1 Syntax

Definition 1 (Syntax of terms)**.** *We write* term *meaning a computation or a value, denoting it σ or τ.*

$$\begin{array}{llll} \textit{Computations} & X, Y, t, u & ::= & t v \mid \uparrow A \mid \Pi x : A.\, X \mid \forall x : A.\, X \mid \mathsf{force}\, v \mid \mathsf{return}\, v \mid \lambda x : A.\, t \mid \\ & & & \mathsf{let}\; x : A := t \;\mathsf{in}\; u \mid \mathsf{dlet}\; x : A := t \;\mathsf{in}\; u \mid \mathsf{rec}_{\Sigma}^{X}(v, t) \mid \mathsf{rec}_{\mathsf{eq}}^{X}(v, t) \\ \\ \textit{Values} & A, B, v, w & ::= & x \mid \downarrow X \mid \Sigma x : A.\, B \mid \mathsf{refl} \mid \mathsf{eq} A\, v\, w \mid \{t\} \mid \langle v, w \rangle \end{array}$$

We use different non-terminal symbols to emphasize the distinction between type-level terms and term-level terms, which manifests properly in section 9. The upper-case literals represent type-terms, and the lower-case represent term-terms (which can be typed with some type-terms) with one exception: in $\mathsf{let}\; x : A := t \;\mathsf{in}\; u$, u can represent a type-term.

2 Computational form of the terms

Let us consider the term syntax from a different perspective:

Definition 2 (Computational syntax of terms)**.**

$$\begin{array}{llll} \textit{Constructors} & C & ::= & \lambda \mid (,) \mid \mathsf{refl} \mid \mathsf{return} \mid \{\} \\ \\ \textit{Eliminators} & E & ::= & @ \mid \mathsf{rec}_{\Sigma}(,) \mid \mathsf{rec}_{\mathsf{eq}}(,) \mid \mathsf{let} \mid \mathsf{dlet} \mid \mathsf{force} \\ \\ \textit{Neutral Formers} & N & ::= & \downarrow \mid \Pi \mid \forall \mid \uparrow \mid \Sigma \mid x \\ \\ \textit{Formers} & F & ::= & C \mid E \mid N \\ \\ \textit{Abstractor Heads} & \vec{x}^{0} & ::= & . \\ & \vec{x}^{n+1} & ::= & x \,.\, \vec{x}^{n} \\ & \vec{x} & ::= & \vec{x}^{0} \mid \vec{x}^{1} \mid \ldots \\ \\ \textit{Abstractors} & P^{n}, Q^{n} & ::= & \vec{x}^{n} \tau \\ & P, Q & ::= & P^{1} \mid P^{2} \mid \ldots \\ \\ \textit{Terms} & \sigma, \tau, \nu & ::= & F\,(\vec{x}^{\mathsf{ar} F_1} \tau_1) \ldots (\vec{x}^{\mathsf{ar} F_{|F|}} \tau_{|F|}) \end{array}$$

Definition 3 (Arity)**.** *For every term former F we define its arity* $\mathsf{ar}F$ *as the array of integers desctibing its arguments. Integer denotes the number of new binding variables "created" by F that can be used in the corresponding subterm. For brevity, we denote length of* $\mathsf{ar}F$ *as $|F|$.*

F	λ	$(,)$	refl	return	$\{\}$	@	rec_{Σ}	$\mathsf{rec}_{\mathsf{eq}}$	let	dlet	force	$\downarrow$	Π	$\forall$	$\uparrow$	Σ	x
$\mathsf{ar}F$	$[1,0]$	$[0,0]$	$[]$	$[0]$	$[0]$	$[0,0]$	$[0,0,0]$	$[0,0,0]$	$[1,0,0]$	$[1,0,0]$	$[0]$	$[0]$	$[0,1]$	$[0,1]$	$[0]$	$[1,0]$	$[]$
$\lvert F\rvert$	1	2	0	1	1	2	3	3	3	3	1	1	2	2	1	2	0

Ilya: Notice that we rearrange the arguments in λ and let-bindings so that any redex is always an eliminator whose *first* argument is a constructor

It is easy to see that the syntax of *terms* from definition 1 defines the *subset* of terms defined by definition 2. In fact, any *well-typed* term must have a form defined by definition 1. We will use these two representation interchangeably.

3 Alpha-equivalence

Definition 4 (Variable Renaming). *Ilya: todo*

Definition 5 (Alpha-equivalence).

$$\frac{\forall i, P_i \sim_\alpha Q_i}{F\, P_1 \dots P_{|F|} \sim_\alpha F\, Q_1 \dots Q_{|F|}} \qquad \texttt{Ilya: todo}$$

4 Susbtititution

Definition 6 (Substitution). *Ilya: todo*

Lemma 1 (Functionality of Substituion). *Substitution is a functional on the classes of alpha-equivalence.*

Ilya: todo

5 Reduction

First, we define the *redex contraction*.

Definition 7 (Redex Contraction). *Ilya: Defined on α-equivalence classes! We define the top-level redex contraction in the following way:*

- $(\lambda x : \nu.\, \sigma)\tau \rightharpoonup \sigma\{x := \tau\}$
- $\mathsf{let}\ x : \nu := \mathsf{return}\, \sigma\ \mathsf{in}\ \tau \rightharpoonup \tau\{x := \sigma\}$
- $\mathsf{dlet}\ x : \nu := \mathsf{return}\, \sigma\ \mathsf{in}\ \tau \rightharpoonup \tau\{x := \sigma\}$
- $\mathsf{force}\, \{\tau\} \rightharpoonup \tau$
- $\mathsf{rec}^\nu_\Sigma(\langle \tau_1, \tau_2 \rangle, \sigma) \rightharpoonup \sigma\, \tau_1\, \tau_2$
- $\mathsf{rec}^\nu_{\mathsf{eq}}(\mathsf{refl}, \tau) \rightharpoonup \tau$

The terms on the left hand side of $\cdot \rightharpoonup \cdot$ are called redexes.

Notice that any redex from definition 7 is an elimination of a constructor, i.e. a term of the form $E\,(C\, \sigma_1 \dots \sigma_{|C|})\, \tau_2 \dots \tau_{|E|}$ where E and C are "matched". Vice versa, if a term of the form $E\,(C\, \sigma_1 \dots \sigma_{|C|})\, \tau_2 \dots \tau_{|E|}$ is *well-typed*, it is a redex.

Informally, reduction of a term τ is a redex contraction happening in some *subterm* of τ.

Definition 8 (Reduction). *Ilya: Defined on α-equivalence classes!*

$$\frac{\tau \rightharpoonup \tau'}{\tau \to \tau'}\ \textsc{Redex} \qquad \frac{\tau \to \tau'}{F\, P_1 \dots (\vec{x}^{\,\mathsf{ar} F_i} \tau) \dots P_{|F|} \to F\, P_1 \dots (\vec{x}^{\,\mathsf{ar} F_i} \tau') \dots P_{|F|}}\ \textsc{Cong}^F_i$$

Lemma 2 (Substitution preserves reduction). *Ilya: Stated for α-equivalence classes!*

$$\frac{\tau \to \tau'}{\tau\{x := \sigma\} \to \tau'\{x := \sigma\}}$$

Proof. Induction on $\tau \to \tau'$. Substitution is congruent, therefore, the induction goes down to the redexes.

- Suppose that $(\lambda x : \nu.\, \sigma)\sigma' \to \sigma\{x := \sigma'\}$. We need to prove that $(\lambda x : \nu.\, \sigma)\sigma'\{y := \tau\} \to \sigma\{x := \sigma'\}\{y := \tau\}$. We know that $(\lambda x : \nu.\, \sigma)\sigma'\{y := \tau\} = (\lambda x : \nu.\, \sigma\{y := \tau\})(\sigma'\{y := \tau\})$, which reduces to $\sigma\{y := \tau\}\{x := \sigma'\{y := \tau\}\}$. But
$$\sigma\{y := \tau\}\{x := \sigma'\{y := \tau\}\} = \sigma\{x := \sigma'\}\{y := \tau\},$$
assuming that $x \notin \mathsf{FV}(\tau)$, which is guaranteed because the substitution is capture-avoiding.
- The other cases are similar or straightforward

□

6 Normal Form

Using the syntax from definition 2, it is convenient to express computational properties of the term, e.g. being in the normal form (NF).

