Abstract

To date, there has been limited work on the semantics of languages with polymorphic effect systems. The application, by Moggi, of strong monads to modelling the semantics of effects has become a mainstream concept in functional programming languages. This was improved upon by Lucasson (?) using a graded monad to model languages with a range of independent and dependent effects at an operational level. A categorical semantics for parametric polymorphism in types was first published by Reynolds (?) allowing a denotational analysis of languages including type parameters. There has been some work on polymorphism over the exception effect (which paper). Despite these works, there has been no work to date on the denotational semantics of languages with general parametric polymorphism over effects.

In this dissertation, I present several pieces of work. Firstly, I shall introduce a modern definition of a lambda-calculus based language with an explicit graded monad to handle a variety of effects. This calculus shall then be extended using polymorphic terms to yield a more general polymorphic-effect-calculus. I shall then give an indexed-category-based denotational semantics for the language, along with an outline of a proof for the soundness of these semantics. Following this, I shall present a method of transforming a model of a non-polymorphic language into a model of the language with polymorphism over effects.

The full proofs can be found online on my github repository (link), since due to the number of theorems and cases, the total size is well over 100 pages of definitions, theorems, and proofs.

Chapter 1

Introduction

1.1 What is Effect Polymorphism?

Effect polymorphism is when the same function in a language can operate on values of similar types but with different effects. It allows the same piece of code to be used in multiple contexts with different type signatures. This manifests in a similar manner to type parameter polymorphism in system-F based languages. Consider the following Scala-style pseudo-code:

```
def check[E: Effect](
    action: Unit => (Unit;e)
): Unit; (IO, e) {
    val ok: Boolean = promptBool(
        "Are you sure you want to do this?"
    )
    If (ok) {
        action()
    } else {
       abort()
    }
}
```

```
check[RealWorld](() => check[RealWorld](FireMissiles))
check[Transaction](SendMoney(Bob, 100, USD))
check[Exception](ThrowException("Not Aborted"))
```

In this example, we are reusing the same “check” function in three different cases with three different effects in a type safe manner. Hence, "check" is polymorphic in the effect parameter it receives. To analyse this language, it would be useful to have an analysis tool that can precisely model these separate, though potentially interdependent effects. A denotational semantics that can account for the parametric polymorphism over effects would be a step towards building such tools.

1.2 An Introduction to Categorical Semantics

In this dissertation, I shall be describing a denotational semantics using category theory. A denotational semantics for a language is a mapping, known as a denotation, $[\![-]\!]_M$, of structures in the language, such as types and terms to mathematical objects in such a way that non-trivial properties of the terms in the language correspond to other properties of the denotations of the terms.

When we specify a denotational semantics of a language in category theory, we look to find a mapping of types and typing environments to objects in a given category.

$$A : \mathtt{Type} \mapsto [\![A]\!]_M \in \mathtt{obj}\ \ \mathbb{C} \tag{1.1}$$

$$\Gamma \mapsto [\![\Gamma]\!]_M \in \mathtt{obj}\ \ \mathbb{C} \tag{1.2}$$

Further more, instances of the type relation should be mapped to morphisms between the relevant objects.

$$\Gamma \vdash v\colon A \mapsto \mathbb{C}([\![\Gamma]\!]_M, [\![A]\!]_M) \tag{1.3}$$

This should occur in a sound manner. That is, for every instance of the $\beta\eta$-equivalence relation between two terms, the denotations of the terms should be equal in the category.

$$\Gamma \vdash v_1 =_{\beta\eta} v_2\colon A \implies [\![\Gamma \vdash v_1\colon A]\!]_M = [\![\Gamma \vdash v_2\colon A]\!]_M \tag{1.4}$$

An example of $\beta\eta$-equivalence is that of the β-reduction of lambda terms. It should be the case that:

$$(\text{Lambda-Beta})\frac{\Gamma, x : A \vdash v_1\colon B \quad \Gamma \vdash v_2\colon A}{\Gamma \vdash (\lambda x : A.v_1)\ v_2 =_{\beta\eta} v_1\,[v_2/x] : B} \tag{1.5}$$

Means that the denotations $[\![\Gamma \vdash (\lambda x : A.v_1)\ v_2\colon B]\!]_M$ and $[\![\Gamma \vdash v_1\,[v_2/x] : B]\!]_M$ are equal.

To prove soundness, we perform rule induction over the derivation of the $\beta\eta$-equivalence relation, such as the lambda-beta-reduction rule above.

Some of the inductive cases require us to quantify what the substitution of terms for variables, such as $[v_1/x]$ does to the denotations of the parent term. Substitution theorems, allow us to quantify this action on denotations in a category theoretic way.

If

$$\Gamma, x : A \vdash v_1\colon B \tag{1.6}$$

And

$$\Gamma \vdash v_2\colon A \tag{1.7}$$

Then

$$[\![\Gamma \vdash v_1\,[v_2/x] : B]\!]_M \tag{1.8}$$

Should be derivable from

$$[\![\Gamma, x : A \vdash v_1 \colon B]\!]_M \tag{1.9}$$

And

$$[\![\Gamma \vdash v_2 \colon A]\!]_M \tag{1.10}$$

A similar concept is that of environment weakening. $\Gamma, x : A$ can derive every typing relation that Γ can, if x is not already in the environment Γ. Hence, $\Gamma, x : A$ is an example of a typing environment that is *weaker* than Γ. A weakening theorem proves that there is a systematic way to generate the denotation of a typing-relation on a term in a weaker environment from the denotation of the same term in a stronger environment.

If

$$\Gamma' \leq_{\text{weaker}} \Gamma \tag{1.11}$$

Then

$$[\![\Gamma' \vdash v \colon A]\!]_M \tag{1.12}$$

should be derivable from

$$[\![\Gamma \vdash v \colon A]\!]_M \tag{1.13}$$

1.2.1 Languages and Their Requirements

Different languages require different structures to be present in a category for the category to be able to interpret terms in the language. Using the concepts defined in 2, I shall now give an introduction to which category-theoretic structures are required to interpret different language features.

One of the simplest, while still interesting, languages to derive a denotational semantics for is the simply typed lambda calculus (STLC). STLC's semantics require a cartesian closed category (CCC, see section 2.1).

Products in the CCC are used to denote the lists of variable types in the typing environment, exponential objects model functions, and the terminal object is used to derive representations of ground terms, such as the unit term, (), as well as the empty typing environment.

- Products are used to construct type environments. $[\![\Gamma]\!]_M = [\![\diamond, x : A, y : B, ...z : C]\!]_M = \mathtt{1} \times [\![A]\!]_M \times [\![B]\!]_M \times ... \times [\![C]\!]_M$
- Terminal objects are used in the denotation of constant terms $[\![\Gamma \vdash \mathtt{C}^A \colon A]\!]_M = [\![\mathtt{C}^A]\!]_M \circ \langle\rangle_{[\![\Gamma]\!]_M}$
- Exponentials are used in the denotations of functions. $[\![\Gamma \vdash \lambda x : A.v \colon A \to B]\!]_M = \mathtt{cur}([\![\Gamma, x : A \vdash v \colon B]\!]_M)$

From this, we can specify what structures categories need to have in order to model more complex languages.

Language Feature	Structure Required
STLC	CCC
If expressions and booleans	Co-product on the terminal object
Single Effect	Strong Monad
Multiple Effects	Strong Graded Monad
Polymorphism	Indexed Category

A single effect can be modelled by adding a strong monad to the category, as shown by Moggi **TODO: Reference**. The monad allows us to generate a unit effect and to compose multiple instances of the effect together in a way that intuitively matches the type system of a monadic language.

$$\text{(Return)}\frac{\Gamma \vdash v\colon A}{\Gamma \vdash \mathtt{return} v\colon \mathtt{M}A} \quad \text{(Bind)}\frac{\Gamma \vdash v_1 : \mathtt{M}A \quad \Gamma, x : A \vdash v_2 : \mathtt{M}B}{\Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 : \mathtt{M}B} \tag{1.14}$$

These type rules can be modelled using the "unit" natural transformation and a combination of the "join" and tensor strength natural transformations respectively.

For a more precise analysis of languages with multiple effects, we can look into the algebra on the effects. A simple example of such an algebra is a partially ordered monoid. The monoid operation defines how to compose effects, and the partial order gives a sub-typing relation to make programming more intuitive with respect to if statements. A category with a strong graded monad allows us to model this algebra in a category theoretic way. It also allows us to do some effect analysis in the type system, as seen in the type rules for return and bind in equation 1.15.

$$\text{(Return)}\frac{\Gamma \vdash v\colon A}{\Gamma \vdash \mathtt{return} v\colon \mathtt{M}_{\mathtt{1}}A} \quad \text{(Bind)}\frac{\Gamma \vdash v_1 : \mathtt{M}_{\epsilon_1}A \quad \Gamma, x : A \vdash v_2 : \mathtt{M}_{\epsilon_2}B}{\Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 : \mathtt{M}_{\epsilon_1 \cdot \epsilon_2}B} \tag{1.15}$$

To express polymorphism over a property P, the language's semantics are expanded to use a new environment specifying the variables ranging over P that are allowed in a given context. This can be seen in the augmented type rules in 1.16.

$$\text{(Gen)}\frac{\Phi, \alpha \mid \Gamma \vdash v\colon A}{\Phi \mid \Gamma \vdash \Lambda\alpha.v\colon \forall\alpha.A} \quad \text{(Spec)}\frac{\Phi \mid \Gamma \vdash v : \forall\alpha.A \quad \Phi \vdash \epsilon}{\Phi \mid \Gamma \vdash v\ \epsilon : A[\epsilon/\alpha]} \tag{1.16}$$

To model these augmented type rules, we can create a category representing the semantics (an S-Category) of the non-polymorphic language at each given context. This collection of non-polymorphic categories can be indexed by a base category which models the operations and relationships between the P-Environment. Morphisms in the base category between environments correspond to re-indexing functors between the S-categories for the relevant environments. These functors, and a right adjoint for the re-indexing functor corresponding to π_1 morphism can then be used to construct the semantics of polymorphic terms. Figure 1.2.1 demonstrates this construction.

In this dissertation, I shall show how these category theoretic building blocks can be put together to give the class of categories that can model polymorphic effect systems.

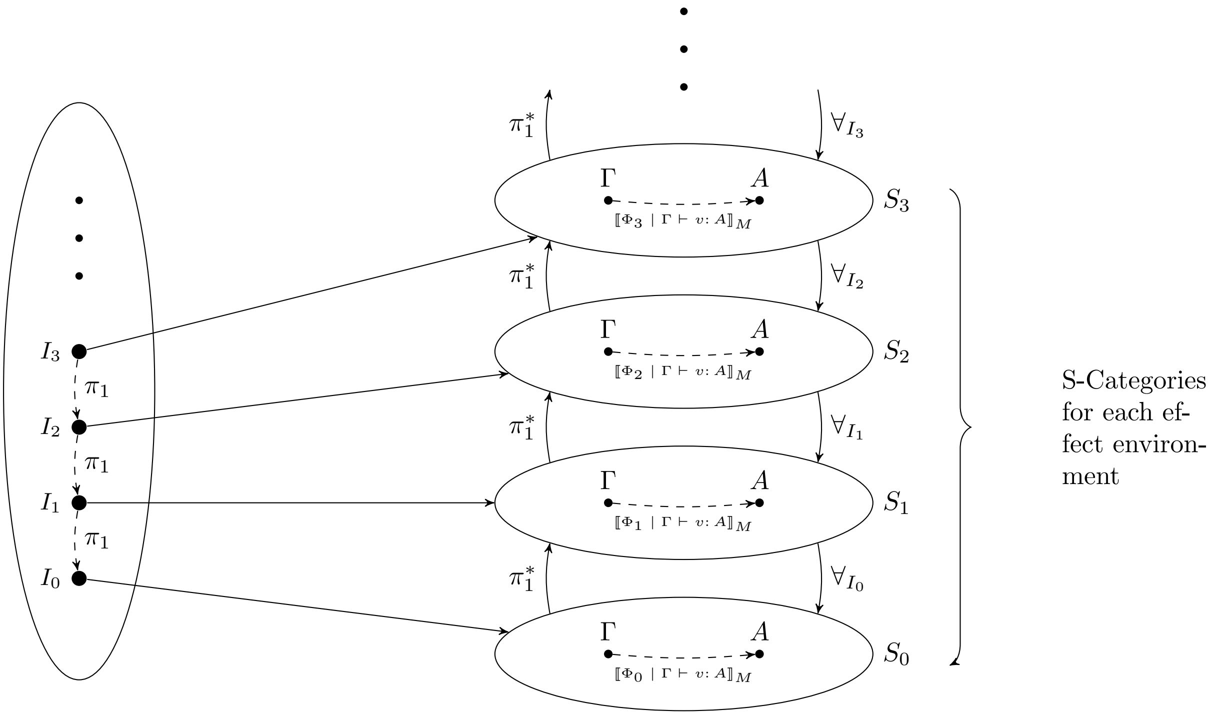

Figure 1.1: Diagram of the structure of an indexed category for modelling a polymorphic language. Solid arrows represent functors and dashed arrows represent internal morphisms. The left hand category is the base category.

