

1 Object Language

1.1 Raw Terms

The untyped syntax for CBPV OSum, add separating connectives, remove error.

Value Types	A	$::=$	X Unit $\text{Case } A$ OSum $A \times A$ $A * A$ $\exists X.A$ $U\underline{B}$
Computation Types	\underline{B}	$::=$	$A \rightarrow \underline{B}$ $A \multimap \underline{B}$ $\forall X.\underline{B}$ FA
Values	V	$::=$	x tt σ $\text{inj}_V V$ (V, V) $(V * V)$ $\text{pack } (A, V) \text{ as } \exists X.A$ $\text{thunk } M$
Computations	M	$::=$	$\lambda x: A.M$ MV $\alpha x: A.M$ $M@V$ $\Lambda X.M$ $M[A]$ $\text{ret } V$ $\text{force } V$ $\text{newcase}_A x; M$ $\text{match } V \text{ with } V \{ \text{inj } x.M \parallel N \}$ $\text{let } (x, x) = V; M$ $\text{let } (x * x) = V; M$ $\text{unpack } (X, x) = V; M$
Value Context	Γ	$::=$	\cdot $\Gamma, x: A$ $\Gamma * x: A$
Type Context	Δ	$::=$	\cdot Δ, X

1.2 Typed Terms

$$\begin{array}{c}
\frac{}{\Delta; \Gamma, x : A \vdash_v x : A} \\
\\
\frac{}{\Delta; \Gamma * x : A \vdash_v x : A} \\
\\
\frac{}{\Delta; \Gamma \vdash_v \text{tt} : \text{Unit}} \\
\\
\frac{\Delta; \Gamma \vdash_v \sigma : \text{Case}A \quad \Delta; \Gamma \vdash_v V : A}{\Delta; \Gamma \vdash_v \text{inj}_\sigma V : \text{OSum}} \\
\\
\frac{\Delta; \Gamma \vdash_v V_1 : A_1 \quad \Delta; \Gamma \vdash_v V_2 : A_2}{\Delta; \Gamma \vdash_v (V_1, V_2) : A_1 \times A_2} \\
\\
\frac{\Delta; \Gamma_1 \vdash_v V_1 : A_1 \quad \Delta; \Gamma_2 \vdash_v V_2 : A_2}{\Delta; \Gamma_1 * \Gamma_2 \vdash_v (V_1 * V_2) : A_1 * A_2} \\
\\
\frac{\Delta; \Gamma \vdash_v V : A[A'/X]}{\Delta; \Gamma \vdash_v \text{pack}(A', V) \text{ as } \exists X.A : \exists X.A} \\
\\
\frac{\Delta; \Gamma \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_v \text{thunk } M : \underline{UB}} \\
\\
\frac{\Delta; \Gamma, x : A \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_c \lambda x : A.M : A \rightarrow \underline{B}} \\
\\
\frac{\Delta; \Gamma \vdash_c M : A \rightarrow \underline{B} \quad \Delta; \Gamma \vdash_v N : A}{\Delta; \Gamma \vdash_c MN : \underline{B}} \\
\\
\frac{\Delta; \Gamma * x : A \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_c \alpha x : A.M : A \multimap \underline{B}} \\
\\
\frac{\Delta; \Gamma_1 \vdash_c M : A \multimap \underline{B} \quad \Delta; \Gamma_2 \vdash_v N : A}{\Delta; \Gamma_1 * \Gamma_2 \vdash_c M @ N : \underline{B}} \\
\\
\frac{\Delta, X; \Gamma \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_c \Lambda X.M : \forall X.\underline{B}} \\
\\
\frac{\Delta; \Gamma \vdash_c M : \forall X.\underline{B} \quad \Delta \vdash A}{\Delta; \Gamma \vdash_c M[A] : \underline{B}[A/X]} \\
\\
\frac{\Delta; \Gamma \vdash_v V : A}{\Delta; \Gamma \vdash_c \text{ret } V : FA}
\end{array}$$