Definition 9 (Normal Form)**.**

$$\frac{\tau\ \mathsf{ATOM}}{\tau\ \mathsf{NF}} \qquad \frac{\tau_1\ \mathsf{NF}\ \ \dots\ \ \tau_{|C|}\ \mathsf{NF}}{C\ \vec{x}\tau_1 \dots \vec{x}\tau_{|C|}\ \mathsf{NF}} \qquad \frac{\tau_1\ \mathsf{NF}\ \ \dots\ \ \tau_{|N|}\ \mathsf{NF}}{N\ \vec{x}\tau_1 \dots \vec{x}\tau_{|N|}\ \mathsf{ATOM}} \qquad \frac{\tau_1\ \mathsf{ATOM} \qquad \tau_2\ \mathsf{NF}\ \ \dots\ \ \tau_{|E|}\ \mathsf{NF}}{E\ \vec{x}\tau_1 \dots \vec{x}\tau_{|E|}\ \mathsf{ATOM}}$$

The intuition is that (i) normal terms are not reducible; (ii) atomic terms are not reducible and, in addition, do not cause reduction when the eliminators are applied to them.

Although it is easy to see that the terms in normal form are not reducible, the opposite is only true for the well-typed terms:

Proposition 1 (Normal form and irreducibility)**.**

$$\frac{\tau\ \mathsf{NF}}{\nexists \tau', \tau \to \tau'} \qquad \frac{\tau\ \textit{is well-typed} \qquad \nexists \tau', \tau \to \tau'}{\tau\ \mathsf{NF}}$$

Hereafter, we assume all the terms are well typed. `Ilya: Well-typedness is required for the unification and equivalence to be well-founded (otherwise induction is not possible). TODO: normalization (halting)!`

7 Safe Occurrence

Another important property that we express in this syntax is *safe occurrence of the variable*. The judgement $x \in \tau\ \mathsf{OK}$ means x occurs safely in τ.

Ideally, we would like to forbid the situations when *in some normal form* of τ, some instantiation of x generates a new redex. In other words, we would like to ensure that *all normal forms* of τ do not contain $E\, x\, \tau_2 \dots \tau_{|E|}$ as a subterm.

However, this property is undecidable by Rice's theorem: notice that (i) we do not require terms to have types at this stage, thus, the system is Turing complete; (ii) the property is non-trivial; (iii) the property judges about the normal forms and thus, is invariant under "algorithmic equivalence". As it is undecidable, it is impossible to express this judgement using well-founded inference rules (i.e. unambiguously generating finite trees).

Since precise syntactic representation of this property is impossible, we under-approximate this property in the following way:

`Ilya: TODO: add safe occurrence in abstractors`

$$\frac{x \in \tau_1\ \mathsf{OK}\ \ \dots\ \ x \in \tau_{|C|}\ \mathsf{OK}}{x \in C\, \tau_1 \dots \tau_{|C|}\ \mathsf{OK}}\ \textsc{C-Cong} \qquad \frac{x \in \tau_1\ \mathsf{OK}\ \ \dots\ \ x \in \tau_{|N|}\ \mathsf{OK}}{x \in N\, \tau_1 \dots \tau_{|N|}\ \mathsf{OK}}\ \textsc{N-Cong}$$

$$\frac{x \notin \mathsf{FV}(E\, \tau_1 \dots \tau_{|E|})}{x \in E\, \tau_1 \dots \tau_{|E|}\ \mathsf{OK}}\ \text{E-FV} \qquad \frac{x \in \tau_1\ \mathsf{OK}\ \ \dots\ \ x \in \tau_{|E|}\ \mathsf{OK} \qquad \tau_1 \neq x \qquad E\, \tau_1 \dots \tau_{|E|}\ \mathsf{INERT}}{x \in E\, \tau_1 \dots \tau_{|E|}\ \mathsf{OK}}\ \textsc{E-Cong}$$

In the last rule, "$\tau_1 \neq x$" means literal syntactic inequality. Intuitively, "$\tau\ \mathsf{INERT}$" means that τ preserves its top-level structure under the reduction, i.e. the reduction always happens in the subterms of τ but never on the top-level. In fact, the relation we define is a little bit stronger, as it also forbids changing of the structure of the eliminator's first argument. Formally, it is defined as follows:

$$\frac{}{N\,\tau_1 \ldots \tau_{|N|}\ \mathsf{INERT}} \qquad \frac{}{C\,\tau_1 \ldots \tau_{|C|}\ \mathsf{INERT}} \qquad \frac{}{E\,(N\,\sigma_1 \ldots \sigma_{|N|})\,\tau_2 \ldots \tau_{|E|}\ \mathsf{INERT}}$$

$$\frac{E'\,\sigma_1 \ldots \sigma_{|E'|}\ \mathsf{INERT}}{E\,(E'\,\sigma_1 \ldots \sigma_{|E'|})\,\tau_2 \ldots \tau_{|E|}\ \mathsf{INERT}}\ \text{EE-INERT}$$

As a heuristics, it is possible to extend the "Safe Occurrence" property by injecting some of the redex contractions from definition 7 into the inference system. Notice that only non-substituting contractions are allowed because the latter would violate the finiteness of the inference trees.

$$\frac{x \in \tau\ \mathsf{OK}}{x \in \mathsf{force}\,\{\tau\}\ \mathsf{OK}} \qquad \frac{x \in @\,(@\,\sigma\,\tau_1)\,\tau_2\ \mathsf{OK} \qquad x \in \tau'\ \mathsf{OK}}{x \in \mathsf{rec}^{\tau'}_{\Sigma}\,(\langle \tau_1, \tau_2\rangle, \sigma)\ \mathsf{OK}} \qquad \frac{x \in \sigma\ \mathsf{OK} \qquad x \in \tau\ \mathsf{OK}}{x \in \mathsf{rec}^{\sigma}_{\mathsf{eq}}(\mathsf{refl}, \tau)\ \mathsf{OK}}$$

Lemma 3 (Correctness of the definition)**.** *Safe occurrence is defined up to alpha-equivalence.*

Proof. Ilya: todo □

Lemma 4 (Conguence of the safe occurrence)**.**

$$\frac{x \in F\,\vec{x}\sigma_1 \ldots \vec{x}\sigma_{|F|}\ \mathsf{OK} \qquad \textit{Ilya: x is not captured by the abstractors}}{x \in \sigma_1\ \mathsf{OK} \qquad \cdots \qquad x \in \sigma_{|F|}\ \mathsf{OK}}$$

Proof. Trivial induction. □

Lemma 5 (Reduction-Substitution Commutativity)**.** *Ilya: stated for α-equivalence classes*

$$\frac{x \in \sigma\ \mathsf{OK} \qquad \tau\ \mathsf{NF} \qquad \sigma\{x := \tau\} \to \sigma'}{\exists \sigma^*\ s.t.\ \sigma \to \sigma^*\ and\ \sigma^*\{x := \tau\} = \sigma'}$$

Or in the commutative diagram form: if $x \in \sigma\ \mathsf{OK}$ and $\tau\ \mathsf{NF}$ then
$$\begin{tikzcd} \sigma \arrow[r, dashed, "\to"] \arrow[d, "x:=\tau"'] & \sigma^* \arrow[d, dashed, "x:=\tau"] \\ \bullet \arrow[r, "\to"'] & \sigma' \end{tikzcd}$$

Proof. Let us destruct the substitution $\sigma\{x := \tau\}$. Notice that $\sigma \neq x$ because $x\{x := \tau\} = \tau \not\to \cdot$. It means that the substitution is performed by congruence: $\sigma = F\,\sigma_1 \ldots \sigma_{|F|}$ (for some $F \neq x$), and $\sigma\{x := \tau\} = F\,(\sigma_1\{x := \tau\}) \ldots (\sigma_{|F|}\{x := \tau\})$. Notice that $x \in \sigma_i\ \mathsf{OK}$ for $i = 1 \ldots |F|$ by lemma 4.

Induction on $\sigma\{x := \tau\} \to \sigma'$. The reduction step can be justified either by the congruence or the redex contraction.

- If the reduction step is done by congruence, then the required σ^* is of the form $F\,\sigma^*_1 \ldots \sigma^*_{|F|}$ where $\sigma^*_1 \ldots \sigma^*_{|F|}$ are constructed by the straightforward application of the induction hypothesis to $\sigma_1 \ldots \sigma_{|F|}$.