Chapter 2

Required Category Theory

Before going further, it is necessary to assert a common level of category theory knowledge. This section is not intended as a tutorial but to jog the memory of the reader, and briefly introduce some new concepts.

2.1 Cartesian Closed Category

Recall that a category is cartesian closed if it has a terminal object, products for all pairs of objects, and exponentials.

2.1.1 Terminal Object

An object, $\mathtt{1}$, is terminal in a category, $\mathbb{C}$ if for all objects $X \in \mathtt{obj}\ \mathbb{C}$, there exists exactly one morphism $\langle\rangle_X : X \to \mathtt{1}$.

2.1.2 Products

There is a product for a pair of objects $X, Y \in \mathtt{obj}\ \mathbb{C}$ if there exists an object and morphisms in C: $X \xleftarrow{\pi_1} (X \times Y) \xrightarrow{\pi_2} Y$

Such that for any other object and morphisms,

$X \xleftarrow{f} Z \xrightarrow{g} Y$

There exists a unique morphism $\langle f, g\rangle : Z \to (X \times Y)$ such that the following commutes:

$$\begin{tikzcd}
 & Z \arrow[dl, "f"'] \arrow[d, "\langle f,g\rangle"] \arrow[dr, "g"] & \\
X & (X \times Y) \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & Y
\end{tikzcd}$$

2.1.3 Exponentials

A category has exponentials if for all objects A, B, it has an object B^A and a morphism $\mathtt{app} : \mathtt{Bool}^A \times A \to B$ and for each $f : (A \times B) \to C$ in $\mathbb{C}$ there exists a unique morphism $\mathtt{cur}(f) : A \to C^B$ such that the following diagram commutes.

$$\begin{tikzcd}
C^B \times B \arrow[r, "\mathtt{app}"] & C \\
A \times B \arrow[u, "\mathtt{cur}(f)\times\mathtt{Id}_B"] \arrow[ur, "f"] &
\end{tikzcd}$$

2.2 Initial Object

An initial object, I of $\mathbb{C}$ is one such that for every other object $X \in \mathtt{obj}\ \mathbb{C}$, there exists a unique morphism $\iota_X : I \to X$. It is the conceptual dual of a terminal objects.

2.3 Co-Product

A co-product is the conceptual dual of a product.

There is a co-product for a pair of objects $X, Y \in \mathtt{obj}\ \mathbb{C}$ if there exists an object and morphisms in C: $X \xrightarrow[\mathtt{inl}]{} (X+Y) \xleftarrow[inr]{} Y$

Such that for any other object and morphisms,

$X \xrightarrow[f]{} Z \xleftarrow[g]{} Y$

There exists a unique morphism $[f, g] : X + Y \to Z$ such that the following commutes:

$$\begin{tikzcd}
& Z & \\
X \arrow[ur, "f"] \arrow[r, "\mathtt{inl}"] & (X+Y) \arrow[u, "{[f,g]}"] & Y \arrow[l, "\mathtt{inr}"'] \arrow[ul, "g"']
\end{tikzcd}$$

2.4 Functors

A functor $F : \mathbb{C} \to \mathbb{D}$ is a mapping of objects:

$$A \in \mathtt{obj}\ \mathbb{C} \mapsto FA \in \mathtt{obj}\ \mathbb{D} \tag{2.1}$$

And morphisms:

$$f : \mathbb{C}(A, B) \mapsto F(f) : \mathbb{D}(FA, FB) \tag{2.2}$$

that preserves the category properties of composition and identity.

$$F(\mathtt{Id}_A) = \mathtt{Id}_{FA} \tag{2.3}$$

$$F(g \circ f) = F(g) \circ F(f) \tag{2.4}$$

2.5 Natural Transformations

A natural transformation θ between to functors $F, G : \mathbb{C} \to \mathbb{D}$ is a collection of morphisms, indexed by objects in $\mathbb{C}$ with $\theta_A : F(A) \to G(A)$ such that diagram in figure 2.5 commutes for each $f : A \to B \in \mathbb{C}$.

$$\begin{tikzcd}
F(A) \arrow[r, "\theta_A"] \arrow[d, "F(f)"] & G(A) \arrow[d, "G(f)"] \\
F(B) \arrow[r, "\theta_B"] & G(B)
\end{tikzcd}$$

Figure 2.1: Naturality of a natural transformation

$$\begin{tikzcd}
T(T(T(A))) \arrow[r, "\mu_{T(A)}"] \arrow[d, "T(\mu_A)"] & T(T(A)) \arrow[d, "\mu_A"] \\
T(T(A)) \arrow[r, "\mu_A"] & T(A)
\end{tikzcd}$$

Figure 2.2: Monad Associativity Laws

$$\begin{tikzcd}
T(A) \arrow[r, "\eta_{T(A)}"] \arrow[d, "T(\eta_A)"] \arrow[dr, equal] & T(T(A)) \arrow[d, "\mu_A"] \\
T(T(A)) \arrow[r, "\mu_A"] & T(A)
\end{tikzcd}$$

Figure 2.3: Monad Left- and Right-Unit laws

2.6 Monad

A monad is famously "a monoid on the category of endofunctors". In less opaque terms, a monad is:

- A functor from $\mathbb{C}$ onto itself. (An endofunctor) $T : \mathbb{C} \to C$
- A "unit" natural transformation $\eta_A : A \to T(A)$
- A "join" natural transformation $\mu_A : T(T(A)) \to T(A)$

Such that the diagrams in figures 2.6, 2.6 commute.

2.7 Graded Monad

A graded monad is a generalisation of a monad to be indexed by a monoidal algebra E. It is made up of:

- An endo-functor indexed by a monoid: $T : (\mathbb{E}, \cdot\, \mathtt{1}) \to [\mathbb{C}, \mathbb{C}]$
- A unit natural transformation: $\eta : \mathtt{Id} \to T_{\mathtt{1}}$
- A join natural transformation: $\mu_{\epsilon_1,\epsilon_2,} : T_{\epsilon_1} T_{\epsilon_2} \to T_{\epsilon_1 \cdot \epsilon_2}$

Such that the diagrams in figures 2.7, 1.15 commute.

$$\begin{tikzcd}
T_{\epsilon_1} T_{\epsilon_2} T_{\epsilon_3} A \arrow[r, "\mu_{\epsilon_1,\epsilon_2,T_{\epsilon_3} A}"] \arrow[d, "T_{\epsilon_1} \mu_{\epsilon_2,\epsilon_3,A}"] & T_{\epsilon_1 \cdot \epsilon_2} T_{\epsilon_3} A \arrow[d, "\mu_{\epsilon_1 \cdot \epsilon_2,\epsilon_3,A}"] \\
T_{\epsilon_1} T_{\epsilon_2 \cdot \epsilon_3} A \arrow[r, "\mu_{\epsilon_1,\epsilon_2 \cdot \epsilon_3,A}"] & T_{\epsilon_1 \cdot \epsilon_2 \cdot \epsilon_3} A
\end{tikzcd}$$

Figure 2.4: Associativity of a graded monad

$$\begin{tikzcd}
T_\epsilon A \arrow[r, "T_\epsilon \eta_A"] \arrow[d, "\eta_{T_\epsilon A}"] \arrow[dr, equal] & T_\epsilon T_{\mathtt{1}} A \arrow[d, "\mu_{\epsilon,\mathtt{1},A}"] \\
T_{\mathtt{1}} T_\epsilon A \arrow[r, "\mu_{\mathtt{1},\epsilon,A}"] & T_\epsilon A
\end{tikzcd}$$

Figure 2.5: Left- and Right- Units of a graded monad

$$\begin{tikzcd}
A \times T_\epsilon B \arrow[r, "f \times \mathtt{Id}_{T_\epsilon B}"] \arrow[d, "\mathtt{t}_{\epsilon,A,B}"] & A' \times T_\epsilon B \arrow[d, "\mathtt{t}_{\epsilon,A',B}"] \\
T_\epsilon(A \times B) \arrow[r, "T_\epsilon(f \times \mathtt{Id}_B)"] & T_\epsilon(A' \times B)
\end{tikzcd}$$

Figure 2.6: Left Naturality of Graded Tensor Strength

$$\begin{tikzcd}
A \times T_\epsilon B \arrow[r, "\mathtt{Id}_A \times T_\epsilon f"] \arrow[d, "\mathtt{t}_{\epsilon,A,B}"] & A \times T_\epsilon B' \arrow[d, "\mathtt{t}_{\epsilon,A,B'}"] \\
T_\epsilon(A \times B) \arrow[r, "T_\epsilon(\mathtt{Id}_A \times f)"] & T_\epsilon(A \times B')
\end{tikzcd}$$

Figure 2.7: Right Naturality of Graded Tensor Strength

$$\begin{tikzcd}
A \times T_\epsilon B \arrow[r, "\mathtt{t}_{\epsilon,A,B}"] \arrow[dr, "\pi_2"'] & T_\epsilon(A \times B) \arrow[d, "T_\epsilon \pi_2"] \\
& T_\epsilon B
\end{tikzcd}$$

Figure 2.8: Tensor Strength Unitor Law

$$\begin{tikzcd}
A \times T_{\epsilon_1} T_{\epsilon_2} B \arrow[r, "\mathtt{t}_{\epsilon_1,A,T_{\epsilon_2}B}"] \arrow[dr, "\mathtt{Id}_A \times \mu_{\epsilon_1,\epsilon_2,B}"'] & T_{\epsilon_1}(A \times T_{\epsilon_2} B) \arrow[r, "T_{\epsilon_1}\mathtt{t}_{\epsilon_2,A,B}"] & T_{\epsilon_1} T_{\epsilon_2}(A \times B) \arrow[d, "\mu_{\epsilon_1,\epsilon_2,A \times B}"] \\
& A \times T_{\epsilon_1 \cdot \epsilon_2} B \arrow[r, "\mathtt{t}_{\epsilon_1 \cdot \epsilon_2,A,B}"] & T_{\epsilon_1 \cdot \epsilon_2}(A \times B)
\end{tikzcd}$$

Figure 2.9: How the tensor strength natural transformation commutes with the join natural transformation

$$\begin{tikzcd}
A \times B \arrow[r, "\mathtt{Id}_A \times \eta_B"] \arrow[dr, "\eta_{A \times B}"'] & A \times T_{\mathbf{1}} B \arrow[d, "\mathtt{t}_{\mathbf{1},A,B}"] \\
& T_{\mathbf{1}}(A \times B)
\end{tikzcd}$$

Figure 2.10: How the tensor strength natural transformation commutes with the unit natural transformation

2.8 Tensor Strength

Tensorial strength over a graded monad gives us the tools necessary to manipulate monadic operations in an intuitive way. A monad with tensor strength is referred to as "strong". Tensorial strength consists of a natural transformation:

$$\mathtt{t}_{\epsilon,A,B} : A \times T_\epsilon B \to T_\epsilon(A \times B) \tag{2.5}$$

Which has well defined interactions with the graded monad morphisms and the product-reordering natural transformation $\alpha_{A,B,C} = \langle \pi_1 \circ \pi_1, \langle \pi_2 \circ \pi_1, \pi_2 \rangle \rangle : ((A \times B) \times C) \to (A \times (B \times C))$, as seen in figures 2.8, 2.8, 2.8, 2.8, 2.8, 2.8.

2.9 Adjunction

An important concept in category theory is that of an Adjunction.

$$\begin{tikzcd}
(A \times B) \times T_\epsilon C \arrow[rr, "\mathtt{t}_{\epsilon,(A \times B),C}"] \arrow[d, "\alpha_{A,B,T_\epsilon C}"] & & T_\epsilon((A \times B) \times C) \arrow[d, "T_\epsilon \alpha_{A,B,C}"] \\
A \times (B \times T_\epsilon C) \arrow[r, "\mathtt{Id}_A \times \mathtt{t}_{\epsilon,B,C}"] & A \times T_\epsilon(B \times C) \arrow[r, "\mathtt{t}_{\epsilon,A,(B \times C)}"] & T_\epsilon(A \times (B \times C))
\end{tikzcd}$$

Figure 2.11: Tensorial strength commutes with the reordering natural transformation.

