A Denotational Semantics for Polymorphic Effect Systems

Alexander Taylor

March 18, 2019

TODO: Fix Spacing: Much of the maths lacks proper spacing. Latex knows about (e.g.) $n + n \log n$ but it doesn't know how to format $\vdash$ ... : as a loosely binding ternary operator in rules etc. Use explicit spacing. (or even).

TODO: Question: do the requirements that you have imply that each hom-set C(I,U) forms a monoid? If they do, it might be worth mentioning this (it makes the requirements on the base category look more natural). (Also, is requiring it to form a preordered monoid a good idea?)

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 introduce a modern definition of a lambda-calculus-based language with an explicit graded monad to handle a variety of effects. This calculus is then extended using polymorphic terms to yield a more general polymorphic effect calculus. Next, I 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 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.

Contents

Chapter 1

Introduction

TODO: "There's a bit of a problem (to me) with the introduction (1.2), because many things are done in quite some detail when what the reader needs is a bit of an intuitive view of what the problem is and where the rest of the document is going."

TODO: I think you need to talk more about what effect *systems* (in particular, types carrying effect information) are before talking about polymorphism.

TODO: "The concepts defined in 2"??? TODO: "Coproduct on the terminal object" is slightly ambiguous. TODO: Unit effect: slightly unclear what this means. TODO: Algebra on effects": what algebra? This is the first mention of effect algebra. TODO: "A simple example of such an algebra": this is the most general notion of effect algebra, not a simple example. TODO: "Over a property P": property is definitely the wrong word to use here. TODO: Paragraph about "augmented type rules": going too technical too quickly (adjunctions etc.) TODO: "as we shall see, in effect-polymorphic languages, types quantify over effects": explain this earlier, because this is the point of the project. TODO: "class of categories": this is not a class of categories: each model has more structure. TODO: Figure 1.2: it isn't clear that Φ_1 is Φ_0, α, etc. Also dotted lines probably aren't a good dea (since they are used for unique morphisms).

1.1 What is an Effect System?

Programs very rarely stand alone without interacting with their environment in some form. This interaction might be to receive input, such as from sensors or buttons, to write output to a file, or to halt with an error. In functional languages, manipulating mutable state can also be considered interacting with the environment. Language terms that produce interactions are known as "effects" as they have some observable effect in the environment other than simply the value that executing the term produces. It is important to be able to reason about the effects that a program term may produce, in order to ensure the soundness of compiler transformations and formal verification. Languages such as Haskell and the languages introduced later in this dissertation do not allow implicit effects. Instead they require the programmer to explicitly use structures called monads **TODO: Reference Moggi** to encode effect analysis in the type of program fragments.

1.2 What is Effect Polymorphism?

Effect polymorphism is when a single 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 **TODO: introduce effect-in-signatures**. This manifests in a similar manner to type parameter polymorphism in system-F-based languages. Consider the following Scala-style pseudocode:

```
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 situations 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. **TODO: Verifying/formally reasoning about?**

1.3 An Introduction to Categorical Semantics

In this dissertation, I describe a denotational semantics using category theory. A denotational semantics for a language is a mapping, the image $[\![X]\!]_M$ of which is known as a denotation, 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.

TODO: Non-trivial, other properties - too vague

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.

TODO: Assuming objects in $\mathbb{C}$ to give types, envs

$$A : \texttt{Type} \mapsto [\![A]\!]_M \in \texttt{obj}\ \mathbb{C} \tag{1.1}$$
$$\Gamma \mapsto [\![\Gamma]\!]_M \in \texttt{obj}\ \mathbb{C} \tag{1.2}$$

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

TODO: What is a typing relation? TODO: Instance of type relation

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

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.

TODO: Maybe better to introduce an equational equivalence?

TODO: "E.g. at various points you stress beta-eta, but never tell me why. Does this suffice, or are there other reduction rules which produce equational axions? There is an axiom (associativity for bind) that doesn't really fit into beta or eta. So the name beta-eta equality should probably be avoided."

TODO: Probably need to make this change elsewhere

TODO: Use a another, general, equational equivalence

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

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. A collection of so-called *substitution* theorems allow us to quantify this action on denotations in a category-theoretic way. For example, if $\Gamma, x : A \vdash v_1\colon B$ and $\Gamma \vdash v_2\colon A$ then $[\![\Gamma \vdash v_1\,[v_2/x] : B]\!]_M$ should be derivable from $[\![\Gamma, x : A \vdash v_1\colon B]\!]_M$ and $[\![\Gamma \vdash v_2\colon A]\!]_M$.

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 **TODO: = Larger** environment from the denotation of the same term in a stronger environment. For example, if $\Gamma' \leq_{\text{weaker}} \Gamma$ then $[\![\Gamma' \vdash v\colon A]\!]_M$ should be derivable from $[\![\Gamma \vdash v\colon A]\!]_M$.

Chapter 2

Background

TODO: This intro needs improvement In this chapter I introduce the monadic, effectful language used in the rest of the dissertation, known from now on as the Effect Calculus (EC). Following this, I then introduce polymorphic terms to the EC which yield the Polymorphic Effect Calculus (PEC).

2.1 Required Category Theory

TODO: Make this more tutorial

TODO: Cite Mac Lane (categories for the working mathematician)?

TODO: Products: binary products? TODO: Some of the definitions (product, etc.) don't really sound like definitions (they talk about when something exists). Say "a product of X and Y is/consists of...". TODO: I for initial object: usually 0. (Also, you don't seem to be using the initial object anywhere.) TODO: "conceptual dual": just dual. TODO: "preserves the category properties of composition and identity": these are structure, not properties. Just say "preserves composition and identities". TODO: You switch from using "X, Y, ..." from objects to "A, B, ...". TODO: "a monad is": a monad consists of? TODO: It might be better to merge some of the figures. For example, all of the diagrams required to commute to get a monad could go into one figure (or maybe just in the text?). This would make it easy to see that they go together.

TODO: Name the base category and the indexed category! TODO: It might be worthwhile to point out the contravariance when you mention the reindexing functors. (Since this is easy to miss.) TODO: Category of cartesian-closed categories: you don't define this. You also might not want to refer to this at all.

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.

2.1.1 Cartesian Closed Category

A category is cartesian closed if it has a terminal object, products for all pairs of objects, and exponentials. These concepts will be explained below.

Terminal Object

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

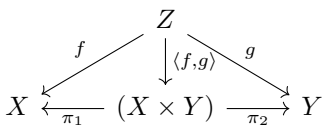
Products

TODO: There is a ..., written ..., for every... 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,

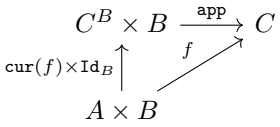
$X \xleftarrow{f} Z \xrightarrow{g} Y$

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



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.



2.1.2 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.

2.1.3 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:

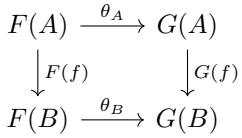
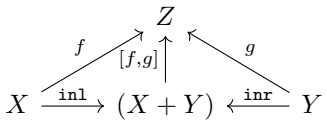


Figure 2.1: Naturality of a natural transformation



2.1.4 Functors

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

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

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.

$$F(\texttt{Id}_A) = \texttt{Id}_{FA} \tag{2.3}$$
$$F(g \circ f) = F(g) \circ F(f) \tag{2.4}$$

2.1.5 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 2.1 commutes for each $f : A \to B \in \mathbb{C}$.

2.1.6 Monad

TODO: Not actually famous A monad is famously "a monoid on the category of endofunctors". In less opaque terms, a monad is:

TODO: Use italics rather than quotes

- A functor, $T : \mathbb{C} \to C$, from $\mathbb{C}$ onto itself, also known as an *endofunctor*.
- 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 2.2, 2.3 commute.

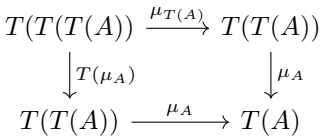


Figure 2.2: Monad Associativity Laws

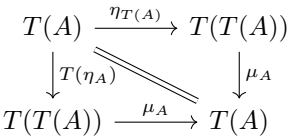


Figure 2.3: Monad Left- and Right-Unit laws

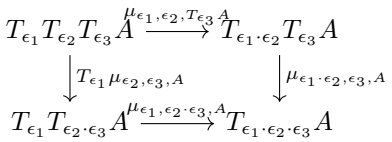


Figure 2.4: Associativity of a graded monad

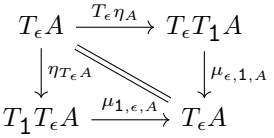


Figure 2.5: Left- and Right- Units of a graded monad

2.1.7 Graded Monad

TODO: "E.g. section 2.7 assumes a "monoidal algebra". There should be a definition of preordered monoid somewhere!" A graded monad is a generalisation of a monad to be indexed by a monoidal algebra E **TODO: Define a monoidal algebra**. It is made up of:

- An endofunctor 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 2.4, 2.16 commute.

2.1.8 Tensor Strength

TODO: section 2.8 on strength. Read the first sentence in isolation! The critical idea that we need the operation of strength to be able to do various programming things is missing. I think strength is quite hard to motivate (given that it's essentially invisible to the programming language), but there should still be something there. 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*. Hence a graded monad with tensorial strength is known as a *strong graded monad*. Tensorial strength consists of a natural transformation:

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

This is required to have 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 2.6, 2.7, 2.8, 2.9, 2.10, 2.11.

2.1.9 Adjunction

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

Given functors F, G:

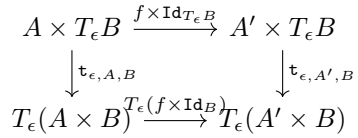


Figure 2.6: Left Naturality of Graded Tensor Strength

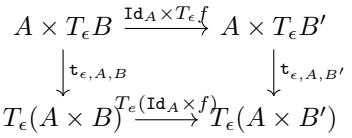


Figure 2.7: Right Naturality of Graded Tensor Strength

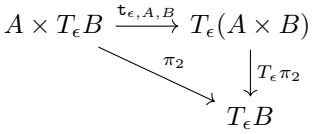


Figure 2.8: Tensor Strength Unitor Law

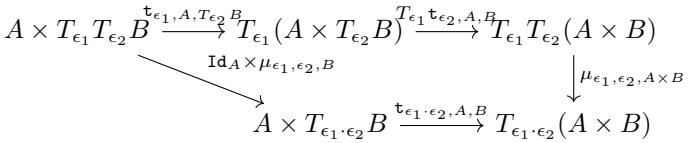


Figure 2.9: How the tensor strength natural transformation commutes with the join natural transformation

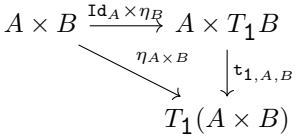


Figure 2.10: How the tensor strength natural transformation commutes with the unit natural transformation

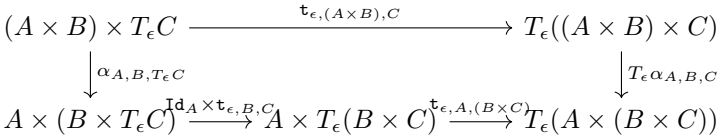
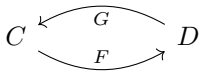


Figure 2.11: Tensorial strength commutes with the reordering natural transformation.



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.

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

2.1.10 Strictly Indexed Category

The final piece of category theory required to understand this dissertation is the concept of a strictly indexed Category. A strictly 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, written $f^* : \mathbb{C}(A) \to \mathbb{C}(B)$, between categories in the target category. The strictly adverb indicates that the indexing in this construction is done by a functor as opposed to the weaker pseudofunctor (weak functor) structure. Since pseudofunctors are not needed to explain anything in this project, I shall leave out their definition, though an interested reader may wish to research them further. [1]

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, $\mathbb{P}$:

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

2.2 Language Features 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 2.1, I shall now give an introduction to which category-theoretic structures are required to interpret different language features.

[1] https://ncatlab.org/nlab/show/pseudofunctor

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 2.1.1).

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

TODO: "There definitely needs to be some explanation of what the terms from Moggi's language (i.e. bind and return) and mean intuitively. Similarly for the new constructs (even though they look a lot like System F). I especially think the type system needs more explanation. Understanding the typing rules for bind and return is probably hard for anyone who hasn't seen graded monads before (i.e. almost everyone)." 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. The monad needs to be strong in order to allow access to variables in the environment from within the monad an example of this can be seen in figure2.12.

$$(\text{Return})\frac{\Gamma \vdash v \colon A}{\Gamma \vdash \mathtt{return} v \colon \mathtt{M} A} \quad (\text{Bind})\frac{\Gamma \vdash v_1 \colon \mathtt{M} A \quad \Gamma, x \colon A \vdash v_2 \colon \mathtt{M} B}{\Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \colon \mathtt{M} B} \tag{2.15}$$

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 subtyping 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 2.16.

TODO: "Do some effect analysis" is a little sloppy

$$(\text{Return})\frac{\Gamma \vdash v \colon A}{\Gamma \vdash \mathtt{return} v \colon \mathtt{M}_{\mathtt{1}} A} \quad (\text{Bind})\frac{\Gamma \vdash v_1 \colon \mathtt{M}_{\epsilon_1} A \quad \Gamma, x \colon A \vdash v_2 \colon \mathtt{M}_{\epsilon_2} B}{\Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \colon \mathtt{M}_{\epsilon_1 \cdot \epsilon_2} B} \tag{2.16}$$

```
let x = 5 in (
    do y <- readInt in (
        return x + y
    )
);
```

TODO: I'm not happy at all with this explanation yet. In the `return` clause, the program makes reference to variables x and y from the environment. Since x is defined outside of the monadic `do` clause, it is not available to clauses in the body of the clause without a means to convert the type $(\Gamma \times TA)$ to $T(\Gamma \times A)$.

Figure 2.12: Program in a monadic effectful language that requires tensor strength of the effect's monad to execute.

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 2.17.

$$(\text{Gen})\frac{\Phi, \alpha \mid \Gamma \vdash v\colon A}{\Phi \mid \Gamma \vdash \Lambda\alpha.v\colon \forall\alpha.A} \quad (\text{Spec})\frac{\Phi \mid \Gamma \vdash v\colon \forall\alpha.A \quad \Phi \vdash \epsilon}{\Phi \mid \Gamma \vdash v\ \epsilon\colon A[\epsilon/\alpha]} \tag{2.17}$$

To model these augmented type rules, we can create a category, known as a *fibre*, representing the semantics 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 fibres 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 2.13 demonstrates this construction. How the fibres are derived from objects in the base category depends on the polymorphic properties of the language being modeled. For example, in the polymorphic lambda calculus (System-F) types are impredicative. That is, types can quantify over any other types, including themselves. This means that there has to be a strong coupling between the base category, which represents type-variable environments and transformations upon them and objects in the fibres, which represent types. This typically manifests in the set of objects in each fibre being in bijection with the set of morphisms from the appropriate type-variable environment in the base category. Since, as we shall see, in effect-polymorphic languages, types quantify over effects, but effects do not quantify over themselves, we can conceptually decouple the objects in the fibres from the base category, meaning that effect-polymorphic models are simpler to define.

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

TODO: The definition of the preordered monoid in Section 3.3.2 is a bit too vague. TODO: Make it clearer that elements of E are now ranged over by e (before it was ϵ) TODO: 'G ::=' should be 'Γ ::=' TODO: The (Weaken) rule needs a side-condition saying that '$\beta \notin \Phi$' (so that '$\Phi \vdash \epsilon\colon$ `Eff`' implies Φ`Ok`). Similarly for terms. There is also a similar problem for subtyping.

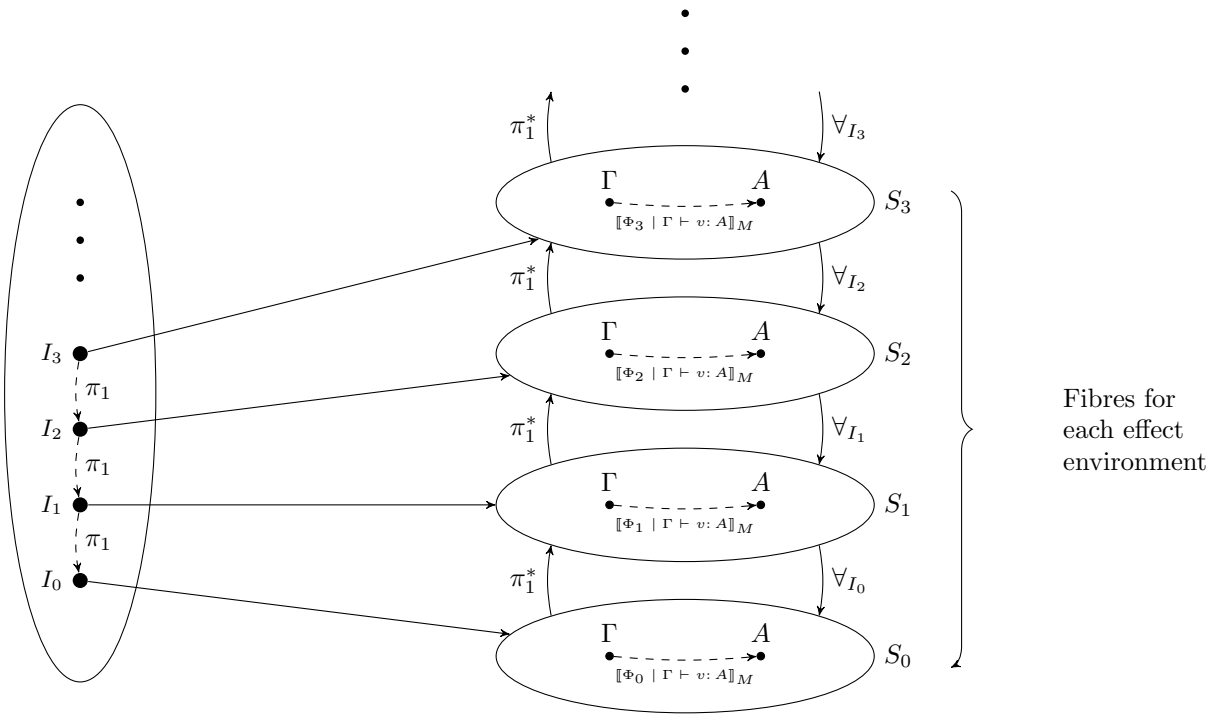


Figure 2.13: Diagram of the structure of an indexed category for modelling a polymorphic language. Solid arrows represent functors and dashed arrows represent internal morphisms. The left hand category is the base category.

2.3 The Effect Calculus

TODO: Section 3.1 you give syntax; but semantics? The lambda bits we know, but there's no explanation of bind/return. Perhaps just saying "Haskell-like semantics" and then explaining a bit? The basic effect calculus is an extension of the simply typed lambda calculus to include constants, `if` expressions, effects, and subtyping. It has terms of the following form:

$$v ::= \mathtt{C}^A \mid x \mid \mathtt{true} \mid \mathtt{false} \mid \mathtt{()} \mid \lambda x : A.v \mid v_1\ v_2 \mid \mathtt{return} v \mid \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 \mid \mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 \tag{2.18}$$

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

$$A, B, C ::= \gamma \mid A \to B \mid \mathtt{M}_\epsilon A \tag{2.19}$$

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

The calculus has a simple, Haskell style semantics. Lambda terms when applied to a parameter substitute their bound variable for the parameter expression, as seen in figure **TODO:** , if statements pick a branch once their condition is decided to be true or false, figure **TODO:** , and finally the monadic effects behave in a similar way to Haskell's monad type class. It is difficult to simply formalise an operational $\beta\eta$-reduction semantics for a monadic language without a concrete instantiation of the graded monad. Instead, the monad should obey the equational laws given in figure **TODO:** .