$$\begin{array}{c}
\frac{\Delta; \Gamma \vdash_v V : U \underline{B}}{\Delta; \Gamma \vdash_c \text{force } V : \underline{B}} \\
\\
\frac{\Delta; \Gamma * (\sigma : \text{Case } A) \vdash_c M : \underline{B} \quad \Delta \vdash A}{\Delta; \Gamma \vdash_c \text{newcase}_A x; M : \underline{B}} \\
\\
\frac{\Delta; \Gamma \vdash_v V : \text{OSum} \quad \Delta; \Gamma \vdash_v \sigma : \text{Case } A \quad \Delta; \Gamma, x : A \vdash M : \underline{B} \quad \Delta; \Gamma \vdash_c N : \underline{B}}{\Delta; \Gamma \vdash_c \text{match } V \text{ with } \sigma \{ \text{inj } x.M \parallel N \} : \underline{B}} \\
\\
\frac{\Delta; \Gamma \vdash_v V : A_1 \times A_2 \quad \Delta; \Gamma, x : A_1, y : A_2 \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_c \text{let } (x, y) = V; M : \underline{B}} \\
\\
\frac{\Delta; \Gamma \vdash_v V : A_1 * A_2 \quad \Delta; \Gamma * x : A_1 * y : A_2 \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_c \text{let } (x * y) = V; M : \underline{B}} \\
\\
\frac{\Delta \vdash \underline{B} \quad \Delta; \Gamma \vdash_v V : \exists X. A \quad \Delta, X; \Gamma, x : A \vdash_c M : \underline{B}}{\Delta; \Gamma \vdash_c \text{unpack}(X, x) = V; M : \underline{B}}
\end{array}$$

2 Meta Language

2.1 Raw Formulas

$$\begin{array}{lcl}
\text{Formula } \phi, \psi & ::= & t =_A u \\
& | & R(t, u) \\
& | & \phi \implies \psi \\
& | & \phi \wedge \psi \\
& | & \phi \vee \psi \\
& | & \exists x : A. \phi \\
& | & \exists X. \phi \\
& | & \exists \underline{X}. \phi \\
& | & \exists R : \text{Rel}[A, B]. \phi \\
& | & \forall x : A. \phi \\
& | & \forall X. \phi \\
& | & \forall \underline{X}. \phi \\
& | & \forall R : \text{Rel}[A, B]. \phi
\end{array}$$

2.2 Typed Formulas

Propositions, or well-formed formulas, use a term environment Γ , type environment Δ and relation environment Θ . The typing judgement for Propositions is $\Delta; \Gamma; \Theta \vdash_p P$. There are value relations and computation relations.

$$\begin{array}{c}
\frac{\Delta; \Gamma \vdash_- t : A \quad \Delta; \Gamma \vdash_- u : A}{\Delta; \Gamma; \Theta \vdash_p t =_A u} \text{ for } v, c \\
\\
\frac{\Delta; \Gamma \vdash_- t : A \quad \Delta; \Gamma \vdash_- u : B \quad R : \text{Rel}_-[A, B] \in \Theta}{\Delta; \Gamma; \Theta \vdash_p R(t, u)} \text{ for } v, c
\end{array}$$

$$\frac{\Delta; \Gamma; \Theta \vdash_p \phi \quad \Delta; \Gamma; \Theta \vdash_p \psi}{\Delta; \Gamma; \Theta \vdash_p \phi \Box \psi} \Box \in \{\wedge, \vee, \implies\}$$

what about *? Something like exists fresh

$$\frac{\Delta; \Gamma, x : A | \Theta \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \exists x : A. \phi}$$

$$\frac{\Delta; \Gamma; \Theta \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \exists X. \phi} X \notin FV(\Delta, \Gamma, \Theta)$$

$$\frac{\Delta; \Gamma; \Theta \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \exists \underline{X}. \phi} \underline{X} \notin FV(\Delta, \Gamma, \Theta)$$

$$\frac{\Delta; \Gamma; \Theta, R : Rel_ [A, B] \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \exists R : Rel_ [A, B]. \phi} \text{ for } \{v, c\}$$

$$\frac{\Delta; \Gamma, x : A | \Theta \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \forall x : A. \phi}$$

$$\frac{\Delta; \Gamma; \Theta \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \forall X. \phi} X \notin FV(\Delta, \Gamma, \Theta)$$

$$\frac{\Delta; \Gamma; \Theta \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \forall \underline{X}. \phi} \underline{X} \notin FV(\Delta, \Gamma, \Theta)$$

$$\frac{\Delta; \Gamma; \Theta, R : Rel_ [A, B] \vdash_p \phi}{\Delta; \Gamma; \Theta \vdash_p \forall R : Rel_ [A, B]. \phi} \text{ for } \{v, c\}$$