Given functors F, G:

$$\begin{tikzcd} C \arrow[r, bend right, "F"'] & D \arrow[l, bend right, "G"'] \end{tikzcd}$$

And natural transformations:

- Unit: $\eta_A : A \to G(FA)$ in $\mathbb{C}$
- Co-unit $\epsilon_B : F(GB) \to B$ in $\mathbb{D}$

Such that

$$\epsilon_{FA} \circ F(\eta_A) = \mathtt{Id}_{FA} \tag{2.6}$$
$$G(\epsilon_B) \circ \eta_{FB} = \mathtt{Id}_{GB} \tag{2.7}$$

We can then use ϵ and η to form a natural isomorphism between morphisms in the two categories.

$$\overline{(-)} : \quad \mathbb{C}(FA, B) \leftrightarrow \mathbb{D}(A, GB) \quad : \widehat{(-)} \tag{2.8}$$
$$f \mapsto G(f) \circ \eta_A \tag{2.9}$$
$$\epsilon \circ F(g) \leftarrow\!\shortmid g \tag{2.10}$$
$$\tag{2.11}$$

2.10 Strict Indexed Category

The final piece of category theory required to understand this dissertation is the concept of a strict indexed Category. A strict indexed category is a functor from a base category into a target category of categories, such as the category of cartesian closed categories. Objects, A, in the base category are mapped to categories $\mathbb{C}(A)$, known as fibres in the target category. Morphisms between objects in the base category, $f : B \to A$, are mapped to functors, $f^* : \mathbb{C}(A) \to \mathbb{C}(B)$, between categories in the target category. Due to the composition laws for functors, $\theta^* \circ \phi^* = (\phi \circ \theta)^*$ and $\mathtt{Id}_A * (B) = B \in \mathtt{obj}\ \mathbb{C}(A)$. For example, we may use the the case of cartesian closed categories indexed by a pre-order:

$$I : \mathbb{P} \to CCCat \quad \text{The indexing functor} \tag{2.12}$$
$$A \in \mathtt{obj}\ \mathbb{P} \mapsto \mathbb{C} \in CCCat \quad \text{Objects are mapped to categories} \tag{2.13}$$
$$A \leq B \mapsto (A \leq B)^* : \mathbb{C} \to \mathbb{D} \quad \text{Morphisms are mapped to functors preserving CCC properties.} \tag{2.14}$$

Chapter 3

The Polymorphic-Effect-Calculus

In this chapter I'll be introducing the monadic, effect-ful language used in the rest of the dissertation, known from now on as the Effect Calculus (EC). Then I shall introduce polymorphic terms to the EC which yield the Polymorphic Effect Calculus (PEC).

3.1 Effect Calculus

The basic effect calculus is an extension of the simply typed lambda calculus to include constants, if-statements, effects, and sub-typing.

It has terms of the following form:

$$v ::= \mathtt{C}^A \mid x \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{()} \mid \lambda x : A.v \mid v_1\ v_2 \mid \mathtt{return} v \mid \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \mid \mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 \tag{3.1}$$

Where $\mathtt{C}^A$ is one of collection of ground constants, and A ranges over the types:

$$A, B, C ::= \gamma \mid A \to B \mid \mathtt{M}_\epsilon A \tag{3.2}$$

Where γ is from a collection of ground types, including $\mathtt{Unit}, \mathtt{Bool}$, and ϵ ranges over pre-ordered monoid of effects: $(E, \cdot, \leq, \mathtt{1})$

TODO: This is very vague so far. I don't want to get too bogged down in the semantics of the Effect Calculus though A full derivation and proof of soundness of the semantics of the Effect Calculus can be found online **TODO: Link** as it is too long to include here and many of the concepts will be repeated in the rest of this dissertation anyway. The categorical semantics of the Effect Calculus require a CCC with a strong graded monad, sub-typing morphisms, sub-effecting natural transformations, and a co-product on the terminal object. These features allow us to prove the soundness of the effect-calculus semantics.

3.2 Polymorphic Effect Calculus

Next, we shall consider the Effect Calculus extended with terms to allow System-F style polymorphism over effects.

$$v ::= .. \mid \Lambda\alpha.v \mid v\ \epsilon \tag{3.3}$$

$$A, B, C ::= ... \mid \forall\alpha.A \tag{3.4}$$

Where effects ϵ now range over the effect pre-ordered monoid augmented with effect variables from an environment $\Phi = \diamond, \alpha, \beta ...$, written as $(E_\Phi, \cdot_\Phi, \leq_\Phi, \mathtt{1})$.

3.3 Type System

3.3.1 Environments

As mentioned before, effects can now include effect variables. These are managed in the type system using a well-formed effect-variable-environment Φ, which is a snoc-list.

$$\Phi ::= \diamond \mid \Phi, \alpha \tag{3.5}$$

3.3.2 Effects

The ground effects form the same monotonous, pre-ordered monoid $(E, \cdot, \mathtt{1}, \leq)$ over ground elements e. For each effect environment Φ, we define a new, symbolic pre-ordered monoid:

$$(E_\Phi, \cdot_\Phi, \mathtt{1}, \leq_\Phi) \tag{3.6}$$

Where E_Φ is the closure of $E \cup \{\alpha \mid \alpha \in \Phi\}$ under $\cdot_\Phi$, which is defined as:

$$() \frac{\epsilon_3 = \epsilon_1 \cdot \epsilon_2}{\epsilon_3 = \epsilon_1 \cdot_\Phi \epsilon_2} \tag{3.7}$$

For variable-free terms and is defined symbolically for variable-containing terms. Further more, we also define the sub-effecting relation in terms of its variables and the ground relation.

$$\epsilon_1 \leq_\Phi \epsilon_2 \Leftrightarrow \forall \sigma \downarrow .\epsilon_1 [\sigma \downarrow] \leq \epsilon_2 [\sigma \downarrow] \tag{3.8}$$

Where $\sigma \downarrow$ denotes any ground-effect-substitution of Φ. That is any substitution of all effect-variables in Φ to ground effects. Where it is obvious from the context, I shall use $\leq$ instead of $\leq_\Phi$.

3.3.3 Types

As stated, types are now generated by the following grammar.

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

3.3.4 Type Environments

As is often the case in similar type systems, a type environment is a snoc-list of term-variable, type pairs, $G ::= \diamond \mid \Gamma, x : A$.

Domain Function on Type Environments

$$\mathtt{dom}(\diamond) = \emptyset \qquad \mathtt{dom}(\Gamma, x : A) = \mathtt{dom}(\Gamma) \cup \{x\}$$

3.3.5 Well-Formed-Ness Predicates

To formalise properties of the type system, it will be useful to have a collection of predicates ensuring that structures in the language are well-behaved with respect to their use of effect variables.

Informally, $\alpha \in \Phi$ if α appears in the list represented by Φ.

The $\mathtt{Ok}$ predicate on effect environments asserts that the effect environment does not contain any duplicate effect-variables.

$$(\text{Atom})\frac{}{\diamond\mathtt{Ok}} \quad (\text{A})\frac{\Phi\mathtt{Ok} \quad \alpha \notin \Phi}{\Phi, \alpha\mathtt{Ok}}$$

Using this, we can define the well-formed-ness relation on effects, $\Phi \vdash \epsilon$. In short, this relation ensures that effects do not reference variables that are not in the effect environment.

$$(\text{Ground})\frac{\Phi\mathtt{Ok}}{\Phi \vdash e} \quad (\text{Var})\frac{\Phi, \alpha\mathtt{Ok}}{\Phi, \alpha \vdash \alpha} \quad (\text{Weaken})\frac{\Phi \vdash \alpha}{\Phi, \beta \vdash \alpha}(\text{if } \alpha \neq \beta) \quad (\text{Monoid Op})\frac{\Phi \vdash \epsilon_1 \quad \Phi \vdash \epsilon_2}{\Phi \vdash \epsilon_1 \cdot \epsilon_2}$$

The well-formed-ness of effects can be used to a similar well-typed-relation on types, $\Phi \vdash A$, which asserts that all effects in the type are well-formed.

$$(\text{Ground})\frac{}{\Phi \vdash \gamma} \quad (\text{Lambda})\frac{\Phi \vdash A \quad \Phi \vdash B}{\Phi \vdash A \to B} \quad (\text{Computation})\frac{\Phi \vdash A \quad \Phi \vdash \epsilon}{\Phi \vdash \mathtt{M}_\epsilon A} \quad (\text{For-All})\frac{\Phi, \alpha \vdash A}{\Phi \vdash \forall \alpha.A}$$

Finally, we can derive the a well-formed-ness of type-environments, $\Phi \vdash \Gamma\mathtt{Ok}$, which ensures that all types in the environment are well formed.

$$(\text{Nil})\frac{}{\Phi \vdash \diamond\mathtt{Ok}} \quad (\text{Var})\frac{\Phi \vdash \Gamma\mathtt{Ok} \quad x \notin \mathtt{dom}(\Gamma) \quad \Phi \vdash A}{\Phi \vdash \Gamma, x : A\mathtt{Ok}}$$

3.3.6 Sub-typing

There exists a sub-typing pre-order relation $\leq:_\gamma$ over ground types. That is:

$$(\text{Reflexive})\frac{}{A \leq:_\gamma A} \quad (\text{Transitive})\frac{A \leq:_\gamma B \quad B \leq:_\gamma C}{A \leq:_\gamma C}$$

We extend this relation with the function, effect, and effect-lambda sub-typing rules to yield the full sub-typing relation under an effect environment, Φ, $\leq:_\Phi$

$$(\text{ground})\frac{A \leq:_\gamma B}{A \leq:_\Phi B} \quad (\text{Fn})\frac{A \leq:_\Phi A' \quad B' \leq:_\Phi B}{A' \to B' \leq:_\Phi A \to B} \quad (\text{All})\frac{A \leq:_\Phi A'}{\forall \alpha.A \leq:_\Phi \forall a.A'} \quad (\text{Effect})\frac{A \leq:_\Phi B \quad \epsilon_1 \leq_\Phi \epsilon_2}{\mathtt{M}_{\epsilon_1} A \leq:_\Phi \mathtt{M}_{\epsilon_2} B}$$

3.3.7 Type Rules

We define a fairly standard set of type rules on the language.

$$\text{(Const)}\frac{\Phi \vdash \Gamma\mathtt{Ok} \quad \Phi \vdash A}{\Phi \mid \Gamma \vdash \mathtt{C}^{A} \colon A} \qquad \text{(Unit)}\frac{\Phi \vdash \Gamma\mathtt{Ok}}{\Phi \mid \Gamma \vdash \mathtt{()} \colon \mathtt{Unit}} \qquad \text{(True)}\frac{\Phi \vdash \Gamma\mathtt{Ok}}{\Phi \mid \Gamma \vdash \mathtt{true} \colon \mathtt{Bool}} \qquad \text{(False)}\frac{\Phi \vdash \Gamma\mathtt{Ok}}{\Phi \mid \Gamma \vdash \mathtt{false} \colon \mathtt{Bool}}$$

$$\text{(Var)}\frac{\Phi \vdash \Gamma, x : A\mathtt{Ok}}{\Phi \mid \Gamma, x : A \vdash x \colon A} \qquad \text{(Weaken)}\frac{\Phi \mid \Gamma \vdash x \colon A \quad \Phi \vdash B}{\Phi \mid \Gamma, y : B \vdash x \colon A}(\text{if } x \neq y) \qquad \text{(Fn)}\frac{\Phi \mid \Gamma, x : A \vdash v \colon \beta}{\Phi \mid \Gamma \vdash \lambda x : A.v \colon A \to B}$$

$$\text{(Sub)}\frac{\Phi \mid \Gamma \vdash v \colon A \quad A \leq \colon_{\Phi} B}{\Phi \mid \Gamma \vdash v \colon B} \qquad \text{(Effect-Abs)}\frac{\Phi, \alpha \mid \Gamma \vdash v \colon A}{\Phi \mid \Gamma \vdash \Lambda\alpha.v \colon \forall\alpha.A} \qquad \text{(Effect-apply)}\frac{\Phi \mid \Gamma \vdash v \colon \forall\alpha.A \quad \Phi \vdash \epsilon}{\Phi \mid \Gamma \vdash v\ \epsilon \colon A\,[\epsilon/\alpha]}$$

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

$$\text{(If)}\frac{\Phi \mid \Gamma \vdash v \colon \mathtt{Bool} \quad \Phi \mid \Gamma \vdash v_1 \colon A \quad \Phi \mid \Gamma \vdash v_2 \colon A}{\Phi \mid \Gamma \vdash \mathtt{if}_A\ V\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 \colon A} \qquad \text{(Do)}\frac{\Phi \mid \Gamma \vdash v_1 \colon \mathtt{M}_{\epsilon_1} A \quad \Phi \mid \Gamma, x : A \vdash v_2 \colon \mathtt{M}_{\epsilon_2} B}{\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \colon \mathtt{M}_{\epsilon_1 \cdot \epsilon_2} B}$$

3.3.8 Ok Lemma

The first lemma used in this dissertation is that: If $\Phi \mid \Gamma \vdash v \colon A$ then $\Phi \vdash \Gamma\mathtt{Ok}$.

Proof If $\Gamma, x : A\mathtt{Ok}$ then by inversion $\Gamma\mathtt{Ok}$ Only the type rule `Weaken` adds terms to the environment from its preconditions to its post-condition and it does so in an `Ok` preserving way. Any type derivation tree has at least one leaf. All leaves are axioms which require $\Phi \vdash \Gamma\mathtt{Ok}$. And all non-axiom derivations preserve the `Ok` property.

