Online appendix to:

Polymorphic Functions with Set-Theoretic Types

Part 1: Syntax, Semantics, and Evaluation

Giuseppe Castagna[1] Kim Nguyễn[2] Zhiwu Xu[1,3] Hyeonseung Im[2] Sergueï Lenglet[4] Luca Padovani[5]

[1]CNRS, PPS, Univ Paris Diderot, Sorbonne Paris Cité, Paris, France [2]LRI, Université Paris-Sud, Orsay, France
[3]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[4] LORIA, Université de Lorraine, Nancy, France [5]Dipartimento di Informatica, Università di Torino, Italy

POPL '14, January 22–24 2014, San Diego, CA, USA.

ACM 978-1-4503-2544-8/14/01…$15.00.
http://dx.doi.org/10.1145/2535838.2535840

A. Explicitly-Typed Calculus

In this section, we define our explicitly-typed λ-calculus with sets of type-substitutions that we outlined in Section 3.

A.1 Types

Definition A.1 (**Types**). *Let $\mathscr{V}$ be a countable set of type variables ranged over by Greek letters $\alpha,\beta,\gamma,\ldots$, and $\mathscr{B}$ a finite set of basic (or constant) types ranged over by b. A* type *is a term co-inductively produced by the following grammar*

$$\begin{array}{lrll} \textbf{\textit{Types}} & t ::= & \alpha & \textit{type variable} \\ & & \mid b & \textit{basic} \\ & & \mid t \times t & \textit{product} \\ & & \mid t \to t & \textit{arrow} \\ & & \mid t \vee t & \textit{union} \\ & & \mid \neg t & \textit{negation} \\ & & \mid \mathbb{0} & \textit{empty} \end{array}$$

that satisfies two additional requirements:

- *(regularity) the term must have a finite number of different sub-terms.*
- *(contractivity) every infinite branch must contain an infinite number of occurrences of atoms (*ie*, either a type variable or the immediate application of a type constructor: basic, product, arrow).*

We use $\mathscr{T}$ to denote the set of all types.

We write $t_1 \backslash t_2$, $t_1 \wedge t_2$, and $\mathbb{1}$ respectively as an abbreviation for $t_1 \wedge \neg t_2$, $\neg(\neg t_1 \vee \neg t_2)$, and $\neg\mathbb{0}$.

Since types are infinite, the accessory definitions on them will be given either by using memoization (*eg*, the definition of $\mathsf{var}()$, the variables occurring in a type: Definition A.2), by co-inductive techniques (*eg*, the definition or the application of type-substitutions: Definition A.5), or by induction on the relation $\triangleright$, but only when induction does not traverse a type constructor (*eg*, the definition of $\mathsf{tlv}()$, the variables occurring at top-level of a type: Definition A.3).

Definition A.2 (**Type variables**). *Let var_0 and var_1 be two functions from $\mathscr{T} \times \mathscr{P}(\mathscr{T})$ to $\mathscr{P}(\mathscr{V})$ defined as:*

$$\begin{array}{rcl} var_0(t, \sqsupseteq) & = & \begin{cases} \emptyset & \textit{if } t \in \sqsupseteq \\ var_1(t, \sqsupseteq \cup \{t\}) & \textit{otherwise} \end{cases} \\ var_1(\alpha, \sqsupseteq) & = & \{\alpha\} \\ var_1(b, \sqsupseteq) & = & \emptyset \\ var_1(t_1 \times t_2, \sqsupseteq) & = & var_0(t_1, \sqsupseteq) \cup var_0(t_2, \sqsupseteq) \\ var_1(t_1 \to t_2, \sqsupseteq) & = & var_0(t_1, \sqsupseteq) \cup var_0(t_2, \sqsupseteq) \\ var_1(t_1 \vee t_2, \sqsupseteq) & = & var_1(t_1, \sqsupseteq) \cup var_1(t_2, \sqsupseteq) \\ var_1(\neg t_1, \sqsupseteq) & = & var_1(t_1, \sqsupseteq) \\ var_1(\mathbb{0}, \sqsupseteq) & = & \emptyset \end{array}$$

The set of type variables *occurring in a type t, written $\mathsf{var}(t)$, is defined as $var_0(t, \emptyset)$. A type t is said to be* ground *or* closed *if and only if $\mathsf{var}(t)$ is empty. We write $\mathscr{T}_0$ to denote the set of all the ground types.*

Definition A.3 (**Top-level variables**). *Let t be a type. The set $\mathsf{tlv}(t)$ of type variables that occur at top level in t, that is, all the variables of t that have at least one occurrence not under a constructor, is defined as:*

$$\begin{array}{rcl} \mathsf{tlv}(\alpha) & = & \{\alpha\} \\ \mathsf{tlv}(b) & = & \emptyset \\ \mathsf{tlv}(t_1 \times t_2) & = & \emptyset \\ \mathsf{tlv}(t_1 \to t_2) & = & \emptyset \\ \mathsf{tlv}(t_1 \vee t_2) & = & \mathsf{tlv}(t_1) \cup \mathsf{tlv}(t_2) \\ \mathsf{tlv}(\neg t_1) & = & \mathsf{tlv}(t_1) \\ \mathsf{tlv}(\mathbb{0}) & = & \emptyset \end{array}$$

Definition A.4 (**Type substitution**). *A* type-substitution *σ is a total mapping of type variables to types that is the identity everywhere but on a finite subset of $\mathscr{V}$, which is called the domain of σ and denoted by $\mathsf{dom}(\sigma)$. We use the notation $\{{}^{t_1}/_{\alpha_1}, \ldots, {}^{t_n}/_{\alpha_n}\}$ to denote the type-substitution that maps α_i to t_i for $i = 1..n$. Given a substitution σ, the* range *of σ is defined as the set of types $\mathsf{ran}(\sigma) = \{\sigma(\alpha) \mid \alpha \in \mathsf{dom}(\sigma)\}$, and the set of type variables occurring in the range is defined as $\mathsf{tvran}(\sigma) = \bigcup_{\alpha \in \mathsf{dom}(\sigma)} \mathsf{var}(\sigma(\alpha))$.*

Definition A.5. *Given a type $t \in \mathscr{T}$ and a type-substitution σ, the application of σ to t is co-inductively defined as follows:*

$$\begin{array}{rcll} b\sigma & = & b & \\ (t_1 \times t_2)\sigma & = & (t_1\sigma) \times (t_2\sigma) & \\ (t_1 \to t_2)\sigma & = & (t_1\sigma) \to (t_2\sigma) & \\ (t_1 \vee t_2)\sigma & = & (t_1\sigma) \vee (t_2\sigma) & \\ (\neg t)\sigma & = & \neg(t\sigma) & \\ \mathbb{0}\sigma & = & \mathbb{0} & \\ \alpha\sigma & = & \sigma(\alpha) & \text{if } \alpha \in \mathsf{dom}(\sigma) \\ \alpha\sigma & = & \alpha & \text{if } \alpha \notin \mathsf{dom}(\sigma) \end{array}$$

Definition A.6. *Let σ_1 and σ_2 be two substitutions such that $\mathsf{dom}(\sigma_1) \cap \mathsf{dom}(\sigma_2) = \emptyset$ ($\sigma_1 \sharp \sigma_2$ for short). Their* union *$\sigma_1 \cup \sigma_2$ is defined as*

$$(\sigma_1 \cup \sigma_2)(\alpha) = \begin{cases} \sigma_1(\alpha) & \alpha \in \mathsf{dom}(\sigma_1) \\ \sigma_2(\alpha) & \alpha \in \mathsf{dom}(\sigma_2) \\ \alpha & \text{otherwise} \end{cases}$$

Definition A.7. *Given two sets of type-substitutions $[\sigma_i]_{i\in I}$ and $[\sigma_j]_{j\in J}$, we define their* composition *as*

$$[\sigma_i]_{i\in I} \circ [\sigma_j]_{j\in J} = [\sigma_i \circ \sigma_j]_{i\in I, j\in J}$$

where

$$\sigma_i \circ \sigma_j(\alpha) = \begin{cases} (\sigma_j(\alpha))\sigma_i & \text{if } \alpha \in \mathsf{dom}(\sigma_j) \\ \sigma_i(\alpha) & \text{if } \alpha \in \mathsf{dom}(\sigma_i) \setminus \mathsf{dom}(\sigma_j) \\ \alpha & \text{otherwise} \end{cases}$$

A.2 Expressions

Definition A.8 (Expressions). *Let $\mathscr{C}$ be a set of constants ranged over by c and $\mathscr{X}$ a countable set of expression variables ranged over by $x,y,z,\ldots$. An* expression *e is a term inductively generated by the following grammar:*

Expressions	$e ::=$	c	*constant*
		$\mid x$	*expression variable*
		$\mid (e, e)$	*pair*
		$\mid \pi_i(e)$	*projection*$(i \in \{1,2\})$
		$\mid \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e$	*abstraction*
		$\mid e\ e$	*application*
		$\mid e \in t\ ?\ e : e$	*type case*
		$\mid e[\sigma_j]_{j\in J}$	*instantiation*

where t_i, s_i range over types, $t \in \mathscr{T}_0$ is a ground type and σ_j ranges over type-substitutions. We write $\mathscr{E}$ to denote the set of all expressions.

A λ-abstraction comes with a non-empty sequence of arrow types (called its *interface*) and a possibly empty set of type-substitutions (called its *decorations*). When the decoration is an empty set, we write $\lambda^{(\wedge_{i\in I} t_i \to s_i)} x.e$ for short.

Since expressions are inductively generated, the accessory definitions on them can be given by induction.

Given a set of type variables Δ and a set of type-substitutions $[\sigma_j]_{j\in J}$, for simplicity, we use the notation $\Delta[\sigma_j]_{j\in J}$ to denote the set of type variables occurring in the applications $\alpha\sigma_j$ for all $\alpha \in \Delta, j \in J$, that is:

$$\Delta[\sigma_j]_{j\in J} \stackrel{\text{def}}{=} \bigcup_{j\in J} \Big(\bigcup_{\alpha\in\Delta} \mathsf{var}(\sigma_j(\alpha)) \Big)$$

Definition A.9. *Let e be an expression. The set $\mathsf{fv}(e)$ of* free variables *of the expression e is defined by induction as:*

$$\begin{array}{rcl} \mathsf{fv}(x) & = & \{x\} \\ \mathsf{fv}(c) & = & \emptyset \\ \mathsf{fv}((e_1, e_2)) & = & \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\ \mathsf{fv}(\pi_i(e)) & = & \mathsf{fv}(e) \\ \mathsf{fv}(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e) & = & \mathsf{fv}(e) \setminus \{x\} \\ \mathsf{fv}(e_1\ e_2) & = & \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\ \mathsf{fv}(e \in t\ ?\ e_1 : e_2) & = & \mathsf{fv}(e) \cup \mathsf{fv}(e_1) \cup \mathsf{fv}(e_2) \\ \mathsf{fv}(e[\sigma_j]_{j\in J}) & = & \mathsf{fv}(e) \end{array}$$

The set $\mathsf{bv}(e)$ *of* bound variables *of the expression* e *is defined by induction as:*

$$\begin{aligned}
\mathsf{bv}(x) &= \emptyset \\
\mathsf{bv}(c) &= \emptyset \\
\mathsf{bv}((e_1, e_2)) &= \mathsf{bv}(e_1) \cup \mathsf{bv}(e_2) \\
\mathsf{bv}(\pi_i(e)) &= \mathsf{bv}(e) \\
\mathsf{bv}(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e) &= \mathsf{bv}(e) \cup \{x\} \\
\mathsf{bv}(e_1\ e_2) &= \mathsf{bv}(e_1) \cup \mathsf{bv}(e_2) \\
\mathsf{bv}(e{\in}t\,?\,e_1 : e_2) &= \mathsf{bv}(e) \cup \mathsf{bv}(e_1) \cup \mathsf{bv}(e_2) \\
\mathsf{bv}(e[\sigma_j]_{j\in J}) &= \mathsf{bv}(e)
\end{aligned}$$

The set $\mathsf{tv}(e)$ *of* type variables *occurring in* e *is defined by induction as:*

$$\begin{aligned}
\mathsf{tv}(x) &= \emptyset \\
\mathsf{tv}(c) &= \emptyset \\
\mathsf{tv}((e_1, e_2)) &= \mathsf{tv}(e_1) \cup \mathsf{tv}(e_2) \\
\mathsf{tv}(\pi_i(e)) &= \mathsf{tv}(e) \\
\mathsf{tv}(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e) &= \mathsf{tv}(e[\sigma_j]_{j\in J}) \cup \mathsf{var}(\textstyle\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j) \\
\mathsf{tv}(e_1\ e_2) &= \mathsf{tv}(e_1) \cup \mathsf{tv}(e_2) \\
\mathsf{tv}(e{\in}t\,?\,e_1 : e_2) &= \mathsf{tv}(e) \cup \mathsf{tv}(e_1) \cup \mathsf{tv}(e_2) \\
\mathsf{tv}(e[\sigma_j]_{j\in J}) &= (\mathsf{tv}(e))[\sigma_j]_{j\in J}
\end{aligned}$$

An expression e *is* closed *if* $\mathsf{fv}(e)$ *is empty.*

Note that the set of type variables in $e[\sigma_j]_{j\in J}$ is the set returned by the "application" of $[\sigma_j]_{j\in J}$ to the set $\mathsf{tv}(e)$.

Next, we define expression substitutions. Recall that, as stated in Section 3.1, when substituting an expression e for a variable y in an expression e', we suppose the polymorphic type variables of e' to be distinct from the monomorphic and polymorphic type variables of e to avoid unwanted captures. In the discussion in Section 3.1, the definitions of the notions of polymorphic and monomorphic variables remain informal. To make them more formal, we would have to distinguish between the two by carrying around a set of type variables Δ which would contain the monomorphic variables that cannot be α-converted. Then all definitions (such as expression substitutions, for example) would have to be parameterized with Δ, making the definitions and technical developments difficult to read just because of α-conversion. Therefore, for the sake of readability, we decided to keep the distinction between polymorphic and monomorphic variables informal.

Definition A.10 (Expression substitution). *An* expression substitution ϱ *is a total mapping of expression variables to expressions that is the identity everywhere but on a finite subset of* $\mathscr{X}$*, which is called the domain of* ϱ *and denoted by* $\mathsf{dom}(\varrho)$*. We use the notation* $\{{}^{e_1}/_{x_1}, \ldots, {}^{e_n}/_{x_n}\}$ *to denote the expression substitution that maps* x_i *into* e_i *for* $i = 1..n$*.*

The definitions of free variables, bound variables, and type variables are extended to expression substitutions as follows.

$$\mathsf{fv}(\varrho) = \bigcup_{x\in\mathsf{dom}(\varrho)} \mathsf{fv}(\varrho(x)), \quad \mathsf{bv}(\varrho) = \bigcup_{x\in\mathsf{dom}(\varrho)} \mathsf{bv}(\varrho(x)), \quad \mathsf{tv}(\varrho) = \bigcup_{x\in\mathsf{dom}(\varrho)} \mathsf{tv}(\varrho(x))$$

Next, we define the application of an expression substitution ϱ to an expression e. To avoid unwanted captures, we remind that we assume that the bound variables of e do not occur in the domain of ϱ and that the polymorphic type variables of e are distinct from the type variables occurring in ϱ (using α-conversion if necessary).

Definition A.11. *Given an expression* $e \in \mathscr{E}$ *and an expression substitution* ϱ*, the application of* ϱ *to* e *is defined as follows:*

$$\begin{aligned}
c\varrho &= c \\
(e_1, e_2)\varrho &= (e_1\varrho, e_2\varrho) \\
(\pi_i(e))\varrho &= \pi_i(e\varrho) \\
(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e)\varrho &= \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.(e\varrho) \\
(e_1\ e_2)\varrho &= (e_1\varrho)\ (e_2\varrho) \\
(e{\in}t\,?\,e_1 : e_2)\varrho &= e\varrho{\in}t\,?\,e_1\varrho : e_2\varrho \\
x\varrho &= \varrho(x) && \text{if } x \in \mathsf{dom}(\varrho) \\
x\varrho &= x && \text{if } x \notin \mathsf{dom}(\varrho) \\
(e[\sigma_j]_{j\in J})\varrho &= (e\varrho)[\sigma_j]_{j\in J}
\end{aligned}$$

In the case for instantiation $(e[\sigma_j]_{j\in J})\varrho$, the σ_j operate on the polymorphic type variables, which we assume distinct from the variables in ϱ (using α-conversion if necessary). As a result, we have $\mathsf{tv}(\varrho) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$. Similarly, in the abstraction case, we have $x \notin \mathsf{dom}(\varrho)$.

Next, we define the relabeling of an expression e with a set of type-substitutions $[\sigma_j]_{j\in J}$, which consists in propagating the σ_j to the λ-abstractions in e if needed. We suppose that the polymorphic type variables in e are distinct from the type variables in the range of σ_j (this is always possible using α-conversion).

Definition A.12 (Relabeling). *Given an expression $e \in \mathscr{E}$ and a set of type-substitutions $[\sigma_j]_{j\in J}$, we define the* relabeling *of e with $[\sigma_j]_{j\in J}$, written $e@[\sigma_j]_{j\in J}$, as e if $\mathsf{tv}(e) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$, and otherwise as follows:*

$$\begin{array}{rcl}
(e_1, e_2)@[\sigma_j]_{j\in J} & = & (e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J}) \\
(\pi_i(e))@[\sigma_j]_{j\in J} & = & \pi_i(e@[\sigma_j]_{j\in J}) \\
(e_1\ e_2)@[\sigma_j]_{j\in J} & = & (e_1@[\sigma_j]_{j\in J})\ (e_2@[\sigma_j]_{j\in J}) \\
(e{\in}t\ ?\ e_1 : e_2)@[\sigma_j]_{j\in J} & = & e@[\sigma_j]_{j\in J}{\in}t\ ?\ e_1@[\sigma_j]_{j\in J} : e_2@[\sigma_j]_{j\in J} \\
(e[\sigma_i]_{i\in I})@[\sigma_j]_{j\in J} & = & e@([\sigma_j]_{j\in J} \circ [\sigma_i]_{i\in I}) \\
(\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_k]_{k\in K}} x.e)@[\sigma_j]_{j\in J} & = & \lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K}} x.e
\end{array}$$

The substitutions are not propagated if they do not affect the variables of e (i.e., if $\mathsf{tv}(e) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$). In particular, constants and variables are left unchanged, as they do not contain any type variable. Suppose now that $\mathsf{tv}(e) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) \neq \emptyset$. In the abstraction case, the propagated substitutions are composed with the decorations of the abstraction, without propagating them further down in the body. Propagation in the body occurs, whenever is needed, that is, during either reduction (see (*Rappl*) in Section A.4) or type-checking (see (*abstr*) in Section A.3). In the instantiation case $e[\sigma_i]_{i\in I}$, we propagate the result of the composition of $[\sigma_i]_{i\in I}$ with $[\sigma_j]_{j\in J}$ in e. The remaining cases are simple inductive cases. Finally notice that in a type case $e{\in}t\ ?\ e_1 : e_2$, we do not apply $[\sigma_j]_{j\in J}$ to t, simply because t is ground.

A.3 Type System

Because of the type directed nature of our calculus (*ie*, the presence of the type-case expression), its dynamic semantics is defined only for well-typed expressions. Therefore, we introduce the type system before giving the reduction rules.

Definition A.13 (Typing environment). *A* typing environment Γ *is a finite mapping from expression variables $\mathscr{X}$ to types $\mathscr{T}$, and written as a finite set of pairs $\{(x_1 : t_1), \ldots, (x_n : t_n)\}$. The set of expression variables defined in Γ is called the domain of Γ, denoted by $\mathsf{dom}(\Gamma)$. The set of type variables occurring in Γ, that is, $\bigcup_{(x:t)\in\Gamma} \mathsf{var}(t)$, is denoted by $\mathsf{var}(\Gamma)$. If Γ is a type environment, then $\Gamma, (x : t)$ is the type environment defined as*

$$(\Gamma, (x : t))(y) = \begin{cases} t & \text{if } y = x \\ \Gamma(y) & \text{otherwise} \end{cases}$$

We extend the definition of type-substitution application to type environments by applying the type-substitution to each type in the type environment as follows:

$$\Gamma\sigma = \{(x : t\sigma) \mid (x : t) \in \Gamma\}$$

The typing judgment for expressions has the form $\Delta \fatsemi \Gamma \vdash e : t$, which states that under the set Δ of (monomorphic) type variables and the typing environment Γ the expression e has type t. When Δ and Γ are both empty, we write $\vdash e : t$ for short. We assume that there is a basic type b_c for each constant c. We write $\sigma \sharp \Delta$ as abbreviation for $\mathsf{dom}(\sigma) \cap \Delta = \emptyset$. The typing rules are given in Figure 3 which are the same as in Section 3 except for the rules for products.

A.4 Operational Semantics

Definition A.14 (Values). *An expression e is a* value *if it is closed, well-typed (*ie, $\vdash e : t$ *for some type t), and produced by the following grammar:*

$$\textbf{\textit{Values}} \qquad v ::= c \mid (v, v) \mid \lambda^{(\wedge_{i\in I} t_i\to s_i)}_{[\sigma_j]_{j\in J}} x.e$$

We write $\mathscr{V}$ to denote the set of all values.

Definition A.15 (Context). *Let the symbol $[\_]$ denote a hole. A* context *$C[\_]$ is an expression with a hole:*

$$\begin{array}{lrcl}
\textbf{\textit{Contexts}} & C[_] & ::= & [_] \\
& & \mid & (C[_], e) \mid (e, C[_]) \\
& & \mid & C[_]\ e \mid e\ C[_] \\
& & \mid & C[_]{\in}t\ ?\ e : e \mid e{\in}t\ ?\ C[_] : e \mid e{\in}t\ ?\ e : C[_] \\
& & \mid & \pi_i(C[_])
\end{array}$$

An evaluation context *$E[\_]$ is a context that implements outermost leftmost reduction:*

$$\begin{array}{lrcl}
\textbf{\textit{Evaluation Contexts}} & E[_] & ::= & [_] \\
& & \mid & (E[_], e) \mid (v, E[_]) \\
& & \mid & E[_]\ e \mid v\ E[_] \\
& & \mid & E[_]{\in}t\ ?\ e : e \\
& & \mid & \pi_i(E[_])
\end{array}$$

We use $C[e]$ and $E[e]$ to denote the expressions obtained by replacing e for the hole in $C[\_]$ and $E[\_]$, respectively.

$$\frac{}{\Delta \fatsemi \Gamma \vdash c : b_c}\ (const) \qquad \frac{}{\Delta \fatsemi \Gamma \vdash x : \Gamma(x)}\ (var) \qquad \frac{\Delta \fatsemi \Gamma \vdash e_1 : t_1 \qquad \Delta \fatsemi \Gamma \vdash e_2 : t_2}{\Delta \fatsemi \Gamma \vdash (e_1, e_2) : t_1 \times t_2}\ (pair)$$

$$\frac{\Delta \fatsemi \Gamma \vdash e : t_1 \times t_2}{\Delta \fatsemi \Gamma \vdash \pi_i(e) : t_i}\ (proj) \qquad \frac{\Delta \fatsemi \Gamma \vdash e_1 : t_1 \to t_2 \qquad \Delta \fatsemi \Gamma \vdash e_2 : t_1}{\Delta \fatsemi \Gamma \vdash e_1 e_2 : t_2}\ (appl)$$

$$\frac{\begin{array}{c}\Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i \in I, j \in J} t_i\sigma_j \to s_i\sigma_j) \\ \forall i \in I, j \in J.\ \Delta' \fatsemi \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s_i\sigma_j\end{array}}{\Delta \fatsemi \Gamma \vdash \lambda^{\wedge_{i \in I} t_i \to s_i}_{[\sigma_j]_{j \in J}} x.e : \bigwedge_{i \in I, j \in J} t_i\sigma_j \to s_i\sigma_j}\ (abstr)$$

$$\frac{\Delta \fatsemi \Gamma \vdash e : t' \qquad \left\{\begin{array}{lcl} t' \not\leq \neg t & \Rightarrow & \Delta \fatsemi \Gamma \vdash e_1 : s \\ t' \not\leq t & \Rightarrow & \Delta \fatsemi \Gamma \vdash e_2 : s \end{array}\right.}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}\ (case) \qquad \frac{\Delta \fatsemi \Gamma \vdash e : t \qquad \sigma \nVdash \Delta}{\Delta \fatsemi \Gamma \vdash e[\sigma] : t\sigma}\ (inst)$$

$$\frac{\forall j \in J.\ \Delta \fatsemi \Gamma \vdash e[\sigma_j] : t_j \qquad |J| > 1}{\Delta \fatsemi \Gamma \vdash e[\sigma_j]_{j \in J} : \bigwedge_{j \in J} t_j}\ (inter) \qquad \frac{\Delta \fatsemi \Gamma \vdash e : s \qquad s \leq t}{\Delta \fatsemi \Gamma \vdash e : t}\ (subsum)$$

Figure 3. Typing rules

We define a small-step call-by-value operational semantics for the calculus. The semantics is given by the relation $\rightsquigarrow$, which is shown in Figure 4. There are four notions of reduction: one for projections, one for applications, one for type cases, and one for instantiations, plus context closure. Henceforth we will establish all the properties for the reduction using generic contexts but, of course, these holds also when the more restrictive evaluation contexts are used.

Notions of reduction:

(*Rproj*) $\pi_i(v_1, v_2) \rightsquigarrow v_i$

(*Rappl*) $(\lambda^{\wedge_{i \in I} t_i \to s_i}_{[\sigma_j]_{j \in J}} x.e')v \rightsquigarrow (e'@[\sigma_j]_{j \in P})\{v/x\}$
where $P = \{j \in J \mid \exists i \in I.\ \vdash v : t_i\sigma_j\}$

(*Rcase*) $(v \in t\,?\,e_1 : e_2) \rightsquigarrow \left\{\begin{array}{ll} e_1 & \text{if } \vdash v : t \\ e_2 & \text{otherwise}\end{array}\right.$

(*Rinst*) $e[\sigma_j]_{j \in J} \rightsquigarrow e@[\sigma_j]_{j \in J}$

Context closure:

(*Rctx*) $\dfrac{e \rightsquigarrow e'}{C[e] \rightsquigarrow C[e']}$

Figure 4. Operational semantics of the calculus

The (*Rproj*) rule is the standard projection rule while the other notions of reduction have already been explained in Section 3.3.

We used a call-by-value semantics for application to ensure the type soundness property: subject reduction (or type preservation) and progress (closed and well-typed expressions which are not values can be reduced), which are discussed in Section B.2. To understand why, consider each basic reduction rule in turn.

The requirement that the argument of a projection must be a value is imposed to ensure that the property of subject reduction holds. Consider the expression $e = \pi_1(e_1, e_2)$ where e_1 is an expression of type t_1 (different from $\mathbb{0}$) and e_2 is a (diverging) expression of type $\mathbb{0}$. Clearly, the type system assigns the type $t_1 \times \mathbb{0}$ to (e_1, e_2). In our system, a product type with an empty component is itself empty, and thus e has type $\mathbb{0}$. Therefore the type of the projection as well has type $\mathbb{0}$ (since $\mathbb{0} \leq \mathbb{0} \times \mathbb{0}$, then by subsumption

$(e_1, e_2) : \mathbb{0} \times \mathbb{0}$ and the result follows from the (*proj*) typing rule). If it were possible to reduce a projection when the argument is not a value, then e could be reduced to e_1, which has type t_1: type preservation would be violated.

Likewise, the reduction rule for applications requires the argument to be a value. Let us consider the application $(\lambda^{(t\to t\times t)\wedge(s\to s\times s)}x.(x,x))(e)$, where $\vdash e : t \vee s$. The type system assigns to the abstraction the type $(t \to t \times t) \wedge (s \to s \times s)$, which is a subtype of $(t \vee s) \to ((t \times t) \vee (s \times s))$. By subsumption, the abstraction has type $(t \vee s) \to ((t \times t) \vee (s \times s))$, and thus, the application has type $(t \times t) \vee (s \times s)$. If the semantics permits to reduce an application when the argument is not a value, then this application could be reduced to the expression (e, e), which has type $(t \vee s) \times (t \vee s)$ but not $(t \times t) \vee (s \times s)$.

Finally, if we allowed $(e{\in}t\,?\,e_1 : e_2)$ to reduce to e_1 when $\vdash e : t$ but e is not a value, we could break type preservation. For example, assume that $\vdash e : \mathbb{0}$. Then the type system would not check anything about the branches e_1 and e_2 (see the typing rule (*case*) in Figure 3) and so e_1 could be ill-typed.

Notice that in all these cases the usage of values ensures subject reduction but it is not a necessary condition: in some cases weaker constraints could be used. For instance, in order to check whether an expression is a list of integers, in general it is not necessary to fully evaluate the whole list: the head and the type of the tail are all that is needed. Studying weaker conditions for the reduction rules is an interesting topic, which we leave for future work, in particular, in the view of adapting our framework to lazy languages.

B. Properties of the Type System

In this section we present some properties of our type system. First, we study its syntactic meta-theory: in particular, we prove admissibility of the intersection rule, a generation lemma for values, and that substitutions preserve typing. These properties are needed to prove *soundness*, the fundamental property which links every type system of a calculus with its operational counterpart: well-typed expressions do not go wrong. Next, we prove that the explicitly-typed calculus is able to derive the same typing judgments as the BCD intersection type system defined by Barendregt, Coppo, and Dezani [1]. Finally, we prove that the expressions of the form $e[\sigma_j]_{j\in J}$ are redundant insofar as their presence in the calculus does not increase its expressive power.

B.1 Syntactic meta-theory

Lemma B.1. *If $\Delta \fatsemi \Gamma \vdash e : t$ and $\mathsf{var}(\Gamma) \subseteq \Delta$, then $\mathsf{var}(\Gamma') \subseteq \Delta'$ holds for every judgment $\Delta' \fatsemi \Gamma' \vdash e' : t'$ in the derivation of $\Delta \fatsemi \Gamma \vdash e : t$.*

Proof. By induction on the derivation of $\Delta \fatsemi \Gamma \vdash e : t$. □

Lemma B.2 (Admissibility of intersection introduction). *Let e be an expression. If $\Delta \fatsemi \Gamma \vdash e : t$ and $\Delta \fatsemi \Gamma \vdash e : t'$, then $\Delta \fatsemi \Gamma \vdash e : t \wedge t'$.*

Proof. The proof proceeds by induction on the two typing derivations. First, assume that these two derivations end with an instance of the same rule corresponding to the top-level constructor of e.

(*const*)**:** both derivations end with an instance of (*const*):

$$\frac{}{\Delta \fatsemi \Gamma \vdash c : b_c}\ (const) \qquad \frac{}{\Delta \fatsemi \Gamma \vdash c : b_c}\ (const)$$

Trivially, we have $b_c \wedge b_c \simeq b_c$, by subsumption, the result follows.

(*var*)**:** both derivations end with an instance of (*var*):

$$\frac{}{\Delta \fatsemi \Gamma \vdash x : \Gamma(x)}\ (var) \qquad \frac{}{\Delta \fatsemi \Gamma \vdash x : \Gamma(x)}\ (var)$$

Trivially, we have $\Gamma(x) \wedge \Gamma(x) \simeq \Gamma(x)$, by subsumption, the result follows.