- If the reduction is the top-level redex contraction, then $\sigma\{x := \tau\}$ is a redex, i.e. F is an eliminator E and $\sigma_1\{x := \tau\}$ is formed by a constructor C. Notice that because $x \in E\,\sigma_1 \ldots \sigma_{|E|}\ \mathsf{OK}$, $\sigma_1 \neq x$. Therefore, the substitution $\sigma_1\{x := \tau\}$ is also done by congruence: $\sigma_1 = C\,\zeta_1 \ldots \zeta_{|C|}$ and thus, $\sigma = E\,(C\,\zeta_1 \ldots \zeta_{|C|})\,\sigma_2 \ldots \sigma_{|E|}$.

  Let us destruct $x \in \sigma\ \mathsf{OK}$. Since σ is not inert, either (i) $x \notin \mathsf{FV}(\sigma)$, then the substitution is the identity, and we can take $\sigma^* = \sigma'$); or (ii) one of the "additional" rules is applied to get $x \in \sigma\ \mathsf{OK}$. In all of these three cases, we can perform the same top-level redex contraction to acquire σ^*. This operation commutes with substitution because all it does is restructuring the top-level form of σ without changing the subterms $\zeta_1, \ldots, \zeta_{|C|}, \sigma_2, \ldots, \sigma_{|E|}$, thus, the required property holds. Ilya: to be fair, the beta-reduction also commutes with the substitution, but we still need the inertness so that OK is preserved under reduction.

□

Lemma 6 (Inertness preservation)**.**

$$\frac{\tau\,\mathsf{INERT} \qquad \tau \to \tau'}{\tau'\,\mathsf{INERT}}$$

Proof. Induction on $\tau\,\mathsf{INERT}$. □

Lemma 7 (Reduction preserves safe occurrence)**.**

$$\frac{x \in \tau\,\mathsf{OK} \qquad \tau \to \tau'}{x \in \tau'\,\mathsf{OK}}$$

Proof. Induction on $x \in \tau\,\mathsf{OK}$.

- For C-Cong (N-Cong), we apply the induction hypothesis and C-Cong (N-Cong, resp.).
- For E-FV, notice that the reduction does not increase the set of free variables, and thus, E-FV is applicable after the reduction of one of the τ_i.
- The E-Cong case is a little bit more complicated. Notice that $\tau_1 \not\twoheadrightarrow x$. This is because if τ_1 is an eliminator, it must be inert by EE-Inert. Then we can consider in which τ_i the reduction happened, apply the induction hypothesis and lemma 6.
- For the additional rules, the reduction can be either by congruence (and then we apply the induction hypothesis, lemma 4 and the same rule) or by the top-level redex contraction, and then the required property is exactly one of the premises.

□

8 Equivalence and Unification

Definition 10 (Syntax of algorithmic terms)**.** *Throughout the algorithm, we will use the auxiliary pre-cooked terms, containing some unassigned parts. For this purpose, we extend the syntax of terms (definition 1) by adding the "hatted" unification (existential) variables $\hat{x}$ to the set of values:*

$$\textit{Values} \mathrel{+}= \hat{x}$$

Similarly, we extend the syntax from definition 2 by adding $\hat{x}$ to the Neutral Formers:

$$\textit{Neutral Formers} \mathrel{+}= \hat{x}$$

Notation 1. *To denote that the term is algorithmic, i.e. potentially contains the unification variables, we use π and ρ. If the term does not contain the unification variables it is called* ground *and denoted as σ and τ.*

Definition 11 (Safe algorithmic term)**.** *We say that the algorithmic term ρ is* safe *iff all the unification variables occur safely in it:*

$$\frac{\forall \hat{x},\ \hat{x} \in \rho\,\mathsf{OK}}{\rho\,\mathsf{OK}}$$

Definition 12 (Equivalence)**.** *We define equivalence on ground terms:*

Reduction closure

$$\frac{\tau_1 \rightarrow \tau_1' \qquad \tau_1' \equiv \tau_2}{\tau_1 \equiv \tau_2}\ \textsc{Red-L} \qquad\qquad \frac{\tau_2 \rightarrow \tau_2' \qquad \tau_1 \equiv \tau_2' \qquad \tau_1\,\mathsf{NF}}{\tau_1 \equiv \tau_2}$$

Congruence

$$\frac{P_1 \equiv Q_1 \qquad \ldots \qquad P_{|F|} \equiv Q_{|F|} \qquad F\,P_1 \ldots P_{|F|}\,\mathsf{NF} \qquad F\,Q_1 \ldots Q_{|F|}\,\mathsf{NF}}{F\,P_1 \ldots P_{|F|} \equiv F\,Q_1 \ldots Q_{|F|}}$$

$$\frac{\overrightarrow{x}^n \sigma \equiv \overrightarrow{x}^n \tau}{x.\overrightarrow{x}^n \sigma \equiv x.\overrightarrow{x}^n \tau} \qquad\qquad \frac{\sigma \equiv \tau}{.\sigma \equiv .\tau}$$

Definition 13 (Binding Context)**.** *Binding context is a* set *of variables that have been bound.*

$$B ::= \cdot \ \mid\ B, x$$

The unification (or algorithmic equivalence) judgement is of the form $\varphi; B \vdash \rho \equiv \rho' \dashv \varphi'$ where ρ and ρ' are algorithmic terms (potentially with unassigned variables), φ and φ' are contexts formed by the following grammar.

Definition 14 (Unification Context)**.** *Unification context is a* set *of pairs:*

$$\varphi, \psi ::= \cdot \ \mid\ \varphi, (\hat{v}{:=}\tau)$$

where τ is a ground term.

Definition 15 (Well-formed unification context)**.** *We say that a unification context φ is well-formed if the mapping it represents is a partial function, whose image terms are normal and ground:*

$$\frac{}{\vdash \cdot} \qquad\qquad \frac{\vdash \varphi \qquad (\hat{x}{:=}\cdot) \notin \varphi \qquad \tau\,\mathsf{NF}}{\vdash \varphi, (\hat{x}{:=}\tau)}$$

Definition 16 (Safe unification context)**.** *We say that a unification context Ω is safe under the binding context B ($B \cap \Omega = \emptyset$) if the free variables of Ω do not intersect with B.*

Definition 17 (Application of the well-formed context)**.** *If the unification context Ω is well-formed, We write $[\Omega]\tau$ meaning the application of the partial (substitution) function represented by Ω to the term τ:*

- $[\cdot]\tau = \tau$
- $[\Omega, (\hat{x}{:=}\sigma)]\tau = ([\Omega]\tau)\{\hat{x} := \sigma\}$

Intuitively, when a context is applied to a term, the components of the context are applied to the term one-by-one. This way, the properties holding for a single substitution, can be lifted up to the context application.

Corollary 1 (Context application commutes with the reduction)**.**

$$\frac{\vdash \Omega \qquad \rho\,\mathsf{OK} \qquad [\Omega]\rho \rightarrow \rho'}{\exists \rho^* \ \textit{s.t.}\ \ \rho \rightarrow \rho^* \ \textit{and}\ [\Omega]\rho^* = \rho'}$$

Proof. Induction on $[\Omega]\rho$ using lemma 5. See section 8 for the details: we acquire ρ^* by consequently applying lemma 5 bottom-to-top to construct $\rho_n^*, \ldots, \rho_2^*, \rho^*$. The premises required for lemma 5 hold because $\vdash \Omega$ and $\rho\,\mathsf{OK}$. □

$$\begin{tikzcd}
\rho \arrow[r, dashed, "\to"] \arrow[d, "\hat{x}_1{:=}\tau_1"'] & \rho^* \arrow[d, dashed, "\hat{x}_1{:=}\tau_1"] \\
\bullet \arrow[r, dashed, "\to"] \arrow[d, dotted, no head] & \rho_2^* \arrow[d, dotted, no head] \\
\bullet \arrow[r, dashed, "\to"] \arrow[d, "\hat{x}_n{:=}\tau_n"'] & \rho_n^* \arrow[d, dashed, "\hat{x}_n{:=}\tau_n"] \\
{[\Omega]\rho} \arrow[r, "\to"] & \rho'
\end{tikzcd}$$

Figure 1: Proof scheme

Corollary 2 (Context application preserves reduction)**.**

$$\frac{\vdash \Omega \qquad \rho \to \rho'}{[\Omega]\rho \to [\Omega]\rho'}$$

Proof. Induction on Ω using lemma 2. $\square$

Corollary 3 (Reduction preserves safety)**.** *lemma 7*

$$\frac{\rho\,\mathsf{OK} \qquad \rho \to \rho'}{\rho'\,\mathsf{OK}}$$

Definition 18 (Unification)**.** *The unification algorithm is defined as follows:*

