1 Syntax

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

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

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.

We reserve x and y to denote term-level variables, and a and b to denote type-level variables.

2 Computational form of the terms

Let us consider the term syntax from a different perspective:

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

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

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
$\mathsf{ar}F$	$[0,1]$	$[0,0]$	$[]$	$[0]$	$[0]$	$[0,0]$	$[0,1,0]$	$[0,2,0]$	$[0,1,0]$	$[0,1,0]$	$[0]$	$[0]$	$[0,1]$	$[0,1]$	$[0]$	$[0,1]$	$[]$
$\lvert F\rvert$	2	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)**.**

$$\frac{}{x\{x \rightsquigarrow z\} = z} \qquad \frac{x \neq y}{x\{y \rightsquigarrow z\} = x} \qquad \frac{F \neq x}{F\, P_1 \ldots P_{|F|}\{y \rightsquigarrow z\} = F\,(P_1\{y \rightsquigarrow z\}) \ldots (P_{|F|}\{y \rightsquigarrow z\})}$$

$$\frac{}{.\tau\{y \rightsquigarrow z\} = .(\tau\{y \rightsquigarrow z\})} \qquad \frac{x' \textit{ is fresh}}{x.P\{y \rightsquigarrow z\} = x'.((P\{x \rightsquigarrow x'\})\{y \rightsquigarrow z\})}$$

Definition 5 (Alpha-equivalence)**.**

$$\frac{\forall i, P_i \sim_\alpha Q_i}{F\, P_1 \ldots P_{|F|} \sim_\alpha F\, Q_1 \ldots Q_{|F|}} \qquad \frac{\sigma \sim_\alpha \tau}{.\sigma \sim_\alpha .\tau} \qquad \frac{y \textit{ is fresh} \qquad \sigma\{x \rightsquigarrow y\} \sim_\alpha \tau\{x \rightsquigarrow y\}}{x.\sigma \sim_\alpha x.\tau}$$

Lemma 1. *Alpha-equivalence is an equivalence relation on terms and abstractors.*

`Ilya: Admitted.`

Lemma 2 (Functionality of Variable Renaming)**.** *Variable Renaming is a functional on the classes of alpha-equivalence.*

`Ilya: Admitted.`

Hereafter, we assume every statement about terms and abstractors defined on the equivalence classes. Whenever we use the "concrete term syntax", we mean the alpha-equivalence class of this term if the term is in the covariant position of the statement or definition (e.g. we are constructing a function returning an equivalence class as an output); and *any term of this form from this class* if the term is in the contravariant position (e.g. we are constructing a function taking an equivalence class as an input).

4 Susbtititution

Definition 6 (Substitution)**.** *`Ilya: todo`*

Lemma 3 (Functionality of Substituion)**.** *Substitution is a functional on the classes of alpha-equivalence.*

`Ilya: Admitted.`

5 Reduction

First, we define the *redex contraction.*

Definition 7 (Redex Contraction)**.** *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 $E\,(C\, P_1 \ldots P_{|C|})\, Q_2 \ldots Q_{|E|}$ where E and C are "matched". Vice versa, if a term of the form $E\,(C\, P_1 \ldots P_{|C|})\, Q_2 \ldots Q_{|E|}$ 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).

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

Lemma 4 (Substitution preserves reduction).

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

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

$\square$

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).

$$\frac{\tau\ \mathsf{ATOM}}{\tau\ \mathsf{NF}} \qquad \frac{\tau_1\ \mathsf{NF}\ \ldots\ \tau_{|C|}\ \mathsf{NF}}{C\, \overrightarrow{x}\tau_1 \ldots \overrightarrow{x}\tau_{|C|}\ \mathsf{NF}} \qquad \frac{\tau_1\ \mathsf{NF}\ \ldots\ \tau_{|N|}\ \mathsf{NF}}{N\, \overrightarrow{x}\tau_1 \ldots \overrightarrow{x}\tau_{|N|}\ \mathsf{ATOM}} \qquad \frac{\tau_1\ \mathsf{ATOM} \qquad \tau_2\ \mathsf{NF}\ \ldots\ \tau_{|E|}\ \mathsf{NF}}{E\, \overrightarrow{x}\tau_1 \ldots \overrightarrow{x}\tau_{|E|}\ \mathsf{ATOM}}$$

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).

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

Definition 10 (Reduction to the Normal Form).

$$\frac{\tau \to \tau' \qquad \tau'\ \mathsf{NF}}{\tau \Downarrow \tau'}$$

Proposition 2 (Reduction-Substitution distributivity).

$$\Downarrow(\sigma\{x := \Downarrow v\}) = \Downarrow(\sigma\{x := v\})$$

7 Safe Occurrence

Ilya: Hmm.. Discussing the type system before actually introducing it...

In this section, we describe the basis of the restriction that we put on the type system to make the type checking and type inference decidable. First, let us motivate why we need to restrict the type system.

7.1 The necessity of the restrictions

We motivate the restrictions that we put on the type system by considering two well-known undecidable problems: the Inhabitation problem and Hilbert's tenth problem. We will show that both of them can be reduced to the typing problems in the unrestricted system.

The Inhabitation Let us consider the term $\langle\rangle$ checked against type $\forall x : \tau.\, \mathsf{1}$. It is easy to see that the checking succeed if and only if τ is inhabited. Thus, to ensure the decidability of the type checking, we require x, bound by some $\forall$, to occur within the body of that $\forall$. In other words, only do we allow $\forall x : \sigma.\, \tau$ to be formed, if $x \in \mathsf{FV}(\tau)$.

Hilbert's Tenth Problem As will be shown in sections 9, 10 and 12, the "driver" of the type inference algorithm is the subtyping algorithm, and the "driver" of the subtyping, is the unification. Let us show that in the unrestricted case, the unification is undecidable.