Chapter 4

Semantics for EC in an S-Category

As hinted at previously, we can interpret the Effect Calculus in a CCC with a strong graded monad and co-products. A further requirement is the appropriate sub-effecting natural transformations. For each instance of $\epsilon_1 \leq \epsilon_2$, there exists a natural transformation $[\![\epsilon_1 \leq \epsilon 2]\!] : T_{\epsilon_1} \to T_{\epsilon_2}$ such that it has interactions with the graded monad as specified in figures 4, 4. We shall call a category fulfilling these properties an S-Category (Semantic Category).

TODO: This has been very vague as I want to save words on the non-polymorphic stuff.

$$\begin{tikzcd}
A \times T_{\epsilon_1} B \arrow[r, "\mathtt{Id}_A \times [\![\epsilon_1 \leq \epsilon_2]\!]_B"] \arrow[d, "\mathtt{t}_{\epsilon_1, A, B}"] & A \times T_{\epsilon_2} B \arrow[d, "\mathtt{t}_{\epsilon_2, A, B}"] \\
T_{\epsilon_1}(A \times B) \arrow[r, "[\![\epsilon_1 \leq \epsilon_2]\!]_{A \times B}"] & T_{\epsilon_2}(A \times B)
\end{tikzcd}$$

Figure 4.1: The interaction of the sub-effect natural transformation with the tensor strength natural transformation.

$$\begin{tikzcd}
T_{\epsilon_1} T_{\epsilon_2} \arrow[r, "T_{\epsilon_1} [\![\epsilon_2 \leq \epsilon_2']\!]_M"] \arrow[d, "\mu_{\epsilon_1, \epsilon_2},"] & T_{\epsilon_1} T_{\epsilon_2'} \arrow[r, "[\![\epsilon_1 \leq \epsilon_1']\!]_{M, T_{\epsilon_2'}}"] & T_{\epsilon_1'} T_{\epsilon_2'} \arrow[d, "\mu_{\epsilon_1', \epsilon_2'},"] \\
T_{\epsilon_1 \cdot \epsilon_2} \arrow[rr, "[\![\epsilon_1 \cdot \epsilon_2 \leq \epsilon_1' \cdot \epsilon_2']\!]_M"] & & T_{\epsilon_1' \cdot \epsilon_2'}
\end{tikzcd}$$

Figure 4.2: The interaction of the sub-effect natural transformation with the graded monad bind natural transformation.

Chapter 5

Semantics For PEC in an Indexed Category

In this chapter, I shall describe the category structure required to interpret an instance of the PEC. I shall then present denotations of each type of structure in the language, such as types, effects, terms, substitutions, and environment weakenings. Finally, I shall provide outlines and interesting cases of the proofs of the lemmas leading up to and including soundness of $\beta\eta$-conversion.

5.1 Required Category Structure

In order to model the polymorphism of PEC, we need to now look at an indexed category. This consists of a base category, $\mathbb{C}$ in which we can interpret the possible effect-environments in, and a mapping from objects in the base category to S-Categories in the category of S-closed categories. This functor shall be denoted as $\mathbb{C}(-)$ and the induced categories $\mathbb{C}(I)$ shall be called "fibres". The term "S-closed" indicates that all functors within this category preserve the properties of S-categories. **TODO: Are these ever fully explained?**. Thus, each morphism $\theta : [\![\Phi']\!]_M \to [\![\Phi]\!]_M$ in $\mathbb{C}$ should induce as S-closed, re-indexing functor $\theta^* : \mathbb{C}(I) \to (I')$ between the fibres.

The essential idea from this point on is that for each relation $\texttt{Env} \vdash \texttt{Conclusion}$ should have a denotation that is an object or morphism in a category. For example, $[\![\Phi \vdash \epsilon\colon \texttt{Effect}]\!]_M$ is a morphism in the base category, $[\![\Phi \vdash A\colon \texttt{Type}]\!]_M$ is an object in the fibre (S-category) induced by Φ, and $[\![\Phi \mid \Gamma \vdash v\colon A]\!]_M$ is morphism between the objects which denote Γ and A in the fibre induced by Φ.

In order to form denotations of well formed effects, ϵ, we need specific objects to exist in the base category $\mathbb{C}$. Firstly, there should exist an object, U, indicating the kind of effects. To denote effect-variable-environments, essentially a list of variables, we need finite products on U, that is $\mathbb{C}$ should have a terminal object, $\mathtt{1}$ and binary products. We can form finite products as so: $U^0 = \mathtt{1}$ and $U^{n+1} = U^n \times U$. From now on, I shall use I to mean U^n for some n.

There is also a requirement that the indexed category can model ground effects, types, and terms. In order to do this, it should have a base-category morphism $[\![e]\!]_M : \mathbb{C}(\mathtt{1}, U)$ for each ground effect e. Furthermore, each fibre should contain and object $[\![\gamma]\!]_M$ for each ground-type γ. Finally, for each constant, $\mathtt{C}^A$, there should exist a morphism in each fibre: $[\![\mathtt{C}^A]\!]_M : \mathtt{1} \to A$. These last two requirements are satisfied by the fibres all being S-categories.

Next up, there needs to be a monoidal operator $\texttt{Mul} : \mathbb{C}(I, U) \times \mathbb{C}(I, U) \to \mathbb{C}(I, U)$. $\texttt{Mul}$ should be natural, which means: $\texttt{Mul}(f, g) \circ \theta = \texttt{Mul}(f \circ \theta, g \circ \theta)$. Secondly, $\texttt{Mul}$ should preserve the operation of the multiplication of ground effects. That is, $\texttt{Mul}([\![e_1]\!]_M, [\![e_2]\!]_M) = [\![e_1 \cdot e_2]\!]_M$ where e_1, e_2 are ground effects.

Our penultimate requirement is that the re-indexing functor induced by $\pi_1 : I \times U \to U$ (that is $\pi_1^* : \mathbb{C}(I) \to \mathbb{C}(I \times U)$) has a right adjoint, $\forall_I : \mathbb{C}(I \times U) \to \mathbb{C}(I)$. As the reader might be able to guess, this functor allows us to interpret quantification over effects.

Finally, $\forall_I$ should satisfy the Beck-Chevalley condition **TODO: Reference**. That is $\theta^* \circ \forall_I = \forall_{I'} \circ (\theta \times \mathtt{Id}_U)^*$, and the natural transformation $\overline{(\theta \times \mathtt{Id}_U)^*(\boldsymbol{\epsilon})}$ between these functors is equal to the identity natural transformation. This allows us to commute the re-indexing functors with the quantification functor.

$$\overline{(\theta \times \mathtt{Id}_U)^*(\boldsymbol{\epsilon})} = \mathtt{Id} : \theta^* \circ \forall_I \to \forall_{I'} \circ (\theta \times \mathtt{Id}_U)^* \in \mathbb{C}(I') \tag{5.1}$$

5.2 Road Map

In figure 5.2, one can see a diagram of the collection of theorems that need to be proved to establish the $\beta\eta$-equivalence soundness of a semantics for PEC.

The first pair of theorems is effect-substitution theorem on effects. These theorems show that substitutions of effects have well-behaved and easily defined action upon the denotations of effects. Using these theorems, we can then move on to characterize the action of effect-substitutions and effect-environment-weakening on the denotations of types and type-environments. From this, we can also look at the action of weakening and substituting effect environments on the sub-typing between types.

The next step is to use these substitution theorems to formalise the action of substitution and weakening of the effect environments on terms. This then allows us to find denotations for the weakening of term-substitutions and type environment weakening, which set us up to prove the typical weakening and substitution theorems upon term-variables and type environments.

Separately, we prove that all derivable denotations for a typing relation instance, $\Phi \mid \Gamma \vdash v\colon A$ have the same denotation. This is important, since sub-typing allows us to find multiple distinct typing derivations for terms, which initially look like they may have distinct denotations. Using a reduction function to transform typing derivations into a unique form, I shall prove that all typing derivations yield equal denotations.

This collection of theorems finally allows us to complete all cases of the $\beta\eta$-equivalence soundness theorem.

5.3 Denotations

We are now equipped to define the denotations of structures in the language. Firstly, we shall define the denotation of the well-formed-ness relation on effects. As stated earlier, the denotation of an effect is a morphism $[\![\Phi \vdash \epsilon\colon \mathtt{Effect}]\!]_M$ in $\mathbb{C}$.

$$[\![\Phi \vdash e\colon \mathtt{Effect}]\!]_M = [\![\epsilon]\!]_M \circ \langle\rangle_I :\to U \qquad [\![\Phi, \alpha \vdash \alpha\colon \mathtt{Effect}]\!]_M = \pi_2 : I \times U \to U$$
$$[\![\Phi, \beta \vdash \alpha\colon \mathtt{Effect}]\!]_M = [\![\Phi \vdash \alpha\colon \mathtt{Effect}]\!]_M \circ \pi_1 : I \times U \to U$$
$$[\![\Phi \vdash \epsilon_1 \cdot \epsilon_2\colon \mathtt{Effect}]\!]_M = \mathtt{Mul}([\![\Phi \vdash \epsilon_2\colon \mathtt{Effect}]\!]_M, [\![\Phi \vdash \epsilon_1\colon \mathtt{Effect}]\!]_M) : I \to U$$

Using these denotations, we are now equipped to define the denotations of types. As stated above, types that are well formed in Φ are denoted by objects in the fibre category $\mathbb{C}(I)$ given by the denotation of Φ. (**TODO: Have I explained that $[\![\Phi]\!]_M = I$?**)

Since the fibre category $\mathbb{C}(I)$ is S-Closed, it has objects for all ground types, a terminal object, graded monad T, exponentials, products, and co-product over $\mathtt{1} + \mathtt{1}$.

$$[\![\Phi \vdash \texttt{Unit}\colon \texttt{Type}]\!]_M = \texttt{1} \qquad [\![\Phi \vdash \texttt{Bool}\colon \texttt{Type}]\!]_M = \texttt{1}+\texttt{1} \qquad [\![\Phi \vdash \gamma\colon \texttt{Type}]\!]_M = [\![\gamma]\!]_M$$

$$[\![\Phi \vdash A \to B\colon \texttt{Type}]\!]_M = ([\![\Phi \vdash B\colon \texttt{Type}]\!]_M)^{([\![\Phi \vdash A:\texttt{Type}]\!]_M)}$$

$$[\![\Phi \vdash \texttt{M}_\epsilon A\colon \texttt{Type}]\!]_M = T_{[\![\Phi\vdash\epsilon:\texttt{Effect}]\!]_M}[\![\Phi \vdash A\colon \texttt{Type}]\!]_M \qquad [\![\Phi \vdash \forall\alpha.A\colon \texttt{Type}]\!]_M = \forall_I([\![\Phi,\alpha \vdash A\colon \texttt{Type}]\!]_M)$$

By using the terminal objects and products present in each fibre, we can now derive denotations of type-environments. $[\![\Phi \vdash \Gamma\texttt{Ok}]\!]_M$ should be an object in the fibre induced by Φ, $\mathbb{C}(I)$.

$$[\![\Phi \vdash \diamond\texttt{Ok}]\!]_M = \texttt{1} \qquad [\![\Phi \vdash \Gamma, x : A\texttt{Ok}]\!]_M = ([\![\Phi \vdash \Gamma\texttt{Ok}]\!]_M \times [\![\Phi \vdash A\colon \texttt{Type}]\!]_M)$$

TODO: Introduce the subtyping relation up above. Another construction that is important is the denotation of sub-typing. For each instance of the sub-typing relation in Φ, $A \leq:_\Phi B$, there exists a denotation in the fibre induced by Φ. $[\![A \leq:_\Phi B]\!]_M \in \mathbb{C}(I)(A,B)$. Since the fibres are S-closed, the ground-instances of the sub-typing relation exist in each fibre anyway.

$$[\![\gamma_1 \leq:_\Phi \gamma_2]\!]_M = [\![\gamma_1 \leq:_\gamma \gamma_2]\!]_M \quad [\![A \to B \leq:_\Phi A' \to B']\!]_M = [\![B \leq:_\Phi B']\!]_M^{A'} \circ B^{[\![A' \leq:_\Phi A]\!]_M}$$

$$[\![\texttt{M}_{\epsilon_1} A \leq:_\Phi \texttt{M}_{\epsilon_2} B]\!]_M = [\![\epsilon_1 \leq_\Phi \epsilon_2]\!]_M \circ T_{\epsilon_1}[\![A \leq:_\Phi B]\!]_M \quad [\![\forall\alpha.A \leq:_\Phi \forall\alpha.B]\!]_M = \forall_I[\![A \leq:_{\Phi,\alpha} B]\!]_M$$

This finally gives us the ability to express the denotations of well-typed terms in an effect environment, Φ as morphisms in the fibre induced by Φ, $\mathbb{C}(I)$. Writing Γ_I and A_I for $[\![\Phi \vdash \Gamma\texttt{Ok}]\!]_M$ and $[\![\Phi \vdash A\colon \texttt{Type}]\!]_M$, we can derive $[\![\Phi \mid \Gamma \vdash v\colon A]\!]_M$ as a morphism in $\mathbb{C}(I)(\Gamma_I, A_I)$.

Since each fibre is an S-category, for each ground constant, $\texttt{C}^A$, there exists $c : \texttt{1} \to A_I$ in $\mathbb{C}(I)$.