```
do b <- Prompt("Are You Sure?") in (
    if b then
        FireMissiles
    else
        AbortMissiles
)
```

```
λ alice: Agent. (
    λ bob: Agent. (
    do amount <- AwaitPayment(alice)
    in SendPayment(bob, amount)
    )
)
```

Figure 2.14: A pair of examples of non-polymorphic programs

```
Λ Action: Effect.
λ ifConfirmed: Action.
λ ifAborted: Action.
do b <- Prompt("Are You Sure?") in (
    if b then
        ifConfirmed
    else
        ifAborted
)
```

Figure 2.15: A polymorphic version of the checking example

As an example, by instantiating the language with the appropriate constants, ground effects, and ground types, we can write the programs in figure 2.14.

Already, we can see where having effect polymorphism in the language would be useful. We could make the first example in figure 2.14 into a more general procedure that prompts a user to confirm any I-O action, as can be seen in the extended example in figure 2.15.

2.4 Polymorphic Effect Calculus

TODO: The notion of S-category is very important. Perhaps put it in a definition environment? TODO: You might want to drop the subscript M from the denotations if M never does anything interesting (it clutters the notation a bit).

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

$$v ::= .. \mid \Lambda\alpha.v \mid v\ \epsilon \tag{2.20}$$

$$A, B, C ::= ... \mid \forall\alpha.A \tag{2.21}$$

$$\epsilon ::= e \mid \alpha \mid \epsilon \cdot \epsilon \tag{2.22}$$

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

2.4.1 Type System

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.

$$\Phi ::= \diamond \mid \Phi, \alpha \tag{2.23}$$

Effects

The ground effects form the same monotonic, 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:

$$(E_\Phi, \cdot_\Phi, \mathtt{1}, \leq_\Phi) \tag{2.24}$$

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

$$()\frac{\epsilon_3 = \epsilon_1 \cdot \epsilon_2}{\epsilon_3 = \epsilon_1 \cdot_\Phi \epsilon_2} \tag{2.25}$$

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

$$\epsilon_1 \leq_\Phi \epsilon_2 \Leftrightarrow \forall \sigma \downarrow . \epsilon_1 [\sigma \downarrow] \leq \epsilon_2 [\sigma \downarrow] \tag{2.26}$$

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 $\leq_\Phi$.

Types

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

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

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

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

Well-Formedness 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 $\texttt{Ok}$ predicate on effect environments asserts that the effect environment does not contain any duplicated effect variables.

$$(\text{Empty})\frac{}{\diamond \texttt{Ok}} \quad (\text{Const})\frac{\Phi \texttt{Ok} \;\; \alpha \notin \Phi}{\Phi, \alpha \texttt{Ok}}$$

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

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

The well-formedness 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.

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

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

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

Subtyping

TODO: "Section 3.3.6 isn't quite proper maths – it's not that there exists, we're defining or assuming that subtyping is trivial between base types (or perhaps saying that a set of base types comes along with its own set of subtype relationships (int ¡ real), but perhaps this is overkill here. Then Φ is *not* a parameter. It's more of a label. " There exists a subtyping pre-order relation $\leq:_\gamma$ over ground types. That is:

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

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

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

Type Rules

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

Inversion in Inductive Proofs In proofs involving assumptions with multiple inductive rules, a useful property is that of inversion. This allows us to case split over the inductive rule that generated the assumption and can allow us to infer the structure of the term.
For example, consider a theorem which assumes $\Phi \mid \Gamma \vdash v : A$, then we can case split over the type rules. If $\Phi \mid \Gamma \vdash v : A$ was generated by the (Apply) rule (2.27), then by inspecting the rule, we can infer that there exist two subterms $\Phi \mid \Gamma \vdash v_1 : A \to B$, $\Phi \mid \Gamma \vdash v_2 : A$ such that $v = v_1\ v_2$.

$$\text{(Apply)}\frac{\Phi \mid \Gamma \vdash v_1 : A \to B \quad \Phi \mid \Gamma \vdash v_2 : A}{\Phi \mid \Gamma \vdash v_1\ v_2 : B} \tag{2.27}$$

Then we might use the inferred subexpressions v_1, v_2 to prove the theorem in this case.

Figure 2.16: An explanation of the concept of inversion.

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

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

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

$$\text{(Return)}\frac{\Phi \mid \Gamma \vdash v : A}{\Phi \mid \Gamma \vdash \texttt{return} v : \texttt{M}_1 A} \quad \text{(Apply)}\frac{\Phi \mid \Gamma \vdash v_1 : A \to B \quad \Phi \mid \Gamma \vdash v_2 : A}{\Phi \mid \Gamma \vdash v_1\ v_2 : B}$$

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

Ok Lemma

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

Proof If $\Phi \vdash \Gamma, x : A \texttt{Ok}$ then by the inversion (See figure 2.16) of the `Ok` rule, we have $\Phi \vdash \Gamma \texttt{Ok}$

Only the type rule `Weaken` adds terms to the environment from its preconditions to its postcondition and it does so in an `Ok` preserving way. Any type derivation tree has at least one leaf. All leaves are axioms which require $\Phi \vdash \Gamma \texttt{Ok}$. And all non-axiom derivations preserve the `Ok` property.

Chapter 3

The Semantics of PEC in an Strictly Indexed Category

TODO: Section 5.1

TODO: "a mapping from objects... This functor": You don't have a functor yet! TODO: Using I as a shorthand for $[\![\Phi]\!]_M$: this is a little vague (e.g. what does I' mean?). Does this really buy anything? TODO: "essentially a list of variables": isn't Φ *literally* a list of variables? TODO: Do you require that all of the objects of C have the form U^n? (I'm not sure if you should or not.) TODO: "Monoidal operator": what does this mean?

TODO: Section 5.3

TODO: "the denotation on an effect is a morphism": give the type of the morphism. TODO: The denotation of $[\![\Phi \vdash e{:}\texttt{Effect}]\!]_M$ has a typo. TODO: Possibly make it clearer where $\epsilon_1\epsilon_2$ comes from. TODO: I suggest explaining the (Effect-App) case of the denotational semantics more, it is quite hard to understand. (I had this problem when reading Crole.)

TODO: Section 5.4

TODO: The substitution typing judgement might need a little more explanation. TODO: Lemma 5.4.1: since the proof relies on 5.4.3, it should probably come later (unless there is a good reason for mentioning it there). TODO: Theorem 5.4.6: it's really hard to parse the statement of this theorem. TODO: Theorem 5.4.9: it's not clear what Δ, Δ' are here TODO: (5.63) and (5.64): these lines don't look right. You shouldn't drop the $\alpha := \alpha$ from the substitution until you pull it outside of the $\forall$ TODO: (5.106) and (5.107) have unmatched brackets. TODO: all denotations for a typing relation are equivalent": all derivations of a typing relation have the same denotation? TODO: I'm not sure about the term S-closure. Closure is a specific term in category theory.

TODO: Section 5.5

TODO: "above instances of...": each of these typing rules has more than one assumption. Do you allow subtyping to be used on all of these assumptions? TODO: You should make it clearer that this is the *definition* of reduced typing derivation. TODO: "As an aside...": this shouldn't be inside the proof (but should still be mentioned somewhere). Put it before or after.

TODO: Section 5.6

TODO: Omitting congruence: the cases for congruence follow from the fact that the semantics is compositional. You should probably mention compositionality somewhere,

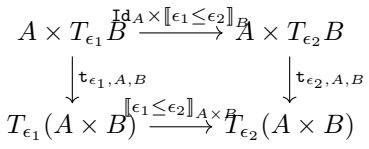


Figure 3.1: The interaction of the subeffect natural transformation with the tensor strength natural transformation.

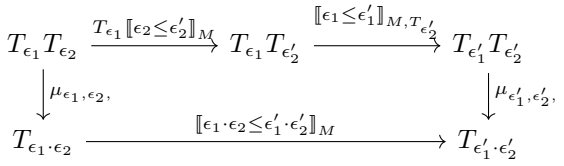


Figure 3.2: The interaction of the subeffect natural transformation with the graded monad bind natural transformation.

because it's important.

In this chapter, I describe the category structure required to interpret an instance of the PEC. I then present denotations of each type of structure in the language, such as types, effects, terms, substitutions, and environment weakenings. Finally, I provide outlines and interesting cases of the proofs of the lemmas leading up to and including soundness of $\beta\eta$-conversion. **TODO: Or equational equivalence** However, firstly I shall give a brief treatment to the semantics of EC.

3.1 Semantics for EC in an S-Category

As suggested in section 2.2, since EC contains multiple effects, STLC terms, and `if` expressions, we should be able interpret its semantics in a cartesian closed category with an appropriate strong graded monad. With the addition of some extra category structure to handle subtyping, which I shall explain shortly, it is indeed possible to interpret EC. This section contains a fairly high-level treatment of the semantics. This is because the concepts introduced are a subset of those required for the semantics of PEC, which I shall explain in more detail later.

TODO: Denotational semantic for similar languages have been presented before - references?

In order to correctly model a given instantiation of EC, that is an EC with a collection of effects, constants, and ground types, there should be a collection of objects in the category to represent the ground types. There should also be a point morphism (a morphism from the terminal object to the appropriate type) for each constant.

In addition to the previously stated requirements, we also require the category, $\mathbb{C}$, to be able to model subtyping and subeffecting. For each instance of the ground subtyping relation, $A \leq:_\gamma B$, there should exist a morphism between the objects representing the ground types A and B. A further requirement is that $\mathbb{C}$ has a collection of subeffecting 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 3.1, 3.2. We can now define morphisms for all forms of subtyping by constructions following the derivation tree for subtyping. For more detail, one can look at section 3.4 and below for the same construction on PEC. From this point onwards, I shall refer to a category that fulfills these properties of having a strong graded monad, CCC, ground objects and points, subeffect natural transformations, and a co-product as an *S-Category* (Semantic Category).

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 are repeated later in the remainder of this dissertation anyway. The categorical semantics of the Effect Calculus requires an S-category not only to simply model all of the features of EC but also to use the various commutivity diagrams and rules to manipulate the expressions encountered when proving properties of the semantics.

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

3.2 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 is denoted from this point onwards as $\mathbb{C}(-)$ and the induced categories $\mathbb{C}(\llbracket\Phi\rrbracket_M)$ are called "fibres". A further notational shorthand I use, inheriting from R. L. Crole **TODO: Reference**, is to write I for the denotation of Φ. The term "S-closed" indicates that all functors derived from $\mathbb{C}$ within this category preserve the properties of S-categories, which are explained in section 3.1. A functor F preserves the properties of S-categories if it preserves each of the features within an S-category. For example $F(X \times Y) = (FX) \times (FY)$. For a full list of properties that an S-closed functor should preserve, please see appendix **TODO: Link/reference**. Thus, each morphism $\theta : \llbracket\Phi'\rrbracket_M \to \llbracket\Phi\rrbracket_M$ in $\mathbb{C}$ should induce as S-closed, re-indexing functor $\theta^* : \mathbb{C}(I) \to \mathbb{C}(I')$ between the fibres.