(*pair*)**:** both derivations end with an instance of (*pair*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_1 : t_1} \quad \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_2 : t_2}}{\Delta \fatsemi \Gamma \vdash (e_1, e_2) : (t_1 \times t_2)}\ (pair) \qquad \frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_1 : t'_1} \quad \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_2 : t'_2}}{\Delta \fatsemi \Gamma \vdash (e_1, e_2) : (t'_1 \times t'_2)}\ (pair)$$

By induction, we have $\Delta \fatsemi \Gamma \vdash e_i : (t_i \wedge t'_i)$. Then the rule (*pair*) gives us $\Delta \fatsemi \Gamma \vdash (e_1, e_2) : (t_1 \wedge t'_1) \times (t_2 \wedge t'_2)$. Moreover, because intersection distributes over products, we have $(t_1 \wedge t'_1) \times (t_2 \wedge t'_2) \simeq (t_1 \times t_2) \wedge (t'_1 \times t'_2)$. Then by (*subsum*), we have $\Delta \fatsemi \Gamma \vdash (e_1, e_2) : (t_1 \times t_2) \wedge (t'_1 \times t'_2)$.

(*proj*)**:** both derivations end with an instance of (*proj*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : t_1 \times t_2}}{\Delta \fatsemi \Gamma \vdash \pi_i(e') : t_i}\ (proj) \qquad \frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : t'_1 \times t'_2}}{\Delta \fatsemi \Gamma \vdash \pi_i(e') : t'_i}\ (proj)$$

By induction, we have $\Delta \fatsemi \Gamma \vdash e' : (t_1 \times t_2) \wedge (t'_1 \times t'_2)$. Since $(t_1 \wedge t'_1) \times (t_2 \wedge t'_2) \simeq (t_1 \times t_2) \wedge (t'_1 \times t'_2)$ (see the case of (*pair*)), by (*subsum*), we have $\Delta \fatsemi \Gamma \vdash e' : (t_1 \wedge t'_1) \times (t_2 \wedge t'_2)$. Then the rule (*proj*) gives us $\Delta \fatsemi \Gamma \vdash \pi_i(e') : t_i \wedge t'_i$.

(*appl*)**:** both derivations end with an instance of (*appl*):

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : t_1 \to t_2} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : t_1}}{\Delta \wr \Gamma \vdash e_1\ e_2 : t_2}\ (appl)$$

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : t_1' \to t_2'} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : t_1'}}{\Delta \wr \Gamma \vdash e_1\ e_2 : t_2'}\ (appl)$$

By induction, we have $\Delta \wr \Gamma \vdash e_1 : (t_1 \to t_2) \wedge (t_1' \to t_2')$ and $\Delta \wr \Gamma \vdash e_2 : t_1 \wedge t_1'$. Because intersection distributes over arrows, we have $(t_1 \to t_2) \wedge (t_1' \to t_2') \leq (t_1 \wedge t_1') \to (t_2 \wedge t_2')$. Then by the rule (*subsum*), we get $\Delta \wr \Gamma \vdash e_1 : (t_1 \wedge t_1') \to (t_2 \wedge t_2')$. Finally, by applying (*appl*), we get $\Delta \wr \Gamma \vdash e_1\ e_2 : t_2 \wedge t_2'$ as expected.

(*abstr*)**:** both derivations end with an instance of (*abstr*):

$$\dfrac{\begin{array}{l}\forall i \in I, j \in J.\ \dfrac{\cdots}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e'@[\sigma_j] : s_i\sigma_j} \\ \Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j)\end{array}}{\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j}$$

$$\dfrac{\begin{array}{l}\forall i \in I, j \in J.\ \dfrac{\cdots}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e'@[\sigma_j] : s_i\sigma_j} \\ \Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j)\end{array}}{\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j}$$

It is clear that

$$(\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j) \wedge (\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j) \simeq \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j$$

By subsumption, the result follows.

(*case*)**:** both derivations end with an instance of (*case*):

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_0 : t_0} \quad \begin{cases} t_0 \not\leq \neg t & \Rightarrow \ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : s} \\ t_0 \not\leq t & \Rightarrow \ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : s} \end{cases}}{\Delta \wr \Gamma \vdash (e_0 \in t\ ?\ e_1 : e_2) : s}\ (case)$$

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_0 : t_0'} \quad \begin{cases} t_0' \not\leq \neg t & \Rightarrow \ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : s'} \\ t_0' \not\leq t & \Rightarrow \ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : s'} \end{cases}}{\Delta \wr \Gamma \vdash (e_0 \in t\ ?\ e_1 : e_2) : s'}\ (case)$$

By induction, we have $\Delta \wr \Gamma \vdash e_0 : t_0 \wedge t_0'$. Suppose $t_0 \wedge t_0' \not\leq \neg t$; then $t_0 \not\leq \neg t$ and $t_0' \not\leq \neg t$. Consequently, the branch e_1 has been type-checked in both cases, and we have $\Delta \wr \Gamma \vdash e_1 : s \wedge s'$ by the induction hypothesis. Similarly, if $t_0 \wedge t_0' \not\leq t$, then we have $\Delta \wr \Gamma \vdash e_2 : s \wedge s'$. Consequently, we have $\Delta \wr \Gamma \vdash (e_0 \in t\ ?\ e_1 : e_2) : s \wedge s'$ by the rule (*case*).

(*inst*)**:** both derivations end with an instance of (*inst*):

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : t} \quad \sigma \sharp \Delta}{\Delta \wr \Gamma \vdash e'[\sigma] : t\sigma}\ (inst) \qquad \dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : t'} \quad \sigma \sharp \Delta}{\Delta \wr \Gamma \vdash e'[\sigma] : t'\sigma}\ (inst)$$

By induction, we have $\Delta \wr \Gamma \vdash e' : t \wedge t'$. Since $\sigma \sharp \Delta$, the rule (*inst*) gives us $\Delta \wr \Gamma \vdash e'[\sigma] : (t \wedge t')\sigma$, that is $\Delta \wr \Gamma \vdash e'[\sigma] : (t\sigma) \wedge (t'\sigma)$.

(*inter*)**:** both derivations end with an instance of (*inter*):

$$\dfrac{\forall j \in J.\ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e'[\sigma_j] : t_j}}{\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j}\ (inter) \qquad \dfrac{\forall j \in J.\ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e'[\sigma_j] : t_j'}}{\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j'}\ (inter)$$

where $|J| > 1$. By induction, we have $\Delta \wr \Gamma \vdash e'[\sigma_j] : t_j \wedge t_j'$ for all $j \in J$. Then the rule (*inter*) gives us $\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J}(t_j \wedge t_j')$, that is, $\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : (\bigwedge_{j\in J} t_j) \wedge (\bigwedge_{j\in J} t_j')$.

Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), for instance,

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : s} \quad s \leq t}{\Delta \wr \Gamma \vdash e' : t}\ (subsum) \qquad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : t'}$$

By induction, we have $\Delta \wr \Gamma \vdash e' : s \wedge t'$. Since $s \leq t$, we have $s \wedge t' \leq t \wedge t'$. Then the rule (*subsum*) gives us $\Delta \wr \Gamma \vdash e' : t \wedge t'$ as expected. □

Lemma B.3 (**Generation for values**). *Let v be a value. Then*

1. *If $\Delta \fatsemi \Gamma \vdash v : b$, then v is a constant c and $b_c \leq b$.*
2. *If $\Delta \fatsemi \Gamma \vdash v : t_1 \times t_2$, then v has the form of (v_1, v_2) with $\Delta \fatsemi \Gamma \vdash v_i : t_i$.*
3. *If $\Delta \fatsemi \Gamma \vdash v : t \to s$, then v has the form of $\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$ with $\bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j) \leq t \to s$.*

Proof. By a simple examination of the rules it is easy to see that a derivation for $\Delta \fatsemi \Gamma \vdash v : t$ is always formed by an instance of the rule corresponding to the kind of v (*i.e.*, (*const*) for constants, (*pair*) for pairs, and (*abstr*) for abstractions), followed by zero or more instances of (*subsum*). By induction on the depth of the derivation it is then easy to prove that if $\Delta \fatsemi \Gamma \vdash v : t$ is derivable, then $t \not\simeq \mathbb{0}$. The lemma then follows by induction on the number of the instances of the subsumption rule that end the derivation of $\Delta \fatsemi \Gamma \vdash v : t$. The base cases are straightforward, while the inductive cases are:

$\underline{\Delta \fatsemi \Gamma \vdash v : b}$**:** v is by induction a constant c such that $b_c \leq b$.

$\underline{\Delta \fatsemi \Gamma \vdash v : t_1 \times t_2}$**:** v is by induction a pair (v_1, v_2) and t' is form of $(t'_1 \times t'_2)$ such that $\Delta \fatsemi \Gamma \vdash v_i : t'_i$. Here we use the fact that the type of a value cannot be $\mathbb{0}$: since $\mathbb{0} \not\simeq (t'_1 \times t'_2) \leq (t_1 \times t_2)$, then we have $t'_i \leq t_i$. Finally, by (*subsum*), we have $\Delta \fatsemi \Gamma \vdash v_i : t_i$.

$\underline{\Delta \fatsemi \Gamma \vdash v : t \to s}$**:** v is by induction an abstraction $\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$ such that $\bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j) \leq t \to s$.

□

Lemma B.4. *Let e be an expression and $[\sigma_j]_{j\in J}, [\sigma_k]_{k\in K}$ two sets of type substitutions. Then*

$$(e@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} = e@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})$$

Proof. By induction on the structure of e.

$\underline{e = c}$**:**

$$\begin{array}{rcl}(c@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& c@[\sigma_k]_{k\in K}\\ &=& c\\ &=& c@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

$\underline{e = x}$**:**

$$\begin{array}{rcl}(x@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& x@[\sigma_k]_{k\in K}\\ &=& x\\ &=& x@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

$\underline{e = (e_1, e_2)}$**:**

$$\begin{array}{rcll}((e_1, e_2)@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& (e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K}\\ &=& ((e_1@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K}, (e_2@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K}) & \text{(by induction)}\\ &=& (e_1@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J}), e_2@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J}))\\ &=& (e_1, e_2)@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

$\underline{e = \pi_i(e')}$**:**

$$\begin{array}{rcll}(\pi_i(e')@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& (\pi_i(e'@[\sigma_j]_{j\in J}))@[\sigma_k]_{k\in K}\\ &=& \pi_i((e'@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K}) & \text{(by induction)}\\ &=& \pi_i(e'@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J}))\\ &=& \pi_i(e')@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

$\underline{e = e_1 e_2}$**:**

$$\begin{array}{rcll}((e_1e_2)@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& ((e_1@[\sigma_j]_{j\in J})(e_2@[\sigma_j]_{j\in J}))@[\sigma_k]_{k\in K}\\ &=& ((e_1@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K})((e_2@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K}) & \text{(by induction)}\\ &=& (e_1@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J}))(e_2@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J}))\\ &=& (e_1e_2)@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

$\underline{e = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_{j'}]_{j'\in J'}} x.e'}$**:**

$$\begin{array}{rcl}((\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_{j'}]_{j'\in J'}} x.e')@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& (\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J} \circ [\sigma_{j'}]_{j'\in J'}} x.e')@[\sigma_k]_{k\in K}\\ &=& (\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J} \circ [\sigma_{j'}]_{j'\in J'}} x.e')\\ &=& (\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_{j'}]_{j'\in J'}} x.e')@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

$\underline{e = e_0 \in t \,?\, e_1 : e_2}$**:** similar to $e = (e_1, e_2)$

$\underline{e = e'[\sigma_i]_{i\in I}}$**:**

$$\begin{array}{rcll}((e'[\sigma_i]_{i\in I})@[\sigma_j]_{j\in J})@[\sigma_k]_{k\in K} &=& (e'@([\sigma_j]_{j\in J} \circ [\sigma_i]_{i\in I}))@[\sigma_k]_{k\in K}\\ &=& e'@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J} \circ [\sigma_i]_{i\in I}) & \text{(by induction)}\\ &=& (e'[\sigma_i]_{i\in I})@([\sigma_k]_{k\in K} \circ [\sigma_j]_{j\in J})\end{array}$$

□

Lemma B.5. *Let e be an expression, ϱ an expression substitution and $[\sigma_j]_{j\in J}$ a set of type substitutions such that $\mathsf{tv}(\varrho) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$. Then $(e\varrho)@[\sigma_j]_{j\in J} = (e@[\sigma_j]_{j\in J})\varrho$.*

Proof. By induction on the structure of e.

$\underline{e = c}$**:**

$$\begin{aligned}(c\varrho)@[\sigma_j]_{j\in J} &= c@[\sigma_j]_{j\in J}\\ &= c\\ &= c\varrho\\ &= (c@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = x}$**:** if $x \notin \mathsf{dom}(\varrho)$, then

$$\begin{aligned}(x\varrho)@[\sigma_j]_{j\in J} &= x@[\sigma_j]_{j\in J}\\ &= x\\ &= x\varrho\\ &= (x@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

Otherwise, let $\varrho(x) = e'$. As $\mathsf{tv}(\varrho) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$, we have $\mathsf{tv}(e') \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$. Then

$$\begin{aligned}(x\varrho)@[\sigma_j]_{j\in J} &= e'@[\sigma_j]_{j\in J}\\ &= e'\\ &= x\varrho\\ &= (x@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = (e_1, e_2)}$**:**

$$\begin{aligned}((e_1,e_2)\varrho)@[\sigma_j]_{j\in J} &= (e_1\varrho, e_2\varrho)@[\sigma_j]_{j\in J}\\ &= ((e_1\varrho)@[\sigma_j]_{j\in J}, (e_2\varrho)@[\sigma_j]_{j\in J})\\ &= ((e_1@[\sigma_j]_{j\in J})\varrho, (e_2@[\sigma_j]_{j\in J})\varrho) \quad \text{(by induction)}\\ &= (e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J})\varrho\\ &= ((e_1,e_2)@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = \pi_i(e')}$**:**

$$\begin{aligned}(\pi_i(e')\varrho)@[\sigma_j]_{j\in J} &= (\pi_i(e'\varrho))@[\sigma_j]_{j\in J}\\ &= \pi_i((e'\varrho)@[\sigma_j]_{j\in J})\\ &= \pi_i((e_1@[\sigma_j]_{j\in J})\varrho) \quad \text{(by induction)}\\ &= \pi_i(e'@[\sigma_j]_{j\in J})\varrho\\ &= ((\pi_i(e'))@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = e_1e_2}$**:**

$$\begin{aligned}((e_1e_2)\varrho)@[\sigma_j]_{j\in J} &= ((e_1\varrho)(e_2\varrho))@[\sigma_j]_{j\in J}\\ &= ((e_1\varrho)@[\sigma_j]_{j\in J})((e_2\varrho)@[\sigma_j]_{j\in J})\\ &= ((e_1@[\sigma_j]_{j\in J})\varrho)((e_2@[\sigma_j]_{j\in J})\varrho) \quad \text{(by induction)}\\ &= ((e_1@[\sigma_j]_{j\in J})(e_2@[\sigma_j]_{j\in J}))\varrho\\ &= ((e_1e_2)@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = \lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_k]_{k\in K}} x.e'}$**:**

$$\begin{aligned}((\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_k]_{k\in K}} x.e')\varrho)@[\sigma_j]_{j\in J} &= (\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_k]_{k\in K}} x.(e'\varrho))@[\sigma_j]_{j\in J}\\ &= \lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K}} x.(e'\varrho)\\ &= (\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K}} x.e')\varrho \quad \text{(because } \mathsf{tv}(\varrho) \cap \textstyle\bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset\text{)}\\ &= ((\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_k]_{k\in K}} x.e')@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = e_0{\in}t\,?\,e_1 : e_2}$**:**

$$\begin{aligned}&((e_0{\in}t\,?\,e_1 : e_2)\varrho)@[\sigma_j]_{j\in J}\\ =\ &((e_0\varrho){\in}t\,?\,(e_1\varrho) : (e_2\varrho))@[\sigma_j]_{j\in J}\\ =\ &((e_0\varrho)@[\sigma_j]_{j\in J}){\in}t\,?\,((e_1\varrho)@[\sigma_j]_{j\in J}) : ((e_2\varrho)@[\sigma_j]_{j\in J})\\ =\ &((e_0@[\sigma_j]_{j\in J})\varrho){\in}t\,?\,((e_1@[\sigma_j]_{j\in J})\varrho) : ((e_2@[\sigma_j]_{j\in J})\varrho) \quad \text{(by induction)}\\ =\ &((e_0@[\sigma_j]_{j\in J}){\in}t\,?\,(e_1@[\sigma_j]_{j\in J}) : (e_2@[\sigma_j]_{j\in J}))\varrho\\ =\ &((e_0{\in}t\,?\,e_1 : e_2)@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

$\underline{e = e'[\sigma_k]_{k\in K}}$**:** using α-conversion on the polymorphic type variables of e, we can assume $\mathsf{tv}(\varrho) \cap \bigcup_{k\in K} \mathsf{dom}(\sigma_k) = \emptyset$. Consequently we have $\mathsf{tv}(\varrho) \cap \bigcup_{j\in J,k\in K} \mathsf{dom}(\sigma_j \circ \sigma_k) = \emptyset$, and we deduce

$$\begin{aligned}((e'[\sigma_k]_{k\in K})\varrho)@[\sigma_j]_{j\in J} &= ((e'\varrho)[\sigma_k]_{k\in K})@[\sigma_j]_{j\in J}\\ &= (e'\varrho)@([\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K})\\ &= (e'\varrho)@[\sigma_j\circ\sigma_k]_{j\in J,k\in K}\\ &= (e'@[\sigma_j\circ\sigma_k]_{j\in J,k\in K})\varrho \quad \text{(by induction)}\\ &= (e'@([\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K}))\varrho\\ &= ((e'[\sigma_k]_{k\in K})@[\sigma_j]_{j\in J})\varrho\end{aligned}$$

□

Lemma B.6 ([Expression] substitution lemma). *Let $e, e_1, \ldots, e_n$ be expressions, $x_1, \ldots, x_n$ distinct variables, and $t, t_1, \ldots, t_n$ types. If $\Delta \fatsemi \Gamma, (x_1 : t_1), \ldots, (x_n : t_n) \vdash e : t$ and $\Delta \fatsemi \Gamma \vdash e_i : t_i$ for all i, then $\Delta \fatsemi \Gamma \vdash e\{e_1/x_1, \ldots, e_n/x_n\} : t$.*

Proof. By induction on the typing derivations for $\Delta \fatsemi \Gamma, (x_1 : t_1), \ldots, (x_n : t_n) \vdash e : t$. We simply "plug" a copy of the derivation for $\Delta \fatsemi \Gamma \vdash e_i : t_i$ wherever the rule (*var*) is used for variable x_i. For simplicity, in what follows, we write Γ' for $\Gamma, (x_1 : t_1), \ldots, (x_n : t_n)$ and ϱ for $\{e_1/x_1, \ldots, e_n/x_n\}$. We proceed by a case analysis on the last applied rule.

(*const*)**:** straightforward.

(*var*)**:** $e = x$ and $\Delta \fatsemi \Gamma' \vdash x : \Gamma'(x)$.

If $x = x_i$, then $\Gamma'(x) = t_i$ and $x\varrho = e_i$. From the premise, we have $\Delta \fatsemi \Gamma \vdash e_i : t_i$. The result follows. Otherwise, $\Gamma'(x) = \Gamma(x)$ and $x\varrho = x$. Clearly, we have $\Delta \fatsemi \Gamma \vdash x : \Gamma(x)$. Thus the result follows as well.

(*pair*)**:** consider the following derivation:

$$\frac{\dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e_1 : t_1} \quad \dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e_1 : t_1}}{\Delta \fatsemi \Gamma' \vdash (e_1, e_2) : (t_1 \times t_2)}\ (pair)$$

By applying the induction hypothesis twice, we have $\Delta \fatsemi \Gamma \vdash e_i\varrho : t_i$. By (*pair*), we get $\Delta \fatsemi \Gamma \vdash (e_1\varrho, e_2\varrho) : (t_1 \times t_2)$, that is, $\Delta \fatsemi \Gamma \vdash (e_1, e_2)\varrho : (t_1 \times t_2)$.

(*proj*)**:** consider the following derivation:

$$\frac{\dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e' : t_1 \times t_2}}{\Delta \fatsemi \Gamma' \vdash \pi_i(e') : t_i}\ (proj)$$

By induction, we have $\Delta \fatsemi \Gamma \vdash e'\varrho : t_1 \times t_2$. Then the rule (*proj*) gives us $\Delta \fatsemi \Gamma \vdash \pi_i(e'\varrho) : t_i$, that is $\Delta \fatsemi \Gamma \vdash \pi_i(e')\varrho : t_i$.

(*abstr*)**:** consider the following derivation:

$$\frac{\begin{array}{l}\dfrac{\ldots}{\forall i \in I, j \in J.\ \Delta' \fatsemi \Gamma', (x : t_i\sigma_j) \vdash e'@[\sigma_j] : s_i\sigma_j} \\ \Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j)\end{array}}{\Delta \fatsemi \Gamma' \vdash \lambda^{\bigwedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j}\ (abstr)$$

By α-conversion, we can ensure that $\mathsf{tv}(\varrho) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$. By induction, we have $\Delta' \fatsemi \Gamma, (x : t_i\sigma_j) \vdash (e'@[\sigma_j])\varrho : s_i\sigma_j$ for all $i \in I$ and $j \in J$. Because $\mathsf{tv}(\varrho) \cap \mathsf{dom}(\sigma_j) = \emptyset$, by Lemma B.5, we get $\Delta' \fatsemi \Gamma, (x : t_i\sigma_j) \vdash (e'\varrho)@[\sigma_j] : s_i\sigma_j$. Then by applying (*abstr*), we obtain $\Delta \fatsemi \Gamma \vdash \lambda^{\bigwedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.(e'\varrho) : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j$. That is, $\Delta \fatsemi \Gamma \vdash (\lambda^{\bigwedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e)\varrho : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j$ (because $\mathsf{tv}(\varrho) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$).

(*case*)**:** consider the following derivation:

$$\frac{\dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e_0 : t'} \qquad \begin{cases} t' \not\leq \neg t \quad \Rightarrow \quad \dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e_1 : s} \\ t' \not\leq t \quad \Rightarrow \quad \dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e_2 : s} \end{cases}}{\Delta \fatsemi \Gamma' \vdash (e_0 \in t\ ?\ e_1 : e_2) : s}\ (case)$$

By induction, we have $\Delta \fatsemi \Gamma \vdash e_0\varrho : t'$ and $\Delta \fatsemi \Gamma \vdash e_i\varrho : s$ (for i such that $\Delta \fatsemi \Gamma' \vdash e_i : s$ has been type-checked in the original derivation). Then the rule (*case*) gives us $\Delta \fatsemi \Gamma \vdash (e_0\varrho \in t\ ?\ e_1\varrho : e_2\varrho) : s$ that is $\Delta \fatsemi \Gamma \vdash (e_0 \in t\ ?\ e_1 : e_2)\varrho : s$.

(*inst*)**:**

$$\frac{\dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e' : s} \qquad \sigma \sharp \Delta}{\Delta \fatsemi \Gamma' \vdash e'[\sigma] : s\sigma}\ (inst)$$

Using α-conversion on the polymorphic type variables of e, we can assume $\mathsf{tv}(\varrho) \cap \mathsf{dom}(\sigma) = \emptyset$. By induction, we have $\Delta \fatsemi \Gamma \vdash e'\varrho : s$. Since $\sigma \sharp \Delta$, by applying (*inst*) we obtain $\Delta \fatsemi \Gamma \vdash (e'\varrho)[\sigma] : s\sigma$, that is, $\Delta \fatsemi \Gamma \vdash (e'[\sigma])\varrho : s\sigma$ because $\mathsf{tv}(\varrho) \cap \mathsf{dom}(\sigma) = \emptyset$.

(*inter*)**:**

$$\frac{\forall j \in J.\ \dfrac{\ldots}{\Delta \fatsemi \Gamma' \vdash e'[\sigma_j] : t_j}}{\Delta \fatsemi \Gamma' \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j}\ (inter)$$

By induction, for all $j \in J$ we have $\Delta \fatsemi \Gamma \vdash (e'[\sigma_j])\varrho : t_j$, that is $\Delta \fatsemi \Gamma \vdash (e'\varrho)[\sigma_j] : t_j$. Then by applying (*inter*) we get $\Delta \fatsemi \Gamma \vdash (e'\varrho)[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j$, that is $\Delta \fatsemi \Gamma \vdash (e'[\sigma_j]_{j\in J})\varrho : \bigwedge_{j\in J} t_j$.

(*subsum*)**:** consider the following derivation:

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma' \vdash e' : s} \quad s \leq t}{\Delta \wr \Gamma' \vdash e' : t}\ (subsum)$$

By induction, we have $\Delta \wr \Gamma \vdash e'\varrho : s$. Then the rule (*subsum*) gives us $\Delta \wr \Gamma \vdash e'\varrho : t$.

□

Definition B.7. *Given two typing environments Γ_1, Γ_2, we define their* intersection *as*

$$(\Gamma_1 \wedge \Gamma_2)(x) = \begin{cases} \Gamma_1(x) \wedge \Gamma_2(x) & \text{if } x \in \mathsf{dom}(\Gamma_1) \cap \mathsf{dom}(\Gamma_2) \\ \text{undefined} & \text{otherwise} \end{cases}$$

We define $\Gamma_2 \leq \Gamma_1$ if $\Gamma_2(x) \leq \Gamma_1(x)$ for all $x \in \mathsf{dom}(\Gamma_1)$, and $\Gamma_1 \simeq \Gamma_2$ if $\Gamma_1 \leq \Gamma_2$ and $\Gamma_2 \leq \Gamma_1$.

Given an expression e and a set Δ of (monomorphic) type variables, we write $e \sharp \Delta$ if $\sigma_j \sharp \Delta$ for all the type substitution σ_j that occur in a subterm of e of the form $e'[\sigma_j]_{j\in J}$ (in other terms, we do not consider the substitutions that occur in the decorations of λ-abstractions).

Lemma B.8 (Weakening). *Let e be an expression, Γ, Γ' two typing environments and Δ' a set of type variables. If $\Delta \wr \Gamma \vdash e : t$, $\Gamma' \leq \Gamma$ and $e \sharp \Delta'$, then $\Delta \cup \Delta' \wr \Gamma' \vdash e : t$.*

Proof. By induction on the derivation of $\Delta \wr \Gamma \vdash e : t$. We perform a case analysis on the last applied rule.

(*const*)**:** straightforward.

(*var*)**:** $\Delta \wr \Gamma \vdash x : \Gamma(x)$. It is clear that $\Delta \cup \Delta' \wr \Gamma' \vdash x : \Gamma'(x)$ by (*var*). Since $\Gamma'(x) \leq \Gamma(x)$, by (*subsum*), we get $\Delta \cup \Delta' \wr \Gamma' \vdash x : \Gamma(x)$.

(*pair*)**:** consider the following derivation:

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : t_1} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : t_2}}{\Delta \wr \Gamma \vdash (e_1, e_2) : t_1 \times t_2}\ (pair)$$

By applying the induction hypothesis twice, we have $\Delta \cup \Delta' \wr \Gamma' \vdash e_i : t_i$. Then by (*pair*), we get $\Delta \cup \Delta' \wr \Gamma' \vdash (e_1, e_2) : t_1 \times t_2$.

(*proj*)**:** consider the following derivation:

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : t_1 \times t_2}}{\Delta \wr \Gamma \vdash \pi_i(e') : t_i}\ (proj)$$

By the induction hypothesis, we have $\Delta \cup \Delta' \wr \Gamma' \vdash e' : t_1 \times t_2$. Then by (*proj*), we get $\Delta \cup \Delta' \wr \Gamma' \vdash \pi_i(e') : t_i$.

(*abstr*)**:** consider the following derivation:

$$\dfrac{\begin{array}{l} \forall i \in I, j \in J.\ \dfrac{\cdots}{\Delta'' \wr \Gamma, (x : t_i\sigma_j) \vdash e'@[\sigma_j] : s_i\sigma_j} \\ \Delta'' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j) \end{array}}{\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j}\ (abstr)$$

By induction, we have $\Delta'' \cup \Delta' \wr \Gamma', (x : t_i\sigma_j) \vdash e'@[\sigma_j] : s_i\sigma_j$ for all $i \in I$ and $j \in J$. Then by (*abstr*), we get $\Delta \cup \Delta' \wr \Gamma' \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j$.

(*case*)**:** consider the following derivation:

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_0 : t'} \quad \left\{ \begin{array}{lcl} t' \not\leq \neg t & \Rightarrow & \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : s} \\ t' \not\leq t & \Rightarrow & \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : s} \end{array} \right.}{\Delta \wr \Gamma \vdash (e_0 \in t\,?\,e_1 : e_2) : s}\ (case)$$

By induction, we have $\Delta \cup \Delta' \wr \Gamma' \vdash e_0 : t_0$ and $\Delta \cup \Delta' \wr \Gamma' \vdash e_i : s$ (for i such that e_i has been type-checked in the original derivation). Then by (*case*), we get $\Delta \cup \Delta' \wr \Gamma' \vdash (e_0 \in t\,?\,e_1 : e_2) : s$.

(*inst*)**:** consider the following derivation:

$$\dfrac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : s} \quad \sigma \sharp \Delta}{\Delta \wr \Gamma \vdash e'[\sigma] : s\sigma}\ (inst)$$

By induction, we have $\Delta \cup \Delta' \wr \Gamma' \vdash e' : s$. Since $e \sharp \Delta'$(*i.e.*, $e'[\sigma] \sharp \Delta'$), we have $\sigma \sharp \Delta'$. Then $\sigma \sharp \Delta \cup \Delta'$. Therefore, by applying (*inst*) we get $\Delta \cup \Delta' \wr \Gamma' \vdash e'[\sigma] : s\sigma$.

(*inter*)**:** consider the following derivation:

$$\frac{\forall j \in J.\ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e'[\sigma_j] : t_j}}{\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j}\ (inter)$$

By induction, we have $\Delta \cup \Delta' \wr \Gamma' \vdash e'[\sigma_j] : t_j$ for all $j \in J$. Then the rule (*inst*) gives us $\Delta \cup \Delta' \wr \Gamma' \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j$.

(*subsum*)**:** there exists a type s such that

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e' : s} \qquad s \leq t}{\Delta \wr \Gamma \vdash e' : t}\ (subsum)$$

By induction, we have $\Delta \cup \Delta' \wr \Gamma' \vdash e' : s$. Then by applying the rule (*subsum*) we get $\Delta \cup \Delta' \wr \Gamma' \vdash e' : t$.

□

The next two lemmas are used to simplify sets of type-substitutions applied to expressions when they are redundant or they work on variables that are not in the expressions.

Lemma B.9 (Useless Substitutions). *Let e be an expression and $[\sigma_k]_{k\in K}$, $[\sigma'_k]_{k\in K}$ two sets of substitutions such that $\mathsf{dom}(\sigma'_k) \cap \mathsf{dom}(\sigma_k) = \emptyset$ and $\mathsf{dom}(\sigma'_k) \cap \mathsf{tv}(e) = \emptyset$ for all $k \in K$. Then*

$$\Delta \wr \Gamma \vdash e@[\sigma_k]_{k\in K} : t \iff \Delta \wr \Gamma \vdash e@[\sigma_k \cup \sigma'_k]_{k\in K} : t$$

Proof. Straightforward. □

Henceforth we use "$\uplus$" to denote the union of multi-sets (*e.g.*, $\{1,2\} \uplus \{1,3\} = \{1,2,1,3\}$).

