A Relational Framework for Higher-Order Shape Analysis

Appendix

1. Core language

1.1 Definitions

Our meta-theory relies on several definitions, which are stated below:

Definition *Simply Typed* λ_R We define simply typed λ_R whose type rules are same as those of simply typed lambda calculus. The rules are reproduced in Fig. 1 for the sake of completeness. The rules reuse Γ to denote simple type environment (against dependent type environment in the dependent type rules of λ_R) that maps variables to their (unrefined) types. Recall that the domain of a λ_R relation is a simple type. The S-APP rule, which sort checks relation applications, uses simple typing judgment to type check the argument.

The relationship between dependent typing judgment and simple typing judgment is established in Lemma 1.8. Since typing judgment under a context Γ relies on well-sortedness of relation applications under same Γ, via well-formedness judgment, using simple typing judgment instead of dependent typing judgment avoids any circular reasoning.

Definition ***(Primitive Types of Constants)*** Just as the dependent typing judgment makes use of a function ty that maps constants to their dependent types, simple typing judgment makes use of the function pty that maps constants to their primitive types. It is defined as following:

$$\begin{array}{lcl} \forall i \in \mathbb{Z},\, pty(i) & = & \mathsf{int} \\ pty(\mathsf{Nil}) & = & \mathsf{intlist} \\ pty(\mathsf{Cons}) & = & \mathsf{int} \to \mathsf{intlist} \to \mathsf{intlist} \end{array}$$

Definition ***(VC Prelude)*** Sec. 3.3 of the paper introduces Γ_R as an ordered map from structural relations to their colon-arrow sorts. MSFOL translation of Γ_R (i.e., $[\![\Gamma_R]\!]_L$) is a set of assertions that assert sorts of uninterpreted relations in MSFOL. Since this set forms the context for verification conditions generated by subtype judgment in λ_R, we call the set as VC prelude.

Formally, $[\![\Gamma_R]\!]_L$ is the smallest set of MSFOL formulas such that forall R, $\cdot \vdash R :: T :\to \{\theta\}$,

$$R : [\![T :\to \{\theta\}]\!]_L \in [\![\Gamma_R]\!]_L$$

Remark ***(Entailment)*** In our proofs, we use $\models_L$ to denote semantic entailment in first-order logic. We write $\phi_1 \models_L \phi_2$ to denote that ϕ_1 semantically entails ϕ_2. Since several deductive systems, such as sequent calculus and natural deduction, are complete for first-order logic, we abuse $\models_L$ notation to also denote logical consequence. However, instead of using a set of hypotheses to the left of $\models_L$, we use a sequence (i.e., ordered set Γ) of hypotheses. We adapt few standard theorems of first-order deductive systems to our setting:

- Deduction Theorem: $\Gamma,\, \phi_1 \models_L \phi_2$ is equivalent to $\Gamma \models_L \phi_1 \Rightarrow \phi_2$.
- Monotonicity of Entailment (or Thinning): if $\Gamma \models_L \phi$, then forall Γ', $\Gamma', \Gamma \models_L \phi$
- Cut Elimination: If $\Gamma_1 \models_L \phi_1$, and $\Gamma_1, \phi_1, \Gamma_2 \models_L \phi_2$, then $\Gamma_1, \Gamma_2 \models_L \phi_2$

Definition *Free Variables* We define a function $freevars$ that returns a set of free variables in expressions (e) and type refinements (ϕ) of λ_R. For expressions, the definition of $freevars$ is straightforward, and follows that of simply typed lambda calculus. For a type refinement ϕ, $freevars(\phi)$ returns the union of $freevars$ of all λ_R values (v) that occur as arguments to relations in type refinements.

Definition *Substitution* Substitution operation substitutes a λ_R value (v) for a variable (z) in a λ_R expression (e), or a λ_R type refinement (ϕ). The definition of capture avoiding substitution for λ_R expressions is standard. The only caveat is that the substitution should also be performed on the type annotations occurring within expressions:

$$\begin{array}{lcll} [v/z]\,\lambda(x:\tau).\,e & = & \lambda(x:[v/z]\,\tau).\,e & \text{if } z = x \\ [v/z]\,\lambda(x:\tau).\,e & = & [v/z]\,alphaConvert(\lambda(x:\tau).\,e) & \text{if } x \in freevars(v) \\ [v/z]\,\lambda(x:\tau).\,e & = & \lambda(x:[v/z]\,\tau).\,[v/z]\,e & \textit{otherwise} \end{array}$$

ST-VAR

$$\frac{(x:T) \in \Gamma}{\Gamma \Vdash x : T}$$

ST-APP

$$\frac{\Gamma \Vdash e : T_1 \to T_2 \quad \Gamma \Vdash v : T_1}{\Gamma \Vdash e\, v : T_2}$$

ST-ABS

$$\frac{T_1 = \|\tau_1\| \quad \Gamma, x : T_1 \Vdash e : T_2}{\Gamma \Vdash \lambda(x:\tau_1).\, e : T_1 \to T_2}$$

ST-CONST

$$\frac{}{\Gamma \Vdash c : pty(c)}$$

ST-LET

$$\frac{\Gamma \Vdash e_1 : T_1 \quad \Gamma, x : T_1 \Vdash e_2 : T_2}{\Gamma \Vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : T_2}$$

ST-MATCH

$$\frac{\Gamma \Vdash v : \mathsf{intlist} \quad \Gamma_c = x : \mathsf{int}, y : \mathsf{intlist} \qquad \Gamma, \Gamma_c \Vdash e1 : T \quad \Gamma \Vdash e2 : T}{\Gamma \Vdash \texttt{match}\ v\ \texttt{with}\ \mathsf{Cons}\, x\, y \Rightarrow e_1\ \texttt{else}\ e_2 : T}$$

Figure 1: Type Rules for simply typed λ_R

Evaluation Rules $\boxed{\text{e1} \longrightarrow \text{e2}}$

$$
\begin{array}{rcll}
(\lambda(x:\tau).\, e)\, v & \longrightarrow & [v/x]\, e & \text{E-}\beta \\
\texttt{let}\ x = v\ \texttt{in}\ e & \longrightarrow & [v/x]\, e & \text{E-LET} \\
(\texttt{match}\ \mathsf{Cons}\, v_1\, v_2\ \texttt{with}\ \mathsf{Cons}\, x\, y \Rightarrow e_1\ \texttt{else}\ e_2) & \longrightarrow & [v_2/y][v_1/x]\, e_1 & \text{E-MCONS} \\
(\texttt{match}\ \mathsf{Nil}\ \texttt{with}\ \mathsf{Cons}\, x\, y \Rightarrow e_1\ \texttt{else}\ e_2) & \longrightarrow & e_2 & \text{E-MNIL}
\end{array}
$$

$$\frac{e_1 \longrightarrow e_2}{E[e_1] \longrightarrow E[e_2]} \qquad \text{E-CO}$$

Evaluation Context $\boxed{\text{E}}$

$$E \ ::= \ \bullet \mid \bullet\, v \mid \texttt{let}\ x = \bullet\ \texttt{in}\ e$$

Figure 2: Operational Semantics of Core Calculus (λ_R)

Substitution operation for types is defined in terms of substitution operation for type refinements. For function types, capture avoidance property needs to be explicitly ensured:

$$
\begin{array}{lcll}
[v/z]\, \{\nu : T \mid \phi\} & = & \{\nu : T \mid [v/z]\, \phi\} & \\
[v/z]\, (x:\tau_1) \to \tau_2 & = & (x : [v/z]\, \tau_1) \to \tau_2 & \text{if } z = x \\
[v/z]\, (x:\tau_1) \to \tau_2 & = & [v/z]\, alphaConvert((x:\tau_1) \to \tau_2) & \text{if } x \in freevars(v) \\
[v/z]\, (x:\tau_1) \to \tau_2 & = & (x : [v/z]\, \tau_1) \to [v/z]\, \tau_2 & \text{otherwise}
\end{array}
$$

We assume a function $alphaConvert$ that performs alpha renaming of bound variable for abstraction expressions and function types. Substitution operation for type refinements is recursively defined in terms of relational predicates, and relational expressions. For relation application expressions, substitution is performed on the value (v') to which the relation is being applied:

$$[v/z]\, R(v') \quad = \quad R([v/z]\, v')$$

Definition ***(Basic Axioms)*** Basic axioms assert sorts, and validity of type refinements of constants in MSFOL. This accounts for theT-CONST rule of λ_R type system, which seeds the typing judgment with assumptions on types of constants. The axioms are stated below:

- Type of Integers: Forall integer constants c, $\models_L\ c : [\![\mathsf{int}]\!]_{\mathsf{L}}$
- Type of Nil: $\models_L\ \mathsf{Nil} : [\![\mathsf{intlist}]\!]_{\mathsf{L}}$
- Type of Cons: $x : [\![\mathsf{int}]\!]_L,\ y : [\![\mathsf{intlist}]\!]_L\ \models_L\ \mathsf{Cons}\ x\ y : [\![\mathsf{intlist}]\!]_{\mathsf{L}}$
- Validity of ϕ_n : $[\![\Gamma_R]\!]_L\ \models_L\ [\mathsf{Nil}/\nu]\, \phi_n$
- Validity of ϕ_c : $[\![\Gamma_R]\!]_L,\ x : [\![\mathsf{int}]\!]_L,\ y : [\![\mathsf{intlist}]\!]_L\ \models_L\ [\mathsf{Cons}\ x\ y/\nu]\, \phi_c$

1.2 Type Safety

We now prove[1] the type safety of λ_R's type system by proving its progress and preservation properties. Call-by-value operational semantics of λ_R are given in Fig. 2.

THEOREM 1.1. (**Progress**) *If* $\cdot \vdash e : \tau$*, then either* e *is a value or there exists an* e' *such that* $e \longrightarrow e'$.