The essential idea from this point on is that we have defined several relations of the form $\texttt{Env} \vdash \texttt{Conclusion}$, such as the typing relation $\Phi \mid \Gamma \vdash v\colon A$, and the well-formedness relation on effects $\Phi \vdash \epsilon$. Each instance of such a relation has a denotation that is an object or morphism in a category. For example, $\llbracket\Phi \vdash \epsilon\colon \texttt{Effect}\rrbracket_M$ is a morphism in the base category, $\llbracket\Phi \vdash A\colon \texttt{Type}\rrbracket_M$ is an object in the fibre (S-category) induced by Φ, and $\llbracket\Phi \mid \Gamma \vdash v\colon A\rrbracket_M$ 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 effect 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 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 $\llbracket e\rrbracket_M : \mathbb{C}(\mathtt{1}, U)$ for each ground effect e. Furthermore, each fibre should contain and object $\llbracket\gamma\rrbracket_M$ for each ground type γ. Finally, for each constant, $\mathtt{C}^A$, there should exist a morphism in each fibre: $\llbracket\mathtt{C}^A\rrbracket_M : \mathtt{1} \to A$. 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, $\texttt{Mul}(\llbracket e_1\rrbracket_M, \llbracket e_2\rrbracket_M) = \llbracket e_1 \cdot e_2\rrbracket_M$ where e_1, e_2 are ground effects.

Our penultimate requirement is that the re-indexing functor induced by $\pi_1 : I \times U \to I$ (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. This right adjoint need not be S-closed.

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

$$\overline{(\theta \times \texttt{Id}_U)^*(\boldsymbol{\epsilon})} = \texttt{Id} : \theta^* \circ \forall_I \to \forall_{I'} \circ (\theta \times \texttt{Id}_U)^* \in \mathbb{C}(I') \tag{3.1}$$

From the Beck-Chevalley condition, we can derive some more naturality conditions that will become useful later. Using the adjunction property, we have that: $\overline{f \circ \pi_1^*(n)} = \overline{f} \circ n$. Secondly, using the Beck-Chevalley condition, we can derive some non-trivial interactions between re-indexing functors and the

adjunction[1].

$$\theta^*\boldsymbol{\eta}_A : \quad \theta^* A \to \theta^* \circ \forall_I \circ \pi_1^* A \tag{3.2}$$

$$\theta^*\boldsymbol{\eta} = \overline{(\theta \times \mathtt{Id}_U)^*(\boldsymbol{\epsilon}_{\pi_1^*})} \circ \theta^*\boldsymbol{\eta} \tag{3.3}$$

$$= (\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^*)(\boldsymbol{\epsilon}_{\pi_1^*}) \circ \boldsymbol{\eta}_{(\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^*) \circ \pi_1^*} \circ \theta^*\boldsymbol{\eta} \tag{3.4}$$

$$= (\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^*)(\boldsymbol{\epsilon}_{\pi_1^*}) \circ \boldsymbol{\eta}_{\theta^* \circ \forall_I \circ \pi_1^*} \circ \theta^*\boldsymbol{\eta} \tag{3.5}$$

$$= (\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^*)(\boldsymbol{\epsilon}_{\pi_1^*}) \circ (\theta^* \circ \forall_I \circ \pi_1^*)\boldsymbol{\eta} \circ \boldsymbol{\eta}_{(\theta \times \mathtt{Id}_U)^*} \tag{3.6}$$

$$= (\theta^* \circ \forall_I)(\boldsymbol{\epsilon}_{\pi_1^*} \circ \pi_1^*\boldsymbol{\eta}) \circ \boldsymbol{\eta}_{(\theta \times \mathtt{Id}_U)^*} \tag{3.7}$$

$$= (\theta^* \circ \forall_I)(\mathtt{Id}) \circ \boldsymbol{\eta}_{(\theta \times \mathtt{Id}_U)^*} \tag{3.8}$$

$$= \boldsymbol{\eta}_{(\theta \times \mathtt{Id}_U)^*} \tag{3.9}$$

Importantly, this gives us the useful naturality condition that allows us to push re-indexing functors into the adjunction terms.

$$\theta^*(\overline{f}) = \theta^*(\forall_I(f) \circ \eta_A) \tag{3.10}$$

$$= \theta^*(\forall_I(f)) \circ \theta^*(\eta_A) \tag{3.11}$$

$$= (\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^*) f \circ \boldsymbol{\eta}_{(\theta \times \mathtt{Id}_U)^* A} \tag{3.12}$$

$$= \overline{(\theta \times \mathtt{Id}_U)^* f} \tag{3.13}$$

$$\tag{3.14}$$

3.3 Road Map

In figure 3.3, 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 made up of the effect substitution and weakening theorem on effects. These theorems show that substitutions of effects have a 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 subtyping 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, $\Phi \mid \Gamma \vdash v : A$ have the same denotation. This is important, since subtyping 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 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.

[1] This part of the proof comes from a stack-overflow answer: https://math.stackexchange.com/questions/188099/beck-chevalley-condition-and-maps-of-adjunctions

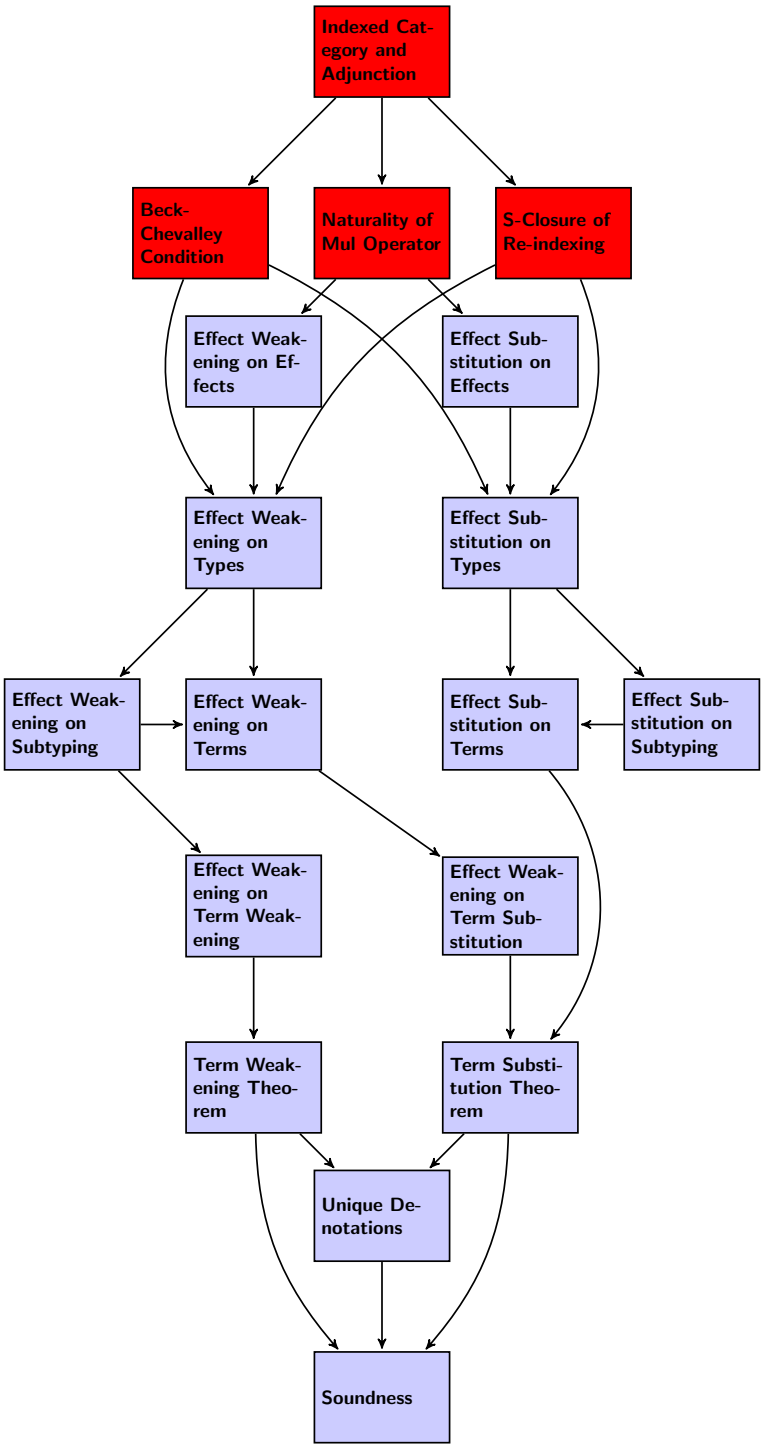


Figure 3.3: A road map of the proof dependencies. Assumptions in red, theorems in blue

3.4 Denotations

We are now equipped to define the denotations of structures in the language. Firstly, we shall define the denotation of the well-formedness relation on effects. As stated earlier, the denotation of an effect is a

morphism $[\![\Phi \vdash \epsilon\colon \texttt{Effect}]\!]_M$ in $\mathbb{C}$.

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

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.

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 $\mathtt{1} + \mathtt{1}$.

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

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

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

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, $\mathbb{C}(I)$.

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

Another construction that is important is the denotation of subtyping. For each instance of the subtyping 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 subtyping relation exist in each fibre anyway.

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

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

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, $\mathtt{C}^A$, there exists $c : \mathtt{1} \to A_I$ in $\mathbb{C}(I)$.

TODO: Make these more readable/fix spacing

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

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

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

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

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

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

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

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

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

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

3.5 Substitution and Weakening Theorems

In this section, I 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 $\Phi \mid \Gamma \vdash v\colon A$, then it should be the case that $\Phi \mid \Gamma, x : A \vdash v\colon A$. We also want to know what happens to the denotation when we change the type environment in this way. In this section, I 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 subtyping 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.

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-formedness 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:

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

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:

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

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

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

We can also define the action of substitutions on effects.

$$\sigma(e) = e \tag{3.15}$$
$$\sigma(\epsilon_1 \cdot \epsilon_2) = (\sigma(\epsilon_1)) \cdot (\sigma(\epsilon_2)) \tag{3.16}$$
$$\diamond(\alpha) = \alpha \tag{3.17}$$
$$(\sigma, \beta := \epsilon)(\alpha) = \sigma(\alpha) \tag{3.18}$$
$$(\sigma, \alpha := \epsilon)(\alpha) = \epsilon \tag{3.19}$$

.

This definition allows us to define, in particular, the identity substitution,$\Phi \vdash \texttt{Id}_\Phi : \Phi$ and the single substitution, $\Phi \vdash [\epsilon/\alpha] : \Phi, \alpha$.

$$\texttt{Id}_\diamond = \diamond \tag{3.20}$$
$$\texttt{Id}_{\Phi,\alpha} = (\texttt{Id}_\Phi, \alpha := \alpha) \tag{3.21}$$
$$[\epsilon/\alpha] = (\texttt{Id}_\Phi, \alpha := \epsilon) \tag{3.22}$$

By inspection, we can find the denotations of these special-case substitutions to be:

$$\llbracket \Phi \vdash \mathtt{Id}_\Phi : \Phi \rrbracket_M = \mathtt{Id}_I \tag{3.23}$$
$$\llbracket \Phi \vdash [\epsilon/\alpha] : (\Phi, \alpha) \rrbracket_M = \langle \mathtt{Id}_I, \llbracket \Phi \vdash \epsilon : \mathtt{Effect} \rrbracket_M \rangle \tag{3.24}$$

These substitutions will form a useful shorthand later. We also need to define the property $\alpha \# \sigma$. This indicates that α does not occur in the domain or any of the substitution expression of σ. Finally, it will be useful to examine how the substitutions generated by extending the effect environment, such as in the case of polymorphic types, relate to the original substitution.

Lemma 3.5.1 (Extension Lemma on Effect Substitutions) *If $\Phi' \vdash \sigma : \Phi$, and $\alpha \# \sigma$, then $(\Phi', \alpha) \vdash (\sigma, \alpha := \alpha) : (\Phi, \alpha)$ with denotation $\llbracket (\Phi', \alpha) \vdash (\sigma, \alpha := \alpha) : (\Phi, \alpha) \rrbracket_M = (\llbracket \Phi' \vdash \sigma : \Phi \rrbracket_M \times \mathit{Id}_U)$*

Proof: This holds by the weakening-theorem on effects, theorem 3.5.3

□

Theorem 3.5.2 (Effect Substitution on Effects) *The substitution theorem on effects is the proposition that if $\Phi \vdash \epsilon$ and $\Phi' \vdash \sigma : \Phi$ then $\Phi' \vdash \sigma(\epsilon)$ and, writing σ for $\llbracket \Phi' \vdash \sigma : \Phi \rrbracket_M$, $\llbracket \Phi' \vdash \sigma(\epsilon) \rrbracket_M = \llbracket \Phi \vdash \sigma \rrbracket_M \circ \sigma$.*

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.

Case Ground: This case holds due to the terminal morphism being unique. Hence, pre-composing it with another morphism yields the terminal morphism.

$$\begin{aligned}
\llbracket \Phi \vdash e : \mathtt{Effect} \rrbracket_M \circ \sigma &= \llbracket e \rrbracket_M \circ \langle \rangle_I \circ \sigma && (3.25)\\
&= \llbracket e \rrbracket_M \circ \langle \rangle_{I'} && (3.26)\\
&= \llbracket \Phi' \vdash e : \mathtt{Type} \rrbracket_M && (3.27)\\
& && (3.28)
\end{aligned}$$

Case Var: Since the structure of the substitution σ depends on the structure of Φ, we can perform inversion to infer the denotation of σ.

$$\begin{aligned}
\llbracket \Phi, \alpha \vdash \alpha : \mathtt{Effect} \rrbracket_M \circ \sigma &= \pi_2 \circ \langle \sigma', \llbracket \Phi' \vdash \epsilon : \mathtt{Effect} \rrbracket_M \rangle \quad \text{By inversion } \sigma = (\sigma', \alpha := \epsilon) && (3.29)\\
&= \llbracket \Phi' \vdash \epsilon : \mathtt{Effect} \rrbracket_M && (3.30)\\
&= \llbracket \Phi' \vdash \sigma'(\alpha) : \mathtt{Effect} \rrbracket_M && (3.31)\\
& && (3.32)
\end{aligned}$$

Case Multiply: We make use of the naturality of `Mul` and induction upon the subterms.

$$\begin{aligned}
\llbracket \Phi \vdash \epsilon_1 \cdot \epsilon_2 : \texttt{Effect} \rrbracket_M \circ \sigma &= \texttt{Mul}(\llbracket \Phi \vdash \epsilon_1 : \texttt{Effect} \rrbracket_M, \llbracket \Phi \vdash \epsilon_2 : \texttt{Effect} \rrbracket_M) \circ \sigma && (3.33)\\
&= \texttt{Mul}(\llbracket \Phi \vdash \epsilon_1 : \texttt{Effect} \rrbracket_M \circ \sigma, \llbracket \Phi \vdash \epsilon_2 : \texttt{Effect} \rrbracket_M \circ \sigma) \quad \text{By Naturality} && (3.34)\\
&= \texttt{Mul}(\llbracket \Phi' \vdash \sigma(\epsilon_1) : \texttt{Effect} \rrbracket_M, \llbracket \Phi \vdash \sigma(\epsilon_2) : \texttt{Effect} \rrbracket_M) && (3.35)\\
&= \llbracket \Phi' \vdash \sigma(\epsilon_1) \cdot \sigma(\epsilon_2) : \texttt{Effect} \rrbracket_M && (3.36)\\
&= \llbracket \Phi' \vdash \sigma(\epsilon_1 \cdot \epsilon_2) : \texttt{Effect} \rrbracket_M && (3.37)\\
& && (3.38)
\end{aligned}$$

□

Theorem 3.5.3 (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* $\llbracket \omega : \Phi' \triangleright \Phi \rrbracket_M$, $\llbracket \Phi' \vdash \epsilon \rrbracket_M = \llbracket \Phi \vdash \sigma \rrbracket_M \circ \omega$.

Proof: This proof also depends on the naturality of `Mul` and case splitting on the structure of ω. It is harder to use inversion on the structure of ω, since the structure of ω does not depend as strongly on the structure of Φ. I present here the cases for variables.

Case Var: We do a case split on ω.

Case: $\omega = \iota$ Then $\Phi' = \Phi$ and $\omega = \texttt{Id}_I$. So the theorem holds trivially.

Case: $\omega = \omega' \times$ Then by the definition of its denotation:

$$\begin{aligned}
\llbracket \Phi, \alpha \vdash \alpha : \texttt{Effect} \rrbracket_M \circ \omega &= \pi_2 \circ (\omega' \times \texttt{Id}_U) && (3.39)\\
&= \pi_2 && (3.40)\\
&= \llbracket \Phi', \alpha \vdash \alpha : \texttt{Effect} \rrbracket_M && (3.41)
\end{aligned}$$

Case: $\omega = \omega' \pi$ Then

$$\llbracket \Phi, \alpha \vdash \alpha : \texttt{Effect} \rrbracket_M = \pi_2 \circ \omega' \circ \pi_1 \tag{3.42}$$

Where $\Phi' = \Phi, \beta$ and $\omega' : \Phi'' \triangleright \Phi$.

So

$$\begin{aligned}
\pi_2 \circ \omega' &= \llbracket \Phi'' \vdash \alpha : \texttt{Effect} \rrbracket_M && (3.43)\\
\pi_2 \circ \omega' \circ \pi_1 &= \llbracket \Phi'', \beta \vdash \alpha : \texttt{Effect} \rrbracket_M && (3.44)\\
&= \llbracket \Phi' \vdash \alpha : \texttt{Effect} \rrbracket_M && (3.45)
\end{aligned}$$

Case Weaken:

$$\llbracket \Phi, \beta \vdash \alpha : \texttt{Effect} \rrbracket_M \circ \omega = \llbracket \Phi \vdash \alpha : \texttt{Effect} \rrbracket_M \circ \pi_1 \circ \omega \tag{3.46}$$

