A Denotational Semantics for Polymorphic Effect Systems

Part III Project

Alexander Taylor, at736

University of Cambridge

May 24, 2019

Motivating Polymorphic Effect Analysis

```
def logAction(
    action: Unit => String
): Unit {
    log.info(action())
}
```

```
logAction(() => (FireMissiles(); "Launched Missiles"))

logAction(() => throwError("My Error"))

logAction(() => readEnvironmentVariables)
```

What is Categorical Denotational Semantics?

- A *compositional* mapping
  $[\![-]\!]$: Language Structure $\to$ Categorical Structure

- Types and type environments map to objects $[\![A]\!] \in \mathtt{obj}\ (\mathbb{C})$

- Well typed terms map to morphisms (arrows) $[\![\Gamma \vdash t \colon A]\!] : [\![\Gamma]\!] \to [\![A]\!]$

- Needs to be *sound* $t_1 \approx t_2 \implies [\![t_1]\!] = [\![t_2]\!]$

- And *adequate* $[\![t_1]\!] = [\![t_2]\!] \implies t_1 \approx t_2$

Contributions

- A sound set of requirements for denotational semantics of effect-polymorphic languages.
- A method to construct models for effect-polymorphic languages in $\mathtt{Set}$.
- A proof of adequacy of such a model.

Language features (2.B) - Graded Monads

A (strong) graded monad consists of:

- An indexed functor $T_\epsilon : \mathbb{C} \to \mathbb{C}$
- Indexed Join and Unit natural transformations
  - $\mu_{\epsilon_1,\epsilon_2,A} : T_{\epsilon_1} T_{\epsilon_2} A \to T_{\epsilon_1 \cdot \epsilon_2} A$
  - $\eta_A : A \to T_1 A$
- Tensor strength natural transformation $\mathtt{t}_{\epsilon,A,B} : A \times T_\epsilon B \to T_\epsilon (A \times B)$

An Effectful Language

$$v ::= \mathtt{k}^{A} \mid x \mid \mathtt{true} \mid \mathtt{false} \mid () \mid \lambda x\colon A.v \mid v_1\ v_2 \mid \mathtt{return}\ v$$
$$\mid \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \mid \mathtt{if}_{A}\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2$$

$$A, B, C ::= \gamma \mid A \to B \mid \mathtt{M}_{\epsilon} A$$

$$(\text{Return})\frac{\Gamma \vdash v\colon A}{\Gamma \vdash \mathtt{return}\ v : \mathtt{M}_{1} A} \qquad (\text{Apply})\frac{\Gamma \vdash v_1\colon A \to B \qquad \Gamma \vdash v_2\colon A}{\Gamma \vdash v_1\ v_2\colon B}$$

An Effectful Language

$$v ::= \mathtt{put}^{\mathtt{M}_1 \mathit{Unit}} \mid x \mid \mathtt{true} \mid \mathtt{false} \mid () \mid \lambda x\colon A.v \mid v_1\ v_2 \mid \mathtt{return}\ v$$
$$\mid \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \mid \mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2$$

$$A, B, C ::= \mathtt{Bool} \mid \mathtt{Unit} \mid A \to B \mid \mathtt{M}_n A$$

$$(\text{Return})\frac{\Gamma \vdash v\colon A}{\Gamma \vdash \mathtt{return}\ v : \mathtt{M}_0 A} \quad (\text{Apply})\frac{\Gamma \vdash v_1\colon A \to B \qquad \Gamma \vdash v_2\colon A}{\Gamma \vdash v_1\ v_2\colon B}$$

$$\mathtt{M}_n A \leq: \mathtt{M}_{n+1} A$$

Semantics of EC

- Can build a model of EC when we have
  - CCC
  - Strong Graded Monad
  - Co-product and Subtyping (morphisms for if-statements)

- We'll call this an S-category

$$(\text{Return})\frac{f = [\![\Gamma \vdash v\colon A]\!]}{[\![\Gamma \vdash \mathtt{return}\ v : \mathtt{M}_1 A]\!] = \eta_A \circ f} \quad (\text{Fn})\frac{f = [\![\Gamma, x\colon A \vdash v\colon B]\!] : \Gamma \times A \to B}{[\![\Gamma \vdash \lambda x\colon A.v : A \to B]\!] = \mathrm{cur}(f) : \Gamma \to B^A}$$

Semantics of EC

- Can build a model of EC with sets and functions

- Types map to sets
  - $[\![()]\!] = \{*\}$, $[\![\texttt{Bool}]\!] = \{\top, \bot\}$
  - $[\![A \to B]\!] = [\![A]\!] \implies [\![B]\!]$
  - $[\![\texttt{M}_n A]\!] = \{(n', a) \mid a \in [\![A]\!], n' \leq n\}$

- Well typed terms map to functions
  - $[\![\Gamma \vdash \texttt{true}\colon \texttt{Bool}]\!] = \rho \mapsto \top$
  - $[\![\Gamma \vdash \texttt{put}\colon \texttt{M}_1\texttt{Unit}]\!] = \rho \mapsto (1, *)$
  - $[\![\Gamma \vdash \texttt{do } x \leftarrow v_1 \texttt{ in } v_2 :]\!] = \rho \mapsto$
    $$\begin{aligned} &\text{let } (n', a) = [\![\Gamma \vdash v_1\colon \texttt{M}_n A]\!](\rho) \\ &\text{and } (m', b) = [\![\Gamma, x\colon A \vdash v_2\colon \texttt{M}_m B]\!])(\rho, a) \\ &\text{in } (m' + n', b) \end{aligned}$$

