1 Syntax

$e, \; p$	$::=$	
	$\vert$	x
	$\vert$	$<>$
	$\vert$	$\lambda^{\Pi} x : A.\, e$
	$\vert$	$\lambda^{\forall} x : A.\, e$
	$\vert$	$e_1 \; e_2$
	$\vert$	$e_1\{e_2\}$
	$\vert$	$\textbf{refl}\; e$
	$\vert$	$\textbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)$
	$\vert$	$\textbf{let}\; x : A = e_1 \;\textbf{in}\; e_2$
	$\vert$	$\widehat{x}[\sigma]$
	$\vert$	_
$A, \; B, \; C, \; D$	$::=$	
	$\vert$	1
	$\vert$	$\Pi x : A.\, B$
bind x in B		
	$\vert$	$\forall x : A.\, B$
bind x in B		
	$\vert$	$e_1 = e_2 : A$
	$\vert$	$\textbf{let}\; x : A = e \;\textbf{in}\; B$

2 Well-formedness

To formulate the completeness properties (at some point in the future), we need to restrict the 'implicit' binders $\lambda^{\forall}$ and $\forall$ to bind variables that are at safe positions in the body. For this purpose, we define the well-formedness relation.

Here V is a set of variables, sV is a set of variables that need to be in the safe positions. Overall, we wish to guarantee that after normalization, substitution of safe variables does not produce new redexes.

The constructor forms are well-formed by congruence: if their components are well-formed. Notice that $\lambda^{\forall}$ and $\forall$ require the bound variable to be safe in the body.

The eliminator forms are well-formed if their components are well-formed and one of the following conditions hold:

- they do not contain variables that are required to be safe
- they are *inert* (see below), which implies that their outer structure withstands reduction and substitution of safe variables.

$\boxed{V\,;sV \vdash e\,\mathbf{WF}}$

$$\frac{x \in V}{V\,;sV \vdash x\,\mathbf{WF}}\quad\textsc{WfWFVar}$$

$$\frac{x \in sV}{V\,;sV \vdash x\,\mathbf{WF}}\quad\textsc{WfWFSVar}$$

$$\frac{}{V\,;sV \vdash \widehat{x}[\sigma]\,\mathbf{WF}}\quad\textsc{WfWFMVar}$$

$$\frac{}{V\,;sV \vdash <>\,\mathbf{WF}}\quad\textsc{WfWFUnit}$$

$$\frac{V\,;sV \vdash A\,\mathbf{WF} \quad V,x\,;sV \vdash e\,\mathbf{WF}}{V\,;sV \vdash \lambda^{\Pi}x : A.\,e\,\mathbf{WF}}\quad\textsc{WfWFLam}$$

$$\frac{V\,;sV \vdash A\,\mathbf{WF} \quad V\,;sV,x \vdash e\,\mathbf{WF}}{V\,;sV \vdash \lambda^{\forall}x : A.\,e\,\mathbf{WF}}\quad\textsc{WfWFPLam}$$

$$\frac{V\,;sV \vdash e\,\mathbf{WF}}{V\,;sV \vdash \mathbf{refl}\,e\,\mathbf{WF}}\quad\textsc{WfWFRefl}$$

$$\frac{V\,;\cdot \vdash e_1\,\mathbf{WF} \quad V\,;\cdot \vdash e_2\,\mathbf{WF}}{V\,;sV \vdash e_1\,e_2\,\mathbf{WF}}\quad\textsc{WfWFAppNoS}$$

$$\frac{V\,;sV \vdash e_1\,\mathbf{WF} \quad V\,;sV \vdash e_2\,\mathbf{WF} \quad sV \vdash (e_1\,e_2)\,\mathbf{INERT}}{V\,;sV \vdash e_1\,e_2\,\mathbf{WF}}\quad\textsc{WfWFApp}$$

