Contents

Chapter 1

Language Definition

1.1 Terms

1.1.1 Value Terms

$$\begin{aligned} v ::= & x \\ & \mid \lambda x : A.C \\ & \mid \mathtt{C}^{A} \\ & \mid \mathtt{()} \\ & \mid \mathtt{true} \mid \mathtt{false} \\ & \mid \Lambda\alpha.v \\ & \mid v\ \epsilon \end{aligned} \tag{1.1}$$

1.1.2 Computation Terms

$$\begin{aligned} C ::= & \mathtt{if}_{\epsilon,A}\ v\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2 \\ & \mid v_1\ v_2 \\ & \mid \mathtt{do}\ x \leftarrow C_1\ \mathtt{in}\ C_2 \\ & \mid \mathtt{return} v \end{aligned} \tag{1.2}$$

1.2 Type System

1.2.1 Effects

The effects should form a monotonous, pre-ordered monoid $(E, \cdot, \mathtt{1}, \leq)$ with ground elements e, and denoted by metavariables ϵ, and in language-effect variables α

1.2.2 Types

Ground Types There exists a set γ of ground types, including `Unit`, `Bool`

Value Types

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

Computation Types Computation types are of the form $\mathtt{M}_{\epsilon} A$

1.2.3 Sub-typing

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

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

We extend this relation with the function and effect-lambda sub-typing rules to yield the full sub-typing relation $\leq:$

- (ground)$\frac{A\leq:_\gamma B}{A\leq: B}$
- (Fn)$\frac{A\leq: A' \quad B'\leq: B \quad \epsilon\leq\epsilon'}{A'\to\mathtt{M}_{\epsilon'}B'\leq: A\to\mathtt{M}_\epsilon B}$
- (All)$\frac{A\leq: A'}{\forall\alpha.A\leq:\forall a.A'}$

1.2.4 Type and Effect Environments

A type environment is a snoc-list of value-variable, type pairs, $G ::= \diamond \mid \Gamma, x : A$. An effect environment is a snoc-list of effect-variables.

$\Phi ::= \diamond \mid \Phi, \alpha$

Domain Function on Type Environments

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

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

$\mathtt{Ok}$ Predicate On Effect Environments

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

Well-Formed-ness of effects We define a relation $\Phi \vdash \epsilon$.

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

Well-Formed-ness of Types We define a relation $\Phi \vdash \tau$ on types.

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

Ok Predicate on Type Environments We now define a predicate on type environments and effect environments: $\Phi \vdash \Gamma \texttt{Ok}$

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

1.2.5 Type Rules

Value Typing Rules

- (Const) $\frac{\Phi \vdash \Gamma \texttt{Ok} \quad \Phi \vdash A}{\Phi \mid \Gamma \vdash \texttt{C}^A{:}A}$
- (Unit) $\frac{\Phi \vdash \Gamma \texttt{Ok}}{\Phi \mid \Gamma \vdash \texttt{()}{:}\texttt{Unit}}$
- (True) $\frac{\Phi \vdash \Gamma \texttt{Ok}}{\Phi \mid \Gamma \vdash \texttt{true}{:}\texttt{Bool}}$
- (False) $\frac{\Phi \vdash \Gamma \texttt{Ok}}{\Phi \mid \Gamma \vdash \texttt{false}{:}\texttt{Bool}}$
- (Var) $\frac{\Phi \vdash \Gamma, x{:}A \texttt{Ok}}{\Phi \mid \Gamma, x{:}A \vdash x{:}A}$
- (Weaken) $\frac{\Phi \mid \Gamma \vdash x{:}A}{\Phi \mid \Gamma, y{:}B \vdash x{:}A}$ (if $x \neq y$)
- (Fn) $\frac{\Phi \mid \Gamma, x{:}A \vdash C{:}\texttt{M}_\epsilon B}{\Phi \mid \Gamma \vdash \lambda x{:}A.C{:}A \to \texttt{M}_\epsilon B}$
- (Sub) $\frac{\Phi \mid \Gamma \vdash v{:}A \quad A \leq{:} B}{\Phi \mid \Gamma \vdash v{:}B}$
- (Effect-Abs) $\frac{\Phi, \alpha \mid \Gamma \vdash v{:}A}{\Phi \mid \Gamma \vdash \Lambda \alpha.v{:}\forall \alpha.A}$
- (Effect-apply) $\frac{\Phi \mid \Gamma \vdash v{:}\forall \alpha.A \quad \Phi \vdash \epsilon}{\Phi \mid \Gamma \vdash v\ \epsilon{:}A[\epsilon/\alpha]}$

