Interpretations of the Gradually-Typed Lambda Calculus

(Distilled Tutorial)

Jeremy G. Siek

University of Colorado at Boulder

jeremy.siek@colorado.edu

Abstract

Gradual typing is an approach to integrating static and dynamic type checking within the same language [Siek and Taha, 2006]. Given the name "gradual typing", one might think that the most interesting aspect is the type system. It turns out that the dynamic semantics of gradually-typed languages is more complex than the static semantics, with many points in the design space [Wadler and Findler, 2009, Siek et al., 2009] and many challenges concerning efficiency [Herman et al., 2007, Hansen, 2007, Siek and Taha, 2007, Siek and Wadler, 2010, Wrigstad et al., 2010, Rastogi et al., 2012]. In this distilled tutorial, we write several definitional interpreters and abstract machines in Scheme, some of which are new, exploring the meaning of gradual typing and the challenges to efficient implementation.

Categories and Subject Descriptors D.3.3 [*Language Constructs and Features*]: Procedures, functions, and subroutines

General Terms Languages, Theory

Keywords casts, coercions, blame tracking, lambda-calculus, Scheme

References

L. T. Hansen. Evolutionary programming and gradual typing in ECMAScript 4 (tutorial). Technical report, ECMA TG1 working group, November 2007.

D. Herman, A. Tomb, and C. Flanagan. Space-efficient gradual typing. In *Trends in Functional Prog. (TFP)*, page XXVIII, April 2007.

A. Rastogi, A. Chaudhuri, and B. Hosmer. The ins and outs of gradual type inference. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '12, pages 481–494, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1083-3. doi: 10.1145/2103656.2103714. URL `http://doi.acm.org/10.1145/2103656.2103714`.

J. G. Siek and W. Taha. Gradual typing for functional languages. In *Scheme and Functional Programming Workshop*, pages 81–92, September 2006.

J. G. Siek and W. Taha. Gradual typing for objects. In *ECOOP 2007*, volume 4609 of *LCNS*, pages 2–27. Springer Verlag, August 2007.

J. G. Siek and P. Wadler. Threesomes, with and without blame. In *POPL '10: Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 365–376, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-479-9.

J. G. Siek, R. Garcia, and W. Taha. Exploring the design space of higher-order casts. In *European Symposium on Programming*, March 2009.

P. Wadler and R. B. Findler. Well-typed programs can't be blamed. In *European Symposium on Programming*, 2009.

T. Wrigstad, F. Z. Nardelli, S. Lebresne, J. Östlund, and J. Vitek. Integrating typed and untyped code in a scripting language. In *POPL '10: Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 377–388, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-479-9.

Scheme and Functional Programming 2012 September, Copenhagen Denmark.