$$\frac{\begin{array}{l}V\,;\cdot \vdash e_1\,\mathbf{WF} \quad V\,;\cdot \vdash e_2\,\mathbf{WF} \quad V\,;\cdot \vdash p\,\mathbf{WF} \quad V\,;\cdot \vdash e\,\mathbf{WF}\\ V\,;\cdot \vdash A\,\mathbf{WF} \quad V,x\,;\cdot \vdash B\,\mathbf{WF}\end{array}}{V\,;sV \vdash \mathbf{subst}\,(p : e_1 = e_2 : A, x.\,B, e)\,\mathbf{WF}}\quad\textsc{WfWFSubstNoS}$$

$$\frac{\begin{array}{l} V\,;sV \vdash e_1\,\mathbf{WF} \quad V\,;sV \vdash e_2\,\mathbf{WF} \quad V\,;sV \vdash p\,\mathbf{WF} \quad V\,;sV \vdash e\,\mathbf{WF} \\ V\,;sV \vdash A\,\mathbf{WF} \quad V,x\,;sV \vdash B\,\mathbf{WF} \\ sV \vdash \mathbf{subst}\,(p : e_1 = e_2 : A, x.\,B, e)\,\mathbf{INERT} \end{array}}{V\,;sV \vdash \mathbf{subst}\,(p : e_1 = e_2 : A, x.\,B, e)\,\mathbf{WF}}\ \textsc{WfWFSubst}$$

$$\frac{\begin{array}{l} V\,;sV \vdash e_1\,\mathbf{WF} \quad V\,;sV \vdash A\,\mathbf{WF} \\ V,x\,;sV \vdash e_2\,\mathbf{WF} \end{array}}{V\,;sV \vdash \mathbf{let}\,x : A = e_1\,\mathbf{in}\,e_2\,\mathbf{WF}}\ \textsc{WfWFLetNoS}$$

The types are not eliminated, so they behave as constructor forms: they are well-formed by congruence.

$\boxed{V\,;sV \vdash A\,\mathbf{WF}}$

$$\frac{}{V\,;sV \vdash 1\,\mathbf{WF}}\ \textsc{WfTWFUnit}$$

$$\frac{V\,;sV \vdash A\,\mathbf{WF} \quad V,x\,;sV \vdash B\,\mathbf{WF}}{V\,;sV \vdash \Pi x : A.\,B\,\mathbf{WF}}\ \textsc{WfTWFPi}$$

$$\frac{V\,;sV \vdash A\,\mathbf{WF} \quad V\,;sV,x \vdash B\,\mathbf{WF}}{V\,;sV \vdash \forall x : A.\,B\,\mathbf{WF}}\ \textsc{WfTWFForall}$$

$$\frac{V\,;sV \vdash A\,\mathbf{WF} \quad V\,;sV \vdash e_1\,\mathbf{WF} \quad V\,;sV \vdash e_2\,\mathbf{WF}}{V\,;sV \vdash e_1 = e_2 : A\,\mathbf{WF}}\ \textsc{WfTWFEq}$$

$$\frac{\begin{array}{l} V\,;sV \vdash A\,\mathbf{WF} \quad V\,;sV \vdash e\,\mathbf{WF} \\ V,x\,;sV \vdash B\,\mathbf{WF} \end{array}}{V\,;sV \vdash \mathbf{let}\,x : A = e\,\mathbf{in}\,B\,\mathbf{WF}}\ \textsc{WfTWFLet}$$

The variables are always inert. The constructor forms are always inert. The eliminator forms forbid their eliminated component to be a safe variable or a constructor form, and require it to be inert. This way, if we assume that the term is well-typed, the eliminated component must be either an unsafe variable or another inert eliminator form.