Lemma B.10 (Redundant Substitutions). *Let $[\sigma_j]_{j\in J}$ and $[\sigma_j]_{j\in J'}$ be two sets of substitutions such that $J' \subseteq J$. Then*

$$\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J \uplus J'} : t \iff \Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : t$$

Proof. Similar to Lemma B.9. □

Lemma B.9 states that if a type variable α in the domain of a type substitution σ does not occur in the applied expression e, namely, $\alpha \in \mathsf{dom}(\sigma) \setminus \mathsf{tv}(e)$, then that part of the substitution is useless and can be safely eliminated. Lemma B.10 states that although our $[\sigma_j]_{j\in J}$ are formally multisets of type-substitutions, in practice they behave as sets, since repeated entries of type substitutions can be safely removed. Therefore, to simplify an expression without altering its type (and semantics), we first eliminate the useless type variables, yielding concise type substitutions, and then remove the redundant type substitutions. It explains why we do not apply relabeling when the domains of the type substitutions do not contain type variables in expressions in Definition A.12.

Moreover, Lemma B.10 also indicates that it is safe to keep only the type substitutions which are different from each other when we merge two sets of substitutions (*e.g.* Lemmas B.13 and B.14). In what follows, without explicit mention, we assume that there are no useless type variables in the domain of any type substitution and no redundant type substitutions in any set of type substitutions.

Lemma B.11 (Relabeling). *Let e be an expression, $[\sigma_j]_{j\in J}$ a set of type substitutions and Δ a set of type variables such that $\sigma_j \sharp \Delta$ for all $j \in J$. If $\Delta \wr \Gamma \vdash e : t$, then*

$$\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t\sigma_j$$

Proof. The proof proceeds by induction and case analysis on the structure of e. For each case we use an auxiliary internal induction on the typing derivation. We label **E** the main (external) induction and **I** the internal induction in what follows.

$e = c$**:** the typing derivation $\Delta \wr \Gamma \vdash e : t$ should end with either (*const*) or (*subsum*). Assume that the typing derivation ends with (*const*). Trivially, we have $\Delta \wr \Gamma \vdash c : b_c$. Since $c@[\sigma_j]_{j\in J} = c$ and $b_c \simeq \bigwedge_{j\in J} b_c\sigma_j$, by subsumption, we have $\Delta \wr \Gamma \vdash c@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} b_c\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*):

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e : s} \qquad s \leq t}{\Delta \wr \Gamma \vdash e : t}\ (subsum)$$

Then by **I**-induction, we have $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$. Since $s \leq t$, we get $\bigwedge_{j\in J} s\sigma_j \leq \bigwedge_{j\in J} t\sigma_j$. Then by applying the rule (*subsum*), we have $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t\sigma_j$.

$\underline{e = x}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*var*) or (*subsum*). Assume that the typing derivation ends with (*var*). Trivially, by (*var*), we get $\Delta \fatsemi \Gamma \vdash x : \Gamma(x)$. Moreover, we have $x@[\sigma_j]_{j\in J} = x$ and $\Gamma(x) = \bigwedge_{j\in J} \Gamma(x)\sigma_j$ (as $\mathsf{var}(\Gamma) \subseteq \Delta$). Therefore, we deduce that $\Delta \fatsemi \Gamma \vdash x@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} \Gamma(x)\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = (e_1, e_2)}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*pair*) or (*subsum*). Assume that the typing derivation ends with (*pair*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_1 : t_1} \quad \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_2 : t_2}}{\Delta \fatsemi \Gamma \vdash (e_1, e_2) : t_1 \times t_2}\ (pair)$$

By **E**-induction, we have $\Delta \fatsemi \Gamma \vdash e_i@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_i\sigma_j$. Then by (*pair*), we get $\Delta \fatsemi \Gamma \vdash (e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J}) : (\bigwedge_{j\in J} t_1\sigma_j \times \bigwedge_{j\in J} t_2\sigma_j)$, that is, $\Delta \fatsemi \Gamma \vdash (e_1, e_2)@[\sigma_j]_{j\in J} : \bigwedge_{j\in J}(t_1 \times t_2)\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = \pi_i(e')}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*proj*) or (*subsum*). Assume that the typing derivation ends with (*proj*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : t_1 \times t_2}}{\Delta \fatsemi \Gamma \vdash \pi_i(e') : t_i}\ (proj)$$

By **E**-induction, we have $\Delta \fatsemi \Gamma \vdash e'@[\sigma_j]_{j\in J} : \bigwedge_{j\in J}(t_1 \times t_2)\sigma_j$, that is, $\Delta \fatsemi \Gamma \vdash e'@[\sigma_j]_{j\in J} : (\bigwedge_{j\in J} t_1\sigma_j \times \bigwedge_{j\in J} t_2\sigma_j)$. Then the rule (*proj*) gives us that $\Delta \fatsemi \Gamma \vdash \pi_i(e'@[\sigma_j]_{j\in J}) : \bigwedge_{j\in J} t_i\sigma_j$, that is, $\Delta \fatsemi \Gamma \vdash \pi_i(e')@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_i\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e_1e_2}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*appl*) or (*subsum*). Assume that the typing derivation ends with (*appl*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_1 : t \to s} \quad \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_2 : t}}{\Delta \fatsemi \Gamma \vdash e_1e_2 : s}\ (pair)$$

By **E**-induction, we have $\Delta \fatsemi \Gamma \vdash e_1@[\sigma_j]_{j\in J} : \bigwedge_{j\in J}(t \to s)\sigma_j$ and $\Delta \fatsemi \Gamma \vdash e_2@[\sigma_j] : \bigwedge_{j\in J} t\sigma_j$. Since $\bigwedge_{j\in J}(t \to s)\sigma_j \leq (\bigwedge_{j\in J} t\sigma_j) \to (\bigwedge_{j\in J} s\sigma_j)$, by (*subsum*), we have $\Delta \fatsemi \Gamma \vdash e_1@[\sigma_j]_{j\in J} : (\bigwedge_{j\in J} t\sigma_j) \to (\bigwedge_{j\in J} s\sigma_j)$. Then by (*appl*), we get

$$\Delta \fatsemi \Gamma \vdash (e_1@[\sigma_j]_{j\in J})(e_2@[\sigma_j]_{j\in J}) : \bigwedge_{j\in J} s\sigma_j$$

that is, $\Delta \fatsemi \Gamma \vdash (e_1e_2)@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e'}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*abstr*) or (*subsum*). Assume that the typing derivation ends with (*abstr*):

$$\frac{\begin{array}{l}\forall i \in I, k \in K.\ \dfrac{\cdots}{\Delta' \fatsemi \Gamma, (x : t_i\sigma_k) \vdash e'@[\sigma_k] : s_i\sigma_k} \\ \Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I,k\in K} t_i\sigma_k \to s_i\sigma_k)\end{array}}{\Delta \fatsemi \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e' : \bigwedge_{i\in I,k\in K} t_i\sigma_k \to s_i\sigma_k}\ (abstr)$$

Using α-conversion, we can assume that $\sigma_j \sharp (\mathsf{var}(\bigwedge_{i\in I,k\in K} t_i\sigma_k \to s_i\sigma_k) \setminus \Delta)$ for $j \in J$. Hence $\sigma_j \sharp \Delta'$. By **E**-induction, we have

$$\Delta' \fatsemi \Gamma, (x : (t_i\sigma_k)) \vdash (e'@[\sigma_k])@[\sigma_j] : (s_i\sigma_k)\sigma_j$$

for all $i \in I$, $k \in K$ and $j \in J$. By Lemma B.4, $(e'@[\sigma_k])@[\sigma_j] = e'@([\sigma_j] \circ [\sigma_k])$. So

$$\Delta' \fatsemi \Gamma, (x : (t_i\sigma_k)) \vdash e'@([\sigma_j] \circ [\sigma_k]) : (s_i\sigma_k)\sigma_j$$

Finally, by (*abstr*), we get

$$\Delta \fatsemi \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j \circ \sigma_k]_{j\in J,k\in K}} x.e' : \bigwedge_{i\in I,j\in J,k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k)$$

that is,

$$\Delta \fatsemi \Gamma \vdash (\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e')@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} (\bigwedge_{i\in I,k\in K} t_i\sigma_k \to s_i\sigma_k)\sigma_j$$

Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e'\in t\,?\,e_1 : e_2}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*case*) or (*subsum*). Assume that the typing derivation ends with (*case*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : t'} \quad \begin{cases} t' \not\leq \neg t & \Rightarrow \quad \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_1 : s} \\ t' \not\leq t & \Rightarrow \quad \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e_2 : s} \end{cases}}{\Delta \fatsemi \Gamma \vdash (e'\in t\,?\,e_1 : e_2) : s}\ (case)$$

By **E**-induction, we have $\Delta \fatsemi \Gamma \vdash e'@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t'\sigma_j$. Suppose $\bigwedge_{j\in J} t'\sigma_j \not\leq \neg t$; then we must have $t' \not\leq \neg t$, and the branch for e_1 has been type-checked. By the **E**-induction hypothesis, we have $\Delta \fatsemi \Gamma \vdash e_1@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$. Similarly, if $\bigwedge_{j\in J} t'\sigma_j \not\leq t$, then the second branch e_2 has been type-checked, and we have $\Delta \fatsemi \Gamma \vdash e_2@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$ by the **E**-induction hypothesis. By (*case*), we have

$$\Delta \fatsemi \Gamma \vdash (e'@[\sigma_j]_{j\in J}\in t\,?\,e_1@[\sigma_j]_{j\in J} : e_2@[\sigma_j]_{j\in J}) : \bigwedge_{j\in J} s\sigma_j$$

that is $\Delta \fatsemi \Gamma \vdash (e'\in t\,?\,e_1 : e_2)@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e'[\sigma]}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*inst*) or (*subsum*). Assume that the typing derivation ends with (*inst*):

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : t} \qquad \sigma \sharp \Delta}{\Delta \fatsemi \Gamma \vdash e'[\sigma] : t\sigma}\ (inst)$$

Consider the set of substitutions $[\sigma_j \circ \sigma]_{j\in J}$. It is clear that $\sigma_j \circ \sigma \sharp \Delta$ for all $j \in J$. By **E**-induction, we have

$$\Delta \fatsemi \Gamma \vdash e'@[\sigma_j \circ \sigma]_{j\in J} : \bigwedge_{j\in J} t(\sigma_j \circ \sigma)$$

that is, $\Delta \fatsemi \Gamma \vdash (e'[\sigma])@[\sigma_j]_{j\in J} : \bigwedge_{j\in J}(t\sigma)\sigma_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e'[\sigma_k]_{k\in K}}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*inter*) or (*subsum*). Assume that the typing derivation ends with (*inter*):

$$\frac{\forall k \in K.\ \dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e'[\sigma_k] : t_k}}{\Delta \fatsemi \Gamma \vdash e'[\sigma_k]_{k\in K} : \bigwedge_{k\in K} t_k}\ (inter)$$

As an intermediary result, we first prove that the derivation can be rewritten as

$$\dfrac{\dfrac{\forall k \in K.\quad \dfrac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : s} \qquad \sigma_k \sharp \Delta}{\Delta \fatsemi \Gamma \vdash e'[\sigma_k] : s\sigma_k}\ (inst)}{\Delta \fatsemi \Gamma \vdash e'[\sigma_k]_{k\in K} : \bigwedge_{k\in K} s\sigma_k}\ (inter) \qquad \bigwedge_{k\in K} s\sigma_k \leq \bigwedge_{k\in K} t_k}{\Delta \fatsemi \Gamma \vdash e'[\sigma_k]_{k\in K} : \bigwedge_{k\in K} t_k}\ (subsum)$$

We proceed by induction on the original derivation. It is clear that each sub-derivation $\Delta \fatsemi \Gamma \vdash e'[\sigma_k] : t_k$ ends with either (*inst*) or (*subsum*). If all the sub-derivations end with an instance of (*inst*), then for all $k \in K$, we have

$$\frac{\dfrac{\cdots}{\Delta \fatsemi \Gamma \vdash e' : s_k} \qquad \sigma_k \sharp \Delta}{\Delta \fatsemi \Gamma \vdash e'[\sigma_k] : s_k\sigma_k}\ (inst)$$

By Lemma B.2, we have $\Delta \fatsemi \Gamma \vdash e' : \bigwedge_{k\in K} s_k$. Let $s = \bigwedge_{k\in K} s_k$. Then by (*inst*), we get $\Delta \fatsemi \Gamma \vdash e'[\sigma_k] : s\sigma_k$. Finally, by (*inter*) and (*subsum*), the intermediary result holds. Otherwise, at least one of the sub-derivations ends with an instance of (*subsum*); the intermediary result also holds by induction.
Now that the intermediary result is proved, we go back to the proof of the lemma. Consider the set of substitutions $[\sigma_j \circ \sigma_k]_{j\in J,k\in K}$. It is clear that $\sigma_j \circ \sigma_k \sharp \Delta$ for all $j \in J, k \in K$. By **E**-induction on e' (*i.e.*, $\Delta \fatsemi \Gamma \vdash e' : s$), we have

$$\Delta \fatsemi \Gamma \vdash e'@[\sigma_j \circ \sigma_k]_{j\in J,k\in K} : \bigwedge_{j\in J,k\in K} s(\sigma_j \circ \sigma_k)$$

that is, $\Delta \fatsemi \Gamma \vdash (e'[\sigma_k]_{k\in K})@[\sigma_j]_{j\in J} : \bigwedge_{j\in J}(\bigwedge_{k\in K} s\sigma_k)\sigma_j$. As $\bigwedge_{k\in K} s\sigma_k \leq \bigwedge_{k\in K} t_k$, we get $\bigwedge_{j\in J}(\bigwedge_{k\in K} s\sigma_k)\sigma_j \leq \bigwedge_{j\in J}(\bigwedge_{k\in K} t_k)\sigma_j$. Then by (*subsum*), the result follows.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

□

Corollary B.12. *If* $\Delta \wr \Gamma \vdash e[\sigma_j]_{j\in J} : t$*, then* $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : t$*.*

Proof. Immediate consequence of Lemma B.11. □

Lemma B.13. *If* $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : t$ *and* $\Delta' \wr \Gamma' \vdash e@[\sigma_j]_{j\in J'} : t'$*, then* $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e@[\sigma_j]_{j\in J\cup J'} : t \wedge t'$

Proof. The proof proceeds by induction and case analysis on the structure of e. For each case we use an auxiliary internal induction on both typing derivations. We label **E** the main (external) induction and **I** the internal induction in what follows.

$\underline{e = c}$**:** $e@[\sigma_j]_{j\in J} = e@[\sigma_j]_{j\in J'} = c$. Clearly, both typing derivations should end with either (*const*) or (*subsum*). Assume that both derivations end with (*const*):

$$\frac{}{\Delta \wr \Gamma \vdash c : b_c}\ (const) \qquad \frac{}{\Delta' \wr \Gamma' \vdash c : b_c}\ (const)$$

Trivially, by (*const*) we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash c : b_c$, that is $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e@[\sigma_j]_{j\in J\cup J'} : b_c$. As $b_c \simeq b_c \wedge b_c$, by (*subsum*), the result follows.
Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), for instance,

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : s} \qquad s \leq t}{\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : t}\ (subsum)$$

Then by **I**-induction on $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J} : s$ and $\Delta' \wr \Gamma' \vdash e@[\sigma_j]_{j\in J'} : t'$, we have

$$\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e@[\sigma_j]_{j\in J\cup J'} : s \wedge t'$$

Since $s \leq t$, we have $s \wedge t' \leq t \wedge t'$. By (*subsum*), the result follows as well.

$\underline{e = x}$**:** $e@[\sigma_j]_{j\in J} = e@[\sigma_j]_{j\in J'} = x$. Clearly, both typing derivations should end with either (*var*) or (*subsum*). Assume that both derivations end with an instance of (*var*):

$$\frac{}{\Delta \wr \Gamma \vdash x : \Gamma(x)}\ (var) \qquad \frac{}{\Delta' \wr \Gamma' \vdash x : \Gamma'(x)}\ (var)$$

Since $x \in \mathsf{dom}(\Gamma)$ and $x \in \mathsf{dom}(\Gamma')$, $x \in \mathsf{dom}(\Gamma \wedge \Gamma')$. By (*var*), we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash x : (\Gamma \wedge \Gamma')(x)$, that is, $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e@[\sigma_j]_{j\in J\cup J'} : \Gamma(x) \wedge \Gamma'(x)$.
Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = (e_1, e_2)}$**:** $e@[\sigma_j]_{j\in J} = (e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J})$ and $e@[\sigma_j]_{j\in J'} = (e_1@[\sigma_j]_{j\in J'}, e_2@[\sigma_j]_{j\in J'})$. Clearly, both typing derivations should end with either (*pair*) or (*subsum*). Assume that both derivations end with an instance of (*pair*):

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1@[\sigma_j]_{j\in J} : s_1} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2@[\sigma_j]_{j\in J} : s_2}}{\Delta \wr \Gamma \vdash (e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J}) : (s_1 \times s_2)}\ (pair)$$

$$\frac{\dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_1@[\sigma_j]_{j\in J'} : s'_1} \quad \dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_2@[\sigma_j]_{j\in J'} : s'_2}}{\Delta' \wr \Gamma' \vdash (e_1@[\sigma_j]_{j\in J'}, e_2@[\sigma_j]_{j\in J'}) : (s'_1 \times s'_2)}\ (pair)$$

By **E**-induction, we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_i@[\sigma_j]_{j\in J\cup J'} : s_i \wedge s'_i$. Then by (*pair*), we get $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_1@[\sigma_j]_{j\in J\cup J'}, e_2@[\sigma_j]_{j\in J\cup J'}) : (s_1 \wedge s'_1) \times (s_2 \wedge s'_2)$, that is $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_1, e_2)@[\sigma_j]_{j\in J\cup J'} : (s_1 \wedge s'_1) \times (s_2 \wedge s'_2)$. Moreover, because intersection distributes over product, we have $(s_1 \wedge s'_1) \times (s_2 \wedge s'_2) \simeq (s_1 \times s_2) \wedge (s'_1 \times s'_2)$. Finally, by applying (*subsum*), we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_1, e_2)@[\sigma_j]_{j\in J\cup J'} : (s_1 \times s_2) \wedge (s'_1 \times s'_2)$.
Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = \pi_i(e')}$**:** $e@[\sigma_j]_{j\in J} = \pi_i(e'@[\sigma_j]_{j\in J})$ and $e@[\sigma_j]_{j\in J'} = \pi_i(e'@[\sigma_j]_{j\in J'})$, where $i = 1, 2$. Clearly, both typing derivations should end with either (*proj*) or (*subsum*). Assume that both derivations end with an instance of (*proj*):

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e'@[\sigma_j]_{j\in J} : s_1 \times s_2}}{\Delta \wr \Gamma \vdash \pi_i(e'@[\sigma_j]_{j\in J}) : s_i}\ (proj) \qquad \frac{\dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e'@[\sigma_j]_{j\in J'} : s'_1 \times s'_2}}{\Delta' \wr \Gamma' \vdash \pi_i(e'@[\sigma_j]_{j\in J'}) : s'_i}\ (proj)$$

By **E**-induction, we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e'@[\sigma_j]_{j\in J\cup J'} : (s_1 \times s_2) \wedge (s'_1 \times s'_2)$. Since $(s_1 \times s_2) \wedge (s'_1 \times s'_2) \simeq (s_1 \wedge s'_1) \times (s_2 \wedge s'_2)$ (See the case of $e = (e_1, e_2)$), by (*subsum*), we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e'@[\sigma_j]_{j\in J\cup J'} : (s_1 \wedge s'_1) \times (s_2 \wedge s'_2)$. Finally, by applying (*proj*), we get $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash \pi_i(e@[\sigma_j]_{j\in J\cup J'}) : s_i \wedge s'_i$, that is $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash \pi_i(e)@[\sigma_j]_{j\in J\cup J'} : s_i \wedge s'_i$.
Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e_1 e_2}$**:** $e@[\sigma_j]_{j\in J} = (e_1@[\sigma_j]_{j\in J})(e_2@[\sigma_j]_{j\in J})$ and $e@[\sigma_j]_{j\in J'} = (e_1@[\sigma_j]_{j\in J'})(e_2@[\sigma_j]_{j\in J'})$. Clearly, both typing derivations should end with either (*appl*) or (*subsum*). Assume that both derivations end with an instance of (*appl*):

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1@[\sigma_j]_{j\in J} : s_1 \to s_2} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2@[\sigma_j]_{j\in J} : s_1}}{\Delta \wr \Gamma \vdash (e_1@[\sigma_j]_{j\in J})\,(e_2@[\sigma_j]_{j\in J}) : s_2}\ (appl)$$

$$\frac{\dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_1@[\sigma_j]_{j\in J'} : s'_1 \to s'_2} \quad \dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_2@[\sigma_j]_{j\in J'} : s'_1}}{\Delta' \wr \Gamma' \vdash (e_1@[\sigma_j]_{j\in J'})\,(e_2@[\sigma_j]_{j\in J'}) : s'_2}\ (appl)$$

By **E**-induction, we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_1@[\sigma_j]_{j\in J\cup J'} : (s_1 \to s_2) \wedge (s'_1 \to s'_2)$ and $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_2@[\sigma_j]_{j\in J\cup J'} : s_1 \wedge s'_1$. Because intersection distributes over arrows, we have $(s_1 \to s_2) \wedge (s'_1 \to s'_2) \leq (s_1 \wedge s'_1) \to (s_2 \wedge s'_2)$. Then by the rule (*subsum*), we get $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_1@[\sigma_j]_{j\in J\cup J'} : (s_1 \wedge s'_1) \to (s_2 \wedge s'_2)$. Finally by (*appl*), we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_1@[\sigma_j]_{j\in J\cup J'})(e_2@[\sigma_j]_{j\in J\cup J'}) : s_2 \wedge s'_2$, that is, $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_1 e_2)@[\sigma_j]_{j\in J\cup J'} : s_2 \wedge s'_2$. Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e'}$**:** $e@[\sigma_j]_{j\in J} = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K}} x.e'$ and $e@[\sigma_j]_{j\in J'} = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J'}\circ[\sigma_k]_{k\in K}} x.e'$. Clearly, both typing derivations should end with either (*abstr*) or (*subsum*). Assume that both derivations end with an instance of (*abstr*):

$$\frac{\begin{array}{l}\forall i \in I, j \in J, k \in K.\ \dfrac{\cdots}{\Delta_1 \wr \Gamma, (x : t_i(\sigma_j \circ \sigma_k)) \vdash e'@[\sigma_j \circ \sigma_k] : s_i(\sigma_j \circ \sigma_k)} \\ \Delta_1 = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J, k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k))\end{array}}{\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}\circ[\sigma_k]_{k\in K}} x.e' : \bigwedge_{i\in I, j\in J, k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k)}$$

$$\frac{\begin{array}{l}\forall i \in I, j \in J', k \in K.\ \dfrac{\cdots}{\Delta_2 \wr \Gamma', (x : t_i(\sigma_j \circ \sigma_k)) \vdash e'@[\sigma_j \circ \sigma_k] : s_i(\sigma_j \circ \sigma_k)} \\ \Delta_2 = \Delta' \cup \mathsf{var}(\bigwedge_{i\in I, j\in J', k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k))\end{array}}{\Delta' \wr \Gamma' \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J'}\circ[\sigma_k]_{k\in K}} x.e' : \bigwedge_{i\in I, j\in J', k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k)}$$

Consider any expression $e'@([\sigma_j]\circ[\sigma_k])$ and any $e_0[\sigma_{j_0}]_{j_0\in J_0}$ in $e'@([\sigma_j]\circ[\sigma_k])$, where $j \in J \cup J', k \in K$. (Then $e_0[\sigma_{j_0}]_{j_0\in J_0}$ must be from e'). All type variables in $\bigcup_{j_0\in J_0} \mathsf{dom}(\sigma_{j_0})$ must be polymorphic, otherwise, $e'@([\sigma_j] \circ [\sigma_k])$ is not well-typed under Δ_1 or Δ_2. Using α-conversion, we can assume that these polymorphic type variables are different from $\Delta_1 \cup \Delta_2$, that is $(\bigcup_{j_0\in J_0} \mathsf{dom}(\sigma_{j_0})) \cap (\Delta_1 \cup \Delta_2) = \emptyset$. So we have $e'@([\sigma_j] \circ [\sigma_k]) \sharp \Delta_1 \cup \Delta_2$. According to Lemma B.8, we have

$$\Delta_1 \cup \Delta_2 \wr \Gamma \wedge \Gamma', (x : t_i(\sigma_j \circ \sigma_k)) \vdash e'@[\sigma_j \circ \sigma_k] : s_i(\sigma_j \circ \sigma_k)$$

for all $i \in I$, $j \in J \cup J'$ and $k \in K$. It is clear that

$$\Delta_1 \cup \Delta_2 = \Delta \cup \Delta' \cup \mathsf{var}(\bigwedge_{i\in I, j\in J\cup J', k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k))$$

By (*abstr*), we have

$$\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J\cup J'}\circ[\sigma_k]_{k\in K}} x.e' : \bigwedge_{i\in I, j\in J\cup J', k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k)$$

that is, $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e@[\sigma_j]_{j\in J\cup J'} : t \wedge t'$, where $t = \bigwedge_{i\in I, j\in J, k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k)$ and $t' = \bigwedge_{i\in I, j\in J', k\in K} t_i(\sigma_j \circ \sigma_k) \to s_i(\sigma_j \circ \sigma_k)$.

Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = (e_0 \in t\,?\,e_1 : e_2)}$**:** $e@[\sigma_j]_{j\in J} = (e_0@[\sigma_j]_{j\in J} \in t\,?\,e_1@[\sigma_j]_{j\in J} : e_2@[\sigma_j]_{j\in J})$ and $e@[\sigma_j]_{j\in J'} = (e_0@[\sigma_j]_{j\in J'} \in t\,?\,e_1@[\sigma_j]_{j\in J'} : e_2@[\sigma_j]_{j\in J'})$. Clearly, both typing derivations should end with either (*case*) or (*subsum*). Assume that both derivations end with an instance of (*case*):

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_0@[\sigma_j]_{j\in J} : t_0} \quad \begin{cases} t_0 \not\leq \neg t & \Rightarrow \ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1@[\sigma_j]_{j\in J} : s} \\ t_0 \not\leq t & \Rightarrow \ \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2@[\sigma_j]_{j\in J} : s} \end{cases}}{\Delta \wr \Gamma \vdash (e_0@[\sigma_j]_{j\in J} \in t\,?\,e_1@[\sigma_j]_{j\in J} : e_2@[\sigma_j]_{j\in J}) : s}\ (case)$$

$$\frac{\dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_0@[\sigma_j]_{j\in J'} : t'_0} \quad \begin{cases} t'_0 \not\leq \neg t & \Rightarrow \ \dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_1@[\sigma_j]_{j\in J'} : s'} \\ t'_0 \not\leq t & \Rightarrow \ \dfrac{\cdots}{\Delta' \wr \Gamma' \vdash e_2@[\sigma_j]_{j\in J'} : s'} \end{cases}}{\Delta' \wr \Gamma' \vdash (e_0@[\sigma_j]_{j\in J'} \in t\,?\,e_1@[\sigma_j]_{j\in J'} : e_2@[\sigma_j]_{j\in J'}) : s'}\ (case)$$

By **E**-induction, we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_0@[\sigma_j]_{j\in J\cup J'} : t_0 \wedge t_0'$. Suppose $t_0 \wedge t_0' \not\leq \neg t$, then we must have $t_0 \not\leq \neg t$ and $t_0' \not\leq \neg t$, and the first branch has been checked in both derivations. Therefore we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_1@[\sigma_j]_{j\in J\cup J'} : s \wedge s'$ by the induction hypothesis. Similarly, if $t_0 \wedge t_0' \not\leq t$, we have $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e_2@[\sigma_j]_{j\in J\cup J'} : s \wedge s'$. By applying the rule (*case*), we have

$$\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_0@[\sigma_j]_{j\in J\cup J'} \in t \,?\, e_1@[\sigma_j]_{j\in J\cup J'} : e_2@[\sigma_j]_{j\in J\cup J'}) : s \wedge s'$$

that is, $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e_0 \in t \,?\, e_1 : e_2)@[\sigma_j]_{j\in J\cup J'} : s \wedge s'$.

Otherwise, there exists at least one typing derivation which ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$e = e'[\sigma_i]_{i\in I}$**:** $e@[\sigma_j]_{j\in J} = e'@([\sigma_j \circ \sigma_i]_{(j,i)\in(J\times I)})$ and $e@[\sigma_j]_{j\in J'} = e'@([\sigma_j \circ \sigma_i]_{(j,i)\in(J'\times I)})$. By **E**-induction on e', we have

$$\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash e'@[\sigma_j \circ \sigma_i]_{(j,i)\in(J\times I)\cup(J'\times I)} : t \wedge t'$$

that is, $\Delta \cup \Delta' \wr \Gamma \wedge \Gamma' \vdash (e'[\sigma_i]_{i\in I})@[\sigma_j]_{j\in J\cup J'} : t \wedge t'$.

□

Corollary B.14. *If* $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J_1} : t_1$ *and* $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J_2} : t_2$*, then* $\Delta \wr \Gamma \vdash e@[\sigma_j]_{j\in J_1\cup J_2} : t_1 \wedge t_2$

Proof. Immediate consequence of Lemmas B.13 and B.8. □

B.2 Type soundness

In this section, we prove the soundness of the type system: well-typed expressions do not "go wrong". We proceed in two steps, commonly known as the *subject reduction* and *progress* theorems:

- Subject reduction: a well-typed expression keeps being well-typed during reduction.
- Progress: a well-typed expression can not be "stuck" (*i.e.*, a well-typed expression which is not value can be reduced).

Theorem B.15 (Subject reduction). *Let e be an expression and t a type. If $\Delta \wr \Gamma \vdash e : t$ and $e \rightsquigarrow e'$, then $\Delta \wr \Gamma \vdash e' : t$.*

Proof. By induction on the derivation of $\Delta \wr \Gamma \vdash e : t$. We proceed by a case analysis on the last rule used in the derivation of $\Delta \wr \Gamma \vdash e : t$.

(*const*)**:** the expression e is a constant. It cannot be reduced. Thus the result follows.

(*var*)**:** similar to the (*const*) case.

(*pair*)**:** $e = (e_1, e_2)$, $t = t_1 \times t_2$. We have $\Delta \wr \Gamma \vdash e_i : t_i$ for $i = 1..2$. There are two ways to reduce e, that is

(1) $(e_1, e_2) \rightsquigarrow (e_1', e_2)$: by induction, we have $\Delta \wr \Gamma \vdash e_1' : t_1$. Then the rule (*pair*) gives us $\Delta \wr \Gamma \vdash (e_1', e_2) : t_1 \times t_2$.

(2) The case $(e_1, e_2) \rightsquigarrow (e_1, e_2')$ is treated similarly.

(*proj*)**:** $e = \pi_i(e_0)$, $t = t_i$, $\Delta \wr \Gamma \vdash e_0 : t_1 \times t_2$.

