts: • **Software and its engineering → Language features**; *Functional languages*; *Formal language definitions*;

Additional Key Words and Phrases: GHC, Haskell, laziness, linear logic, linear types, polymorphism, typestate

ACM Reference Format:
Jean-Philippe Bernardy, Mathieu Boespflug, Ryan R. Newton, Simon Peyton Jones, and Arnaud Spiwack. 2018. Linear Haskell: Practical Linearity in a Higher-Order Polymorphic Language. *Proc. ACM Program. Lang.* 2, POPL, Article 5 (January 2018), 36 pages. https://doi.org/10.1145/3158093

This paper appears in the Proceeding of the ACM Conference on Principles of Programming Languages (POPL) 2018. This version includes an Appendix that gives an operational semantics for the core language, and proofs of the metatheoretical results stated in the paper.

1 INTRODUCTION

Despite their obvious promise, and a huge research literature, linear type systems have not made it into mainstream programming languages, even though linearity has inspired uniqueness typing in Clean, and ownership typing in Rust. We take up this challenge by extending Haskell with linear

Authors' addresses: Jean-Philippe Bernardy, University of Gothenburg, Department of Philosophy, Linguistics and Theory of Science, Olof Wijksgatan 6, Gothenburg, 41255, Sweden, jean-philippe.bernardy@gu.se; Mathieu Boespflug, Tweag I/O, Paris, France, m@tweag.io; Ryan R. Newton, Indiana University, Bloomington, IN, USA, rrnewton@indiana.edu; Simon Peyton Jones, Microsoft Research, Cambridge, UK, simonpj@microsoft.com; Arnaud Spiwack, Tweag I/O, Paris, France, arnaud.spiwack@tweag.io.

2475-1421/2018/1-ART5
https://doi.org/10.1145/3158093