0.1 Beta and Eta Equivalence

0.1.1 Beta conversions

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

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

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

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

0.1.2 Equivalence Relation

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

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

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

0.1.3 Congruences

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

- (Return) $\frac{\Gamma \vdash v_1 =_{\beta\eta} v_2{:}A}{\Gamma \vdash \mathtt{return}\, v_1 =_{\beta\eta} \mathtt{return}\, v_2{:}\mathtt{M}_1 A}$

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

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

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

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

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

0.2 Beta-Eta equivalence implies both have same type

Each derivation of $\Gamma \vdash t =_{\beta\eta} t' \colon \tau$ can be converted to a derivation of $\Gamma \vdash t \colon \tau$ and $\Gamma \vdash t' \colon \tau$ by induction over the beta-eta equivalence relation derivation.

0.2.1 Equivalence Relations

Case Reflexive By inversion we have a derivation of $\Gamma \vdash t \colon \tau$.

Case Symmetric By inversion $\Gamma \vdash t' =_{\beta\eta} t \colon \tau$. Hence by induction, derivations of $\Gamma \vdash t' \colon \tau$ and $\Gamma \vdash t \colon \tau$ are given.

Case Transitive By inversion, there exists t_2 such that $\Gamma \vdash t_1 =_{\beta\eta} t_2 \colon \tau$ and $\Gamma \vdash t_2 =_{\beta\eta} t_3 \colon \tau$. Hence by induction, we have derivations of $\Gamma \vdash t_1 \colon \tau$ and $\Gamma \vdash t_3 \colon \tau$

0.2.2 Beta conversions

Case Lambda By inversion, we have $\Gamma, x : A \vdash C\colon \mathtt{M}_\epsilon B$ and $\Gamma \vdash v\colon A$. Hence by the typing rules, we have:

$$\text{(Apply)}\frac{\text{(Lambda)}\frac{\Gamma,x:A\vdash C:\mathtt{M}_\epsilon B}{\Gamma\vdash\lambda x:A.C:A\to\mathtt{M}_\epsilon B} \qquad \Gamma \vdash v\colon A}{\Gamma \vdash (\lambda x : A.C) \quad v\colon \mathtt{M}_\epsilon A}$$

By the substitution rule **TODO: which?**, we have

$$\text{(Substitution)}\frac{\Gamma, x : A \vdash C\colon \mathtt{M}_\epsilon B \qquad \Gamma \vdash v\colon A}{\Gamma \vdash C\,[v/x] : \mathtt{M}_\epsilon B}$$

Case Left Unit **TODO: Use left unit commutivity diagram**

Case Right Unit **TODO: Use left unit commutivity diagram** Let $f = [\![\Gamma \vdash C\colon \mathtt{M}_\epsilon A]\!]_M$ **TODO: Proof of $T_\epsilon \pi_2$ distributing with tensor strength to give π_2**

Tensor Strength and Projection

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

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

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

$$\begin{aligned} [\![\Gamma \vdash \mathtt{do} \quad x \leftarrow C \quad \mathtt{in} \quad \mathtt{return} x\colon \mathtt{M}_\epsilon A]\!]_M &= \mu_{\epsilon,\mathtt{1},A} \circ T_\epsilon(\eta_A \circ \pi_2) \circ \mathtt{t}_{\epsilon,\Gamma,A} \circ \langle \mathtt{Id}_\Gamma, f\rangle \\ &= T_\epsilon\pi_2 \circ \mathtt{t}_{\epsilon,\Gamma,A} \circ \langle \mathtt{Id}_\Gamma, f\rangle \\ &= \pi_2 \circ \langle \mathtt{Id}_\Gamma, f\rangle \\ &= f \end{aligned} \tag{2}$$

Case Associative **TODO: Long proof from book. Maybe use a big diagram.**

0.2.3 Congruences

Each congruence rule corresponds exactly to a type derivation rule. To convert to a type derivation, convert all preconditions, then use the equivalent type derivation rule. **TODO: These can be proved simply by using the recursive case and substituting values**

Case Lambda

Case Return

Case Apply

Case Bind

Case If

Case Subtype

Case subeffect

0.3 Beta-Eta equivalent terms have equal denotations

If $t \vdash t' =_{\beta\eta} \tau$: then $[\![\Gamma \vdash t\colon \tau]\!]_M = [\![\Gamma \vdash t'\colon \tau]\!]_M$

By induction over Beta-eta equivalence relation.

0.3.1 Equivalence Relation

The cases over the equivalence relation laws hold by the uniqueness of denotations and the fact that equality over morphisms is an equivalence relation.

Case Reflexive Equality is reflexive, so if $\Gamma \vdash t\colon \tau$ then $[\![\Gamma \vdash t\colon \tau]\!]_M$ is equal to itself.

Case Symmetric By inversion, if $\Gamma \vdash t =_{\beta\eta} t'\colon \tau$ then $\Gamma \vdash t' =_{\beta\eta} t\colon \tau$, so by induction $[\![\Gamma \vdash t'\colon \tau]\!]_M = [\![\Gamma \vdash t\colon \tau]\!]_M$ and hence $[\![\Gamma \vdash t\colon \tau]\!]_M = [\![\Gamma \vdash t'\colon \tau]\!]_M$

Case Transitive There must exist t_2 such that $\Gamma \vdash t_1 =_{\beta\eta} t_2\colon \tau$ and $\Gamma \vdash t_2 =_{\beta\eta} t_3\colon \tau$, so by induction, $[\![\Gamma \vdash t_1\colon \tau]\!]_M = [\![\Gamma \vdash t_2\colon \tau]\!]_M$ and $[\![\Gamma \vdash t_2\colon \tau]\!]_M = [\![\Gamma \vdash t_3\colon \tau]\!]_M$. Hence by transitivity of equality, $[\![\Gamma \vdash t_1\colon \tau]\!]_M = [\![\Gamma \vdash t_3\colon \tau]\!]_M$

0.3.2 Beta Conversions

These cases are typically proved using the properties of a cartesian closed category with a strong graded monad.

Case Lambda

Case Left Unit

Case Right Unit

Case Associative

0.3.3 Congruences

These cases can be proved fairly mechanically by assuming the preconditions, using induction to prove that the matching pairs of subexpressions have equal denotations, then constructing the denotations of the expressions using the equal denotations which gives trivially equal denotations.

Case Lambda

Case Return

Case Apply

Case Bind

Case If

Case Subtype

Case subeffect