Polymorphic algebraic effects: theoretical properties and implementation

Wiktor Kuchta

Chapter 1

Introduction

A programming language needs to be more than just a lambda calculus, capable only of functional abstraction and evaluation of expressions. Programs need to have an effect on the outside world and, thinking more locally, in our programs we would like to have fragments which do not merely reduce to a value in isolation, but affect the surrounding code in interesting ways. This is what we call computational effects, the typical examples of which are: input/output, mutable state, exceptions, nondeterminism, and coroutines.

Today, we are at the mercy of programming language designers to include the effects we would like to use and make sure that they all interact with each other well. In recent years, however, a new approach, called *algebraic effects*, has emerged. Algebraic effects allow programmers to express all the usual computational effects as library code and use them in the usual direct style (as opposed to eg. monadic encodings, which do not always play well with the rest of the language and aren't easily composed). These implementations of effects are instances of one concept and hence interact with each other predictably.

More concretely, algebraic effects may be thought of as resumable exceptions. We can *perform an operation*, just as we can raise an exception in a typical programming language. The operation is then handled by the nearest enclosing *handler* for the specific operation. The handler receives the value given at perform-point and also, unlike normal exception handlers, a *continuation*, a first-class functional value representing the rest of the code to execute inside the handler from the perform-point. By calling the continuation with a value we can resume at the perform-point, as if performing the operation evaluated to the value. But more interestingly, we can resume multiple times, or never, or store the continuation for later use.

This access to the continuation, while powerful, might also be a recipe for disaster. That's why languages with algebraic effects typically have *type-and-effect systems*, not only tracking the type of value an expression might evaluate to, but also what kind of effects it may perform on the way.

Chapter 2

The logical relation

The logical relation is adapted from [1] with a few changes: our relation is unary, we additionally have polymorphic effects, we don't have type lambdas.

$$\begin{aligned}[\![\mathsf{T}]\!] &= \mathcal{P}(\mathsf{Val}) = \mathsf{Type} \\ [\![\mathsf{E}]\!] = [\![\mathsf{R}]\!] &= \mathcal{P}(\mathsf{Exp} \times \mathbb{N} \times \mathsf{Type}) = \mathsf{Eff}\end{aligned}$$

$$\mathrm{Obs} = \{e \mid \exists v.\, e \to^* v\}$$

$$\begin{aligned}\mathcal{E}[\![\tau/\rho]\!]_\eta &= \{t \mid \forall K \in \mathcal{K}[\![\tau/\rho]\!]_\eta.\, K[t] \in \mathrm{Obs}\} \\ \mathcal{K}[\![\tau/\rho]\!]_\eta &= \{K \mid \forall v \in [\![\tau]\!]_\eta.\, K[v] \in \mathrm{Obs} \wedge \forall s \in \mathcal{S}[\![\tau/\rho]\!]_\eta.\, K[s] \in \mathrm{Obs}\} \\ \mathcal{S}[\![\tau/\rho]\!]_\eta &= \{K[\mathsf{do}\, v] \mid \exists n, \mu.\, (v, n, \mu) \in [\![\rho]\!]_\eta \wedge n\text{-free}(K) \wedge \forall e \in \mu.\, K[e] \in \mathcal{E}[\![\tau/\rho]\!]_\eta\}\end{aligned}$$

$$\begin{aligned}[\![\tau_1 \to_\rho \tau_2]\!]_\eta &= \{\lambda x.\, t \mid \forall v \in [\![\tau_1]\!].\, [x \mapsto v]t \in \mathcal{E}[\![\tau_2/\rho]\!]_\eta\} \\ [\![\forall \alpha :: \kappa.\, \tau]\!]_\eta &= \{v \mid \forall \mu \in [\![\kappa]\!].\, v \in [\![\tau]\!]_{[\alpha \mapsto \mu]\eta}\} \\ [\![\alpha]\!]_\eta &= \{v \mid \forall K.(\forall u \in \eta(\alpha).K[u] \in \mathrm{Obs}) \implies K[v] \in \mathrm{Obs}\}\end{aligned}$$

$$\begin{aligned}[\![\tau_1 \Rightarrow \tau_2]\!]_\eta &= \{(v, 0, [\![\tau_2]\!]_\eta) \mid v \in [\![\tau_1]\!]_\eta\} \\ [\![\forall \alpha :: \kappa.\, \varepsilon]\!]_\eta &= \{t \mid \exists \mu \in [\![\kappa]\!].\, t \in [\![\varepsilon]\!]_{[\alpha \mapsto \mu]\eta}\} \\ [\![\varepsilon \cdot \rho]\!]_\eta &= [\![\varepsilon]\!]_\eta \cup \{(v, n+1, \mu) \mid (v, n, \mu) \in [\![\rho]\!]_\eta\}\end{aligned}$$

$$\begin{aligned}\mathcal{E}[\![\tau/\rho]\!]_\eta(X) &= \{t \mid \forall K \in \mathcal{K}[\![\tau/\rho]\!]_\eta(X).\, K[t] \in \mathrm{Obs}\} \\ \mathcal{K}[\![\tau/\rho]\!]_\eta(X) &= \{K \mid \forall v \in [\![\tau]\!]_\eta.\, K[v] \in \mathrm{Obs} \wedge \forall s \in \mathcal{S}[\![\tau/\rho]\!]_\eta(X).\, K[s] \in \mathrm{Obs}\} \\ \mathcal{S}[\![\tau/\rho]\!]_\eta(X) &= \{K[\mathsf{do}\, v] \mid \exists n, \mu.\, (v, n, \mu) \in [\![\rho]\!]_\eta \wedge n\text{-free}(K) \wedge \forall e \in \mu.\, K[e] \in X\}\end{aligned}$$

Bibliography

[1] Dariusz Biernacki, Maciej Piróg, Piotr Polesiuk, and Filip Sieczkowski. Handle with care: relational interpretation of algebraic effects and handlers. *Proc. ACM Program. Lang.*, 2(POPL):8:1–8:30, 2018.