Base rules

$$\frac{(\hat{v}{:=}\cdot) \notin \varphi \qquad \tau\,\mathsf{NF} \qquad B \cap \tau = \emptyset}{B; \varphi \vdash \hat{v} \equiv \tau \dashv \varphi, (\hat{v}{:=}\tau)}\ \textsc{U-Add}$$

$$\frac{(\hat{v}{:=}\tau) \in \varphi \qquad \textit{Ilya: up-to-alpha-equivalence!}}{B; \varphi \vdash \hat{v} \equiv \tau \dashv \varphi}\ \textsc{U-Keep}$$

Reduction closure

$$\frac{\rho_1 \to \rho_1' \qquad \varphi \vdash \rho_1' \equiv \tau_2 \dashv \varphi'}{B; \varphi \vdash \rho_1 \equiv \tau_2 \dashv \varphi'}\ \textsc{Red-L} \qquad \frac{\tau_2 \to \tau_2' \qquad \varphi \vdash \rho_1 \equiv \tau_2' \dashv \varphi' \qquad \rho_1\,\mathsf{NF}}{B; \varphi \vdash \rho_1 \equiv \tau_2 \dashv \varphi'}\ \textsc{Red-R}$$

Congruence

$$\frac{B; \varphi_0 \vdash P_1 \equiv Q_1 \dashv \varphi_1 \ \ \ldots \ \ B; \varphi_{|F|-1} \vdash P_{|F|} \equiv Q_{|F|} \dashv \varphi_{|F|} \qquad F\,P_1 \ldots P_{|F|}\,\mathsf{NF} \qquad F\,Q_1 \ldots Q_{|F|}\,\mathsf{NF}}{B; \varphi_0 \vdash F\,P_1 \ldots P_{|F|} \equiv F\,Q_1 \ldots Q_{|F|} \dashv \varphi_{|F|}}$$

$$\frac{B, x; \varphi \vdash \vec{x}^n \rho \equiv \vec{x}^n \tau \dashv \psi}{B; \varphi \vdash x.\vec{x}^n \rho \equiv x.\vec{x}^n \tau \dashv \psi} \qquad \frac{B; \varphi \vdash \rho \equiv \tau \dashv \psi}{B; \varphi \vdash .\rho \equiv .\tau \dashv \psi}$$

We prove the soundness and completeness of the unification w.r.t. the equality defined above. Intuitively, soundness means that the output context produced by the unification algorithm does not make the terms non-unifiable.

Lemma 8 (Unification soundness)**.**

$$\frac{\vdash \varphi_1 \qquad B; \varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2 \qquad B \cap \varphi_1 = \emptyset}{\vdash \varphi_2 \qquad [\varphi_2]\rho \equiv \tau \qquad \varphi_1 \subseteq \varphi_2 \qquad B \cap \varphi_2 = \emptyset}$$

Proof. Induction on $B; \varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2$.

- $\dfrac{(\hat{v}{:=}\cdot) \notin \varphi_1 \qquad \tau\,\mathsf{NF} \qquad B \cap \tau = \emptyset}{B; \varphi_1 \vdash \hat{v} \equiv \tau \dashv \varphi_1, (\hat{v}{:=}\tau)}$ Then $\rho = \hat{v}$ and $\varphi_2 = \varphi_1, (\hat{v}{:=}\tau)$.
  - It is easy to see that $\vdash \varphi_2$ because $\vdash \varphi_1, (\hat{v}{:=}\tau)$ by definition of the well-formed context (all the required premises are given);
  - $[\varphi_2]\rho = [\varphi_1, (\hat{v}{:=}\tau)]\hat{v} = \tau \equiv \tau$;
  - $\varphi_1 \subseteq \varphi_1, (\hat{v}{:=}\tau)$ by definition.
  - $B \cap \mathsf{FV}(\varphi_1, (\hat{v}{:=}\tau)) = B \cap \mathsf{FV}(\varphi_1) \cup B \cap \mathsf{FV}(\tau)$, which is empty because $B \cap \varphi_1 = \emptyset$ and $B \cap \tau = \emptyset$. Hence $B \cap \varphi_2 = \emptyset$.
- $\dfrac{(\hat{v}{:=}\tau) \in \varphi}{B; \varphi \vdash \hat{v} \equiv \tau \dashv \varphi}$ Then $\rho = \hat{v}$ and $\varphi_1 = \varphi_2 = \varphi$.
  - $\vdash \varphi_2$ because $\varphi_2 = \varphi_1$ and $\vdash \varphi_1$ is in the premises;
  - $[\varphi_2]\hat{v} \equiv \tau$ because $\vdash \varphi_2$ and $(\hat{v}{:=}\tau) \in \varphi_2$ `Ilya: alpha-equivalence`;
  - $\varphi \subseteq \varphi$ trivially.
  - $B \cap \varphi_2 = \emptyset$ because $\varphi_2 = \varphi_1$ and $B \cap \varphi_1 = \emptyset$ is in the premises.
- $\dfrac{\rho \to \rho' \qquad \varphi_1 \vdash \rho' \equiv \tau \dashv \varphi_2}{\varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2}$ Then by the induction hypothesis: $\varphi_1 \subseteq \varphi_2$, $\vdash \varphi_2$, $B \cap \varphi_2 = \emptyset$, and $[\varphi_2]\rho' \equiv \tau$. To prove that $[\varphi_2]\rho \equiv \tau$ we apply Red-L:

  $$\frac{[\varphi_2]\rho \to [\varphi_2]\rho' \qquad [\varphi_2]\rho' \equiv \tau}{[\varphi_2]\rho \equiv \tau}\ \textsc{Red-L}$$

  Here $[\varphi_2]\rho \to [\varphi_2]\rho'$ holds by corollary 2.
- $\dfrac{\tau \to \tau' \qquad \varphi_1 \vdash \rho \equiv \tau' \dashv \varphi_2 \qquad \rho\,\mathsf{NF}}{\varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2}$ Analogously to the previous case.
- $\dfrac{\psi_0 \vdash \rho_1 \equiv \tau_1 \dashv \psi_1 \qquad \ldots \qquad \psi_{|F|-1} \vdash \rho_{|F|} \equiv \tau_{|F|} \dashv \psi_{|F|} \qquad F\,\rho_1 \ldots \rho_{|F|}\,\mathsf{NF} \qquad F\,\tau_1 \ldots \tau_{|F|}\,\mathsf{NF}}{\psi_0 \vdash F\,\rho_1 \ldots \rho_{|F|} \equiv F\,\tau_1 \ldots \tau_{|F|} \dashv \psi_{|F|}}$ Then $\varphi_1 = \psi_0$, $\varphi_2 = \psi_{|F|}$, $\rho = F\,\rho_1 \ldots \rho_{|F|}$, $\tau = F\,\tau_1 \ldots \tau_{|F|}$.

  We can apply the induction hypothesis to the first unification judgement in the premise (i.e. to $\psi_0 \vdash \rho_1 \equiv \tau_1 \dashv \psi_1$) acquiring: $\vdash \psi_1$ and $[\psi_1]\rho_1 \equiv \tau_1$. Then, because $\vdash \psi_1$, we can apply the induction hypothesis to the second premise. Continuing this process, we acquire:
  - $\varphi_1 = \psi_0 \subseteq \cdots \subseteq \psi_{|F|} = \varphi_2$;
  - $\vdash \varphi_2$
  - $[\psi_i]\rho_i \equiv \tau_i$ for $i = 1 \ldots |F|$. Hence, because $\psi_i \subseteq \varphi_2$, $[\varphi_2]\rho_i \equiv \tau_i$, which implies that $[\varphi_2]F\,\rho_1 \ldots \rho_{|F|} \equiv F\,\tau_1 \ldots \tau_{|F|}$

□

Lemma 9 (Unification completeness)**.**

$$\frac{\vdash \Omega \qquad \rho\,\mathsf{OK} \qquad [\Omega]\rho \equiv \tau}{\forall B \vdash \Omega.\ \forall \varphi \subseteq \Omega.\ \exists \psi \subseteq \Omega.\ B; \varphi \vdash \rho \equiv \tau \dashv \psi}$$

Proof. Induction on $[\Omega]\rho \equiv \tau$.

