Emulating algebraic effects with polymorphic variants

Ayaz Hafiz
ayaz.hafiz.1@gmail.com

Abstract

Roc is a new pure functional language that models effects via the continuation monad. Our design for effects composes into a state machine that can be executed by an arbitrary external party, and supports both sequential and parallel runtime execution of effects. Effects are traced in the type system as polymorphic variants. This design decision has enabled effects to compose naturally via type unifications, without the need for machinery like monad transformers. This composition provides ergonomics similar to those of algebraic effects, but avoids needing an effect system disjoint from the type system or support for GADTs. Our system can serve as the compilation target for an alegbraic effect system.

Our type system infers what function values can inhabit instances of function types across a program via lambda set specialization [1], enabling our implementation to guarantee defunctionalization. This guarantee means that continuations in composed effects typically require no allocation for their closures, and are always compiled to direct calls.

ACM Reference Format:
Ayaz Hafiz. 2024. Emulating algebraic effects with polymorphic variants. 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

References

[1] William Brandon, Benjamin Driscoll, Frank Dai, Wilson Berkow, and Mae Milano. 2023. Better Defunctionalization through Lambda Set Specialization. *Proc. ACM Program. Lang.* 7, PLDI, Article 146 (jun 2023), 24 pages. https://doi.org/10.1145/3591260

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