(1) $e_0 \rightsquigarrow e_0'$: $e' = \pi_i(e_0')$. By induction, we have $\Delta \wr \Gamma \vdash e_0' : t_1 \times t_2$. Then the rule (*proj*) gives us $\Delta \wr \Gamma \vdash e' : t_i$.

(2) $e_0 = (v_1, v_2)$: $e' = v_i$. By Lemma B.3, we get $\Delta \wr \Gamma \vdash e' : t_i$.

(*appl*)**:** $e = e_1\, e_2$, $\Delta \wr \Gamma \vdash e_1 : t \to s$ and $\Delta \wr \Gamma \vdash e_2 : t$.

(1) $e_1\, e_2 \rightsquigarrow e_1'\, e_2$ or $e_1\, e_2 \rightsquigarrow e_1\, e_2'$: similar to the case of (*pair*).

(2) $e_1 = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$, $e_2 = v_2$, $e' = (e_0@[\sigma_j]_{j\in P})\{v_2/x\}$ and $P = \{j \in J \mid \exists i \in I.\ \Delta \wr \Gamma \vdash v_2 : t_i\sigma_j\}$: by Lemma B.3, we have $\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j \leq t \to s$. From the subtyping for arrow types we deduce that $t \leq \bigvee_{i\in I, j\in J} t_i\sigma_j$ and that for any non-empty set $P \subseteq I \times J$ if $t \not\leq \bigvee_{(i,j)\in I\times J\setminus P} t_i\sigma_j$, then $\bigwedge_{(i,j)\in P} s_i\sigma_j \leq s$. Let $P_0 = \{(i,j) \mid \Delta \wr \Gamma \vdash v_2 : t_i\sigma_j\}$. Since $\Delta \wr \Gamma \vdash v_2 : t$ and $t \leq \bigvee_{i\in I, j\in J} t_i\sigma_j$, P_0 is non-empty. Also notice that $t \not\leq \bigvee_{(i,j)\in I\times J\setminus P_0} t_i\sigma_j$, since otherwise there would exist some $(i,j) \notin P_0$ such that $\Delta \wr \Gamma \vdash v_2 : t_i\sigma_j$. As a consequence, we get $\bigwedge_{(i,j)\in P_0} s_i\sigma_j \leq s$. Moreover, since e_1 is well-typed under Δ and Γ, there exists an instance of the rule (*abstr*) which infers a type $\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j$ for e_1 under Δ and Γ and whose premise is $\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e_0@[\sigma_j] : s_i\sigma_j$ for all $i \in I$ and $j \in J$, where $\Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j)$. By Lemma B.13, we get $\Delta' \wr \bigwedge_{(i,j)\in P_0}(\Gamma, (x : t_i\sigma_j)) \vdash e_0@[\sigma_j]_{j\in P_0} : \bigwedge_{(i,j)\in P_0} s_i\sigma_j$. Since $\Gamma, (x : \bigwedge_{(i,j)\in P_0} t_i\sigma_j) \simeq \bigwedge_{(i,j)\in P_0}(\Gamma, (x : t_i\sigma_j))$, then from Lemma B.8 we have $\Delta' \wr \Gamma, (x : \bigwedge_{(i,j)\in P_0} t_i\sigma_j) \vdash e_0@[\sigma_j]_{j\in P_0} : \bigwedge_{(i,j)\in P_0} s_i\sigma_j$ and a fortiori $\Delta \wr \Gamma, (x : \bigwedge_{(i,j)\in P_0} t_i\sigma_j) \vdash e_0@[\sigma_j]_{j\in P_0} : \bigwedge_{(i,j)\in P_0} s_i\sigma_j$. Furthermore, by definition of P_0 and the admissibility of the intersection introduction (Lemma B.2) we have that $\Delta \wr \Gamma \vdash v_2 : \bigwedge_{(i,j)\in P_0} t_i\sigma_j$. Thus by Lemma B.6, we get $\Delta \wr \Gamma \vdash e' : \bigwedge_{(i,j)\in P_0} s_i\sigma_j$. Finally, by (*subsum*), we obtain $\Delta \wr \Gamma \vdash e' : s$ as expected.

(*abstr*): $e = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$. It cannot be reduced. Thus the result follows.
(*case*): $e = e_0 \in s \,?\, e_1 : e_2$.
(1) $e_0 \rightsquigarrow e_0'$ or $e_1 \rightsquigarrow e_1'$ or $e_2 \rightsquigarrow e_2'$: similar to the case of (*pair*).
(2) $e_0 = v_0$ and $\vdash e_0 : s$: we have $e' = e_1$. The typing rule gives us $\Delta \fatsemi \Gamma \vdash e_1 : t$, thus the result follows.
(3) otherwise ($e_0 = v_0$): we have $e' = e_2$. Similar to the above case.
(*inst*): $e = e_1[\sigma]$, $\Delta \fatsemi \Gamma \vdash e_1 : s$, $\sigma \sharp \Delta$ and $e \rightsquigarrow e_1@[\sigma]$. By applying Lemma B.11, we get $\Delta \fatsemi \Gamma \vdash e_1@[\sigma] : s\sigma$.
(*inter*): $e = e_1[\sigma_j]_{j\in J}$, $\Delta \fatsemi \Gamma \vdash e_1[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j$ and $e \rightsquigarrow e_1@[\sigma_j]_{j\in J}$. By applying Corollary B.12, we get $\Delta \fatsemi \Gamma \vdash e_1@[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j$.
(*subsum*): there exists a type s such that $\Delta \fatsemi \Gamma \vdash e : s \leq t$ and $e \rightsquigarrow e'$. By induction, we have $\Delta \fatsemi \Gamma \vdash e' : s$, then by subsumption we get $\Delta \fatsemi \Gamma \vdash e' : t$.

□

Theorem B.16 (Progress). *Let e be a well-typed closed expression, that is, $\vdash e : t$ for some t. If e is not a value, then there exists an expression e' such that $e \rightsquigarrow e'$.*

Proof. By induction on the derivation of $\vdash e : t$. We proceed by a case analysis of the last rule used in the derivation of $\vdash e : t$.

(*const*): immediate since a constant is a value.
(*var*): impossible since a variable cannot be well-typed in an empty environment.
(*pair*): $e = (e_1, e_2)$, $t = t_1 \times t_2$, and $\vdash e_i : t_i$ for $i = 1..2$. If one of the e_i can be reduced, then e can also be reduced. Otherwise, by induction, both e_1 and e_2 are values, and so is e.
(*proj*): $e = \pi_i(e_0)$, $t = t_i$, and $\vdash e_0 : t_1 \times t_2$. If e_0 can be reduced to e_0', then $e \rightsquigarrow \pi_i(e_0')$. Otherwise, e_0 is a value. By Lemma B.3, we get $e_0 = (v_1, v_2)$, and thus $e \rightsquigarrow v_i$.
(*appl*): $e = e_1\ e_2$, $\vdash e_1 : t \to s$ and $\vdash e_2 : t$. If one of the e_i can be reduced, then e can also be reduced. Otherwise, by induction, both e_1 and e_2 are values. By Lemma B.3, we get $e_1 = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$ such that $\bigwedge_{i\in I, j\in J} t_i\sigma_j \to s_i\sigma_j \leq t \to s$. By the definition of subtyping for arrow types, we have $t \leq \bigvee_{i\in I, j\in J} t_i\sigma_j$. Moreover, as $\vdash e_2 : t$, the set $P = \{j \in J \mid \exists i \in I.\ \vdash e_2 : t_i\sigma_j\}$ is non-empty. Then $e \rightsquigarrow (e_0@[\sigma_j]_{j\in P})\{e_2/x\}$.
(*abstr*): the expression e is an abstraction which is well-typed under the empty environment. It is thus a value.
(*case*): $e = e_0 \in s \,?\, e_1 : e_2$. If e_0 can be reduced, then e can also be reduced. Otherwise, by induction, e_0 is a value v. If $\vdash v : s$, then we have $e \rightsquigarrow e_1$. Otherwise, $e \rightsquigarrow e_2$.
(*inst*): $e = e_1[\sigma]$, $t = s\sigma$ and $\vdash e_1 : s$. Then $e \rightsquigarrow e_1@[\sigma]$.
(*inter*): $e = e_1[\sigma_j]_{j\in J}$, $t = \bigwedge_{j\in J} t_j$ and $\vdash e_1[\sigma_j] : t_j$ for all $j \in J$. It is clear that $e \rightsquigarrow e_1@[\sigma_j]_{j\in J}$.
(*subsum*): straightforward application of the induction hypothesis.

□

We now conclude that the type system is type sound.

Corollary B.17 (Type soundness). *Let e be a well-typed closed expression, that is, $\vdash e : t$ for some t. Then either e diverges or it returns a value of type t.*

Proof. Consequence of Theorems B.16 and B.15. □

B.3 Expressing intersection types

We now prove that the calculus with explicit substitutions is able to derive the same typings as the Barendregt, Coppo, Dezani (BCD) intersection type system [1] without the universal type ω. We remind the BCD types (a strict subset of $\mathscr{T}$), the BCD typing rules (without ω) and subtyping relation in Figure 5, where we use m to range over pure λ-calculus expressions. To make the correspondence between the systems easier, we adopt a n-ary version of the intersection typing rule. Henceforth, we use D to range over BCD typing derivations. We first remark that the BCD subtyping relation is included in the one of this work.

Lemma B.18. *If $t_1 \leq_{BCD} t_2$ then $t_1 \leq t_2$.*

Proof. All the BCD subtyping rules are admissible in [11] and, a fortiori, in our type system. □

Types:

$$t ::= \alpha \mid t \to t \mid t \wedge t$$

Typing rules:

$$\frac{}{\Gamma \vdash_{BCD} x : \Gamma(x)}\ (\textit{BCD var}) \qquad \frac{\Gamma \vdash_{BCD} m_1 : t_1 \to t_2 \qquad \Gamma \vdash m_2 : t_1}{\Gamma \vdash_{BCD} m_1\ m_2 : t_2}\ (\textit{BCD app})$$

$$\frac{\Gamma, x : t_1 \vdash_{BCD} m : t_2}{\Gamma \vdash_{BCD} \lambda x.m : t_1 \to t_2}\ (\textit{BCD abstr}) \qquad \frac{\Gamma \vdash_{BCD} m : t_i \quad i \in I}{\Gamma \vdash_{BCD} m : \bigwedge_{i \in I} t_i \quad |I| > 1}\ (\textit{BCD inter})$$

$$\frac{\Gamma \vdash_{BCD} m : t_1 \qquad t_1 \leq_{BCD} t_2}{\Gamma \vdash m : t_2}\ (\textit{BCD sub})$$

Subtyping relation:

$$\frac{}{t \leq_{BCD} t} \qquad \frac{}{t \leq_{BCD} t \wedge t} \qquad \frac{}{t_1 \wedge t_2 \leq_{BCD} t_1} \qquad \frac{}{(t_1 \to t_2) \wedge (t_1 \to t_3) \leq_{BCD} t_1 \to (t_2 \wedge t_3)}$$

$$\frac{t_1 \leq_{BCD} t_2 \qquad t_2 \leq_{BCD} t_3}{t_1 \leq_{BCD} t_3} \qquad \frac{t_1 \leq_{BCD} t_3 \qquad t_2 \leq_{BCD} t_4}{t_1 \wedge t_2 \leq_{BCD} t_3 \wedge t_4} \qquad \frac{t_3 \leq_{BCD} t_1 \qquad t_2 \leq_{BCD} t_4}{t_1 \to t_2 \leq_{BCD} t_3 \to t_4}$$

Figure 5. The BCD type system

In this subsection, we restrict the grammar of expressions with explicit substitutions to

$$e ::= x \mid e\ e \mid \lambda^{\wedge_{i \in I} s_i \to t_i}_{[\sigma_j]_{j \in J}} x.e \tag{21}$$

and we write $\lceil e \rceil$ for the pure λ-calculus expression obtained by removing all types references (i.e., interfaces and decorations) from e. Given a pure λ-calculus expression m and a derivation of a judgement $\Gamma \vdash_{BCD} m : t$, we build an expression e such that $\lceil e \rceil = m$ and $\Delta \wr \Gamma \vdash e : t$ for some set Δ of type variables. With the restricted grammar of (21), the intersection typing rule is used only in conjunction with the abstraction typing rule. We prove that it is possible to put a similar restriction on derivations of BCD typing judgements.

Definition B.19. *Let D be a BCD typing derivation. We say D is in* intersection-abstraction normal form *if (BCD inter) is used only immediately after (BCD abstr) in D, that is to say, all uses of (BCD inter) in D are of the form*

$$\frac{\dfrac{\dfrac{D_i}{\Gamma, x : t_i \vdash_{BCD} m : s_i}}{\Gamma \vdash_{BCD} \lambda x.m : t_i \to s_i}\, i \in I}{\Gamma \vdash_{BCD} \lambda x.m : \bigwedge_{i \in I} t_i \to s_i}$$

Definition B.20. *Let D be a (BCD) typing derivation. We define the* size *of D, denoted by $|D|$, as the number of rules used in D.*

We prove that any BCD typing judgement can be proved with a derivation in intersection-abstraction normal form.

Lemma B.21. *If $\Gamma \vdash_{BCD} m : t$, then there exists a derivation in intersection-abstraction normal form proving this judgement.*

Proof. Let D be the derivation proving $\Gamma \vdash_{BCD} m : t$. We proceed by induction on the size of D. If $|D| = 1$ then the rule (*BCD var*) has been used, and D is in intersection-abstraction normal form. Otherwise, assume that $|D| > 1$. We proceed by case analysis on the last rule used in D.

(*BCD sub*)**:** D ends with (*BCD sub*):

$$D = \frac{D' \qquad t' \leq t}{\Gamma \vdash_{BCD} m : t}$$

where D' proves a judgement $\Gamma \vdash_{BCD} m : t'$. By the induction hypothesis, there exists a derivation D'' in intersection-abstraction normal form which proves the same judgement as D'. Then

$$\frac{D'' \qquad t' \leq t}{\Gamma \vdash_{BCD} m : t}$$

is in intersection-abstraction normal form and proves the same judgement as D.

(*BCD abstr*)**:** similar to the case of (*BCD sub*).

(*BCD app*)**:** similar to the case of (*BCD sub*).

(*BCD inter*)**:** D ends with (*BCD inter*):

$$D = \frac{D_i}{\Gamma \vdash_{BCD} m : t} i \in I$$

where each D_i proves a judgement $\Gamma \vdash_{BCD} m : t_i$ and $t = \bigwedge_{i \in I} t_i$. We distinguish several cases.

If one of the derivations ends with (*BCD sub*)**,** there exists $i_0 \in I$ such that

$$D_{i_0} = \frac{D'_{i_0} \qquad t'_{i_0} \leq t_{i_0}}{\Gamma \vdash_{BCD} m : t_{i_0}}$$

The derivation

$$D' = \frac{D_i \qquad D'_{i_0}}{\Gamma \vdash_{BCD} m : \bigwedge\limits_{i \in I \setminus \{i_0\}} t_i \wedge t'_{i_0}} i \in I \setminus \{i_0\}$$

is smaller than D, so by the induction hypothesis, there exists D'' in intersection-abstraction normal form which proves the same judgement as D'. Then the derivation

$$\frac{D'' \qquad \bigwedge\limits_{i \in I \setminus \{i_0\}} t_i \wedge t'_{i_0} \leq_{BCD} \bigwedge\limits_{i \in I} t_i}{\Gamma \vdash_{BCD} m : t}$$

is in intersection-abstraction normal form, and proves the same judgement as D.

If one of the derivations ends with (*BCD inter*)**,** there exists $i_0 \in I$ such that

$$D_{i_0} = \frac{D_{j,i_0}}{\Gamma \vdash_{BCD} m : \bigwedge\limits_{j \in J} t_{j,i_0}} j \in J$$

with $t_{i_0} = \bigwedge_{j \in J} t_{j,i_0}$. The derivation

$$D' = \frac{D_i \qquad D_{j,i_0}}{\Gamma \vdash_{BCD} m : t} \begin{array}{c} i \in I \setminus \{i_0\} \\ j \in J \end{array}$$

is smaller than D, so by the induction hypothesis, there exists D'' in intersection-abstraction normal form which proves the same judgement as D', which is the same as the judgement of D.

If all the derivations are uses of (*BCD var*)**,** then for all $i \in I$, we have

$$D_i = \frac{}{\Gamma \vdash_{BCD} x : \Gamma(x)}$$

which implies $t = \bigwedge_{i \in I} \Gamma(x)$ and $m = x$. Then the derivation

$$\frac{\dfrac{}{\Gamma \vdash_{BCD} x : \Gamma(x)} \qquad \Gamma(x) \leq_{BCD} t}{\Gamma \vdash_{BCD} x : t}$$

is in intersection-abstraction normal form and proves the same judgement as D.

If all the derivations end with (*BCD app*)**,** then for all $i \in I$, we have

$$D_i = \frac{D_i^1 \qquad D_i^2}{\Gamma \vdash_{BCD} m_1\ m_2 : t_i}$$

where $m = m_1\ m_2$, D_i^1 proves $\Gamma \vdash_{BCD} m_1 : s_i \to t_i$, and D_i^2 proves $\Gamma \vdash_{BCD} m_2 : s_i$ for some s_i. Let

$$D_1 = \frac{D_i^1}{\Gamma \vdash_{BCD} m_1 : \bigwedge\limits_{i \in I} s_i \to t_i} i \in I \qquad\qquad D_2 = \frac{D_i^2}{\Gamma \vdash_{BCD} m_2 : \bigwedge\limits_{i \in I} s_i} i \in I$$

Both D_1 and D_2 are smaller than D, so by the induction hypothesis, there exist D'_1, D'_2 in intersection-abstraction normal form which prove the same judgements as D_1 and D_2 respectively.

Then the derivation

$$\dfrac{\dfrac{D_1' \qquad \bigwedge_{i\in I} s_i \to t_i \leq_{BCD} (\bigwedge_{i\in I} s_i) \to (\bigwedge_{i\in I} t_i)}{\Gamma \vdash_{BCD} m_1 : (\bigwedge_{i\in I} s_i) \to (\bigwedge_{i\in I} t_i)} \qquad D_2'}{\Gamma \vdash m_1\ m_2 : t}$$

is in intersection-abstraction normal form and proves the same derivation as D.

If all the derivations end with (*BCD abstr*)**,** then for all $i \in I$, we have

$$D_i = \dfrac{D_i'}{\Gamma \vdash_{BCD} \lambda x.m' : t_i}\ (\textit{BCD abstr})$$

with $m = \lambda x.m'$. For all $i \in I$, D_i' is smaller than D, so by induction there exists D_i'' in intersection-abstraction normal form which proves the same judgement as D_i'. Then the derivation

$$\dfrac{\dfrac{D_i''}{\Gamma \vdash_{BCD} \lambda x.m' : t_i}\, i \in I}{\Gamma \vdash_{BCD} \lambda x.m' : \bigwedge_{i\in I} t_i}$$

is in intersection-abstraction normal form, and proves the same judgement as D.

□

We now sketch the principles behind the construction of e from D in intersection-abstraction normal form. If D proves a judgement $\Gamma \vdash_{BCD} \lambda x.m : t \to s$, without any top-level intersection, then we simply put $t \to s$ in the interface of the corresponding expression $\lambda^{t\to s} x.e$.

For a judgement $\Gamma \vdash_{BCD} \lambda x.m : \bigwedge_{i\in I} t_i \to s_i$, we build an expression $\lambda^{\alpha\to\beta}_{[\sigma_i]_{i\in I}} x.e$ where each σ_i corresponds to the derivation which types $\lambda x.m$ with $t_i \to s_i$. For example, let $m = \lambda f.\lambda x.f\ x$, and consider the judgement $\vdash_{BCD} m : ((t_1 \to t_2) \to t_1 \to t_2) \wedge ((s_1 \to s_2) \to s_1 \to s_2)$. We first annotate each abstraction in m with types $\alpha_j \to \beta_j$, where α_j, β_j are fresh, distinct variables, giving us $e = \lambda^{\alpha_1\to\beta_1} f.\lambda^{\alpha_2\to\beta_2} x.f\ x$. Comparing $\lambda^{\alpha_2\to\beta_2} x.f\ x$ to the judgement $f : t_1 \to t_2 \vdash_{BCD} \lambda x.f\ x : t_1 \to t_2$ and e to $\vdash_{BCD} m : (t_1 \to t_2) \to t_1 \to t_2$, we compute $\sigma_1 = \{t_1 \to t_2/\alpha_1, t_1 \to t_2/\beta_1, t_1/\alpha_2, t_2/\beta_2\}$. We compute similarly σ_2 from the derivation of $\vdash_{BCD} m : (s_1 \to s_2) \to s_1 \to s_2$, and we obtain finally

$$\vdash \lambda^{\alpha_1\to\beta_1}_{[\sigma_1,\sigma_2]} f.\lambda^{\alpha_2\to\beta_2} x.f\ x : ((t_1 \to t_2) \to t_1 \to t_2) \wedge ((s_1 \to s_2) \to s_1 \to s_2)$$

as wished.

The problem becomes more complex when we have nested uses of the intersection typing rule. For example, let $m = \lambda f.\lambda g.g\ (\lambda x.f\ (\lambda y.x\ y))$ and consider the judgement $\vdash_{BCD} m : (t_f \to t_g \to t_4) \wedge (s_f \to s_g \to s_7)$ with

$$\begin{aligned} t_f &= (t_1 \to t_2) \to t_3 \\ t_g &= t_f \to t_4 \\ s_f &= ((s_1 \to s_2) \to s_3) \wedge ((s_4 \to s_5) \to s_6) \\ s_g &= s_f \to s_7 \end{aligned}$$

Notice that, to derive $\vdash_{BCD} m : s_f \to s_g \to s_7$, we have to prove $f : s_f, g : s_g \vdash_{BCD} \lambda x.f\ (\lambda y.x\ y) : s_f$, which requires the *(BCD inter)* rule. As before, we annotate m with fresh interfaces, obtaining $\lambda^{\alpha_1\to\beta_1} f.\lambda^{\alpha_2\to\beta_2} g.g\ (\lambda^{\alpha_3\to\beta_3} x.f\ (\lambda^{\alpha_4\to\beta_4} y.x\ y))$. Because the intersection typing rule is used twice (once to type m, and once to type $m' = \lambda x.f\ (\lambda y.x\ y)$), we want to compute four substitutions $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ to obtain a decorated expression $\lambda^{\alpha_1\to\beta_1}_{[\sigma_1,\sigma_2]} f.\lambda^{\alpha_2\to\beta_2} g.g\ (\lambda^{\alpha_3\to\beta_3}_{[\sigma_3,\sigma_4]} x.f\ (\lambda^{\alpha_4\to\beta_4} y.x\ y))$. The difficult part is in computing σ_3 and σ_4; in one case (corresponding to the branch $\vdash_{BCD} m : t_f \to t_g \to t_4$), we want $\sigma_3 = \sigma_4 = \{t_1 \to t_2/\alpha_3, t_3/\beta_3, t_1/\alpha_4, t_2/\beta_4\}$ to obtain $f : t_f, g : t_g \vdash \lambda^{\alpha_3\to\beta_3}_{[\sigma_3,\sigma_4]} x.f\ (\lambda^{\alpha_4\to\beta_4} y.x\ y) : t_f$, and in the other case (corresponding to the derivation $\vdash_{BCD} m : s_f \to s_g \to s_7$), we want $\sigma_3 = \{s_1 \to s_2/\alpha_3, s_3/\beta_3, s_1/\alpha_4, s_2/\beta_4\}$ and $\sigma_4 = \{s_4 \to s_5/\alpha_3, s_6/\beta_3, s_4/\alpha_4, s_5/\beta_4\}$ to obtain $f : s_f, g : s_g \vdash \lambda^{\alpha_3\to\beta_3}_{[\sigma_3,\sigma_4]} x.f\ (\lambda^{\alpha_4\to\beta_4} y.x\ y) : s_f$. To resolve this issue, we use intermediate fresh variables $\alpha_3', \beta_3', \alpha_4', \beta_4'$ and $\alpha_3'', \beta_3'', \alpha_4'', \beta_4''$ in the definition of σ_3 and σ_4. We define

$$\begin{aligned} \sigma_1 =& \{t_f/\alpha_1, t_g \to t_4/\beta_1, t_g/\alpha_2, t_4/\beta_2, t_1 \to t_2/\alpha_3', t_3/\beta_3', t_1/\alpha_4', t_2/\beta_4', \\ & t_1 \to t_2/\alpha_3'', t_3/\beta_3'', t_1/\alpha_4'', t_2/\beta_4''\} \\ \sigma_2 =& \{s_f/\alpha_1, s_g \to s_7/\beta_1, s_g/\alpha_2, s_7/\beta_2, s_1 \to s_2/\alpha_3', s_3/\beta_3', s_1/\alpha_4', s_2/\beta_4', \\ & s_4 \to s_5/\alpha_3'', s_6/\beta_3'', s_4/\alpha_4'', s_5/\beta_4''\} \\ \sigma_3 =& \{\alpha_3'/\alpha_3, \beta_3'/\beta_3, \alpha_4'/\alpha_4, \beta_4'/\beta_4\} \\ \sigma_4 =& \{\alpha_3''/\alpha_3, \beta_3''/\beta_3, \alpha_4''/\alpha_4, \beta_4''/\beta_4\} \end{aligned}$$

Because the substitutions compose themselves, we obtain

$$\vdash \lambda^{\alpha_1\to\beta_1}_{[\sigma_1,\sigma_2]} f.\lambda^{\alpha_2\to\beta_2} g.g\ (\lambda^{\alpha_3\to\beta_3}_{[\sigma_3,\sigma_4]} x.f\ (\lambda^{\alpha_4\to\beta_4} y.x\ y)) : (t_f \to t_g \to t_4) \wedge (s_f \to s_g \to s_7)$$

as wished.

In the next lemma, given n derivations $D_1, \ldots, D_n$ in intersection-abstraction normal form for a same expression m, we construct an expression e containing fresh interfaces and decorations with fresh variables if needed (as explained in the above example) and n substitutions $\sigma_1, \ldots, \sigma_n$ corresponding to $D_1, \ldots, D_n$. Let $\mathsf{var}(D_1, \ldots, D_n)$ denote the set of type variables occurring in the types in $D_1, \ldots, D_n$.

Lemma B.22. *Let m be a pure λ-calculus expression, Δ be a set of type variables, and $D_1, \ldots, D_n$ be derivations in intersection-abstraction normal form such that D_i proves $\Gamma_i \vdash_{BCD} m : t_i$ for all i. Let Δ' be a set of type variables such that $\mathsf{var}(D_1, \ldots, D_n) \subseteq \Delta'$. There exist $e, \sigma_1, \ldots, \sigma_n$ such that $\lceil e \rceil = m$, $\mathsf{dom}(\sigma_1) = \ldots = \mathsf{dom}(\sigma_n) \subseteq \mathsf{tv}(e)$, $\mathsf{tv}(e) \cap (\Delta \cup \Delta') = \emptyset$, and $\Delta' \wr \Gamma_i \vdash e@[\sigma_i] : t_i$ for all i.*

Proof. We proceed by induction on the sum of the size of $D_1, \ldots, D_n$. If this sum is equal to n, then each D_i is a use of the *(BCD var)* rule, and we have $m = x$ for some x. Let $e = x$ and σ_i be the identity; we can then easily check that the result holds. Otherwise, assume this sum is strictly greater than n. We proceed by case analysis on $D_1, \ldots, D_n$.

Case 1: If one of the derivations ends with (*BCD sub*), there exists i_0 such that

$$D_{i_0} = \frac{D'_{i_0} \qquad t'_{i_0} \leq_{BCD} t_{i_0}}{\Gamma_{i_0} \vdash_{BCD} m : t_{i_0}}$$

Clearly, the sum of the size of $D_1, \ldots, D'_{i_0}, \ldots, D_n$ is smaller than that of $D_1, \ldots, D_n$, and $\mathsf{var}(D_1, \ldots, D'_{i_0}, \ldots, D_n) \subseteq \Delta'$. So by the induction hypothesis, we have

$$\begin{aligned} \exists e, \sigma_1, \ldots, \sigma_n. \quad & \lceil e \rceil = m \\ & \text{and } \mathsf{dom}(\sigma_1) = \ldots = \mathsf{dom}(\sigma_n) \subseteq \mathsf{tv}(e) \\ & \text{and } \mathsf{tv}(e) \cap (\Delta \cup \Delta') = \emptyset \\ & \text{and } \forall i \in \{1, \ldots, n\} \setminus \{i_0\}.\ \Delta' \wr \Gamma_i \vdash e@[\sigma_i] : t_i \\ & \text{and } \Delta' \wr \Gamma_{i_0} \vdash e@[\sigma_{i_0}] : t'_{i_0}. \end{aligned}$$

Since $t'_{i_0} \leq_{BCD} t_{i_0}$, by Lemma B.18, we have $t'_{i_0} \leq t_{i_0}$. Therefore $\Delta' \wr \Gamma_{i_0} \vdash e@[\sigma_{i_0}] : t_{i_0}$ holds, and for all i, we have $\Delta' \wr \Gamma_i \vdash e@[\sigma_i] : t_i$ as wished.

Case 2: If all the derivations end with (*BCD app*), then we have $m = m_1\ m_2$, and for all i

$$D_i = \frac{D^1_i \qquad D^2_i}{\Gamma_i \vdash_{BCD} m_1\ m_2 : t_i}$$

where D^1_i proves $\Gamma_i \vdash_{BCD} m_1 : s_i \to t_i$ and D^2_i proves $\Gamma_i \vdash_{BCD} m_2 : s_i$ for some s_i. Applying the induction hypothesis on $D^1_1, \ldots, D^1_n$ (with Δ and Δ'), we have

$$\begin{aligned} \exists e_1, \sigma^1_1, \ldots, \sigma^1_n. \quad & \lceil e \rceil_1 = m_1 \\ & \text{and } \mathsf{dom}(\sigma^1_1) = \ldots = \mathsf{dom}(\sigma^1_n) \subseteq \mathsf{tv}(e_1) \\ & \text{and } \mathsf{tv}(e_1) \cap (\Delta \cup \Delta') = \emptyset \\ & \text{and } \forall i \in \{1, \ldots, n\}.\ \Delta' \wr \Gamma_i \vdash e_1@[\sigma^1_i] : s_i \to t_i. \end{aligned}$$

Similarly, by induction on $D^2_1, \ldots, D^2_n$ (with $\Delta \cup \mathsf{tv}(e_1)$ and Δ'),

$$\begin{aligned} \exists e_2, \sigma^2_1, \ldots, \sigma^2_n. \quad & \lceil e \rceil_2 = m_2 \\ & \text{and } \mathsf{dom}(\sigma^2_1) = \ldots = \mathsf{dom}(\sigma^2_n) \subseteq \mathsf{tv}(e_2) \\ & \text{and } \mathsf{tv}(e_2) \cap (\Delta \cup \mathsf{tv}(e_1) \cup \Delta') = \emptyset \\ & \text{and } \forall i \in \{1, \ldots, n\}.\ \Delta' \wr \Gamma_i \vdash e_2@[\sigma^2_i] : s_i. \end{aligned}$$

From $\mathsf{tv}(e_2) \cap (\Delta \cup \mathsf{tv}(e_1) \cup \Delta') = \emptyset$, we deduce $\mathsf{tv}(e_1) \cap \mathsf{tv}(e_2) = \emptyset$. Let $i \in \{1, \ldots, n\}$. Because $\mathsf{dom}(\sigma^1_i) \subseteq \mathsf{tv}(e_1)$ and $\mathsf{dom}(\sigma^2_i) \subseteq \mathsf{tv}(e_2)$, we have $\mathsf{dom}(\sigma^1_i) \cap \mathsf{dom}(\sigma^2_i) = \emptyset$, $\mathsf{dom}(\sigma^1_i) \cap \mathsf{tv}(e_2) = \emptyset$, and $\mathsf{dom}(\sigma^2_i) \cap \mathsf{tv}(e_1) = \emptyset$. Consequently, by Lemma B.9, we have $\Delta' \wr \Gamma_i \vdash e_1@[\sigma^1_i \cup \sigma^2_i] : s_i \to t_i$ and $\Delta' \wr \Gamma_i \vdash e_2@[\sigma^1_i \cup \sigma^2_i] : s_i$. Therefore, we have $\Delta' \wr \Gamma_i \vdash (e_1\ e_2)@[\sigma^1_i \cup \sigma^2_i] : t_i$. So we have the required result with $e = e_1\ e_2$ and $\sigma_i = \sigma^1_i \cup \sigma^2_i$.