Proof By induction on type derivation. Cases:

- Case T-VAR: e is a variable x. By inversion on $\cdot \vdash x : \tau$ we get $x : \tau \in \cdot$, which is absurd. Proof follows from *ex falso quodlibet*.
- Case T-CONST: e is a constant c, which is a value.
- Case T-APP: e is of form $e_1\ v$, where $\cdot \vdash e_1 : (x : \tau_1) \to \tau_2$, and $\cdot \vdash v : \tau_1$. By IH, either e_1 can take a step or e_1 is a value.
  - $e_1 \longrightarrow e_2$. As per our definition of evaluation contexts (Fig. 2), $e_1\ v \longrightarrow e_2\ v$; So, $e' = e_2\ v$.
  - e_1 is a value v_1. By inversion on v_1, followed by eliminating cases using the assumption $\cdot \vdash v_1 : (x : \tau_1) \to \tau_2$, we are left with following cases:
    - v_1 is of form $\lambda x : \tau.\, e_3$, for some e_3: Consequently, $e = \lambda x : \tau.\, e_3\ v$, which reduces by E-$\beta$ to $[v/x]e_3$
    - v_1 is Cons: $\mathsf{Cons}\ v$ is a value.
    - v_1 is $\mathsf{Cons}\ v_2$, for some v_2: $\mathsf{Cons}\ v_2\ v$ is a value.
- Case T-ABS: $e = \lambda x : \tau.\, e_1$, which is a value.
- Case T-LET: $e = \texttt{let}\ x = e_1\ \texttt{in}\ e_2$, where $\cdot \vdash e_1 : \tau_1$, and $\cdot, x : \tau_1 \vdash e_2 : \tau_2$. Similar to T-APP, we have two cases:
  - $e_1 \longrightarrow e_1'$, in which case $e \longrightarrow \texttt{let}\ x = e_1'\ \texttt{in}\ e_2$ (as per our definition of evaluation contexts).
  - e_1 is a value v_1, in which case e reduces by rule E-LET to $[v_1/x]e_2$.
- Case T-SUB: $\cdot \vdash e : \tau'$, where $\tau' <: \tau$. Proof follows from IH.
- Case T-MATCH: $e = \texttt{match}\ v\ \texttt{with Cons}\ x\ y \Rightarrow e_1\ \texttt{else}\ e_2$, where $\cdot \vdash v : \{\nu : \mathsf{intlist} \mid \phi\}$. By inversion on v and eliminating absurd cases, we are left with two cases:
  - $v = \texttt{Cons}\ v_1\ v_2$, in which case e reduces by E-MCONS to $[v_1/x][v_2/y]e_1$.
  - $v = \texttt{Nil}$, in which case e reduces by M-ENIL to e_2.

∎

LEMMA 1.2. (**Context Invariance for Well-Formedness**) *If* $\Gamma \vdash \tau$*, then forall* Γ' *such that* $\|\Gamma'\| = \|\Gamma\|$, $\Gamma' \vdash \tau$

Proof Since well-formedness of a type directly derives from well-sortedness of relational expressions occurring in its type refinement, It suffices to prove that:

$$\text{forall } r\text{, if } \Gamma \vdash r :: \{\theta\}\text{, then forall } \Gamma' \text{ such that } \|\Gamma'\| = \|\Gamma\|,\ \Gamma' \vdash r :: \{\theta\}.$$

We prove this by induction on $\Gamma \vdash r :: \{\theta\}$. Cases S-UNION and S-CROSS follow directly from inductive hypotheses. The only interesting case is S-APP, where $r = R(v)$, for some R and v. The proof is by inversion on $\Gamma \vdash R(v) :: \{\theta\}$. Hypotheses:

$$\begin{array}{ll} \cdot \vdash R :: T :\to \{\theta\} & (H1) \\ \|\Gamma\| \Vdash v : T & (H2) \end{array}$$

Rewriting $H2$ using $\|\Gamma\| = \|\Gamma'\|$:

$$\|\Gamma'\| \Vdash v : T \quad (H3)$$

Applying S-APP rule over $H1$ and $H3$ proves the goal. ∎

LEMMA 1.3. (**Cut Elimination**) *Forall* x, e, τ, ϕ, *and* Γ, *If* $[\![\Gamma_R]\!]_L \models_L [\![\phi]\!]_L$, *and* $\phi, \Gamma \vdash e : \tau$, *then* $\Gamma \vdash e : \tau$

Proof by induction on $\phi, \Gamma \vdash e : \tau$. Cases:

- Case T-VAR : e is a variable y. Inversion of $\phi, \Gamma \vdash y : \tau$ produces $y : \tau \in \phi, \Gamma$. From the definition of type environment, it follows that $y : \tau \in \Gamma$. Applying T-VAR produces proof.

[1] A note on the notation adapted in writing proofs: Most proofs are by induction or inversion, leading to cases. Hypotheses and inductive hypotheses are named as per $H[0-9]+$ and $IH[0-9]+$ grammars, respectively. Names refer to a different hypotheses in different cases. Coq tactic names are used used to convey proof strategy, wherever applicable.

- Case T-CONST: e is a constant c. Proof trivial, as constants have same type irrespective of the context.
- Case T-SUB: Hypotheses:

$$\begin{array}{cl} \tau = \tau_2 & (H0) \\ [\![\Gamma_R]\!]_L \models_L [\![\phi]\!]_L & (H1) \\ \phi, \Gamma \vdash \tau_1 <: \tau_2 & (H3) \end{array}$$

  Inductive hypothesis is:

$$\Gamma \vdash e : \tau_1 \quad (IH0)$$

  It remains to show that $\Gamma \vdash \tau_1 <: \tau$, which we prove by induction on subtype derivation in $H3$. Cases:
  - SubCase SUBT-BASE: Hypotheses:

$$\begin{array}{cl} \tau_1 = \{\nu : T \,|\, \phi_1\} & (H4) \\ \tau_2 = \{\nu : T \,|\, \phi_2\} & (H5) \\ \phi, \Gamma \vdash \{\nu : T \,|\, \phi_1\} & (H6) \\ \phi, \Gamma \vdash \{\nu : T \,|\, \phi_2\} & (H7) \\ [\![\Gamma_R]\!]_L \models_L [\![\phi]\!]_L \Rightarrow [\![\Gamma]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi_2]\!]_L & (H8) \end{array}$$

    Since $\|\phi, \Gamma\| = \|\Gamma\|$, we can apply Lemma 1.2 to derive the following from $H6-7$:

$$\begin{array}{cl} \Gamma \vdash \{\nu : T \,|\, \phi_1\} & (H8) \\ \Gamma \vdash \{\nu : T \,|\, \phi_2\} & (H9) \end{array}$$

    From $H8$, using deduction theorem of first order logic, we obtain:

$$[\![\Gamma_R]\!]_L,\, [\![\phi]\!]_L,\, [\![\Gamma]\!]_L,\, [\![\phi_1]\!]_L \models_L [\![\phi_2]\!]_L \quad (H10)$$

    We apply cut elimination theorem of logical consequence in first-order logic to $H1$ and $H10$ and derive:

$$[\![\Gamma_R]\!]_L,\, [\![\Gamma]\!]_L,\, [\![\phi_1]\!]_L \models_L [\![\phi_2]\!]_L \quad (H11)$$

    Using the the deduction, $H11$ is equivalent to:

$$[\![\Gamma_R]\!]_L \models_L [\![\Gamma]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi_2]\!]_L \quad (H12)$$

    Finally, applying SUBT-BASE rule to $H8$,$H9$, and $H12$ leads us to conclude that $\Gamma \vdash \tau_1 <: \tau_2$
  - Case SUBT-ARROW: One can derive proof for this case by simply applying SUBT-ARROW to inductive hypotheses.
- Cases T-APP, T-ABS, T-LET, T-MATCH: Proofs for these cases follow straightforwardly from respective inductive hypotheses.

∎

LEMMA 1.4. (λ_R **Type System is Conservative**) *forall* Γ, *if* $\Gamma \vdash e : \tau$ *then* $\|\Gamma\| \Vdash e : \|\tau\|$.

Proof By induction on $\Gamma \vdash e : \tau$. Cases:

- Case T-VAR: e is a variable x. Hypotheses:

$$(x : \tau) \in \Gamma$$

  From the definition of $\|\Gamma\|$, we know that $(x : \|\tau\|) \in \|\Gamma\|$. Applying ST-VAR gives the proof.
- Case T-CONST:e is a constant c. Case analyzing c:
  - c is an integer constant: Observing that $\|\mathsf{int}\| = \mathsf{int}$, and $ty(c) = pty(c)$, gives us proof.
  - c is Nil: Observing that $ty(\mathsf{Nil}) = \{\nu : \mathsf{intlist} \,|\, \phi_n\}$, and $\|ty(\mathsf{Nil})\| = \mathsf{intlist} = pty(\mathsf{Nil})$, gives us proof
  - c is Cons: Proof similar to the Nil case.
- Case T-APP: $e = e_1\ v$, where:

$$\begin{array}{cl} \Gamma \vdash e_1 : (x : \tau_1) \to \tau_2 & (H0) \\ \Gamma \vdash v_1 : \tau_1 & (H1) \end{array}$$

  Inductive hypotheses (after simplifying $\|(x : \tau_1) \to \tau_2\|$):

$$\begin{array}{cl} \|\Gamma\| \Vdash e_1 : \|\tau_1\|) \to \|\tau_2\| & (IH0) \\ \|\Gamma\| \Vdash v_1 : \|\tau_1\| & (IH1) \end{array}$$

  Applying ST-APP on $IH0-1$ gives proof.

- Case T-SUB: Hypotheses:

$$\begin{array}{cc} \Gamma \vdash e : \tau_1 & (H0) \\ \tau_1 <: \tau & (H1) \end{array}$$

Inductive hypothesis:

$$\|\Gamma\| \Vdash e : \|\tau_1\| \quad (IH0)$$

By inversion on $H1$, it is easy to derive that $\|\tau_1\| = \|\tau\|$. Using this to rewrite $IH0$ produces proof.

- Case T-ABS: $e = \lambda(x : \tau_1).\, e_1$, and $e = \lambda(x : \|\tau_1\|).\, e_1$. Hypotheses:

$$\begin{array}{cc} \tau = (x : \tau_1) \to \tau_2 & (H4) \\ \Gamma, x : \tau_1 \vdash e_1 : \tau_2 & (H5) \\ \|\tau\| = \|\tau_1\| \to \|\tau_2\| & (H6) \end{array}$$

Inductive hypotheses (after unfolding $\|\Gamma, x : \tau\|$ to $\|\Gamma\|, x : \|\tau_1\|$):

$$\|\Gamma\|, x : \|\tau_1\| \Vdash e_1 : \tau_2 \quad (IH0)$$

Applying ST-ABS on $IH0$ gives proof.