- $\dfrac{[\Omega]\rho \to \rho' \qquad \rho' \equiv \tau}{[\Omega]\rho \equiv \tau}$ By corollary 1, there exists ρ^* s.t. $\rho \to \rho^*$ and $[\Omega]\rho^* = \rho'$.
  By lemma 7, $\rho^*\,\mathsf{OK}$. Then we apply the induction hypothis to Ω, ρ^*, and τ. To acquire $\forall \varphi \subseteq \Omega.\ \exists \psi \subseteq \Omega.\ \varphi \vdash \rho^* \equiv \tau \dashv \psi$, where we can replace ρ^* with ρ by Red-L.
- $\dfrac{\tau \to \tau' \qquad [\Omega]\rho \equiv \tau' \qquad [\Omega]\rho\,\mathsf{NF}}{[\Omega]\rho \equiv \tau}$ We can apply the induction hypothesis to Ω, ρ, and τ' right away to acquire $\forall \varphi \subseteq \Omega.\ \exists \psi \subseteq \Omega.\ \varphi \vdash \rho \equiv \tau' \dashv \psi$, where we replace τ' with τ by Red-R.
- $\dfrac{\sigma_1 \equiv \tau_1 \qquad \dots \qquad \sigma_{|F|} \equiv \tau_{|F|} \qquad F\,\sigma_1 \dots \sigma_{|F|}\,\mathsf{NF} \qquad F\,\tau_1 \dots \tau_{|F|}\,\mathsf{NF}}{F\,\sigma_1 \dots \sigma_{|F|} \equiv F\,\tau_1 \dots \tau_{|F|}}$ Then $[\Omega]\rho = F\,\sigma_1 \dots \sigma_{|F|}$ and $\tau = F\,\tau_1 \dots \tau_{|F|}$. Let us destruct $[\Omega]\rho$. `Ilya: we need a lemma to destruct it this way`
  - $\rho = \hat{x}$ and $(\hat{x}{:=}F\,\sigma_1 \dots \sigma_{|F|}) \in \Omega$. Let us consider an arbitrary $\varphi \subseteq \Omega$. φ must be well-formed, then either $(\hat{x}{:=}F\,\sigma_1 \dots \sigma_{|F|}) \in \varphi$ or $(\hat{x}{:=}\cdot) \notin \varphi$.
    * $(\hat{x}{:=}F\,\sigma_1 \dots \sigma_{|F|}) \in \varphi$ then we take $\psi = \varphi$ and apply U-Keep.
    * $(\hat{x}{:=}\cdot) \notin \varphi$ then we take $\psi = \varphi, (\hat{x}{:=}F\,\sigma_1 \dots \sigma_{|F|})$ and apply U-Add. The term to which we assign $\hat{x}$ is in the normal form by one of the premises.
  - $\rho = F\,\rho_1 \dots \rho_{|F|}$, $F \neq \hat{x}$, and $[\Omega]\rho_i = \sigma_i$ for $i = 1 \dots |F|$.
    By the corollary of lemma 4, $\rho_i\,\mathsf{OK}$. So we can apply the induction hypothesis to all the components to acquire $|F|$ facts: $\forall \varphi \subseteq \Omega.\ \exists \psi \subseteq \Omega.\ \varphi \vdash \rho_i \equiv \tau_i \dashv \psi$.
    Let us apply the first fact to an arbitrary $\varphi = \psi_0 \subseteq \Omega$ to acquire $\psi_1 \subseteq \Omega$. Then we apply the second fact to ψ_1, acquiring $\psi_2 \subseteq \Omega$. Repeating the process, we have: $\psi_0 \vdash \rho_1 \equiv \tau_1 \dashv \psi_1,\ \dots,\ \psi_{|F|-1} \vdash \rho_{|F|} \equiv \tau_{|F|} \dashv \psi_{|F|}$.
    Notice that by lemma 2 and proposition 1, $F\,\rho_1 \dots \rho_{|F|}\,\mathsf{NF}$. Then we apply the congruence unification rule and get $\varphi \vdash F\,\rho_1 \dots \rho_{|F|} \equiv F\,\tau_1 \dots \tau_{|F|} \dashv \psi_{|F|}$, i.e. $\varphi \vdash \rho \equiv \tau \dashv \psi_{|F|}$, so we take $\psi_{|F|}$ as ψ.

□

9 Typing

Definition 19 (Typing declarative context)**.**

$$\textit{Contexts} \quad \Gamma \quad ::= \quad \cdot \ \mid\ \Gamma, x : A \ \mid\ \Gamma, B\,\mathsf{vtype}$$

To make the typing decidable, we restrict the system in several ways. In particular, when we form $\forall x : A.\, X$, we require x to belong to $\mathsf{FV}(X)$ and occur safely in X.

9.1 Context Well-formedness

$$\frac{}{\vdash \cdot}\ \textsc{Ctx0} \qquad\qquad \frac{\vdash \Gamma}{\vdash \Gamma, x\,\mathsf{vtype}}\ \textsc{CtxIT} \qquad\qquad \frac{\Gamma \vdash A\,\mathsf{vtype} \qquad \vdash \Gamma}{\vdash \Gamma, x : A}\ \textsc{CtxI}$$

9.2 Context Formation and Var

Here, j denotes the context entry: either $(x : A)$ or $(x\,\mathsf{vtype})$.

$$\frac{j \in \Gamma}{j \in (\Gamma, y : B)}\ \textsc{CtxExt} \qquad \frac{j \in \Gamma}{j \in (\Gamma, B\,\mathsf{vtype})}\ \textsc{CtxExtT} \qquad \frac{}{j \in (\Gamma, j)}\ \textsc{CtxInit}$$

$$\frac{x : A \in \Gamma}{\Gamma, x : A \vdash_v x \Rightarrow A}\ \textsc{Var} \qquad \frac{x\,\mathsf{vtype} \in \Gamma}{\Gamma \vdash x\,\mathsf{vtype}}\ \textsc{VarT}$$

9.3 Subsumption

$$\frac{\Gamma \vdash_c t \Rightarrow Y \qquad \Gamma \vdash X \leqslant^c Y}{\Gamma \vdash_c t \Leftarrow X}\ \leqslant^c \qquad \frac{\Gamma \vdash_v v \Rightarrow B \qquad \Gamma \vdash A \leqslant^v B}{\Gamma \vdash_v v \Leftarrow A}\ \leqslant^v$$

9.4 Universes

$$\frac{\Gamma \vdash A\,\mathsf{vtype}}{\Gamma \vdash \uparrow\! A\,\mathsf{ctype}}\ \mathcal{F} \qquad \frac{\Gamma \vdash X\,\mathsf{ctype}}{\Gamma \vdash \downarrow\! X\,\mathsf{vtype}}\ \mathcal{U} \qquad \frac{\Gamma \vdash A\,\mathsf{vtype} \qquad \Gamma, x : A \vdash X\,\mathsf{ctype}}{\Gamma \vdash \Pi x : A.\, X\,\mathsf{ctype}}\ \Pi$$

$$\frac{\Gamma \vdash A\,\mathsf{vtype} \qquad \Gamma, x : A \vdash X\,\mathsf{ctype} \qquad x \in \mathsf{FV}(X) \qquad x \in X\,\mathsf{OK}}{\Gamma \vdash \forall x : A.\, X\,\mathsf{ctype}}\ \forall$$

$$\frac{\Gamma \vdash A\,\mathsf{vtype} \qquad \Gamma, x : A \vdash B\,\mathsf{vtype}}{\Gamma \vdash \Sigma x : A.\, B\,\mathsf{ctype}}\ \Sigma \qquad \frac{\Gamma \vdash A\,\mathsf{vtype} \qquad \Gamma \vdash_v v \Leftarrow A \qquad \Gamma \vdash_v w \Leftarrow A}{\Gamma \vdash \mathsf{eq} A\, v\, w\,\mathsf{vtype}}\ \mathsf{eq}$$

$$\frac{\Gamma, x : A \vdash X\,\mathsf{ctype} \qquad \Gamma \vdash_c e \Leftarrow A}{\Gamma \vdash (\mathsf{let}\ x : A\ := \ e\ \mathsf{in}\ X)\,\mathsf{ctype}}\ \textsc{Let-type}$$