Similarly, we perform a case split on structure of w:

Case: $\omega = \iota$ Then $\Phi' = \Phi, \beta$ so $\omega = \texttt{Id}_I$ So $\llbracket \Phi, \beta \vdash \alpha : \texttt{Effect} \rrbracket_M \circ \omega = \llbracket \Phi' \vdash \alpha : \texttt{Effect} \rrbracket_M$

Case: $\omega = \omega'\pi_1$ Then $\Phi' = (\Phi'', \gamma)$ and $\omega = \omega' \circ \pi_1$ Where $\omega' : \Phi'' \triangleright \Phi, \beta$. So

$$\begin{aligned}
[\![\Phi, \beta \vdash \alpha{:}\,\mathtt{Effect}]\!]_M \circ \omega &= [\![\Phi, \beta \vdash \alpha{:}\,\mathtt{Effect}]\!]_M \circ \omega' \circ \pi_1 && (3.47)\\
&= \Phi'' \vdash \alpha{:}\,\mathtt{Effect} \circ \pi_1 && (3.48)\\
&= \Phi'', \gamma \vdash \alpha{:}\,\mathtt{Effect} && (3.49)\\
&= \Phi' \vdash \alpha{:}\,\mathtt{Effect} && (3.50)\\
& && (3.51)
\end{aligned}$$

Case: $\omega = \omega'\times$ Then $\Phi' = \Phi'', \beta$ and $\omega' : \Phi'' \triangleright \Phi$

So

$$\begin{aligned}
[\![\Phi, \beta \vdash \alpha{:}\,\mathtt{Effect}]\!]_M \circ \omega &= [\![\Phi \vdash \alpha{:}\,\mathtt{Effect}]\!]_M \circ \pi_1 \circ (\omega' \times \mathtt{Id}_U) && (3.52)\\
&= [\![\Phi \vdash \alpha{:}\,\mathtt{Effect}]\!]_M \circ \omega' \circ \pi_1 && (3.53)\\
&= [\![\Phi'' \vdash \alpha{:}\,\mathtt{Effect}]\!]_M \circ \pi_1 && (3.54)\\
&= [\![\Phi' \vdash \alpha{:}\,\mathtt{Effect}]\!]_M && (3.55)\\
& && (3.56)
\end{aligned}$$

□

We can then move on to state and prove the weakening and substitution theorems on types, subtyping, and type environments. The general structure of these theorems, as well as the term-based theorems 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 3.5.4 (Effect Substitution on Types) *The specific effect substitution theorem on types is if* $\Phi \vdash A$ *and* $\Phi' \vdash \sigma{:}\,\Phi$*, 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.

Case Monad: This case makes use of the S-closure of σ^*. Specifically the S-closure property upon the graded monad functor T: $\sigma^*(T_f A) = T_{f \circ \sigma}(\sigma^* A)$.

$$\begin{aligned}
\sigma^* [\![\Phi \vdash \mathtt{M}_\epsilon A \colon \mathtt{Type}]\!]_M &= \sigma^* (T_{[\![\Phi \vdash \epsilon : \mathtt{Effect}]\!]_M} [\![\Phi \vdash A \colon \mathtt{Type}]\!]_M) && (3.57)\\
&= T_{[\![\Phi \vdash \epsilon : \mathtt{Effect}]\!]_M \circ \sigma} \sigma^* ([\![\Phi \vdash A \colon \mathtt{Type}]\!]_M) && (3.58)\\
&= [\![\Phi' \vdash (\mathtt{M}_\epsilon A)\,[\sigma] : \mathtt{Type}]\!]_M && (3.59)
\end{aligned}$$

Case Quantification: This case makes use of the Beck-Chevalley condition and the fact that $[\![(\Phi', \alpha) \vdash (\sigma, \alpha := \alpha) \colon (\Phi, \alpha)]\!]_M = \sigma \times \mathtt{Id}_U$, which we can induct upon.

$$\begin{aligned}
\sigma^* [\![\Phi \vdash \forall \alpha . A \colon \mathtt{Type}]\!]_M &= \sigma^* (\forall_I ([\![\Phi, \alpha \vdash A \colon \mathtt{Type}]\!]_M)) && (3.60)\\
&= \forall_I ((\sigma \times \mathtt{Id}_U)^* [\![\Phi, \alpha \vdash A \colon \mathtt{Type}]\!]_M) \quad \text{By Beck-Chevalley} && (3.61)\\
&= \forall_I ([\![\Phi', \alpha \vdash A\,[\sigma, \alpha := \alpha] : \mathtt{Type}]\!]_M) \quad \text{By induction} && (3.62)\\
&= \forall_I ([\![\Phi', \alpha \vdash A\,[\sigma] : \mathtt{Type}]\!]_M) && (3.63)\\
&= [\![\Phi' \vdash \forall \alpha . A\,[\sigma] : \mathtt{Type}]\!]_M && (3.64)\\
&= [\![\Phi' \vdash (\forall \alpha . A)\,[\sigma] : \mathtt{Type}]\!]_M && (3.65)\\
& && (3.66)
\end{aligned}$$

□

Theorem 3.5.5 (Effect Substitution on Type Environments) *Similarly, the effect-substitution theorem on type environments is that if* $\Phi \vdash \Gamma\, \mathtt{Ok}$*, then* $\Phi' \vdash \Gamma\,[\sigma]\, \mathtt{Ok}$ *and* $[\![\Phi' \vdash \Gamma\,[\sigma]\, \mathtt{Ok}]\!]_M = \sigma^* [\![\Phi \vdash \Gamma\, \mathtt{Ok}]\!]_M$.

Proof: By induction on the derivation on $[\![\Phi \vdash \Gamma \mathtt{Ok}]\!]_M$ whilst making use of S-closure of the re-indexing functor.

Case Nil: We make use of the fact that S-closure means that the terminal object is preserved.

$$\begin{aligned}
\sigma^* [\![\Phi \vdash \diamond \mathtt{Ok}]\!]_M &= \sigma^* \mathtt{1} && (3.67)\\
&= \mathtt{1} \quad \text{By S-closure} && (3.68)\\
&= [\![\Phi' \vdash \diamond \mathtt{Ok}]\!]_M && (3.69)\\
&= [\![\Phi' \vdash \diamond\,[\sigma]\, \mathtt{Ok}]\!]_M && (3.70)\\
& && (3.71)
\end{aligned}$$

Case Var: S-closure means that $\sigma^*(A \times B) = (\sigma^* A) \times (\sigma^* B)$.

$$\begin{aligned}
\sigma^* [\![\Phi \vdash \Gamma, x : A \mathtt{Ok}]\!]_M &= \sigma^* ([\![\Phi \vdash \Gamma \mathtt{Ok}]\!]_M \times [\![\Phi \vdash A \colon \mathtt{Type}]\!]_M) && (3.72)\\
&= (\sigma^* [\![\Phi \vdash \Gamma \mathtt{Ok}]\!]_M \times \sigma^* [\![\Phi \vdash A \colon \mathtt{Type}]\!]_M) && (3.73)\\
&= ([\![\Phi' \vdash \Gamma\,[\sigma]\, \mathtt{Ok}]\!]_M \times [\![\Phi' \vdash A\,[\sigma] : \mathtt{Type}]\!]_M) && (3.74)\\
&= [\![\Phi' \vdash \Gamma\,[\sigma], x : A\,[\sigma]\, \mathtt{Ok}]\!]_M && (3.75)\\
&= [\![\Phi' \vdash (\Gamma, x : A)\,[\sigma]\, \mathtt{Ok}]\!]_M && (3.76)\\
& && (3.77)
\end{aligned}$$

□

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

Theorem 3.5.6 (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: In a similar fashion, we make use of the Beck-Chevalley condition and the S-closure of σ^*.

Case Quantification: This case uses the Beck-Chevalley condition and the fact that $[\![\omega \times : (\Phi', \alpha) \triangleright (\Phi, \alpha)]\!]_M = w \times \mathtt{Id}_U$. This property is used in conjunction with induction to change the environment from (Φ, α) to (Φ', α).

$$\begin{aligned}
\omega^* [\![\Phi \vdash \forall \alpha.A\colon \mathtt{Type}]\!]_M &= \omega^* (\forall_I ([\![\Phi, \alpha \vdash A\colon \mathtt{Type}]\!]_M)) && (3.78)\\
&= \forall_I ((\omega \times \mathtt{Id}_U)^* [\![\Phi, \alpha \vdash A\colon \mathtt{Type}]\!]_M) \quad \text{By Beck-Chevalley} && (3.79)\\
&= \forall_I ([\![\Phi', \alpha \vdash A\colon \mathtt{Type}]\!]_M) \quad \text{By induction} && (3.80)\\
&= \forall_I ([\![\Phi', \alpha \vdash A\colon \mathtt{Type}]\!]_M) && (3.81)\\
&= [\![\Phi' \vdash \forall \alpha.A\colon \mathtt{Type}]\!]_M && (3.82)\\
& && (3.83)
\end{aligned}$$

Case Function: This case makes use of the S-closure property that ω^* preserves exponentials. Specifically $\omega^*(B^A) = (\omega^* B)^{(\omega^* A)}$.

$$\begin{aligned}
\omega^* [\![\Phi \vdash A \to B\colon \mathtt{Type}]\!]_M &= \omega^* ([\![\Phi \vdash B\colon \mathtt{Type}]\!]_M^{[\![\Phi \vdash A:\mathtt{Type}]\!]_M}) && (3.84)\\
&= \omega^* ([\![\Phi \vdash B\colon \mathtt{Type}]\!]_M)^{\omega^* ([\![\Phi \vdash A:\mathtt{Type}]\!]_M)} && (3.85)\\
&= [\![\Phi' \vdash B\colon \mathtt{Type}]\!]_M^{[\![\Phi' \vdash A:\mathtt{Type}]\!]_M} && (3.86)\\
&= [\![\Phi' \vdash A \to B\colon \mathtt{Type}]\!]_M && (3.87)\\
& && (3.88)
\end{aligned}$$

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

□

Next we consider the action of weakening and substitution on subtyping relations.

Theorem 3.5.7 (Effect Substitution on Subtyping) *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.

Case Ground: This case holds by the property of the S-closure of σ^* that σ^* preserves the ground-type subtyping denotation.

$$\sigma^* (\gamma_1 \leq:_\gamma \gamma_2) = (\gamma_1 \leq:_\gamma \gamma_2) \tag{3.89}$$

Case Fn: This case holds by several S-closure properties of σ^*. σ^* preserves the exponential morphisms: $\sigma^*(\mathtt{cur}(f)) = \mathtt{cur}(\sigma^*(f))$ and $\sigma^*(\mathtt{app}) = \mathtt{app}$. σ^* should also preserve the product morphisms: $\sigma^*(\pi_1) = \pi_1$, $\sigma^*(\pi_2) = \pi_2$ and $\sigma^* \langle f, g \rangle = \langle \sigma^* f, \sigma^* g \rangle$. Hence $\sigma^*(f \times g) = (\sigma^* f \times \sigma^* g)$.

$$\begin{aligned}
\sigma^* [\![(A \to B) \leq_{:\Phi} A' \to B']\!]_M &= \sigma^*([\![B \leq_{:\Phi} B']\!]_M^{A'} \circ B^{[\![A' \leq_{:\Phi} A]\!]_M}) && (3.90)\\
&= \sigma^*(\mathtt{cur}([\![B \leq_{:\Phi} B']\!]_M \circ \mathtt{app})) \circ \sigma^*(\mathtt{cur}(\mathtt{app} \circ (\mathtt{Id}_B \times [\![A' \leq_{:\Phi} A]\!]_M))) && (3.91)\\
&= \mathtt{cur}(\sigma^*([\![B \leq_{:\Phi} B']\!]_M) \circ \mathtt{app}) \circ \mathtt{cur}(\mathtt{app} \circ (\mathtt{Id}_B \times \sigma^*([\![A' \leq_{:\Phi} A]\!]_M))) && (3.92)\\
&= \mathtt{cur}([\![B\,[\sigma] \leq_{:\Phi'} B'\,[\sigma]\,]\!]_M \circ \mathtt{app}) \circ \mathtt{cur}(\mathtt{app} \circ (\mathtt{Id}_{B[\sigma]} \times [\![A'\,[\sigma] \leq_{:\Phi'} A\,[\sigma]\,]\!]_M)) && (3.93)\\
&= [\![(A\,[\sigma]) \to (B\,[\sigma]) \leq_{:\Phi'} (A'\,[\sigma]) \to (B'\,[\sigma])]\!]_M && (3.94)\\
&= [\![(A \to B)\,[\sigma] \leq_{:\Phi'} (A' \to B')\,[\sigma]\,]\!]_M && (3.95)
\end{aligned}$$

□

Similarly we can form the symmetrical weakening theorem.

Theorem 3.5.8 (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 at 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 3.5.9 (Effect Substitution on Terms) *If $\Phi' \vdash \sigma : \Phi$ and Δ derives $\Phi \mid \Gamma \vdash v : A$ then there exists Δ' deriving $\Phi' \mid \Gamma\,[\sigma] \vdash v\,[\sigma] : A\,[\sigma]$ and $\Delta' = \sigma^*(\Delta)$.*

Proof: This proof makes use of the previous effect-substitution theorems, and the adjunction of quantification and the re-indexing functor of projection.

TODO: Missing the proof of existence of derivation

Case Effect-Lambda: This case makes use of the naturality condition 3.10 and simple reductions.

If Δ derives $\Phi \mid \Gamma \vdash \Lambda\alpha.v\colon \forall\alpha.A$ then by inversion, there exists Δ_1 deriving $\Phi, \alpha \mid \Gamma \vdash v\colon A$ such that:

$$\Delta = \overline{\Delta_1} \tag{3.96}$$

It is also the case that:

$$\sigma \times \mathtt{Id} = [\![(\Phi', \alpha) \vdash (\sigma, \alpha := \epsilon)\colon (\Phi, \alpha)]\!]_M \tag{3.97}$$

So

$$\begin{aligned}
\sigma^*\Delta &= \sigma^*(\overline{\Delta_1}) && (3.98)\\
&= \overline{(\sigma \times \mathtt{Id}_U)^*\Delta_1} \quad \text{By naturality} && (3.99)\\
&= \overline{\Delta_1'} \quad \text{By induction} && (3.100)\\
&= \Delta' && (3.101)
\end{aligned}$$

Case Effect-Application: This is a more complex case, as it makes use of several naturality properties and the adjunction $\pi_1^* \dashv \forall_I$.

By inversion, if Δ derives $\Phi \mid \Gamma \vdash v\ \epsilon\colon A\,[\epsilon/\alpha]$ then there exists Δ_1 deriving $\Phi \mid \Gamma \vdash v\colon \forall\alpha.A$ and $h = [\![\Phi \vdash \epsilon\colon \mathtt{Effect}]\!]_M$ such that:

$$\Delta = \langle \mathtt{Id}_\Gamma, h\rangle^* \left(\epsilon_{[\![\Phi, \beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M}\right) \circ \Delta_1 \tag{3.102}$$

So, due to the substitution theorem on effects,

$$h \circ \sigma = [\![\Phi \vdash \epsilon\colon \mathtt{Effect}]\!]_M \circ \sigma = [\![\Phi' \vdash \sigma(\epsilon)\colon \mathtt{Effect}]\!]_M = h' \tag{3.103}$$

Hence, by applying the re-indexing functor to Δ, we have:

$$\begin{aligned}
\sigma^*\Delta &= \sigma^*(\langle \mathtt{Id}_\Gamma, h\rangle^* \left(\epsilon_{[\![\Phi, \beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M}\right) \circ \Delta_1) && (3.104)\\
&= (\langle \mathtt{Id}_\Gamma, h\rangle \circ \sigma)^*(\epsilon_{[\![\Phi, \beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M}) \circ \sigma^*(\Delta_1) && (3.105)\\
&= ((\sigma \times \mathtt{Id}_U) \circ \langle \mathtt{Id}_\Gamma, h \circ \sigma\rangle)^*(\epsilon_{[\![\Phi, \beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M}) \circ \Delta_1') && (3.106)\\
&= (\langle \mathtt{Id}_\Gamma, h'\rangle)^*((\sigma \times \mathtt{Id}_U)^*\epsilon_{[\![\Phi, \beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M}) \circ \Delta_1') && (3.107)\\
& && (3.108)
\end{aligned}$$

Looking at the inner part of the functor application: Let

$$\begin{aligned}
A = [\![\Phi, \beta \vdash A\,[\beta/\alpha] : \mathtt{Type}]\!]_M && (3.109)\\
&& (3.110)
\end{aligned}$$

$$
\begin{aligned}
(\sigma \times \mathtt{Id}_U)^* \epsilon_{[\![\Phi, \beta \vdash A[\beta/\alpha]:\mathtt{Type}]\!]_M} &= (\sigma \times \mathtt{Id}_U)^* \epsilon_A && (3.111)\\
&= (\sigma \times \mathtt{Id}_U)^* (\widehat{\mathtt{Id}_{\forall_I(A)}}) && (3.112)\\
&= \widehat{\widehat{(\sigma \times \mathtt{Id}_U)^* (\widehat{\mathtt{Id}_{\forall_I(A)}})}} \quad \text{By bijection} && (3.113)\\
&= \widehat{\sigma^* (\widehat{\mathtt{Id}_{\forall_I(A)}})} \quad \text{By naturality} && (3.114)\\
&= \widehat{\sigma^* (\mathtt{Id}_{\forall_I(A)})} \quad \text{By bijection} && (3.115)\\
&= \widehat{\mathtt{Id}_{\forall_{I'}(A \circ (\sigma \times \mathtt{Id}_U))}} \quad \text{By S-Closure, naturality} && (3.116)\\
&= \widehat{\mathtt{Id}_{\forall_{I'}(A[\sigma, \alpha:=\alpha])}} \quad \text{By Substitution theorem} && (3.117)\\
&= \epsilon_{A[\sigma]} && (3.118)
\end{aligned}
$$

Going back to the original expression:

$$
\begin{aligned}
