1 Syntax

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

2 Safety

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, including normal variables (x) and safe variables ($\underline{x}$), whose positions are restricted. 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.

$\boxed{V \vdash e\ \mathbf{OK}}$

$$\frac{x \in V}{V \vdash x\ \mathbf{OK}}\ \textsc{OkOKVar}$$

$$\frac{}{V \vdash \widehat{x}[\sigma]\ \mathbf{OK}}\ \textsc{OkOKMVar}$$

$$\frac{}{V \vdash \texttt{<>}\ \mathbf{OK}}\ \textsc{OkOKUnit}$$

$$\frac{V \vdash A\ \mathbf{OK} \quad V, x \vdash e\ \mathbf{OK}}{V \vdash \lambda^{\Pi} x : A.\, e\ \mathbf{OK}}\ \textsc{OkOKLam}$$

$$\frac{V \vdash A\ \mathbf{OK} \quad V, \underline{x} \vdash e\ \mathbf{OK}}{V \vdash \lambda^{\forall} \underline{x} : A.\, e\ \mathbf{OK}}\ \textsc{OkOKPLam}$$

$$\frac{V \vdash e\ \mathbf{OK}}{V \vdash \mathbf{refl}\ e\ \mathbf{OK}}\ \textsc{OkOKRefl}$$

$$\frac{\begin{array}{l} e_1, e_2\ \mathbf{are\ ground} \\ V \vdash e_1\ \mathbf{OK} \quad V \vdash e_2\ \mathbf{OK} \end{array}}{V \vdash e_1\, e_2\ \mathbf{OK}}\ \textsc{OkOKAppNoS}$$

$$\frac{V \vdash e_1\ \mathbf{OK} \quad V \vdash e_2\ \mathbf{OK} \quad (e_1\, e_2)\ \mathbf{INERT}}{V \vdash e_1\, e_2\ \mathbf{OK}}\ \textsc{OkOKApp}$$

$$\frac{\begin{array}{l} V \vdash e_1\ \mathbf{OK} \quad V \vdash e_2\ \mathbf{OK} \quad V \vdash p\ \mathbf{OK} \quad V \vdash e\ \mathbf{OK} \\ V \vdash A\ \mathbf{OK} \quad V, x \vdash B\ \mathbf{OK} \\ e_1, e_2, p, e, A, B\ \mathbf{are\ ground} \end{array}}{V \vdash \mathbf{subst}\ (p : e_1 = e_2 : A, x.\, B, e)\ \mathbf{OK}}\ \textsc{OkOKSubstNoS}$$

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

$$\frac{V \vdash e_1\ \mathbf{OK} \quad V \vdash A\ \mathbf{OK} \quad V, x \vdash e_2\ \mathbf{OK}}{V \vdash \mathbf{let}\, x : A = e_1\ \mathbf{in}\ e_2\ \mathbf{OK}}\ \textsc{OkOKLetNoS}$$

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

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.

$\boxed{e\ \mathbf{INERT}}$

$$\frac{}{x\ \mathbf{INERT}}\ \textsc{OkInertVar}$$

$$\frac{}{\underline{x}\ \mathbf{INERT}}\ \textsc{OkInertSVar}$$

$$\frac{}{<>\ \mathbf{INERT}}\ \textsc{OkInertUnit}$$

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

$$\frac{}{\lambda^{\forall} \underline{x} : A.\, e\ \mathbf{INERT}}\ \textsc{OkInertPLam}$$

$$\frac{}{\mathbf{refl}\ e\ \mathbf{INERT}}\ \textsc{OkInertRefl}$$

$$\frac{}{x\ e\ \mathbf{INERT}}\ \textsc{OkInertAppVar}$$

$$\frac{e_1\ e_2\ \mathbf{INERT}}{(e_1\ e_2)\ e_3\ \mathbf{INERT}}\ \textsc{OkInertAppApp}$$

$$\frac{\mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)\ \mathbf{INERT}}{(\mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e))\ e_3\ \mathbf{INERT}}\ \textsc{OkInertAppSubst}$$

$$\frac{}{\mathbf{subst}\,(y : e_1 = e_2 : A, x.\, B, e)\ \mathbf{INERT}}\ \textsc{OkInertSubstVar}$$

$$\frac{e_3\ e_4\ \mathbf{INERT}}{\mathbf{subst}\,((e_3\ e_4) : e_1 = e_2 : A, x.\, B, e)\ \mathbf{INERT}}\ \textsc{OkInertSubstApp}$$