TODO: needs to be indexed by n instead

An Ugly Example

```
let twiceIO = λ action: M_IO Unit. (
    do _ <- action in action
)

let twiceState = λ action: M_State Unit. (
    do _ <- action in action
)
do _ <- twiceState(increment) in twiceIO(writeLog)
```

Let's Add Polymorphism

$$v ::= .. \mid \Lambda\alpha.v \mid v\ \epsilon$$

$$A, B, C ::= ... \mid \forall\alpha.A$$

$$\epsilon ::= e \mid \alpha \mid \epsilon \cdot \epsilon$$

$$(\text{Effect-Gen})\frac{\Phi, \alpha \mid \Gamma \vdash v \colon A}{\Phi \mid \Gamma \vdash \Lambda\alpha.v \colon \forall\alpha.A} \qquad (\text{Effect-Spec})\frac{\Phi \mid \Gamma \vdash v \colon \forall\alpha.A \qquad \Phi \vdash \epsilon}{\Phi \mid \Gamma \vdash v\ \epsilon \colon A[\epsilon/\alpha]}$$

An Ugly Example - With a Makeover

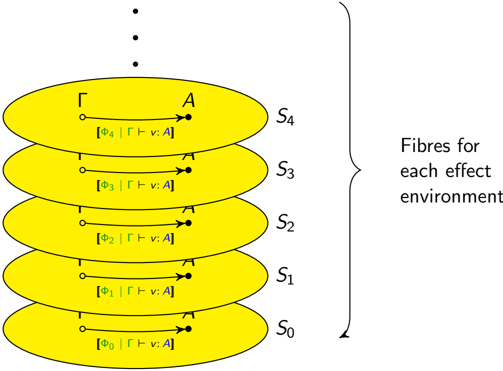
```
let twice = Λ eff.(
    λ action: M_eff Unit. (
        do _ <- action in action
    )
)

do _ <- (twice State increment) in (twice IO writeLog)
```

How do we Model the Semantics of a Polymorphic Language?

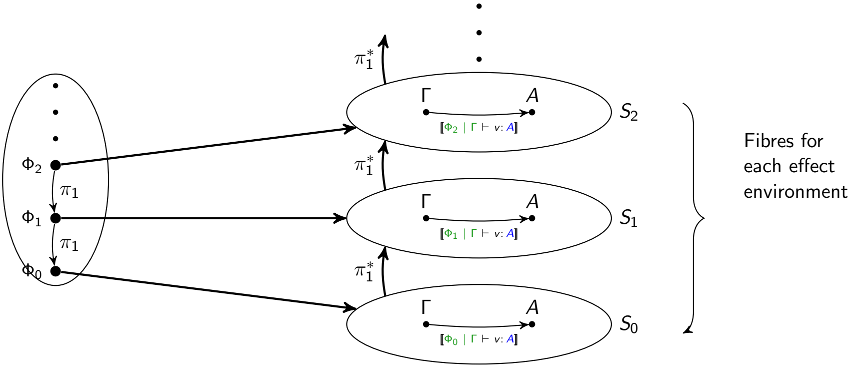
- For a fixed effect variable environment Φ and terms with no polymorphic sub-terms, we have EC
- Effect-variable environments of length n are isomorphic by α-equivalence

How do we Model the Semantics of a Polymorphic Language?

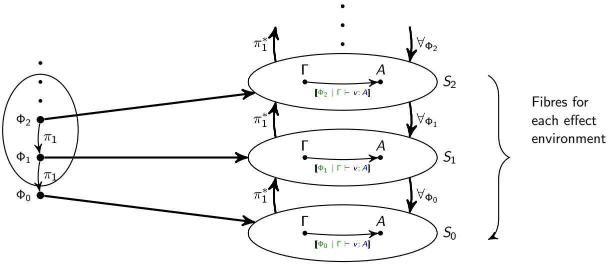
- So we instantiate an S-category for each environment.
- The type rule for quantification requires us to move between categories **TODO: Type rule here**.
- Functors are required.



Indexed Category



Instantiating a Model (1)



- Can we actually instantiate a category with the required structure?
- Starting point: a model of EC in Set

Instantiating a Model (2) - 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 a set $[\![\Phi \vdash A\colon \mathtt{Type}]\!] : E^n \to \mathtt{obj}\ \ (\mathtt{Set})$.

- Morphisms are dependent functions that return functions in $f : (\vec{\epsilon} : E^n) \to A\vec{\epsilon} \to B\vec{\epsilon}$

- These fibres have S-Category features

Instantiating a Model (4) - Functors and Adjunctions

- Re-indexing functors act by pre-composition

$$\begin{aligned} A \in & \quad [E^n, \texttt{Set}] \\ \theta^*(A)\vec{\epsilon_m} = & \quad A(\theta(\vec{\epsilon_m})) \\ \theta^*(f)\vec{\epsilon_m} = & \quad f(\theta(\vec{\epsilon_m})) : \theta^*(A) \to \theta^*(B) \end{aligned}$$

- The quantification functor takes a product over all ground effects

$$\forall_{E^n}(A)\vec{\epsilon_n} = \Pi_{\epsilon \in E} A(\vec{\epsilon_n}, \epsilon)$$

The End

- Sound: Proved for all indexed S-Categories ✓
- Compositional: By the definition of denotations ✓
- Adequate: Proved for an instantiation in `Set` ✓

Dissertation:

`https://github.com/Al153/Part3Project/tree/master/diss`