\sigma^* \Delta &= (\langle \mathtt{Id}_\Gamma, h' \rangle)^* (\epsilon_{A[\sigma]}) \circ \Delta_1') && (3.119)\\
&= \Delta' && (3.120)\\
& && (3.121)
\end{aligned}
$$

□

Similarly, we can derive the weakening theorem on terms.

Theorem 3.5.10 (Effect Weakening on Terms) *If $\omega : \Phi' \triangleright \Phi$ and Δ derives $\Phi \mid \Gamma \vdash v\colon A$ then there exists Δ' deriving $\Phi' \mid \Gamma \vdash v\colon A$ and $\Delta' = \omega^* \Delta$.*

Proof: **TODO: Prove typing** This theorem is proved in a similar fashion to the substitution theorem, and many of its cases are the same.

Case Subtype: If Δ derives $\Phi \mid \Gamma \vdash v\colon B$ them by inversion, there exists Δ_1 deriving $\Phi \mid \Gamma \vdash v\colon A$, such that $\Delta = [\![A \leq:_\Phi B]\!]_M \circ \Delta_1$

So, using the weakening of the subtyping morphism, and induction, we can derive Δ'.

$$
\begin{aligned}
\omega^*(\Delta) &= \omega^* [\![A \leq:_\Phi B]\!]_M \circ \omega^* \Delta_1 && (3.122)\\
&= [\![A_{I'} \leq:_{\Phi'} B_{I'}]\!]_M \circ \Delta_1' \quad \text{By induction} && (3.123)\\
&= D' && (3.124)
\end{aligned}
$$

Case Lambda: This case holds by induction and the S-closure of ω^*.

If Δ derives $\Phi \mid \Gamma \vdash \lambda x : A.v\colon A \to B$ then by inversion there exists Δ_1 deriving $\Phi \mid \Gamma, x : A \vdash v\colon B$ such that:

$$
\Delta = \mathtt{cur}(\Delta_1) \tag{3.125}
$$

Using induction and S-closure, we can derive Δ'.

$$
\begin{aligned}
\omega^*(\Delta) &= \omega^*(\mathtt{cur}(\Delta_1)) && (3.126)\\
&= \mathtt{cur}(\omega^*(\Delta_1)) \quad \text{By S-closure} && (3.127)\\
&= \mathtt{cur}(\Delta_1') \quad \text{By induction} && (3.128)\\
&= \Delta' && (3.129)
\end{aligned}
$$

□

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.

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

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

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

$$[\![\Phi \vdash \omega\times : \Gamma', x : A \triangleright \Gamma, x : B]\!]_M = [\![\Phi \vdash \omega : \Gamma' \triangleright \Gamma]\!]_M \times [\![A \leq:_\Phi B]\!]_M : \Gamma' \times A \to \Gamma \times B$$

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

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

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

$$[\![\Phi \mid \Gamma' \vdash \diamond : \diamond]\!]_M = \langle\rangle_{\Gamma'} : \Gamma' \to \mathtt{1} \quad [\![\Phi \mid \Gamma' \vdash (\sigma, x := v) : \Gamma, x : A]\!]_M = \langle [\![\Phi \mid \Gamma' \vdash \Gamma :]\!]_{\mathbf{M}}, [\![\mathbf{\Phi} \mid \mathbf{\Gamma'} \vdash \mathbf{v} : \mathbf{A}]\!]_{\mathbf{M}} \rangle : \Gamma' \to \Gamma \times \mathtt{1}$$

We also need to explain the action of these term substitutions on terms. We define the action of applying a substitution σ on term v as $v\,[\sigma]$. The property $x\#\sigma$ indicates tha x does not occur in the domain of σ or as a free variable in any of its substituted terms.

$$x\,[\diamond] = x \tag{3.130}$$
$$x\,[\sigma, x := v] = v \tag{3.131}$$
$$x\,[\sigma, x' := v'] = x\,[\sigma] \quad \text{If } x \neq x' \tag{3.132}$$
$$\mathtt{C}^A\,[\sigma] = \mathtt{C}^A \tag{3.133}$$
$$(\lambda x : A.v)\,[\sigma] = \lambda x : A.(v\,[\sigma]) \quad \text{If } x\#\sigma \tag{3.134}$$
$$(\mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2)\,[\sigma] = \mathtt{if}_A\ v\,[\sigma]\ \mathtt{then}\ v_1\,[\sigma]\ \mathtt{else}\ v_2\,[\sigma] \tag{3.135}$$
$$(v_1\ v_2)\,[\sigma] = (v_1\,[\sigma])\ v_2\,[\sigma] \tag{3.136}$$
$$(\mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2) = \mathtt{do}\ x \leftarrow (v_1\,[\sigma])\ \mathtt{in}\ (v_2\,[\sigma]) \quad \text{If } x\#\sigma \tag{3.137}$$
$$(\Lambda\alpha.v)\,[\sigma] = \Lambda\alpha.(v\,[\sigma]) \tag{3.138}$$
$$(v\ \epsilon)\,[\sigma] = (v\,[\sigma])\ \epsilon \tag{3.139}$$
$$\tag{3.140}$$

In order to prove the quantification case of typing-environment weakening and substitution theorems on terms, as can be seen in the road map in figure 3.3, it will be necessary to be able to weaken the effect environment on term-environment weakenings and substitutions

Let us first consider the action of effect weakenings on these morphisms. The weakening theorem on term-environment weakenings is as so:

Theorem 3.5.11 (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* $[\![\Phi' \vdash \omega : \Gamma' \triangleright \Gamma]\!]_M = \omega_1^* [\![\Phi \vdash \omega : \Gamma' \triangleright \Gamma]\!]_M$.

Proof: By induction on the derivation of ω. making use of weakening on types, type environments, and subtyping.

Case Id: Then $\omega = \iota$, so its denotation is $\omega = \mathtt{Id}_{\Gamma_I}$

So

$$\omega_1^*(\mathtt{Id}_{\Gamma_I}) = \mathtt{Id}_{\Gamma_{I'}} = [\![\Phi' \vdash \iota : \Gamma \triangleright \Gamma]\!]_M \tag{3.141}$$

Case Project: Then $\omega = \omega'\pi$

$$(\text{Project})\frac{\Phi \vdash \omega' : \Gamma' \triangleright \Gamma}{\Phi \vdash \omega\pi : \Gamma', x : A \triangleright \Gamma} \tag{3.142}$$

So $\omega = \omega' \circ \pi_1$

Hence

$$\begin{aligned}
\omega_1^*(\omega) &= \omega_1^*(\omega') \circ \omega_1^*(\pi_1) && (3.143)\\
&= [\![\Phi' \vdash \omega' : \Gamma' \triangleright \Gamma]\!]_M \circ \pi_1 && (3.144)\\
&= [\![\Phi' \vdash \omega'\pi : (\Gamma', x : A) \triangleright \Gamma]\!]_M && (3.145)\\
&= [\![\Phi' \vdash \omega : (\Gamma', x : A) \triangleright \Gamma]\!]_M && (3.146)
\end{aligned}$$

Case Extend: Then $\omega = \omega'\times$

$$(\text{Extend})\frac{\Phi \vdash \omega' : \Gamma' \triangleright \Gamma \;\; A \leq:_\Phi B}{\Phi \vdash \omega\times : (\Gamma', x : A) \triangleright (\Gamma, x : B)} \tag{3.147}$$

So $\omega = \omega' \times [\![A \leq:_\Phi B]\!]_M$

Hence

$$\begin{aligned}
\omega_1^*(\omega) &= (\omega_1^*(\omega') \times \omega_1^*([\![A \leq:_\Phi B]\!]_M) && (3.148)\\
&= ([\![\Phi' \vdash \omega' : \Gamma' \triangleright \Gamma]\!]_M \times [\![A \leq:_{\Phi'} B]\!]_M) && (3.149)\\
&= [\![\Phi' \vdash \omega : (\Gamma', x : A) \triangleright (\Gamma, x : B)]\!]_M && (3.150)
\end{aligned}$$

□

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

Theorem 3.5.12 (Effect Weakening on Term Substitution) *If* $\Phi \mid \Gamma' \vdash \sigma\colon \Gamma$ *and* $\omega : \Phi' \triangleright \Phi$ *then* $\Phi' \mid \Gamma' \vdash \sigma\colon \Gamma$ *and* $[\![\Phi' \mid \Gamma' \vdash \sigma\colon \Gamma]\!]_M = \omega^* [\![\Phi \mid \Gamma' \vdash \sigma\colon \Gamma]\!]_M$

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

Case Nil: Then $\sigma = \langle\rangle_{\Gamma'_I}$, so $\omega^*(\sigma) = \langle\rangle_{\Gamma'_{I'}} = [\![\Phi' \mid \Gamma' \vdash \sigma : \Gamma]\!]_M$

Case Var: Then $\sigma = (\sigma', x := v)$

$$\begin{aligned}
\omega^*\sigma &= \omega * \langle \sigma', [\![\Phi \mid \Gamma \vdash v : A]\!]_M \rangle && (3.151)\\
&= \langle \omega^*\sigma', \omega^*[\![\Phi \mid \Gamma \vdash v : A]\!]_M \rangle && (3.152)\\
&= \langle [\![\Phi' \mid \Gamma' \vdash \sigma' : \Gamma]\!]_M, [\![\Gamma' \mid \Phi' \vdash v : A]\!]_M \rangle && (3.153)\\
&= [\![\Phi' \mid \Gamma' \vdash \sigma : \Gamma, x : A]\!]_M && (3.154)
\end{aligned}$$

□

As with effect substitutions, it is useful to define a couple of special substitutions and lemmas on term substitutions. Firstly, we define the identity and single substitutions in an equivalent way.

$$\begin{aligned}
\Phi \mid \Gamma \vdash \mathtt{Id}_\Gamma : \Gamma && (3.155)\\
\mathtt{Id}_\diamond = \diamond && (3.156)\\
\mathtt{Id}_{\Gamma, x:A} = (\mathtt{Id}_\Gamma, x := x) && (3.157)\\
[v/x] = (\mathtt{Id}_\Gamma, x := v) && (3.158)\\
&& (3.159)
\end{aligned}$$

By inspection, these also have simple denotations, similar to those seen in the case of effect substitutions (Equation 3.23).

$$\begin{aligned}
[\![\Phi \mid \Gamma \vdash \mathtt{Id}_\Gamma : \Gamma]\!]_M : \Gamma \to \Gamma && (3.160)\\
[\![\Phi \mid \Gamma \vdash \mathtt{Id}_\Gamma : \Gamma]\!]_M = \mathtt{Id}_\Gamma && (3.161)\\
[\![\Phi \mid \Gamma \vdash [v/x] : \Gamma, x : A]\!]_M : \Gamma \to \Gamma \times A && (3.162)\\
[\![\Phi \mid \Gamma \vdash [v/x] : \Gamma, x : A]\!]_M = \langle \mathtt{Id}_\Gamma, [\![\Phi \mid \Gamma \vdash v : A]\!]_M \rangle && (3.163)
\end{aligned}$$

Furthermore, it will be useful to have an analogue of the extension lemma for term substitutions.

Lemma 3.5.13 (Extension Lemma on Term Substitutions) *If* $\Phi \mid \Gamma' \vdash \sigma : \Gamma$ *and* $x \# \sigma$ *then* $\Phi \mid (\Gamma', x : A) \vdash (\sigma, x := x) : (\Gamma, x : A)$ *with denotation* $[\![\Phi \mid (\Gamma', x : A) \vdash (\sigma, x := x) : (\Gamma, x : A)]\!]_M = [\![\Phi \mid \Gamma' \vdash \sigma : \Gamma]\!]_M \times \mathit{Id}_A$

Proof: Makes use of the weakening on terms (theorem 3.5.14). If $\Phi \mid \Gamma' \vdash \sigma : \Gamma$ then $\Phi \mid (\Gamma', x : A) \vdash \sigma : \Gamma$ with denotation $\sigma \circ \pi_1$.

□

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. They demonstrate that the can model the action of applying well-formed type-environment

changes by pre-composing the morphisms to be acted on with a morphism modelling the change in environment.

The term-weakening theorem is as so:

Theorem 3.5.14 (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.

Case Effect-Lambda: This case makes use of the effect weakening of term weakenings.

If Δ derives $\Phi \mid \Gamma \vdash \Lambda\alpha.v : \forall\alpha.A$, then by inversion, we have Δ_1 such that

$$\Delta = (\text{Effect-Fn})\frac{()\,\frac{\Delta_1}{\Phi,\alpha \mid \Gamma \vdash v : A}}{\Phi \mid \Gamma \vdash \Lambda\alpha.v : \forall\epsilon.A} \tag{3.164}$$

By induction, we derive Δ_1' such that

$$\Delta' = (\text{Effect-Fn})\frac{()\,\frac{\Delta_1'}{\Phi,\alpha \mid \Gamma' \vdash v : A}}{\Phi \mid \Gamma' \vdash (\Lambda\alpha.v) : \forall\epsilon.A} \tag{3.165}$$

Where

$$\begin{aligned}
\Delta_1' &= \Delta_1 \circ [\![\Phi, \alpha \vdash \omega : \Gamma' \triangleright \Gamma]\!]_M && (3.166)\\
&= \Delta_1 \circ [\![\iota\pi : \Phi, a \triangleright \Phi]\!]^*_M(\omega) && (3.167)\\
&= \Delta_1 \circ \pi_1^*(\omega) && (3.168)
\end{aligned}$$

Hence

$$\begin{aligned}
\Delta \circ \omega &= \overline{\Delta_1} \circ \omega && (3.169)\\
&= \overline{\Delta_1 \circ \pi_1^*(\omega)} && (3.170)\\
&= \overline{\Delta_1'} && (3.171)\\
&= \Delta' && (3.172)
\end{aligned}$$

Case Bind: This case makes use of the properties of an S-category, specifically the tensor strength on the graded monad. By inversion, we have derivations Δ_1, Δ_2 such that:

$$\Delta = (\text{Bind})\frac{()\,\frac{\Delta_1}{\Phi \mid \Gamma \vdash v_1 : \mathtt{M}_{\epsilon_1} A} \quad ()\,\frac{\Delta_2}{\Phi \mid \Gamma, x : A \vdash v_2 : \mathtt{M}_{\epsilon_2} B}}{\Phi \mid \Gamma \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 : \mathtt{M}_{\epsilon_1 \cdot \epsilon_2} B} \tag{3.173}$$

If $\Phi \vdash \omega : \Gamma' \triangleright \Gamma$ then $\Phi \vdash \omega\times : \Gamma', x : A \triangleright \Gamma, x : A$, so by induction, we can derive Δ_1', Δ_2' such that:

$$\Delta' = (\text{Bind})\frac{()\,\frac{\Delta_1'}{\Phi \mid \Gamma' \vdash v_1 : \mathtt{M}_{\epsilon_1} A} \quad ()\,\frac{\Delta_2'}{\Phi \mid \Gamma', x : A \vdash v_2 : \mathtt{M}_{\epsilon_2} B}}{\Phi \mid \Gamma' \vdash \mathtt{do}\ x \leftarrow v_1\ \mathtt{in}\ v_2 : \mathtt{M}_{\epsilon_1 \cdot \epsilon_2} B} \tag{3.174}$$

This preserves denotations:

$$\Delta' = \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2' \circ \mathtt{t}_{\epsilon_1,\Gamma',A} \circ \langle \mathtt{Id}_{G'}, \Delta_1' \rangle \quad \text{By definition} \tag{3.175}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} (\Delta_2 \circ (\omega \times \mathtt{Id}_A)) \circ \mathtt{t}_{\epsilon_1,\Gamma',A} \circ \langle \mathtt{Id}_{G'}, \Delta_1 \circ \omega \rangle \quad \text{By induction on } \Delta_1', \Delta_2' \tag{3.176}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2 \circ \mathtt{t}_{\epsilon_1,\Gamma,A} \circ \langle \omega, \Delta_1 \circ \omega \rangle \quad \text{By tensor strength} \tag{3.177}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2 \circ \mathtt{t}_{\epsilon_1,\Gamma,A} \circ \langle \mathtt{Id}_{\Gamma}, \Delta_1 \rangle \circ \omega \quad \text{By product property} \tag{3.178}$$
$$= \Delta \quad \text{By definition} \tag{3.179}$$

Case Return: We have the sub-derivation Δ_1 such that

$$\Delta = (\text{Return}) \frac{() \frac{\Delta_1}{\Phi | \Gamma \vdash v : A}}{\Phi \mid \Gamma \vdash \mathtt{return} v : \mathtt{M_1} A} \tag{3.180}$$

Hence, by induction, with $\Phi \vdash \omega : \Gamma' \triangleright \Gamma$, we find the derivation Δ_1' such that:

$$\Delta' = (\text{Return}) \frac{() \frac{\Delta_1'}{\Phi | \Gamma' \vdash v : A}}{\Phi \mid \Gamma' \vdash \mathtt{return} v : \mathtt{M_1} A} \tag{3.181}$$

This preserves denotations:

$$\Delta' = \eta_A \circ \Delta_1' \quad \text{By definition} \tag{3.182}$$