9.5 $\mathcal{F}$ and $\mathcal{U}$

$$\frac{\Gamma \vdash_c t \Leftarrow X}{\Gamma \vdash_v \{t\} \Leftarrow \downarrow\! X}\ \mathcal{U}\mathrm{I}{\Leftarrow} \qquad \frac{\Gamma \vdash_c t \Rightarrow X}{\Gamma \vdash_v \{t\} \Rightarrow \downarrow\! X}\ \mathcal{U}\mathrm{I}{\Rightarrow} \qquad \frac{\Gamma \vdash_v v \Leftarrow \downarrow\! X}{\Gamma \vdash_c \mathsf{force}\, v \Leftarrow X}\ \mathcal{U}\mathrm{E}{\Leftarrow}$$

$$\frac{\Gamma \vdash_v v \Rightarrow \downarrow\! X}{\Gamma \vdash_c \mathsf{force}\, v \Rightarrow X}\ \mathcal{F}\mathrm{E}{\Rightarrow} \qquad \frac{\Gamma \vdash_v v \Leftarrow A}{\Gamma \vdash_c \mathsf{return}\, v \Leftarrow \uparrow\! A}\ \mathcal{F}\mathrm{I}{\Leftarrow} \qquad \frac{\Gamma \vdash_v v \Rightarrow A}{\Gamma \vdash_c \mathsf{return}\, v \Rightarrow \uparrow\! A}\ \mathcal{F}\mathrm{I}{\Rightarrow}$$

9.6 Let and Dependent Let

$$\frac{\Gamma, x : A \vdash_c u \Rightarrow X \qquad \Gamma \vdash_c t \Leftarrow \uparrow\! A \qquad \Gamma \vdash X\,\mathsf{ctype} \qquad \Gamma \vdash A\,\mathsf{vtype}}{\Gamma \vdash_c \mathsf{let}\ x : A\ := \ t\ \mathsf{in}\ u \Rightarrow X}\ \textsc{Let}{\Rightarrow}$$

$$\frac{\Gamma \vdash_c t \Leftarrow \uparrow\! A \qquad \Gamma \vdash X\,\mathsf{ctype} \qquad \Gamma, x : A \vdash_c u \Leftarrow X}{\Gamma \vdash_c \mathsf{let}\ x : A\ := \ t\ \mathsf{in}\ u \Leftarrow X}\ \textsc{Let}{\Leftarrow}$$

$$\frac{\Gamma, x : A \vdash_c u \Rightarrow X \qquad \Gamma \vdash_c t \Leftarrow \uparrow\! A \qquad \Gamma, x : A \vdash X\,\mathsf{ctype}}{\Gamma \vdash_c \mathsf{dlet}\ x : A\ := \ t\ \mathsf{in}\ u \Rightarrow (\mathsf{let}\ x : A\ := \ t\ \mathsf{in}\ X)}\ \textsc{DLet}{\Rightarrow}$$

$$\frac{\Gamma \vdash_c t \Leftarrow \uparrow\! A \qquad \Gamma, x : A \vdash X\,\mathsf{ctype} \qquad \Gamma, x : A \vdash_c u \Leftarrow X}{\Gamma \vdash_c \mathsf{dlet}\ x : A\ := \ t\ \mathsf{in}\ u \Leftarrow (\mathsf{let}\ x : A\ := \ t\ \mathsf{in}\ X)}\ \textsc{DLet}{\Leftarrow}$$

9.7 ∀, Π, and Σ

$$\frac{\Gamma, x : A \vdash X\ \mathsf{ctype} \qquad \Gamma, x : A \vdash_c t \Leftarrow X}{\Gamma \vdash_c \lambda x : A.\, t \Leftarrow \forall x : A.\, X}\ \forall\mathrm{I}\Leftarrow$$

$$\frac{\Gamma, x : A \vdash X\ \mathsf{ctype} \qquad \Gamma, x : A \vdash_c t \Leftarrow X}{\Gamma \vdash_c \lambda x : A.\, t \Leftarrow \Pi x : A.\, X}\ \Pi\mathrm{I}\Leftarrow \qquad \frac{\Gamma \vdash_c t \Rightarrow \Pi x : A.\, X \qquad \Gamma \vdash_v v \Leftarrow A}{\Gamma \vdash_c t\, v \Rightarrow X\{x := v\}}\ \Pi\mathrm{E}$$

$$\frac{\Gamma \vdash_v v \Leftarrow A \qquad \Gamma \vdash_v w \Leftarrow B\{x := v\} \qquad \Gamma \vdash \Sigma x : A.\, B\ \mathsf{vtype}}{\Gamma \vdash_v \langle v, w\rangle \Leftarrow \Sigma x : A.\, B}\ \Sigma\mathrm{I}\Leftarrow$$

$$\frac{\Gamma, p : (\Sigma x : A.\, B) \vdash X\ \mathsf{ctype} \qquad \Gamma \vdash_v v \Leftarrow \Sigma x : A.\, B \qquad \Gamma \vdash_c t \Leftarrow \Pi(x : A)(y : B).\, X\{p := \langle x, y\rangle\}}{\Gamma \vdash_c \mathsf{rec}^X_\Sigma(v, t) \Rightarrow X\{p := v\}}\ \Sigma\mathrm{E}$$

9.8 Equality

$$\frac{\Gamma \vdash A\,\mathsf{vtype} \qquad \Gamma \vdash_v v \Leftarrow A}{\Gamma \vdash_v \mathsf{refl} \Leftarrow \mathsf{eq} A\, v\, v}\ \mathsf{eqI}$$

$$\frac{\Gamma \vdash_v v \Leftarrow \mathsf{eq} A\, w_1\, w_2 \qquad \Gamma, x : A, p : \mathsf{eq} A\, w_1\, x \vdash X\ \mathsf{ctype} \qquad \Gamma \vdash_c t \Leftarrow X\, w_1\, \mathsf{refl}}{\Gamma \vdash_c \mathsf{rec}^X_{\mathsf{eq}}(v, t) \Leftarrow X\{x := w_2\}\{p := v\}}\ \mathsf{eqE}\Leftarrow$$

10 Declarative Subtyping

$$\frac{\Gamma \vdash A_1 \leqslant^v B_1 \qquad \Gamma, x : A_1 \vdash A_2 \leqslant^v B_2}{\Gamma \vdash \Sigma x : A_1.\, A_2 \leqslant^v \Sigma x : B_1.\, B_2}\ \leqslant\Sigma \qquad \frac{\Gamma \vdash A_2 \leqslant^v A_1 \qquad \Gamma, x : A_2 \vdash X_1 \leqslant^c X_2}{\Gamma \vdash \Pi x : A_1.\, X_1 \leqslant^c \Pi x : A_2.\, X_2}\ \leqslant\Pi$$

$$\frac{\Gamma \vdash X_1 \leqslant^c X_2 \qquad \Gamma \vdash X_2 \leqslant^c X_1}{\Gamma \vdash {\downarrow} X_1 \leqslant^v {\downarrow} X_2}\ \leqslant\mathcal{U} \qquad \frac{\Gamma \vdash A_1 \leqslant^v A_2 \qquad \Gamma \vdash A_2 \leqslant^v A_1}{\Gamma \vdash {\uparrow} A_1 \leqslant^c {\uparrow} A_2}\ \leqslant\mathcal{F}$$

$$\frac{\Gamma \vdash A \leqslant^v B \qquad \Gamma \vdash v_1 \equiv v_2 : A \qquad \Gamma \vdash w_1 \equiv w_2 : A}{\Gamma \vdash \mathsf{eq} A\, v_1\, w_1 \leqslant^v \mathsf{eq} B\, v_2\, w_2}\ \leqslant\textsc{Eq}$$

$$\frac{X \text{ is } x\text{-neutral} \qquad \Gamma \vdash v : A \qquad \Gamma \vdash X\{x := v\} \leqslant^c Y}{\Gamma \vdash (\forall x : A.\, X) \leqslant^c Y}\ \forall\leqslant \qquad \frac{\Gamma, y : A \vdash X \leqslant^c Y}{\Gamma \vdash X \leqslant^c (\forall y : A.\, Y)}\ \leqslant\forall$$

$$\frac{\Gamma \vdash e \equiv \mathsf{return}\, v : A \qquad \Gamma \vdash X\{x := v\} \leqslant^c Y}{\Gamma \vdash (\mathsf{let}\ x : A := e\ \mathsf{in}\ X) \leqslant^c Y}\ \textsc{Let}\leqslant$$

$$\frac{\Gamma \vdash e \equiv \mathsf{return}\, v : A \qquad \Gamma \vdash X \leqslant^c Y\{y := v\}}{\Gamma \vdash X \leqslant^c (\mathsf{let}\ y : A := e\ \mathsf{in}\ Y)}\ \leqslant\textsc{Let}$$

$$\frac{\Gamma \vdash A \leqslant^v B \qquad \Gamma \vdash B \leqslant^v A \qquad \Gamma \vdash e_1 \equiv e_2 : A \qquad \Gamma, x : A \vdash X \leqslant^c Y}{\Gamma \vdash \mathsf{let}\ x : A := e_1\ \mathsf{in}\ X \leqslant^c \mathsf{let}\ x : B := e_2\ \mathsf{in}\ Y}\ \textsc{Let}\leqslant\textsc{Let}$$