- T-LET: Proof closely resembles that for T-ABS case.
- T-MATCH: $e = \texttt{match}\ v\ \texttt{with Cons x y} \Rightarrow e_1\ \texttt{else}\ e_2$, and $e = \texttt{match}\ v\ \texttt{with Cons x y} \Rightarrow e_1\ \texttt{else}\ e_2$. Hypotheses:

$$\begin{array}{cc} \Gamma \vdash v : \mathsf{intlist} & (H0) \\ \Gamma \vdash \mathsf{Nil} : \{\nu : \mathsf{intlist} \mid \phi_n\} & (H1) \\ \Gamma \vdash \mathsf{Cons} : x : \mathsf{int} \to y : \mathsf{intlist} \to \{\nu : \mathsf{intlist} \mid \phi_c\} & (H2) \\ \Gamma_c = x : \mathsf{int},\, y : \mathsf{intlist},\, \phi & (H4) \\ \Gamma_n = \phi_n & (H5) \\ \Gamma \vdash \tau & (H6) \\ \Gamma, \Gamma_c \vdash e_1 : \tau & (H7) \\ \Gamma, \Gamma_n \vdash e_2 : \tau & (H8) \end{array}$$

Inductive hypotheses:

$$\begin{array}{cc} \|\Gamma\| \vdash v : \mathsf{intlist} & (IH0) \\ \|\Gamma\|, \|\Gamma_c\| \vdash e_1 : \|\tau\| & (IH1) \\ \|\Gamma\|, \|\Gamma_n\| \vdash e_2 : \|\tau\| & (IH2) \end{array}$$

Where,

$$\begin{array}{c} \|\Gamma_c\| = x : \mathsf{int},\, y : \mathsf{intlist} \\ \|\Gamma_n\| = \cdot \end{array}$$

Applying ST-MATCH on $IH0 - 2$ produces proof.

∎

We assert the substitution lemma for simply typed λ_R, but elide its proof as it closely follows the proof of substitution lemma for simply typed lambda calculus. The lemma is stated thus:

LEMMA 1.5. (**Substitution Preserves Simple Typing**) *forall* Γ, x, e, T_1, *and* T_2, *if* $\cdot, x : T_1, \Gamma \vdash e : T_2$ *and* $\cdot \vdash v : T_1$, *then* $\Gamma \vdash [v/x]\, e : T_2$.

LEMMA 1.6. (**Substitution Preserves Well-formedness of Relational Expressions**) *forall* Γ, *if* $x : \tau_1, \Gamma \vdash r :: \{\theta\}$ *and* $\cdot \vdash v : \tau_1$, *then* $[v/x]\, \Gamma \vdash [v/x]\, r :: \{\theta\}$.

Proof by induction on the derivation of $x : \tau_1, \Gamma \vdash r :: \{\theta\}$. Cases:

- Case S-APP: $r = R(v_1)$, and $[v/x]\, r = R([v/x]\, v_1)$. After expanding $\|x : \tau_1, \Gamma\|$ to $x : \|\tau_1\|, \|\Gamma\|$, hypotheses are:

$$x : \|\tau_1\|, \|\Gamma\| \Vdash v_1 : T \quad (H0)$$

Applying Lemma 1.4 on $\cdot \vdash v : \tau_1$ gives:

$$\cdot \Vdash v : \|\tau_1\| \quad (H1)$$

Applying the substitution lemma for simple type judgment of λ_R (Lemma 1.5) on $H0$ and $H1$, we derive:

$$\|\Gamma\| \Vdash [v/x]\, v_1 : T \quad (H2)$$

From the definition of substitution and erasure operations on type environments, we have that $\|[v/x]\,\Gamma\| = \|\Gamma\|$. Using this to rewrite $H2$:

$$\|[v/x]\,\Gamma\| \Vdash [v/x]\, v_1 : T \quad (H3)$$

Applying S-APP to $H3$ produces proof.

- Cases S-UNION, and S-CROSS: Proof follows trivially from inductive hypotheses.

∎

LEMMA 1.7. (**Substitution Preserves Well-formedness**) *forall* Γ, *if* $x : \tau_1, \Gamma \vdash \tau$ *and* $\cdot \vdash v : \tau_1$, *then* $[v/x]\,\Gamma \vdash [v/x]\,\tau$.

Proof Well-formedness judgment of λ_R types directly follows that of type refinements, which is in-turn dependent on well-formedness of relational predicates in type refinements, and ultimately on well-sortedness of relational expressions that constitute such predicates. Therefore, it suffices to show that substitution preserves the sort of relational expressions, which follows from Lemma 1.6 ∎

LEMMA 1.8. (**Abstract Type of a λ_R Value**) *forall* v, *if* $\cdot \vdash v : \{\nu : T \,|\, \phi\}$, *then* $\models_L v : [\![T]\!]_L$ *and* $[\![\Gamma_R]\!]_L \models_L [v/\nu]\,[\![\phi]\!]_L$.

Proof By case analysis on v. Cases:

- Case v is a variable x. Inversion of $\cdot \vdash x : \{\nu : T \,|\, \phi\}$ leads to absurdity. *ex falso quodlibet*.
- Case v is an abstraction $\lambda x : \tau.\, e$. Again, inversion leads to absurdity, as an abstraction cannot have a dependent base type.
- Case v is an integer c. Induction on $\cdot \vdash c : \{\nu : T \,|\, \phi\}$ leads to two relevant cases:
  - SubCase T-CONST: $T = \mathsf{int}$ and $\phi = true$. We know that $\models_L true$ is trivially valid. From the Definition 1.1, we also have that $\models_L c : [\![\mathsf{int}]\!]_L$.
  - SubCase T-SUB : Hypotheses:

$$\begin{array}{rl} \cdot \vdash c : \tau_1 & (H0) \\ \cdot \vdash \tau_1 <: \{\nu : T \,|\, \phi\} & (H1) \end{array}$$

By inversion on $H1$, we have:

$$\begin{array}{rl} \tau_1 = \{\nu : T \,|\, \phi_1\} & (H2) \\ \cdot \vdash c : \{\nu : T \,|\, \phi_1\} & (H3) \\ \cdot \vdash \{\nu : T \,|\, \phi_1\} & (H4) \\ \cdot \vdash \{\nu : T \,|\, \phi\} & (H4) \\ [\![\Gamma_R]\!]_L \models_L \nu : [\![T]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi]\!]_L & (H6) \end{array}$$

Inductive hypotheses:

$$\begin{array}{rl} \models_L c : [\![T]\!]_L & (IH0) \\ [\![\Gamma_R]\!]_L \models_L [c/\nu]\,[\![\phi_1]\!]_L & (IH1) \end{array}$$

Since ν occurs free in $H6$, applying the universal quantification introduction rule ($\forall I$):

$$[\![\Gamma_R]\!]_L \models_L \forall \nu.\, (\nu : [\![T]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi]\!]_L)$$

Now, eliminating the quantifier (rule $\forall E$) by instantiating the bound variable with c:

$$[\![\Gamma_R]\!]_L \models_L c : [\![T]\!]_L \Rightarrow [c/\nu]\,[\![\phi_1]\!]_L \Rightarrow [c/\nu]\,[\![\phi]\!]_L \quad (H8)$$

Using weakened form of $IH0$ (with $[\![\Gamma_R]\!]_L$ introduced in its context using weakening theorem of first order logic), $IH1$, and $H8$ we have:

$$\models_L [c/\nu]\,[\![\phi]\!]_L \quad (H9)$$

Proof follows from $IH0$ and $H9$.

- Case v is Nil: From Definition 1.1, we have that:

$$\begin{array}{rl} T = \mathsf{intlist} & (H0) \\ \models_L \ \mathsf{Nil} : [\![\mathsf{intlist}]\!]_\mathsf{L} & (H1) \\ [\![\Gamma_R]\!]_L \ \models_L \ [\mathsf{Nil}/\nu]\,[\![\phi]\!]_L & (H2) \end{array}$$

Proof follows straightforwardly from hypotheses.
- Case v is $\mathsf{Cons}\ v_1\ v_2$, such that

$$\cdot \vdash \mathsf{Cons}\ v_1\ v_2 \ : \ \{\nu : T \,|\, \phi\} \quad (H0)$$

We first show that $T = \mathsf{intlist}$. Applying Lemma 1.4 on $H0$, we have:

$$\cdot \Vdash \mathsf{Cons}\ v_1\ v_2 \ : \ T \quad (H1)$$

By inversion on the simple type derivation (Figure 1) in $H1$, we show that $T = \mathsf{intlist}$. Using this to rewrite $H0$:

$$\cdot \vdash \mathsf{Cons}\ v_1\ v_2 \ : \ \{\nu : \mathsf{intlist} \,|\, \phi\} \quad (H2)$$

From $H1$, and Definition 1.1, we prove that

$$\models_L \ \mathsf{Cons}\ v_1\ v_2 : [\![\mathsf{intlist}]\!]_L$$

- Cases when v is Cons, or v is $\mathsf{Cons}\ v_1$, for some value v_1, lead to contradiction as v cannot have a base dependent type in these cases.

■

LEMMA 1.9. (**Substitution Preserves Subtyping**) *forall* Γ*, if* $., x : \tau, \Gamma \vdash \tau_1 <: \tau_2$ *and* $\cdot \vdash v : \tau$*, then* $[v/x]\,\Gamma \vdash [v/x]\,\tau_1 <: [v/x]\,\tau_2$.