$$= \eta_A \circ \Delta_1 \circ \omega \quad \text{By induction of } \Delta_1, \Delta_1' \tag{3.183}$$
$$= \Delta \circ \omega \quad \text{By Definition} \tag{3.184}$$

□

The term-substitution theorem is formed similarly:

Theorem 3.5.15 (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: TODO: More detail on these cases This proof proceeds by induction on the derivation Δ similarly to the term weakening theorem and making use of weakening of term substitutions.

Case Weaken: By inversion, $\Gamma = \Gamma', y : B$ and $\sigma = \sigma', y := v$ and we have Δ_1 deriving:

$$(\text{Weaken}) \frac{() \frac{\Delta_1}{\Phi | \Gamma'' \vdash x : A}}{\Phi \mid \Gamma'', y : B \vdash x : A} \tag{3.185}$$

Also by inversion of the well-formedness of $\Phi \mid \Gamma' \vdash \sigma : \Gamma$, we have $\Phi \mid \Gamma' \vdash \sigma' : \Gamma''$ and

$$[\![\Phi \mid \Gamma' \vdash \sigma : \Gamma]\!]_M = \langle [\![\Phi \mid \Gamma' \vdash \sigma : \Gamma'']\!]_M, [\![\Phi \mid \Gamma' \vdash v : B]\!]_M \rangle \tag{3.186}$$

Hence by induction on Δ_1 we have Δ_1' such that

$$() \frac{\Delta_1'}{\Phi \mid \Gamma' \vdash x\,[\sigma] : A} \tag{3.187}$$

Hence

$$\begin{aligned}
\Delta' &= \Delta_1' \quad \text{By definition} && (3.188)\\
&= \Delta_1 \circ \sigma' \quad \text{By induction} && (3.189)\\
&= \Delta_1 \circ \pi_1 \circ \langle \sigma', [\![\Phi \mid \Gamma' \vdash v : B]\!]_M \rangle \quad \text{By product property} && (3.190)\\
&= \Delta_1 \circ \pi_1 \circ \sigma \quad \text{By defintion of the denotation of } \sigma && (3.191)\\
&= \Delta \circ \sigma \quad \text{By defintion.} && (3.192)
\end{aligned}$$

Case Lambda: By inversion, we have Δ_1 such that

$$\Delta = (\text{Fn}) \frac{() \frac{\Delta_1}{\Phi | \Gamma, x : A \vdash v : B}}{\Phi \mid \Gamma \vdash \lambda x : A . v : A \to B} \tag{3.193}$$

By induction of Δ_1 we have Δ_1' such that

$$\Delta' = (\text{Fn}) \frac{() \frac{\Delta_1'}{\Phi | \Gamma', x : A \vdash (v[\sigma]) : B}}{\Phi \mid \Gamma \vdash (\lambda x : A . v)\,[\sigma] : A \to B} \tag{3.194}$$

By induction and the extension lemma, we have:

$$\Delta_1' = \Delta_1 \circ (\sigma \times \mathtt{Id}_A) \tag{3.195}$$

Hence:

$$\begin{aligned}
\Delta' &= \mathtt{cur}(\Delta_1') \quad \text{By definition} && (3.196)\\
&= \mathtt{cur}(\Delta_1 \circ (\sigma \times \mathtt{Id}_A)) \quad \text{By induction and extension lemma.} && (3.197)\\
&= \mathtt{cur}(\Delta_1) \circ \sigma \quad \text{By the exponential property (Uniqueness)} && (3.198)\\
&= \Delta \circ \sigma \quad \text{By Definition} && (3.199)\\
& && (3.200)
\end{aligned}$$

□

3.6 Uniqueness of Denotations

Up until this point we have hat to be careful about the implicit typing derivation for every term denotation $[\![\Phi \mid \Gamma \vdash v : A]\!]_M$. We are now equipped with the tools to prove that all derivations of the same typing relation instance $\Phi \mid \Gamma \vdash v : A$ 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 first introduce the concept of a *reduced* typing derivation that is unique to each term and type in each typing environment. Next, I present a function, *reduce*, which recursively maps typing derivations to their reduced equivalent. I also prove that this function preserves the denotation of the derivations. That is $[\![\Phi \mid \Gamma \vdash v : A]\!]_M = [\![\mathit{reduce}(\Phi \mid \Gamma \vdash v : A)]\!]_M$. 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 subtyping. The subtyping rule can be inserted into different places in a derivation to derive the same typing relation. Hence, the reduction function focuses on only placing subtyping rule instances in specific places. In particular, we only allow the subtyping 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 subtyping rule uses when it is necessary maintain syntactic correctness of the derivation.

Theorem 3.6.1 (Uniqueness of reduced Derivations) *These reduced derivations are unique.*

Proof: This proof proceeds by induction on the term structure, making use of the unique derivations of the subterms to show that a reduced derivation of the whole term must also be unique. There are no cases for subtyping, as it is not a syntactic feature. As an aside, if subtyping were a syntactic feature, such as explicit casts, then the rules for the typing relation would be entirely syntax directed, and hence we would not have the ambiguity that these theorems are required in order to solve.

Case Bind: This case makes use of the weakening theorem on typing environments.

Let

$$(\text{Subtype})\frac{()\,\frac{\Delta}{\Phi|\Gamma\vdash v_1:\mathtt{M}_{\epsilon_1}A} \quad (\text{Computation})\,\frac{A\leq:_\Phi A' \quad \epsilon_1\leq_\Phi\epsilon_1'}{\mathtt{M}_{\epsilon_1}A\leq:_\Phi\mathtt{M}_{\epsilon_1'}A'}}{\Phi\mid\Gamma\vdash v_1{:}\,\mathtt{M}_{\epsilon_1'}A'} \tag{3.201}$$

$$(\text{Subtype})\frac{()\,\frac{\Delta'}{\Phi|\Gamma,x:A\vdash v_2:\mathtt{M}_{\epsilon_2}B} \quad (\text{Computation})\,\frac{B\leq:_\Phi B' \quad \epsilon_2\leq_\Phi\epsilon_2'}{\mathtt{M}_{\epsilon_2}B\leq:_\Phi\mathtt{M}_{\epsilon_2'}B'}}{\Phi\mid\Gamma,x:A\vdash v_2{:}\,\mathtt{M}_{\epsilon_2'}B'} \tag{3.202}$$

Be the respective unique reduced type derivations of the subterms. By weakening, $\Phi\vdash\iota\times:(\Gamma,x:A)\triangleright(\Gamma,x:A')$ so if there's a derivation of $\Phi\mid(\Gamma,x:A')\vdash v_2{:}\,B$, there's also one of $\Phi\mid\Gamma,x:A\vdash v_2{:}\,B$.

$$(\text{Subtype})\frac{()\,\frac{\Delta''}{\Phi|(\Gamma,x:A')\vdash v_2:\mathtt{M}_{\epsilon_2}B} \quad (\text{Computation})\,\frac{B\leq:_\Phi B' \quad \epsilon_2\leq_\Phi\epsilon_2'}{\mathtt{M}_{\epsilon_2}B\leq:_\Phi\mathtt{M}_{\epsilon_2'}B'}}{\Phi\mid(\Gamma,x:A')\vdash v_2{:}\,\mathtt{M}_{\epsilon_2'}B'} \tag{3.203}$$

Since the effects monoid operation is monotone, if $\epsilon_1\leq_\Phi\epsilon_1'$ and $\epsilon_2\leq_\Phi\epsilon_2'$ then $\epsilon_1\cdot\epsilon_2\leq_\Phi\epsilon_1'\cdot\epsilon_2'$

Hence the reduced type derivation of $\Phi\mid\Gamma\vdash\mathtt{do}\ x\leftarrow v_1\ \mathtt{in}\ v_2{:}\,\mathtt{M}_{\epsilon_1'\cdot\epsilon_2'}B'$ is the following:

TODO: Make this and the other smaller

$$(\text{Type})\frac{(\text{Bind})\dfrac{(\text{Subtype})\dfrac{()\,\frac{\Delta}{\Phi|\Gamma\vdash v_1:\mathtt{M}_{\epsilon_1}A} \quad (\text{Computation})\,\frac{A\leq:_\Phi A' \quad \epsilon_1\leq_\Phi\epsilon_1'}{\mathtt{M}_{\epsilon_1}A\leq:_\Phi\mathtt{M}_{\epsilon_1'}A'}}{\Phi|\Gamma\vdash v_1:\mathtt{M}_{\epsilon_1'}A'} \quad (\text{Subtype})\dfrac{()\,\frac{\Delta''}{\Phi|\Gamma,x:A'\vdash v_2:\mathtt{M}_{\epsilon_2}B} \quad (\text{Computation})\,\frac{B\leq:_\Phi B' \quad \epsilon_2\leq_\Phi\epsilon_2'}{\mathtt{M}_{\epsilon_2}B\leq:_\Phi\mathtt{M}_{\epsilon_2'}B'}}{\Phi|\Gamma,x:A'\vdash v_2:\mathtt{M}_{\epsilon_2'}B'}}{\Phi|\Gamma\vdash\mathtt{do}\ x\leftarrow v_1\ \mathtt{in}\ v_2:\mathtt{M}_{\epsilon_1\cdot\epsilon_2}B}}{\Phi\mid\Gamma\vdash\mathtt{do}\ x\leftarrow v_1\ \mathtt{in}\ v_2{:}\,\mathtt{M}_{\epsilon_1'\cdot\epsilon_2'}B'} \tag{3.204}$$

Case Effect-Fn: The unique reduced derivation of $\Phi\mid\Gamma\vdash\Lambda\alpha.A{:}\,\forall\alpha.B$

is

$$(\text{Subtype})\frac{(\text{Effect-Fn})\,\frac{()\,\frac{\Delta}{\Phi,\alpha|\Gamma\vdash v:A}}{\Phi|\Gamma\vdash\Lambda\alpha.v:\forall\alpha.A} \quad \forall\alpha.A\leq_\Phi\forall\alpha.B}{\Phi\mid\Gamma\vdash\Lambda\alpha.B{:}\,\forall\alpha.B} \tag{3.205}$$

Where

$$(\text{Subtype})\frac{()_{\overline{\Phi,\alpha|\Gamma\vdash v:A}}^{\Delta} \quad A \leq:_{\Phi,\alpha} B}{\Phi,\alpha \mid \Gamma \vdash v: B} \tag{3.206}$$

Is the unique reduced derivation of $\Phi, \alpha \mid \Gamma \vdash v: B$

□

The *reduce* function maps each derivation to its reduced equivalent. It does this by pushing subtyping 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 in figure 3.4.

TODO: Diagram of a type-derivation tree being reduced

Theorem 3.6.2 (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 $\Delta' = reduce(\Delta) \implies \Delta' = \Delta$.*

Proof: We proceed by induction over the structure of Δ, making use of the substitution and weakening theorems. We make use of the the definition of the *reduce* function as defined in figure 3.4. We also use the definitions of Δ_1, Δ_2... from the figure. **TODO: May need to define Δ_1.. locally.**

Case Apply: This case makes use of the fact that composing subtyping morphisms gives the transitive subtyping morphism. Let

$$f = [\![A \leq:_\Phi A']\!]_M : A \to A' \tag{3.215}$$
$$f' = [\![A'' \leq:_\Phi A]\!]_M : A'' \to A \tag{3.216}$$
$$g = [\![B' \leq:_\Phi B]\!]_M : B' \to B \tag{3.217}$$
$$\tag{3.218}$$

Hence

$$[\![A' \to B' \leq:_\Phi A \to B]\!]_M = (g)^A \circ (B')^f \tag{3.219}$$
$$= \mathtt{cur}(\mathtt{app} \circ \mathtt{app}) \circ \mathtt{cur}(\mathtt{app} \circ (\mathtt{Id} \times f)) \tag{3.220}$$
$$= \mathtt{cur}(g \circ \mathtt{app} \circ (\mathtt{Id} \times f)) \tag{3.221}$$

Then

$$\Delta = \mathtt{app} \circ \langle \Delta_1, \Delta_2 \rangle \quad \text{By definition} \tag{3.222}$$
$$= \mathtt{app} \circ \langle \mathtt{cur}(g \circ \mathtt{app} \circ (\mathtt{Id} \times f)) \circ \Delta_1', f' \circ \Delta_2' \rangle \quad \text{By reductions of } \Delta_1, \Delta_2 \tag{3.223}$$
$$= \mathtt{app} \circ (\mathtt{cur}(g \circ \mathtt{app} \circ (\mathtt{Id} \times f)) \times \mathtt{Id}_A) \circ \langle \Delta_1', f' \circ \Delta_2' \rangle \quad \text{Factoring out} \tag{3.224}$$
$$= g \circ \mathtt{app} \circ (\mathtt{Id} \times f) \circ \langle \Delta_1', f' \circ \Delta_2' \rangle \quad \text{By the exponential property} \tag{3.225}$$
$$= g \circ \mathtt{app} \circ \langle \Delta_1', f \circ f' \circ \Delta_2' \rangle \tag{3.226}$$
$$= \Delta' \quad \text{By defintion} \tag{3.227}$$

Case Bind: This is a long case that makes use of the typing-environment weakening theorem on terms.

Case Apply: To find:

$$reduce((\text{Apply})\frac{()\,\frac{\Delta_1}{\Phi|\Gamma\vdash v_1:A\to B}\quad()\,\frac{\Delta_2}{\Phi|\Gamma\vdash v_2:A}}{\Phi\mid\Gamma\vdash v_1\ v_2\colon B}) \tag{3.207}$$

Let

$$(\text{Subtype})\frac{()\,\frac{\Delta_1'}{\Phi|\Gamma\vdash v_1:A'\to B'}\quad A'\to B'\leq:_\Phi A\to B}{\Phi\mid\Gamma\vdash v_1\colon A\to B}=reduce(\Delta_1) \tag{3.208}$$

$$(\text{Subtype})\frac{()\,\frac{\Delta_2'}{\Phi|\Gamma\vdash v:A'}\quad A'\leq:_\Phi A}{\Phi\mid\Gamma\vdash v_1\colon A}=reduce(\Delta_2) \tag{3.209}$$

In

$$(\text{Subtype})\frac{(\text{Apply})\frac{()\,\frac{\Delta_1'}{\Phi|\Gamma\vdash v_1:A'\to B'}\quad(\text{Subtype})\frac{()\,\frac{\Delta_2'}{\Phi|\Gamma\vdash v_2:A''}\quad A''\leq:_\Phi A\leq:_\Phi A'}{\Phi|\Gamma\vdash v_2:A'}}{\Phi|\Gamma\vdash v_1\ v_2:B'}\quad B'\leq:_\Phi B}{\Phi\mid\Gamma\vdash v_1\ v_2\colon B} \tag{3.210}$$

Case Bind: To find

$$reduce((\text{Bind})\frac{()\,\frac{\Delta_1}{\Phi|\Gamma\vdash v_1:\mathtt{M}_{\epsilon_1}A}\quad()\,\frac{\Delta_2}{\Phi|\Gamma,x:A\vdash v_2:\mathtt{M}_{\epsilon_2}B}}{\Phi\mid\Gamma\vdash \mathtt{do}\ x\leftarrow v_1\ \mathtt{in}\ v_2\colon\mathtt{M}_{\epsilon_1\cdot\epsilon_2}B}) \tag{3.211}$$

Let

$$(\text{Subtype})\frac{()\,\frac{\Delta_1'}{\Phi|\Gamma\vdash v_1:\mathtt{M}_{\epsilon_1'}A'}\quad(\text{Computation})\frac{\epsilon_1'\leq_\Phi\epsilon_1\quad A'\leq:_\Phi A}{\mathtt{M}_{\epsilon_1'}A'\leq:_\Phi\mathtt{M}_{\epsilon_1}A}}{\Phi\mid\Gamma\vdash v_1\colon\mathtt{M}_{\epsilon_1}A}=reduce(\Delta_1) \tag{3.212}$$

Since $\Phi\vdash(i,\times):(\Gamma,x:A')\triangleright(\Gamma,x:A)$ if $A'\leq:_\Phi A$, and by $\Delta_2=\Phi\mid(\Gamma,x:A)\vdash v_2\colon\mathtt{M}_{\epsilon_2}B$, there also exists a derivation Δ_3 of $\Phi\mid(\Gamma,x:A')\vdash v_2\colon\mathtt{M}_{\epsilon_2}B$. Δ_3 is derived from Δ_2 simply by inserting a (Subtype) rule below all instances of the (Var) rule.
Let

$$(\text{Subtype})\frac{()\,\frac{\Delta_3'}{\Phi|\Gamma,x:A'\vdash v_2:\mathtt{M}_{\epsilon_2'}B'}\quad(\text{Computation})\frac{\epsilon_1'\leq_\Phi\epsilon_2\quad B'\leq:_\Phi B}{\mathtt{M}_{\epsilon_1'}B'\leq:_\Phi\mathtt{M}_{\epsilon_2}B}}{\Phi\mid\Gamma,x:A'\vdash v_2\colon\mathtt{M}_{\epsilon_2}B}=reduce(\Delta_3) \tag{3.213}$$

Since the effects monoid operation is monotone, if $\epsilon_1\leq_\Phi\epsilon_1'$ and $\epsilon_2\leq_\Phi\epsilon_2'$ then $\epsilon_1\cdot\epsilon_2\leq_\Phi\epsilon_1'\cdot\epsilon_2'$
Then the result of reduction of the whole bind expression is:

$$(\text{Subtype})\frac{(\text{Bind})\frac{()\,\frac{\Delta_1'}{\Phi|\Gamma\vdash v_1:\mathtt{M}_{\epsilon_1'}A'}\quad()\,\frac{\Delta_3'}{\Phi|\Gamma,x:A'\vdash v_2:\mathtt{M}_{\epsilon_2'}B'}}{\Phi|\Gamma\vdash\mathtt{do}\ x\leftarrow v_1\ \mathtt{in}\ v_2:\mathtt{M}_{\epsilon_1'\cdot\epsilon_2'}B}\quad(\text{Computation})\frac{\epsilon_1'\cdot\epsilon_2'\leq_\Phi\epsilon_1\cdot\epsilon_2\quad B'\leq:_\Phi B}{\mathtt{M}_{\epsilon_1'\cdot\epsilon_2'}B'\leq:_\Phi\mathtt{M}_{\epsilon_1\cdot\epsilon_2}B}}{\Phi\mid\Gamma\vdash\mathtt{do}\ x\leftarrow v_1\ \mathtt{in}\ v_2\colon\mathtt{M}_{\epsilon_1\cdot\epsilon_2}B} \tag{3.214}$$