TODO: Make these more readable/fix spacing

$$\text{(Unit)}\frac{\Phi \vdash \Gamma\texttt{Ok}}{[\![\Phi \mid \Gamma \vdash \texttt{()}\colon \texttt{Unit}]\!]_M = \langle\rangle_\Gamma : \Gamma_I \to \texttt{1}} \qquad \text{(Const)}\frac{\Phi \vdash \Gamma\texttt{Ok}}{[\![\Phi \mid \Gamma \vdash \texttt{C}^A\colon A]\!]_M = [\![\texttt{C}^A]\!]_M \circ \langle\rangle_\Gamma : \Gamma \to [\![A]\!]_M}$$

$$\text{(True)}\frac{\Phi \vdash \Gamma\texttt{Ok}}{[\![\Phi \mid \Gamma \vdash \texttt{true}\colon \texttt{Bool}]\!]_M = \texttt{inl} \circ \langle\rangle_\Gamma : \Gamma \to [\![\texttt{Bool}]\!]_M = \texttt{1}+\texttt{1}} \qquad \text{(False)}\frac{\Phi \vdash \Gamma\texttt{Ok}}{[\![\Phi \mid \Gamma \vdash \texttt{false}\colon \texttt{Bool}]\!]_M = \texttt{inr} \circ \langle\rangle_\Gamma : \Gamma \to [\![\texttt{Boo}}$$

$$\text{(Var)}\frac{\Phi \vdash \Gamma\texttt{Ok}}{[\![\Phi \mid \Gamma, x : A \vdash x\colon A]\!]_M = \pi_2 : \Gamma \times A \to A} \qquad \text{(Weaken)}\frac{f = [\![\Phi \mid \Gamma \vdash x\colon A]\!]_M : \Gamma \to A}{[\![\Phi \mid \Gamma, y : B \vdash x\colon A]\!]_M = f \circ \pi_1 : \Gamma \times B \to A}$$

$$\text{(Lambda)}\frac{f = [\![\Phi \mid \Gamma, x : A \vdash v\colon B]\!]_M : \Gamma \times A \to B}{[\![\Phi \mid \Gamma \vdash \lambda x : A.v\colon A \to B]\!]_M = \texttt{cur}(f) : \Gamma \to (B)^A}$$

$$\text{(Subtype)}\frac{f = [\![\Phi \mid \Gamma \vdash v\colon A]\!]_M : \Gamma \to A \quad g = [\![A \leq:_\Phi B]\!]_M}{[\![\Phi \mid \Gamma \vdash v\colon B]\!]_M = g \circ f : \Gamma \to B} \qquad \text{(Return)}\frac{f = [\![\Phi \mid \Gamma \vdash v\colon A]\!]_M}{[\![\Phi \mid \Gamma \vdash \texttt{return}v\colon \texttt{M}_\texttt{1} A]\!]_M = \eta_A \circ f}$$

$$\text{(If)}\frac{f = [\![\Phi \mid \Gamma \vdash v\colon \texttt{Bool}]\!]_M : \Gamma \to \texttt{1}+\texttt{1} \quad g = [\![\Phi \mid \Gamma \vdash v_1\colon \texttt{M}_\epsilon A]\!]_M \quad h = [\![\Phi \mid \Gamma \vdash v_2\colon \texttt{M}_\epsilon A]\!]_M}{[\![\Phi \mid \Gamma \vdash \texttt{if}_{\epsilon,A}\ v\ \texttt{then}\ v_1\ \texttt{else}\ v_2\colon \texttt{M}_\epsilon A]\!]_M = \texttt{app} \circ (([\texttt{cur}(g \circ \pi_2), \texttt{cur}(h \circ \pi_2)] \circ f) \times \texttt{Id}_\Gamma) \circ \delta_\Gamma : \Gamma \to T_\epsilon A}$$

$$(\text{Bind})\frac{f = [\![\Phi \mid \Gamma \vdash v_1{:}\,\mathtt{M}_{\epsilon_1} A : \Gamma \to T_{\epsilon_1} A \quad g = [\![\Phi \mid \Gamma, x : A \vdash v_2{:}\,\mathtt{M}_{\epsilon_2} B]\!]_M]\!]_M : \Gamma \times A \to T_{\epsilon_2} B}{[\![\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2{:}\,\mathtt{M}_{\epsilon_1 \cdot \epsilon_2} B]\!]_M = \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} g \circ \mathtt{t}_{\Gamma,A,\epsilon_1} \circ \langle \mathtt{Id}_\Gamma, f\rangle : \Gamma \to T_{\epsilon_1 \cdot \epsilon_2} B}$$

$$(\text{Apply})\frac{f = [\![\Phi \mid \Gamma \vdash v_1{:}\, A \to B]\!]_M : \Gamma \to (B)^A \quad g = [\![\Phi \mid \Gamma \vdash v_2{:}\, A]\!]_M : \Gamma \to A}{[\![\Phi \mid \Gamma \vdash v_1\ v_2{:}\, \beta]\!]_M = \mathtt{app} \circ \langle f, g\rangle : \Gamma \to B}$$

$$(\text{Effect-Lambda})\frac{f = [\![\Phi, \alpha \mid \Gamma \vdash v{:}\, A]\!]_M : \mathbb{C}(I \times U, W)(\Gamma, A)}{[\![\Phi \mid \Gamma \vdash \Lambda\alpha.A{:}\,\forall\epsilon.A]\!]_M = \overline{f} : \mathbb{C}(I)(\Gamma, \forall_I(A))}$$

$$(\text{Effect-App})\frac{g = [\![\Phi \mid \Gamma \vdash v{:}\,\forall\alpha.A]\!]_M : \mathbb{C}(I)(\Gamma, \forall_I(A)) \quad h = [\![\Phi \vdash \epsilon{:}\,\mathtt{Effect}]\!]_M : \mathbb{C}(I, U)}{[\![\Phi \mid \Gamma \vdash v\ \epsilon{:}\, A\,[\epsilon/\alpha]\,]\!]_M = \langle \mathtt{Id}_I, h\rangle^* \,(\epsilon_{[\![\Phi,\beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M}) \circ g : \mathbb{C}(I)(\Gamma, A\,[\epsilon/\alpha])}$$

5.4 Substitution and Weakening Theorems

In this section, I shall introduce and prove a series of utility theorems, which will help us prove cases in future theorems. These weakening and substitution theorems are concerned with a change-in-environment of typing derivations and their associated denotations. If $\Phi \mid \Gamma \vdash v{:}\, A$, then it should be the case that $\Phi \mid \Gamma, x : A \vdash v{:}\, A$. We also want to know what happens to the denotation when we change the type environment in this way. In this section, I shall introduce the tools for manipulating the type and effect environments in this way.

Substitutions and weakenings are two distinct ways of manipulating an effect or typing environment. Weakening acts as a kind of sub-typing of the environment. If we insert fresh variables into an environment, then any expression that was typeable under the previous environment should also be typeable under the the new environment. This change of environment should also have a predictable effect on the denotations of any expressions to which it is applied.

Substitution considers what happens when we simultaneously replace all variables in one expression, that is typeable under an environment, with expressions that are well formed under another environment. The resulting substituted expression should be typeable under the new environment, and the denotation of the new expression should be composed from the denotation of the old relation and the denotations of the expressions that replace the variables.

As we go on, I shall define and state the denotations of specific substitutions and weakenings, upon both the effect environment and the typing environment.

TODO: Needs an introduction to what substitutions and weakenings are TODO: This section is very terse so far. needs a lot of filling out still

- Substitution - Simultaneous substitution of variables to expressions that are well formed in another environment
- Weakening is a kind of sub-typing on environments.

In this dissertation, substitutions and weakenings come in two flavours: weakening and substitution of the effect-variable environment and weakening and substitution of the typing environment. For each of these there is a family of theorems defining the effects of the applying a substitution and weakening to the various language structures and their denotations, such as well-formed-ness and typing relations.

The first family of theorems is that of weakening and substitution of the effect environment. Weakenings are a relation between effect-environments $\omega : \Phi' \triangleright \Phi$, that are defined as so:

$$(\mathrm{Id})\frac{\Phi\texttt{Ok}}{\iota : \Phi \triangleright \Phi} \quad (\mathrm{Project})\frac{\omega : \Phi' \triangleright \Phi}{\omega\pi : (\Phi', \alpha) \triangleright \Phi} \quad (\mathrm{Extend})\frac{\omega : \Phi' \triangleright \Phi}{\omega\times : (\Phi', \alpha) \triangleright (\Phi, \alpha)}$$

With inductively defined denotations:

$$[\![\iota : \Phi \triangleright \Phi]\!]_M = \texttt{Id}_I : I \to I \quad [\![w\pi : \Phi', \alpha \triangleright \Phi]\!]_M = [\![\omega : \Phi' \triangleright \Phi]\!]_M \circ \pi_1 : I' \times U \to I$$

$$[\![w\times : \Phi', \alpha \triangleright \Phi, \alpha]\!]_M = ([\![\omega : \Phi' \triangleright \Phi]\!]_M \times \texttt{Id}_U) : I' \times U \to I \times U$$

Substitutions are also an inductively defined relation between effect-environments, with inductively defined denotations. Substitutions may be represented as a snoc-list of variable-effect pairs.

$$\sigma ::= \diamond \mid \sigma, \alpha := \epsilon$$

The substitution relation between effect environments is defined as so:

$$(\mathrm{Nil})\frac{\Phi'\texttt{Ok}}{\Phi' \vdash \diamond\colon \diamond} \qquad (\mathrm{Extend})\frac{\Phi' \vdash \sigma\colon \Phi \quad \Phi' \vdash \epsilon \quad \alpha \notin \Phi}{\Phi' \vdash \sigma, \alpha := \epsilon\colon (\Phi, \alpha)}$$

The denotations of substitutions, $[\![\Phi' \vdash \sigma\colon \Phi]\!]_M : \mathbb{C}(I', I)$, are defined as so:

$$[\![\Phi' \vdash \diamond\colon \diamond]\!]_M = \langle\rangle_I : \mathbb{C}(I', \mathtt{1}) \qquad [\![\Phi' \vdash (\sigma, \alpha := \epsilon)\colon \Phi, \alpha]\!]_M = \langle [\![\Phi' \vdash \sigma\colon \Phi]\!]_M, [\![\Phi \vdash \epsilon\colon \texttt{Effect}]\!]_M \rangle : \mathbb{C}(I', I \times U)$$

We can also define the action of substitutions on effects.

$$\sigma(e) = e \tag{5.2}$$
$$\sigma(\epsilon_1 \cdot \epsilon_2) = (\sigma(\epsilon_1)) \cdot (\sigma(\epsilon_2)) \tag{5.3}$$
$$\diamond(\alpha) = \alpha \tag{5.4}$$
$$(\sigma, \beta := \epsilon)(\alpha) = \sigma(\alpha) \tag{5.5}$$
$$(\sigma, \alpha := \epsilon)(\alpha) = \epsilon \tag{5.6}$$

.

Theorem 5.4.1 (Effect Substitution on Effects) *The substitution theorem on effects is the proposition that if $\Phi \vdash \epsilon$ and $\Phi' \vdash \sigma\colon \Phi$ then $\Phi' \vdash \sigma(\epsilon)$ and, writing σ for $[\![\Phi' \vdash \sigma\colon \Phi]\!]_M$, $[\![\Phi' \vdash \sigma(\epsilon)]\!]_M = [\![\Phi \vdash \sigma]\!]_M \circ \sigma$.*

Proof: The proof of this depends on the naturality of `Mul` and inversion to narrow down case splitting on the structure of the effect environments. **TODO: Pick a pair of cases**

Theorem 5.4.2 (Effect Weakening on Effects) *The weakening theorem proceeds similarly. If $\Phi \vdash \epsilon$ and $\omega : \Phi' \triangleright \Phi$ then $\Phi' \vdash \omega$ and, writing ω for $[\![\omega : \Phi' \triangleright \Phi]\!]_M$, $[\![\Phi' \vdash \epsilon]\!]_M = [\![\Phi \vdash \sigma]\!]_M \circ \omega$.*

Proof: This proof also depends on the naturality of `Mul` and case splitting on the structure of ω. **TODO: Pick a pair of cases**.

We can then move on to state and prove the weakening and substitution theorems on types, sub-typing, and type environments. The general structure of these theorems, as well as later, is that when we want to quantify the effect of a morphism $\theta : I' \to I$ between objects in the base category on structure in the fibres $\mathbb{C}(I)$, we should simply apply the associated re-indexing functor $\theta^* : \mathbb{C}(I) \to \mathbb{C}(I')$ to the structure. The proof of the soundness of this operation is driven by the S-closure of the re-indexing functor.