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 $\mathsf{rec}^X_{\mathbb{N}}(v, base, step)$ to the computations with obvious typing inference rules. We also add $\mathsf{rec}^X_{\mathbb{N}}(\mathsf{0}, b, s) \rightharpoonup b$ and $\mathsf{rec}^X_{\mathbb{N}}(\mathsf{succ}(v), b, s) \rightharpoonup s\, v\, \mathsf{rec}^X_{\mathbb{N}}(v, b, s)$ to the contraction rules.

Then the unification that we will not provide is, for example, the following: $\mathsf{rec}^X_{\mathbb{N}}(\hat{v}, \mathsf{0}, \lambda x\, y.\, \mathsf{0}) \equiv \mathsf{0}$, where $\hat{v}$ is the unification variable that we must initialize. Although $(\hat{v}{:=}\mathsf{0})$ solves this unification, we rule out such cases from our unification algorithm. This is because having the unification variable on the reducible position means being able to "invert" the recursion, which is too powerful to be decidable.

Specifically, after we defined integers, arithmetic operations, we can define 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.

7.2 The Restriction on Quantifiers

To deal with both aforementioned undecidabilities, we introduce a syntactic judgement "*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 $E\, x\, \tau_2 \ldots \tau_{|E|}$ 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 via $x \in^? \tau\, \mathsf{OK}$ judgement:

Definition 11 (Safe Occurrence)**.** `Ilya: TODO: add safe occurrence in abstractors`

$$\frac{x \in^? \tau_1\, \mathsf{OK} \;\; \ldots \;\; x \in^? \tau_{|C|}\, \mathsf{OK}}{x \in^? C\, \vec{x}\tau_1 \ldots \vec{x}\tau_{|C|}\, \mathsf{OK}}\ \textsc{C-Cong} \qquad \frac{x \in^? \tau_1\, \mathsf{OK} \;\; \ldots \;\; x \in^? \tau_{|N|}\, \mathsf{OK}}{x \in^? N\, \vec{x}\tau_1 \ldots \vec{x}\tau_{|N|}\, \mathsf{OK}}\ \textsc{N-Cong}$$

$$\frac{x \notin \mathsf{FV}(E\, \tau_1 \ldots \tau_{|E|})}{x \in^? E\, \vec{x}\tau_1 \ldots \vec{x}\tau_{|E|}\, \mathsf{OK}}\ \textsc{E-FV}$$

$$\frac{x \in^? \tau_1\, \mathsf{OK} \;\; \ldots \;\; x \in^? \tau_{|E|}\, \mathsf{OK} \qquad \tau_1 \neq x \qquad E\, \vec{x}\tau_1 \ldots \vec{x}\tau_{|E|}\, \mathsf{INERT}}{x \in^? E\, \vec{x}\tau_1 \ldots \vec{x}\tau_{|E|}\, \mathsf{OK} \qquad \texttt{Ilya: (implicit } \alpha\texttt{-rename!)}}\ \textsc{E-Cong}$$

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:

Definition 12 (Inert Terms)**.**

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

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

As a heuristics, it is possible to extend the "Safe Occurrence" property by embedding *some* of the redex contractions from definition 7 into the inference system. Notice that only non-substituting contractions are allowed. This is because otherwise, we embed the full evaluation into the inference system, which makes it non-well-founded (and undecidable).

Definition 13 (Safe Occurrence Extension)**.**

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

`Ilya: The blue rules are experimental!`

$$\frac{x \notin \tau \qquad x \notin \nu \qquad x \in^? \sigma\ \mathsf{OK}}{x \in^? (\lambda y : \nu.\,\sigma)\tau\ \mathsf{OK}} \qquad \frac{x \notin \sigma \qquad x \notin \nu \qquad x \in^? \tau\ \mathsf{OK}}{x \in^? \mathsf{let}\ y : \nu\ :=\ \mathsf{return}\,\sigma\ \mathsf{in}\ \tau\ \mathsf{OK}}$$

$$\frac{x \notin \sigma \qquad x \notin \nu \qquad x \in^? \tau\ \mathsf{OK}}{x \in^? \mathsf{dlet}\ y : \nu\ :=\ \mathsf{return}\,\sigma\ \mathsf{in}\ \tau\ \mathsf{OK}}$$

Lemma 5 (Conguence of the safe occurrence)**.**

$$\frac{x \in^? F\,P_1 \ldots P_{|F|}\ \mathsf{OK}}{x \in^? P_1\ \mathsf{OK} \qquad \cdots \qquad x \in^? P_{|F|}\ \mathsf{OK}}$$

Proof. Trivial induction. $\square$

Lemma 6 (Reduction-Substitution Commutativity)**.**

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

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 5.

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.`

□

Corollary 1 (Normalization-Substitution Commutativity)**.**

$$\frac{x \in^? \sigma\,\mathsf{OK} \qquad \tau\,\mathsf{NF}}{\Downarrow(\sigma\{x := \tau\}) = (\Downarrow\sigma)\{x := \tau\}} \qquad\qquad \frac{x \in^? \sigma\,\mathsf{OK}}{\Downarrow(\sigma\{x := \tau\}) = (\Downarrow\sigma)\{x := \Downarrow\tau\}}$$

Proof. By replicating lemma 6. If τ is not in the normal form, proposition 2 is applied. □

Lemma 7 (Reduction preserves inertness)**.**

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

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

Lemma 8 (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 7.

- For the additional rules, the reduction can be either by congruence (and then we apply the induction hypothesis, lemma 5 and the same rule) or by the top-level redex contraction, and then the required property is exactly one of the premises.