Figure 3.4: Some cases of the reduce function

Let

$$f = [\![A' \leq:_\Phi A]\!]_M : A' \to A \tag{3.228}$$
$$g = [\![B' \leq:_\Phi B]\!]_M : B' \to B \tag{3.229}$$
$$h_1 = [\![\epsilon_1' \leq_\Phi \epsilon_1]\!]_M : T_{\epsilon_1'} \to T_{\epsilon_1} \tag{3.230}$$
$$h_2 = [\![\epsilon_2' \leq_\Phi \epsilon_2]\!]_M : T_{\epsilon_2'} \to T_{\epsilon_2} \tag{3.231}$$
$$h = [\![\epsilon_1' \cdot \epsilon_2' \leq_\Phi \epsilon_1 \cdot \epsilon_2]\!]_M : T_{\epsilon_1' \cdot \epsilon_2'} \to T_{\epsilon_1 \cdot \epsilon_2} \tag{3.232}$$

Due to the denotation of the weakening used to derive Δ_3 from Δ_2, we have

$$\Delta_3 = \Delta_2 \circ (\mathtt{Id}_\Gamma \times f) \tag{3.233}$$

And due to the reduction of Δ_3, we have

$$\Delta_3 = h_{2,B} \circ T_{\epsilon_2'} g \circ \Delta_3' \tag{3.234}$$

So:

$$\Delta = \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2 \circ \mathtt{t}_{\epsilon_1,\Gamma,A} \circ \langle \mathtt{Id}_\Gamma, \Delta_1 \rangle \quad \text{By definition.} \tag{3.235}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2 \circ \mathtt{t}_{\epsilon_1,\Gamma,A} \circ \left\langle \mathtt{Id}_\Gamma, h_{1,A} \circ T_{\epsilon_1'} f \circ \Delta_1' \right\rangle \quad \text{By reduction of } \Delta_1. \tag{3.236}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2 \circ \mathtt{t}_{\epsilon_1,\Gamma,A} \circ (\mathtt{Id}_\Gamma \times h_{1,A}) \circ \left\langle \mathtt{Id}_\Gamma, T_{\epsilon_1'} f \circ \Delta_1' \right\rangle \quad \text{Factor out } h_1 \tag{3.237}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ T_{\epsilon_1} \Delta_2 \circ h_{1,(\Gamma \times A)} \circ \mathtt{t}_{\epsilon_1',\Gamma,A} \circ \left\langle \mathtt{Id}_\Gamma, T_{\epsilon_1'} f \circ \Delta_1' \right\rangle \quad \text{Tensor strength and subeffecting } h_1 \tag{3.238}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ h_{1,B} \circ T_{\epsilon_1'} \Delta_2 \circ \mathtt{t}_{\epsilon_1',\Gamma,A} \circ \left\langle \mathtt{Id}_\Gamma, T_{\epsilon_1'} f \circ \Delta_1' \right\rangle \quad \text{Naturality of } h_1 \tag{3.239}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ h_{1,B} \circ T_{\epsilon_1'} \Delta_2 \circ \mathtt{t}_{\epsilon_1',\Gamma,A} \circ (\mathtt{Id}_\Gamma \times T_{\epsilon_1'} f) \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{Factor out pairing again} \tag{3.240}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ h_{1,B} \circ T_{\epsilon_1'} (\Delta_2 \circ (\mathtt{Id}_\Gamma \times f)) \circ \mathtt{t}_{\epsilon_1',\Gamma,A'} \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{Tensorstrength} \tag{3.241}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ h_{1,B} \circ T_{\epsilon_1'} (\Delta_3) \circ \mathtt{t}_{\epsilon_1',\Gamma,A'} \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{By the definition of } \Delta_3 \tag{3.242}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ h_{1,B} \circ T_{\epsilon_1'} (h_{2,B} \circ T_{\epsilon_2'} g \circ \Delta_3') \circ \mathtt{t}_{\epsilon_1',\Gamma,A'} \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{By the reduction of } \Delta_3 \tag{3.243}$$
$$= \mu_{\epsilon_1,\epsilon_2,B} \circ h_{1,B} \circ T_{\epsilon_1'} h_{2,B} \circ T_{\epsilon_1'} T_{\epsilon_2'} g \circ T_{\epsilon_1'} \Delta_3' \circ \mathtt{t}_{\epsilon_1',\Gamma,A'} \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{Factor out the functor} \tag{3.244}$$
$$= h_B \circ \mu_{\epsilon_1',\epsilon_2',B} \circ T_{\epsilon_1'} T_{\epsilon_2'} g \circ T_{\epsilon_1'} \Delta_3' \circ \mathtt{t}_{\epsilon_1',\Gamma,A'} \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{By the } \mu \text{ and Subtype rule} \tag{3.245}$$
$$= h_B \circ T_{\epsilon_1' \cdot \epsilon_2'} g \circ \mu_{\epsilon_1',\epsilon_2',B'} \circ T_{\epsilon_1'} \Delta_3' \circ \mathtt{t}_{\epsilon_1',\Gamma,A'} \circ \langle \mathtt{Id}_\Gamma, \Delta_1' \rangle \quad \text{By naturality of } \mu \tag{3.246}$$
$$= \Delta' \quad \text{By definition} \tag{3.247}$$

□

$$(\text{Lambda}) \frac{(\text{Subtype}) \frac{\Phi | \Gamma, x:A \vdash v:B \quad B \leq:_\Phi B'}{\Phi | \Gamma, x:A \vdash v:B'}}{\Phi \mid \Gamma \vdash \lambda x:A.v: A \to B'} \tag{3.248}$$

$$(\text{Subtype}) \frac{(\text{Lambda}) \frac{\Phi | \Gamma, x:A \vdash v:B}{\Phi | \Gamma \vdash \lambda x:A.v: A \to B} \quad A \to B \leq: A \to B'}{\Phi \mid \Gamma \vdash \lambda x:A.v: A \to B'} \tag{3.249}$$

Figure 3.5: Two derivations of the same type relation. The second derivation is reduced.

3.7 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, I 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, as seen in figure 3.6, 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, seen in figure 3.7, which formalise how the reduction of subexpressions 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 as seen in figure 3.8.

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

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

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

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

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

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

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

Figure 3.6: The $\beta\eta$-reduction rules for PEC

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

Theorem 3.7.1 (Soundness) *If* $\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2 : A$, *then* $\Phi \mid \Gamma \vdash v_1 : A$, $\Phi \mid \Gamma \vdash v_2 : A$, *and* $[\![\Phi \mid \Gamma \vdash v_1 : A]\!]_M = [\![\Phi \mid \Gamma \vdash v_2 : A]\!]_M$.

Proof: The proof proceeds by induction on the definition of the $\beta\eta$-equivalence relation.

This proof has a lot of cases and each of the $\beta\eta$-reductions makes use of many of the requirements we have placed on the indexed S-category.

I have omitted the congruence cases here as they hold through simple application of the inductive hypothesis on subterms. Similarly, the equivalence relation cases hold simply because equality on mor-

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

$$\text{(Lambda)}\frac{\Phi \mid \Gamma, x : A \vdash v_1 =_{\beta\eta} v_2 : B}{\Phi \mid \Gamma \vdash \lambda x : A.v_1 =_{\beta\eta} \lambda x : A.v_2 : A \to B} \quad \text{(Return)}\frac{\Phi \mid \Gamma \vdash v_1 =_{\beta\eta} v_2 : A}{\Phi \mid \Gamma \vdash \texttt{return} v_1 =_{\beta\eta} \texttt{return} v_2 : \mathtt{M}_{\mathbf{1}} A}$$

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

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

Figure 3.7: The congruence rules for PEC

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

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

Figure 3.8: Rules expanding the $\beta\eta$-reduction and congruence relation to an equivalence relation

phisms is an equivalence relation by definition. I do however, give a selection of the $\beta\eta$-reduction cases to demonstrate the necessity of the S-category requirements.

Case Right Unit: This case makes use of the right-unit monad law.

Let $f = [\![\Phi \mid \Gamma \vdash v : \mathtt{M}_\epsilon A]\!]_M$

$$\begin{aligned}[\![\Phi \mid \Gamma \vdash \texttt{do}\ x \leftarrow v\ \texttt{in}\ \texttt{return} x : \mathtt{M}_\epsilon A]\!]_M &= \mu_{\epsilon,\mathbf{1},A} \circ T_\epsilon(\eta_A \circ \pi_2) \circ \mathtt{t}_{\epsilon,\Gamma,A} \circ \langle \mathtt{Id}_\Gamma, f \rangle \\ &= T_\epsilon \pi_2 \circ \mathtt{t}_{\epsilon,\Gamma,A} \circ \langle \mathtt{Id}_\Gamma, f \rangle \\ &= \pi_2 \circ \langle \mathtt{Id}_\Gamma, f \rangle \\ &= f\end{aligned} \tag{3.250}$$

Case If-True: This case makes use of the co-product diagram on $\mathtt{1} + \mathtt{1}$.

Let

$$f = [\![\Phi \mid \Gamma \vdash v_1 : A]\!]_M \tag{3.251}$$
$$g = [\![\Phi \mid \Gamma \vdash v_2 : A]\!]_M \tag{3.252}$$
$$\tag{3.253}$$

Then

$$\begin{aligned}
[\![\Phi \mid \Gamma \vdash \mathtt{if}_A\ v\ \mathtt{then}\ v_1\ \mathtt{else}\ v_2 \colon A]\!]_M &= \mathtt{app} \circ (([\mathtt{cur}(f \circ \pi_2), \mathtt{cur}(g \circ \pi_2)] \circ \mathtt{inl} \circ \langle\rangle_\Gamma) \times \mathtt{Id}_\Gamma) \circ \delta_\Gamma \\
&= \mathtt{app} \circ ((\mathtt{cur}(f \circ \pi_2) \circ \langle\rangle_\Gamma) \times \mathtt{Id}_\Gamma) \circ \delta_\Gamma \\
&= \mathtt{app} \circ (\mathtt{cur}(f \circ \pi_2) \times \mathtt{Id}_\Gamma) \circ (\langle\rangle_\Gamma \times \mathtt{Id}_\Gamma) \circ \delta_\Gamma \\
&= f \circ \pi_2 \circ \langle \langle\rangle_\Gamma , \mathtt{Id}_\Gamma \rangle \\
&= f \\
&= [\![\Phi \mid \Gamma \vdash v_1 \colon A]\!]_M
\end{aligned} \tag{3.254}$$

Case Effect-Beta: This case makes use of the adjunction properties of $\forall_I, \pi_1^*$.

let

$$h = [\![\Phi \vdash \epsilon \colon \mathtt{Effect}]\!]_M \tag{3.255}$$
$$f = [\![\Phi, \alpha \mid \Gamma \vdash v \colon A]\!]_M \tag{3.256}$$
$$A = [\![\Phi, \alpha \vdash A\,[\alpha/\alpha] : \mathtt{Type}]\!]_M \tag{3.257}$$

Then

$$[\![\Phi \mid \Gamma \vdash \Lambda\alpha.v \colon \forall\alpha.A]\!]_M = \overline{f} \tag{3.258}$$

So

$$[\![\Phi \mid \Gamma \vdash (\Lambda\alpha.v)\ \epsilon \colon \forall\alpha.A]\!]_M = \langle \mathtt{Id}_I, h \rangle^* (\epsilon_A) \circ \overline{f} \tag{3.259}$$
$$= \langle \mathtt{Id}_I, h \rangle^* (\epsilon_A) \circ \langle \mathtt{Id}_I, h \rangle^* (\pi_1^*(\overline{f})) \quad \text{Identity functor} \tag{3.260}$$
$$= \langle \mathtt{Id}_I, h \rangle^* (\epsilon_A \circ \pi_1^*(\overline{f})) \tag{3.261}$$
$$= \langle \mathtt{Id}_I, h \rangle^* (f) \quad \text{By adjunction} \tag{3.262}$$
$$= [\![\Phi \mid \Gamma \vdash v\,[\epsilon/\alpha] : A\,[e/\alpha]\,]\!]_M \quad \text{By substitution theorem} \tag{3.263}$$
$$\tag{3.264}$$

□

This completes the proof of soundness for this semantics.

Chapter 4

Instantiating a Model of PEC

TODO: "We have proved that we can form a model": this is a little confusing because chapter 6 is about forming models from less data. TODO: "There exist Set-based models": this is the case for many side-effects, but not all of them. TODO: "Let C be an S-category formed from Set.": don't use C here (I'm not sure that you actually have to name the S-category at all.) TODO: Define the base category before the fibres. TODO: When you mention the functor category, you need to make it clear that E^n is a discrete category. TODO: The objects of the base category have the form E^n, but the fibres are indexed by natural numbers, so strictly speaking you aren't indexing by the right thing here. It might be better to change the objects of the base category to just be natural numbers. TODO: "discrete sub category": Eff is not discrete (it has non-trivial morphisms). Do you mean full subcategory? (Also, should this be full? Why not restrict to monotone functions?) TODO: You say that C is complete without saying what complete means. (But you don't need completeness, you just need products because E^n is discrete.) TODO: "finite product over the countable set of effects": if E is countable, then this would be a countable product. (But also, you don't know that E is countable.) The size of the product doesn't really matter here, as long as the category has products of the right shape

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 effectful 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 use a `Set` based S-category as a starting point. In this section, I 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 derived from `Set`. That is, $\mathbb{C}$ contains a graded monad $\mathtt{T}^0, \mu^0, \eta^0, \mathtt{t}$, is cartesian closed, has a co-product on the terminal object $\mathtt{1} = \{^*\}$, has subtyping functions $[\![A \leq :_\gamma B]\!]_M : A \to B$ for each instance of the ground subtyping 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 marked 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.

TODO: This spacing will be fixed once I tweak around with my macros Using α-equivalence, we can see that all effect environments of the same length are equivalent. For example, $((\diamond, \alpha, \beta) \mid \Gamma \vdash \lambda f : \mathtt{Int} \to \mathtt{M}_{\alpha\cdot\beta}\mathtt{Unit}.(f\ 0) : \mathtt{M}_{\alpha\cdot\beta}\mathtt{Unit})$ and $((\diamond, \gamma, \delta) \mid \Gamma \vdash \lambda f : \mathtt{Int} \to \mathtt{M}_{\gamma\cdot\delta}\mathtt{Unit}.(f\ 0) : \mathtt{M}_{\gamma\cdot\delta}\mathtt{Unit})$ are equivalent under α-equivalence. Hence, we can reduce an effect environment to the natural number n indicating its length.

Next we must 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

Cartesian Closed Category

Since $\mathbb{C}$ is based on `Set` and a CCC, and , since E is small, E^n is small, $[E^n, \mathbb{C}]$ is a CCC.

$$(A \times B)\vec{\epsilon} = (A\vec{\epsilon}) \times (B\vec{\epsilon}) \tag{4.3}$$
$$\mathtt{1}\vec{\epsilon} = \mathtt{1} \tag{4.4}$$
$$(B^A)\vec{\epsilon} = (B\vec{\epsilon})^{(A\vec{\epsilon})} \tag{4.5}$$
$$\pi_1\vec{\epsilon} = \pi_1 \tag{4.6}$$