2.3 Typed Relations

Relations are of the form $(x : A, y : B). \phi$ where ϕ is a proposition that can use x, y . The typing judgement for relations is $\Delta; \Gamma; \Theta \vdash_r (x : A, y : B). \phi : Rel_ [A, B]$. The body of the relation is a proposition. Here we pay attention to the difference between value and computation relations.

again, what about *?

$$\frac{\Delta; \Gamma, x : A \vdash_v t : C \quad \Delta; \Gamma, y : B \vdash_v u : C}{\Delta; \Gamma; \Theta \vdash_r (x : A, y : B). t =_C u : Rel_v [A, B]}$$

secretly inserting stoup

$$\frac{\Delta; \Gamma | x : \underline{A} \vdash_c t : \underline{C} \quad \Delta; \Gamma | y : \underline{B} \vdash_c u : \underline{C}}{\Delta; \Gamma; \Theta \vdash_r (x : \underline{A}, y : \underline{B}). t =_{\underline{C}} u : Rel_c [\underline{A}, \underline{B}]}$$

Given some $x : A$ and $y : B$ the terms t, u are related by R , thus we have a relation on A, B . Think of these like a lambda abstraction over two parameters. If the body is related, we can

$$\frac{\Delta; \Gamma, x : A \vdash_v t : C \quad \Delta; \Gamma, y : B \vdash_v u : D}{\Delta; \Gamma; \Theta, R : Rel_v[C, D] \vdash_r (x : A, y : B).R(t, u) : Rel_v[A, B]}$$

$$\frac{\Delta; \Gamma | x : \underline{A} \vdash_c t : \underline{C} \quad \Delta; \Gamma | y : \underline{B} \vdash_c u : \underline{D}}{\Delta; \Gamma; \Theta, \underline{R} : Rel_c[\underline{C}, \underline{D}] \vdash_r (x : \underline{A}, y : \underline{B}).\underline{R}(t, u) : Rel_c[\underline{A}, \underline{B}]}$$

What is the intuition here? This rule is in figure 5 of the PE logic paper.

$$\frac{\Delta; \Gamma; \Theta \vdash_p \phi \quad \Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\psi : Rel_v A, B}{\Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\phi \implies \psi : Rel_v[A, B]} \text{ also } c \text{ version}$$

?

$$\frac{\Delta; \Gamma; \Theta \vdash ? \quad \Delta; \Gamma; \Theta \vdash ?}{\Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\phi \wedge \psi : Rel_v[A, B]} \text{ also } c \text{ version}$$

$$\frac{\Delta; \Gamma, z : C; \Theta \vdash_r (x : A, y : B).\phi : Rel_v[A, B]?}{\Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\forall(z : C).\phi : Rel_v[A, B]} \text{ also } c \text{ version}$$

$$\frac{\Delta, X; \Gamma; \Theta \vdash_r (x : A, y : B).\phi : Rel_v[A, B]?}{\Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\forall X.\phi : Rel_v[A, B]} \text{ also } c \text{ version}$$

$$\frac{\Delta, X; \Gamma; \Theta, R : Rel_{\mathbf{n}}[C, D] \vdash_r (x : A, y : B).\phi : Rel_{\mathbf{m}}[A, B]}{\Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\forall(R : Rel_{\mathbf{n}}[C, D]).\phi : Rel_{\mathbf{m}}[A, B]} \mathbf{n}, \mathbf{m} \in \{v, c\}$$

Analogous versions for \exists connectives.

