Can a type system emulate the ergonomics of structural subtyping while being unification based?

Ayaz Hafiz
ayaz.hafiz.1@gmail.com

Abstract

In our development of the Roc programming language, we have found that structural subtyping (a-la TypeScript, ℂDuce) is a model that many developers are able to intuit and be productive in. However, Hindley-Milner-style type systems based on unification have numerous benefits too - they readily admit principal type inference, and ease the effort of implementing a monomorphizing compiler for a program when unification induces type equality. Systems like ML-Sub [1] and Simple-sub [2] demonstrate that principal type inference can be recovered in such systems extended with subtyping, though their support for subtyping constructs is limited. Recently, it has been shown that a form of subtyping can be encoded via Rank-1 parametric polymorphism, but that no record subtyping can be encoded as row polymorphism [3]. We would like to discuss what opportunities we have as designers and implementers of type systems to emulate subtyping to a great degree in HM-style type systems, both faithfully and only partially.

We will discuss to what extent row and variant polymorphism can aid these goals. To what extent can other tricks, like implicit unbound type variables in positive (output) positions, help us emulate the feel of subtyping in a structural HM-style type system? What opportunities do we have to implement type systems that feature subtyping, and automatically translate them to an HM-based system? What are the implications on error messages and diagnostic suggestions that can be provided to users in both kinds of systems?

ACM Reference Format:
Ayaz Hafiz. 2024. Can a type system emulate the ergonomics of structural subtyping while being unification based?. In *Proceedings of Workshop on the Implementation of Type Systems (Conference acronym 'XX)*. ACM, New York, NY, USA, 1 page. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, January 20, 2024, London, United Kingdom

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

References

[1] Stephen Dolan and Alan Mycroft. 2017. Polymorphism, Subtyping, and Type Inference in MLsub. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages* (Paris, France) *(POPL '17)*. Association for Computing Machinery, New York, NY, USA, 60–72. https://doi.org/10.1145/3009837.3009882

[2] Lionel Parreaux. 2020. The Simple Essence of Algebraic Subtyping: Principal Type Inference with Subtyping Made Easy (Functional Pearl). *Proc. ACM Program. Lang.* 4, ICFP, Article 124 (aug 2020), 28 pages. https://doi.org/10.1145/3409006

[3] Wenhao Tang, Daniel Hillerström, James McKinna, Michel Steuwer, Ornela Dardha, Rongxiao Fu, and Sam Lindley. 2023. Structural Subtyping as Parametric Polymorphism. *Proc. ACM Program. Lang.* 7, OOPSLA2, Article 260 (oct 2023), 29 pages. https://doi.org/10.1145/3622836