Specifically, effect substitutions have the following actions on types, and type-environments:

$$\begin{aligned}
\gamma\,[\sigma] &= \gamma \\
(A \to B)\,[\sigma] &= (A\,[\sigma]) \to (B\,[\sigma]) \\
(\mathtt{M}_\epsilon A)\,[\sigma] &= \mathtt{M}_{\sigma(\epsilon)}(A\,[\sigma]) \\
(\forall\alpha.A)\,[\sigma] &= \forall\alpha.(A\,[\sigma]) \quad \text{If } \alpha\#\sigma
\end{aligned}$$

$$\begin{aligned}
\diamond\,[\sigma] &= \diamond \\
(\Gamma, x : A)\,[\sigma] &= (\Gamma\,[\sigma]\,, x : (A\,[\sigma]))
\end{aligned}$$

Theorem 5.4.3 (Effect Substitution on Types) *The specific effect-substitution theorem on types is if $\Phi \vdash A$ and $\Phi' \vdash \sigma\colon\Phi$, then $\Phi' \vdash A\,[\sigma]$ and $[\![\Phi' \vdash A\,[\sigma]\,]\!]_M = \sigma^*[\![\Phi \vdash A]\!]_M$.*

Proof: By S-closure of σ^* and the Beck-Chevalley Condition. **TODO: Pick Cases (Quantification?)**

Theorem 5.4.4 (Effect Substitution on Type Environments) *Similarly, the effect-substitution theorem on type environments is that if $\Phi \vdash \Gamma\,\mathit{Ok}$, then $\Phi' \vdash \Gamma\,[\sigma]\,\mathit{Ok}$ and $[\![\Phi' \vdash \Gamma\,[\sigma]\,\mathit{Ok}]\!]_M = \sigma^*[\![\Phi \vdash \Gamma\,\mathit{Ok}]\!]_M$.*

Proof: By S-closure of σ^*. **TODO: Give the same cases?**

The effect-weakening theorem on types and type environments is also very similar.

Theorem 5.4.5 *]Effect Weakening on Types] If $\omega : \Phi' \triangleright \Phi$ then $\Phi \vdash A$ implies $\Phi' \vdash A \wedge [\![\Phi' \vdash A]\!]_M = \omega^*[\![\Phi \vdash A]\!]_M$ and $\Phi \vdash \Gamma\,\mathit{Ok}$ then $[\![\Phi' \vdash \Gamma\,\mathit{Ok}]\!]_M = \omega^*[\![\Phi \vdash \Gamma\,\mathit{Ok}]\!]_M$*

Proof: Similarly making use of S-closure and the Beck-Chevalley Condition.

TODO: Quantification case for types.

The proof for type environments follows the same steps as effect-substitution proof.

Next we shall consider the action of weakening and substitution on sub-typing relations.

Theorem 5.4.6 (Effect Substitution on Sub-typing) *If $A \leq:_\Phi B$ and $\Phi' \vdash \sigma\colon\Phi$, then $A\,[\sigma] \leq:_{\Phi'} B\,[\sigma]$ and $[\![A\,[\sigma] \leq:_{\Phi'} B\,[\sigma]\,]\!]_M = [\sigma]^*\,[\![A \leq:_\Phi B]\!]_M$.*

Proof: By rule induction over the definition of the subtype relation, making use of S-closure and the effect-substitution theorem on types.

TODO: Give case of function types.

Similarly we can form the symmetrical weakening theorem.

Theorem 5.4.7 (Effect Weakening on Subtyping) *If $A \leq:_{\Phi} B$ and $\omega : \Phi' \triangleright \Phi$, then $A \leq:_{\Phi'} B$ and $[\![A \leq:_{\Phi'} B]\!]_M = \omega^*[\![A \leq:_{\Phi} B]\!]_M$.*

Proof: The cases hold the same as in the corresponding substitution theorem.

We are now a point to define and prove the effect weakening and substitution theorems on terms. Following the intuition above that changes of index object should be modelled by applying the re-indexing functor to the morphisms denoting the terms, we can construct the theorems. Firstly, we must define the operation of effect substitutions on terms.

$$
\begin{aligned}
x\,[\sigma] &= x \\
\mathtt{C}^A\,[\sigma] &= \mathtt{C}^{(A[\sigma])} \\
(\lambda x : A.v)\,[\sigma] &= \lambda x : (A\,[\sigma]).(v\,[\sigma]) \\
(\mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2)\,[\sigma] &= \mathtt{if}_{(A[\sigma])}\ v\,[\sigma]\ \mathtt{then}\ v_1\,[\sigma]\ \mathtt{else}\ v_2\,[\sigma] \\
(v_1\ v_2)\,[\sigma] &= (v_1\,[\sigma])\ v_2\,[\sigma] \\
(\mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2)\,[\sigma] &= \mathtt{do}\ x \leftarrow (v_1\,[\sigma])\ \mathtt{in}\ (v_2\,[\sigma]) \\
(\Lambda\alpha.v)\,[\sigma] &= \Lambda\alpha.(v\,[\sigma]) \quad \text{If } \alpha\#\sigma \\
(v\ \epsilon)\,[\sigma] &= (v\,[\sigma])\ \sigma(\epsilon)
\end{aligned}
$$

The substitution theorem is defined as so:

Theorem 5.4.8 (Effect Substitution on Terms) *if $\Phi' \vdash \sigma{:}\,\Phi$ and $\Phi \mid \Gamma \vdash v{:}\,A$ then $\Phi' \mid \Gamma\,[\sigma] \vdash v\,[\sigma] : A\,[\sigma]$ and $[\![\Phi' \mid \Gamma\,[\sigma] \vdash v\,[\sigma] : A\,[\sigma]\,]\!]_M = \sigma^*[\![\Phi \mid \Gamma \vdash v{:}\,A]\!]_M$.*

Proof: This proof makes use of the previous effect-substitution theorems, and the adjunction of quantification and the re-indexing functor of projection. **TODO: Pick cases: Effect Application, Effect-Lambda, any more?**

Similarly, we can derive the weakening theorem on terms.

Theorem 5.4.9 (Effect Weakening on Terms) *If $\omega : \Phi' \triangleright \Phi$ and $\Phi \mid \Gamma \vdash v{:}\,A$ then $\Phi' \mid \Gamma \vdash v{:}\,A$ and $[\![\Phi' \mid \Gamma \vdash v{:}\,A]\!]_M = \omega^*([\![\Phi \mid \Gamma \vdash v{:}\,A]\!]_M)$.*

Proof: Follows in a similar fashion to that of the substitution theorem. **TODO: Case Subtyping**.

Now we are at a point to start considering weakenings and substitution of the typing environment. Type environment weakenings are inductively defined with respect to an effect environment.

$$
(\text{Id})\frac{\Phi \vdash \Gamma\mathtt{Ok}}{\Phi \vdash \iota : \Gamma \triangleright \Gamma} \quad (\text{Project})\frac{\Phi \vdash \omega : \Gamma' \triangleright \Gamma \quad x \notin \mathtt{dom}(\Gamma')}{\Phi \vdash \omega\pi : \Gamma, x : A \triangleright \Gamma} \quad (\text{Extend})\frac{\Phi \vdash \omega : \Gamma' \triangleright \Gamma \quad x \notin \mathtt{dom}(\Gamma') \quad A \leq: B}{\Phi \vdash w\times : \Gamma', x : A \triangleright \Gamma, x : B}
$$

With denotations defined as morphisms in a fibre: $[\![\Phi \vdash \omega : \Gamma' \triangleright \Gamma]\!]_M : \Gamma' \to \Gamma \in \mathbb{C}(I)$.

$$\llbracket \Phi \vdash \iota : \Gamma \triangleright \Gamma \rrbracket_M = \mathtt{Id}_\Gamma : \Gamma \to \Gamma \in \mathbb{C}(I) \quad \llbracket \Phi \vdash \omega\pi : \Gamma', ax \triangleright \Gamma \rrbracket_M = \llbracket \Phi \vdash \omega : \Gamma' \triangleright \Gamma \rrbracket_M \circ \pi_1 : \Gamma' \times A \to \Gamma$$

$$\llbracket \Phi \vdash \omega\times : \Gamma', x : A \triangleright \Gamma, x : B \rrbracket_M = \llbracket \Phi \vdash \omega : \Gamma' \triangleright \Gamma \rrbracket_M \times \llbracket A \leq :_\Phi B \rrbracket_M : \Gamma' \times A \to \Gamma \times B$$

Type-environment substitutions are also derived inductively with respect to a effect environment.

$$(\text{Nil})\frac{\Phi \vdash \Gamma' \mathtt{Ok}}{\Phi \mid \Gamma' \vdash \diamond : \diamond} \quad (\text{Extend})\frac{\Phi \mid \Gamma' \vdash \sigma : \Gamma \quad x \notin \mathtt{dom}(\Gamma) \quad \Phi \mid \Gamma' \vdash v : A}{\Phi \mid \Gamma' \vdash (\sigma, x := v) : (\Gamma, x : A)}$$

With denotations defined as morphisms in the appropriate fibre category: $\llbracket \Phi \mid \Gamma' \vdash \sigma : \Gamma \rrbracket_M : \Gamma' \to \in \mathbb{C}(I)$

$$\llbracket \Phi \mid \Gamma' \vdash \diamond : \diamond \rrbracket_M = \langle\rangle_{\Gamma'} : \Gamma' \to \mathtt{1} \quad \llbracket \Phi \mid \Gamma' \vdash (\sigma, x := v) : \Gamma, x : A \rrbracket_M = \langle \llbracket \Phi \mid \Gamma' \vdash \Gamma : \rrbracket_{\mathbf{M}}, \llbracket \boldsymbol{\Phi} \mid \boldsymbol{\Gamma}' \vdash \mathbf{v} : \mathbf{A} \rrbracket_{\mathbf{M}} \rangle : \Gamma' \to \Gamma \times \mathtt{1}$$

TODO: Explain weakening theorem on these

Firstly, we must consider the action of effect weakenings on these morphisms. The weakening theorem on term-environment weakenings is as so:

Theorem 5.4.10 (Effect Weakening on Term Weakening) *If $\omega_1 : \Phi' \triangleright \Phi$ and $\Phi \vdash \omega : \Gamma' \triangleright \Gamma$ then $\Phi' \vdash \omega : \Gamma' \triangleright G$ and $\llbracket \Phi' \vdash \omega : \Gamma' \triangleright \Gamma \rrbracket_M = \omega_1^* \llbracket \Phi \vdash \omega : \Gamma' \triangleright \Gamma \rrbracket_M$.*

Proof: By induction on the derivation of ω. making use of weakening on types, type environments, and sub-typing. **TODO: Pick some cases.**

Secondly, we can form the weakening theorem on term-environment substitutions.

Theorem 5.4.11 (Effect Weakening on Term Substitution) *If $\Phi \mid \Gamma' \vdash \sigma : \Gamma$ and $\omega : \Phi' \triangleright \Phi$ then $\Phi' \mid \Gamma' \vdash \sigma : \Gamma$ and $\llbracket \Phi' \mid \Gamma' \vdash \sigma : \Gamma \rrbracket_M = \omega^* \llbracket \Phi \mid \Gamma' \vdash \sigma : \Gamma \rrbracket_M$*

Proof: By induction on the definition of σ, making use of the weakening on terms, types, and type environments. **TODO: Both cases.**

TODO: Be careful about denotations of derivations. TODO: Show cases that make use of the above therorems

Now we can move onto the term substitution and weakening theorems. These theorems are the final step before we prove that all denotations for a typing relation are equivalent and then move onto soundness. **TODO: Finish this sentence** These theorems show that we can construct

The term-weakening theorem is as so:

Theorem 5.4.12 (Term Weakening) *If $\Phi \vdash \omega : \Gamma' \triangleright \Gamma$ and Δ is a derivation of $\Phi \mid \Gamma \vdash v : A$ then we can derive Δ', a derivation of $\Phi \mid \Gamma' \vdash v : A$ with denotation $\Delta' = \Delta \circ \omega$.*

Proof: By induction on the derivation of Δ. Making use of the weakening of effect environments on term weakenings. **TODO: Cases, including polymorphic gen case which makes use of weakening from theorem above. Maybe pick the If case too as a more complex case?**

The term-substitution theorem is formed similarly:

Theorem 5.4.13 (Term Substitution) *If $\Phi \mid \Gamma' \vdash \sigma : \Gamma$ and Δ is a derivation of $\Phi \mid \Gamma \vdash v : A$, then we can construct Δ' deriving $\Phi \mid \Gamma' \vdash v[\sigma] : A$ with denotation $\Delta' = \Delta \circ \sigma$.*

Proof: By induction on the derivation Δ, making use of weakening of term substitutions. **TODO: Pick some cases**

5.5 Uniqueness of Denotations

Up until this point we have hat to be careful about the implicit typing derivation for every term denotation $[\![\Phi \mid \Gamma \vdash v\colon A]\!]_M$. We are now equipped with the tools to prove that all derivations of the same typing relation instance $\Phi \mid \Gamma \vdash v\colon A$ induce the same denotation. This allows us to no-longer worry about the equality of denotations, which will be helpful in the soundness proof.