□

Notice that *no occurrence means safe occurrence*, that is $x \in^? \tau\,\mathsf{OK}$ does not imply $x \in \mathsf{FV}(\tau)$. We use notation $x \in^! \tau\,\mathsf{OK}$ to denote $x \in^? \tau\,\mathsf{OK}$ *and* $x \in \mathsf{FV}(\tau)$.

Definition 14 (Strictly Safe Occurrence).

$$\frac{x \in^{?} \tau\,\mathsf{OK} \qquad x \in \mathsf{FV}(\tau)}{x \in^{!} \tau\,\mathsf{OK}}$$

Proposition 3 (Safe occurrence in a redex). *If τ is a top-level redex and $x \in^{!} \tau\,\mathsf{OK}$, then the redex is non-substitutive, i.e. the judgement $x \in^{?} \tau\,\mathsf{OK}$ was constructed by one of the rules from definition* 13

Lemma 9 (Reduction preserves strictly safe occurrence).

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

Proof. Induction on $\tau \to \tau'$. Since the redex contraction is special case of reduction, by lemma 8, it preserves the non-strict safe occurrence. Let us show that the actual occurrence $x \in \mathsf{FV}(\tau)$ is preserved as well.

x can only disappear after the contraction of the redex $r \subseteq \tau$ if x occurred inside this redex (i.e. $x \in \mathsf{FV}(r)$). By the trivial implication of lemma 5, since $r \subseteq \tau$ and $x \in^{?} \tau\,\mathsf{OK}$, $x \in^{?} r\,\mathsf{OK}$. Then by proposition 3, r can only be non-substitutive. However, non-substitutive redexes preserve the variables, hence, x could not have disappeared.

□

Definition 15 (Safe Existential Variable Set).

$$\mathsf{EV}^{\mathsf{OK}}(\rho) = \{\hat{x} \mid \hat{x} \in^{!} \rho\,\mathsf{OK}\}$$

Corollary 2 (Reduction preserves the Safe Existential Variable Set).

$$\frac{\tau \to \tau'}{\mathsf{EV}^{\mathsf{OK}}(\tau) = \mathsf{EV}^{\mathsf{OK}}(\tau')} \qquad \frac{\tau \Downarrow \tau'}{\mathsf{EV}^{\mathsf{OK}}(\tau) = \mathsf{EV}^{\mathsf{OK}}(\tau')}$$

Definition 16 (Existential Variable Set).

$$\mathsf{EV}(\rho) = \{\hat{x} \mid \hat{x} \in \mathsf{FV}(\rho)\}$$

Proposition 4 (Existential Variable Set is monotonous w.r.t. Reduction).

$$\frac{\tau \to \tau'}{\mathsf{EV}(\tau) \supseteq \mathsf{EV}(\tau')} \qquad \frac{\tau \Downarrow \tau'}{\mathsf{EV}(\tau) \supseteq \mathsf{EV}(\tau')}$$

8 Equivalence and Unification

Definition 17 (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 18 (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 19 (Binding Context)**.**

$$\Gamma ::= \cdot \;\mid\; \Gamma,\; x$$

We define equivalence on terms. Notice that the terms are not necessarily ground. The unification variables are treated as normal ones. However, since unification variables cannot be abstrated over, the binding context Γ consists of normal variables only.

Definition 20 (Equivalence)**.**

Reduction closure

$$\frac{\rho_1 \to \rho_1' \qquad \Gamma \vdash \rho_1' \equiv \rho_2}{\Gamma \vdash \rho_1 \equiv \rho_2}\ \textsc{Red-L} \qquad\qquad \frac{\rho_2 \to \rho_2' \qquad \Gamma \vdash \rho_1 \equiv \rho_2' \qquad \rho_1\ \mathsf{NF}}{\Gamma \vdash \rho_1 \equiv \rho_2}$$

Congruence

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

$$\frac{\Gamma, x \vdash \vec{x}^n \pi \equiv \vec{x}^n \rho}{\Gamma \vdash x.\vec{x}^n \pi \equiv x.\vec{x}^n \rho} \qquad\qquad \frac{\Gamma \vdash \sigma \equiv \rho}{\Gamma \vdash .\sigma \equiv .\rho}$$

Lemma 10 (Equivalent terms have equal normal forms)**.**

$$\frac{\Gamma \vdash \pi \equiv \rho}{\Downarrow\pi = \Downarrow\rho}$$

Lemma 11 (Equivalent terms have equal safe existential variable sets)**.**

$$\frac{\Gamma \vdash \pi \equiv \rho}{\mathsf{EV}^{\mathsf{OK}}(\pi) = \mathsf{EV}^{\mathsf{OK}}(\rho)}$$

Proof. Trivial induction using corollary 2. □

Definition 21 (Free Variable Environment)**.** *Free variable environment E is a (partial)* mapping *from unification variables to sets of regular variables. $E[\hat{v}]$ is a set Γ of variables associated with $\hat{v}$ that are allowed to be used in the initialization. Morally, Γ is a binding context at the moment when $\hat{v}$ was introduced.*

$$E ::= \cdot \;\mid\; E,\; \hat{v} \mapsto \Gamma$$

Definition 22 (Admissible term)**.**

$$\frac{\mathsf{FV}(\tau) \subseteq \Gamma}{\Gamma \vdash \tau}$$

Definition 23 (Unification Context)**.** *Unification context represents a (partial) solution of the unification problem. Syntactically, it is a* set *of pairs. Each pair represent an initialization of a unification variable:*

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

where τ is a ground term.

The unification (or algorithmic equivalence) judgement is of the form $E; \varphi \vdash \rho \equiv \rho' \dashv \varphi'$ where ρ and ρ' are algorithmic terms (potentially with unassigned variables), φ and φ' are unification contexts, E is a free variable environment.