Computation typing rules

- (Return) $\frac{\Phi \mid \Gamma \vdash v{:}A}{\Phi \mid \Gamma \vdash \texttt{return} v{:}\texttt{M}_{\texttt{1}} A}$
- (Apply) $\frac{\Phi \mid \Gamma \vdash v_1{:}A \to \texttt{M}_\epsilon B \quad \Phi \mid \Gamma \vdash v_2{:}A}{\Phi \mid \Gamma \vdash v_1\ v_2{:}\texttt{M}_\epsilon B}$
- (if) $\frac{\Phi \mid \Gamma \vdash v{:}\texttt{Bool} \quad \Phi \mid \Gamma \vdash C_1{:}\texttt{M}_\epsilon A \quad \Phi \mid \Gamma \vdash C_2{:}\texttt{M}_\epsilon A}{\Phi \mid \Gamma \vdash \texttt{if}_{\epsilon,A}\ V \texttt{ then } C_1 \texttt{ else } C_2{:}\texttt{M}_\epsilon A}$
- (Do) $\frac{\Phi \mid \Gamma \vdash C_1{:}\texttt{M}_{\epsilon_1} A \quad \Phi \mid \Gamma, x{:}A \vdash C_2{:}\texttt{M}_{\epsilon_2} B}{\Phi \mid \Gamma \vdash \texttt{do}\ x \leftarrow C_1 \texttt{ in } C_2{:}\texttt{M}_{\epsilon_1 \cdot \epsilon_2} B}$
- (Subeffect) $\frac{\Phi \mid \Gamma \vdash C{:}\texttt{M}_{\epsilon_1} A \quad A \leq{:} B \quad \epsilon_1 \leq \epsilon_2}{\Phi \mid \Gamma \vdash C{:}\texttt{M}_{\epsilon_2} B}$

1.2.6 Ok Lemma

If $\Phi \mid \Gamma \vdash t{:}\tau$ then $\Phi \vdash \Gamma \texttt{Ok}$.

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

1.3 Beta-Eta-Equivalence

1.3.1 Beta-Eta conversions

- (Lambda-Beta) $\frac{\Phi|\Gamma,x{:}A\vdash C{:}\mathtt{M}_\epsilon B \quad \Phi|\Gamma\vdash v{:}A}{\Phi|\Gamma\vdash(\lambda x{:}A.C)\ v=_{\beta\eta}C[x/v]{:}\mathtt{M}_\epsilon B}$

- (Lambda-Eta) $\frac{\Phi|\Gamma\vdash v{:}A\to\mathtt{M}_\epsilon B}{\Phi|\Gamma\vdash\lambda x{:}A.(v\ x)=_{\beta\eta}v{:}A\to\mathtt{M}_\epsilon B}$

- (Left Unit) $\frac{\Phi|\Gamma\vdash v{:}A \quad \Phi|\Gamma,x{:}A\vdash C{:}\mathtt{M}_\epsilon B}{\Phi|\Gamma\vdash\mathtt{do}\ x\leftarrow\mathtt{return}v\ \mathtt{in}\ C=_{\beta\eta}C[V/x]{:}\mathtt{M}_\epsilon B}$

- (Right Unit) $\frac{\Phi|\Gamma\vdash C{:}\mathtt{M}_\epsilon A}{\Phi|\Gamma\vdash\mathtt{do}\ x\leftarrow C\ \mathtt{in}\ \mathtt{return}x=_{\beta\eta}C{:}\mathtt{M}_\epsilon A}$