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

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: $\boxed{e_1 \rightsquigarrow e_2}$

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

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

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

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

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

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

(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:

- reduction does not eliminate metavariables and the safe variables;
- reduction preserves safety of the metavariables and the safe variables.

3.1 Properties

Lemma 1 (Reduction preserves intertness)**.**

Lemma 2 (Reduction preserves safety)**.** *If* $V \vdash e_0\ \textbf{OK}$ *and* $e_0 \rightsquigarrow e_1$ *then* $V \vdash e_1\ \textbf{OK}$*. If* $V \vdash A_0\ \textbf{OK}$ *and* $A_0 \rightsquigarrow A_1$ *then* $V \vdash A_1\ \textbf{OK}$*.*

Proof. By induction on the reduction relation.

- If the last step of the reduction is a congruence rule on a constructor form (RedRedLamA, RedRedLame, RedRedPLamA, RedRedPLame, RedRedRefl) then the safety of the metavariables is proved by the induction hypothesis and the corresponding safety rule (OkOKLam, OkOKPLam, or OkOKRefl).
- RedRedApp1 Then our term is the application $p_1\, p_2$, and its well-formedness is given by one of the two rules:
  - OkOKAppNoS Then the well-formedness follows immediately by the induction hypothesis applied to $V \vdash p_1\ \textbf{OK}$.
  - OkOKApp Then we will also need to prove $p_1'\, p_2\ \textbf{INERT}$,

□

4 Normalization

Definition 1. *e is normalized if there is no e' such that $e \rightsquigarrow e'$.*

Definition 2. *e' is a normal form of e if $e \rightsquigarrow^* e'$ and e' is normalized.*

5 Well-formedness

We define well-formedness of types and terms in the standard congruent way. The interesting part is the base case for the metavariables (WfWFAVar),

$\boxed{\Gamma \,;\, M\Gamma \vdash e}$

$$\frac{x : A \in \Gamma}{\Gamma \,;\, M\Gamma \vdash x}\ \textsc{WfWFVar}$$

$$\frac{\Gamma \,;\, M\Gamma \vdash A \quad \Gamma, \underline{x} : A \,;\, M\Gamma \vdash e}{\Gamma \,;\, M\Gamma \vdash \lambda^{\forall} \underline{x} : A.\, e}\ \textsc{WfWFPLam}$$

$$\frac{(\Gamma' \vdash \widehat{x} \Leftarrow A) \in M\Gamma \quad \Gamma' \subseteq \Gamma \quad \Gamma \,;\, M\Gamma \vdash \sigma : \Gamma'}{\Gamma \,;\, M\Gamma \vdash \widehat{x}[\sigma]}\ \textsc{WfWFAVar}$$

6 Metavariables and metasubstitution

Definition 3. *Metasubstitution $M\sigma$ is a mapping from metavariables to terms. The specification $\Gamma \,;\, M\Gamma_2 \vdash M\sigma : M\Gamma_1$ states that*

7 Conversion

Definition 4. *$\Gamma \,;\, M\Gamma \vdash e_0 \equiv e_1$ is defined as alpha-equivalence transitively closed under reduction.*

8 Unification

Definition 5. *Suppose that two terms e_0 and e_1 are well-typed in the context Γ and metacontext $M\Gamma$, i.e., $\Gamma \,;\, M\Gamma \vdash e_0$ and $\Gamma \,;\, M\Gamma \vdash e_1$. Then the metasubstitution $M\sigma_0$ is a unifier of e_0 and e_1 if*