To prove that all typing denotations have the same denotation, I shall first introduce the concept of a *reduced* typing derivation that is unique to each term and type in each typing environment. Next, I shall present a function, *reduce*, which recursively maps typing derivations to their reduced equivalent. I shall also prove that this function preserves the denotation of the derivations. That is $[\![\Phi \mid \Gamma \vdash v\colon A]\!]_M = [\![\mathit{reduce}(\Phi \mid \Gamma \vdash v\colon A)]\!]_M$. Hence, we can conclude, since all derivations for a typing relation instance reduce to the same unique typing derivation, and that the reduction function preserves the denotations, that all derivations of a typing derivation have the same denotation.

The need for reduced typing derivations comes about because of sub-typing. The sub-typing rule can be inserted into different places in a derivation to derive the same typing relation. Hence, the reduction function focuses on only placing sub-typing rule instances in specific places. In particular, we shall only allow the sub-typing rule to occur at the root of the derivation or above instances of the if-rule and apply-rule in reduced derivations. This has the effect of only introducing sub-typing rule uses when it is necessary maintain syntactic-correctness.

Theorem 5.5.1 (Uniqueness of reduced Derivations) *These reduced derivations are unique.*

Proof: By induction on term structure. **TODO: Cases (in particular, Apply) TODO: Point out that we don't have a case for subtyping as it's not a syntactic feature.**

The *reduce* function maps each derivation to its reduced equivalent. It shall do this by pushing sub-typing rules from the leaves of the derivation tree down towards the root of the tree. The function case splits on the root of the tree and works up recursively. Some cases of the function are given below. **TODO: These cases**.

Theorem 5.5.2 (Reduction preserves denotations) *If the derivation Δ' is the result of applying reduce to Δ then the denotations of the derivations are equal. That is $\Delta' = \mathit{reduce}(\Delta) \implies \Delta' = \Delta$.*

Proof: By induction over the structure of Δ, making use of the substitution and weakening theorems. (**TODO: How? more specificity**)

TODO: Pick cases (Apply, polymorphism gen rule)

5.6 Soundness

We are now at a stage where we can state and prove the most important theorem for a denotational semantics: soundness with respect to $\beta\eta$-reductions. Soundness follows from the common sense requirement that terms that are equivalent in a given language should also have equivalent denotations. In our case, we shall introduce a $\beta\eta$-equivalence relation and then prove that equivalent terms have equal denotations.

The $\beta\eta$-equivalence relation is a rule based relation with three main flavours of rules. Firstly, there are the $\beta\eta$-reductions which formalise how we expect the program to execute given an appropriate implementation. We give a $\beta\eta$-reduction for each term transition, such as the application of lambda terms or the execution of an if-expression. Secondly, there are congruences, formalise how the reduction of sub-expressions affects the rest of the expression in a compositional way. Finally, we extend this relation into an equivalence relation by closing it under transitivity, reflexivity and symmetry.

TODO: Figure with the beta-eta-reductions

Now we can state the $\beta\eta$-soundness theorem.

Theorem 5.6.1 (Soundness) *If* $\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2 \colon A$, *then* $\Phi \mid \Gamma \vdash v_1 \colon A$, $\Phi \mid \Gamma \vdash v_2 \colon A$, *and* $[\![\Phi \mid \Gamma \vdash v_1 \colon A]\!]_M = [\![\Phi \mid \Gamma \vdash v_2 \colon A]\!]_M$.

Proof: By rule induction on the definition of $\beta\eta$-equivalence, making use of monad laws for monadic cases, the adjunction for effect-polymorphic cases, the co-product for if-expressions, and the CCC property of S-categories for the lambda-term $\beta\eta$-reductions.

TODO: Cases. How many?

This completes the proof of soundness for this semantics.

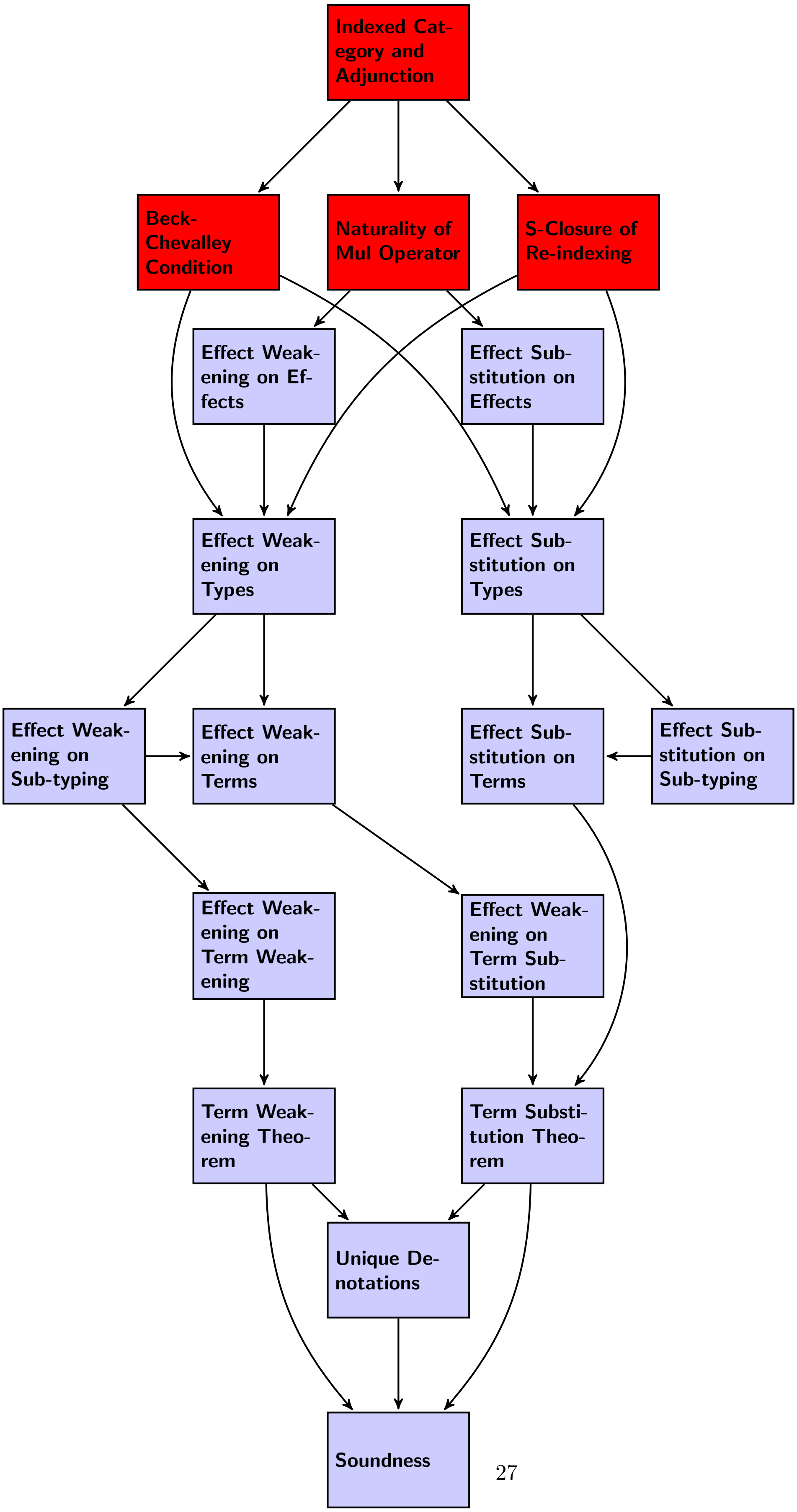

Figure 5.1: A road map of the proof dependencies. Assumptions in red, theorems in blue

$$(\text{Lambda})\frac{(\text{Sub-Type})\frac{\Phi|\Gamma,x:A\vdash v:B \quad B\leq:_\Phi B'}{\Phi|\Gamma,x:A\vdash v:B'}}{\Phi \mid \Gamma \vdash \lambda x : A.v\colon A \to B'} \tag{5.7}$$

$$(\text{Sub-Type})\frac{(\text{Lambda})\frac{\Phi|\Gamma,x:A\vdash v:B}{\Phi|\Gamma\vdash\lambda x:A.v:A\to B} \quad A \to B \leq: A \to B'}{\Phi \mid \Gamma \vdash \lambda x : A.v\colon A \to B'} \tag{5.8}$$

Figure 5.2: Two derivations of the same type-relation. The second derivation is reduced.

$$(\text{Lambda-Beta})\frac{\Phi \mid \Gamma, x : A \vdash v_2\colon B \quad \Phi \mid \Gamma \vdash v_1\colon A}{\Phi \mid \Gamma \vdash (\lambda x : A.v_1)\ v_2 =_{\beta\eta} v_1\,[v_2/x]\colon B} \quad (\text{Lambda-Eta})\frac{\Phi \mid \Gamma \vdash v\colon A \to B}{\Phi \mid \Gamma \vdash \lambda x : A.(v\ x) =_{\beta\eta} v\colon A \to B}$$

$$(\text{Left Unit})\frac{\Phi \mid \Gamma \vdash v_1\colon A \quad \Phi \mid \Gamma, x : A \vdash v_2\colon \mathtt{M}_\epsilon B}{\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow \mathtt{return} v_1\ \mathtt{in}\ v_2 =_{\beta\eta} v_2\,[v_1/x]\colon \mathtt{M}_\epsilon B} \quad (\text{Right Unit})\frac{\Phi \mid \Gamma \vdash v\colon \mathtt{M}_\epsilon A}{\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow v\ \mathtt{in}\ \mathtt{return} x =_{\beta\eta} v\colon \mathtt{M}_\epsilon A}$$

$$(\text{Associativity})\frac{\Phi \mid \Gamma \vdash v_1\colon \mathtt{M}_{\epsilon_1} A \quad \Phi \mid \Gamma, x : A \vdash v_2\colon \mathtt{M}_{\epsilon_2} B \quad \Phi \mid \Gamma, y : B \vdash v_3\colon \mathtt{M}_{\epsilon_3} C}{\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ (\mathtt{do}\ y \leftarrow v_2\ \mathtt{in}\ v_3) =_{\beta\eta} \mathtt{do}\ y \leftarrow (\mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2)\ \mathtt{in}\ v_3\colon \mathtt{M}_{\epsilon_1 \cdot \epsilon_2 \cdot \epsilon_3} C}$$

$$(\text{Unit})\frac{\Phi \mid \Gamma \vdash v\colon \mathtt{Unit}}{\Phi \mid \Gamma \vdash v =_{\beta\eta} \mathtt{()}\colon \mathtt{Unit}}$$

$$(\text{if-true})\frac{\Phi \mid \Gamma \vdash v_1\colon A \quad \Phi \mid \Gamma \vdash v_2\colon A}{\Phi \mid \Gamma \vdash \mathtt{if}_A\ \mathtt{true}\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 =_{\beta\eta} v_1\colon A} \quad (\text{if-false})\frac{\Phi \mid \Gamma \vdash v_2\colon A \quad \Phi \mid \Gamma \vdash v_1\colon A}{\Phi \mid \Gamma \vdash \mathtt{if}_A\ \mathtt{false}\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 =_{\beta\eta} v_2\colon A}$$

$$(\text{If-Eta})\frac{\Phi \mid \Gamma, x : \mathtt{Bool} \vdash v_2\colon A \quad \Phi \mid \Gamma \vdash v_1\colon \mathtt{Bool}}{\Phi \mid \Gamma \vdash \mathtt{if}_A\ v_1\ \mathtt{then}\ v_2\,[\mathtt{true}/x]\ \mathtt{else}\ v_2\,[\mathtt{false}/x] =_{\beta\eta} v_2\,[v_1/x]\colon A}$$

$$(\text{Effect-beta})\frac{\Phi \vdash \epsilon \quad \Phi, \alpha \mid \Gamma \vdash v\colon A}{\Phi \mid \Gamma \vdash (\Lambda\alpha.v\ \epsilon) =_{\beta\eta} v\,[\epsilon/\alpha]\colon A\,[\epsilon/\alpha]} \quad (\text{Effect-eta})\frac{\Phi \mid \Gamma \vdash v\colon \forall\alpha.A}{\Phi \mid \Gamma \vdash \Lambda\alpha.(v\ \alpha) =_{\beta\eta} v\colon \forall\alpha.A}$$