Definition 24 (Admissible unification context)**.** *The unification context φ is admissible by the environment E if the term τ it assigns to the unification variable $\hat{v}$ is admissible by the set of variables $E[\hat{v}]$ (in particular, E is defined on $\hat{v}$).*

$$\frac{}{E \vdash \cdot} \qquad\qquad \frac{E[\hat{v}] \vdash \tau}{E \vdash \varphi, (\hat{v}{:=}\tau)}$$

Definition 25 (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 26 (Application of the well-formed context)**.** *If the unification context φ is well-formed, We write $[\varphi]\tau$ meaning the application of the partial (substitution) function represented by φ to the term τ:*

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

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 3 (Context application commutes with the reduction)**.**

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

$$\begin{tikzcd}
\rho \arrow[r, dashed, "\rightarrow"] \arrow[d, "\hat{x}_1{:=}\tau_1"'] & \rho^* \arrow[d, dashed, "\hat{x}_1{:=}\tau_1"] \\
\bullet \arrow[r, dashed, "\rightarrow"] \arrow[d, dotted, no head] & \rho^*_2 \arrow[d, dotted, no head] \\
\bullet \arrow[r, dashed, "\rightarrow"] \arrow[d, "\hat{x}_n{:=}\tau_n"'] & \rho^*_n \arrow[d, dashed, "\hat{x}_n{:=}\tau_n"] \\
{[\Omega]\rho} \arrow[r, "\rightarrow"] & \rho'
\end{tikzcd}$$

Figure 1: Proof scheme

Proof. Induction on $[\Omega]\rho$ using lemma 6. See fig. 1 for the details: we acquire ρ^* by consequently applying lemma 6 bottom-to-top to construct $\rho^*_n, \ldots, \rho^*_2, \rho^*$. The premises required for lemma 6 hold because $\vdash \Omega$ and $\rho\,\mathsf{OK}$. □

Corollary 4 (Context application preserves reduction)**.**

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

Proof. Induction on Ω using lemma 4. □

Corollary 5 (Reduction preserves safety)**.**

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

Proof. Follows from lemma 8. □

Definition 27 (Unification)**.** *The unification algorithm is defined as follows:*

Base rules

$$\frac{(\hat{v}{:=}\cdot) \notin \varphi \qquad \tau\ \mathsf{NF} \qquad E[\hat{v}] \vdash \tau}{\Gamma; E; \varphi \vdash \hat{v} \equiv \tau \dashv \varphi, (\hat{v}{:=}\tau)}\ \textsc{U-Add} \qquad \frac{(\hat{v}{:=}\tau) \in \varphi}{\Gamma; E; \varphi \vdash \hat{v} \equiv \tau \dashv \varphi}\ \textsc{U-Keep}$$

Reduction closure

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

Congruence

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

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

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 12 (Unification soundness)**.**

$$\frac{\vdash \varphi_1 \qquad \Gamma; E; \varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2 \qquad E \vdash \varphi_1 \qquad \mathsf{im}(E) \subseteq \Gamma}{\vdash \varphi_2 \qquad \Gamma \vdash [\varphi_2]\rho \equiv \tau \qquad \varphi_1 \subseteq \varphi_2 \qquad E \vdash \varphi_2}$$

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

- $\dfrac{(\hat{v}{:=}\cdot) \notin \varphi_1 \qquad \tau\ \mathsf{NF} \qquad E[\hat{v}] \vdash \tau}{\Gamma; E; \varphi_1 \vdash \hat{v} \equiv \tau \dashv \varphi_1, (\hat{v}{:=}\tau)}$ 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);
  - $[\varphi_2]\rho = [\varphi_1, (\hat{v}{:=}\tau)]\hat{v} = \tau$. $\Gamma \vdash \tau \equiv \tau$;
  - $\varphi_1 \subseteq \varphi_1, (\hat{v}{:=}\tau)$ by definition;
  - $E \vdash \varphi_1, (\hat{v}{:=}\tau)$ because $E \vdash \varphi_1$ and $E[\hat{v}] \vdash \tau$.
- $\dfrac{(\hat{v}{:=}\tau) \in \varphi}{\Gamma; E; \varphi \vdash \hat{v} \equiv \tau \dashv \varphi}$ 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;
  - $\Gamma \vdash [\varphi_2]\hat{v} \equiv \tau$ because $\vdash \varphi_2$ and $(\hat{v}{:=}\tau) \in \varphi_2$.

- $\varphi \subseteq \varphi$ trivially.
- $E \vdash \varphi_2$ because $\varphi_2 = \varphi_1$ and $E \vdash \varphi_1$ is in the lemma premises.

- $\dfrac{\rho \to \rho' \qquad \Gamma; E; \varphi_1 \vdash \rho' \equiv \tau \dashv \varphi_2}{\Gamma; E; \varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2}$ Then by the induction hypothesis: $\varphi_1 \subseteq \varphi_2$, $\vdash \varphi_2$, $E \vdash \varphi_2$, and $\Gamma \vdash [\varphi_2]\rho' \equiv \tau$. To prove that $\Gamma \vdash [\varphi_2]\rho \equiv \tau$ we apply Red-L:

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

Here $[\varphi_2]\rho \to [\varphi_2]\rho'$ holds by corollary 4.

- $\dfrac{\tau \to \tau' \qquad \Gamma; E; \varphi_1 \vdash \rho \equiv \tau' \dashv \varphi_2 \qquad \rho\,\mathsf{NF}}{\Gamma; E; \varphi_1 \vdash \rho \equiv \tau \dashv \varphi_2}$ Analogously to the previous case.

