1 Type theory

1.1 Syntax

Expression e, A, B ::=	x	variable
	$\mid \mathsf{Type}$	type of types
	$\mid \Pi x : A \,.\, B$	product[1]
	$\mid \mathsf{Eq}_A(e_1, e_2)$	strict equality
	$\mid \mathsf{refl}_A\, e$	reflexivity
	$\mid \lambda x : A \,.\, B \,.\, e$	λ-abstraction[2]
	$\mid e_1 \mathbin{@^{x:A.B}} e_2$	application[3]
Context Γ ::=	$\cdot$	empty context
	$\mid \Gamma, x : A$	context extension

Note that both application and λ-abstraction are tagged with full typing information.

1.2 Judgements

$\Gamma\ \mathsf{ctx}$	Γ is a context
$\Gamma \vdash e : A$	e has type A
$\Gamma \vdash e_1 = e_2 : A$	e_1 and e_2 of type A are equal

1.3 Inference rules

1.3.1 Contexts

The set of variables bound by Γ is denoted by $|\Gamma|$.

$$\textsc{ctx-empty}\quad \frac{}{\cdot\ \mathsf{ctx}} \qquad\qquad \textsc{ctx-extend}\quad \frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash A : \mathsf{Type} \qquad x \notin |\Gamma|}{(\Gamma, x : A)\ \mathsf{ctx}}$$

1.3.2 Types

$$\textsc{type-type}\quad \frac{}{\Gamma \vdash \mathsf{Type} : \mathsf{Type}} \qquad\qquad \textsc{type-pi}\quad \frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma, x : A \vdash B : \mathsf{Type}}{\Gamma \vdash (\Pi x : A \,.\, B) : \mathsf{Type}}$$

$$\textsc{type-eq}\quad \frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma \vdash e_1 : A \qquad \Gamma \vdash e_2 : A}{\Gamma \vdash \mathsf{Eq}_A(e_1, e_2) : \mathsf{Type}}$$

[3] x is bound in B
[3] x is bound in B and e
[3] x is bound in B

1.3.3 Terms

$$\textsc{term-var}\quad \frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}
\qquad
\textsc{term-refl}\quad \frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma \vdash e : A}{\Gamma \vdash (\mathsf{refl}_A\, e) : \mathsf{Eq}_A(e, e)}$$

$$\textsc{term-fun}\quad \frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma, x : A \vdash B : \mathsf{Type} \qquad \Gamma, x : A \vdash e : B}{\Gamma \vdash (\lambda x : A \,.\, B \,.\, e) : \Pi x : A \,.\, B}$$

$$\textsc{term-app}\quad \frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma, x : A \vdash B : \mathsf{Type} \qquad \Gamma \vdash e_1 : \Pi x : A \,.\, B \qquad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 \mathbin{@^{x:A.B}} e_2 : \{e_2/x\}B}$$

1.3.4 Equality

$$\textsc{eq-reflection}\quad \frac{\Gamma \vdash e : \mathsf{Eq}_A(e_1, e_2)}{\Gamma \vdash e_1 = e_2 : A}
\qquad
\textsc{eq-type}\quad \frac{\Gamma \vdash e : A \qquad \Gamma \vdash A = B : \mathsf{Type}}{\Gamma \vdash e : B}$$

$$\textsc{eq-eq}\quad \frac{\Gamma \vdash e_1 = e_2 : A \qquad \Gamma \vdash A = B : \mathsf{Type}}{\Gamma \vdash e_1 = e_2 : B}$$

$$\textsc{eq-subst}\quad \frac{\Gamma, x : A, y : A \vdash B : \mathsf{Type} \qquad \Gamma, z : A \vdash e : \{z/x, z/y\}B \qquad \Gamma \vdash e_1 = e_2 : A}{\Gamma \vdash \{e_1/z\}e : \{e_1/x, e_2/y\}B}$$

1.3.5 Congruence rules

$$\textsc{cong-pi}\quad \frac{\Gamma \vdash A = A' : \mathsf{Type} \qquad \Gamma, x : A \vdash B = B' : \mathsf{Type}}{\Gamma \vdash (\Pi x : A \,.\, B) = (\Pi x : A' \,.\, B') : \mathsf{Type}}$$

$$\textsc{cong-refl}\quad \frac{\Gamma \vdash A = A' : \mathsf{Type} \qquad \Gamma \vdash e = e' : A}{\Gamma \vdash \mathsf{refl}_A\, e = \mathsf{refl}_{A'}\, e' : \mathsf{Eq}_A(e, e)}$$

$$\textsc{cong-eq}\quad \frac{\Gamma \vdash A = A' : \mathsf{Type} \qquad \Gamma \vdash e_1 = e_1' : A \qquad \Gamma \vdash e_2 = e_2' : A}{\Gamma \vdash (\mathsf{Eq}_A(e_1, e_2)) = (\mathsf{Eq}_{A'}(e_1', e_2')) : \mathsf{Type}}$$