- (Associativity) $\frac{\Phi|\Gamma\vdash C_1{:}\mathtt{M}_{\epsilon_1}A \quad \Phi|\Gamma,x{:}A\vdash C_2{:}\mathtt{M}_{\epsilon_2}B \quad \Phi|\Gamma,y{:}B\vdash C_3{:}\mathtt{M}_{\epsilon_3}C}{\Phi|\Gamma\vdash\mathtt{do}\ x\leftarrow C_1\ \mathtt{in}\ (\mathtt{do}\ y\leftarrow C_2\ \mathtt{in}\ C_3)=_{\beta\eta}\mathtt{do}\ y\leftarrow(\mathtt{do}\ x\leftarrow C_1\ \mathtt{in}\ C_2)\ \mathtt{in}\ C_3{:}\mathtt{M}_{\epsilon_1\cdot\epsilon_2\cdot\epsilon_3}C}$

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

- (if-true) $\frac{\Phi|\Gamma\vdash C_1{:}\mathtt{M}_\epsilon A \quad \Phi|\Gamma\vdash C_2{:}\mathtt{M}_\epsilon A}{\Phi|\Gamma\vdash\mathtt{if}_{\epsilon,A}\ \mathtt{true}\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2=_{\beta\eta}C_1{:}\mathtt{M}_\epsilon A}$

- (if-false) $\frac{\Phi|\Gamma\vdash C_2{:}\mathtt{M}_\epsilon A \quad \Phi|\Gamma\vdash C_1{:}\mathtt{M}_\epsilon A}{\Phi|\Gamma\vdash\mathtt{if}_{\epsilon,A}\ \mathtt{false}\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2=_{\beta\eta}C_2{:}\mathtt{M}_\epsilon A}$

- (If-Eta) $\frac{\Phi|\Gamma,x{:}\mathtt{Bool}\vdash C{:}\mathtt{M}_\epsilon A \quad \Phi|\Gamma\vdash v{:}\mathtt{Bool}}{\Phi|\Gamma\vdash\mathtt{if}_{\epsilon,A}\ v\ \mathtt{then}\ C[\mathtt{true}/x]\ \mathtt{else}\ C\big[\mathtt{false}/x\big]=_{\beta\eta}C[V/x]{:}\mathtt{M}_\epsilon A}$

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

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

1.3.2 Equivalence Relation

- (Reflexive) $\frac{\Phi|\Gamma\vdash t{:}\tau}{\Phi|\Gamma\vdash t=_{\beta\eta}t{:}\tau}$

- (Symmetric) $\frac{\Phi|\Gamma\vdash t_1=_{\beta\eta}t_2{:}\tau}{\Phi|\Gamma\vdash t_2=_{\beta\eta}t_1{:}\tau}$

- (Transitive) $\frac{\Phi|\Gamma\vdash t_1=_{\beta\eta}t_2{:}\tau \quad \Phi|\Gamma\vdash t_2=_{\beta\eta}t_3{:}\tau}{\Phi|\Gamma\vdash t_1=_{\beta\eta}t_3{:}\tau}$

1.3.3 Congruences

- (Effect-Abs) $\frac{\Phi,\alpha|\Gamma\vdash v_1=_{\beta\eta}v_2{:}A}{\Phi|\Gamma\vdash\Lambda\alpha.v_1=_{\beta\eta}\Lambda\alpha.v_2{:}\forall\alpha.A}$

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

- (Lambda) $\frac{\Phi|\Gamma,x{:}A\vdash C_1=_{\beta\eta}C_2{:}\mathtt{M}_\epsilon B}{\Phi|\Gamma\vdash\lambda x{:}A.C_1=_{\beta\eta}\lambda x{:}A.C_2{:}A\to\mathtt{M}_\epsilon B}$

- (Return) $\frac{\Phi|\Gamma\vdash v_1=_{\beta\eta}v_2{:}A}{\Phi|\Gamma\vdash\mathtt{return}v_1=_{\beta\eta}\mathtt{return}v_2{:}\mathtt{M}_{\mathtt{1}}A}$