- *$\Gamma \,;\, M\Gamma_0 \vdash M\sigma_0 : M\Gamma$ and*
- *$\Gamma \,;\, M\Gamma_0 \vdash [M\sigma_0]e_0 \equiv [M\sigma_0]e_1$.*

The unifier $M\sigma_0$ is the most general if for any other unifier $\Gamma \,;\, M\Gamma_1 \vdash M\sigma_1 : M\Gamma$ of e_0 and e_1, there exists a metasubstitution $\Gamma \,;\, M\Gamma_1 \vdash M\tau : M\Gamma_0$ such that $; M\Gamma_1 \vdash M\sigma_1 \equiv M\tau \circ M\sigma_0$.

A Appendix

$\boxed{e_1 \rightsquigarrow e_2}$

$$\frac{A_0 \rightsquigarrow A_1}{\lambda^{\Pi} x : A_0.\, e \rightsquigarrow \lambda^{\Pi} x : A_1.\, e}\ \textsc{RedRedLamA}$$

$$\frac{e_0 \rightsquigarrow e_1}{\lambda^{\Pi} x : A.\, e_0 \rightsquigarrow \lambda^{\Pi} x : A.\, e_1}\ \textsc{RedRedLamE}$$

$$\frac{A_0 \rightsquigarrow A_1}{\lambda^{\forall} \underline{x} : A_0.\, e \rightsquigarrow \lambda^{\forall} \underline{x} : A_1.\, e}\ \textsc{RedRedPLamA}$$

$$\frac{e_0 \rightsquigarrow e_1}{\lambda^{\forall} \underline{x} : A.\, e_0 \rightsquigarrow \lambda^{\forall} \underline{x} : A.\, e_1}\ \textsc{RedRedPLamE}$$

$$\frac{e_0 \rightsquigarrow e_1}{\mathbf{refl}\, e_0 \rightsquigarrow \mathbf{refl}\, e_1}\ \textsc{RedRedRefl}$$

$$\frac{e_0 \rightsquigarrow e_1}{e_0\ e \rightsquigarrow e_1\ e}\ \textsc{RedRedApp1}$$

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

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

$$\frac{p_0 \rightsquigarrow p_1}{\mathbf{subst}\,(p_0 : e_1 = e_2 : A, x.\, B, e) \rightsquigarrow \mathbf{subst}\,(p_1 : e_1 = e_2 : A, x.\, B, e)}\ \textsc{RedRedSubstP}$$

$$\frac{e_0 \rightsquigarrow e_1}{\mathbf{subst}\,(p : e_0 = e_2 : A, x.\, B, e) \rightsquigarrow \mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)}\ \textsc{RedRedSubstE1}$$

$$\frac{e_0 \rightsquigarrow e_2}{\mathbf{subst}\,(p : e_1 = e_0 : A, x.\, B, e) \rightsquigarrow \mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)}\ \textsc{RedRedSubstE2}$$

$$\frac{A_0 \rightsquigarrow A}{\mathbf{subst}\,(p : e_1 = e_2 : A_0, x.\, B, e) \rightsquigarrow \mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)}\ \textsc{RedRedSubstA}$$

$$\frac{B_0 \rightsquigarrow B}{\mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B_0, e) \rightsquigarrow \mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)}\ \textsc{RedRedSubstB}$$

$$\frac{e_0 \rightsquigarrow e}{\mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e_0) \rightsquigarrow \mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)}\ \textsc{RedRedSubstE}$$

$$\frac{e_0, e_2, e_2, A, B\ \mathbf{are\ ground}}{\mathbf{subst}\,((\mathbf{refl}\, e_0) : e_1 = e_2 : A, x.\, B, e) \rightsquigarrow e}\ \textsc{RedRedSubstRed}$$