Proof By induction on the subtype judgment. Cases:

- Case SUBT-ARROW: Proof is a straightforward application of SUBT-ARROW on inductive hypotheses.
- Case SUBT-BASE: τ_1 is of form $\{\nu : T \,|\, \phi_1\}$, and τ_2 is of form $\{\nu : T \,|\, \phi_2\}$. By destructing τ, we have two cases:
  - SubCase $\tau = \{\nu : T_x \,|\, \phi_x\}$ for some T_x and ϕ_x: Hypotheses:

$$\begin{array}{rl} .\,x : \tau,\, \Gamma \vdash \tau_1 <: \tau_2 & (H0) \\ \cdot \vdash v \ : \ \{\nu : T_x \,|\, \phi_x\} & (H1) \\ .\,x : \tau,\, \Gamma \vdash \{\nu : T \,|\, \phi_1\} & (H2) \\ .\,x : \tau,\, \Gamma \vdash \{\nu : T \,|\, \phi_2\} & (H3) \\ [\![\Gamma_R]\!]_L \ \models_L \ [\![x : \{\nu : T_x \,|\, \phi_x\},\, \Gamma, \nu : T]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi_2]\!]_L & (H4) \end{array}$$

  Expanding $H4$:

$$[\![\Gamma_R]\!]_L \ \models_L \ x : [\![T_x]\!]_L \Rightarrow [\![\phi_x]\!]_L \Rightarrow [\![\Gamma,\, \nu : T]\!]_L \Rightarrow [\![\phi_1]\!]_L \ \Rightarrow \ [\![\phi_2]\!]_L \quad (H5)$$

  Since x occurs free in the above formula, using the universal quantification introduction rule $(\forall I)$, we have:

$$[\![\Gamma_R]\!]_L \ \models_L \ \forall x.\, x : [\![T_x]\!]_L \Rightarrow [\![\phi_x]\!]_L \Rightarrow [\![\Gamma,\, \nu : T]\!]_L \Rightarrow [\![\phi_1]\!]_L \ \Rightarrow \ [\![\phi_2]\!]_L$$

  Now, using the elimination rule of universal quantification $(\forall E)$ to instantiate the bound x with v:

$$[\![\Gamma_R]\!]_L \ \models_L \ [v/x]\,(\, x : [\![T_x]\!]_L \Rightarrow [\![\phi_x]\!]_L \Rightarrow [\![\Gamma,\, \nu : T]\!]_L \Rightarrow [\![\phi_1]\!]_L \ \Rightarrow \ [\![\phi_2]\!]_L) \quad (H6)$$

  Distributing the substitution operation:

$$[\![\Gamma_R]\!]_L \ \models_L \ (v : [\![T_x]\!]_L \Rightarrow [\![[v/x]\,\phi_x]\!]_L \Rightarrow [\![[v/x]\,\Gamma,\, \nu : T]\!]_L \Rightarrow [\![[v/x]\,\phi_1]\!]_L \ \Rightarrow \ [\![[v/x]\,\phi_2]\!]_L) \quad (H7)$$

  Now, from Lemma 1.8, using the hypothesis $H2$, we derive:

$$\begin{array}{rl} \models_L \ v : [\![T_x]\!]_L & (H9) \\ [\![\Gamma_R]\!]_L \ \models_L \ [\![[v/x]\,\phi_x]\!]_L & (H10) \end{array}$$

  Using $H8$, weakened $H9$, where $[\![\Gamma_R]\!]_L$ is introduced in its context, and $H10$, we derive:

$$[\![\Gamma_R]\!]_L \ \models_L \ [\![[v/x]\,\Gamma,\, \nu : T]\!]_L \Rightarrow [\![[v/x]\,\phi_1]\!]_L \ \Rightarrow \ [\![[v/x]\,\phi_2]\!]_L) \quad (H11)$$

  Applying Lemma 1.7 over $H2$ and $H3$ yields:

$$\begin{array}{rl} [v/x]\,\Gamma \vdash \{\nu : T \,|\, [v/x]\,\phi_1\} & (H12) \\ [v/x]\,\Gamma \vdash \{\nu : T \,|\, [v/x]\,\phi_2\} & (H13) \end{array}$$

  Applying SUBT-BASE on $H11 - 13$ proves the theorem.

- SubCase $\tau = \tau_{x_1} \to \tau_{x_2}$, for some τ_{x_1} and τ_{x_2}. First, we observe that all free variables in well-formed type refinements (i.e., arguments to relations. Please refer to the syntactic class τ_R in Fig. 1 of the paper.) have type int or intlist, therefore $x : \tau_{x_1} \to \tau_{x_2}$ cannot occur free in ϕ_1, ϕ_2, and type-refinements in Γ. Consequently:

$$[v/x]\,\Gamma = \Gamma \quad (H0)$$
$$[v/x]\,\phi_1 = \phi_1 \quad (H1)$$
$$[v/x]\,\phi_2 = \phi_2 \quad (H2)$$

Rewriting inductive hypotheses (not shown here) using $H0 - 3$ and applying SUBT-BASE results in proof.

∎

LEMMA 1.10. (**Weakening**) *if* $\Gamma \vdash e : \tau$ *then forall* Γ', $\quad \Gamma', \Gamma \vdash e : \tau$.

Proof by induction on $\Gamma \vdash e : \tau$ derivation. In most cases, proof follows directly from inductive hypotheses. The only interesting case is T-SUB:

- Case T-SUB: Hypothesis:

$$\Gamma \vdash e : \tau \quad (H0)$$
$$\tau_1 <: \tau \quad (H1)$$

By inductive hypotheses, we have:

$$\Gamma', \Gamma \vdash e : \tau_1 \quad (IH0)$$

To apply SUBT-BASE in order to prove the lemma, it suffices to prove that $\Gamma', \Gamma \vdash \tau_1 <: \tau$:, which we prove by induction on $H1$. Cases:

  - SubCase SUBT-BASE: Hypotheses:

$$\tau_1 = \{\nu : T \,|\, \phi_1\} \quad (H2)$$
$$\tau = \{\nu : T \,|\, \phi\} \quad (H3)$$
$$[\![\Gamma_R]\!]_L \models_L [\![\Gamma, \nu : T]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi]\!]_L \quad (H4)$$

  To prove that $\Gamma', \Gamma \vdash \tau_1 <: \tau$:, it suffices to prove that

$$[\![\Gamma']\!]_L,\, [\![\Gamma_R]\!]_L \models_L [\![\Gamma, \nu : T]\!]_L \Rightarrow [\![\phi_1]\!]_L \Rightarrow [\![\phi]\!]_L$$

  Which follows from $H4$ by monotonicity of entailment (or thinning) in first-order logic.
  - SubCase SUBT-ARROW: Proof follows trivially from inductive hypotheses by applying SUBT-ARROW.

∎

LEMMA 1.11. (**Well-Typedness Implies Well-Formedness**) *forall* v, *and* τ, *if* $\cdot \vdash v : \tau$ *then* $\cdot \vdash \tau$.

Proof is by case analysis on the structure of v, followed by induction on the typing derivation $\Gamma \vdash v : \tau$, for each case of v. After trivially discharging the cases that result in contradiction, we are left with following cases:

- Case T-SUB: Proof follows from the premises of SUBT-BASE rule, which explicitly assert well-formedness of types involved in subtype judgment.
- Case T-CONST: Type refinement $true$ for integer constants is well-formed under any context. Well-formedness of Nil and Cons type refinements are explicitly asserted in Definition 1.1.
- Case T-ABS: Proof by applying WF-FUN on inductive hypotheses.

∎

Since expressions of λ_R are in A-Normal form by construction, we only need substitution lemma for value substitutions.

LEMMA 1.12. (**Substitution Preserves Typing**) *forall* Γ, x, e, τ_1, *and* τ_2, *if* $\cdot, x : \tau_1, \Gamma \vdash e : \tau_2$ *and* $\cdot \vdash v : \tau_1$, *then* $[v/x]\,\Gamma \vdash [v/x]\,e : [v/x]\,\tau_2$.

Proof . Hypotheses (after generalizing dependent Γ and τ_2):

$$\forall \Gamma, \forall \tau_2, \cdot, x : \tau_1, \Gamma \vdash e : \tau_2 \quad (H0)$$
$$\cdot \vdash v : \tau_1 \quad (H1)$$

First, using $H1$ and Lemma 1.11, we derive the following:

$$\cdot \vdash \tau_1 \quad (H2)$$

Now, we proceed by induction on $H0$. Cases:

- Case T-VAR: e is a variable y such that:

$$\cdot,\, x : \tau_1,\, \Gamma \vdash y \,:\, \tau_2 \quad (H3)$$

We have two subcases:

  - SubCase $y = x$: Since a variable is never bound twice in the environment, by inversion on $\cdot,\, x : \tau_1,\, \Gamma \vdash x \,:\, \tau_2$, we know that:

$$\tau_1 = \tau_2 \quad (H4)$$

Since $[v/x]\, x = v$ it remains to prove that $[v/x]\, \Gamma \vdash v \,:\, [v/x]\, \tau_2$. Rewriting using $H4$, the goal is:

$$[v/x]\, \Gamma \vdash v \,:\, [v/x]\, \tau_1$$

From $H2$, we know that τ_1 is well-formed under empty context; so, its type-refinement is closed. Hence, $[v/x]\, \tau_1 = \tau_1$. Using this to rewrite the goal:

$$[v/x]\, \Gamma \vdash v \,:\, \tau_1$$

From $H1$, we know that $\cdot \vdash v \,:\, \tau_1$. Applying the weakening lemma (Lemma 1.10), with bound Γ' instantiated to $[v/x]\, \Gamma$ gives us the required proof.

  - SubCase $y \neq x$: From $H3$, since $y \neq x$, we have:

$$(y : \tau_2) \in \Gamma \quad (H5)$$

Applying definition of substitution lifted to type environments yields the following:

$$(y : [v/x]\, \tau_2) \in [v/x]\, \Gamma \quad (H6)$$

Since $[v/x]\, y = y$, applying T-VAR rule using $H6$ lets us conclude that

$$[v/x]\, \Gamma \vdash [v/x]\, y \,:\, [v/x]\, \tau_2$$

which is the required goal.