- (Apply) $\frac{\Phi|\Gamma\vdash v_1=_{\beta\eta}v_1'{:}A\to\mathtt{M}_\epsilon B \quad \Phi|\Gamma\vdash v_2=_{\beta\eta}v_2'{:}A}{\Phi|\Gamma\vdash v_1\ v_2=_{\beta\eta}v_1'\ v_2'{:}\mathtt{M}_\epsilon B}$

- (Bind) $\frac{\Phi|\Gamma\vdash C_1=_{\beta\eta}C_1'{:}\mathtt{M}_{\epsilon_1}A \quad \Phi|\Gamma,x{:}A\vdash C_2=_{\beta\eta}C_2'{:}\mathtt{M}_{\epsilon_2}B}{\Phi|\Gamma\vdash\mathtt{do}\ x\leftarrow C_1\ \mathtt{in}\ C_2=_{\beta\eta}\mathtt{do}\ c\leftarrow C_1'\ \mathtt{in}\ C_2'{:}\mathtt{M}_{\epsilon_1\cdot\epsilon_2}B}$

- (If) $\frac{\Phi|\Gamma\vdash v=_{\beta\eta}v'{:}\mathtt{Bool} \quad \Phi|\Gamma\vdash C_1=_{\beta\eta}C_1'{:}\mathtt{M}_\epsilon A \quad \Phi|\Gamma\vdash C_2=_{\beta\eta}C_2'{:}\mathtt{M}_\epsilon A}{\Phi|\Gamma\vdash\mathtt{if}_{\epsilon,A}\ v\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2=_{\beta\eta}\mathtt{if}_{\epsilon,A}\ v\ \mathtt{then}\ C_1'\ \mathtt{else}\ C_2'{:}\mathtt{M}_\epsilon A}$

- (Subtype) $\frac{\Phi | \Gamma \vdash v =_{\beta\eta} v' : A \quad A \leq: B}{\Phi | \Gamma \vdash v =_{\beta\eta} v' : B}$

- (Subeffect) $\frac{\Phi | \Gamma \vdash C =_{\beta\eta} C' : \mathtt{M}_{\epsilon_1} A \quad A \leq: B \quad \epsilon_1 \leq \epsilon_2}{\Phi | \Gamma \vdash C =_{\beta\eta} C' : \mathtt{M}_{\epsilon_2} B}$

Chapter 2

Category Requirements

2.1 CCC

The section should be a cartesian closed category. That is it should have:

- A Terminal object $\mathtt{1}$
- Binary products
- Exponentials

Further more, it should have a co-product of the terminal object $\mathtt{1}$. This is required for the beta-eta equivalence of `if-then-else` terms.

$\mathtt{1} \xrightarrow{inl} A \xleftarrow{inr} \mathtt{1}$

For each:

$\mathtt{1} \xrightarrow{f} A \xleftarrow{g} \mathtt{1}$

There exists unique $[f, g] : \mathtt{1} + \mathtt{1} \to A$ such that:

$$\begin{tikzcd} & A & \\ \mathtt{1} \arrow[ur, "f"] \arrow[r, "\mathtt{inl}"'] & \mathtt{1} + \mathtt{1} \arrow[u, "{[f,g]}"] & \mathtt{1} \arrow[ul, "g"'] \arrow[l, "\mathtt{inr}"] \end{tikzcd}$$

2.2 Graded Pre-Monad

The category should have a graded pre-monad. That is:

- An endo-functor indexed by the po-monad on effects: $T : (\mathbb{E}, \cdot\, \mathtt{1}, \leq) \to \mathtt{Cat}(\mathbb{C}, \mathbb{C})$
- A unit natural transformation: $\eta : \mathtt{Id} \to T_{\mathtt{1}}$
- A join natural transformation: $\mu_{\epsilon_1, \epsilon_2,} : T_{\epsilon_1} T_{\epsilon_2} \to T_{\epsilon_1 \cdot \epsilon_2}$

Subject to the following commutative diagrams:

2.2.1 Left Unit

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

2.2.2 Right Unit

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

2.2.3 Associativity

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

2.3 Tensor Strength

The category should also have tensorial strength over its products and monads. That is, it should have a natural transformation

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

Satisfying the following rules:

2.3.1 Left Naturality

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

2.3.2 Right Naturality

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

2.3.3 Unitor Law

$$
\begin{tikzcd}
\mathtt{1} \times T_\epsilon A \arrow[r, "\mathtt{t}_{\epsilon,\mathtt{1},A}"] \arrow[dr, "\lambda_{T_\epsilon A}"'] & T_\epsilon(\mathtt{1} \times A) \arrow[d, "T_\epsilon(\lambda_A)"] \\
 & T_\epsilon A
\end{tikzcd}
$$

Where $\lambda : \mathtt{1} \times \mathtt{Id} \to \mathtt{Id}$ is the left-unitor. ($\lambda = \pi_2$)

Tensor Strength and Projection Due to the left-unitor law, we can develop a new law for the commutivity of π_2 with $\mathtt{t}_{,,}$

$$\pi_{2,A,B} = \pi_{2,\mathtt{1},B} \circ (\langle\rangle_A \times \mathtt{Id}_B)$$

And $\pi_{2,\mathtt{1}}$ is the left unitor, so by tensorial strength:

$$
\begin{aligned}
T_\epsilon \pi_2 \circ \mathtt{t}_{\epsilon,A,B} &= T_\epsilon \pi_{2,\mathtt{1},B} \circ T_\epsilon(\langle\rangle_A \times \mathtt{Id}_B) \circ \mathtt{t}_{\epsilon,A,B} \\
&= T_\epsilon \pi_{2,1,B} \circ \mathtt{t}_{\epsilon,\mathtt{1},B} \circ (\langle\rangle_A \times \mathtt{Id}_B) \\
&= \pi_{2,1,B} \circ (\langle\rangle_A \times \mathtt{Id}_B) \\
&= \pi_2
\end{aligned}
\tag{2.1}
$$

So the following commutes:

$$
\begin{tikzcd}
A \times T_\epsilon B \arrow[r, "\mathtt{t}_{\epsilon,A,B}"] \arrow[dr, "\pi_2"] & T_\epsilon(A \times B) \arrow[d, "T_\epsilon \pi_2"] \\
& T_\epsilon B
\end{tikzcd}
$$

2.3.4 Commutativity with Join

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

2.4 Commutivity with Unit

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

2.5 Commutivity with α

Let $\alpha_{A,B,C} = \langle \pi_1 \circ \pi_1, \langle \pi_2 \circ \pi_1, \pi_2 \rangle\rangle : ((A \times B) \times C) \to (A \times (B \times C))$

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

TODO: Needed?

2.6 Subeffecting

For each instance of the pre-order $(\mathbb{E}, \leq)$, $\epsilon_1 \leq \epsilon_2$, there exists a natural transformation $[\![\epsilon_1 \leq \epsilon_2]\!] : T_{\epsilon_1} \to T_{\epsilon_2}$ that commutes with $\mathtt{t}_{,,}$:

2.6.1 Subeffecting and Tensor Strength

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

2.6.2 Sub-effecting and Monadic Join

Since the monoid operation on effects is monotone, we can introduce the following diagram.

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

2.7 Subtyping

The denotation of ground types $[\![\_]\!]_M$ is a functor from the pre-order category of ground types $(\gamma, \leq:_\gamma)$ to $\mathbb{C}$. This pre-ordered sub-category of $\mathbb{C}$ is extended with the rule for function subtyping to form a larger pre-ordered sub-category of $\mathbb{C}$.

$$
\text{(Function Subtyping)} \frac{f = [\![A' \leq: A]\!]_M \quad g = [\![B \leq: B']\!]_M \quad h = [\![\epsilon_1 \leq \epsilon_2]\!]}{rhs = [\![A \to \mathtt{M}_{\epsilon_1} B \leq: A' \to \mathtt{M}_{\epsilon_2} B']\!]_M : (T_{\epsilon_1} B)^A \to (T_{\epsilon_2} B')^{A'}}
$$

$$
\begin{aligned}
rhs =& (h_{B'} \circ T_{\epsilon_1} g)^{A'} \circ (T_{\epsilon_1} B)^f \\
=& \mathtt{cur}(h_{B'} \circ T_{\epsilon_1} g \circ \mathtt{app}) \circ \mathtt{cur}(\mathtt{app} \circ (\mathtt{Id}_{T_{\epsilon_1} B^{A'}} \times f))
\end{aligned} \tag{2.2}
$$