- $\dfrac{\begin{array}{c}\Gamma; E; \psi_0 \vdash P_1 \equiv Q_1 \dashv \psi_1 \\ \dots \qquad \Gamma; E; \psi_{|F|-1} \vdash P_{|F|} \equiv Q_{|F|} \dashv \psi_{|F|} \qquad F\,P_1 \dots P_{|F|}\,\mathsf{NF} \qquad F\,Q_1 \dots Q_{|F|}\,\mathsf{NF}\end{array}}{\Gamma; E; \psi_0 \vdash F\,\rho_1 \dots \rho_{|F|} \equiv F\,\tau_1 \dots \tau_{|F|} \dashv \psi_{|F|}}$

  Then $\varphi_1 = \psi_0$, $\varphi_2 = \psi_{|F|}$, $\rho = F\,P_1 \dots P_{|F|}$, $\tau = F\,Q_1 \dots Q_{|F|}$.

  We can apply the induction hypothesis to the first unification judgement in the premise (i.e. to $\Gamma; E; \psi_0 \vdash P_1 \equiv Q_1 \dashv \psi_1$) acquiring: $\vdash \psi_1$, $E \vdash \psi_1$, and $\Gamma \vdash [\psi_1]P_1 \equiv Q_1$. Then, because $\vdash \psi_1$, we can apply the induction hypothesis to the second premise. Continuing this process, we acquire:

  - $\varphi_1 = \psi_0 \subseteq \dots \subseteq \psi_{|F|} = \varphi_2$;
  - $\vdash \varphi_2$
  - $E \vdash \varphi_2$
  - $\Gamma \vdash [\psi_i]P_i \equiv Q_i$ for $i = 1 \dots |F|$. Hence, because $\psi_i \subseteq \varphi_2$, $\Gamma \vdash [\varphi_2]P_i \equiv Q_i$, which implies that $\Gamma \vdash [\varphi_2]F\,P_1 \dots P_{|F|} \equiv F\,Q_1 \dots Q_{|F|}$

- $\dfrac{\Gamma, x; E; \varphi_1 \vdash \vec{x}^n\rho \equiv \vec{x}^n\tau \dashv \varphi_2}{\Gamma; E; \varphi_1 \vdash x.\vec{x}^n\rho \equiv x.\vec{x}^n\tau \dashv \varphi_2}$

  Then we apply the induction hypothesis to $\Gamma, x; E; \varphi_1 \vdash \vec{x}^n\rho \equiv \vec{x}^n\tau \dashv \varphi_2$ and acquire

  - $\vdash \varphi_2$
  - $\varphi_1 \subseteq \varphi_2$
  - $E \vdash \varphi_2$
  - $[\varphi_2]\vec{x}^n\rho \equiv \vec{x}^n\tau$. By the Barendrecht's convention, $x \notin \Gamma$, hence, $x \notin \mathsf{im}(E)$. Since $E \vdash \varphi_2$, $x \notin \mathsf{FV}(\varphi_2)$. Therefore, $x.[\varphi_2]\vec{x}^n\rho = [\varphi_2]x.\vec{x}^n\rho$, which gives us the required equivalence. `Ilya: extract a lemma`

- $\dfrac{\Gamma; E; \varphi \vdash \rho \equiv \tau \dashv \psi}{\Gamma; E; \varphi \vdash .\rho \equiv .\tau \dashv \psi}$ Trivially by the induction hypothesis.

$\square$

Lemma 13 (Unification completeness)**.**

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

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

- $\dfrac{[\Omega]\rho \to \rho' \qquad \Gamma \vdash \rho' \equiv \tau}{\Gamma \vdash [\Omega]\rho \equiv \tau}$ By corollary 3, there exists ρ^* s.t. $\rho \to \rho^*$ and $[\Omega]\rho^* = \rho'$.

  By lemma 8, ρ^* OK. Then we apply the induction hypothesis to E, Ω, ρ^*, and τ. To acquire $\forall\varphi \subseteq \Omega.\ \exists\psi \subseteq \Omega.\ \Gamma; E; \varphi \vdash \rho^* \equiv \tau \dashv \psi$, where we can replace ρ^* with ρ by Red-L.

- $\dfrac{\tau \to \tau' \qquad \Gamma \vdash [\Omega]\rho \equiv \tau' \qquad [\Omega]\rho\,\mathsf{NF}}{\Gamma \vdash [\Omega]\rho \equiv \tau}$ We can apply the induction hypothesis to E, Ω, ρ, Γ, and τ' right away to acquire $\forall\varphi \subseteq \Omega.\ \exists\psi \subseteq \Omega.\ \Gamma; E; \varphi \vdash \rho \equiv \tau' \dashv \psi$, where we replace τ' with τ by Red-R.

- $\dfrac{\Gamma \vdash Q_1 \equiv Q'_1 \qquad \dots \qquad \Gamma \vdash Q_{|F|} \equiv Q'_{|F|} \qquad F\,Q_1 \dots Q_{|F|}\,\mathsf{NF} \qquad F\,Q'_1 \dots Q'_{|F|}\,\mathsf{NF}}{\Gamma \vdash F\,Q_1 \dots Q_{|F|} \equiv F\,Q'_1 \dots Q'_{|F|}}$ Then $[\Omega]\rho = F\,Q_1 \dots Q_{|F|}$ and $\tau = F\,Q'_1 \dots Q'_{|F|}$. Let us destruct $[\Omega]\rho$. `Ilya: we need a lemma to destruct it this way`

  - $\rho = \hat{x}$ and $(\hat{x}{:=}F\,Q_1 \dots Q_{|F|}) \in \Omega$. Let us consider an arbitrary $\varphi \subseteq \Omega$. φ is well-formed, then either $(\hat{x}{:=}F\,Q_1 \dots Q_{|F|}) \in \varphi$ or $(\hat{x}{:=}\cdot) \notin \varphi$.
    - $(\hat{x}{:=}F\,Q_1 \dots Q_{|F|}) \in \varphi$ then we take $\psi = \varphi$ and apply U-Keep.
    - $(\hat{x}{:=}\cdot) \notin \varphi$ then we take $\psi = \varphi, (\hat{x}{:=}F\,Q_1 \dots Q_{|F|})$ and apply U-Add. The term $F\,Q_1 \dots Q_{|F|}$ is in the normal form by one of the premises. $E[\hat{x}] \vdash F\,Q_1 \dots Q_{|F|}$ because $E \vdash \Omega \ni (\hat{x}{:=}F\,Q_1 \dots Q_{|F|})$.
  - $\rho = F\,P_1 \dots P_{|F|}$, $F \neq \hat{x}$, $[\Omega]P_i = Q_i$ and $\Gamma \vdash Q_i \equiv Q'_i$ for $i = 1 \dots |F|$.