$$\frac{A_0 \rightsquigarrow A}{\mathbf{let}\, x : A_0 = e\, \mathbf{in}\, e_0 \rightsquigarrow \mathbf{let}\, x : A = e\, \mathbf{in}\, e_0}\ \textsc{RedRedLetA}$$

$$\frac{e_0 \rightsquigarrow e_1}{\mathbf{let}\, x : A = e_0 \,\mathbf{in}\, e \rightsquigarrow \mathbf{let}\, x : A = e_1 \,\mathbf{in}\, e_2} \quad \textsc{RedRedLetE1}$$

$$\frac{e_0 \rightsquigarrow e_2}{\mathbf{let}\, x : A = e \,\mathbf{in}\, e_0 \rightsquigarrow \mathbf{let}\, x : A = e_1 \,\mathbf{in}\, e_2} \quad \textsc{RedRedLetE2}$$

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

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

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

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

$\boxed{\Gamma\,;M\Gamma \vdash e}$

$$\frac{x : A \in \Gamma}{\Gamma\,;M\Gamma \vdash x} \quad \textsc{WfWFVar}$$

$$\frac{}{\Gamma\,;M\Gamma \vdash \texttt{<>}} \quad \textsc{WfWFUnit}$$

$$\frac{\Gamma\,;M\Gamma \vdash A \qquad \Gamma, x : A\,;M\Gamma \vdash e}{\Gamma\,;M\Gamma \vdash \lambda^{\Pi} x : A.\, e} \quad \textsc{WfWFLam}$$

$$\frac{\Gamma\,;M\Gamma \vdash A \qquad \Gamma, \underline{x} : A\,;M\Gamma \vdash e}{\Gamma\,;M\Gamma \vdash \lambda^{\forall} \underline{x} : A.\, e} \quad \textsc{WfWFPLam}$$

$$\frac{\Gamma\,;M\Gamma \vdash e_1 \qquad \Gamma\,;M\Gamma \vdash e_2}{\Gamma\,;M\Gamma \vdash e_1\, e_2} \quad \textsc{WfWFApp}$$

$$\frac{\Gamma\,;M\Gamma \vdash e_1 \qquad \Gamma\,;M\Gamma \vdash e_2}{\Gamma\,;M\Gamma \vdash e_1\{e_2\}} \quad \textsc{WfWFPApp}$$

$$\frac{\Gamma\,;M\Gamma \vdash e_1}{\Gamma\,;M\Gamma \vdash e_1\{\_\}} \quad \textsc{WfWFPAppU}$$

$$\frac{\Gamma\,;M\Gamma \vdash e}{\Gamma\,;M\Gamma \vdash \mathbf{refl}\, e} \quad \textsc{WfWFRefl}$$

$$\frac{\begin{array}{c}\Gamma\,;M\Gamma \vdash p \qquad \Gamma\,;M\Gamma \vdash e_1 \qquad \Gamma\,;M\Gamma \vdash e_2 \\ \Gamma\,;M\Gamma \vdash A \qquad \Gamma, x : A\,;M\Gamma \vdash B \qquad \Gamma\,;M\Gamma \vdash e\end{array}}{\Gamma\,;M\Gamma \vdash \mathbf{subst}\,(p : e_1 = e_2 : A, x.\, B, e)} \quad \textsc{WfWFSubst}$$

$$\frac{\begin{array}{c}\Gamma\,;M\Gamma \vdash e_1 \qquad \Gamma\,;M\Gamma \vdash A \\ \Gamma, x : A\,;M\Gamma \vdash e_2\end{array}}{\Gamma\,;M\Gamma \vdash \mathbf{let}\, x : A = e_1 \,\mathbf{in}\, e_2} \quad \textsc{WfWFLet}$$

$$\frac{(\Gamma' \vdash \widehat{x} \Leftarrow A) \in M\Gamma \quad \Gamma' \subseteq \Gamma \quad \Gamma\,;M\Gamma \vdash \sigma : \Gamma'}{\Gamma\,;M\Gamma \vdash \widehat{x}[\sigma]}\ \textsc{WfWFAVar}$$