Chapter 3

Denotations

3.1 Helper Morphisms

3.1.1 Diagonal and Twist Morphisms

In the definition and proofs (Especially of the the If cases), I make use of the morphisms twist and diagonal.

$$\tau_{A,B} : (A \times B) \to (B \times A) = \langle \pi_2, \pi_1 \rangle \tag{3.1}$$
$$\delta_A : A \to (A \times A) = \langle \mathtt{Id}_A, \mathtt{Id}_A \rangle \tag{3.2}$$

3.2 Denotations of Types

3.2.1 Denotation of Ground Types

3.2.2 Denotation of Polymorphic Types

3.2.3 Denotation of Computation Type

3.2.4 Denotation of Function Types

3.2.5 Denotation of Type Environments

3.2.6 Denotation of Value Terms

3.2.7 Denotation of Computation Terms

Chapter 4

Unique Denotations

4.1 Reduced Type Derivation

A reduced type derivation is one where subtype and subeffect rules must, and may only, occur at the root or directly above an **if**, or **apply** rule.

In this section, I shall prove that there is at most one reduced derivation of $\Gamma \vdash t\colon \tau$. Secondly, I shall present a function for generating reduced derivations from arbitrary typing derivations, in a way that does not change the denotations. These imply that all typing derivations of a type-relation have the same denotation.

4.2 Reduced Type Derivations are Unique

4.2.1 Variables

4.2.2 Constants

4.2.3 Value Terms

4.2.4 Computation Terms

4.3 Each type derivation has a reduced equivalent with the same denotation.

4.3.1 Constants

4.3.2 Value Types

4.3.3 Computation Types

4.4 Denotations are Equivalent

Chapter 5

Weakening

5.1 Weakening Definition

5.1.1 Relation

5.1.2 Weakening Denotations

5.2 Weakening Theorems

5.2.1 Domain Lemma

5.2.2 Theorem 1

5.2.3 Theorem 2

5.2.4 Theorem 3

5.3 Proof of Theorems 2 and 3

5.3.1 Variable Terms

5.3.2 Computation Terms

Chapter 6

Substitution

6.1 Introduce Substitutions

6.1.1 Substitutions as SNOC lists

$$\sigma ::= \diamond \mid \sigma, x := v \tag{6.1}$$

6.1.2 Trivial Properties of substitutions

$\mathtt{fv}(\sigma)$

$$\mathtt{fv}(\diamond) = \emptyset \tag{6.2}$$
$$\mathtt{fv}(\sigma, x := v) = \mathtt{fv}(\sigma) \cup \mathtt{fv}(v) \tag{6.3}$$

$\mathtt{dom}(\sigma)$

$$\mathtt{dom}(\diamond) = \emptyset \tag{6.4}$$
$$\mathtt{dom}(\sigma, x := v) = \mathtt{dom}(\sigma) \cup \{x\} \tag{6.5}$$

$x\#\sigma$

$$x\#\sigma \Leftrightarrow x \notin (\mathtt{fv}(\sigma) \cup \mathtt{dom}(\sigma`)) \tag{6.6}$$