11 Algorithmic Subtyping

$$\frac{\varphi \vdash A_1 \leqslant^v B_1 \dashv \varphi' \qquad \varphi', x : A_1 \vdash A_2 \leqslant^v B_2 \dashv \varphi''}{\varphi \vdash \Sigma x : A_1.\, A_2 \leqslant^v \Sigma x : B_1.\, B_2 \dashv \varphi''} \leqslant \Sigma$$

$$\frac{\varphi \vdash A_2 \leqslant^v A_1 \dashv \varphi' \qquad \varphi', x : A_2 \vdash [\varphi']X_1 \leqslant^c X_2 \dashv \varphi''}{\varphi \vdash \Pi x : A_1.\, X_1 \leqslant^c \Pi x : A_2.\, X_2 \dashv \varphi''} \leqslant \Pi$$

$$\frac{\varphi \vdash X_2 \leqslant^c X_1 \dashv \varphi' \qquad \varphi' \vdash X_1 \leqslant^c [\varphi']X_2 \dashv \varphi''}{\varphi \vdash {\downarrow} X_1 \leqslant^v {\downarrow} X_2 \dashv \varphi''} \leqslant \mathcal{U} \qquad \frac{\varphi \vdash A_2 \leqslant^v A_1 \dashv \varphi' \qquad \varphi' \vdash [\varphi']A_1 \leqslant^v A_2 \dashv \varphi''}{\varphi \vdash {\uparrow} A_1 \leqslant^c {\uparrow} A_2 \dashv \varphi''} \leqslant \mathcal{F}$$

$$\frac{\varphi \vdash A \leqslant^v B \dashv \varphi' \qquad \varphi' \vdash v_1 \equiv [\varphi']v_2 \dashv \varphi'' \qquad \varphi'' \vdash w_1 \equiv [\varphi'']w_2 \dashv \varphi'''}{\Gamma \vdash \mathsf{eq} A\, v_1\, w_1 \leqslant^v \mathsf{eq} B\, v_2\, w_2} \leqslant\text{EQ}$$

$$\frac{\varphi, \hat{v} \vdash X\{x := \hat{v}\} \leqslant^c Y \dashv \varphi'}{\varphi \vdash (\forall x : A.\, X) \leqslant^c Y \dashv \varphi'} \forall\leqslant \qquad \frac{\varphi, y : A \vdash X \leqslant^c Y \dashv \varphi'}{\varphi \vdash X \leqslant^c (\forall y : A.\, Y) \dashv \varphi'} \leqslant\forall$$

$$\frac{e \text{ and } A \text{ are ground} \qquad \varphi, \hat{v} \vdash e \equiv \mathsf{return}\, \hat{v} \dashv \varphi', (\hat{v}{:=}v) \qquad \varphi' \vdash [\varphi']X\{x := v\} \leqslant^c Y \dashv \varphi', (\hat{v}{:=}v)}{\varphi \vdash (\mathsf{let}\ x : A := e\ \mathsf{in}\ X) \leqslant^c Y \dashv \varphi''} \text{LET}\leqslant$$

$$\frac{\varphi, \hat{v} \vdash e \equiv \mathsf{return}\, \hat{v} \dashv \varphi', (\hat{v}{:=}v) \qquad \varphi' \vdash [\varphi']X \leqslant^c Y\{y := v\} \dashv \varphi''}{\varphi \vdash X \leqslant^c (\mathsf{let}\ y : A := e\ \mathsf{in}\ Y) \dashv \varphi''} \leqslant\text{LET}$$

$$\frac{\begin{array}{c}\varphi \vdash B \leqslant^v A \dashv \varphi_1 \\ \varphi_1 \vdash [\varphi_1]A \leqslant^v B \dashv \varphi_2 \qquad \varphi_2 \vdash [\varphi_1]e_2 \equiv e_1 \dashv \varphi_3 \qquad \varphi_3, x : [\varphi_1]A \vdash [\varphi_3]X \leqslant^c Y \dashv \varphi_4\end{array}}{\varphi \vdash \mathsf{let}\ x : A := e_1\ \mathsf{in}\ X \leqslant^c \mathsf{let}\ x : B := e_2\ \mathsf{in}\ Y \dashv \varphi_4} \text{LET}\leqslant\text{LET}$$

Ilya: In (let$\leqslant$), type A can have unresolved variables, hence, the first unification in the premises can instantiate them, and thus, must be algorithmic.

Ilya: In (let$\leqslant$), we require e to be ground, which makes the algorithmic system incomplete w.r.t. the declarative system.
E.g. $(\forall x : 1.\, \mathsf{let}\ y : 1 := \mathsf{return}\, x\ \mathsf{in}\ {\uparrow}\mathsf{eq}{\downuparrows}1\, \{y\}\, \{y\}) \leqslant \mathsf{eq}{\downuparrows}1\, \{\mathsf{return}\,()\}\, \{\mathsf{return}\,()\}$ holds for the declarative system but not for the algorithmic.

11.1 Natural Numbers and Undecidability

The type system can be easily extended with natural numbers. To this purpose, we must add $\mathbb{N}$, $\mathsf{0}$, and $\mathsf{succ}(v)$ to the values, and $\mathsf{rec}^X_{\mathbb{N}}(v, \mathit{base}, \mathit{step})$ to the computations with obvious typing inference rules. We also add $\mathsf{rec}^X_{\mathbb{N}}(\mathsf{0}, b, s) \to b$ and $\mathsf{rec}^X_{\mathbb{N}}(\mathsf{succ}(v), b, s) \to s\, v\, \mathsf{rec}^X_{\mathbb{N}}(v, b, s)$ to the reduction rules.

Notice that we do not unify under the induction operators. For example, $\varphi \vdash \mathsf{rec}^X_{\mathbb{N}}(\hat{v}, \mathsf{0}, \lambda x\, y.\, \mathsf{0}) \equiv \mathsf{0} \dashv \varphi, (\hat{v}{:=}\mathsf{0})$ is not admissible. Moreover, the general unification is undecidable in this case.

Roughly, this is because we can easily define integers, arithmetic operations, and hence, any arbitrary polynomial $P(\hat{x_1}, \ldots, \hat{x_n})$. The unification of this polynomial with $\mathsf{0}$ corresponds to solving a diophantine equation, which is undecidable.

11.2 Invariants

We should be able to infer $\varphi \vdash (\mathsf{let}\ x : {\uparrow}\mathsf{eq} A\, \hat{u}\, \hat{v} := \hat{w}\ \mathsf{in}\ {\uparrow}\mathit{Int}) \leqslant^c {\uparrow}\mathit{Int} \dashv \varphi$. As you can see, the unused existential variables $\hat{u}$ and $\hat{v}$ stay uninitialized. It breaks the invariant that the subtyping algorithm 'makes' both sides of $\leqslant$ ground (i.e. all existential variables are initialized in the output context). As such, we weaken the notion of 'ground' terms in such a way that they might have existential variables as long as they are not used in the outcome.

`Ilya: I've just realized that what I would like to mean by the usage of the variables depends on the evaluation. So maybe it's worth trying another approach, e.g. instantiate existential variables with '?' and promise that it won't cause any problem in the unification.`

12 Properties

`Ilya: Outdated`

Lemma 10 (Mode-correctness)**.** *Each rule in section 9 is mode-correct. Specifically, as defined in [***dunfield2021:bidirection**

1. *The premises are mode-correct: for each premise, every input meta-variable is known from the input of the rule's conclusion and the outputs of the earlier premises.*
2. *The conclusion is mode-correct: if all premises have been derived, the outputs of the conclusion are known.*

Proof. First, we prove the mode-correctness of *conclusion* for each rule. Note that it is only relevant for the *synthesizing* rules, because for the *checking* rules, the resulting type is given as an input.

(Var) A is known from the input of the conclusion.

(Universes) For rules in section 9.4 ($\mathcal{U}$, $\mathcal{F}$, Π, Σ, eq), the resulting type (a universe) is the only possible option.