    By the corollary of lemma 5, P_i OK. So we can apply the induction hypothesis to all the components to acquire $|F|$ facts: $\forall\varphi \subseteq \Omega.\ \exists\psi \subseteq \Omega.\ \Gamma; E; \varphi \vdash P_i \equiv Q'_i \dashv \psi$.

    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: $\Gamma; E; \psi_0 \vdash P_1 \equiv Q'_1 \dashv \psi_1$, $\dots$, $\Gamma; E; \psi_{|F|-1} \vdash P_{|F|} \equiv Q'_{|F|} \dashv \psi_{|F|}$.

    Notice that by lemma 4 and proposition 1, $F\,P_1 \dots P_{|F|}\,\mathsf{NF}$. Then we apply the congruence unification rule and get $\Gamma; E; \varphi \vdash F\,P_1 \dots P_{|F|} \equiv F\,Q'_1 \dots Q'_{|F|} \dashv \psi_{|F|}$, i.e. $\Gamma; E; \varphi \vdash \rho \equiv \tau \dashv \psi_{|F|}$, so we take $\psi_{|F|}$ as ψ.

- $\dfrac{\Gamma, x \vdash \vec{x}^n\sigma \equiv \vec{x}^n\tau}{\Gamma \vdash x.\vec{x}^n\sigma \equiv x.\vec{x}^n\tau}$

  Then $[\Omega]\rho = x.\vec{x}^n\sigma$, which means that $\rho = x.\vec{x}^n\rho'$, and $[\Omega]\vec{x}^n\rho' = \vec{x}^n\sigma$. Notice that $\vec{x}^n\rho'$ OK, which means the induction hypothesis is applicable and gives us $\forall\varphi \subseteq \Omega.\ \exists\psi \subseteq \Omega.\ \Gamma, x; E; \varphi \vdash \vec{x}^n\rho' \equiv \vec{x}^n\tau \dashv \psi$. Then we can apply the corresponding unification rule to get the required unification judgement: $\Gamma; E; \varphi \vdash x.\vec{x}^n\rho' \equiv x.\vec{x}^n\tau \dashv \psi$.

- $\dfrac{\sigma \equiv \tau}{.\sigma \equiv .\tau}$ Trivially by the induction hypothesis.

$\square$

9 Typing

Definition 28 (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

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

9.2 Context Formation and Var

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

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

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

9.3 Subsumption

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

9.4 Universes

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

$$\frac{\Gamma \vdash A\ \mathsf{vtype} \qquad \Gamma, x : A \vdash X\ \mathsf{ctype} \qquad x \in^! X\ \mathsf{OK}}{\Gamma \vdash \forall x : A.\, X\ \mathsf{ctype}}\ \forall \qquad \frac{\Gamma \vdash A\ \mathsf{vtype} \qquad \Gamma, x : A \vdash B\ \mathsf{vtype}}{\Gamma \vdash \Sigma x : A.\, B\ \mathsf{vtype}}\ \Sigma$$

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

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

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

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

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

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

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

9.7 $\forall$, Π, and Σ

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

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

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

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

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

9.8 Equality

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

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

10 Declarative Subtyping

Ilya: Value subtyping is symmetric!!!

$$\frac{a\ \mathsf{vtype} \in \Gamma}{\Gamma \vdash a \leqslant^v a}\ \leqslant\textsc{Var} \qquad \frac{\Gamma \vdash A_1 \leqslant^v B_1 \qquad \Gamma, x : A_1 \vdash A_2 \leqslant^v B_2}{\Gamma \vdash \Sigma x : A_1.\, A_2 \leqslant^v \Sigma x : B_1.\, B_2}\ \leqslant\Sigma$$

$$\frac{\Gamma \vdash A_2 \leqslant^v A_1 \qquad \Gamma, x : A_2 \vdash X_1 \leqslant^c X_2}{\Gamma \vdash \Pi x : A_1.\, X_1 \leqslant^c \Pi x : A_2.\, X_2}\ \leqslant\Pi \qquad \frac{\Gamma \vdash X_1 \leqslant^c X_2 \qquad \Gamma \vdash X_2 \leqslant^c X_1}{\Gamma \vdash \downarrow X_1 \leqslant^v \downarrow X_2}\ \leqslant\mathcal{U}$$

$$\frac{\Gamma \vdash A_1 \leqslant^v A_2 \qquad \Gamma \vdash A_2 \leqslant^v A_1}{\Gamma \vdash \uparrow A_1 \leqslant^c \uparrow A_2}\ \leqslant\mathcal{F} \qquad \frac{\Gamma \vdash A \leqslant^v B \qquad \mathsf{vars}(\Gamma) \vdash v_1 \equiv v_2 \qquad \mathsf{vars}(\Gamma) \vdash w_1 \equiv w_2}{\Gamma \vdash \mathsf{eq} A\, v_1\, w_1 \leqslant^v \mathsf{eq} B\, v_2\, w_2}\ \leqslant\textsc{Eq}$$

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

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

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

11 The Safe Variable Invariant

In section 12, we will prove the exhaustiveness of the algorithmic subtyping. For this purpose, we need to provide a certain invariant of how the existential variables are distributed in the well-formed types.

