A Denotational Semantics for Polymorphic Effect Systems

Part III Project

Alexander Taylor, at736

University of Cambridge

April 10, 2019

Introduction Slide

- code example - "apply" function + three different invocations - Launch Missile - Throw Error - Read environment variables

What is denotational Semantics?

- Type relation instance - mapping function ($[\![-]\!]$) - compositional, sound, adequate? - equivalence $\Leftrightarrow$ equal denotations

Denotational Semantics using Category Theory

(Objects, Morphisms, etc)

Language features (1)

cartesian closed categories - pairs, unit, and functions

Language features (2)

Monads, graded monads

- diagrams, natural transformations

Language Features (3)

Subtyping, Subeffecting, If-Expressions

- If expression example - Co-product diagram

An Effectful Language

EC Syntax + example program

Semantics of EC

- example of some denotational rules - return? - lambda? - bind? - S-Category - definition

An Ugly Example

- Example of a program that would benefit from polymorphism.

Let's add polymorphism

- PEC Syntax, Type System (Particularly Gen and Spec rules)

An Ugly Example - With a Makeover

- Example of a program that would benefit from polymorphism.

How do we Model the Semantics of a Polymorphic Language?

- For a given effect variable environment Φ, excluding the polymorphic terms, we have EC, which there exist models for. - Effect-variable environments of length n are isomorphic by α-equivalence

How do we Model the Semantics of a Polymorphic Language?

- Stack of S-categories and their morphisms

- type rule for generalisation - "Need functors"

Base Category

- We need a way of reasoning about effect-variable environment categorically

- We can model effects and environments in new category.

- Objects: $\mathtt{1}$, U, U^n (write I for U^n) - Morphisms: $[\![e]\!] : \mathtt{1} \to U$ - Monoidal operator $\mathtt{Mul} : \mathbb{C}(I, U) \times \mathbb{C}(I, U) \to \mathbb{C}(I, U)$ - Can represent each effect environment as an object I, and common transformations between environments, such as weakening and substitutions, are morphisms between effect environments.

Indexed Category

- full index diagram with fibres, re-indexing functor

Quantification

- Quantification functor definition

Instantiating a Model (1)

final indexed category construction

- Can we actually instantiate a category with the required structure?

- Models of particular instantiations of EC based on `Set` exist.

- Next step is use a `Set`-based model to build a model of EC

Instantiating a Model (2) - Base Category

- Category of monotone functions of ground effects (with no variables) to ground effects. - $[\![\diamond, \alpha \vdash \alpha \cdot \mathtt{IO} \colon \mathtt{Effect}]\!] = e \mapsto e \cdot \mathtt{IO}$ - $\mathtt{Mul}(f, g)\vec{\epsilon} = (f\vec{\epsilon}) \cdot (g\vec{\epsilon})$

Instantiating a Model (3) - Fibres

- The fibre $\mathbb{C}(n)$ is the category of functors $[E^n, \mathtt{Set}]$ - I.E. objects are functions that take a vector of ground effects and return sets - Morphisms are functions that return functions in $\mathtt{Set}$ - S-Category features (products, exponentials, graded monad) can be constructed pointwise (Graded monad definition)

Instantiating a Model (4) - Functors and Adjunctions

- Re-indexing functors are formed by pre-composition - $\forall_{E^n}$ functor is formed by a product over all effects. - Adjunction operations become pairing and projection.

The End

- Dissertation and github links