Case 3: If all the derivations end with (*BCD abstr*), then $m = \lambda x.m_1$, and for all i,

$$D_i = \frac{D'_i}{\Gamma_i \vdash_{BCD} m : t_i}$$

where D'_i proves $\Gamma_i, x : t^1_i \vdash_{BCD} m_1 : t^2_i$ and $t_i = t^1_i \to t^2_i$. By the induction hypothesis,

$$\begin{aligned} \exists e_1, \sigma'_1, \ldots, \sigma'_n. \quad & \lceil e \rceil_1 = m_1 \\ & \text{and } \mathsf{dom}(\sigma'_1) = \ldots = \mathsf{dom}(\sigma'_n) \subseteq \mathsf{tv}(e_1) \\ & \text{and } \mathsf{tv}(e_1) \cap (\Delta \cup \Delta') = \emptyset \\ & \text{and } \forall i \in \{1, \ldots, n\}.\ \Delta' \wr \Gamma_i, x : t^1_i \vdash e_1@[\sigma'_i] : t^2_i. \end{aligned}$$

Let α, β be two fresh type variables. So $\{\alpha, \beta\} \cap (\Delta \cup \Delta') = \emptyset$ and $\{\alpha, \beta\} \cap \mathsf{tv}(e_1) = \emptyset$. Take $i \in \{1, \ldots, n\}$. Let $\sigma_i = \{t^1_i/\alpha, t^2_i/\beta\} \cup \sigma'_i$, and $e = \lambda^{\alpha\to\beta} x.e_1$. We have $\mathsf{dom}(\sigma_i) =$

$\{\alpha,\beta\} \cup \mathsf{dom}(\sigma'_i) \subseteq \{\alpha,\beta\} \cup \mathsf{tv}(e_1) = \mathsf{tv}(e)$. Besides, we have $\mathsf{tv}(e) \cap (\Delta \cup \Delta') = \emptyset$. Because $\mathsf{tv}(e_1) \cap \{\alpha,\beta\} = \emptyset$, we have $\mathsf{dom}(\sigma'_i) \cap \{\alpha,\beta\} = \emptyset$, and $\Delta' \wr \Gamma_i, x : t_i^1 \vdash e_1@[\sigma_i] : t_i^2$ by Lemma B.9, which is equivalent to $\Delta' \wr \Gamma_i, x : \alpha\sigma_i \vdash e_1@[\sigma_i] : \beta\sigma_i$. Because $\Delta' \cup \mathsf{var}(t_i^1 \to t_i^2) = \Delta'$, by the abstraction rule, we have $\Delta' \wr \Gamma_i \vdash \lambda^{\alpha\to\beta}_{[\sigma_i]} x.e_1 : t_i$, *i.e.*, $\Delta' \wr \Gamma_i \vdash (\lambda^{\alpha\to\beta}.e_1)@[\sigma_i] : t_i$. Therefore, we have the required result.

Case 4: If one of the derivations ends with (*BCD inter*), then $m = \lambda x.m_1$. The derivations end with either (*BCD inter*) or (*BCD abstr*) (we omit the already treated case of (*BCD sub*)). For simplicity, we suppose they all end with (*BCD inter*), as it is the same if some of them end with (*BCD abstr*) (note that Case 3 is a special case of Case 4). For all i, we have

$$D_i = \dfrac{\dfrac{D_i^j}{\Gamma_i \vdash_{BCD} m : s_i^j \to t_i^j}}{\Gamma_i \vdash_{BCD} m : \bigwedge\limits_{j\in J_i} s_i^j \to t_i^j} j \in J_i$$

where D_i^j proves $\Gamma_i, x : s_i^j \vdash_{BCD} m_1 : t_i^j$ for all $j \in J_i$ and $t_i = \bigwedge_{j\in J_i} s_i^j \to t_i^j$ for all i. By the induction hypothesis on the sequence of D_i^j,

$$\begin{aligned}\exists e_1, (\sigma_1^j)_{j\in J_1}, \dots, (\sigma_n^j)_{j\in J_n}. \quad & \lceil e_1 \rceil = m_1 \\ & \text{and } \forall i,i',j,j'.\ \mathsf{dom}(\sigma_i^j) = \mathsf{dom}(\sigma_{i'}^{j'}) \text{ and } \mathsf{dom}(\sigma_i^j) \subseteq \mathsf{tv}(e_1) \\ & \text{and } \mathsf{tv}(e_1) \cap (\Delta \cup \Delta') = \emptyset \\ & \text{and } \forall i,j.\ \Delta' \wr \Gamma_i, x : s_i^j \vdash e_1@[\sigma_i^j] : t_i^j.\end{aligned}$$

Let $p = max_{i\in\{1,\dots,n\}}\{|J_i|\}$. For all i, we complete the sequence of substitutions (σ_i^j) so that it contains exactly p elements, by repeating the last one $p - |J_i|$ times, and we number them from 1 to p. All the σ_i^j have the same domain (included in $\mathsf{tv}(e_1)$), that we number from 1 to q. Then $\sigma_i^j = \bigcup_{k\in\{1,\dots,q\}}\{t_{i,j}^k/\alpha_k\}$ for some types $(t_{i,j}^k)$. Let $\alpha, \beta, (\alpha_{j,k})_{j\in\{1,\dots,p\},k\in\{1,\dots,q\}}$, $(\alpha_{j,0},\beta_{j,0})_{j\in\{1,\dots,p\}}$ be fresh pairwise distinct variables (which do not occur in $\Delta \cup \Delta' \cup \mathsf{tv}(e_1)$). For all $j \in \{1,\dots,p\}$, $i \in \{1,\dots,n\}$, we define:

$$\begin{aligned}\sigma_j &= \bigcup_{k\in\{1,\dots,q\}} \{\alpha_{j,k}/\alpha_k\} \cup \{\alpha_{j,0}/\alpha, \beta_{j,0}/\beta\} \\ e &= \lambda^{\alpha\to\beta}_{[\sigma_j]_{j\in\{1,\dots,p\}}} x.e_1 \\ \sigma_i &= \bigcup_{j\in\{1,\dots,p\},k\in\{1,\dots,q\}} \{t_{i,j}^k/\alpha_{j,k}\} \cup \bigcup_{j\in\{1,\dots,p\}} \{s_i^j/\alpha_{j,0}, t_i^j/\beta_{j,0}\}\end{aligned}$$

For all i, j, k, we have by construction $(\alpha_k\sigma_j)\sigma_i = \alpha_k\sigma_i^j$, $(\alpha\sigma_j)\sigma_i = s_i^j$, and $(\beta\sigma_j)\sigma_i = t_i^j$. Moreover, since

$$\begin{aligned}\mathsf{tv}(e) &= \mathsf{tv}(e_1@[\sigma_j]_{j\in\{1,\dots,p\}}) \cup \textstyle\bigcup_{j\in\{1,\dots,p\}} \mathsf{var}((\alpha\to\beta)\sigma_j) \\ &= (\mathsf{tv}(e_1))[\sigma_j]_{j\in\{1,\dots,p\}} \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}} \\ &\supseteq (\mathsf{dom}(\sigma_i^j))[\sigma_j]_{j\in\{1,\dots,p\}} \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}} \\ &= (\{\alpha_k\}_{k\in\{1,\dots,q\}})[\sigma_j]_{j\in\{1,\dots,p\}} \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}} \\ &= \{\alpha_{j,k},\beta_{j,k}\}_{j\in\{1,\dots,p\},k\in\{1,\dots,q\}} \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}}\end{aligned}$$

and

$$\begin{aligned}\mathsf{tv}(e) &= (\mathsf{tv}(e_1))[\sigma_j]_{j\in\{1,\dots,p\}} \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}} \\ &\subseteq \mathsf{tv}(e_1) \cup \textstyle\bigcup_{j\in\{1,\dots,p\}} \mathsf{tvran}(\sigma_j) \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}} \\ &= \mathsf{tv}(e_1) \cup \{\alpha_{j,k},\beta_{j,k}\}_{j\in\{1,\dots,p\},k\in\{1,\dots,q\}} \cup \{\alpha_{j,0},\beta_{j,0}\}_{j\in\{1,\dots,p\}}\end{aligned}$$

we have $\mathsf{dom}(\sigma_i) \subseteq \mathsf{tv}(e)$ and $\mathsf{tv}(e) \cap (\Delta \cup \Delta') = \emptyset$. Because $\Delta' \wr \Gamma_i, x : s_i^j \vdash e_1@[\sigma_i^j] : t_i^j$, by Lemma B.9, we have $\Delta' \wr \Gamma_i, x : s_i^j \vdash e_1@[\sigma_i \circ \sigma_j] : t_i^j$, which is equivalent to $\Delta' \wr \Gamma_i, x : \alpha(\sigma_i \circ \sigma_j) \vdash e_1@[\sigma_i \circ \sigma_j] : \beta(\sigma_i \circ \sigma_j)$ for all i, j. Since $\Delta' \cup \bigcup_{j\in\{1,\dots,p\}} \mathsf{var}(s_i^j \to t_i^j) = \Delta'$, for a given i, by the abstraction typing rule we have $\Delta' \wr \Gamma_i \vdash \lambda^{\alpha\to\beta}_{[\sigma_i\circ\sigma_j]_{j\in\{1,\dots,p\}}} x.e_1 : \bigwedge_{j\in\{1,\dots,p\}} s_i^j \to t_i^j \leq \bigwedge_{j\in J_i} s_i^j \to t_i^j = t_i$. This is equivalent to $\Delta' \wr \Gamma_i \vdash \lambda^{\alpha\to\beta}_{[\sigma_j]_{j\in\{1,\dots,p\}}} x.e_1@[\sigma_i] : t_i$, hence $\Delta' \wr \Gamma_i \vdash e@[\sigma_i] : t_i$ holds for all i, as wished.

□

We are now ready to prove the main result of this subsection.

Theorem B.23. *If $\Gamma \vdash_{BCD} m : t$, then there exist e, Δ such that $\Delta \wr \Gamma \vdash e : t$ and $\lceil e \rceil = m$.*

Proof. By Lemma B.21, there exists D in intersection-abstraction normal form such that D proves $\Gamma \vdash_{BCD} m : t$. Let Δ be a set of type variables such that $\mathsf{var}(D) \subseteq \Delta$. We prove by induction on $|D|$ that there exists e such that $\Delta \wr \Gamma \vdash e : t$ and $\lceil e \rceil = m$.

Case (*BCD var*)**:** The expression m is a variable and the result holds with $e = m$.

Case (*BCD sub*)**:** We have

$$D = \frac{D' \qquad t' \leq_{BCD} t}{\Gamma \vdash_{BCD} m : t}$$

where D' in intersection-abstraction normal form and proves $\Gamma \vdash_{BCD} m : t'$. Clearly we have $|D'| < |D|$ and $\mathsf{var}(D') \subseteq \Delta$, so by the induction hypothesis, there exists e such that $\lceil e \rceil = m$ and $\Delta \wr \Gamma \vdash e : t'$. By Lemma B.18, we have $t' \leq t$, therefore we have $\Delta \wr \Gamma \vdash e : t$, as wished.

Case (*BCD app*)**:** We have

$$D = \frac{D_1 \qquad D_2}{\Gamma \vdash_{BCD} m : t}$$

where D_1 proves $\Gamma \vdash_{BCD} m_1 : s \to t$, D_2 proves $\Gamma \vdash_{BCD} m_2 : s$, $m = m_1\ m_2$, and both D_1 and D_2 are in intersection-abstraction normal form. Since $|D_i| < |D|$ and $\mathsf{var}(D_i) \subseteq \Delta$, by the induction hypothesis, there exist e_1 and e_2 such that $\lceil e_1 \rceil = m_1$, $\lceil e_2 \rceil = m_2$, $\Delta \wr \Gamma \vdash e_1 : s \to t$, and $\Delta \wr \Gamma \vdash e_2 : s$. Consequently we have $\Delta \wr \Gamma \vdash e_1\ e_2 : t$, with $\lceil e_1\ e_2 \rceil = m_1\ m_2$, as wished.

Case (*BCD abstr*) **(or** (*BCD inter*)**):** Because D is in intersection-abstraction normal form, we have

$$D = \frac{\dfrac{D_i}{\Gamma \vdash_{BCD} \lambda x.m' : s_i \to t_i}\, i \in I}{\Gamma \vdash_{BCD} m : t}$$

where each D_i is in intersection-abstraction normal form and proves $\Gamma, x : s_i \vdash_{BCD} m' : t_i$, $t = \bigwedge_{i \in I} s_i \to t_i$, and $m = \lambda x.m'$. Since $\bigcup_{i \in I} \mathsf{var}(D_i) \subseteq \Delta$, by Lemma B.22, there exist e', $\sigma_1, \ldots, \sigma_n$ such that $\lceil e' \rceil = m'$, $\mathsf{dom}(\sigma_1) = \ldots = \mathsf{dom}(\sigma_n) \subseteq \mathsf{tv}(e')$, and $\Delta \wr \Gamma, x : s_i \vdash e'@[\sigma_i] : t_i$ for all $i \in I$. Let α, β be two fresh type variables. We define $\sigma'_i = \sigma_i \cup \{s_i/\alpha, t_i/\beta\}$ for all $i \in I$. Because $\mathsf{dom}(\sigma_i) \cap \{\alpha, \beta\} = \emptyset$ and $\mathsf{tv}(e') \cap \{\alpha, \beta\} = \emptyset$, by Lemma B.9 we have $\Delta \wr \Gamma, x : s_i \vdash e'@[\sigma'_i] : t_i$, which is equivalent to $\Delta \wr \Gamma, x : \alpha\sigma'_i \vdash e'@[\sigma'_i] : \beta\sigma'_i$. Note that $\Delta \cup \mathsf{var}(\bigwedge_{i \in I}(\alpha \to \beta)\sigma'_i) = \Delta \cup \mathsf{var}(\bigwedge_{i \in I} s_i \to t_i) = \Delta$ by definition of Δ, so by rule *(abstr)*, we have $\Delta \wr \Gamma \vdash \lambda^{\alpha \to \beta}_{[\sigma'_i]_{i \in I}} x.e' : \bigwedge_{i \in I} s_i \to t_i$. Hence we have the required result with $e = \lambda^{\alpha \to \beta}_{[\sigma'_i]_{i \in I}} x.e'$.

□

B.4 Elimination of sets of type-substitutions

In this section we prove that the expressions of the form $e[\sigma_j]_{j \in J}$ are redundant insofar as their presence in the calculus does not increase its expressive power. For that we consider a subcalculus, called *normalized calculus*, in which sets of type-substitutions appear only in decorations.

Definition B.24. *A* normalized expression e *is an expression without any subterm of the form* $e[\sigma_j]_{j \in J}$*, i.e., an expression respecting the following grammar:*

$$e ::= c \mid x \mid (e, e) \mid \pi_i(e) \mid e\ e \mid \lambda^{\bigwedge_{i \in I} s_i \to t_i}_{[\sigma_j]_{j \in J}} x.e \mid e \in t\,?\,e : e$$

The set of all normalized expressions is denoted as $\mathscr{E}_N$.

We then define an embedding of the full calculus into this subcalculus as follows:

Definition B.25. *The* embedding *emd*$(.)$ *is mapping from* $\mathscr{E}$ *to* $\mathscr{E}_N$ *defined as*

$$\begin{aligned}
\mathit{emd}(c) &= c \\
\mathit{emd}(x) &= x \\
\mathit{emd}((e_1, e_2)) &= (\mathit{emd}(e_1), \mathit{emd}(e_2)) \\
\mathit{emd}(\pi_i(e)) &= \pi_i(\mathit{emd}(e)) \\
\mathit{emd}(\lambda^{\bigwedge_{i \in I} t_i \to s_i}_{[\sigma_j]_{j \in J}} x.e) &= \lambda^{\bigwedge_{i \in I} t_i \to s_i}_{[\sigma_j]_{j \in J}} x.\mathit{emd}(e) \\
\mathit{emd}(e_1 e_2) &= \mathit{emd}(e_1)\mathit{emd}(e_2) \\
\mathit{emd}(e \in t\,?\,e_1 : e_2) &= \mathit{emd}(e) \in t\,?\,\mathit{emd}(e_1) : \mathit{emd}(e_2) \\
\mathit{emd}(e[\sigma_j]_{j \in J}) &= \mathit{emd}(e@[\sigma_j]_{j \in J})
\end{aligned}$$

We want to prove that the subcalculus has the same expressive power as the full calculus, namely, given an expression and its embedding, they reduce to the same value. We proceed in several steps, using auxiliary lemmas.

First we show that the embedding preserves values.

Lemma B.26. *Let* $v \in \mathscr{V}$ *be a value. Then* $\mathit{emd}(v) \in \mathscr{V}$.

Proof. Straightforward. □

Then we prove that values and their embeddings have the same types.

Lemma B.27. *Let* $v \in \mathscr{V}$ *be a value. Then* $\vdash v : t \iff \vdash \mathit{emd}(v) : t$.

Proof. By induction and case analysis on v (note that $\mathit{emd}(.)$ does not change the types in the interfaces). □

We now want to prove the embedding preserves reduction, that is if an expression e reduces to e' in the full calculus, then its embedding $\mathit{emd}(e)$ reduces to $\mathit{emd}(e')$ in the subcalculus. Before that we show a substitution lemma.

Lemma B.28. *Let e be an expression, x an expression variable and v a value. Then $\mathit{emd}(e\{v/x\}) = \mathit{emd}(e)\{\mathit{emd}(v)/x\}$.*

Proof. By induction and case analysis on e.

$\underline{c}$**:**

$$\begin{aligned}\mathit{emd}(c\{v/x\}) &= \mathit{emd}(c)\\ &= c\\ &= c\{\mathit{emd}(v)/x\}\\ &= \mathit{emd}(c)\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{y}$**:**

$$\begin{aligned}\mathit{emd}(y\{v/x\}) &= \mathit{emd}(y)\\ &= y\\ &= y\{\mathit{emd}(v)/x\}\\ &= \mathit{emd}(y)\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{x}$**:**

$$\begin{aligned}\mathit{emd}(x\{v/x\}) &= \mathit{emd}(v)\\ &= x\{\mathit{emd}(v)/x\}\\ &= \mathit{emd}(x)\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{(e_1, e_2)}$**:**

$$\begin{aligned}\mathit{emd}((e_1, e_2)\{v/x\}) &= \mathit{emd}((e_1\{v/x\}, e_2\{v/x\}))\\ &= (\mathit{emd}(e_1\{v/x\}), \mathit{emd}(e_2\{v/x\}))\\ &= (\mathit{emd}(e_1)\{\mathit{emd}(v)/x\}, \mathit{emd}(e_2)\{\mathit{emd}(v)/x\}) \quad \text{(by induction)}\\ &= (\mathit{emd}(e_1), \mathit{emd}(e_2))\{\mathit{emd}(v)/x\}\\ &= \mathit{emd}((e_1, e_2))\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{\pi_i(e')}$**:**

$$\begin{aligned}\mathit{emd}(\pi_i(e')\{v/x\}) &= \mathit{emd}(\pi_i(e'\{v/x\}))\\ &= \pi_i(\mathit{emd}(e'\{v/x\}))\\ &= \pi_i(\mathit{emd}(e')\{\mathit{emd}(v)/x\}) \quad \text{(by induction)}\\ &= \pi_i(\mathit{emd}(e'))\{\mathit{emd}(v)/x\}\\ &= \mathit{emd}(\pi_i(e'))\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{e_1 e_2}$**:**

$$\begin{aligned}\mathit{emd}((e_1 e_2)\{v/x\}) &= \mathit{emd}((e_1\{v/x\})(e_2\{v/x\}))\\ &= \mathit{emd}(e_1\{v/x\})\mathit{emd}(e_2\{v/x\})\\ &= (\mathit{emd}(e_1)\{\mathit{emd}(v)/x\})(\mathit{emd}(e_2)\{\mathit{emd}(v)/x\}) \quad \text{(by induction)}\\ &= (\mathit{emd}(e_1)\mathit{emd}(e_2))\{\mathit{emd}(v)/x\}\\ &= \mathit{emd}(e_1 e_2)\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.e_0}$**:** using α-conversion, we can assume that $\mathsf{tv}(v) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$.

$$\begin{aligned}\mathit{emd}((\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.e_0)\{v/x\}) &= \mathit{emd}(\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.e_0\{v/x\})\\ &= \lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.\mathit{emd}(e_0\{v/x\})\\ &= \lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.(\mathit{emd}(e_0)\{\mathit{emd}(v)/x\}) \quad \text{(by induction)}\\ &= (\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.\mathit{emd}(e_0))\{\mathit{emd}(v)/x\}\\ &= (\mathit{emd}(\lambda^{\wedge_{i\in I} t_i\to s_i}_{[\sigma_j]_{j\in J}} y.e_0))\{\mathit{emd}(v)/x\}\end{aligned}$$

$\underline{e_0 \in t\,?\,e_1 : e_2}$**:**

$$\begin{aligned}&\mathit{emd}((e_0 \in t\,?\,e_1 : e_2)\{v/x\})\\ =\ &\mathit{emd}((e_0\{v/x\}) \in t\,?\,(e_1\{v/x\}) : (e_2\{v/x\}))\\ =\ &\mathit{emd}(e_0\{v/x\}) \in t\,?\,\mathit{emd}(e_1\{v/x\}) : \mathit{emd}(e_2\{v/x\})\\ =\ &\mathit{emd}(e_0)\{\mathit{emd}(v)/x\} \in t\,?\,(\mathit{emd}(e_1)\{\mathit{emd}(v)/x\}) : (\mathit{emd}(e_2)\{\mathit{emd}(v)/x\}) \quad \text{(by induction)}\\ =\ &(\mathit{emd}(e_0) \in t\,?\,\mathit{emd}(e_1) : \mathit{emd}(e_2))\{\mathit{emd}(v)/x\}\\ =\ &\mathit{emd}(e_0 \in t\,?\,e_1 : e_2)\{\mathit{emd}(v)/x\}\end{aligned}$$

$e'[\sigma_j]_{j\in J}$: using α-conversion, we can assume that $\mathsf{tv}(v) \cap \bigcup_{j\in J} \mathsf{dom}(\sigma_j) = \emptyset$

$$\begin{aligned} \mathit{emd}((e'[\sigma_j]_{j\in J})\{v/x\}) &= \mathit{emd}((e'\{v/x\})[\sigma_j]_{j\in J}) \\ &= \mathit{emd}((e'\{v/x\})@[\sigma_j]_{j\in J}) \\ &= \mathit{emd}((e'@[\sigma_j]_{j\in J})\{v/x\}) \quad \text{(Lemma B.5)} \\ &= \mathit{emd}(e'@[\sigma_j]_{j\in J})\{\mathit{emd}(v)/x\} \quad \text{(by induction)} \\ &= \mathit{emd}(e'[\sigma_j]_{j\in J})\{\mathit{emd}(v)/x\} \end{aligned}$$

□

Lemma B.29. *Let $e \in \mathscr{E}$ be an expression. If $e \rightsquigarrow e'$, then $\mathit{emd}(e) \rightsquigarrow^* \mathit{emd}(e')$.*

Proof. By induction and case analysis on e.

c, x: irreducible.

(e_1, e_2): there are two ways to reduce e:
(1) $e_1 \rightsquigarrow e_1'$. By induction, $\mathit{emd}(e_1) \rightsquigarrow^* \mathit{emd}(e_1')$. Then we have $(\mathit{emd}(e_1), \mathit{emd}(e_2)) \rightsquigarrow^* (\mathit{emd}(e_1'), \mathit{emd}(e_2))$, that is, $\mathit{emd}((e_1, e_2)) \rightsquigarrow^* \mathit{emd}((e_1', e_2))$.
(2) $e_2 \rightsquigarrow e_2'$. Similar to the subcase above.

$\pi_i(e_0)$: there are two ways to reduce e:
(1) $e_0 \rightsquigarrow e_0'$. By induction, $\mathit{emd}(e_0) \rightsquigarrow^* \mathit{emd}(e_0')$. Then we have $\pi_i(\mathit{emd}(e_0)) \rightsquigarrow^* \pi_i(\mathit{emd}(e_0'))$, that is, $\mathit{emd}(\pi_i(e_0)) \rightsquigarrow^* \mathit{emd}(\pi_i(e_0'))$.
(2) $e_0 = (v_1, v_2)$ and $e \rightsquigarrow v_i$. According to Lemma B.26, $\mathit{emd}((v_1, v_2)) \in \mathscr{V}$. Moreover, $\mathit{emd}((v_1, v_2)) = (\mathit{emd}(v_1), \mathit{emd}(v_2))$. Therefore, $\pi_i(\mathit{emd}(v_1), \mathit{emd}(v_2)) \rightsquigarrow \mathit{emd}(v_i)$, which is the same as $\mathit{emd}(\pi_i(v_1, v_2)) \rightsquigarrow \mathit{emd}(v_i)$.

$e_1 e_2$: there are three ways to reduce e:
(1) $e_1 \rightsquigarrow e_1'$. Similar to the case of (e_1, e_2).
(2) $e_2 \rightsquigarrow e_2'$. Similar to the case of (e_1, e_2).
(3) $e_1 = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$, $e_2 = v_2$ and $e_1 e_2 \rightsquigarrow (e_0@[\sigma_j]_{j\in P})\{v_2/x\}$, where $P = \{j \in J \mid \exists i \in I.\ \vdash v_2 : t_i\sigma_j\}$. According to Lemma B.27, we have $\vdash v_2 : t_i\sigma_j \iff \vdash \mathit{emd}(v_2) : t_i\sigma_j$, thus we have

$$\{j \in J \mid \exists i \in I.\ \vdash \mathit{emd}(v_2) : t_i\sigma_j\} = \{j \in J \mid \exists i \in I.\ \vdash v_2 : t_i\sigma_j\}.$$

Therefore, $\mathit{emd}(e_1)\mathit{emd}(v_2) \rightsquigarrow \mathit{emd}(e_0@[\sigma_j]_{j\in P})\{\mathit{emd}(v_2)/x\}$. Moreover, by lemma B.28, we can get

$$\mathit{emd}(e_0@[\sigma_j]_{j\in P})\{\mathit{emd}(v_2)/x\} = \mathit{emd}(e_0@[\sigma_j]_{j\in P}\{v_2/x\}),$$

which proves this case.

$\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$: It cannot be reduced. Thus the result follows.

$e_0 \in t\,?\,e_1 : e_2$: there are three ways to reduce e:
(1) $e_i \rightsquigarrow e_i'$. Similar to the case of (e_1, e_2).
(2) $e_0 = v_0$, $\vdash v_0 : t$ and $e \rightsquigarrow e_1$. According to Lemmas B.26 and B.27, $\mathit{emd}(v_0) \in \mathscr{V}$ and $\vdash \mathit{emd}(v_0) : t$. So we have $\mathit{emd}(v_0) \in t\,?\,\mathit{emd}(e_1) : \mathit{emd}(e_2) \rightsquigarrow \mathit{emd}(e_1)$.
(3) $e_0 = v_0$, $\nvdash v_0 : t$ and $e \rightsquigarrow e_2$. According to Lemmas B.26 and B.27, $\mathit{emd}(v_0) \in \mathscr{V}$ and $\nvdash \mathit{emd}(v_0) : t$. Therefore, $\mathit{emd}(v_0) \in t\,?\,\mathit{emd}(e_1) : \mathit{emd}(e_2) \rightsquigarrow \mathit{emd}(e_2)$.

$e_0[\sigma_j]_{j\in J}$: $e \rightsquigarrow e_0@[\sigma_j]_{j\in J}$. By Definition B.25, $\mathit{emd}(e_0[\sigma_j]_{j\in J}) = \mathit{emd}(e_0@[\sigma_j]_{j\in J})$. Therefore, the result follows.

□

Although the embedding preserves the reduction, it does not indicate that an expression and its embedding reduce to the same value. This is because that there may be some subterms of the form $e[\sigma_j]_{j\in J}$ in the body expression of an abstraction value. For example, the expression

$$(\lambda^{\mathtt{Int}\to\mathtt{Int}\to\mathtt{Int}} z.\lambda^{\mathtt{Int}\to\mathtt{Int}} y.((\lambda^{\alpha\to\alpha} x.x)[\{\mathtt{Int}/\alpha\}]42))3$$

reduces to

$$\lambda^{\mathtt{Int}\to\mathtt{Int}} y.((\lambda^{\alpha\to\alpha} x.x)[\{\mathtt{Int}/\alpha\}]42),$$

while its embedding reduces to

$$\lambda^{\mathtt{Int}\to\mathtt{Int}} y.((\lambda^{\alpha\to\alpha}_{[\{\mathtt{Int}/\alpha\}]} x.x)42).$$

However, the embedding of the value returned by an expression is the value returned by the embedding of the expression. For instance, consider the example above again:

$$\mathit{emd}(\lambda^{\mathtt{Int}\to\mathtt{Int}} y.((\lambda^{\alpha\to\alpha} x.x)[\{\mathtt{Int}/\alpha\}]42)) = \lambda^{\mathtt{Int}\to\mathtt{Int}} y.((\lambda^{\alpha\to\alpha}_{[\{\mathtt{Int}/\alpha\}]} x.x)42)$$

Next, we want to prove an inversion of Lemma B.29, that is, if the embedding $\mathit{emd}(e)$ of an expression e reduces to e', then there exists e'' such that its embedding is e' and e reduces to e''. Prior to that we prove two auxiliary lemmas: the inversions for values and for relabeled expressions.

Lemma B.30. *Let $e \in \mathscr{E}$ an expression. If $\mathit{emd}(e) \in \mathscr{V}$, then there exists a value $v \in \mathscr{V}$ such that $e \rightsquigarrow^*_{(\mathit{Rinst})} v$ and $\mathit{emd}(e) = \mathit{emd}(v)$. More specifically,*

(1) *if* $\mathit{emd}(e) = c$, *then* $e \rightsquigarrow^*_{(\mathit{Rinst})} c$ *and* $\mathit{emd}(e) = c$.
(2) *if* $\mathit{emd}(e) = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$, *then there exists* e'_0 *such that* $e \rightsquigarrow^*_{(\mathit{Rinst})} \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e'_0$ *and* $\mathit{emd}(e'_0) = e_0$.
(3) *if* $\mathit{emd}(e) = (v_1, v_2)$, *then there exist* v_1, v_2 *such that* $e \rightsquigarrow^*_{(\mathit{Rinst})} (v'_1, v'_2)$ *and* $\mathit{emd}(v'_i) = v_i$.