- Case T-CONST: SubCases:
  - e is an integer constant c: Proof trivial as $[v/x]\, c = c$ and c has type int under any context (Definition 1.1 and T-CONST).
  - e is Nil, or e is Cons: Hypotheses:

$$\begin{array}{ll} \Gamma \vdash \mathsf{Nil} \,:\, ty(\mathsf{Nil}) & (H1) \\ \Gamma \vdash \mathsf{Cons} \,:\, ty(\mathsf{Cons}) & (H2) \end{array}$$

From the definition of ty (Definition 1.1), we know that type refinements of Nil and Cons are well-formed under empty context; so, they are closed. Consequently $[v/x]\, ty(\mathsf{Nil}) = ty(\mathsf{Nil})$, and $[v/x]\, ty(\mathsf{Cons}) = ty(\mathsf{Cons})$. Also, $[v/x]\, \mathsf{Nil} = \mathsf{Nil}$ and $[v/x]\, \mathsf{Cons} = \mathsf{Cons}$. Therefore, proof follows from $H1$ and $H2$.

- Case T-APP: e is a function application of form $e_1\, v_1$, where:

$$\begin{array}{ll} \cdot,\, x : \tau_1, \Gamma \vdash e_1 \,:\, (y : \tau_3) \rightarrow \tau_2 & (H4) \\ \cdot,\, x : \tau_1, \Gamma \vdash v_1 \,:\, \tau_3 & (H5) \end{array}$$

Inductive hypotheses, after trivially instantiating bound Γ and τ_2:

$$\begin{array}{ll} [v/x]\, \Gamma \vdash [v/x]\, e_1 \,:\, [v/x]\, ((y : \tau_3) \rightarrow \tau_4) & (IH0) \\ [v/x]\, \Gamma \vdash [v/x]\, v_1 \,:\, [v/x]\, \tau_3 & (IH1) \end{array}$$

Pushing the substitution to the level of base types in the arrow type:

$$[v/x]\, \Gamma \vdash [v/x]\, e_1 \,:\, (y : [v/x]\, \tau_3) \rightarrow [v/x]\, \tau_4 \quad (H6)$$

Since $[v/x]\, (e_1\, v_1) = [v/x]\, e_1\ [v/x]\, v_1$, the goal is to prove:

$$[v/x]\, \Gamma \vdash [v/x]\, e_1\ [v/x]\, v_1 \,:\, [v/x]\, \tau_2$$

Applying T-APP using $H6$ and $IH1$ proves the goal.

- Case T-SUB: Hypotheses:

$$\begin{array}{ll} x : \tau_1,\, \Gamma \vdash e \,:\, \tau_3 & (H4) \\ x : \tau_1,\, \Gamma \vdash \tau_3 <: \tau_2 & (H5) \\ [v/x]\,\Gamma \vdash [v/x]\,e \,:\, [v/x]\,\tau_3 & (IH0) \end{array}$$

From Lemma 1.9, which states that substitution preserves subtyping, we know that:

$$[v/x]\,\Gamma \vdash [v/x]\,\tau_3 <: [v/x]\,\tau_2 \quad (H6)$$

Applying T-SUB rule on $IH0$ and $H6$ completes the proof.

- Case T-ABS: e is of form $\lambda y : \tau.\, e_1$. Hypotheses:

$$\begin{array}{ll} \tau_2 = (y : \tau_3) \to \tau_4 & (H4) \\ x : \tau_1,\, \Gamma \vdash \tau_3 & (H5) \\ x : \tau_1,\, \Gamma,\, y : \tau_3 \vdash e_1 \,:\, \tau_4 & (H6) \end{array}$$

We note that $y \notin freevars(v)$ as $\cdot \vdash v \,:\, \tau$ (from $H1$). Therefore, substitution $[v/x]\,e$ is always capture avoiding. Further, we assume that a variable cannot be bound twice in the environment (Γ). This eliminates the case of x and y being equal, leaving us with only case where $x \neq y$:

  - SubCase (lambda bound y not same as x): Using Lemma 1.7, which asserts that substitution preserves well-formedness of types, we derive the following from $H5$:

$$[v/x]\,\Gamma \vdash [v/x]\,\tau_3 \quad (H7)$$

  By instantiating the bound Γ and τ_2 in IH with $(\Gamma,\, y \,:\, \tau_3)$ and τ_4, respectively, and using $H6$, we derive the following:

$$[v/x]\,(\Gamma,\ y : \tau_3) \vdash [v/x]\,e_1 \,:\, [v/x]\,\tau_4$$

  which, as $y \neq x$, expands to the following:

$$[v/x]\,\Gamma,\ y : [v/x]\,\tau_3 \vdash [v/x]\,e_1 \,:\, [v/x]\,\tau_4 \quad (H8)$$

  By the definition of substitution, since $y \neq x$, we have the following:

$$[v/x]\,(\lambda y : \tau.\, e_1) = \lambda(y : [v/x]\,\tau_3.\, [v/x]\,e_1)$$

$$[v/x]\,((y : \tau_3) \to \tau_4) = (y : [v/x]\,\tau_3) \to [v/x]\,\tau_4$$

  It remains to show that

$$\Gamma \vdash \lambda(y : [v/z]\,\tau.\, [v/x]\,e_1) \,:\, (y : [v/x]\,\tau_3) \to [v/x]\,\tau_4$$

  which follows by applying the rule T-ABS with $H7$ and $H8$.

- Case T-LET: e is of form `let` y `=` e_1 `in` e_2. Hypotheses:

$$\begin{array}{ll} x : \tau_1,\, \Gamma \vdash e_1 \,:\, \tau_3 & (H4) \\ x : \tau_1,\, \Gamma \vdash \tau_2 & (H5) \\ x : \tau_1,\, \Gamma,\, y : \tau_3 \vdash e_2 \,:\, \tau_2 & (H6) \end{array}$$

Since $y \notin freevars(v)$, $[v/x]\,e_2$ avoids variable capture. Since a variable cannot be bound twice in Γ, x and y cannot be equal. This leaves us with one case for $[v/x]\,e$:

  - SubCase $y \neq x$: Using Lemma 1.7, which asserts well-formedness preservation under substitution, we get the following from $H5$:

$$[v/x]\,\Gamma \vdash [v/x]\,\tau_2 \quad (H7)$$

  Instantiating bound Γ and τ_2 in IH appropriately gives us following hypotheses:

$$\begin{array}{ll} [v/x]\,\Gamma \vdash [v/x]\,e_1 \,:\, [v/x]\,\tau_3 & (H8) \\ [v/x]\,(\Gamma,\, y : \tau_3) \vdash [v/x]\,e_2 \,:\, [v/x]\,\tau_2 & (H9) \end{array}$$

  Since $x \neq y$, $H9$ is equivalent to:

$$[v/x]\,\Gamma,\, y : [v/x]\,\tau_3 \vdash [v/x]\,e_2 \,:\, [v/x]\,\tau_2 \quad (H10)$$

  Applying T-LET rule on $H8$ and $H10$ lets us conclude:

$$[v/x]\,\Gamma \vdash \texttt{let}\ y = [v/x]\,e_1\ \texttt{in}\ [v/x]\,e_2 \,:\, [v/x]\,\tau_2$$

  which is what needs to be proven.

- Case T-MATCH: e is of form $\texttt{match}\ v'\ \texttt{with}\ \mathsf{Cons}\ x'\ y'\ \Rightarrow\ e_1\ \texttt{else}\ e_2$, where

$$\begin{array}{cr} x:\tau_1,\,\Gamma\,\vdash\,v'\,:\,\mathsf{intlist} & (H4)\\ x:\tau_1,\,\Gamma\,\vdash\,\mathsf{Nil}\,:\,\{\nu:\mathsf{intlist}\,|\,\phi_n\} & (H5)\\ x:\tau_1,\,\Gamma\,\vdash\,\mathsf{Cons}\,:\,x':\mathsf{int}\rightarrow y':\mathsf{intlist}\rightarrow\{\nu:\mathsf{intlist}\,|\,\phi_c\} & (H6)\\ \Gamma_c = x':\mathsf{int}, y':\mathsf{intlist},\,[v'/\nu]\,\phi_c & (H7)\\ \Gamma_n = [v'/\nu]\,\phi_n & (H8)\\ x:\tau_1,\,\Gamma\,\vdash\,\tau & (H9)\\ x:\tau_1,\,\Gamma,\,\Gamma_c\,\vdash\,e_1\,:\,\tau & (H10)\\ x:\tau_1,\,\Gamma,\,\Gamma_n\,\vdash\,e_2\,:\,\tau & (H11)\end{array}$$

From $H9$, after applying Lemma 1.7, we get:

$$[v/x]\,\Gamma\,\vdash\,[v/x]\,\tau \quad (H12)$$

We assert that x cannot be equal to x', or y', as it leads to x being bound twice in Γ. Therefore, we are left with one case:

  - SubCase $x \neq x'$, and $x \neq y'$: Substitution $[v/x]\,e$ can be expanded to

$$\texttt{match}\ [v/x]\,v'\ \texttt{with}\ \mathsf{Cons}\,x'\ y'\ \Rightarrow\ [v/x]\,e_1\ \texttt{else}\ [v/x]\,e_2$$

Inductive hypotheses (after pushing substitutions into type refinements):

$$\begin{array}{cr} [v/x]\,\Gamma\,\vdash\,[v/x]\,v'\,:\,\mathsf{intlist} & (IH1)\\ [v/x]\,\Gamma\,\vdash\,\mathsf{Nil}\,:\,\{\nu:\mathsf{intlist}\,|\,[v/x]\,\phi_n\} & (IH2)\\ [v/x]\,\Gamma\,\vdash\,\mathsf{Cons}\,:\,x':\mathsf{int}\rightarrow y':\mathsf{intlist}\rightarrow\{\nu:\mathsf{intlist}\,|\,[v/x]\,\phi_c\} & (IH3)\\ [v/x]\,\Gamma,\,[v/x]\,\Gamma_c\,\vdash\,[v/x]\,e_1\,:\,[v/x]\,\tau & (IH4)\\ [v/x]\,\Gamma,\,[v/x]\,\Gamma_n\,\vdash\,[v/x]\,e_2\,:\,[v/x]\,\tau & (IH5)\end{array}$$