2.4 Deduction Rules

The judgement for deduction sequence are of the form $\Delta; \Gamma; \Theta; \Phi \vdash_d \psi$ where Δ is a type environment, Γ is a term environment, Θ is a relation environment, Φ is a proposition environment, and ψ is a proposition. like term intro and elim, but without proof terms also for computations?

$$\frac{\Delta; \Gamma \vdash_v t : A}{\Delta; \Gamma; \Theta; \Phi \vdash_d t =_A t} \text{ refl}$$

$$\frac{\Delta; \Gamma; \Theta; \Phi \vdash_d t =_A u \quad \Delta; \Gamma; \Theta; \Phi \vdash_d \phi[t/x]}{\Delta; \Gamma; \Theta; \Phi \vdash_d \phi[u/x]} \text{ subst}$$

$$\frac{\Delta; \Gamma; \Theta; \Phi, \phi \vdash_d \psi}{\Delta; \Gamma; \Theta; \Phi \vdash_d \phi \implies \psi} \implies \text{ Intro}$$

$$\frac{\Delta; \Gamma; \Theta; \Phi \vdash_d \phi \implies \psi \quad \Delta; \Gamma; \Theta; \Phi \vdash_d \phi}{\Delta; \Gamma; \Theta; \Phi \vdash_d \psi} \implies \text{Elim}$$

and familiar rules for logical and (\wedge)

$$\frac{\Delta; \Gamma, (x : A); \Theta; \Phi \vdash_d \phi}{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall(x : A).\phi} \forall \text{ Term Intro}$$

$$\frac{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall(x : A).\phi \quad \Delta; \Gamma \vdash_v t : A}{\Delta; \Gamma; \Theta; \Phi \vdash_d \phi[t/x]} \forall \text{ Term Elim}$$

$$X \notin FV(..), \text{ also for } c \frac{\Delta, X; \Gamma; \Theta; \Phi \vdash_d \phi}{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall X.\phi} \forall \text{ Type Intro}$$

$$\text{also for } c \frac{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall X.\phi \quad \Delta \vdash A}{\Delta; \Gamma; \Theta; \Phi \vdash_d \phi[A/X]} \forall \text{ Type Elim}$$

$$\text{also for } c \frac{\Delta; \Gamma; \Theta, R : Rel_v[A, B]; \Phi \vdash_d \phi}{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall(R : Rel_v[A, B]).\phi} \forall \text{ Rel Intro}$$

$$\text{also for } c \frac{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall(R : Rel_v[A, B]).\phi \quad \Delta; \Gamma; \Theta \vdash_r (x : A, y : B).\psi : Rel_v[A, B]}{\Delta; \Gamma; \Theta; \Phi \vdash_d \phi[\psi[t/x, u/y]/R(t, u)]} \forall \text{ Rel Elim}$$

2.5 Axioms & Axiom Schemas

2.5.1 Congruences

$$\frac{\Delta; \Gamma, (x : A) \vdash_c t, u : \underline{B} \quad \Delta; \Gamma, (x : A); \Theta; \Phi \vdash_d t = u}{\Delta; \Gamma; \Theta; \Phi \vdash_d (\lambda(x : A).t) =_{A \rightarrow \underline{B}} (\lambda(x : A).u)} \lambda \text{ cong}, x \notin FV(\Phi)$$

2.5.2 Beta / Eta Laws

$$\frac{}{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall(u : A).((\lambda x : A.t)u =_B t[u/x])} \lambda\beta$$

$$\frac{}{\Delta; \Gamma; \Theta; \Phi \vdash_d \forall X. \forall Y. \forall(f : X \rightarrow Y).((\lambda x : X.f x) =_{X \rightarrow Y} f)} \lambda\eta$$

2.5.3 Parametricity

2.6 Relational interpretation of Types

Let \mathbf{X} and $\underline{\mathbf{X}}$ be vectors of value type and computation type variables of length n . Let $\underline{\rho}$ be a vector of value relations $\Delta; \Gamma; \Theta \vdash_r \rho_i : Rel_v[C_i, C'_i]$ for all $i \in 1..n$. Let $\underline{\rho}$ be a vector of computation relations $\Delta; \Gamma; \Theta \vdash_r \rho_i : Rel_c[\underline{C}_i, \underline{C}'_i]$ for all $i \in 1..n$. Let A be a **value type** with $FTV(A) \in \{\mathbf{X}, \underline{\mathbf{X}}\}$. Define:

$$A[\underline{\rho}/\mathbf{X}, \underline{\rho}/\underline{\mathbf{X}}] : Rel_v[A[\underline{C}/\mathbf{X}, \underline{C}/\underline{\mathbf{X}}], A[\underline{C}'/\mathbf{X}, \underline{C}'/\underline{\mathbf{X}}]]$$

by induction on A . [Note: PE Logic defines the relational interpretation of value types, but not computation types?](#) . It seems like this relation needs to be indexed by worlds to describe how Case A and OSum are related.