Proof. By induction and case analysis on $\mathit{emd}(e)$.

$\underline{c}$**:** according to Definition B.25, e should be the form of $c[\sigma_{j_1}]_{j_1\in J_1} \dots [\sigma_{j_n}]_{j_n\in J_n}$, where $n \geq 0$. Clearly, we have $e \rightsquigarrow^*_{(\mathit{Rinst})} c$ and $\mathit{emd}(e) = c$.

$\underline{\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0}$**:** according to Definition B.25, e should be the form of

$$(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_{j_0}]_{j_0\in J_0}} x.e'_0)[\sigma_{j_1}]_{j_1\in J_1} \dots [\sigma_{j_n}]_{j_n\in J_n}$$

where $\mathit{emd}(e'_0) = e_0$, $[\sigma_{j_n}]_{j_n\in J_n} \circ \dots \circ [\sigma_{j_1}]_{j_1\in J_1} \circ [\sigma_{j_0}]_{j_0\in J_0} = [\sigma_j]_{j\in J}$, and $n \geq 0$. Moreover, it is clear that $e \rightsquigarrow^*_{(\mathit{Rinst})} \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e'_0$. Let $v = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e'_0$ and the result follows.

$\underline{(v_1, v_2)}$**:** according to Definition B.25, e should be the form of $(e_1, e_2)[\sigma_{j_1}]_{j_1\in J_1} \dots [\sigma_{j_n}]_{j_n\in J_n}$, where $\mathit{emd}(e_i@[\sigma_{j_1}]_{j_1\in J_1}@\dots@[\sigma_{j_n}]_{j_n\in J_n}) = v_i$ and $n \geq 0$. Moreover, it is easy to get that

$$e \rightsquigarrow^*_{(\mathit{Rinst})} (e_1@[\sigma_{j_1}]_{j_1\in J_1}@\dots@[\sigma_{j_n}]_{j_n\in J_n}, e_2@[\sigma_{j_1}]_{j_1\in J_1}@\dots@[\sigma_{j_n}]_{j_n\in J_n})$$

By induction on v_i, there exists v'_i such that $e_i@[\sigma_{j_1}]_{j_1\in J_1}@\dots@[\sigma_{j_n}]_{j_n\in J_n} \rightsquigarrow^*_{(\mathit{Rinst})} v'_i$ and $\mathit{emd}(e_i@[\sigma_{j_1}]_{j_1\in J_1}@\dots@[\sigma_{j_n}]_{j_n\in J_n}) = \mathit{emd}(v'_i)$. Let $v = (v'_1, v'_2)$. Then we have $e \rightsquigarrow^*_{(\mathit{Rinst})} (v'_1, v'_2)$ and $\mathit{emd}(v) = (\mathit{emd}(v'_1), \mathit{emd}(v'_2)) = (v_1, v_2) = \mathit{emd}(e)$. Therefore, the result follows.

□

Lemma B.31. *Let* $e \in \mathscr{E}$ *be an expression and* $[\sigma_j]_{j\in J}$ *a set of substitutions. If* $\mathit{emd}(e@[\sigma_j]_{j\in J}) \rightsquigarrow e'$, *then there exists* e'' *such that* $e@[\sigma_j]_{j\in J} \rightsquigarrow^+ e''$ *and* $\mathit{emd}(e'') = e'$.

Proof. By induction and case analysis on e.

$\underline{c, x}$**:** straightforward.

$\underline{(e_1, e_2)}$**:** $\mathit{emd}(e@[\sigma_j]_{j\in J}) = (\mathit{emd}(e_1@[\sigma_j]_{j\in J}), \mathit{emd}(e_2@[\sigma_j]_{j\in J}))$. There are two ways to reduce $\mathit{emd}(e@[\sigma_j]_{j\in J})$:
(1) $\mathit{emd}(e_1@[\sigma_j]_{j\in J}) \rightsquigarrow e'_1$. By induction, there exists e''_1 such that $e_1@[\sigma_j]_{j\in J} \rightsquigarrow^+ e''_1$ and $\mathit{emd}(e''_1) = e'_1$. Then we have $(e_1@[\sigma_j]_{j\in J}, e_2@[\sigma_j]_{j\in J}) \rightsquigarrow^+ (e''_1, e_2@[\sigma_j]_{j\in J})$ and $\mathit{emd}((e''_1, e_2@[\sigma_j]_{j\in J})) = (e'_1, \mathit{emd}(e_2@[\sigma_j]_{j\in J}))$.
(2) $\mathit{emd}(e_2@[\sigma_j]_{j\in J}) \rightsquigarrow e'_2$. Similar to the subcase above.

$\underline{\pi_i(e_0)}$**:** $\mathit{emd}(e@[\sigma_j]_{j\in J}) = \pi_i(\mathit{emd}(e_0@[\sigma_j]_{j\in J}))$. There are two ways to reduce $\mathit{emd}(e@[\sigma_j]_{j\in J})$:
(1) $\mathit{emd}(e_0@[\sigma_j]_{j\in J}) \rightsquigarrow e'_0$. By induction, there exists e''_0 such that $e_0@[\sigma_j]_{j\in J} \rightsquigarrow^+ e''_0$ and $\mathit{emd}(e''_0) = e'_0$. Then we have $\pi_i(e_0@[\sigma_j]_{j\in J}) \rightsquigarrow^+ \pi_i(e''_0)$ and $\mathit{emd}(\pi_i(e''_0)) = \pi_i(e'_0)$.
(2) $\mathit{emd}(e_0@[\sigma_j]_{j\in J}) = (v_1, v_2)$ and $\mathit{emd}(e@[\sigma_j]_{j\in J}) \rightsquigarrow v_i$. According to Lemma B.30, there exist v'_1 and v'_2 such that $e_0@[\sigma_j]_{j\in J} \rightsquigarrow^*_{(\mathit{Rinst})} (v'_1, v'_2)$ and $\mathit{emd}(v'_i) = v_i$. Then $\pi_i(e_0@[\sigma_j]_{j\in J}) \rightsquigarrow^+ v'_i$. The result follows.

$\underline{e_1 e_2}$**:** $\mathit{emd}(e@[\sigma_j]_{j\in J}) = \mathit{emd}(e_1@[\sigma_j]_{j\in J})\mathit{emd}(e_2@[\sigma_j]_{j\in J})$. There are three possible ways to reduce $\mathit{emd}(e@[\sigma_j]_{j\in J})$:
(1) $\mathit{emd}(e_1@[\sigma_j]_{j\in J}) \rightsquigarrow e'_1$. Similar to the case of (e_1, e_2).
(2) $\mathit{emd}(e_2@[\sigma_j]_{j\in J}) \rightsquigarrow e'_2$. Similar to the case of (e_1, e_2).
(3) $\mathit{emd}(e_1@[\sigma_j]_{j\in J}) = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e_0$, $\mathit{emd}(e_2@[\sigma_j]_{j\in J}) = v_2$ and

$$\mathit{emd}(e@[\sigma_j]_{j\in J}) \rightsquigarrow (e_0@[\sigma_k]_{k\in P})\{v_2/x\},$$

where $P = \{k \in K \mid \exists i \in I.\ \vdash v_2 : t_i\sigma_k\}$. According to Lemma B.30, we have (i) there exists e'_0 such that $e_1@[\sigma_j]_{j\in J} \rightsquigarrow^*_{(\mathit{Rinst})} \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e'_0$ and $\mathit{emd}(e'_0) = e_0$; and (ii) there exists v'_2 such that $e_2@[\sigma_j]_{j\in J} \rightsquigarrow^*_{(\mathit{Rinst})} v'_2$ and $\mathit{emd}(v'_2) = v_2$. Moreover, by Lemma B.27, we get $\vdash v_2 : t_i\sigma_k \iff \vdash v'_2 : t_i\sigma_k$, thus $\{k \in K \mid \exists i \in I.\ \vdash v_2 : t_i\sigma_k\} = \{k \in K \mid \exists i \in I.\ \vdash v'_2 : t_i\sigma_k\}$. Therefore, $e@[\sigma_j]_{j\in J} \rightsquigarrow^+ (e'_0@[\sigma_k]_{k\in P})\{v'_2/x\}$. Finally, by lemma B.28, $\mathit{emd}(e'_0@[\sigma_k]_{k\in P}\{v'_2/x\}) = \mathit{emd}(e'_0)@[\sigma_k]_{k\in P}\{\mathit{emd}(v'_2)/x\} = e_0@[\sigma_k]_{k\in P}\{v_2/x\}$.

$\underline{\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e_0}$**:** It cannot be reduced. Thus the result follows.

$\underline{e_0 \in t\,?\,e_1 : e_2}$**:** $\mathit{emd}(e@[\sigma_j]_{j\in J}) = \mathit{emd}(e_0@[\sigma_j]_{j\in J}) \in t\,?\,\mathit{emd}(e_1@[\sigma_j]_{j\in J}) : \mathit{emd}(e_2@[\sigma_j]_{j\in J})$. There are three ways to reduce $\mathit{emd}(e@[\sigma_j]_{j\in J})$:
(1) $\mathit{emd}(e_i@[\sigma_j]_{j\in J}) \rightsquigarrow e'_i$. Similar to the case of (e_1, e_2).
(2) $\mathit{emd}(e_0@[\sigma_j]_{j\in J}) = v_0, \vdash v_0 : t$ and $\mathit{emd}(e@[\sigma_j]_{j\in J}) \rightsquigarrow \mathit{emd}(e_1@[\sigma_j]_{j\in J})$. According to Lemma B.30, there exists v'_0 such that $e_0@[\sigma_j]_{j\in J} \rightsquigarrow^*_{(\mathit{Rinst})} v'_0$ and $\mathit{emd}(v'_0) = v_0$. Moreover, by Lemma B.27, $\vdash v'_0 : t$. So $e@[\sigma_j]_{j\in J} \rightsquigarrow e_1@[\sigma_j]_{j\in J}$.
(3) $\mathit{emd}(e_0@[\sigma_j]_{j\in J}) = v_0, \nvdash v_0 : t$ and $\mathit{emd}(e@[\sigma_j]_{j\in J}) \rightsquigarrow \mathit{emd}(e_2@[\sigma_j]_{j\in J})$. Similar to the subcase above.

$e_0[\sigma_k]_{k\in K}$: $emd(e@[\sigma_j]_{j\in J}) = emd(e_0@([\sigma_j]_{j\in J} \circ [\sigma_k]_{k\in K}))$. By induction, the result follows.

□

Lemma B.32. *Let $e \in \mathscr{E}$ be an expression. If $emd(e) \rightsquigarrow e'$, then there exists e'' such that $e \rightsquigarrow^+ e''$ and $emd(e'') = e'$.*

Proof. By induction and case analysis on e.

c, x: straightforward.
(e_1, e_2): $emd(e) = (emd(e_1), emd(e_2))$. There are two ways to reduce $emd(e)$:
(1) $emd(e_1) \rightsquigarrow e_1'$. By induction, there exists e_1'' such that $e_1 \rightsquigarrow^+ e_1''$ and $emd(e_1'') = e_1'$. Then we have $(e_1, e_2) \rightsquigarrow^+ (e_1'', e_2)$ and $emd((e_1'', e_2)) = (e_1', emd(e_2))$.
(2) $emd(e_2) \rightsquigarrow e_2'$. Similar to the subcase above.
$\pi_i(e_0)$: $emd(e) = \pi_i(emd(e_0))$. There are two ways to reduce $emd(e)$:
(1) $emd(e_0) \rightsquigarrow e_0'$. By induction, there exists e_0'' such that $e_0 \rightsquigarrow^+ e_0''$ and $emd(e_0'') = e_0'$. Then we have $\pi_i(e_0) \rightsquigarrow^+ \pi_i(e_0'')$ and $emd(\pi_i(e_0'')) = \pi_i(e_0')$.
(2) $emd(e_0) = (v_1, v_2)$ and $emd(e) \rightsquigarrow v_i$. According to Lemma B.30, there exist v_1' and v_2' such that $e_0 \rightsquigarrow^*_{(Rinst)} (v_1', v_2')$ and $emd(v_i') = v_i$. Then $\pi_i(e_0) \rightsquigarrow^+ v_i'$. The result follows.
$e_1 e_2$: $emd(e) = emd(e_1)emd(e_2)$. There are three ways to reduce $emd(e)$:
(1) $emd(e_1) \rightsquigarrow e_1'$. Similar to the case of (e_1, e_2).
(2) $emd(e_2) \rightsquigarrow e_2'$. Similar to the case of (e_1, e_2).
(3) $emd(e_1) = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$, $emd(e_2) = v_2$ and $emd(e) \rightsquigarrow (e_0@[\sigma_j]_{j\in P})\{v_2/x\}$, where $P = \{j \in J \mid \exists i \in I.\ \vdash v_2 : t_i\sigma_j\}$. According to Lemma B.30, we have (i) there exists e_0' such that $e_1 \rightsquigarrow^*_{(Rinst)} \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_k]_{k\in K}} x.e_0'$ and $emd(e_0') = e_0$; and (ii) there exists v_2' such that $e_2 \rightsquigarrow^*_{(Rinst)} v_2'$ and $emd(v_2') = v_2$. Moreover, by Lemma B.27, we get $\vdash v_2 : t_i\sigma_j \iff \vdash v_2' : t_i\sigma_j$, thus $\{j \in J \mid \exists i \in I.\ \vdash v_2 : t_i\sigma_j\} = \{j \in J \mid \exists i \in I.\ \vdash v_2' : t_i\sigma_j\}$. Therefore, $e \rightsquigarrow^+ (e_0'@[\sigma_j]_{j\in P})\{v_2'/x\}$. Finally, by lemma B.28, $emd(e_0'@[\sigma_j]_{j\in P}\{v_2'/x\}) = emd(e_0')@[\sigma_j]_{j\in P}\{emd(v_2')/x\} = e_0@[\sigma_j]_{j\in P}\{v_2/x\}$.
$\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e_0$: It cannot be reduced. Thus the result follows.
$e_0 \in t\ ?\ e_1 : e_2$: $emd(e) = emd(e_0) \in t\ ?\ emd(e_1) : emd(e_2)$. There are three ways to reduce $emd(e)$:
(1) $emd(e_i) \rightsquigarrow e_i'$. Similar to the case of (e_1, e_2).
(2) $emd(e_0) = v_0, \vdash v_0 : t$ and $emd(e) \rightsquigarrow emd(e_1)$. According to Lemma B.30, there exists v_0' such that $e_0 \rightsquigarrow^*_{(Rinst)} v_0'$ and $emd(v_0') = v_0$. Moreover, by Lemma B.27, $\vdash v_0' : t$. So $e \rightsquigarrow e_1$.
(3) $emd(e_0) = v_0, \nvdash v_0 : t$ and $emd(e) \rightsquigarrow emd(e_2)$. Similar to the subcase above.
$e_0[\sigma_j]_{j\in J}$: $emd(e_0[\sigma_j]_{j\in J}) = emd(e_0@[\sigma_j]_{j\in J})$ and $e_0[\sigma_j]_{j\in J} \rightsquigarrow e_0@[\sigma_j]_{j\in J}$. By Lemma B.31, the result follows.

□

Thus we have the following theorem

Theorem B.33. *Let $e \in \mathscr{E}$ be an expression.*

(1) *if $e \rightsquigarrow^* v$, then $emd(e) \rightsquigarrow^* emd(v)$.*
(2) *if $emd(e) \rightsquigarrow^* v$, then there exists $v' \in \mathscr{V}$ such that $e \rightsquigarrow^* v'$ and $emd(v') = v$.*

Proof. (1): By induction on the reduction and by Lemma B.29.
(2): By induction on the reduction and by Lemma B.32.

□

In addition, it is easy to prove that the subcalculus $\mathscr{E}_N$ is closed under the reduction rules, and we can safely disregard (*Rinst*) since it cannot be applied. Then the normalized calculus also possess, for example, the soundness property.

C. Algorithmic Type Checking

The typing rules provided in Section A.3 are not syntax-directed because of the presence of the subsumption rule. In this section we present an equivalent type system with syntax-directed rules. In order to define it we consider the rules of Section A.3. First, we merge the rules (*inst*) and (*inter*) into one rule (since we prove that intersection is interesting only to merge different instances of a same type), and then we consider where subsumption is used and whether it can be postponed by moving it down the derivation tree.

C.1 Merging Intersection and Instantiation

Intersection is used to merge different types derived for the same term. In this calculus, we can derive different types for a term because of either subsumption or instantiation. However, the intersection of different super-types can be obtained by subsumption itself (if $t \leq t_1$ and $t \leq t_2$, then $t \leq t_1 \wedge t_2$), so intersection is really useful only to merge different instances of a same type, as we can see with rule (*inter*) in Figure 3. Note that all the subjects in the premise of (*inter*) share the same structure $e[\sigma]$, and the typing derivations of these terms must end with either (*inst*) or (*subsum*). We show that we can in fact postpone the uses of (*subsum*) after (*inter*), and we can therefore merge the rules (*inst*) and (*inter*) into one rule (*instinter*) as follows:

$$\frac{\Delta \fatsemi \Gamma \vdash e : t \qquad \forall j \in J.\ \sigma_j \sharp \Delta \qquad |J| > 0}{\Delta \fatsemi \Gamma \vdash e[\sigma_j]_{j \in J} : \bigwedge_{j \in J} t\sigma_j}\ (\textit{instinter})$$

Let $\Delta \fatsemi \Gamma \vdash_m e : t$ denote the typing judgments derivable in the type system with the typing rule (*instinter*) but not (*inst*) and (*inter*). The following theorem proves that the type system $\vdash_m$ (m stands for "merged") is equivalent to the original one $\vdash$.

Theorem C.1. *Let e be an expression. Then $\Delta \fatsemi \Gamma \vdash_m e : t \iff \Delta \fatsemi \Gamma \vdash e : t$.*

Proof. $\Rightarrow$**:** It is clear that (*inst*) is a special case of (*instinter*) where $|J| = 1$. We simulate each instance of (*instinter*) where $|J| > 1$ by using several instances of (*inst*) followed by one instance of (*inter*). In detail, consider the following derivation

$$\dfrac{\dfrac{\dfrac{\vdots}{\Delta' \fatsemi \Gamma' \vdash e' : t'} \qquad \sigma_j \sharp \Delta'}{\Delta' \fatsemi \Gamma' \vdash e'[\sigma_j]_{j \in J} : \bigwedge_{j \in J} t'\sigma_j}\ (\textit{instinter})}{\dfrac{\ldots \qquad \vdots \qquad \ldots}{\Delta \fatsemi \Gamma \vdash e : t}}$$

We can rewrite this derivation as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Delta' \fatsemi \Gamma' \vdash e' : t'} \qquad \sigma_1 \sharp \Delta'}{\Delta' \fatsemi \Gamma' \vdash e'[\sigma_1] : t'\sigma_1}\ (\textit{inst}) \quad \ldots \quad \dfrac{\dfrac{\vdots}{\Delta' \fatsemi \Gamma' \vdash e' : t'} \qquad \sigma_{|J|} \sharp \Delta'}{\Delta' \fatsemi \Gamma' \vdash e'[\sigma_{|J|}] : t'\sigma_{|J|}}\ (\textit{inst})}{\Delta' \fatsemi \Gamma' \vdash e'[\sigma_j]_{j \in J} : \bigwedge_{j \in J} t'\sigma_j}\ (\textit{inter})}{\dfrac{\ldots \qquad \vdots \qquad \ldots}{\Delta \fatsemi \Gamma \vdash e : t}}$$

$\Leftarrow$**:** The proof proceeds by induction and case analysis on the structure of e. For each case we use an auxiliary internal induction on the typing derivation. We label **E** the main (external) induction and **I** the internal induction in what follows.

$\underline{e = c}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*const*) or (*subsum*). If the typing derivation ends with (*const*), the result follows straightforward.
Otherwise, the typing derivation ends with an instance of (*subsum*):

$$\frac{\dfrac{\vdots}{\Delta \fatsemi \Gamma \vdash e : s} \qquad s \leq t}{\Delta \fatsemi \Gamma \vdash e : t}\ (\textit{subsum})$$

Then by **I**-induction, we have $\Delta \fatsemi \Gamma \vdash_m e : s$. Since $s \leq t$, by subsumption, we get $\Delta \fatsemi \Gamma \vdash_m e : t$.

$\underline{e = x}$**:** similar to the case of $e = c$.

$\underline{e = (e_1, e_2)}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*pair*) or (*subsum*). Assume that the typing derivation ends with (*pair*):

$$\frac{\dfrac{\vdots}{\Delta \fatsemi \Gamma \vdash e_1 : t_1} \qquad \dfrac{\vdots}{\Delta \fatsemi \Gamma \vdash e_2 : t_2}}{\Delta \fatsemi \Gamma \vdash (e_1, e_2) : t_1 \times t_2}\ (\textit{pair})$$

By **E**-induction, we have $\Delta \fatsemi \Gamma \vdash_m e_i : t_i$. Then the rule (*pair*) gives us $\Delta \fatsemi \Gamma \vdash_m (e_1, e_2) : t_1 \times t_2$. Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = \pi_i(e')}$**:** the typing derivation $\Delta \fatsemi \Gamma \vdash e : t$ should end with either (*proj*) or (*subsum*). Assume that the typing derivation ends with (*proj*):

$$\frac{\dfrac{\vdots}{\Delta \fatsemi \Gamma \vdash e' : (t_1 \times t_2)}}{\Delta \fatsemi \Gamma \vdash \pi_i(e') : t_i}\ (\textit{proj})$$

By **E**-induction, we have $\Delta \fatsemi \Gamma \vdash_m e' : (t_1 \times t_2)$. Then the rule (*proj*) gives us $\Delta \fatsemi \Gamma \vdash_m \pi_i(e') : t_i$. Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e_1 e_2}$**:** the typing derivation $\Delta \wr \Gamma \vdash e : t$ should end with either (*appl*) or (*subsum*). Assume that the typing derivation ends with (*appl*):

$$\frac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e_1 : t_1 \to t_2} \quad \dfrac{\dots}{\Delta \wr \Gamma \vdash e_2 : t_1}}{\Delta \wr \Gamma \vdash e_1 e_2 : t_2}\ (appl)$$

By **E**-induction, we have $\Delta \wr \Gamma \vdash_m e_1 : t_1 \to t_2$ and $\Delta \wr \Gamma \vdash_m e_2 : t_1$. Then the rule (*appl*) gives us $\Delta \wr \Gamma \vdash_m e_1 e_2 : t_2$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e'}$**:** the typing derivation $\Delta \wr \Gamma \vdash e : t$ should end with either (*abstr*) or (*subsum*). Assume that the typing derivation ends with (*abstr*):

$$\frac{\begin{array}{l}\forall i \in I, j \in J.\ \dfrac{\dots}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e'@[\sigma_j] : s_i\sigma_j} \\ \Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j))\end{array}}{\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)}\ (abstr)$$

By **E**-induction, for all $i \in I$ and $j \in J$, we have $\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash_m e'@[\sigma_j] : s_i\sigma_j$. Then the rule (*abstr*) gives us $\Delta \wr \Gamma \vdash_m \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e' : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e' \in t\,?\,e_1 : e_2}$**:** the typing derivation $\Delta \wr \Gamma \vdash e : t$ should end with either (*case*) or (*subsum*). Assume that the typing derivation ends with (*case*):

$$\frac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e' : t'} \quad \begin{cases} t' \not\leq \neg t \quad \Rightarrow \quad \dfrac{\dots}{\Delta \wr \Gamma \vdash e_1 : s} \\ t' \not\leq t \quad \Rightarrow \quad \dfrac{\dots}{\Delta \wr \Gamma \vdash e_2 : s} \end{cases}}{\Delta \wr \Gamma \vdash (e' \in t\,?\,e_1 : e_2) : s}\ (case)$$

By **E**-induction, we have $\Delta \wr \Gamma \vdash_m e' : t'$ and $\Delta \wr \Gamma \vdash_m e_i : s$ (for i such that e_i has been effectively type-checked). Then the rule (*case*) gives us $\Delta \wr \Gamma_m \vdash (e' \in t\,?\,e_1 : e_2) : s$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e'[\sigma]}$**:** the typing derivation $\Delta \wr \Gamma \vdash e : t$ should end with either (*inst*) or (*subsum*). Assume that the typing derivation ends with (*inst*):

$$\frac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e' : t} \quad \sigma \,\sharp\, \Delta}{\Delta \wr \Gamma \vdash e'[\sigma] : t\sigma}\ (inst)$$

By **E**-induction, we have $\Delta \wr \Gamma \vdash_m e' : t$. Since $\sigma \,\sharp\, \Delta$, applying (*instinter*) where $|J| = 1$, we get $\Delta \wr \Gamma \vdash_m e'[\sigma] : t\sigma$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

$\underline{e = e'[\sigma_j]_{j\in J}}$**:** the typing derivation $\Delta \wr \Gamma \vdash e : t$ should end with either (*inter*) or (*subsum*). Assume that the typing derivation ends with (*inter*):

$$\frac{\forall j \in J.\ \dfrac{\dots}{\Delta \wr \Gamma \vdash e'[\sigma_j] : t_j} \quad |J| > 1}{\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j}\ (inter)$$

As an intermediary result, we first prove that the derivation can be rewritten as

$$\frac{\forall j \in J.\ \dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e' : s} \quad \sigma_j \,\sharp\, \Delta}{\Delta \wr \Gamma \vdash e'[\sigma_j] : s\sigma_j}\ (inst)}{\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j}\ (inter) \qquad \bigwedge_{j\in J} s\sigma_j \leq \bigwedge_{j\in J} t_j}{\Delta \wr \Gamma \vdash e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t_j}\ (subsum)$$

We proceed by induction on the original derivation. It is clear that each sub-derivation $\Delta \wr \Gamma \vdash e'[\sigma_j] : t_j$ ends with either (*inst*) or (*subsum*). If all the sub-derivations end with an instance of (*inst*), then for all $j \in J$, we have

$$\frac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e' : s_j} \quad \sigma_j \,\sharp\, \Delta}{\Delta \wr \Gamma \vdash e'[\sigma_j] : s_j\sigma_j}\ (inst)$$

By Lemma B.2, we have $\Delta \wr \Gamma \vdash e' : \bigwedge_{j\in J} s_j$. Let $s = \bigwedge_{j\in J} s_j$. Then by (*inst*), we get $\Delta \wr \Gamma \vdash e'[\sigma_j] : s\sigma_j$. Finally, by (*inter*) and (*subsum*), the intermediary result holds. Otherwise, there is at least one of the sub-derivations ends with an instance of (*subsum*), the intermediary result

also hold by induction.
Now that the intermediary result is proved, we go back to the proof of the lemma. By **E**-induction on e' (*i.e.*, $\Delta \wr \Gamma \vdash e' : s$), we have $\Delta \wr \Gamma \vdash_m e' : s$. Since $\sigma_j \not\sharp \Delta$, applying (*instinter*), we get $\Delta \wr \Gamma \vdash_m e'[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$. Finally, by subsumption, we get $\Delta \wr \Gamma \vdash_m e'[\sigma]_{j\in J} : \bigwedge_{j\in J} t_j$.
Otherwise, the typing derivation ends with an instance of (*subsum*), similar to the case of $e = c$, the result follows by **I**-induction.

□

From now on we will use $\vdash$ to denote $\vdash_m$, that is the system with the merged rule.

C.2 Algorithmic Typing Rules

In this section, we analyze the typing derivations produced by the rules of Section A.3 to see where subsumption is needed and where it can be pushed down the derivation tree. We need first some preliminary definitions and decomposition results about pair and function types to deal with the projection and application rules.

C.2.1 Pair types

A type s is a pair type if $s \leq \mathbb{1} \times \mathbb{1}$. If an expression e is typeable with a pair type s, we want to compute from s a valid type for $\pi_i(e)$. In $\mathbb{C}$Duce, a pair type s is a finite union of product types, which can be decomposed into a finite set of pairs of types, denoted as $\boldsymbol{\pi}(s)$. For example, we decompose $s = (t_1 \times t_2) \vee (s_1 \times s_2)$ as $\boldsymbol{\pi}(s) = \{(t_1, t_2), (s_1, s_2)\}$. We can then compute easily a type $\boldsymbol{\pi}_i(s)$ for $\pi_i(e)$ as $\boldsymbol{\pi}_i(s) = t_i \vee s_i$ (we used boldface symbols to distinguish these type operators from the projections used in expressions). In the calculus considered here, the situation becomes more complex because of type variables, especially top level ones. Let s be a pair type that contains a top-level variable α. Since $\alpha \not\leq \mathbb{1} \times \mathbb{1}$ and $s \leq \mathbb{1} \times \mathbb{1}$, then it is not possible that $s \simeq s' \vee \alpha$. In other terms the top-level variable cannot appear alone in a union: it must occur intersected with some product type so that it does not "overtake" the $\mathbb{1} \times \mathbb{1}$ bound. Consequently, we have $s \simeq s' \wedge \alpha$ for some $s' \leq \mathbb{1} \times \mathbb{1}$. However, in a typing derivation starting from $\Delta \wr \Gamma \vdash e : s$ and ending with $\Delta \wr \Gamma \vdash \pi_i(e) : t$, there exists an intermediary step where e is assigned a type of the form $(t_1 \times t_2)$ (and that verifies $s \leq (t_1 \times t_2)$) before applying the projection rule. So it is necessary to get rid of the top-level variables of s (using subsumption) before computing the projection. The example above shows that α does not play any role since it is the s' component that will be used to subsume s to a product type. To say it otherwise, since e has type s for all possible assignment of α, then the typing derivation must hold also for $\alpha = \mathbb{1}$. In whatever way we look at it, the top-level type variables are useless and can be safely discarded when decomposing s.

Given a type t, we write $\mathsf{dnf}(t)$ for a disjunctive normal form of t, which is defined in [5]. Formally, we define the decomposition of a pair type as follows:

Definition C.2. *Let τ be a disjunctive normal form such that $\tau \leq \mathbb{1} \times \mathbb{1}$. We define the* decomposition *of τ as follows:*

$$\begin{aligned}
&\boldsymbol{\pi}(\textstyle\bigvee_{i\in I} \tau_i) = \bigcup_{i\in I} \boldsymbol{\pi}(\tau_i)\\
&\boldsymbol{\pi}(\textstyle\bigwedge_{j\in P}(t_1^j \times t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \times t_2^k) \wedge \bigwedge_{\alpha\in P_\forall} \alpha \wedge \bigwedge_{\alpha'\in N_\forall} \neg\alpha') \quad (|P| > 0)\\
&\qquad = \boldsymbol{\pi}(\textstyle\bigvee_{N'\subseteq N}((\bigwedge_{j\in P} t_1^j \wedge \bigwedge_{k\in N'} \neg t_1^k) \times (\bigwedge_{j\in P} t_2^j \wedge \bigwedge_{k\in N\setminus N'} \neg t_2^k)))\\
&\boldsymbol{\pi}((t_1 \times t_2)) = \begin{cases} \{(t_1, t_2)\} & t_1 \not\simeq \mathbb{0} \text{ and } t_2 \not\simeq \mathbb{0}\\ \emptyset & \text{otherwise}\end{cases}
\end{aligned}$$

and the i-th projection as $\boldsymbol{\pi}_i(\tau) = \bigvee_{(s_1,s_2)\in\boldsymbol{\pi}(\tau)} s_i$.
For all type t such that $t \leq \mathbb{1} \times \mathbb{1}$, the decomposition of t is defined as

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(\mathsf{dnf}((\mathbb{1} \times \mathbb{1}) \wedge t))$$

and the i-th projection as $\boldsymbol{\pi}_i(t) = \bigvee_{(s_1,s_2)\in\boldsymbol{\pi}(\mathsf{dnf}((\mathbb{1}\times\mathbb{1})\wedge t))} s_i$

The decomposition of a union of pair types is the union of each decomposition. When computing the decomposition of an intersection of product types and top-level type variables, we compute all the possible distributions of the intersections over the products, and we discard the top-level variables, as discussed above. Finally, the decomposition of a product is the pair of two components, provided that both components are not empty.