Definition 29 (The Trace of the Term)**.** *Suppose ρ is a term. Let us define $\mathsf{tr}(\rho)$ as*

$$\mathsf{tr}(\rho) = \mathsf{EV}(\Downarrow\rho)$$

Definition 30 (The Contextual Background of the Term)**.** *Suppose Γ is a context, ρ is a term. Let us define $\mathsf{bg}(\Gamma, \rho)$ as the minimal set of ground variables, such that*

1. $\mathsf{FV}(\Downarrow\rho) \setminus \mathsf{EV}(\Downarrow\rho) \subseteq \mathsf{bg}(\Gamma, \rho)$
2. *if $x \in \mathsf{bg}(\Gamma, \rho)$ and $(x : \pi) \in \Gamma$ then $\mathsf{FV}(\Downarrow\pi) \setminus \mathsf{EV}(\Downarrow\pi) \subseteq \mathsf{bg}(\Gamma, \rho)$.*

Intuitively, we take an empty set, add all the non-existential variables of ρ into it, and recursively repeat this process for the types of the added variables.

Definition 31 (The Contextual Trace of the Term)**.** *Suppose Γ is a context, ρ is a term. Let us define $\mathsf{tr}(\Gamma, \rho)$ as*

$$\mathsf{tr}(\Gamma, \rho) = \mathsf{tr}(\rho) \cup \bigcup_{(x:\pi) \leftarrow \Gamma,\ x \leftarrow \mathsf{bg}(\Gamma, \rho)} \mathsf{tr}(\pi)$$

Ilya: the old definition:

$$\mathsf{tr}(\Gamma, \rho) = \mathsf{tr}(\rho) \cup \{\mathsf{tr}(\pi) \mid (x : \pi) \leftarrow \Gamma,\ x \leftarrow \mathsf{FV}(\Downarrow\rho) \setminus \mathsf{EV}(\Downarrow\rho)\}$$

11.1 Trace Transferred from Types

Definition 32 (Reticent Terms Syntax).

$$\begin{array}{llcl} \textit{Reticent Values} & r_v & ::= & \hat{x} \mid \{r_c\} \\ \textit{Reticent Computations} & r_c & ::= & \mathsf{return}\, r_v \mid \mathsf{let}\ x : A := t \ \mathsf{in}\ r_c \mid \mathsf{dlet}\ x : A := t \ \mathsf{in}\ r_c \end{array}$$

Proposition 5 (Inferred Types of Reticent Terms).

$$\frac{\Gamma \vdash_v r_v \Rightarrow A}{r_v = \hat{x} \quad or \quad A = \downarrow X} \qquad\qquad \frac{\Gamma \vdash_c r_c \Rightarrow X}{X = \uparrow A \quad or \quad X = \mathsf{let}\ x : A := t \ \mathsf{in}\ Y}$$

Lemma 14 (Inferred Types Trace). *Intuitively, the non-reticent terms (contextually) expose the trace of the inferred types.*

$$\frac{v\,\mathsf{OK} \qquad \Gamma \vdash_v v \Rightarrow A}{\mathsf{tr}(A) \subseteq \mathsf{tr}(\Gamma, v) \ \ or\ v = r_v} \qquad\qquad \frac{t\,\mathsf{OK} \qquad \Gamma \vdash_c t \Rightarrow X}{\mathsf{tr}(X) \subseteq \mathsf{tr}(\Gamma, t) \ \ or\ t = r_c}$$

Proof. Mutual induction. □

11.2 Trace Transferred from Context

As the declarative type formation inference rules are intertwined with typing and subtyping, the inductive proof forces us to augment the proved invariant. Specifically, the invariant we are proving is spread across the following three lemmas. The proof should be read as *one mutual induction.*

Lemma 15 (The Safe Variable Invariant for Type Formation).

$$\frac{\Gamma[\hat{x} : A] \vdash X\ \mathsf{ctype} \qquad \hat{x} \in^! X\,\mathsf{OK}}{\mathsf{tr}(\Gamma, A) \subseteq \mathsf{tr}(\Gamma, X)} \qquad\qquad \frac{\Gamma[\hat{x} : A] \vdash B\ \mathsf{vtype} \qquad \hat{x} \in^! B\,\mathsf{OK}}{\mathsf{tr}(\Gamma, A) \subseteq \mathsf{tr}(\Gamma, B)}$$

Lemma 16 (The Safe Variable Invariant for Typing).

$$\frac{v\,\mathsf{OK} \qquad \Gamma[\hat{x} : A] \vdash_v v \Leftarrow C \qquad \hat{x} \in^! v\,\mathsf{OK}}{\mathsf{tr}(\Gamma, A) \subseteq \mathsf{tr}(\Gamma, v) \cup \mathsf{tr}(\Gamma, C)} \qquad\qquad \frac{e\,\mathsf{OK} \qquad \Gamma[\hat{x} : A] \vdash_v e \Leftarrow Y \qquad \hat{x} \in^! e\,\mathsf{OK}}{\mathsf{tr}(\Gamma, A) \subseteq \mathsf{tr}(\Gamma, e) \cup \mathsf{tr}(\Gamma, Y)}$$

$$\frac{v\,\mathsf{OK} \qquad \Gamma[\hat{x} : A] \vdash_v v \Rightarrow C \qquad \hat{x} \in^! v\,\mathsf{OK}}{\mathsf{tr}(\Gamma, A) \subseteq \mathsf{tr}(\Gamma, v) \cup \mathsf{tr}(\Gamma, C)} \qquad\qquad \frac{e\,\mathsf{OK} \qquad \Gamma[\hat{x} : A] \vdash_v e \Rightarrow Y \qquad \hat{x} \in^! e\,\mathsf{OK}}{\mathsf{tr}(\Gamma, A) \subseteq \mathsf{tr}(\Gamma, e) \cup \mathsf{tr}(\Gamma, Y)}$$

Lemma 17 (The Safe Variable Invariant for Subtyping).

$$\frac{\Gamma \vdash A \leqslant^v B}{\mathsf{tr}(\Gamma, A) = \mathsf{tr}(\Gamma, B)} \qquad\qquad \frac{\Gamma \vdash X \leqslant^c Y}{\mathsf{tr}(\Gamma, X) \subseteq \mathsf{tr}(\Gamma, Y)}$$

Proof of lemma 15. Mutual (with lemmas 16 and 17) induction on the type formation judgement.