$$\begin{aligned}
X_i[\underline{\rho}, \underline{\rho}] &= \rho_i \\
\text{Unit}[\underline{\rho}, \underline{\rho}] &= (x : \text{Unit}, y : \text{Unit}). x =_{\text{Unit}} y \\
\text{CaseA}[\underline{\rho}, \underline{\rho}] &= (x : \text{Case } (A[\underline{C}/\underline{X}, \underline{C}/\underline{X}]), y : \text{Case } (A[\underline{C}'/\underline{X}, \underline{C}'/\underline{X}])). \\
&\quad \text{foo} \\
\text{OSum}[\underline{\rho}, \underline{\rho}] &= \text{same injection, related values} \\
A \times A'[\underline{\rho}, \underline{\rho}] &= ((x, y) : A \times A'[\underline{C}/\underline{X}, \underline{C}/\underline{X}], (x', y') : A \times A'[\underline{C}'/\underline{X}, \underline{C}'/\underline{X}]). \\
&\quad A[\underline{\rho}, \underline{\rho}](x, x') \wedge A'[\underline{\rho}, \underline{\rho}](y, y') \\
A * A'[\underline{\rho}, \underline{\rho}] &= \text{same as product.. but fiddle with world index?} \\
\exists X. A[\underline{\rho}, \underline{\rho}] &= \text{standard} \\
UB[\underline{\rho}, \underline{\rho}] &= \text{related thunks?} \\
(A \multimap B)[\underline{\rho}, \underline{\rho}] &= \text{related inputs to related outputs?}
\end{aligned}$$

2.7 Example Derivations

2.7.1 Equality Reasoning

Transitivity

$$\begin{array}{c}
\frac{\dots; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X y) \wedge (y =_X z)}{\dots; (x =_X y) \wedge (y =_X z) \vdash_d y =_X z} \quad \frac{\dots; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X y) \wedge (y =_X z)}{\dots; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X y)} \\
\frac{\dots; (x =_X y) \wedge (y =_X z) \vdash_d y =_X z \quad \dots; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X y)[y/y]}{\dots; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X y)[z/y]} \text{subst} \\
\frac{X; x, y, z; \cdot; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X y)[z/y]}{X; x, y, z; \cdot; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X z)} \\
\frac{X; x, y, z; \cdot; (x =_X y) \wedge (y =_X z) \vdash_d (x =_X z)}{\vdash_d \forall X. \forall (xyz : X). (x =_X y) \wedge (y =_X z) \implies (x =_X z)}
\end{array}$$

Symmetry Use IdExt and opRel?

2.7.2 Extensionality

$$\begin{array}{c}
\frac{X, \underline{Y}; f, g, x; \cdot; (\forall(x : X).fx =_{\underline{Y}} gx) \vdash_d \forall(x : X).(fx =_{\underline{Y}} gx) \text{ in } \Phi \quad \frac{\dots \vdash_v x : X}{\dots} \text{Var}}{\dots \vdash_d \forall(x : X).(fx =_{\underline{Y}} gx)} \forall \text{Elim} \\
\frac{X, \underline{Y}; f, g, x; \cdot; (\forall(x : X).fx =_{\underline{Y}} gx) \vdash_d (fx =_{\underline{Y}} gx)}{X, \underline{Y}; f, g; \cdot; (\forall(x : X).fx =_{\underline{Y}} gx) \vdash_d ((\lambda x : X.fx) =_{X \rightarrow \underline{Y}} (\lambda x : X.gx))} \lambda \text{cong} \\
\frac{X, \underline{Y}; f, g; \cdot; (\forall(x : X).fx =_{\underline{Y}} gx) \vdash_d (f =_{X \rightarrow \underline{Y}} g)}{X, \underline{Y}; f, g; \cdot; \cdot \vdash_d ((\forall(x : X).fx =_{\underline{Y}} gx) \implies f =_{X \rightarrow \underline{Y}} g)} \lambda\eta \\
\frac{X, \underline{Y}; f, g; \cdot; \cdot \vdash_d ((\forall(x : X).fx =_{\underline{Y}} gx) \implies f =_{X \rightarrow \underline{Y}} g)}{\dots \vdash_d \forall(f, g : X \rightarrow \underline{Y}).((\forall(x : X).fx =_{\underline{Y}} gx) \implies f =_{X \rightarrow \underline{Y}} g)} \text{Intros} \\
\frac{\dots \vdash_d \forall(f, g : X \rightarrow \underline{Y}).((\forall(x : X).fx =_{\underline{Y}} gx) \implies f =_{X \rightarrow \underline{Y}} g)}{\dots \vdash_d \forall(f, g : X \rightarrow \underline{Y}).((\forall(x : X).fx =_{\underline{Y}} gx) \implies f =_{X \rightarrow \underline{Y}} g)} \text{Intros}
\end{array}$$