($\mathcal{U}$I$\Rightarrow$) X is known from the output of $\Gamma \vdash_c t \Rightarrow X$.

($\mathcal{F}$E$\Rightarrow$) X is known from the output of $\Gamma \vdash_v v \Rightarrow {\downarrow} X$.

($\mathcal{F}$I$\Rightarrow$) A is known from the output of $\Gamma \vdash_v v \Rightarrow {\downarrow} X$.

(Let$\Rightarrow$) X is known from the output of $\Gamma, x : A \vdash_c u \Rightarrow X$.

(DLet$\Rightarrow$) x, A, and t are given in the input of the conclusion; X is known from the output of $\Gamma, x : A \vdash_c u \Rightarrow X$.

(ΠE) v is given in the input of the conclusion; x and X are known from the output of $\Gamma \vdash_c t \Rightarrow \Pi x : A.\, X$.

(ΣE) X and v are given in the input of the conclusion.

(eqE$\Rightarrow$) X and v are given in the input of the conclusion; w_2 is known from the output of $\Gamma \vdash_v v \Rightarrow \mathsf{eq} A\, w_1\, w_2$.

Second, we let us show the mode-correctness of *premises*.

(Ctx0) There are no premises.

(CtxI) Γ and A are given in the input of the conclusion.

(CtxExt) X, A, and Γ are given in the input of the conclusion.

(CtxInit) There are no premises.

(Var) x, A, and Γ are given in the input of the conclusion.

(EqivC) Γ, t, and X are given in the input of the conclusion; Y is known from the output of $\Gamma \vdash_c t \Rightarrow Y$.

(EqivV) Γ, v, and A are given in the input of the conclusion; B is known from the output of $\Gamma \vdash_v v \Rightarrow B$.

($\mathcal{F}$) Γ and A are given in the input of the conclusion.

($\mathcal{U}$) Γ and X are given in the input of the conclusion.

(Π) Γ, A, x, and X are given in the input of the conclusion.

(Σ) Γ, A, x, and B are given in the input of the conclusion.

(eq) Γ, A, v, and w are given in the input of the conclusion.

($\Box^v$ and $\Box^c$) There are no premises.

($\mathcal{U}$I$\Leftarrow$) Γ, t, and X are given in the input of the conclusion.

($\mathcal{U}$I$\Rightarrow$) Γ and t are given in the input of the conclusion.

($\mathcal{U}$E$\Leftarrow$) Γ, v, and X are given in the input of the conclusion.

($\mathcal{F}$E$\Rightarrow$) Γ and v are given in the input of the conclusion.

($\mathcal{F}$I$\Leftarrow$) Γ, v, and A are given in the input of the conclusion.

($\mathcal{F}$I$\Rightarrow$) Γ and v are given in the input of the conclusion.

(Let$\Rightarrow$) Γ, x, A, u, and t are given in the input of the conclusion; X is known from the output of $\Gamma, x : A \vdash_c u \Rightarrow X$.

(Let$\Leftarrow$) Γ, t, A, X, x, and u are given in the input of the conclusion.

(DLet$\Rightarrow$) Γ, x, A, u, and t are given in the input of the conclusion. X is known from the output of $\Gamma, x : A \vdash_c u \Rightarrow X$.

(DLet$\Leftarrow$) Γ, t, A, x, X, and u are given in the input of the conclusion.

(ΠI$\Leftarrow$) Γ, x, A, X, and t are given in the input of the conclusion.

(ΠE) Γ, t, and v are given in the input of the conclusion; A is known from the output of $\Gamma \vdash_c t \Rightarrow \Pi x : A.\, X$

(ΣI$\Leftarrow$) Γ, v, A, w, B, x are given in the input of the conclusion.

(ΣE) Γ, X, and v, are given in the input of the conclusion; x, A, and B are known from the output of $\Gamma \vdash_c X \Rightarrow \Sigma x : A.\, B \to \Box^c$; y is an arbitrary fresh variable.

(eqI) Γ, A, and v are given in the input of the conclusion.

(eqE$\Rightarrow$) Γ, v, X, and t are given in the input of the conclusion; A and w_1 are known from the output of $\Gamma \vdash_v v \Rightarrow \mathsf{eq} A\, w_1\, w_2$.

$\Box$

Lemma 11 (Context Soundness)**.** *If $x : A \in \Gamma$ and $\vdash \Gamma$ then $\Gamma \vdash_v A \Leftarrow \Box^v$*

Proof. `Ilya: By trivial induction on` $x : A \in \Gamma$ $\Box$

Lemma 12 (Regularity)**.** *The types synthesized by $\Rightarrow$ are well-formed. Specifically, the following properties hold*

1. *if $\vdash \Gamma$ and $\Gamma \vdash_v v \Rightarrow A$ then $\Gamma \vdash_v A \Leftarrow \Box^v$*
2. *if $\vdash \Gamma$ and $\Gamma \vdash_c t \Rightarrow X$ then $\Gamma \vdash_v X \Leftarrow \Box^c$*

Proof. We prove this property by mutual structural induction on $\Gamma \vdash_v v \Rightarrow A$ and $\Gamma \vdash_c t \Rightarrow X$. Let us consider the synthesizing rules.

(Var) Since $(x : A)$ belongs to a *well-formed* context Γ, the property we need ($\Gamma \vdash_v A \Leftarrow \Box^v$) holds by lemma 11.

(Universes) Each rule in section 9.4 synthesizes either $\Box^v$ or $\Box^c$. The desired properties hold by the following derivation trees:

$$\dfrac{\dfrac{}{\Gamma \vdash_v \Box^v \Rightarrow \Box^v}\,\Box^v \qquad \Box^v \equiv \Box^v}{\Gamma \vdash_v \Box^v \Leftarrow \Box^v}\,\textsc{EqivV} \qquad\qquad \dfrac{\dfrac{}{\Gamma \vdash_c \Box^c \Rightarrow \Box^c}\,\Box^c \qquad \Box^c \equiv \Box^c}{\Gamma \vdash_c \Box^c \Leftarrow \Box^c}\,\textsc{EqivC}$$

($\mathcal{U}$I$\Rightarrow$ and $\mathcal{F}$I$\Rightarrow$) The following derivation trees prove the required properties, where † and ‡ are derived from the inductive hypotheses.

$$\dfrac{\dfrac{\dfrac{\dagger}{\Gamma \vdash_c X \Leftarrow \Box^c}}{\Gamma \vdash_v {\downarrow}X \Rightarrow \Box^v}\,\mathcal{U} \qquad {\downarrow}X \equiv {\downarrow}X}{\Gamma \vdash_v {\downarrow}X \Leftarrow \Box^v}\,\textsc{EqivC} \qquad\qquad \dfrac{\dfrac{\dfrac{\ddagger}{\Gamma \vdash_v A \Leftarrow \Box^v}}{\Gamma \vdash_c {\uparrow}A \Rightarrow \Box^c}\,\mathcal{F} \qquad {\uparrow}A \equiv {\uparrow}A}{\Gamma \vdash_c {\uparrow}A \Leftarrow \Box^c}\,\textsc{EqivV}$$

($\mathcal{F}$E$\Rightarrow$) `Ilya: Depends on` $\Box^v \equiv \cdot$

(Let$\Rightarrow$) The desired property is in the premises.

(DLet$\Rightarrow$) The following derivation tree proves the required property. $\Gamma \vdash_c t \Leftarrow {\uparrow}A$ and $\Gamma, x : A \vdash_c u \Rightarrow \Box^c$ are given as premises.

$$\dfrac{\Gamma \vdash_c t \Leftarrow {\uparrow}A \qquad \dfrac{\dfrac{}{\Gamma \vdash_c \Box^c \Rightarrow \Box^c}\,\Box^c}{\Gamma \vdash_c \Box^c \Leftarrow \Box^c} \qquad \dfrac{\Gamma, x : A \vdash_c u \Rightarrow \Box^c}{\Gamma, x : A \vdash_c u \Leftarrow \Box^c}}{\Gamma \vdash_c \mathsf{let}\ x : A\ := \ t\ \mathsf{in}\ u \Leftarrow \Box^c}\,\textsc{Let}\Leftarrow$$

(ΠE) `Ilya: Depends on` $\equiv$ `and requires the substitution lemma`

(ΣE) By applying (ΠE) to the first two premises.

(eqE$\Rightarrow$) `Ilya: The same trick as in (`Σ`E) doesn't work...`

□