$\boxed{sV \vdash e\,\mathbf{INERT}}$

$$\frac{}{sV \vdash x\,\mathbf{INERT}}\ \textsc{WfInertVar}$$

$$\frac{}{sV \vdash \texttt{<>}\,\mathbf{INERT}}\ \textsc{WfInertUnit}$$

$$\frac{}{sV \vdash \lambda^{\Pi} x : A.\,e\,\mathbf{INERT}}\ \textsc{WfInertLam}$$

$$\frac{}{sV \vdash \lambda^{\forall} x : A.\,e\,\mathbf{INERT}}\ \textsc{WfInertPLam}$$

$$\frac{}{sV \vdash \mathbf{refl}\,e\,\mathbf{INERT}}\ \textsc{WfInertRefl}$$

$$\frac{x \notin sV}{sV \vdash x\,e\,\mathbf{INERT}}\ \textsc{WfInertAppVar}$$

$$\frac{sV \vdash e_1\, e_2\ \textbf{INERT}}{sV \vdash (e_1\, e_2)\, e_3\ \textbf{INERT}}\ \textsc{WfInertAppApp}$$

$$\frac{sV \vdash \textbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)\ \textbf{INERT}}{sV \vdash (\textbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e))\, e_3\ \textbf{INERT}}\ \textsc{WfInertAppSubst}$$

$$\frac{y \notin sV}{sV \vdash \textbf{subst}\,(y : e_1 = e_2 : A, x.\, B, e)\ \textbf{INERT}}\ \textsc{WfInertSubstVar}$$

$$\frac{sV \vdash e_3\, e_4\ \textbf{INERT}}{sV \vdash \textbf{subst}\,((e_3\, e_4) : e_1 = e_2 : A, x.\, B, e)\ \textbf{INERT}}\ \textsc{WfInertSubstApp}$$

$$\frac{sV \vdash \textbf{subst}\,(p : e_0 = e_1 : A_0, x.\, B_0, e_2)\ \textbf{INERT}}{sV \vdash \textbf{subst}\,(\textbf{subst}\,(p : e_0 = e_1 : A_0, x.\, B_0, e_2) : e_3 = e_4 : A_1, y.\, B_1, e_5)\ \textbf{INERT}}\ \textsc{WfInertSubstSubst}$$

3 Reduction

In the reduction, we first reduce the components of terms (congruence rules). Then, if the outer structure is an eliminator form, there are several rules where something interesting happens: $\boxed{e_1 \rightsquigarrow e_2}$

$$\frac{}{(\lambda^{\Pi} x : A.\, e_0)\ e \rightsquigarrow \textbf{let}\ x : A = e\ \textbf{in}\ e_0}\ \textsc{RedRedAppLam}$$

$$\frac{}{(\lambda^{\forall} x : A.\, e_0)\{e\} \rightsquigarrow \textbf{let}\ x : A = e\ \textbf{in}\ e_0}\ \textsc{RedRedAppPLam}$$

$$\frac{cE\ \textbf{is safe}}{\textbf{let}\ x : A = e\ \textbf{in}\ cE[x] \rightsquigarrow \textbf{let}\ x : A = e\ \textbf{in}\ cE[e]}\ \textsc{RedRedLRedSafe}$$

$$\frac{e\ \textbf{is neutral}}{\textbf{let}\ x : A = e\ \textbf{in}\ cE[x] \rightsquigarrow \textbf{let}\ x : A = e\ \textbf{in}\ cE[e]}\ \textsc{RedRedLRedNeut}$$

$$\frac{e\ \textbf{is ground}}{\textbf{let}\ x : A = e\ \textbf{in}\ cE[x] \rightsquigarrow \textbf{let}\ x : A = e\ \textbf{in}\ cE[e]}\ \textsc{RedRedLRedGr}$$

$$\frac{x \notin \textbf{fv}\ e_2 \qquad A, e_1\ \textbf{are ground}}{\textbf{let}\ x : A = e_1\ \textbf{in}\ e_2 \rightsquigarrow e_2}\ \textsc{RedRedLNotin}$$

(Here cE is an evaluation context, i.e. a term with a hole in a term position; cE **is safe** means that the hole is in a safe position)

The invariant that we are preserving is that

- reduction does not eliminate metavariables
- reduction preserves safety of the metavariables