Expansions of $[v/x]\,\Gamma_c$ and $[v/x]\,\Gamma_n$ are given below:

$$\begin{array}{cr} [v/x]\,\Gamma_c = x':\mathsf{int}, y':\mathsf{intlist},\,[v/x]\,[v'/\nu]\,\phi_c & (H13)\\ [v/x]\,\Gamma_n = [v/x]\,[v'/\nu]\,\phi_n & (H14)\end{array}$$

Since Nil and Cons are constants, from the definition of ty (Definition 1.1) and T-CONST:

$$\begin{array}{c} \cdot\,\vdash\,\mathsf{Nil}\,:\,\{\nu:\mathsf{intlist}\,|\,\phi_n\}\\ \cdot\,\vdash\,\mathsf{Cons}\,:\,x':\mathsf{int}\rightarrow y':\mathsf{intlist}\rightarrow\{\nu:\mathsf{intlist}\,|\,\phi_c\}\end{array}$$

From Lemma 1.11, we know that types of Nil and Cons are well-formed under empty context. By inverting well-formedness derivation of Nil type (via WF-BASE rule in Fig. 3 of the paper), and well-formedness derivation of Cons type (twice through WF-FUN, and once through WF-BASE), we derive:

$$\begin{array}{cr} \cdot\,\vdash\,\phi_n & (H15)\\ x':\mathsf{int},\,y':\mathsf{intlist}\,\vdash\,\phi_c & (H16)\end{array}$$

From $H15$, we conclude that:

$$x \notin freevars(\phi_n) \quad (H17)$$

Similarly, from $H17$, since $x \neq x'$ and $x \neq y'$, we conclude:

$$x \notin freevars(\phi_c) \quad (H18)$$

Using $H17-18$, and the definition of substitution operation, we rewrite $H13-14$ as:

$$\begin{array}{cr} [v/x]\,\Gamma_c = x':\mathsf{int}, y':\mathsf{intlist},\,[[v/x]\,v'/\nu]\,\phi_c & (H19)\\ [v/x]\,\Gamma_n = [[v/x]\,v'/\nu]\,\phi_n & (H20)\end{array}$$

Finally, by applying T-MATCH on $IH1-5$ and $H19-20$ leads us to conclude that:

$$[v/x]\,\Gamma\,\vdash\,\texttt{match}\ [v/x]\,v'\ \texttt{with}\ \mathsf{Cons}\,x'\ y'\ \Rightarrow\ [v/x]\,e_1\ \texttt{else}\ [v/x]\,e_2\,:\,[v/x]\,\tau$$

Which is what needs to be proven.

■

THEOREM 1.13. (**Preservation**) *if* $\cdot \vdash e : \tau$, *and* $e \longrightarrow e'$, *then* $\cdot \vdash e' : \tau$.

Proof by induction on type derivation $\cdot \vdash e : \tau$. Cases:

- Case T-VAR: e is a variable x. By inversion on $\cdot \vdash x : \tau$ we get $x : \tau \in \cdot$, which is absurd. Proof follows from *ex falso quodlibet*.
- Cases T-CONST and T-ABS: Constants and abstractions are values, therefore cannot take a step.
- Case T-APP: e is of form $e_1\ v$, where $\cdot \vdash e_1 : (x : \tau_1) \to \tau_2$, and $\cdot \vdash v : \tau_1$. Therefore, $\cdot \vdash e : [v/x]\,\tau_2$. Inversion on $e_1\ v \longrightarrow e'$. Cases:
  - SubCase E-CO: $e_1 \longrightarrow e_1'$. Therefore, $e_1\ v \longrightarrow e_1'\ v$. By IH, $\cdot \vdash e_1' : (x : \tau_1) \to \tau_2$. Hence, $\cdot \vdash e_1'\ v : [v/x]\,\tau_2$.
  - SubCase E-APP: Inversion on $\cdot \vdash e_1 : (x : \tau_1) \to \tau_2$. Cases:
    - SubSubCase $e_1 = \lambda x : \tau_3.\, e_2$. Therefore, $e_1\ v \longrightarrow [v/x]\,e_2$. By inversion on $\cdot \vdash \lambda x : \tau_3.\, e_2 : \tau_1 \to \tau_2$, we know that $\tau_3 = \tau_1$, and

      $$\cdot, x : \tau_1 \vdash e_2 : \tau_2 \quad (H0)$$

      Now, using $H0$ and the substitution lemma (Lemma 1.12), with bound Γ instantiated with $\cdot$, we get $\cdot \vdash [v/x]\,e_2 : [v/x]\,\tau_2$. Hence, $e' = [v/x]\,e_2$ has same type as e.
    - SubSubCase $e_1 = \mathsf{Cons}$, or $e_1 = \mathsf{Cons}\ v_1$, for some v_1: Both $\mathsf{Cons}\ v$, and $\mathsf{Cons}\ v_1\ v$ are values, therefore cannot take a step.
- Case T-SUB: Hypotheses:

  $$\begin{array}{ll} \cdot \vdash e : \tau_1 & (H0) \\ \tau_1 <: \tau & (H1) \end{array}$$

  Inductive hypothesis:

  $$If \quad e \longrightarrow e', \quad then \quad \cdot \vdash e' : \tau_1 \quad (IH)$$

  Proof follows from IH and $H1$.
- Case T-LET: e is of form `let` x `=` e_1 `in` e_2, where:

  $$\begin{array}{rl} \cdot \vdash e_1 : \tau_1 & (H0) \\ \cdot, x : \tau_1 \vdash e_2 : \tau & (H1) \\ \cdot \vdash \tau & (H2) \end{array}$$

  Inductive Hypothesis:

  $$If \quad e_1 \longrightarrow e_1', \quad then \quad \cdot \vdash e_1' : \tau_1 \quad (IH)$$

  By inversion on $e \longrightarrow e'$, we have two cases:
  - SubCase E-CO: $e_1 \longrightarrow e_1'$ and $e \longrightarrow$ `let` x `=` e_1' `in` e_2. Applying IH, we know that:

    $$\cdot \vdash e_1' : \tau_1 \quad (H3)$$

    Applying T-LET using hypotheses $H3$, $H1$ and $H2$, we conclude that $\cdot \vdash \texttt{let}\ x = e_1'\ \texttt{in}\ e_2 : \tau$.
  - SubCase E-LET: $e_1 \longrightarrow v$ and $e \longrightarrow [v/x]\,e_2$. Using $H0$ and $H1$, and applying the substitution lemma (Lemma 1.12), we derive the following:

    $$\cdot \vdash [v/x]\,e_2 : [v/x]\,\tau \quad (H4)$$

    From hypothesis $H2$, we know that τ is well-formed under empty context. Rewriting $H4$ with $[v/x]\,\tau = \tau$ lets us conclude:

    $$\cdot \vdash [v/x]\,e_2 : \tau \quad (H4)$$
- Case T-MATCH: e is of form `match` v `with` $\mathsf{Cons}\ x\ y \Rightarrow e_1$ `else` e_2, where:

  $$\begin{array}{cl} \cdot \vdash v : \mathsf{intlist} & (H0) \\ \cdot \vdash \mathsf{Nil} : \{\nu : \mathsf{intlist} \mid \phi_n\} & (H1) \\ \cdot \vdash \mathsf{Cons} : x : \mathsf{int} \to y : \mathsf{intlist} \to \{\nu : \mathsf{intlist} \mid \phi_c\} & (H2) \\ \Gamma_c = x : \mathsf{int}, y : \mathsf{intlist},\ [v/\nu]\,\phi_c & (H4) \\ \Gamma_n = [v/\nu]\,\phi_n & (H5) \\ \cdot \vdash \tau & (H6) \\ \Gamma_c \vdash e_1 : \tau & (H7) \\ \Gamma_n \vdash e_2 : \tau & (H8) \end{array}$$

  By inversion on $e \longrightarrow e'$, we have two cases:

- SubCase E-MCONS: Hypotheses:

$$v = \mathsf{Cons}\ v_1\ v_2 \qquad (H9)$$
$$e' = [v_2/y]\,[v_1/x]\,e_1 \quad (H10)$$

By inversion on $H0$, we derive:

$$\cdot \vdash v_1 : \mathsf{int} \qquad (H11)$$
$$\cdot \vdash v_2 : \mathsf{intlist} \quad (H12)$$

Using $H4$ and $H7$, and twice applying the substitution lemma (Lemma 1.12), we derive the following:

$$[v_2/y]\,[v_1/x]\,[v/\nu]\,\phi_c \vdash [v_2/y]\,[v_1/x]\,e_1 : [v_2/y]\,[v_1/x]\,\tau \quad (H13)$$

From Definition 1.1, which asserts the validity of type refinements of Cons and Nil, we get:

$$[\![\Gamma_R]\!]_L,\ x : [\![\mathsf{int}]\!]_L,\ y : [\![\mathsf{intlist}]\!]_L \models_L [\![[\mathsf{Cons}\ x\ y/\nu]\,\phi_c]\!]_L \quad (H14)$$

Using $H11-12$, and instantiating x, and y with v_1 and v_2, respectively:

$$[\![\Gamma_R]\!]_L \models_L [\![[v_2/y]\,[v_1/x]\,[v/\nu]\,\phi_c]\!]_L \quad (H15)$$

Now, applying the cut elimination lemma (Lemma 1.3) on $H13$ and $H15$, we derive:

$$\cdot \vdash [v_2/y]\,[v_1/x]\,e_1 : [v_2/y]\,[v_1/x]\,\tau \quad (H16)$$

From $H6$, we know that type refinement of τ is a closed term; therefore, $[v_2/y]\,[v_1/x]\,\tau = \tau$. Using this fact to rewrite $H16$, we conclude that:

$$\cdot \vdash [v_2/y]\,[v_1/x]\,e_1 : \tau$$

- SubCase E-MNIL: $v = \mathsf{Nil}$. $e' = e_2$. From $H5$ and $H8$:

$$[v/\nu]\,\phi_n \vdash e_2 : \tau \quad (H9)$$

As in the case of E-MCONS, using Definition 1.1 and cut elimination lemma (Lemma 1.3), we conclude that:

$$\cdot \vdash e_2 : \tau$$

■

THEOREM 1.14. (**Type Safety**) *if* $\cdot \vdash e : \tau$*, then either* e *is a value, or* $e \longrightarrow e'$ *and* $\cdot \vdash e' : \tau$.