2 The meta language

Value v ::=	x	variable
	$\mid (\Gamma \vdash e : A)$	judgement
Computation c ::=	$\mathtt{val}\ v$	value
	$\mid \mathtt{let}\ x = c_1\ \mathtt{in}\ c_2$	binding
	$\mid \mathtt{Type}$	type of types
	$\mid \mathtt{forall}\ x\,,v$	product
	$\mid v_1\ \mathtt{==}\ v_2$	strict equality
	$\mid \mathtt{refl}\ v$	reflexivity
	$\mid \mathtt{lambda}\ x\,.\,v$	λ-abstraction
	$\mid v_1\ \mathtt{@}\ v_2$	application

3 Operational semantics of the kernel

eval-Type

$$\frac{}{\mathsf{Type} \Longrightarrow \mathsf{Type} : \mathsf{Type}\ [\cdot]}$$

eval-Var

$$\frac{\mathsf{fresh}\ \alpha}{x \Longrightarrow x : \alpha\ [\alpha :_x \mathsf{Type}, x : \alpha]}$$

eval-Prod

$$\frac{c_1 \Longrightarrow e_1 : A_1\ [\Gamma_1] \qquad c_2 \Longrightarrow e_2 : A_2\ [\Gamma_2] \qquad \Gamma_2 \rightsquigarrow (\Gamma_2', x :_\emptyset B)}{\Pi x : c_1 . c_2 \Longrightarrow \Pi x : e_1 . e_2 : \mathsf{Type} \begin{bmatrix} \Gamma_1 \bowtie \Gamma_2', \\ \_ :_{\epsilon_1} \mathsf{Eq}_{\mathsf{Type}}(A_1, \mathsf{Type}), \\ \_ : \Pi x : B . (\mathsf{Eq}_{\mathsf{Type}}(A_2, \mathsf{Type})), \\ \epsilon_1 : \mathsf{Eq}_{\mathsf{Type}}(e_1, B) \end{bmatrix}}$$

eval-Eq

$$\frac{c_1 \Longrightarrow e_1 : A_1\ [\Gamma_1] \qquad c_2 \Longrightarrow e_2 : A_2\ [\Gamma_2] \qquad \mathsf{fresh}\ \alpha}{\mathsf{Eq}(c_1, c_2) \Longrightarrow \mathsf{Eq}_\alpha(e_1, e_2) : \mathsf{Type}\ [\Gamma_1 \bowtie \Gamma_2, \alpha :_{\epsilon_1, \epsilon_2} \mathsf{Type}, \epsilon_1 : \mathsf{Eq}_{\mathsf{Type}}(\alpha, A_1), \epsilon_2 : \mathsf{Eq}_{\mathsf{Type}}(\alpha, A_2)]}$$

eval-Refl

$$\frac{c \Longrightarrow e : A\ [\Gamma]}{\mathsf{refl}\ c \Longrightarrow \mathsf{refl}_A\ e : (\mathsf{Eq}_A(e, e))\ [\Gamma]}$$

eval-Fun

$$\frac{c \Longrightarrow e : A\ [\Gamma] \qquad \Gamma \rightsquigarrow (\Gamma', x :_\emptyset B)}{(\lambda x . c) \Longrightarrow (\lambda x : B . e) : (\Pi x : B . A)\ [\Gamma']}$$

eval-App

$$\frac{c_1 \Longrightarrow e_1 : A_1\ [\Gamma_1] \qquad c_2 \Longrightarrow e_2 : A_2\ [\Gamma_2] \qquad \mathsf{fresh}\ \alpha}{c_1\ @\ c_2 \Longrightarrow (e_1\ @^{x:\alpha.\beta @^{\_:\alpha.\mathsf{Type}} x}\ e_2) : (\beta\ @^{\_:\alpha.\mathsf{Type}}\ e_2) \begin{bmatrix} \Gamma_1 \bowtie \Gamma_2, \\ \alpha :_{\epsilon_1, \epsilon_2, \beta} \mathsf{Type}, \\ \beta :_{\epsilon_1} \alpha \to \mathsf{Type}, \\ \epsilon_1 : \mathsf{Eq}_{\mathsf{Type}}(A_1, \Pi x : \alpha . \beta\ @^{\_:\alpha.\mathsf{Type}}\ x), \\ \epsilon_2 : \mathsf{Eq}_{\mathsf{Type}}(\alpha, A_2) \end{bmatrix}}$$