6.1.3 Effect of substitutions

We define the effect of applying a substitution σ as

$$t\,[\sigma]$$

$$x\,[\diamond] = x \tag{6.7}$$
$$x\,[\sigma, x := v] = v \tag{6.8}$$
$$x\,[\sigma, x' := v'] = x\,[\sigma] \quad \text{If } x \neq x' \tag{6.9}$$
$$\mathtt{C}^A\,[\sigma] = \mathtt{C}^A \tag{6.10}$$
$$(\lambda x : A.C)\,[\sigma] = \lambda x : A.(C\,[\sigma]) \quad \text{If } x\#\sigma \tag{6.11}$$
$$(\mathtt{if}_{\epsilon,A}\ v\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2)\,[\sigma] = \mathtt{if}_{\epsilon,A}\ v\,[\sigma]\ \mathtt{then}\ C_1\,[\sigma]\ \mathtt{else}\ C_2\,[\sigma] \tag{6.12}$$
$$(v_1\ v_2)\,[\sigma] = (v_1\,[\sigma])\ v_2\,[\sigma] \tag{6.13}$$
$$(\mathtt{do}\ x \leftarrow C_1\ \mathtt{in}\ C_2) = \mathtt{do}\ x \leftarrow (C_1\,[\sigma])\ \mathtt{in}\ (C_2\,[\sigma]) \quad \text{If } x\#\sigma \tag{6.14}$$
$$\tag{6.15}$$

6.1.4 Well Formedness

6.1.5 Simple Properties Of Substitution

If $\Gamma' \vdash \sigma\colon \Gamma$ then: **TODO: Number these**

Property 1: $\Gamma\texttt{Ok}$ and $\Gamma'\texttt{Ok}$ Since $\Gamma'\texttt{Ok}$ holds by the Nil-axiom. $\Gamma\texttt{Ok}$ holds by induction on the well-formed-ness relation.

Property 2: $\omega : \Gamma'' \triangleright \Gamma'$ implies $\Gamma'' \vdash \sigma\colon \Gamma$. By induction over well-formed-ness relation. For each $x := v$ in σ, $\Gamma'' \vdash v\colon A$ holds if $\Gamma' \vdash v\colon A$ holds.

Property 3: $x \notin (\texttt{dom}(\Gamma) \cup \texttt{dom}(\Gamma''))$ implies $(\Gamma', x : A) \vdash (\sigma, x := x)\colon (\Gamma, x : A)$ Since $\iota\pi : \Gamma', x : A \triangleright \Gamma'$, so by (Property 2) **TODO: Better referencing here**,

$$\Gamma', x : A \vdash \sigma\colon \Gamma$$

In addition, $\Gamma', x : A \vdash x\colon A$ trivially, so by the rule **Extend**, well-formed-ness holds for

$$(\Gamma', x : A) \vdash (\sigma, x := v)\colon (\Gamma, x : A) \tag{6.16}$$

6.2 Substitution Preserves Typing

6.2.1 Variables

Case Var

Case Weaken

6.2.2 Other Value Terms

Case Lambda

Case Constants

6.2.3 Computation Terms

Case Return

Case Apply

Case If

Case Bind

6.2.4 Sub-typing and Sub-effecting

Case Sub-type

Case Sub-effect

6.3 Semantics of Substitution

6.3.1 Denotation of Substitutions

6.3.2 Extension Lemma

6.3.3 Substitution Theorem

6.3.4 Proof For Value Terms

Case Var

Case Weaken

Case Constants

Case Lambda

Case Sub-type

6.3.5 Proof For Computation Terms

Case Return

Case Apply

Case If

Case Bind

Case Subeffect

6.4 The Identity Substitution

6.4.1 Properties of the Identity Substitution

Property 1

Property 2

Chapter 7

Beta Eta Equivalence (Soundness)

7.1 Beta and Eta Equivalence

7.1.1 Beta-Eta conversions

7.1.2 Equivalence Relation

7.1.3 Congruences

7.2 Beta-Eta Equivalence Implies Both Sides Have the Same Type

7.2.1 Equivalence Relations

Case Symmetric

Case Transitive

7.2.2 Beta conversions

Case Lambda

Case Associativity

Case Eta

Case If-True

7.2.3 Congruences

Case Lambda

Case Return

Case Apply

Case Bind

Case If

Case Subtype

Case subeffect

7.3 Beta-Eta equivalent terms have equal denotations

7.3.1 Equivalence Relation

Case Reflexive

Case Symmetric

Case Transitive

7.3.2 Beta Conversions

Case Lambda

Case Left Unit

Case Right Unit

Case Associative

Case Eta

Case If-True

Case If-False

7.3.3 Case If-Eta

7.3.4 Congruences

Case Lambda

Case Return

Case Apply

Case Bind

Case If

Case Subtype

Case subeffect