Figure 5.3: The $\beta\eta$-reduction rules for PEC

$$(\text{Effect-Abs})\frac{\Phi, \alpha \mid \Gamma \vdash v_1 =_{\beta\eta} v_2\colon A}{\Phi \mid \Gamma \vdash \Lambda\alpha.v_1 =_{\beta\eta} \Lambda\alpha.v_2\colon \forall\alpha.A} \quad (\text{Effect-Apply})\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2\colon \forall\alpha.A \quad \Phi \vdash \epsilon}{\Phi \mid \Gamma \vdash v_1\ \epsilon =_{\beta\eta} v_2\ \epsilon\colon A\,[\epsilon/\alpha]}$$

$$(\text{Lambda})\frac{\Phi \mid \Gamma, x : A \vdash v_1 =_{\beta\eta} v_2\colon B}{\Phi \mid \Gamma \vdash \lambda x : A.v_1 =_{\beta\eta} \lambda x : A.v_2\colon A \to B} \quad (\text{Return})\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2\colon A}{\Phi \mid \Gamma \vdash \mathtt{return} v_1 =_{\beta\eta} \mathtt{return} v_2\colon \mathtt{M}_1 A}$$

$$(\text{Apply})\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_1'\colon A \to B \quad \Phi \mid \Gamma \vdash v_2 =_{\beta\eta} v_2'\colon A}{\Phi \mid \Gamma \vdash v_1\ v_2 =_{\beta\eta} v_1'\ v_2'\colon B} \quad (\text{Bind})\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_1'\colon \mathtt{M}_{\epsilon_1} A \quad \Phi \mid \Gamma, x : A \vdash v_2 =_{\beta\eta} v_2'\colon \mathtt{M}_{\epsilon_2} B}{\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 =_{\beta\eta} \mathtt{do}\ c \leftarrow v_1'\ \mathtt{in}\ v_2'\colon \mathtt{M}_{\epsilon_1 \cdot \epsilon_2} B}$$

$$(\text{If})\frac{\Phi \mid \Gamma \vdash v =_{\beta\eta} v'\colon \mathtt{Bool} \quad \Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_1'\colon A \quad \Phi \mid \Gamma \vdash v_2 =_{\beta\eta} v_2'\colon A}{\Phi \mid \Gamma \vdash \mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 =_{\beta\eta} \mathtt{if}_A\ v\ \mathtt{then}\ v_1'\ \mathtt{else}\ v_2'\colon A} \quad (\text{Subtype})\frac{\Phi \mid \Gamma \vdash v =_{\beta\eta} v'\colon A \quad A \leq:_\Phi B}{\Phi \mid \Gamma \vdash v =_{\beta\eta} v'\colon B}$$

Figure 5.4: The congruence rules for PEC

$$\text{(Reflexive)}\frac{\Phi \mid \Gamma \vdash v\colon A}{\Phi \mid \Gamma \vdash v =_{\beta\eta} v\colon A} \qquad \text{(Symmetric)}\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2\colon A}{\Phi \mid \Gamma \vdash v_2 =_{\beta\eta} v_1\colon A}$$

$$\text{(Transitive)}\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2\colon A \quad \Phi \mid \Gamma \vdash v_2 =_{\beta\eta} v_3\colon A}{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_3\colon A}$$

Figure 5.5: Rules expanding the $\beta\eta$-reduction and congruence relation to an equivalence relation

Chapter 6

Instantiating a Model of PEC

Now we have proved that we can form a model of PEC in an appropriate indexed S-category, it remains to show that it is feasible to construct such an indexed category. There exist `Set`-based models for the semantics of effect-ful languages with a graded monad, such as the Effect Calculus **TODO: A reference for this**. More specifically, it is possible to treat `Set` as an S-category. Hence, I shall use a `Set` based S-category as a starting point. In this section, I shall demonstrate how to construct an strictly indexed S-category which can model the PEC from such an S-category.

Let $\mathbb{C}$ be an S-category formed from `Set`.(**TODO: Is this correct parlance?**) That is, $\mathbb{C}$ contains a graded monad $\mathtt{T}^0, \mu^0_{,,}, \eta^0, \mathtt{t}^0_{,,}$, is cartesian closed, has a co-product on the terminal object $\mathtt{1} = \{*\}$, has sub-typing functions $[\![A \leq:_\gamma B]\!]_M : A \to B$ for each instance of the ground sub-typing relation, and has natural transformations $[\![\epsilon_1 \leq_0 \epsilon_2]\!]_M : \mathtt{T}^0_{\epsilon_1} \to \mathtt{T}^0_{\epsilon_2}$. Since $\mathbb{C}$ is a model for the EC, it is graded by a pre-ordered monoid on ground effects: $(E, \leq_0, \mathtt{1}, \cdot)$. I have indexed each of these S-category properties with 0 to indicate that they occur in the bottom category $\mathbb{C}$, induced by the empty effect environment.

Since, using α-equivalence, we can see that all effect environments of the same length are equivalent, we can reduce an effect environment to the natural number n indicating its length.

Next we shall pick our fibre-categories for non-zero values of n. A simple instantiation is to pick each fibre $\mathbb{C}(n)$ to be the functor-category $[E^n, \mathbb{C}]$. That is, the category of functions returning an object in $\mathbb{C}$ given n ground effects. Morphisms between objects are point-wise (?) functions between th results of their objects in C. If $m \in [E^n, \mathbb{C}](A, B)$ then $m\vec{\epsilon} \in \mathbb{C}(A\vec{\epsilon}, B\vec{\epsilon})$. Shortly, I shall prove that these categories are indeed S-closed.

We also need to define the base category. This shall be `Eff`, the discrete sub-category of `Set`, populated by the set of ground effects E as the effect-object U, and its finite products.

$$E^0 = \mathtt{1} = \{*\} \tag{6.1}$$

$$E^{n+1} = E^n \times E \tag{6.2}$$

Morphisms $E^n \to E$ are functions taking n ground-effect parameters and returning a ground effect.

The fibres, $[E^n, C]$ are S-categories. This can be proved by constructing the S-category structures point-wise with respect to their parameter $\vec{\epsilon} \in E^n$.

TODO: Fill in all the S-category constructions

Now we need to define the required morphisms between fibres. Firstly, for any function $\theta : E^m \to E^n$ in `Eff`, there should exist the re-indexing functor $\theta^* : [E^n, \mathbb{C}] \to [E^m, \mathbb{C}]$. A simple instantiation is the pre-composition functor.

$$
\begin{aligned}
A \in \quad & [E^n, \mathbb{C}] && (6.3)\\
\theta^*(A)\vec{\epsilon_m} = \quad & A(\theta(\vec{\epsilon_m})) && (6.4)\\
f : \quad & A \to B && (6.5)\\
\theta^*(f)\vec{\epsilon_m} = \quad & f(\theta(\vec{\epsilon_m})) : \theta^*(A) \to \theta^*(B) && (6.6)\\
& && (6.7)
\end{aligned}
$$

This also obeys the composition law of re-indexing functors.

$$
\begin{aligned}
\theta^*(\phi^* A)\vec{\epsilon} &= \phi^*(A)(\theta\vec{\epsilon}) && (6.8)\\
&= A(\phi(\theta\vec{\epsilon})) && (6.9)\\
&= A((\phi \circ \theta)\vec{\epsilon}) && (6.10)\\
&= (\phi \circ \theta)^*(A)\vec{\epsilon} && (6.11)
\end{aligned}
$$

The re-indexing functors are also S-closed, since all of the S-category features are proved point-wise.

Proof: TODO: Cases: Exponential and Bind?

Next, the re-indexing functor π_1^* should have a right-adjoint, $\forall_{E^n}$. Here, we shall pick $\forall_{E^n}$ to be defined as a finite product over the countable set of effects. This is possible, since types and effects are not impredicative (that is, they quantify over themselves)

$$
\begin{aligned}
\forall_{E^n} :& [E^{n+1}, \mathbb{C}] \to [E^n, \mathbb{C}] && (6.12)\\
\forall_{E^n}(A)\vec{\epsilon_n} =& \Pi_{\epsilon \in E} A(\vec{\epsilon_n}, \epsilon) && (6.13)\\
\forall_{E^n}(f)\vec{\epsilon_n} =& \Pi_{\epsilon \in E} f(\vec{\epsilon_n}, \epsilon) && (6.14)
\end{aligned}
$$

TODO: Is the functor S-closed?

We can now prove that $\pi_1^* \dashv \forall_{E^n}$. To do this, we need functors natural a natural bijection between morphisms in the $[E^n, \mathbb{C}]$ and $[E^{n+1}, \mathbb{C}]$.

$$
\overline{(-)} : [E^{n+1}, \mathbb{C}](\pi_1^* A, B) \rightleftharpoons [E^n, \mathbb{C}](A, \forall_{E^n} B) : \widehat{(-)} \tag{6.15}
$$

The left- and right-wards components of this bijection can be derived as follows. The leftwards component maps each morphism to a finite pairing of the morphism over each ground effect.

$$
\begin{aligned}
m : \quad & \pi_1^* A \to B && (6.16)\\
\overline{m} : \quad & A \to \forall_{E^n} B && (6.17)\\
\overline{m}(\vec{\epsilon_n}) = \quad & \langle m(\vec{\epsilon_n}, \epsilon) \rangle_{e \in E} && (6.18)
\end{aligned}
$$

The inverse is simply to project out the appropriate value of ϵ from the product.

$$
\begin{aligned}
n : \quad & A \to \forall_{E^n} B && (6.19)\\
\widehat{n} : \quad & \pi_1^* A \to B && (6.20)\\
\widehat{n}(\vec{\epsilon_n}, \epsilon_{n+1}) = \quad & \pi_\epsilon \circ g(\vec{\epsilon_n}) && (6.21)
\end{aligned}
$$

These transformations give rise to the unit and co-unit of the adjunction:
With unit:

$$
\begin{aligned}
\boldsymbol{\eta}_A : \quad & A \to \forall_{E^n} \pi_1^* A && (6.22)\\
\boldsymbol{\eta}_A(\vec{\epsilon_n}) = \quad & \left\langle \mathtt{Id}_{A(\vec{\epsilon_n}, e)} \right\rangle_{\epsilon \in E} && (6.23)
\end{aligned}
$$

And co-unit

$$
\begin{aligned}
\boldsymbol{\epsilon}_B : \quad & \pi_1^* \forall_{E^n} B \to B && (6.24)\\
\boldsymbol{\epsilon}_B(\vec{\epsilon_n}, \epsilon) = \quad & \pi_\epsilon : \Pi_{e \in E} B(\vec{\epsilon_n}, \epsilon) \to \Pi_{e \in E} B(\vec{\epsilon_n}, \epsilon) && (6.25)
\end{aligned}
$$

The unit and co-unit allow us to prove that this construction is an adjunction. For any $g : \pi_1^* A \to B$,

$$
\begin{aligned}
(\boldsymbol{\epsilon}_B \circ \pi_1^*(\overline{g}))(\vec{\epsilon_n}, \epsilon_{n+1}) &= \pi_{\epsilon_{n+1}} \circ \langle g(\vec{\epsilon_n}, \epsilon') \rangle_{\epsilon' \in E} && (6.26)\\
&= g(\vec{\epsilon_n}, \epsilon_{n+1}) && (6.27)
\end{aligned}
$$

So $\boldsymbol{\epsilon}_B \circ \pi_1^*(\overline{g}) = g$.
Finally, we need to prove that the beck-chevalley condition holds For $\theta : E^m \to E^n$.
Firstly, the functors $(\theta^* \circ \forall_{E^n})$ and $(\forall_{E^m} \circ (\theta \times \mathtt{Id}_E)^*)$ are equal.

Proof:

$$
\begin{aligned}
((\theta^* \circ \forall_{E^n})A)\vec{\epsilon_n} &= \theta^*(\forall_{E^n} A)\vec{\epsilon_n} && (6.28)\\
&= (\forall_{E^n} A)(\theta(\vec{\epsilon_n})) && (6.29)\\
&= \Pi_{\epsilon \in E}(A(\theta(\vec{\epsilon_n}), \epsilon)) && (6.30)\\
&= \Pi_{\epsilon \in E}(((\theta \times \mathtt{Id}_U)^* A)(\vec{\epsilon_n}, \epsilon)) && (6.31)\\
&= \forall_{E^m}((\theta \times \mathtt{Id}_E)^* A)\vec{\epsilon_n} && (6.32)\\
&= ((\forall_{E^m} \circ (\theta \times \mathtt{Id}_E)^*)A)\vec{\epsilon_n} && (6.33)
\end{aligned}
$$

Secondly, the natural transformation $\overline{(\theta \times \mathtt{Id}_U)^* \boldsymbol{\epsilon}}$ is equal to the identity natural transformation.

Proof:

$$
\begin{aligned}
\overline{(\theta \times \mathtt{Id}_U)^* \boldsymbol{\epsilon}_A}\vec{\epsilon} &= \langle (\theta \times \mathtt{Id}_U)^* \boldsymbol{\epsilon}_A(\vec{\epsilon}, \epsilon) \rangle_{\epsilon \in E} && (6.34)\\
&= \langle \boldsymbol{\epsilon}_A(\theta\vec{\epsilon}, \epsilon) \rangle_{e \in E} && (6.35)\\
&= \langle \pi_\epsilon \rangle_{\epsilon \in E} : \Pi_{\epsilon \in E} A(\theta\vec{\epsilon}, \epsilon) \to \Pi_{\epsilon \in E} A(\theta\vec{\epsilon}, \epsilon) && (6.36)\\
&= \mathtt{Id}_{\Pi_{\epsilon \in E} A(\theta\vec{\epsilon}, \epsilon)} && (6.37)\\
&= \mathtt{Id}_{\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^* A}\vec{\epsilon} && (6.38)\\
&= \mathtt{Id}_{\theta^* \circ \forall_I} && (6.39)
\end{aligned}
$$

Hence we have proof that our construction is indeed a valid indexed S-category.