2.7.3 Identity Extension Lemma

By Induction on Types.

For Unit Recall

$$eq_A := (x : A, y : A).x =_A y$$

and substitution of a relation into a base type is just the relational interpretation of the base type.

$$Unit[eq_{Unit}] = \llbracket Unit \rrbracket_{Rel} = \{(tt, tt)\}$$

What rules are missing here to make this proof go through?

$$\begin{array}{c}
\frac{\dots; \Phi, \{u = tt, v = tt\} \vdash_d tt =_{Unit} tt}{\dots; \Phi, u(\llbracket Unit \rrbracket_{Rel})v \vdash_d u =_{Unit} v} \text{Refl} \quad \frac{\dots; \Phi, tt =_{Unit} tt \vdash_d tt(\llbracket Unit \rrbracket_{Rel})tt}{\dots; \Phi, u =_{Unit} v \vdash_d u(\llbracket Unit \rrbracket_{Rel})v} \text{By Def} \\
\frac{\dots; \Phi, u(\llbracket Unit \rrbracket_{Rel})v \vdash_d u =_{Unit} v}{\dots \vdash_d u(\llbracket Unit \rrbracket_{Rel})v \implies u =_{Unit} v} \text{Unit } \eta \quad \frac{\dots; \Phi, u =_{Unit} v \vdash_d u(\llbracket Unit \rrbracket_{Rel})v}{\dots \vdash_d u =_{Unit} v \implies u(\llbracket Unit \rrbracket_{Rel})v} \text{Unit } \eta? \\
\frac{\dots \vdash_d u(\llbracket Unit \rrbracket_{Rel})v \implies u =_{Unit} v}{\dots \vdash_d u(\llbracket Unit \rrbracket_{Rel})v \implies u(\llbracket Unit \rrbracket_{Rel})v} \text{Intro} \quad \frac{\dots \vdash_d u =_{Unit} v \implies u(\llbracket Unit \rrbracket_{Rel})v}{\dots \vdash_d u =_{Unit} v \implies u(\llbracket Unit \rrbracket_{Rel})v} \text{Intro} \\
\frac{\dots \vdash_d u(\llbracket Unit \rrbracket_{Rel})v \implies u =_{Unit} v}{\dots \vdash_d u(\llbracket Unit \rrbracket_{Rel})v \implies u(\llbracket Unit \rrbracket_{Rel})v} \text{Logical Equiv} \\
\frac{\dots; u : Unit, v : Unit; \cdot; \cdot \vdash_d u(\llbracket Unit \rrbracket_{Rel})v \equiv u =_{Unit} v}{\dots; \cdot; \cdot; \cdot \vdash_d \forall u : Unit, v : Unit. u(\llbracket Unit \rrbracket_{Rel})v \equiv u =_{Unit} v} \text{Intros} \\
\frac{\dots; \cdot; \cdot; \cdot \vdash_d \forall u : Unit, v : Unit. u(\llbracket Unit \rrbracket_{Rel})v \equiv u =_{Unit} v}{\dots; \cdot; \cdot; \cdot \vdash_d \forall u : Unit, v : Unit. u(Unit[eq_{Unit}])v \equiv u =_{Unit} v} \text{Relational Subst}
\end{array}$$