VarT Holds vacuously, since $\hat{x} \notin^! a$.

□

Proof of lemma 17. Mutual (with lemmas 15 and 16) induction on the subtyping judgement.

$\leqslant$**Var** Trivial.

$\leqslant \Sigma$ By the induction hypothesis and congruence of $\mathsf{EV}()$ and $\Downarrow$.

$\leqslant \Pi$ Similar to the $\leqslant \Sigma$ case. It is essential that the proved property is symmetrical for the value-subtyping.

$\leqslant \mathcal{U}$ By the induction hypothesis. The symmetry of the premise of the rule is essential.

$\leqslant \mathcal{F}$ By the induction hypothesis. The symmetry of the premise is *not* essential here.

$\leqslant$**Eq** By the induction hypothesis and lemma 10.

$\leqslant \forall$ $\mathsf{tr}(X) \subseteq \mathsf{tr}(Y) \subseteq \mathsf{tr}(\forall y : A. Y)$.

let$\leqslant$ **and** $\leqslant$**let** After one evaluation step, the proven inclusion coincides with the induction hypothesis.

let$\leqslant$**let** By congruence, the induction hypothesis, and lemma 10.

$\forall \leqslant$

$$\begin{aligned}
& \mathsf{tr}(\forall x : A. X) && \text{(by congruence)} \\
\subseteq\ & \mathsf{tr}(A) \cup \mathsf{tr}(X) && \text{(by lemma 15, because } \Gamma, x : A \vdash X \ \mathsf{ctype}) \\
\subseteq\ & \mathsf{tr}(X) \\
\subseteq\ & \mathsf{EV}((\Downarrow X)\{x := \Downarrow v\}) && \text{(by corollary 1)} \\
=\ & \mathsf{tr}((X\{x := v\})) && \text{(by the IH)} \\
\subseteq\ & \mathsf{tr}(Y)
\end{aligned}$$

□

12 Algorithmic Subtyping

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

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

$$\frac{\Gamma; E; \varphi \vdash X_2 \leqslant^c X_1 \dashv \varphi' \qquad \Gamma; E; \varphi' \vdash X_1 \leqslant^c [\varphi']X_2 \dashv \varphi''}{\Gamma; E; \varphi \vdash \downarrow X_1 \leqslant^v \downarrow X_2 \dashv \varphi''} \leqslant \mathcal{U}$$

$$\frac{\Gamma; E; \varphi \vdash A_2 \leqslant^v A_1 \dashv \varphi' \qquad \Gamma; E; \varphi' \vdash [\varphi']A_1 \leqslant^v A_2 \dashv \varphi''}{\Gamma; E; \varphi \vdash \uparrow A_1 \leqslant^c \uparrow A_2 \dashv \varphi''} \leqslant \mathcal{F}$$

$$\frac{\Gamma; E; \varphi \vdash A \leqslant^v B \dashv \varphi' \qquad \mathsf{vars}(\Gamma); E; \varphi' \vdash v_2 \equiv v_1 \dashv \varphi'' \qquad \mathsf{vars}(\Gamma); E; \varphi'' \vdash w_2 \equiv w_1 \dashv \varphi'''}{\Gamma \vdash \mathsf{eq} A\, v_1\, w_1 \leqslant^v \mathsf{eq} B\, v_2\, w_2} \leqslant \text{EQ}$$

$$\frac{\Gamma; E, (\hat{v} \mapsto \mathsf{vars}(\Gamma)); \varphi \vdash X\{x \rightsquigarrow \hat{v}\} \leqslant^c Y \dashv \varphi'}{\Gamma; E; \varphi \vdash (\forall x : A.\, X) \leqslant^c Y \dashv \varphi'} \forall\leqslant \qquad \frac{\Gamma, y : A; E; \varphi \vdash X \leqslant^c Y \dashv \varphi'}{\Gamma; E; \varphi \vdash X \leqslant^c (\forall y : A.\, Y) \dashv \varphi'} \leqslant\forall$$

$$\frac{\mathsf{vars}(\Gamma) \vdash e \equiv \mathsf{return}\, v \qquad \Gamma; E; \varphi \vdash X\{x := v\} \leqslant^c Y \dashv \varphi'}{\Gamma; E; \varphi \vdash (\mathsf{let}\, x : A := e\, \mathsf{in}\, X) \leqslant^c Y \dashv \varphi'} \text{LET}\leqslant$$

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

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

12.1 Invariants

We should be able to infer $\varphi \vdash (\mathsf{let}\, x : \uparrow \mathsf{eq} A\, \hat{u}\, \hat{v} := \hat{w}\, \mathsf{in} \uparrow Int) \leqslant^c \uparrow Int \dashv \varphi$. 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.`

13 Properties

`Ilya: Outdated`

Lemma 18 (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 19 (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 20 (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 19.

(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 $\dagger$ and $\ddagger$ 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 (`ΣE`) doesn't work...`

$\square$