Proof follows directly from progress (Theorem 1.1), and preservation (Theorem 1.13) properties. ■

1.3 MSFOL Semantics of Type Refinements

We now prove that the exercise of ascribing MSFOL semantics to type refinements is complete. The metatheory relies on certain definitions given below:

Definition *Nominal Type System for MSFOL lanugage* We define a nominal type system that assigns MSFOL sorts (τ^F) to MSFOL propositions. The type system is nominal in the sense that the sorts assigned by the type system need not necessarily relate to the actual sort of a proposition under MSFOL. Sorts are always assigned under an empty sort environment. The type system is defined in Fig 3.

Definition *Join of Nominal Types* We define join operation on nominal types recursively as:

$$\begin{array}{lcl} \textit{bool} \bowtie \textit{bool} & = & \textit{bool} \\ \textit{bool} \bowtie T^F \to \tau^F & = & T^F \to \tau^F \\ T^F \to \tau^F \bowtie \textit{bool} & = & T^F \to \tau^F \\ T^F \to \tau_1^F \bowtie \tau_2^F & = & T^F \to (\tau_1^F \bowtie \tau_2^F) \end{array}$$

Definition *Substitution Operation on Propositions.* Capture avoiding substitution, where variable capture is with respect to the variable bound by quantifiers, is assumed on MSFOL propositions.

$$\boxed{\cdot \vdash \phi^F \,:\, \tau^F, \quad \cdot \vdash \phi^L \,:\, \tau^F}$$

QF-PROP-SORT

$$\frac{}{\cdot \vdash \phi^F \,:\, \mathit{bool}}$$

Q-PROP-SORT

$$\frac{\cdot \vdash \phi^L \,:\, \tau^F}{\cdot \vdash \forall(k : T^F).\, \phi^L \,:\, T^F \to \mathit{bool}}$$

Figure 3: Nominal Type System for MSFOL propositions

Definition In our meta-theory, we use $\odot$ to denote any boolean connective such that, for any two MSFOL formulas, ϕ_1^L and ϕ_2^L, $\phi_1^L \odot \phi_2^L$ is an MSFOL formula.

LEMMA 1.15. (η_{wrap} **is type safe**) *forall* ϕ^F*, there exists and MSFOL proposition* ϕ^L *such that* $\eta_{wrap}(\phi^F, \tau^F) = \phi^L$*, and* $\cdot \vdash \phi^L \,:\, \tau^F$

Proof By induction on τ^F. Cases:

- Case $\tau^F = \mathit{bool}$: $\eta_{wrap}(\phi^F, \mathit{bool}) = \phi^F$, which is an MSFOL proposition. From QF-PROP-SORT, we also know that $\cdot \vdash \phi^F \,:\, \mathit{bool}$.
- Case $\tau^F = T^F \to \tau_2^F$: Eta-wrap expansion is:$\eta_{wrap}(\phi^F, T^F \to \tau^F) = \forall(k : T^F).\, \eta_{wrap}(\phi^F\, k, \tau_2^F)$. Inductive hypothesis tells us that there exists an MSFOL proposition ϕ_k^L, such that:

$$\begin{array}{rl} \phi_k^L = \eta_{wrap}(\phi^F\, k, \tau_2^F) & (IH0) \\ \cdot \vdash \phi_k^L \,:\, \tau_2^F & (IH1) \end{array}$$

  Now, $\phi^L = \forall(k : T^F).\, \phi_k^L$ is an MSFOL proposition. Further, applying Q-PROP-SORT on $IH1$ lets us conclude:

$$\cdot \vdash \phi^L \,:\, T^F \to \tau_2^F$$

∎

LEMMA 1.16. (**MSFOL sort of a relation**) *forall R, if* $\cdot \vdash R :: \tau_R$*, then there exists an MSFOL proposition* ϕ_R^L *such that* $[\![R]\!]_L = \phi_R^L$*, and* $\cdot \vdash \phi_R^L \,:\, [\![\tau_R]\!]_L$.

Proof We proceed by case analysis on R.

- Case $R = R_{id}$: From S-REL-ID, we know that:

$$\cdot \vdash R_{id} :: \mathsf{int} \to \{\mathsf{int}\} \quad (H0)$$

  Also, from Fig. 4:

$$\begin{array}{rl} [\![\mathsf{int} \to \{\mathsf{int}\}]\!]_L = [\![\mathsf{int}]\!]_L \to [\![\mathsf{int}]\!]_L \to \mathit{bool} & (H1) \\ [\![R_{id}]\!]_L = \forall(j : [\![\mathsf{int}]\!]_L).\, \forall(k : [\![\mathsf{int}]\!]_L).j = k & (H2) \end{array}$$

  From the definition of MSFOL encoding of types, we know that $[\![\mathsf{int}]\!]_L = \mathcal{F}(\mathsf{int})$. Using this to rewrite $H2$, we deduce that:

$$\phi_R^L = \forall(j : \mathcal{F}(\mathsf{int})).\, \forall(k : \mathcal{F}(\mathsf{int})).j = k \quad (H3)$$

  Applying Q-PROP-SORT twice, we also know that:

$$\cdot \vdash \forall(j : [\![\mathsf{int}]\!]_L).\, \forall(k : [\![\mathsf{int}]\!]_L).j = k \;:\; [\![\mathsf{int}]\!]_L \to [\![\mathsf{int}]\!]_L \to \mathit{bool} \quad (H4)$$

  Therefore:

$$\cdot \vdash \forall(j : \mathcal{F}(\mathsf{int})).\, \forall(k : \mathcal{F}(\mathsf{int})).j = k \;:\; [\![\mathsf{int}]\!]_L \to [\![\mathsf{int}]\!]_L \to \mathit{bool} \quad (H5)$$

  From $H3$, and rewriting $H5$ with $H1$ gives us proof.
- R is any relation that is not R_{id}: From Fig. 4:

$$[\![R]\!]_L = \eta_{wrap}(R, [\![\Gamma_R(R)]\!]_L) \quad (H0)$$

  Since $\cdot \vdash R :: \tau_R$, from the definition of ordered map Γ_R, we know that $\Gamma_R(R) = \tau_R$. Rewriting $H0$:

$$[\![R]\!]_L = \eta_{wrap}(R, \tau_R) \quad (H1)$$

  Now, applying Lemma 1.15 gives us proof.

■

LEMMA 1.17. (**Substitution Preserves Nominal Typing**) *forall* ϕ^L*, x, and y, if* $\cdot \vdash \phi^L : \tau^F$*, then* $\cdot \vdash [y/x]\,\phi^L : \tau^F$

Proof trivial, as substitution operation substitutes one variable for another in an MSFOL formula, and all variables have type *bool* under nominal type system. ■

LEMMA 1.18. ($\gamma_\sqcup$ **correctness**) *forall* ϕ_1^L*,* ϕ_2^L*, and* τ^F*, if* $\cdot \vdash \phi_1^L : \tau^F$*, and* $\cdot \vdash \phi_2^L : \tau^F$*, then there exists an MSFOL proposition* ϕ^L *such that* $\gamma_\sqcup(\phi_1^L, \odot, \phi_2^L) = \phi^L$*, and* $\cdot \vdash \phi^L : \tau^F$*.*

Proof by simultaneous induction (i.e., double induction followed by elimination of absurd cases) on nominal typing derivations $\cdot \vdash \phi_1^L : \tau^F$, and $\cdot \vdash \phi_2^L : \tau^F$. Cases:

- Case Q-PROP-SORT : τ^F is of form $T^F \rightarrow \tau_2^F$. Propositions ϕ_1^L, and ϕ_2^L are of form $\forall(k : T^F).\,\phi_{11}^L$ and $\forall(k : T^F).\,\phi_{21}^L$, respectively, such that:

$$\begin{array}{ll} \cdot \vdash \phi_{11}^L : \tau_2^F & (H0) \\ \cdot \vdash \phi_{12}^L : \tau_2^F & (H1) \end{array}$$

  From inductive hypotheses, we know that there exists an MSFOL prop ϕ_3^L such that:

$$\begin{array}{ll} \gamma_\sqcup(\phi_{11}^L, \odot, \phi_{12}^L) = \phi_3^L & (IH0) \\ \cdot \vdash \phi_3^L : \tau_2 & (IH1) \end{array}$$

  From the definition of $\gamma_\sqcup$, we have that:

$$\gamma_\sqcup(\forall(k : T^F).\,\phi_{11}^L, \odot, \forall(k : T^F).\,\phi_{12}^L) = \forall(k : T^F).\,\gamma_\sqcup(\phi_{11}^L, \odot, \phi_{12}^L) \quad (H2)$$

  Rewriting $H2$ using $IH0$, we have:

$$\gamma_\sqcup(\forall(k : T^F).\,\phi_{11}^L, \odot, \forall(k : T^F).\,\phi_{12}^L) = \forall(k : T^F).\,\phi_3^L \quad (H2)$$

  Hence, ϕ^L is $\forall(k : T^F).\,\phi_3^L$. It remains to prove that $\cdot \vdash \forall(k : T^F).\,\phi_3^L : \tau^F$, which can be done by applying Q-PROP-SORT on $IH1$.
- Case QF-PROP-SORT: $\tau^F = bool$. Inversion on $\cdot \vdash \phi_1^L : bool$, and $\cdot \vdash \phi_2^L : bool$ lets us infer that ϕ_1^L and ϕ_2^L are quantifier-free propositions ϕ_1^F and ϕ_2^F, respectively. From the definition of $\gamma_\sqcup$, we know that:

$$\gamma_\sqcup(\phi_1^F, \odot, \phi_2^F) = \phi_1^F \odot \phi_2^F$$

  Proof is obtained by observing that $\phi_1^F \odot \phi_2^F$ is a quantifier-free MSFOL formula, which, by QF-PROP-SORT rule has type *bool*.