For $X \rightarrow \underline{Y}$ Recall

$$\begin{aligned}
(X \rightarrow \underline{Y})[eq_X, eq_Y] &= (f : (X \rightarrow \underline{Y})g : (X \rightarrow \underline{Y})). \\
&\quad \forall(x : X). \forall(x' : X). \\
&\quad x(eq_X)x' \implies (fx)(eq_Y)(gx')
\end{aligned}$$

One direction.

$$\begin{array}{c}
\frac{\text{(the relation in above on } f, g) \text{ in } \Phi}{\dots} \text{Var} \quad \frac{\dots \vdash_v x : X}{\dots} \text{Var} \quad \frac{\dots \vdash_v x : X}{\dots} \text{Var} \quad \frac{\dots \vdash_v x : X}{\dots} \text{Var} \quad \frac{\dots \vdash_v x : X}{\dots} \text{Var} \\
\frac{\dots \vdash_d \forall(x' : X).x(eq_X)x' \implies fx =_{\underline{Y}} gx'}{\dots} \forall \text{ Elim} \quad \frac{\dots \vdash_v x : X}{\dots} \forall \text{ Elim} \quad \frac{\dots \vdash_v x : X}{\dots} \forall \text{ Elim} \quad \frac{\dots \vdash_v x : X}{\dots} \forall \text{ Elim} \quad \frac{\dots \vdash_v x : X}{\dots} \forall \text{ Elim} \\
\frac{\dots; f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \vdash_d x(eq_X)x \implies fx =_{\underline{Y}} gx}{\dots} \text{IdExt} \quad \frac{\dots; f, g, x; \dots; f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \vdash_d fx(eq_Y)gx}{\dots} \text{IdExt} \\
\frac{\dots; f, g, x; \dots; f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \vdash_d fx =_{\underline{Y}} gx}{\dots; f, g; \dots; f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \vdash_d (\lambda x : X.fx) =_{X \rightarrow \underline{Y}} (\lambda x : X.gx)} \lambda \text{ cong} \\
\frac{\dots; f, g; \dots; f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \vdash_d f =_{X \rightarrow \underline{Y}} g}{\dots; f, g; \dots \vdash_d f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \implies f =_{X \rightarrow \underline{Y}} g} \text{Intros} \\
\frac{\dots \vdash_d \forall(f, g : X \rightarrow \underline{Y}).f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \implies f =_{X \rightarrow \underline{Y}} g}{\dots \vdash_d \forall(f, g : X \rightarrow \underline{Y}).f((X \rightarrow \underline{Y})[eq_X, eq_Y])g \implies f =_{X \rightarrow \underline{Y}} g} \text{Intros}
\end{array}
\implies \text{Elim}$$

2.7.4 Identity Function

$$\frac{\frac{X; f : \forall X.X \rightarrow FX, x : X; \cdot \vdash_d f[X]x =_{FX} \text{ret } x}{\cdot; f : \forall X.X \rightarrow FX; \cdot \vdash_d \forall X.X \rightarrow FX} \text{Intro}$$

2.7.5 Church Encodings

We can only church encode the F type.

2.7.6 Practice

Unit Practice parametricity proofs for System F. Let

$$Unit := \forall X.X \rightarrow X$$

and

$$tt := \Lambda X.\lambda(x : X).x$$

Prove

$$\vdash_d : \forall(u : Unit).tt = u$$

By the parametricity schema for type Unit, we have

$$\forall(p : (\forall X.X \rightarrow X)).\forall Y.\forall Z.\forall(R : Rel[Y, Z]).((X \rightarrow X)[R])(p[Y], p[Z])$$