We now prove that the top-level type variables can be safely eliminated in a well-founded (convex) model with infinite support (see [5] for the definitions of model, convexity and infinite support).

Lemma C.3. *Let $\leq$ be the subtyping relation induced by a well-founded (convex) model with infinite support. Then*

$$\bigwedge_{p\in P} (t_p \times s_p) \wedge \alpha \leq \bigvee_{n\in N} (t_n \times s_n) \iff \bigwedge_{p\in P} (t_p \times s_p) \leq \bigvee_{n\in N} (t_n \times s_n)$$

Proof. The result trivially holds if $\bigwedge_{p\in P} (t_p \times s_p) = \mathbb{0}$ or $|P| = 0$ (*ie*, $\bigwedge_{p\in P} (t_p \times s_p) = \mathbb{1}$). Let us examine the remaining cases:

$\Leftarrow$**:** straightforward.

$\Rightarrow$**:** Assume that $\bigwedge_{p\in P}(t_p\times s_p)\not\leq\bigvee_{n\in N}(t_n\times s_n)$. Let τ be the type $\bigwedge_{p\in P}(t_p\times s_p)\wedge\bigwedge_{n\in N}\neg(t_n\times s_n)$. Then there exists an assignment η such that $[\![\tau]\!]\eta\neq\emptyset$ (see the subtyping relation defined in [5]). Using the procedure explore_pos defined in the proof of Lemma 3.23 in [5], we can generate an element d belonging to $[\![\tau]\!]\eta$.[7] The procedure explore_pos also generates an assignment η_0 for the type variables in $\mathsf{var}(\tau)$. We define η' such that $\eta'(\alpha)=\eta_0(\alpha)\cup\{d\}$, $\eta'(\neg\alpha)=\eta_0(\neg\alpha)\setminus\{d\}$, and $\eta'=\eta_0$ otherwise. Then we have $[\![\tau\wedge\alpha]\!]\eta'\neq\emptyset$, which implies $\bigwedge_{p\in P}(t_p\times s_p)\wedge\alpha\not\leq\bigvee_{n\in N}(t_n\times s_n)$. The result follows by the contrapositive.

□

The decomposition of pair types defined above has the following properties:

Lemma C.4. *Let $\leq$ be the subtyping relation induced by a well-founded (convex) model with infinite support and t a type such that $t\leq\mathbb{1}\times\mathbb{1}$. Then*

(1) *For all $(t_1,t_2)\in\boldsymbol{\pi}(t)$, we have $t_1\not\simeq\mathbb{0}$ and $t_2\not\simeq\mathbb{0}$*
(2) *For all s_1,s_2, we have $t\leq(s_1\times s_2)\iff\bigvee_{(t_1,t_2)\in\boldsymbol{\pi}(t)}(t_1\times t_2)\leq(s_1\times s_2)$*

Proof.

(1)**:** straightforward.
(2)**:** Since $t\leq\mathbb{1}\times\mathbb{1}$, we have

$$t\simeq\bigvee_{(P,N)\in\mathsf{dnf}(t)}((\mathbb{1}\times\mathbb{1})\wedge\bigwedge_{j\in P\setminus\mathscr{V}}(t_1^j\times t_2^j)\wedge\bigwedge_{k\in N\setminus\mathscr{V}}\neg(t_1^k\times t_2^k)\wedge\bigwedge_{\alpha\in P\cap\mathscr{V}}\alpha\wedge\bigwedge_{\alpha'\in N\cap\mathscr{V}}\neg\alpha')$$

If $t\simeq\mathbb{0}$, then $\boldsymbol{\pi}(t)=\emptyset$, and the result holds. Assume that $t\not\simeq\mathbb{0}$, $|P|>0$ and each summand of $\mathsf{dnf}(t)$ is not equivalent to $\mathbb{0}$ as well. Let $\ulcorner t\urcorner$ denote the type $\bigvee_{(P,N)\in\mathsf{dnf}(t)}(\bigwedge_{j\in P\setminus\mathscr{V}}(t_1^j\times t_2^j)\wedge\bigwedge_{k\in N\setminus\mathscr{V}}\neg(t_1^k\times t_2^k))$. Using the set-theoretic interpretation of types we have that $\ulcorner t\urcorner$ is equivalent to

$$\bigvee_{(P,N)\in\mathsf{dnf}(t)}\Big(\bigvee_{N'\subseteq N\setminus\mathscr{V}}\Big(\Big(\bigwedge_{j\in P\setminus\mathscr{V}}t_1^j\wedge\bigwedge_{k\in N'}\neg t_1^k\Big)\times\Big(\bigwedge_{j\in P\setminus\mathscr{V}}t_2^j\wedge\bigwedge_{k\in(N\setminus\mathscr{V})\setminus N'}\neg t_2^k\Big)\Big)\Big)$$

This means that, $\ulcorner t\urcorner$ is a equivalent to a union of product types. Let us rewrite this union more explicitly, that is, $\ulcorner t\urcorner\simeq\bigvee_{i\in I}(t_1^i\times t_2^i)$ obtained as follows

$$\bigvee_{(P,N)\in\mathsf{dnf}(t)}\Big(\bigvee_{N'\subseteq N\setminus\mathscr{V}}\Big(\overbrace{\Big(\bigwedge_{j\in P\setminus\mathscr{V}}t_1^j\wedge\bigwedge_{k\in N'}\neg t_1^k\Big)}^{t_1^i}\times\overbrace{\Big(\bigwedge_{j\in P\setminus\mathscr{V}}t_2^j\wedge\bigwedge_{k\in(N\setminus\mathscr{V})\setminus N'}\neg t_2^k\Big)}^{t_2^i}\Big)\Big)$$

We have

$$\boldsymbol{\pi}(t)=\{(t_1^i,t_2^i)\mid i\in I\text{ and }t_1^i\not\simeq\mathbb{0}\text{ and }t_2^i\not\simeq\mathbb{0}\}$$

Finally, for all pair of types s_1 and s_2, we have

$$\begin{aligned}
&t\leq(s_1\times s_2)\\
\iff&\bigvee_{(P,N)\in\mathsf{dnf}(t)}\Big(\bigwedge_{j\in P\setminus\mathscr{V}}(t_1^j\times t_2^j)\wedge\bigwedge_{k\in N\setminus\mathscr{V}}\neg(t_1^k\times t_2^k)\wedge\bigwedge_{\alpha\in P\cap\mathscr{V}}\alpha\wedge\bigwedge_{\alpha'\in N\cap\mathscr{V}}\neg\alpha'\Big)\leq(s_1\times s_2)\\
\iff&\bigvee_{(P,N)\in\mathsf{dnf}(t)}\Big(\bigwedge_{j\in P\setminus\mathscr{V}}(t_1^j\times t_2^j)\wedge\bigwedge_{k\in N\setminus\mathscr{V}}\neg(t_1^k\times t_2^k)\wedge\bigwedge_{\alpha\in P\cap\mathscr{V}}\alpha\wedge\bigwedge_{\alpha'\in N\cap\mathscr{V}}\neg\alpha'\wedge\neg(s_1\times s_2)\Big)\leq\mathbb{0}\\
\iff&\bigvee_{(P,N)\in\mathsf{dnf}(t)}\Big(\bigwedge_{j\in P\setminus\mathscr{V}}(t_1^j\times t_2^j)\wedge\bigwedge_{k\in N\setminus\mathscr{V}}\neg(t_1^k\times t_2^k)\wedge\neg(s_1\times s_2)\Big)\leq\mathbb{0}\quad\text{(Lemma C.3)}\\
\iff&\bigvee_{(P,N)\in\mathsf{dnf}(t)}\Big(\bigwedge_{j\in P\setminus\mathscr{V}}(t_1^j\times t_2^j)\wedge\bigwedge_{k\in N\setminus\mathscr{V}}\neg(t_1^k\times t_2^k)\Big)\leq(s_1\times s_2)\\
\iff&\ulcorner t\urcorner\leq(s_1\times s_2)\\
\iff&\bigvee_{(t_1,t_2)\in\boldsymbol{\pi}(t)}(t_1\times t_2)\leq(s_1\times s_2)
\end{aligned}$$

□

Lemma C.5. *Let s be a type such that $s\leq(t_1\times t_2)$. Then*

(1) $s\leq(\boldsymbol{\pi}_1(s)\times\boldsymbol{\pi}_2(s))$
(2) $\boldsymbol{\pi}_i(s)\leq t_i$

[7] Strictly speaking, the procedure explore_pos of Lemma 3.23 in [5] supposes τ contains only finite product types, but it can be extended to infinite product types by Lemma 3.24 in [5]

Proof. (1): according to the proof of Lemma C.4, $\bigvee_{(s_1,s_2)\in\boldsymbol{\pi}(s)}(s_1\times s_2)$ is equivalent to the type obtained from s by ignoring all the top-level type variables. Then it is trivial that $s \leq \bigvee_{(s_1,s_2)\in\boldsymbol{\pi}(s)}(s_1\times s_2)$ and then $s \leq (\boldsymbol{\pi}_1(s)\times\boldsymbol{\pi}_2(s))$.

(2): since $s \leq (t_1\times t_2)$, according to Lemma C.4, we have $\bigvee_{(s_1,s_2)\in\boldsymbol{\pi}(s)}(s_1\times s_2) \leq (t_1\times t_2)$. So for all $(s_1,s_2)\in\boldsymbol{\pi}(s)$, we have $(s_1\times s_2)\leq(t_1\times t_2)$. Moreover, as s_i is not empty, we have $s_i \leq t_i$. Therefore, $\boldsymbol{\pi}_i(s)\leq t_i$.

□

Lemma C.6. *Let t and s be two types such that $t\leq\mathbb{1}\times\mathbb{1}$ and $s\leq\mathbb{1}\times\mathbb{1}$. Then $\boldsymbol{\pi}_i(t\wedge s)\leq\boldsymbol{\pi}_i(t)\wedge\boldsymbol{\pi}_i(s)$.*

Proof. Let $t=\bigvee_{j_1\in J_1}\tau_{j_1}$ and $s=\bigvee_{j_2\in J_2}\tau_{j_2}$ such that

$$\tau_j = (t_j^1\times t_j^2)\wedge\bigwedge_{\alpha\in P_j}\alpha\wedge\bigwedge_{\alpha'\in N_j}\neg\alpha'$$

and $\tau_j\not\simeq\mathbb{0}$ for all $j\in J_1\cup J_2$. Then we have $t\wedge s=\bigvee_{j_1\in J_1,j_2\in J_2}\tau_{j_1}\wedge\tau_{j_2}$. Let $j_1\in J_1$ and $j_2\in J_2$. If $\tau_{j_1}\wedge\tau_{j_2}\simeq\mathbb{0}$, we have $\boldsymbol{\pi}_i(\tau_{j_1}\wedge\tau_{j_2})=\mathbb{0}$. Otherwise, $\boldsymbol{\pi}_i(\tau_{j_1}\wedge\tau_{j_2})=t^i_{j_1}\wedge t^i_{j_2}=\boldsymbol{\pi}_i(\tau_{j_1})\wedge\boldsymbol{\pi}_i(\tau_{j_2})$. For both cases, we have $\boldsymbol{\pi}_i(\tau_{j_1}\wedge\tau_{j_2})\leq\boldsymbol{\pi}_i(\tau_{j_1})\wedge\boldsymbol{\pi}_i(\tau_{j_2})$. Therefore

$$\begin{array}{rl}\boldsymbol{\pi}_i(t\wedge s)\simeq & \bigvee_{j_1\in J_1,j_2\in J_2}\boldsymbol{\pi}_i(\tau_{j_1}\wedge\tau_{j_2})\\ \leq & \bigvee_{j_1\in J_1,j_2\in J_2}(\boldsymbol{\pi}_i(\tau_{j_1})\wedge\boldsymbol{\pi}_i(\tau_{j_2}))\\ \simeq & (\bigvee_{j_1\in J_1}\boldsymbol{\pi}_i(\tau_{j_1}))\wedge(\bigvee_{j_2\in J_2}\boldsymbol{\pi}_i(\tau_{j_2}))\\ \simeq & \boldsymbol{\pi}_i(t)\wedge\boldsymbol{\pi}_i(s)\end{array}$$

□

For example, $\boldsymbol{\pi}_1((\texttt{Int}\times\texttt{Int})\wedge(\texttt{Int}\times\texttt{Bool}))=\boldsymbol{\pi}_1(\mathbb{0})=\mathbb{0}$, while $\boldsymbol{\pi}_1((\texttt{Int}\times\texttt{Int}))\wedge\boldsymbol{\pi}_1((\texttt{Int}\times\texttt{Bool}))=\texttt{Int}\wedge\texttt{Int}=\texttt{Int}$.

Lemma C.7. *Let t be a type and σ be a type substitution such that $t\leq\mathbb{1}\times\mathbb{1}$. Then $\boldsymbol{\pi}_i(t\sigma)\leq\boldsymbol{\pi}_i(t)\sigma$*

Proof. We put t into its disjunctive normal form $\bigvee_{j\in J}\tau_j$ such that

$$\tau_j = (t_j^1\times t_j^2)\wedge\bigwedge_{\alpha\in P_j}\alpha\wedge\bigwedge_{\alpha'\in N_j}\neg\alpha'$$

and $\tau_j\not\simeq\mathbb{0}$ for all $j\in J$. Then we have $t\sigma=\bigvee_{j\in J}\tau_j\sigma$. So $\boldsymbol{\pi}_i(t\sigma)=\bigvee_{j\in J}\boldsymbol{\pi}_i(\tau_j\sigma)$. Let $j\in J$. If $\tau_j\sigma\simeq\mathbb{0}$, then $\boldsymbol{\pi}_i(\tau_j\sigma)=\mathbb{0}$ and trivially $\boldsymbol{\pi}_i(\tau_j\sigma)\leq\boldsymbol{\pi}_i(\tau_j)\sigma$. Otherwise, we have $\tau_j\sigma=(t_j^1\sigma\times t_j^2\sigma)\wedge(\bigwedge_{\alpha\in P_j}\alpha\wedge\bigwedge_{\alpha'\in N_j}\neg\alpha')\sigma$. By Lemma C.6, we get $\boldsymbol{\pi}_i(\tau_j\sigma)\leq t_j^i\sigma\wedge\boldsymbol{\pi}_i((\bigwedge_{\alpha\in P_j}\alpha\wedge\bigwedge_{\alpha'\in N_j}\neg\alpha')\sigma)\leq t_j^i\sigma\simeq\boldsymbol{\pi}_i(\tau_j)\sigma$. Therefore, $\bigvee_{j\in J}\boldsymbol{\pi}_i(\tau_j\sigma)\leq\bigvee_{j\in J}\boldsymbol{\pi}_i(\tau_j)\sigma$, that is, $\boldsymbol{\pi}_i(t\sigma)\leq\boldsymbol{\pi}_i(t)\sigma$. □

For example, $\boldsymbol{\pi}_1(((\texttt{Int}\times\texttt{Int})\wedge\alpha)\{(\texttt{Int}\times\texttt{Bool})/\alpha\})=\boldsymbol{\pi}_1((\texttt{Int}\times\texttt{Int})\wedge(\texttt{Int}\times\texttt{Bool}))=\mathbb{0}$, while $(\boldsymbol{\pi}_1((\texttt{Int}\times\texttt{Int})))\{(\texttt{Int}\times\texttt{Bool})/\alpha\}=\texttt{Int}\{(\texttt{Int}\times\texttt{Bool})/\alpha\}=\texttt{Int}$.

Lemma C.8. *Let t be a type such that $t\leq\mathbb{1}\times\mathbb{1}$ and $[\sigma_k]_{k\in K}$ be a set of type substitutions. Then $\boldsymbol{\pi}_i(\bigwedge_{k\in K}t\sigma_k)\leq\bigwedge_{k\in K}\boldsymbol{\pi}_i(t)\sigma_k$*

Proof. Consequence of Lemmas C.6 and C.7. □

C.2.2 Function types

A type t is a function type if $t\leq\mathbb{0}\to\mathbb{1}$. In order to type the application of a function having a function type t, we need to determine the domain of t, that is, the set of values the function can be safely applied to. This problem has been solved for ground function types in [11]. Again, the problem becomes more complex if t contains top-level type variables. Another issue is to determine what is the result type of an application of a function type t to an argument of type s (where s belongs to the domain of t), knowing that both t and s may contain type variables.

Following the same reasoning as with pair types, if a function type t contains a top-level variable α, then $t\simeq t'\wedge\alpha$ for some function type t'. In a typing derivation for a judgment $\Delta\fatsemi\Gamma\vdash e_1\ e_2:t$ which contains $\Delta\fatsemi\Gamma\vdash e_1:t$, there exists an intermediary step where we assign a type $t_1\to t_2$ to e_1 (with $t\leq t_1\to t_2$) before using the application rule. It is therefore necessary to eliminate the top-level variables from the function type t before we can type an application. Once more, the top-level variables are useless when computing the domain of t and can be safely discarded.

Formally, we define the domain of a function type as follows:

Definition C.9 (Domain). *Let τ be a disjunctive normal form such that $\tau \leq \mathbb{0} \to \mathbb{1}$. We define* $\mathsf{dom}(\tau)$*, the* domain *of τ, as:*

$$\begin{aligned}
&\mathsf{dom}(\textstyle\bigvee_{i\in I} \tau_i) = \bigwedge_{i\in I} \mathsf{dom}(\tau_i)\\
&\mathsf{dom}(\textstyle\bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\mathscr{V}} \alpha \wedge \bigwedge_{\alpha'\in N_\mathscr{V}} \neg\alpha')\\
&\qquad = \begin{cases} \mathbb{1} & \text{if } \bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\mathscr{V}} \alpha \wedge \bigwedge_{\alpha'\in N_\mathscr{V}} \neg\alpha' \simeq \mathbb{0}\\ \bigvee_{j\in P} t_1^j & \text{otherwise}\end{cases}
\end{aligned}$$

For any type t such that $t \leq \mathbb{0} \to \mathbb{1}$, the domain of t is defined as

$$\mathsf{dom}(t) = \mathsf{dom}(\mathsf{dnf}((\mathbb{0} \to \mathbb{1}) \wedge t))$$

We also define a decomposition operator $\boldsymbol{\phi}$ that —akin to the decomposition operator $\boldsymbol{\pi}$ for product types— decomposes a function type into a finite set of pairs:

Definition C.10. *Let τ be a disjunctive normal form such that $\tau \leq \mathbb{0} \to \mathbb{1}$. We define the* decomposition *of τ as:*

$$\begin{aligned}
&\boldsymbol{\phi}(\textstyle\bigvee_{i\in I} \tau_i) = \bigcup_{i\in I} \boldsymbol{\phi}(\tau_i)\\
&\boldsymbol{\phi}(\textstyle\bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\mathscr{V}} \alpha \wedge \bigwedge_{\alpha'\in N_\mathscr{V}} \neg\alpha')\\
&\qquad = \begin{cases} \emptyset & \text{if } \bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\mathscr{V}} \alpha \wedge \bigwedge_{\alpha'\in N_\mathscr{V}} \neg\alpha' \simeq \mathbb{0}\\ \{(\bigvee_{j\in P'} t_1^j, \bigwedge_{j\in P\setminus P'} t_2^j) \mid P' \subsetneq P\} & \text{otherwise}\end{cases}
\end{aligned}$$

For any type t such that $t \leq \mathbb{0} \to \mathbb{1}$, the decomposition of $t$is defined as

$$\boldsymbol{\phi}(t) = \boldsymbol{\phi}(\mathsf{dnf}((\mathbb{0} \to \mathbb{1}) \wedge t)).$$

The set $\boldsymbol{\phi}(t)$ satisfies the following fundamental property: for every arrow type $s \to s'$, the constraint $t \leq s \to s'$ holds if and only if $s \leq \mathsf{dom}(t)$ holds and for all $(t_1, t_2) \in \boldsymbol{\phi}(t)$, either $s \leq t_1$ or $t_2 \leq s'$ hold (see Lemma C.12). As a result, the minimum type

$$t \cdot s = \min\{s' \mid t \leq s \to s'\}$$

exists, and it is defined as the union of all t_2 such that $s \not\leq t_1$ and $(t_1, t_2) \in \boldsymbol{\phi}(t)$ (see Lemma C.13). The type $t \cdot s$ is used to type the application of an expression of type t to an expression of type s.

As with pair types, in a well-founded (convex) model with infinite support, we can safely eliminate the top-level type variables.

Lemma C.11. *Let $\leq$ be the subtyping relation induced by a well-founded (convex) model with infinite support. Then*

$$\bigwedge_{p\in P} (t_p \to s_p) \wedge \alpha \leq \bigvee_{n\in N} (t_n \to s_n) \iff \bigwedge_{p\in P} (t_p \to s_p) \leq \bigvee_{n\in N} (t_n \to s_n)$$

Proof. Similar to the proof of Lemma C.3. □

Lemma C.12. *Let $\leq$ be the subtyping relation induced by a well-founded (convex) model with infinite support and t a type such that $t \leq \mathbb{0} \to \mathbb{1}$. Then*

$$\forall s_1, s_2 \,.\, (t \leq s_1 \to s_2) \iff \begin{cases} s_1 \leq \mathsf{dom}(t)\\ \forall (t_1, t_2) \in \boldsymbol{\phi}(t) \,.\, (s_1 \leq t_1) \text{ or } (t_2 \leq s_2)\end{cases}$$

Proof. Since $t \leq \mathbb{0} \to \mathbb{1}$, we have

$$t \simeq \bigvee_{(P,N)\in\mathsf{dnf}(t)} ((\mathbb{0} \to \mathbb{1}) \wedge \bigwedge_{j\in P\setminus\mathscr{V}} (t_1^j \to t_2^j) \wedge \bigwedge_{k\in N\setminus\mathscr{V}} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P\cap\mathscr{V}} \alpha \wedge \bigwedge_{\alpha'\in N\cap\mathscr{V}} \neg\alpha')$$

If $t \simeq \mathbb{0}$, then $\mathsf{dom}(t) = \mathbb{1}$, $\boldsymbol{\phi}(t) = \emptyset$, and the result holds. If $t \simeq t_1 \vee t_2$, then $t_1 \leq \mathbb{0} \to \mathbb{1}$, $t_2 \leq \mathbb{0} \to \mathbb{1}$, $\mathsf{dom}(t) = \mathsf{dom}(t_1) \wedge \mathsf{dom}(t_2)$ and $\boldsymbol{\phi}(t) = \boldsymbol{\phi}(t_1) \cup \boldsymbol{\phi}(t_2)$. So the result follows if it also holds for t_1 and t_2. Thus, without loss of generality, we can assume that t has the following form:

$$t \simeq \bigwedge_{j\in P} (t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\mathscr{V}} \alpha \wedge \bigwedge_{\alpha'\in N_\mathscr{V}} \neg\alpha'$$

where $P \neq \emptyset$ and $t \not\simeq \mathbb{0}$. Then $\mathsf{dom}(t) = \bigvee_{j\in P} t_1^j$ and $\boldsymbol{\phi}(t) = \{(\bigvee_{j\in P'} t_1^j, \bigwedge_{j\in P\setminus P'} t_2^j) \mid P' \subsetneq P\}$. For every pair of types s_1 and s_2, we have

$$\begin{array}{lll} & t \leq (s_1 \to s_2) & \\ \iff & \bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\mathcal{V}} \alpha \wedge \bigwedge_{\alpha'\in N_\mathcal{V}} \neg\alpha' \leq (s_1 \to s_2) & \\ \iff & \bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k) \leq (s_1 \to s_2) & \text{(Lemma C.11)} \\ \iff & \bigwedge_{j\in P}(t_1^j \to t_2^j) \leq s_1 \to s_2 & (\ulcorner t \urcorner \not\simeq \mathbb{0} \text{ and Lemma 3.12 in [5]}) \\ \iff & \forall P' \subseteq P.\ \left(s_1 \leq \bigvee_{j\in P'} t_1^j\right) \vee \left(P \neq P' \wedge \bigwedge_{j\in P\setminus P'} t_2^j \leq s_2\right) & \end{array}$$

where $\ulcorner t \urcorner = \bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N} \neg(t_1^k \to t_2^k)$.

□

Lemma C.13. *Let t and s be two types. If $t \leq s \to \mathbb{1}$, then $t \leq s \to s'$ has a smallest solution s', which is denoted as $t \cdot s$.*

Proof. Since $t \leq s \to \mathbb{1}$, by Lemma C.12, we have $s \leq \mathsf{dom}(t)$. Then the assertion $t \leq s \to s'$ is equivalent to:

$$\forall (t_1, t_2) \in \boldsymbol{\phi}(t).\ (s \leq t_1) \mathsf{or} (t_2 \leq s')$$

that is:

$$\left(\bigvee_{(t_1,t_2)\in\boldsymbol{\phi}(t) \text{ s.t. } (s\not\leq t_1)} t_2\right) \leq s'$$

Thus the type $\bigvee_{(t_1,t_2)\in\boldsymbol{\phi}(t) \text{ s.t. } (s\not\leq t_1)} t_2$ is a lower bound for all the solutions.

By the subtyping relation on arrows it is also a solution, so it is the smallest one. To conclude, it suffices to take it as the definition for $t \cdot s$. □

We now prove some properties of the operators $\mathsf{dom}(\_)$ and "$\_ \cdot \_$".

Lemma C.14. *Let t be a type such that $t \leq \mathbb{0} \to \mathbb{1}$ and t', s, s' be types. Then*

(1) *if $s' \leq s \leq \mathsf{dom}(t)$, then $t \cdot s' \leq t \cdot s$.*
(2) *if $t' \leq t$, $s \leq \mathsf{dom}(t')$ and $s \leq \mathsf{dom}(t)$, then $t' \cdot s \leq t \cdot s$.*

Proof. (1) Since $s' \leq s$, we have $s \to t \cdot s \leq s' \to t \cdot s$. By definition of $t \cdot s$, we have $t \leq s \to t \cdot s$, therefore $t \leq s' \to t \cdot s$ holds. Consequently, we have $t \cdot s' \leq t \cdot s$ by definition of $t \cdot s'$.

(2) By definition, we have $t \leq s \to t \cdot s$, which implies $t' \leq s \to t \cdot s$. Therefore, $t \cdot s$ is a solution to $t' \leq s \to s'$, hence we have $t' \cdot s \leq t \cdot s$.

□

Lemma C.15. *Let t and s be two types such that $t \leq \mathbb{0} \to \mathbb{1}$ and $s \leq \mathbb{0} \to \mathbb{1}$. Then $\mathsf{dom}(t) \vee \mathsf{dom}(s) \leq \mathsf{dom}(t \wedge s)$.*

Proof. Let $t = \bigvee_{i_1\in I_1} \tau_{i_1}$ and $s = \bigvee_{i_2\in I_2} \tau_{i_2}$ such that $\tau_i \not\simeq \mathbb{0}$ for all $i \in I_1 \cup I_2$. Then we have $t \wedge s = \bigvee_{i_1\in I_1, i_2\in I_2} \tau_{i_1} \wedge \tau_{i_2}$. Let $i_1 \in I_1$ and $i_2 \in I_2$. If $\tau_{i_1} \wedge \tau_{i_2} \simeq \mathbb{0}$, then $\mathsf{dom}(\tau_{i_1} \wedge \tau_{i_2}) = \mathbb{1}$. Otherwise, $\mathsf{dom}(\tau_{i_1} \wedge \tau_{i_2}) = \mathsf{dom}(\tau_{i_1}) \vee \mathsf{dom}(\tau_{i_2})$. In both cases, we have $\mathsf{dom}(\tau_{i_1} \wedge \tau_{i_2}) \geq \mathsf{dom}(\tau_{i_1}) \vee \mathsf{dom}(\tau_{i_2})$. Therefore

$$\begin{array}{rl} \mathsf{dom}(t \wedge s) \simeq & \bigwedge_{i_1\in I_1, i_2\in I_2} \mathsf{dom}(\tau_{i_1} \wedge \tau_{i_2}) \\ \geq & \bigwedge_{i_1\in I_1, i_2\in I_2} (\mathsf{dom}(\tau_{i_1}) \vee \mathsf{dom}(\tau_{i_2})) \\ \simeq & \bigwedge_{i_1\in I_1} (\bigwedge_{i_2\in I_2} (\mathsf{dom}(\tau_{i_1}) \vee \mathsf{dom}(\tau_{i_2}))) \\ \simeq & \bigwedge_{i_1\in I_1} (\mathsf{dom}(\tau_{i_1}) \vee (\bigwedge_{i_2\in I_2} \mathsf{dom}(\tau_{i_2}))) \\ \geq & \bigwedge_{i_1\in I_1} (\mathsf{dom}(\tau_{i_1})) \\ \simeq & \mathsf{dom}(t) \end{array}$$

Similarly, $\mathsf{dom}(t \wedge s) \geq \mathsf{dom}(s)$. Therefore $\mathsf{dom}(t) \vee \mathsf{dom}(s) \leq \mathsf{dom}(t \wedge s)$. □

For example, $\mathsf{dom}((\mathtt{Int} \to \mathtt{Int}) \wedge \neg(\mathtt{Bool} \to \mathtt{Bool})) \vee \mathsf{dom}(\mathtt{Bool} \to \mathtt{Bool}) = \mathtt{Int} \vee \mathtt{Bool}$, while $\mathsf{dom}(((\mathtt{Int} \to \mathtt{Int}) \wedge \neg(\mathtt{Bool} \to \mathtt{Bool})) \wedge (\mathtt{Bool} \to \mathtt{Bool})) = \mathsf{dom}(\mathbb{0}) = \mathbb{1}$.