■

LEMMA 1.19. ($\gamma_\bowtie$ **correctness**) *forall* ϕ_1^L*,* ϕ_2^L*,* τ_1^F*, and* τ_2^F*, if* $\cdot \vdash \phi_1^L : \tau_1^F$*, and* $\cdot \vdash \phi_2^L : \tau_2^F$*, then there exists an MSFOL proposition* ϕ^L *such that* $\gamma_\bowtie(\phi_1^L, \odot, \phi_2^L) = \phi^L$*, and* $\cdot \vdash \phi^L : \tau_1^F \bowtie \tau_2^F$*.*

Proof by simultaneous inductions on nominal typing derivations $\cdot \vdash \phi_1^L : \tau_1^F$, and $\cdot \vdash \phi_2^L : \tau_2^F$. We will have four cases, one for each case of join. Proof proceeds similar to the proof of Lemma 1.18. ■

LEMMA 1.20. (**Translation for relational expressions**) *Forall* Γ*, r, and* θ*, if* $\Gamma \vdash r :: \{\theta\}$*, then there exists a* ϕ^L *such that* $[\![r]\!]_L = \phi^L$*, and* $\cdot \vdash \phi^L : [\![\{\theta\}]\!]_L$*.*

Proof By induction on the sort derivation $\Gamma \vdash r :: \{\theta\}$. Cases:

- Case S-UNION : $r = r_1 \cup r_2$, for some r_1, r_2. Hypotheses:

$$\begin{array}{ll} \Gamma \vdash r_1 :: \{\theta\} & (H0) \\ \Gamma \vdash r_2 :: \{\theta\} & (H1) \end{array}$$

  From inductive hypotheses, we know that there exist two propositions, ϕ_1^L and ϕ_2^L, such that:

$$\begin{array}{ll} [\![r_1]\!]_L = \phi_1^L & (IH0) \\ [\![r_2]\!]_L = \phi_2^L & (IH0) \\ \cdot \vdash \phi_1^L : [\![\theta]\!]_L & (IH2) \\ \cdot \vdash \phi_2^L : [\![\theta]\!]_L & (IH3) \end{array}$$

We know that $[\![r_1 \cup r_2]\!]_L = \gamma_{\sqcup}([\![r_1]\!]_L, \vee, [\![r_2]\!]_L)$. Therefore, the goal is to prove that there exists a ϕ^L, such that:

$$\gamma_{\sqcup}([\![r_1]\!]_L, \vee, [\![r_2]\!]_L) = \phi^L$$
$$\cdot \vdash \phi^L : [\![\theta]\!]_L$$

Applying Lemma 1.18 using $IH2-3$ proves the goal.

- Case S-CROSS: Similar to S-UNION case. We make use of Lemma 1.19 to prove the goal.
- Case S-APP : r is of form $R\,v$, for some relation R, and λ_R value v. Hypotheses:

$$\cdot \vdash R :: T :\rightarrow \{\theta\} \quad (H0)$$
$$\|\Gamma\| \Vdash v : T \quad (H1)$$

From definition of MSFOL encoding for colon-arrow types:

$$[\![T :\rightarrow \{\theta\}]\!]_L = [\![T]\!]_L \rightarrow [\![\theta]\!]_L \quad (H2)$$

Since $T \in \{\mathsf{int}, \mathsf{intlist}\}$, we have the following cases for v:

- SubCase v is a variable x: $r = R(x)$. From the definition of MSFOL encoding:

$$[\![R(x)]\!]_L = Inst([\![R]\!]_L, x) \quad (H3)$$

From Lemma 1.16, we know that $[\![R]\!]_L$ is an MSFOL formula ϕ^L_R such that $\cdot \vdash \phi^L_R : [\![T :\rightarrow \{\theta\}]\!]_L$. Rewriting using $H2$:

$$\cdot \vdash \phi^L_R : [\![T]\!]_L \rightarrow [\![\theta]\!]_L \quad (H4)$$

By inversion on $H4$, we know that ϕ^L_R is of form $\forall(k : [\![T]\!]_L).\,\phi^L_k$, where

$$\cdot \vdash \phi^L_k : [\![\theta]\!]_L \quad (H5)$$

Rewriting $H3$:

$$[\![R(x)]\!]_L = Inst(\forall(k : [\![T]\!]_L).\,\phi^L_k, x) \quad (H6)$$

From the definition of $Inst$, $H6$ reduces to:

$$[\![R(x)]\!]_L = [x/k]\,\phi^L_k \quad (H7)$$

Therefore, $[\![R(x)]\!]_L$ is an MSFOL formula. Further, Applying substitution lemma of nominal typing (Lemma **??**) on $H5$, we also have that

$$\cdot \vdash [x/k]\,\phi^L_k : [\![\theta]\!]_L$$

This concludes the proof for current SubCase.

- SubCase v is Nil: r is $R(\mathsf{Nil})$. Hypotheses:

$$\cdot \vdash R(\mathsf{Nil}) :: \{\theta\} \quad (H0)$$

By inversion on $H0$:

$$\cdot \vdash R :: \mathsf{intlist} :\rightarrow \{\theta\} \quad (H1)$$

By inversion on $H1$, for some relational expressions r_1 and r_2:

$$R \triangleq \langle \mathsf{Nil} \Rightarrow r_1 \mid \mathsf{Cons}\,x\,y \Rightarrow r_2 \rangle \quad (H2)$$
$$\cdot \vdash r_1 :: \{\theta\} \quad (H3)$$

Using $H3$, the inductive hypothesis tells us that there exists a ϕ^L, such that

$$[\![r_1]\!]_L = \phi^L \quad (H4)$$
$$\cdot \vdash \phi^L :: [\![\theta]\!]_L \quad (H5)$$

From the definition of MSFOL encoding:

$$[\![R(\mathsf{Nil})]\!]_L = [\![\Sigma_R(R)(\mathsf{Nil})]\!]_L$$

Recall that Σ_R maps relation names to their definitions, and we treat the relation definition as a map from constructor patterns to relational expressions. Therefore, $\Sigma_R(R)(\mathsf{Nil}) = r_1$, and $[\![\Sigma_R(R)(\mathsf{Nil})]\!]_L = [\![r_1]\!]_L$. Consequently, proof follows directly from $H4-5$.

- Case v is $\mathsf{Cons}\, v_1\, v_2$, for some λ_R values v_1 and v_2. From the type of Cons under simple type system, we know that:

$$\begin{array}{c} \|\Gamma\| \Vdash v_1 : \mathsf{int} \\ \|\Gamma\| \Vdash v_2 : \mathsf{intlist} \end{array}$$

As a corollary of Lemma 1.4, from previous two hypotheses, we have:

$$\begin{array}{cc} \|\Gamma\| \vdash v_1 : \mathsf{int} & (H0) \\ \|\Gamma\| \vdash v_2 : \mathsf{intlist} & (H1) \end{array}$$

Further, we have the following hypothesis:

$$\Gamma \vdash R(\mathsf{Cons}\, v_1\, v_2) :: \{\theta\} \quad (H2)$$

By inversion on $H0$:

$$\Gamma \vdash R :: \mathsf{intlist} :\to \{\theta\} \quad (H3)$$

By inversion on $H1$, for some relational expressions r_1 and r_2:

$$\begin{array}{cc} R \triangleq \langle \mathsf{Nil} \Rightarrow r_1 \mid \mathsf{Cons}\, x\, y \Rightarrow r_2 \rangle & (H4) \\ \cdot, x : \mathsf{int}, y : \mathsf{intlist} \vdash r_2 :: \{\theta\} & (H5) \end{array}$$

From $H5$, and $H0-1$:

$$\cdot \vdash [v_2/y]\,[v_1/x]\, r_2 :: \{\theta\} \quad (H5)$$

From the definition of MSFOL encoding:

$$[\![R(\mathsf{Cons}\, v_1\, v_2)]\!]_L = [\![\Sigma_R(R)(\mathsf{Cons}\, v_1\, v_2)]\!]_L \quad (H6)$$

Using the definition of Σ_R, followed by desugaring:

$$[\![R(\mathsf{Cons}\, v_1\, v_2)]\!]_L = [v_2/y]\,[v_1/x]\, r_2 \quad (H7)$$

From $H5$, which asserts that $[v_2/y]\,[v_1/x]\, r_2$ is well-sorted under empty environment not containing type bindings for Cons and Nil, we know that arguments to relations in r_2 are smaller than $\mathsf{Cons}\, v_1\, v_2$. The goal can now be proved by induction on the size of relation arguments.

∎

THEOREM 1.21. (**Completeness of MSFOL semantics**) *Forall ϕ, Γ, if $\Gamma \vdash \phi$, then there exists an MSFOL proposition ϕ^L such that $[\![\phi]\!]_L = \phi^L$.*

Proof by induction on $\Gamma \vdash \phi$. Cases:

- Case WF-REF: $\phi = \phi_1 \wedge \phi_2$, or $\phi = \phi_1 \vee \phi_2$. From inductive hypothesis, we have that $[\![\phi_1]\!]_L$ and $[\![\phi_2]\!]_L$ are both MSFOL formulas. Since conjunctions and disjunctions of MSFOL formulas are MSFOL formulas themselves, proof follows.
- Case WF-PRED: ϕ is of form $r_1 = r_2$, or $r_1 \subseteq r_2$, for some relational expressions r_1 and r_2. Hypotheses:

$$\begin{array}{cc} \Gamma \vdash r_1 :: \{\theta\} & (H0) \\ \Gamma \vdash r_2 :: \{\theta\} & (H1) \end{array}$$

where, θ is a tuple sort. From lemma 1.20, we know that there exist two MSFOL propositions ϕ_1^L and ϕ_2^L, such that:

$$\begin{array}{cc} \cdot \vdash \phi_1^L : [\![\{\theta\}]\!]_L & (H2) \\ \cdot \vdash \phi_2^L : [\![\{\theta\}]\!]_L & (H3) \end{array}$$

Now, since:

$$\begin{array}{cc} [\![r_1 = r_2]\!]_L = \gamma_\sqcup([\![r_1]\!]_L, \Leftrightarrow, [\![r_2]\!]_L) & (H4) \\ [\![r_1 \subseteq r_2]\!]_L = \gamma_\sqcup([\![r_1]\!]_L, \Rightarrow, [\![r_2]\!]_L) & (H4) \end{array}$$

applying Lemma 1.18, using $H2-3$ gives us the proof.

∎