Specialized to u, X, X , we have

$$\forall(R : Rel[X, X]).((X \rightarrow X)[R])(u[X], u[X])$$

which *evaluates* to

$$\forall(R : Rel[X, X]).\forall(y : X).\forall(z : X).R(y, z) \implies R(u[X]y, u[X]z)$$

choosing R to be $X; x : X; \cdot \vdash_r (y : X, z : X).z = x$, we use $\forall Rel \text{ Elim}$ to get

$$\frac{\frac{\forall(R : Rel[X, X]).\forall(y : X).\forall(z : X).R(y, z) \implies R(u[X]y, u[X]z)}{\vdash_d \forall(y : X).\forall(z : X).z = x \implies u[X]z = x}}{\quad} \quad \frac{X; x : X; \cdot \vdash_r (y : X, z : X).z = x}{\quad}$$

specializing y, z to x, x

$$x = x \implies u[X]x = x$$

which we can apply to refl to yield

$$u[X]x = x$$

$$\frac{\frac{\frac{\vdots}{X; u : Unit, x : X \vdash_d x = u[X](x)}{X; u : Unit \vdash_d \lambda(x : X).x = \lambda(x : X).u[X](x)} \lambda \text{ cong}}{u : Unit \vdash_d \Lambda X. \lambda(x : X).x = \Lambda X. \lambda(x : X).u[X](x)} \Lambda \text{ cong}}{\frac{u : Unit \vdash_d tt =_{\forall X. X \rightarrow X} u}{\vdash_d : \forall(u : Unit).tt =_{\forall X. X \rightarrow X} u} \text{ Intros}} \text{ Eta}$$

2.7.7 OSum Free Theorems

Try to prove

$$\begin{aligned} & \vdash_d \forall A B. \\ & \quad \forall(y : \text{OSum}). \\ & \quad \forall(f : U(\forall X. \text{Case } X * \text{OSum} \rightarrow FX)). \\ & \quad \forall(k \ k' : U(A \rightarrow B)). \\ & \quad \text{newcase}_A \sigma; x \leftarrow (!f)[A]\sigma y; (!k)x \\ & \quad = \\ & \quad \text{newcase}_A \sigma; x \leftarrow (!f)[A]\sigma y; (!k')x \end{aligned}$$

The intuition is?: Any term f can't use the fresh case to extract a value from OSum. So f is effectively a term of type $U(\forall X. FX)$? After you force f and apply it to $A, (!f)[A]$, you get an effectful value x of A . Then for any two possible continuations of type $A \rightarrow B$, applying the continuations to x yields related results?

$$\frac{\dots \vdash_d \text{newcase}_A \sigma; x \leftarrow (!f)[A]\sigma y; (!k)x = \text{newcase}_A \sigma; x \leftarrow (!f)[A]\sigma y; (!k')x}{\vdash_d \forall \dots \text{newcase}_A \sigma; x \leftarrow (!f)[A]\sigma y; (!k)x = \text{newcase}_A \sigma; x \leftarrow (!f)[A]\sigma y; (!k')x} \text{ Intros}$$

2.7.8 cruft

Looking for free theorems for types containing our separating connectives, OSum, and Case. From the gradual parametricity paper: given

$$\vdash M : \forall^\nu X. ? \rightarrow X$$

and $\vdash V : ?$ then

$$\text{unseal}_X(M\{X \cong A\}V)\text{true}$$

either diverges or errors.

In CBPV OSum:

$$\vdash_c M : \forall X. \text{Case} X \multimap (\text{OSum} \rightarrow FX)$$

should be uninhabited. If error \mathcal{U} was added to the language, then

$$A; \sigma : \text{Case} A * d : \text{OSum} \vdash_c (M[A]@ \sigma) d : FX$$

should always error.

$$A; \sigma : \text{Case} A * d : \text{OSum}; \cdot \vdash_d \forall (M : X. \text{Case} X \multimap (\text{OSum} \rightarrow FX)). (M[A]@ \sigma) d =_{FX} \mathcal{U}$$

Seems like we have to state this property with the context loaded since we dont have a freshness quantifier. Would we want something like?:

$$\vdash_d \forall A. \mathcal{N}(\sigma : \text{Case} A). \forall (d : \text{OSum}) \forall (M : X. \text{Case} X \multimap (\text{OSum} \rightarrow FX)). (M[A]@ \sigma) d =_{FX} \mathcal{U}$$

in Fresh Logic [CITE], Gabbay decomposes \mathcal{N} into

$$\mathcal{N}a. \phi(a) := \exists S \in \text{Fin} \mathbf{A}. \forall a \notin S. \phi(a)$$