$$\pi_2\vec{\epsilon} = \pi_2 \tag{4.7}$$
$$\mathtt{app}\vec{\epsilon} = \mathtt{app} \tag{4.8}$$
$$\mathtt{cur}(f)\vec{\epsilon} = \mathtt{cur}(f\vec{\epsilon}) \tag{4.9}$$
$$\langle f, g\rangle\ \vec{\epsilon} = \langle f\vec{\epsilon}, g\vec{\epsilon}\rangle \tag{4.10}$$
$$\tag{4.11}$$

Figure 4.1: The construction of CCC features in the functor category $[E^n, \mathbb{C}]$

$\mathbb{C}$ given n ground effects. Morphisms between objects are functions which, given a collection of ground effects, yield a morphism (function) 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.

$$E^0 = \mathtt{1} = \{*\} \tag{4.1}$$
$$E^{n+1} = E^n \times E \tag{4.2}$$

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 pointwise with respect to their parameter $\vec{\epsilon} \in E^n$ as seen in figures 4.1 - 4.6. Full proofs of the naturality and monad laws can be found **TODO: Reference**.

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 an S-closed re-indexing functor $\theta^* : [E^n, \mathbb{C}] \to [E^m, \mathbb{C}]$. A simple instantiation is the pre-composition functor, as seen in figure 4.7. This also obeys the composition law of re-indexing functors figure 4.8 and is S-closed.

Next, the re-indexing functor π_1^* should have a right adjoint, $\forall_{E^n}$. Here, we pick $\forall_{E^n}$ to be defined as a finite product over the countable set of effects (figure 4.10). This is possible because types and effects are not impredicative (that is, they quantify over themselves), meaning that the set of effects is countable or finite.

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

$$\overline{(-)} : [E^{n+1}, \mathbb{C}](\pi_1^* A, B) \rightleftharpoons [E^n, \mathbb{C}](A, \forall_{E^n} B) : \widehat{(-)} \tag{4.57}$$

The leftwards and rightwards 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. The inverse

The Terminal Co-Product

We can define the co-product pointwise.

$$
\begin{aligned}
(\mathtt{1}+\mathtt{1})\vec{\epsilon} &= (\mathtt{1}\vec{\epsilon}+\mathtt{1}\vec{\epsilon}) && (4.12)\\
&= (\mathtt{1}+\mathtt{1}) && (4.13)\\
\mathtt{inl}\vec{\epsilon} &= \mathtt{inl} && (4.14)\\
\mathtt{inr}\vec{\epsilon} &= \mathtt{inr} && (4.15)\\
[f,g]\vec{\epsilon} &= [f\vec{\epsilon}, g\vec{\epsilon}] && (4.16)\\
& && (4.17)
\end{aligned}
$$

Figure 4.2: The construction of co-product features in the functor category $[E^n, \mathbb{C}]$

Ground Types and Terms

Each ground type in the non-polymorphic calculus has a fixed denotation $[\![\gamma]\!]_M \in \mathtt{obj}\ \mathbb{C}$. The ground type in the polymorphic calculus hence has a denotation represented by the constant function.

$$
\begin{aligned}
[\![\gamma]\!]_M : &\quad E^n \to \mathtt{obj}\ \mathbb{C} && (4.18)\\
\vec{\epsilon} \mapsto &\quad [\![\gamma]\!]_M && (4.19)\\
& && (4.20)
\end{aligned}
$$

Each constant term $\mathtt{C}^A$ in the non-polymorphic calculus has a fixed denotation $[\![\mathtt{C}^A]\!]_M \in \mathbb{C}(\mathtt{1}, A)$. So the morphism $[\![\mathtt{C}^A]\!]_M$ in $[E^n, \mathbb{C}]$ is the corresponding constant dependently typed morphism.

$$
\begin{aligned}
[\![\mathtt{C}^A]\!]_M : &\quad [E^n, \mathbb{C}](\mathtt{1}, A) && (4.21)\\
\vec{\epsilon} \mapsto &\quad [\![\mathtt{C}^A]\!]_M && (4.22)
\end{aligned}
$$

Figure 4.3: The definition of ground types and terms in the category $[E^n, \mathbb{C}]$

Strong Graded Monad

Given the strong graded monad $(\mathtt{T}^0, \eta^0, \mu^0, \mathtt{t}^0)$ on $\mathbb{C}$ we can construct an appropriate graded monad on $[E^n, \mathbb{C}]$.

$$\mathtt{T}^n : \quad (E^n, \cdot, \leq_n, \mathtt{1}_n) \to [[E^n, \mathbb{C}], [E^n, \mathbb{C}]] \tag{4.23}$$
$$(\mathtt{T}^n_f A)\vec{\epsilon} = \quad \mathtt{T}^0_{(f\vec{\epsilon})} A\vec{\epsilon} \tag{4.24}$$
$$(\eta^n_A)\vec{\epsilon} = \quad \eta^0_{A\vec{\epsilon}} \tag{4.25}$$
$$(\mu^n_{f,g,A})\vec{\epsilon} = \quad \mu^0_{(f\vec{\epsilon}),(g\vec{\epsilon}),(A\vec{\epsilon})} \tag{4.26}$$
$$(\mathtt{t}^n_{f,A,B})\vec{\epsilon} = \quad \mathtt{t}^0_{(f\vec{\epsilon}),(A\vec{\epsilon}),(B\vec{\epsilon})} \tag{4.27}$$

Through some mechanical proof and the naturality of the $\mathbb{C}$ strong graded monad, these morphisms are natural in their type parameters and form a strong graded monad in $[E^n, \mathbb{C}]$

Figure 4.4: The definition of the graded monad in the category $[E^n, C]$

Subeffecting

Given a collection of subeffecting natural transformation in $\mathbb{C}$,

$$[\![\epsilon_1 \leq_0 \epsilon_2]\!]_M : \quad \mathtt{T}^0_{\epsilon_1} \to \mathtt{T}^0_{\epsilon_2} \tag{4.28}$$

We can form subeffect natural transformations in $[E^n, \mathbb{C}]$:

$$[\![f \leq_n g]\!]_M : \quad \mathtt{T}^n_f \to \mathtt{T}^n_g \tag{4.29}$$
$$[\![f \leq_n g]\!]_M A\vec{\epsilon} : \quad \mathtt{T}^n_{f\vec{\epsilon}}(A\vec{\epsilon}) \to \mathtt{T}^n_{g\vec{\epsilon}}(B\vec{\epsilon}) \tag{4.30}$$
$$= \quad [\![f\vec{\epsilon} \leq_0 g\vec{\epsilon}]\!]_M A\vec{\epsilon} \tag{4.31}$$

Figure 4.5: Subeffect natural transformations in the category $[E^n, \mathbb{C}]$

Subtyping

Subtyping in $[E^n, \mathbb{C}]$ holds via subtyping in $\mathbb{C}$

$$[\![A \leq:_n B]\!]_M : \quad A \to B \tag{4.32}$$
$$[\![A \leq:_n B]\!]_M \vec{\epsilon} = \quad [\![A\vec{\epsilon} \leq:_0 B\vec{\epsilon}]\!]_M \tag{4.33}$$

So the subtyping relation $A \leq: B$ forms a morphism in $[E^n, \mathbb{C}]$

Figure 4.6: Definition of subtyping morphisms in the functor category $[E^n, \mathbb{C}]$

The Pre-composition Functor

$$
\begin{aligned}
A \in &\quad [E^n, \mathbb{C}] && (4.34)\\
\theta^*(A)\vec{\epsilon_m} = &\quad A(\theta(\vec{\epsilon_m})) && (4.35)\\
f: &\quad A \to B && (4.36)\\
\theta^*(f)\vec{\epsilon_m} = &\quad f(\theta(\vec{\epsilon_m})) : \theta^*(A) \to \theta^*(B) && (4.37)\\
& && (4.38)
\end{aligned}
$$

Figure 4.7: A description of the pre-composition functor.

Composition of Pre-Composition Functor

$$
\begin{aligned}
\theta^*(\phi^* A)\vec{\epsilon} &= \phi^*(A)(\theta\vec{\epsilon}) && (4.39)\\
&= A(\phi(\theta\vec{\epsilon})) && (4.40)\\
&= A((\phi \circ \theta)\vec{\epsilon}) && (4.41)\\
&= (\phi \circ \theta)^*(A)\vec{\epsilon} && (4.42)
\end{aligned}
$$

Figure 4.8: The pre-composition functor composes in a contravariant fashion.

S-Closure of Re-Indexing Functors

Theorem 4.0.1 *The re-indexing functors are also S-closed. That is all the S-category features in* $[E^n, \mathbb{C}]$ *are preserved by* θ^* *in* $[E^m, \mathbb{C}]$*. For a full list of features please see appendix* ***TODO: Reference****.*

Proof: All of the S-category features are proved pointwise.

Case Graded Monad Functor:

$$
\begin{aligned}
(\theta^* \mathtt{T}^n_f A)\vec{\epsilon} &= \mathtt{T}^n_f A(\theta\vec{\epsilon}) && (4.43)\\
&= \mathtt{T}^0_{(f(\theta\vec{\epsilon}))}(A(\theta\vec{\epsilon})) && (4.44)\\
&= (\mathtt{T}^m_{(f\circ\theta)}\theta^* A)\vec{\epsilon} && (4.45)\\
& && (4.46)
\end{aligned}
$$

Case Terminal Object:

$$
\begin{aligned}
(\theta^* \mathtt{1})\vec{\epsilon} &= \mathtt{1}(\theta\vec{\epsilon}) && (4.47)\\
&= \mathtt{1} && (4.48)\\
&= \mathtt{1}\vec{\epsilon} && (4.49)\\
& && (4.50)
\end{aligned}
$$

$$
\begin{aligned}
(\theta^* \langle\rangle_A)\vec{\epsilon} &= \langle\rangle_A (\theta\vec{\epsilon}) && (4.51)\\
&= \langle\rangle_{A(\theta\vec{\epsilon})} && (4.52)\\
&= \langle\rangle_{\theta^* A}\, \vec{\epsilon} && (4.53)
\end{aligned}
$$

□

Figure 4.9: The pre-composition re-indexing functor f^* is S-Closed

The Quantification Functor

$$
\begin{aligned}
\forall_{E^n} &: [E^{n+1}, \mathbb{C}] \to [E^n, \mathbb{C}] && (4.54)\\
\forall_{E^n}(A)\vec{\epsilon_n} &= \Pi_{\epsilon\in E} A(\vec{\epsilon_n}, \epsilon) && (4.55)\\
\forall_{E^n}(f)\vec{\epsilon_n} &= \Pi_{\epsilon\in E} f(\vec{\epsilon_n}, \epsilon) && (4.56)
\end{aligned}
$$

Figure 4.10: Definition of the quantification functor

The Adjunction Operations

$$m: \quad \pi_1^* A \to B \tag{4.58}$$
$$\overline{m}: \quad A \to \forall_{E^n} B \tag{4.59}$$
$$\overline{m}(\vec{\epsilon_n}) = \quad \langle m(\vec{\epsilon_n}, \epsilon) \rangle_{e \in E} \tag{4.60}$$

$$n: \quad A \to \forall_{E^n} B \tag{4.61}$$
$$\widehat{n}: \quad \pi_1^* A \to B \tag{4.62}$$
$$\widehat{n}(\vec{\epsilon_n}, \epsilon_{n+1}) = \quad \pi_\epsilon \circ g(\vec{\epsilon_n}) \tag{4.63}$$

Figure 4.11: The definition of the adjunction's bijection operations

TODO: fix overflow here TODO: Align frames

Unit

$$\boldsymbol{\eta}_A: \quad A \to \forall_{E^n} \pi_1^* A \tag{4.64}$$
$$\boldsymbol{\eta}_A(\vec{\epsilon_n}) = \quad \left\langle \mathtt{Id}_{A(\vec{\epsilon_n}, e)} \right\rangle_{\epsilon \in E} \tag{4.65}$$

Co-Unit

$$\boldsymbol{\epsilon}_B: \quad \pi_1^* \forall_{E^n} B \to B \tag{4.66}$$
$$\boldsymbol{\epsilon}_B(\vec{\epsilon_n}, \epsilon) = \quad \pi_\epsilon : \Pi_{e \in E} B(\vec{\epsilon_n}, \epsilon) \to \Pi_{e \in E} B(\vec{\epsilon_n}, \epsilon) \tag{4.67}$$

Figure 4.12: The unit and co-unit of the adjunction

is simply to project out the appropriate value of ϵ from the product. These mappings can be seen in figure 4.11. These transformations give rise to the unit and co-unit of the adjunction, which are defined in figure 4.12. The unit and co-unit allow us to prove that this construction is an adjunction (figure 4.13).

Finally, we need to prove that the Beck-Chevalley condition given in 3.2 holds. The proof of the two appropriate theorems are given in figures 4.14 and 4.15

Hence we have proof that our construction is indeed a valid indexed S-category. Importantly, this shows that reasonable models of the PEC are possible to instantiate and that our requirements do not overconstrain potential models to the point that they are not viable for doing actual analysis.

Verifying We Have an Adjunction For any $g : \pi_1^* A \to B$,

$$(\boldsymbol{\epsilon}_B \circ \pi_1^*(\overline{g}))(\vec{\epsilon_n}, \epsilon_{n+1}) = \pi_{\epsilon_{n+1}} \circ \langle g(\vec{\epsilon_n}, \epsilon') \rangle_{\epsilon' \in E} \tag{4.68}$$

$$= g(\vec{\epsilon_n}, \epsilon_{n+1}) \tag{4.69}$$

So $\boldsymbol{\epsilon}_B \circ \pi_1^*(\overline{g}) = g$.

□

Figure 4.13: Proof that we have formed an adjunction

Theorem 4.0.2 (Beck-Chevalley condition, part I) *Firstly, the functors* $(\theta^* \circ \forall_{E^n})$ *and* $(\forall_{E^m} \circ (\theta \times \mathtt{Id}_E)^*)$ *are equal.*

Proof:

$$((\theta^* \circ \forall_{E^n})A)\vec{\epsilon_n} = \theta^*(\forall_{E^n} A)\vec{\epsilon_n} \tag{4.70}$$

$$= (\forall_{E^n} A)(\theta(\vec{\epsilon_n})) \tag{4.71}$$

$$= \Pi_{\epsilon \in E}(A(\theta(\vec{\epsilon_n}), \epsilon)) \tag{4.72}$$

$$= \Pi_{\epsilon \in E}(((\theta \times \mathtt{Id}_U)^* A)(\vec{\epsilon_n}, \epsilon)) \tag{4.73}$$

$$= \forall_{E^m}((\theta \times \mathtt{Id}_E)^* A)\vec{\epsilon_n} \tag{4.74}$$

$$= ((\forall_{E^m} \circ (\theta \times \mathtt{Id}_E)^*)A)\vec{\epsilon_n} \tag{4.75}$$

□

Figure 4.14: Proof that the adjunction satisfies the first part of the Beck-Chevalley condition.

Theorem 4.0.3 (Beck-Chevalley Condition, Part II) *Secondly, the natural transformation* $\overline{(\theta \times \mathtt{Id}_U)^* \boldsymbol{\epsilon}}$ *is equal to the identity natural transformation.*

Proof:

$$\begin{aligned}
\overline{(\theta \times \mathtt{Id}_U)^* \boldsymbol{\epsilon}_A} \vec{\epsilon} &= \langle (\theta \times \mathtt{Id}_U)^* \boldsymbol{\epsilon}_A (\vec{\epsilon}, \epsilon) \rangle_{\epsilon \in E} && (4.76)\\
&= \langle \boldsymbol{\epsilon}_A (\theta \vec{\epsilon}, \epsilon) \rangle_{e \in E} && (4.77)\\
&= \langle \pi_\epsilon \rangle_{\epsilon \in E} : \Pi_{\epsilon \in E} A(\theta \vec{\epsilon}, \epsilon) \to \Pi_{\epsilon \in E} A(\theta \vec{\epsilon}, \epsilon) && (4.78)\\
&= \mathtt{Id}_{\Pi_{\epsilon \in E} A(\theta \vec{\epsilon}, \epsilon)} && (4.79)\\
&= \mathtt{Id}_{\forall_{I'} \circ (\theta \times \mathtt{Id}_U)^* A \vec{\epsilon}} && (4.80)\\
&= \mathtt{Id}_{\theta^* \circ \forall_I} && (4.81)
\end{aligned}$$

□

Figure 4.15: Proof that the adjunciton satisfies the second part of the Beck-Chevalley condition.

Chapter 5

Conclusion

This dissertation has introduced a denotational semantics for polymorphic effect systems in category theory and has proved that non-trivial models capable of interpreting effect-polymorphic languages can indeed be instantiated. The final piece of the puzzle would be to find an adequate model of a given effect-polymorphic language. This means instantiating a sound model such that if the denotations of two terms are equal in the model, then the terms themselves are equal upto some property, such as $\beta\eta$-equivalence or effect behaviour. This is typically a hard problem which is not addressed in presentations of denotational semantics. **TODO: Explain why**.

This work forms a potential foundation for more precise analysis tools which could be used to improve compilers and interpreters in the future. A more precise analysis of programs allows more specific optimisations to be made, where they would not be considered sound under a non-polymorphic system.