Challenges and Progress toward Efficient Gradual Typing (Invited Talk)

Jeremy Siek
Indiana University, USA

Abstract

Mixing static and dynamic type checking in the same language is catching on, with the TypeScript and Flow variants of JavaScript, the MyPy and Reticulated variants of Python, the Strongtalk and Gradualtalk variants of Smalltalk, as well as Typed Racket, Typed Clojure, and Perl 6. The gradual typing approach to such mixing seeks to protect the statically typed code from the dynamically typed code, allowing compilers to leverage type information when optimizing the static code. Unfortunately, ensuring soundness requires runtime checking at the boundaries of typed and untyped code, and the cost of this checking can drown out the performance benefits of optimization. For example, in Typed Racket, some partially typed programs are 1000X slower than the untyped or fully typed version of the same program. But all is not lost! In this talk I present the results of ongoing research to tame the runtime overheads of gradual typing in the context of a prototype compiler, named Grift, that we are developing at Indiana University.

CCS Concepts • Theory of computation → Type structures; • Software and its engineering → Compilers;

Keywords Gradual typing; performance; type soundness; dynamic typing.

ACM Reference Format:

Jeremy Siek. 2017. Challenges and Progress toward Efficient Gradual Typing (Invited Talk). In *Proceedings of 13th ACM SIGPLAN International Symposium on Dynamic Languages (DLS'17)*. ACM, New York, NY, USA, 1 page. <https://doi.org/10.1145/3133841.3148570>

Biography

Jeremy Siek is an Associate Professor at Indiana University Bloomington. Jeremy's areas of research include programming language design, type systems, mechanized theorem proving, and optimizing compilers. Jeremy's Ph.D. thesis explored foundations for constrained templates, aka the "concepts" proposal for C++. Prior to that, Jeremy developed the Boost Graph Library, a C++ generic library for graph algorithms and data structures. Jeremy post-doc'd at Rice University where he and Walid Taha developed the idea of gradual typing: a type system that integrates both dynamic and static typing in the same programming language. Jeremy is currently working on several open questions regarding gradual typing. Is it possible to create a high-performance implementation of a gradually-typed language? Is there an intuitive specification for when a higher-order cast should fail? In 2009 Jeremy received the NSF CAREER award to fund his project: "Bridging the Gap Between Prototyping and Production". In 2010 and again in 2015, Jeremy was awarded a Distinguished Visiting Fellowship from the Scottish Informatics and Computer Science Alliance. From 2015 to the present Jeremy has been collaborating on the NSF-funded project Gradual Typing Across the Spectrum.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

DLS'17, October 24, 2017, Vancouver, Canada

© 2017 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5526-1/17/10...\$15.00

<https://doi.org/10.1145/3133841.3148570>