Lemma C.16. *Let t be a type and σ be a type substitution such that $t \leq \mathbb{0} \to \mathbb{1}$. Then $\mathsf{dom}(t)\sigma \leq \mathsf{dom}(t\sigma)$*

Proof. We put t into its disjunctive normal form $\bigvee_{i\in I}\tau_i$ such that $\tau_i \not\simeq \mathbb{0}$ for all $i \in I$. Then we have $t\sigma = \bigvee_{i\in I}\tau_i\sigma$. So $\mathsf{dom}(t\sigma) = \bigwedge_{i\in I}\mathsf{dom}(\tau_i\sigma)$. Let $i \in I$. If $\tau_i\sigma \simeq \mathbb{0}$, then $\mathsf{dom}(\tau_i\sigma) = \mathbb{1}$. Otherwise, let $\tau_i = \bigwedge_{j\in P}(t_1^j \to t_2^j) \wedge \bigwedge_{k\in N}\neg(t_1^k \to t_2^k) \wedge \bigwedge_{\alpha\in P_\gamma}\alpha \wedge \bigwedge_{\alpha'\in N_\gamma}\neg\alpha'$. Then $\mathsf{dom}(\tau_i) = \bigvee_{j\in P} t_1^j$ and $\mathsf{dom}(\tau_i\sigma) = \bigvee_{j\in P} t_1^j\sigma \vee \mathsf{dom}((\bigwedge_{\alpha\in P_\gamma}\alpha \wedge \bigwedge_{\alpha'\in N_\gamma}\neg\alpha')\sigma \wedge \mathbb{0} \to \mathbb{1})$. In both cases, we have $\mathsf{dom}(\tau_i)\sigma \leq \mathsf{dom}(\tau_i\sigma)$. Therefore, $\bigwedge_{i\in I}\mathsf{dom}(\tau_i)\sigma \leq \bigwedge_{i\in I}\mathsf{dom}(\tau_i\sigma)$, that is, $\mathsf{dom}(t)\sigma \leq \mathsf{dom}(t\sigma)$. □

For example, $\mathsf{dom}((\mathtt{Int} \to \mathtt{Int}) \wedge \neg\alpha)\{^{(\mathtt{Int}\to\mathtt{Int})}/_{\alpha}\} = \mathtt{Int}\{^{(\mathtt{Int}\to\mathtt{Int})}/_{\alpha}\} = \mathtt{Int}$, while $\mathsf{dom}(((\mathtt{Int} \to \mathtt{Int}) \wedge \neg\alpha)\{^{(\mathtt{Int}\to\mathtt{Int})}/_{\alpha}\}) = \mathsf{dom}((\mathtt{Int} \to \mathtt{Int}) \wedge \neg(\mathtt{Int} \to \mathtt{Int})) = \mathbb{1}$.

Lemma C.17. *Let t be a type such that $t \leq \mathbb{0} \to \mathbb{1}$ and $[\sigma_k]_{k\in K}$ be a set of type substitutions. Then $\bigwedge_{k\in K}\mathsf{dom}(t)\sigma_k \leq \mathsf{dom}(\bigwedge_{k\in K} t\sigma_k)$*

Proof.

$$\begin{array}{rll} \bigwedge_{k\in K}\mathsf{dom}(t)\sigma_k \leq & \bigwedge_{k\in K}\mathsf{dom}(t\sigma_k) & \text{(by Lemma C.16)} \\ \leq & \bigvee_{k\in K}\mathsf{dom}(t\sigma_k) & \\ \leq & \mathsf{dom}(\bigwedge_{k\in K} t\sigma_k) & \text{(by Lemma C.15)} \end{array}$$

□

Lemma C.18. *Let t_1, s_1, t_2 and s_2 be types such that $t_1 \cdot s_1$ and $t_2 \cdot s_2$ exists. Then $(t_1 \wedge t_2) \cdot (s_1 \wedge s_2)$ exists and $(t_1 \wedge t_2) \cdot (s_1 \wedge s_2) \leq (t_1 \cdot s_1) \wedge (t_2 \cdot s_2)$.*

Proof. According to Lemma C.13, we have $s_i \leq \mathsf{dom}(t_i)$ and $t_i \leq s_i \to (t_i \cdot s_i)$. Then by Lemma C.15, we get $s_1 \wedge s_2 \leq \mathsf{dom}(t_1) \wedge \mathsf{dom}(t_2) \leq \mathsf{dom}(t_1 \wedge t_2)$. Moreover, $t_1 \wedge t_2 \leq (s_1 \to (t_1 \cdot s_1)) \wedge (s_2 \to (t_2 \cdot s_2)) \leq (s_1 \wedge s_2) \to ((t_1 \cdot s_1) \wedge (t_2 \cdot s_2))$. Therefore, $(t_1 \wedge t_2) \cdot (s_1 \wedge s_2)$ exists and $(t_1 \wedge t_2) \cdot (s_1 \wedge s_2) \leq (t_1 \cdot s_1) \wedge (t_2 \cdot s_2)$. □

For example, $((\mathtt{Int} \to \mathtt{Bool}) \wedge (\mathtt{Bool} \to \mathtt{Bool})) \cdot (\mathtt{Int} \wedge \mathtt{Bool}) = \mathbb{0}$, while $((\mathtt{Int} \to \mathtt{Bool}) \cdot \mathtt{Int}) \wedge ((\mathtt{Bool} \to \mathtt{Bool}) \cdot \mathtt{Bool}) = \mathtt{Bool} \wedge \mathtt{Bool} = \mathtt{Bool}$.

Lemma C.19. *Let t and s be two types such that $t \cdot s$ exists and σ be a type substitution. Then $(t\sigma) \cdot (s\sigma)$ exists and $(t\sigma) \cdot (s\sigma) \leq (t \cdot s)\sigma$.*

Proof. Because $t \cdot s$ exists, we have $s \leq \mathsf{dom}(t)$ and $t \leq s \to (t \cdot s)$. Then $s\sigma \leq \mathsf{dom}(t)\sigma$ and $t\sigma \leq s\sigma \to (t \cdot s)\sigma$. By Lemma C.16, we get $\mathsf{dom}(t)\sigma \leq \mathsf{dom}(t\sigma)$. So $s\sigma \leq \mathsf{dom}(t\sigma)$. Therefore, $(t\sigma) \cdot (s\sigma)$ exists. Moreover, since $(t \cdot s)\sigma$ is a solution to $t\sigma \leq s\sigma \to s'$, by definition, we have $(t\sigma) \cdot (s\sigma) \leq (t \cdot s)\sigma$. □

For example, $(((\mathtt{Int} \to \mathtt{Int}) \wedge \neg\alpha)\sigma) \cdot (\mathtt{Int}\sigma) = \mathbb{0} \cdot \mathtt{Int} = \mathbb{0}$, while $(((\mathtt{Int} \to \mathtt{Int}) \wedge \neg\alpha) \cdot \mathtt{Int})\sigma = \mathtt{Int}\sigma = \mathtt{Int}$, where $\sigma = \{^{(\mathtt{Int}\to\mathtt{Int})}/_{\alpha}\}$.

Lemma C.20. *Let t and s be two types and $[\sigma_k]_{k\in K}$ be a set of type substitutions such that $t \cdot s$ exists. Then $(\bigwedge_{k\in K} t\sigma_k) \cdot (\bigwedge_{k\in K} s\sigma_k)$ exists and $(\bigwedge_{k\in K} t\sigma_k) \cdot (\bigwedge_{k\in K} s\sigma_k) \leq \bigwedge_{k\in K}(t \cdot s)\sigma_k$.*

Proof. According to Lemmas C.19 and C.18, $(\bigwedge_{k\in K} t\sigma_k) \cdot (\bigwedge_{k\in K} s\sigma_k)$ exists. Moreover,

$$\begin{array}{rll} \bigwedge_{k\in K}(t \cdot s)\sigma_k \geq & \bigwedge_{k\in K}(t\sigma_k \cdot s\sigma_k) & \text{(Lemma C.19)} \\ \geq & (\bigwedge_{k\in K} t\sigma_k) \cdot (\bigwedge_{k\in K} s\sigma_k) & \text{(Lemma C.18)} \end{array}$$

□

C.2.3 Syntax-Directed Rules

Because of subsumption, the typing rules provided in Section A.3 are not syntax-directed and so they do not yield a type-checking algorithm directly. In simply type λ-calculus, subsumption is used to bridge gaps between the types expected by functions and the actual types of their arguments in applications [14]. In our calculus, we identify four situations where the subsumption is needed, namely, the rules for projections, abstractions, applications, and type cases. To see why, we consider a typing derivation ending with each typing rule whose immediate sub-derivation ends with (*subsum*). For each case, we explain how the use of subsumption can be pushed through the typing rule under consideration, or how the rule should be modified to take subtyping into account.

First we consider the case where a typing derivation ends with (*subsum*) whose immediate sub-derivation also ends with (*subsum*). The two consecutive uses of (*subsum*) can be merged into one, because the subtyping relation is transitive.

Lemma C.21. *If $\Delta\,;\Gamma \vdash e : t$, then there exists a derivation for $\Delta\,;\Gamma \vdash e : t$ where there are no consecutive instances of* (*subsum*).

Proof. Assume that there exist two consecutive instances of (*subsum*) occurring in a derivation of $\Delta \wr \Gamma \vdash e : t$, that is,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\Delta' \wr \Gamma' \vdash e' : s'_2} \quad s'_2 \leq s'_1}{\Delta' \wr \Gamma' \vdash e' : s'_1}\,(subsum) \quad s'_1 \leq t'}{\Delta' \wr \Gamma' \vdash e' : t'}\,(subsum)}{\dots \qquad \vdots \qquad \dots}}{\Delta \wr \Gamma \vdash e : t}$$

Since $s'_2 \leq s'_1$ and $s'_1 \leq t'$, we have $s'_2 \leq t'$. So we can rewrite this derivation as follows:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta' \wr \Gamma' \vdash e' : s'_2} \quad s'_2 \leq t'}{\Delta' \wr \Gamma' \vdash e' : t'}\,(subsum)}{\dots \qquad \vdots \qquad \dots}}{\Delta \wr \Gamma \vdash e : t}$$

Therefore, the result follows. □

Next, consider an instance of (*pair*) such that one of its sub-derivations ends with an instance of (*subsum*), for example, the left sub-derivation:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e_1 : s_1} \quad s_1 \leq t_1}{\Delta \wr \Gamma \vdash e_1 : t_1}\,(subsum) \quad \dfrac{\dots}{\Delta \wr \Gamma \vdash e_2 : t_2}}{\Delta \wr \Gamma \vdash (e_1, e_2) : (t_1 \times t_2)}\,(pair)$$

As $s_1 \leq t_1$, we have $s_1 \times t_2 \leq t_1 \times t_2$. Then we can move subsumption down through the rule (*pair*), giving the following derivation:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e_1 : s_1} \quad \dfrac{\dots}{\Delta \wr \Gamma \vdash e_2 : t_2}}{\Delta \wr \Gamma \vdash (e_1, e_2) : (s_1 \times t_2)}\,(pair) \quad s_1 \times t_2 \leq t_1 \times t_2}{\Delta \wr \Gamma \vdash (e_1, e_2) : (t_1 \times t_2)}\,(subsum)$$

The rule (*proj*) is a little trickier than (*pair*). Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e : s} \quad s \leq t_1 \times t_2}{\Delta \wr \Gamma \vdash e : (t_1 \times t_2)}\,(subsum)}{\Delta \wr \Gamma \vdash \pi_i(e) : t_i}\,(proj)$$

As $s \leq t_1 \times t_2$, s is a pair type. According to the decomposition of s and Lemma C.5, we can rewrite the previous derivation into the following one:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e : s} \quad s \leq \mathbb{1} \times \mathbb{1}}{\Delta \wr \Gamma \vdash \pi_i(e) : \boldsymbol{\pi}_i(s)} \quad \boldsymbol{\pi}_i(s) \leq t_i}{\Delta \wr \Gamma \vdash \pi_i(e) : t_i}$$

Note that the subtyping check $s \leq \mathbb{1} \times \mathbb{1}$ ensures that s is a pair type.

Next consider an instance of (*abstr*) (where $\Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j))$. All the sub-derivations may end with (*subsum*):

$$\dfrac{\forall i \in I, j \in J.\ \dfrac{\dfrac{\dots}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s'_{ij}} \quad s'_{ij} \leq s_i\sigma_j}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s_i\sigma_j}}{\Delta \wr \Gamma \vdash \lambda^{\bigwedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)}\,(abstr)$$

Without subsumption, we would assign the type $\bigwedge_{i\in I,j\in J}(t_i\sigma_j \to s'_{ij})$ to the abstraction, while we want to assign the type $\bigwedge_{i\in I,j\in J}(t_i\sigma_j \to s_i\sigma_j)$ to it because of the type annotations. Consequently, we have to keep the subtyping checks $s'_{ij} \leq s_i\sigma_j$ as side-conditions of an algorithmic typing rule for abstractions.

$$\dfrac{\forall i \in I, j \in J. \quad \Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s'_{ij} \qquad s'_{ij} \leq s_i\sigma_j}{\Delta \wr \Gamma \vdash \lambda^{\bigwedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)}$$

In (*appl*) case, suppose that both sub-derivations end with (*subsum*):

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e_1 : t} \quad t \leq t' \to s'}{\Delta \wr \Gamma \vdash e_1 : t' \to s'} \quad \dfrac{\dfrac{\dots}{\Delta \wr \Gamma \vdash e_1 : s} \quad s \leq t'}{\Delta \wr \Gamma \vdash e_2 : t'}}{\Delta \wr \Gamma \vdash e_1\ e_2 : s'}\,(appl)$$

Since $s \leq t'$, then by the contravariance of arrow types we have $t' \to s' \leq s \to s'$. Hence, such a derivation can be rewritten as

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e_1 : t} \quad t \leq t' \to s'}{\Delta \fatsemi \Gamma \vdash e_1 : t' \to s'} \qquad t' \to s' \leq s \to s'}{\Delta \fatsemi \Gamma \vdash e_1 : s \to s'} \qquad \dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e_1 : s}}{\Delta \fatsemi \Gamma \vdash e_1\ e_2 : s'}\ (appl)$$

Applying Lemma C.21, we can merge the two adjacent instances of (*subsum*) into one:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e_1 : t} \quad t \leq s \to s'}{\Delta \fatsemi \Gamma \vdash e_1 : s \to s'} \qquad \dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e_1 : s}}{\Delta \fatsemi \Gamma \vdash e_1\ e_2 : s'}\ (appl)$$

A syntax-directed typing rule for applications can then be written as follows

$$\dfrac{\Delta \fatsemi \Gamma \vdash e_1 : t \quad \Delta \fatsemi \Gamma \vdash e_2 : s \quad t \leq s \to s'}{\Delta \fatsemi \Gamma \vdash e_1 e_2 : s'}$$

where subsumption is used as a side condition to bridge the gap between the function type and the argument type.

This typing rule is not algorithmic yet, because the result type s' can be any type verifying the side condition. Using Lemma C.12, we can equivalently rewrite the side condition as $t \leq \mathbb{0} \to \mathbb{1}$ and $s \leq \mathsf{dom}(t)$ without involving the result type s'. The first condition ensures that t is a function type and the second one that the argument type s can be safely applied by t. Moreover, we assign the type $t \cdot s$ to the application, which is by definition the smallest possible type for it. We obtain then the following algorithmic typing rule.

$$\dfrac{\Delta \fatsemi \Gamma \vdash e_1 : t \quad \Delta \fatsemi \Gamma \vdash e_2 : s \quad t \leq \mathbb{0} \to \mathbb{1} \quad s \leq \mathsf{dom}(t)}{\Delta \fatsemi \Gamma \vdash e_1 e_2 : t \cdot s}$$

Next, let us discuss the rule (*case*):

$$\dfrac{\Delta \fatsemi \Gamma \vdash e : t' \qquad \begin{cases} t' \not\leq \neg t & \Rightarrow \quad \Delta \fatsemi \Gamma \vdash e_1 : s \\ t' \not\leq t & \Rightarrow \quad \Delta \fatsemi \Gamma \vdash e_2 : s \end{cases}}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}\ (case)$$

The rule covers four different situations, depending on which branches of the type-cases are checked: (i) no branch is type-checked, (ii) the first branch e_1 is type-checked, (iii) the second branch e_2 is type-checked, and (iv) both branches are type-checked. Each case produces a corresponding algorithmic rule.

In case (i), we have simultaneously $t' \leq t$ and $t' \leq \neg t$, which means that $t' = \mathbb{0}$. Consequently, e does not reduce to a value (otherwise, subject reduction would be violated), and neither does the whole type case. Consequently, we can assign type $\mathbb{0}$ to the whole type.

$$\dfrac{\dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e : \mathbb{0}}}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : \mathbb{0}}$$

Suppose we are in case (ii) and the sub-derivation for the first branch e_1 ends with (*subsum*):

$$\dfrac{\Delta \fatsemi \Gamma \vdash e : t' \quad t' \leq t \qquad \dfrac{\dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e_1 : s_1} \quad s_1 \leq s}{\Delta \fatsemi \Gamma \vdash e_1 : s}}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}\ (case)$$

Such a derivation can be rearranged as:

$$\dfrac{\dfrac{\Delta \fatsemi \Gamma \vdash e : t' \quad t' \leq t \quad \dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e_1 : s_1}}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s_1} \qquad s_1 \leq s}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}$$

Moreover, (*subsum*) might also be used at the end of the sub-derivation for e:

$$\dfrac{\dfrac{\dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e : t''} \quad t'' \leq t'}{\Delta \fatsemi \Gamma \vdash e : t'} \qquad t' \leq t \quad \Delta \fatsemi \Gamma \vdash e_1 : s}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}\ (case)$$

From $t'' \leq t'$ and $t' \leq t$, we deduce $t'' \leq t$ by transitivity. Therefore this use of subtyping can be merged with the subtyping check of the type case rule. We then obtain the following algorithmic rule.

$$\dfrac{\dfrac{\dots}{\Delta \fatsemi \Gamma \vdash e : t''} \quad t'' \leq t \quad \Delta \fatsemi \Gamma \vdash e_1 : s}{\Delta \fatsemi \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}$$

We obtain a similar rule for case (iii), except that e_2 is type-checked instead of e_1, and t'' is tested against $\neg t$.

Finally, consider case (iv). We have to type-check both branches and each typing derivation may end with (*subsum*):

$$\dfrac{\Delta \wr \Gamma \vdash e : t' \quad \begin{cases} t' \not\leq \neg t \quad \text{and} \quad \dfrac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash e_1 : s_1} \quad s_1 \leq s}{\Delta \wr \Gamma \vdash e_1 : s} \\ t' \not\leq t \quad \text{and} \quad \dfrac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash e_2 : s_2} \quad s_2 \leq s}{\Delta \wr \Gamma \vdash e_2 : s} \end{cases}}{\Delta \wr \Gamma \vdash (e \in t \,?\, e_1 : e_2) : s}\ (case)$$

Subsumption is used there just to unify s_1 and s_2 into a common type s, which is used to type the whole type case. Such a common type can also be obtained by taking the least upper-bound of s_1 and s_2, i.e., $s_1 \vee s_2$. Because $s_1 \leq s$ and $s_2 \leq s$, we have $s_1 \vee s_2 \leq s$, and we can rewrite the derivation as follows:

$$\dfrac{\dfrac{\Delta \wr \Gamma \vdash e : t' \quad \begin{cases} t' \not\leq \neg t \quad \text{and} \quad \dfrac{\ldots}{\Delta \wr \Gamma \vdash e_1 : s_1} \\ t' \not\leq t \quad \text{and} \quad \dfrac{\ldots}{\Delta \wr \Gamma \vdash e_2 : s_2} \end{cases}}{\Delta \wr \Gamma \vdash (e \in t \,?\, e_1 : e_2) : s_1 \vee s_2}\ (case) \qquad s_1 \vee s_2 \leq s}{\Delta \wr \Gamma \vdash (e \in t \,?\, e_1 : e_2) : s}$$

Suppose now that the sub-derivation for e ends with (*subsum*):

$$\dfrac{\dfrac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash e : t''} \quad t'' \leq t'}{\Delta \wr \Gamma \vdash e : t'} \quad \begin{cases} t' \not\leq \neg t \quad \text{and} \quad \Delta \wr \Gamma \vdash e_1 : s_1 \\ t' \not\leq t \quad \text{and} \quad \Delta \wr \Gamma \vdash e_2 : s_2 \end{cases}}{\Delta \wr \Gamma \vdash (e \in t \,?\, e_1 : e_2) : s_1 \vee s_2}\ (case)$$

The relations $t'' \leq t'$, $t' \not\leq \neg t$ do not necessarily imply $t'' \not\leq \neg t$, and $t'' \leq t'$, $t' \not\leq t$ do not necessarily imply $t'' \not\leq \neg t$. Therefore, by using the type t'' instead of t' for e, we may type-check less branches. If so, then we would be in one of the cases $(i) - (iii)$, and the result type (*i.e.*, a type among $\mathbb{0}$, s_1 or s_2) for the whole type case would be smaller than $s_1 \vee s_2$. It would then be possible to type the type case with $s_1 \vee s_2$ by subsumption. Otherwise, we type-check as many branches with t'' as with t', and we can modify the rule into

$$\dfrac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash e : t''} \quad \begin{cases} t'' \not\leq \neg t \quad \text{and} \quad \Delta \wr \Gamma \vdash e_1 : s_1 \\ t'' \not\leq t \quad \text{and} \quad \Delta \wr \Gamma \vdash e_2 : s_2 \end{cases}}{\Delta \wr \Gamma \vdash (e \in t \,?\, e_1 : e_2) : s_1 \vee s_2}$$

Finally, consider the case where the last rule in a derivation is (*instinter*) and all its sub-derivations end with (*subsum*):

$$\dfrac{\dfrac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash e : s} \quad s \leq t}{\Delta \wr \Gamma \vdash e : t}\ (subsum) \qquad \forall j \in J.\ \sigma_j \not\sharp \Delta}{\Delta \wr \Gamma \vdash e[\sigma_j]_{j \in J} : \bigwedge_{j \in J} t\sigma_j}\ (instinter)$$

Since $s \leq t$, we have $\bigwedge_{j \in J} s\sigma_j \leq \bigwedge_{j \in J} t\sigma_j$. So such a derivation can be rewritten into

$$\dfrac{\dfrac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash e : s} \quad \forall j \in J.\ \sigma_j \not\sharp \Delta}{\Delta \wr \Gamma \vdash e[\sigma_j]_{j \in J} : \bigwedge_{j \in J} s\sigma_j}\ (instinter) \qquad \bigwedge_{j \in J} s\sigma_j \leq \bigwedge_{j \in J} t\sigma_j}{\Delta \wr \Gamma \vdash e[\sigma_j]_{j \in J} : \bigwedge_{j \in J} t\sigma_j}\ (subsum)$$

In conclusion, by applying the aforementioned transformations repeatedly, we can rewrite an arbitrary typing derivation into a special form where subsumption are used at the end of sub-derivations of projections, abstractions or applications, in the conditions of type cases and at the very end of the whole derivation. Thus, this transformations yields a set of syntax-directed typing rules, given in Figure 6. Let $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t$ denote the typing judgments derivable by the set of syntax-directed typing rules.

Theorem C.22 (Soundness). *Let e be an expression. If $\Gamma \vdash_{\mathscr{A}} e : t$, then $\Gamma \vdash e : t$.*

Proof. By induction on the typing derivation of $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t$. We proceed by a case analysis on the last rule used in the derivation.

(ALG-CONST)**:** straightforward.
(ALG-VAR)**:** straightforward.

$$\frac{}{\Delta \wr \Gamma \vdash_{\mathscr{A}} c : b_c}\ (\textsc{Alg-const}) \qquad \frac{}{\Delta \wr \Gamma \vdash_{\mathscr{A}} x : \Gamma(x)}\ (\textsc{Alg-var})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : t_1 \qquad \Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : t_2}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e_1, e_2) : t_1 \times t_2}\ (\textsc{Alg-pair}) \qquad \frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t \qquad t \leq \mathbb{1} \times \mathbb{1}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} \pi_i(e) : \boldsymbol{\pi}_i(t)}\ (\textsc{Alg-proj})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : t \qquad \Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s \qquad t \leq \mathbb{0} \to \mathbb{1} \qquad s \leq \mathsf{dom}(t)}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 e_2 : t \cdot s}\ (\textsc{Alg-appl})$$

$$\frac{\begin{array}{l}\Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i \in I, j \in J} (t_i\sigma_j \to s_i\sigma_j)) \\ \forall i \in I, j \in J.\ \Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash_{\mathscr{A}} e@[\sigma_j] : s'_{ij} \qquad s'_{ij} \leq s_i\sigma_j\end{array}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} \lambda^{\bigwedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i \in I, j \in J} (t_i\sigma_j \to s_i\sigma_j)}\ (\textsc{Alg-abstr})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : \mathbb{0}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : \mathbb{0}}\ (\textsc{Alg-case-none})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t' \qquad t' \leq t \qquad t' \not\leq \neg t \qquad \Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : s_1}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : s_1}\ (\textsc{Alg-case-fst})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t' \qquad t' \leq \neg t \qquad t' \not\leq t \qquad \Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s_2}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : s_2}\ (\textsc{Alg-case-snd})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t' \qquad \begin{cases} t' \not\leq \neg t & \text{and} \quad \Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : s_1 \\ t' \not\leq t & \text{and} \quad \Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s_2 \end{cases}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : s_1 \vee s_2}\ (\textsc{Alg-case-both})$$

$$\frac{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t \qquad \forall j \in J.\ \sigma_j \sharp \Delta \qquad |J| > 0}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e[\sigma_j]_{j\in J} : \bigwedge_{j \in J} t\sigma_j}\ (\textsc{Alg-inst})$$

Figure 6. Syntax-directed typing rules

(ALG-PAIR)**:** consider the derivation

$$\frac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : t_1} \qquad \dfrac{\ldots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : t_2}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e_1, e_2) : t_1 \times t_2}$$

Applying the induction hypothesis twice, we get $\Delta \wr \Gamma \vdash e_i : t_i$. Then by applying the rule (*pair*), we have $\Delta \wr \Gamma \vdash (e_1, e_2) : t_1 \times t_2$.

(ALG-PROJ)**:** consider the derivation

$$\frac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t} \qquad t \leq \mathbb{1} \times \mathbb{1}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} \pi_i(e) : \boldsymbol{\pi}_i(t)}$$

By induction, we have $\Delta \wr \Gamma \vdash e : t$. According to Lemma C.5, we have $t \leq (\boldsymbol{\pi}_1(t) \times \boldsymbol{\pi}_2(t))$. Then by (*subsum*), we get $\Delta \wr \Gamma \vdash e : (\boldsymbol{\pi}_1(t) \times \boldsymbol{\pi}_2(t))$. Finally, the rule (*proj*) gives us $\Delta \wr \Gamma \vdash \pi_i(e) : \boldsymbol{\pi}_i(t)$.

(ALG-APPL)**:** consider the derivation

$$\frac{\dfrac{\ldots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : t} \qquad \dfrac{\ldots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s} \qquad t \leq \mathbb{0} \to \mathbb{1} \qquad s \leq \mathsf{dom}(t)}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 e_2 : t \cdot s}$$

By induction, we have $\Delta \wr \Gamma \vdash e_1 : t$ and $\Delta \wr \Gamma \vdash e_2 : s$. According to Lemma C.13, we have

$$t \cdot s = \min\{s' \mid t \leq s \to s'\}$$

Note that the conditions $t \leq \mathbb{0} \to \mathbb{1}$ and $s \leq \mathsf{dom}(t)$ ensure that such a type exists. It is clear $t \leq s \to (t \cdot s)$. Then by (*subsum*), we get $\Delta \wr \Gamma \vdash e_1 : s \to (t \cdot s)$. Finally, the rule (*appl*) gives us $\Delta \wr \Gamma \vdash e_1 e_2 : t \cdot s$.

(ALG-ABSTR)**:** consider the derivation

$$\frac{\forall i \in I, j \in J.\ \dfrac{\cdots}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash_{\mathscr{A}} e@[\sigma_j] : s'_{ij}} \qquad s'_{ij} \leq s_i\sigma_j}{\Delta \wr \Gamma \vdash_{\mathscr{A}} \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)}$$

with $\Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j))$. By induction, for all $i \in I$ and $j \in J$, we have $\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s'_{ij}$. Since $s'_{ij} \leq s_i\sigma_j$, by (*subsum*), we get $\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s_i\sigma_j$. Finally, the rule (*abstr*) gives us $\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)$.

(ALG-CASE-NONE)**:** consider the derivation

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : \mathbb{0}}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e{\in}t\,?\,e_1 : e_2) : \mathbb{0}}$$

By induction, we have $\Delta \wr \Gamma \vdash e : \mathbb{0}$. No branch is type-checked by the rule (*case*), so any type can be assigned to the type case expression, and in particular we have $\Delta \wr \Gamma \vdash (e{\in}t\,?\,e_1 : e_2) : \mathbb{0}$

(ALG-CASE-FST)**:** consider the derivation

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t'} \quad t' \leq t \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : s_1}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e{\in}t\,?\,e_1 : e_2) : s_1}$$

By induction, we have $\Delta \wr \Gamma \vdash e : t'$ and $\Delta \wr \Gamma \vdash e_1 : s_1$. As $t' \leq t$, then we only need to type-check the first branch. Therefore, by the rule (*case*), we have $\Delta \wr \Gamma \vdash (e{\in}t\,?\,e_1 : e_2) : s_1$.

(ALG-CASE-SND)**:** similar the case of (ALG-CASE-FST).

(ALG-CASE-BOTH)**:** consider the derivation

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t'} \quad \begin{cases} t' \not\leq \neg t \quad \text{and} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : s_1} \\ t' \not\leq t \quad \text{and} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s_2} \end{cases}}{\Delta \wr \Gamma \vdash_{\mathscr{A}} (e{\in}t\,?\,e_1 : e_2) : s_1 \vee s_2}$$

By induction, we have $\Delta \wr \Gamma \vdash e : t'$, $\Delta \wr \Gamma \vdash e_1 : s_1$ and $\Delta \wr \Gamma \vdash e_2 : s_2$. It is clear that $s_1 \leq s_1 \vee s_2$ and $s_2 \leq s_1 \vee s_2$. Then by (*subsum*), we get $\Delta \wr \Gamma \vdash e_1 : s_1 \vee s_2$ and $\Delta \wr \Gamma \vdash e_2 : s_1 \vee s_2$. Moreover, as $t' \not\leq \neg t$ and $t' \not\leq t$, we have to type-check both branches. Finally, by the rule (*case*), we get $\Delta \wr \Gamma \vdash (e{\in}t\,?\,e_1 : e_2) : s_1 \vee s_2$.

(ALG-INST)**:** consider the derivation

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t} \quad \forall j \in J.\ \sigma_j \not\sharp \Delta}{\Delta \wr \Gamma \vdash_{\mathscr{A}} e[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t\sigma_j}$$

By induction, we have $\Delta \wr \Gamma \vdash e : t$. As $\forall j \in J.\ \sigma_j \not\sharp \Delta$, by (*instinter*), we get $\Delta \wr \Gamma \vdash e[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t\sigma_j$.

□

Theorem C.23 (**Completeness).** *Let $\leq$ be a subtyping relation induced by a well-founded (convex) model with infinite support and e an expression. If $\Delta \wr \Gamma \vdash e : t$, then there exists a type s such that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : s$ and $s \leq t$.*

Proof. By induction on the typing derivation of $\Delta \wr \Gamma \vdash e : t$. We proceed by case analysis on the last rule used in the derivation.

(*const*)**:** straightforward (take s as b_c).

(*var*)**:** straightforward (take s as $\Gamma(x)$).

(*pair*)**:** consider the derivation

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e_1 : t_1} \quad \dfrac{\cdots}{\Delta \wr \Gamma \vdash e_2 : t_2}}{\Delta \wr \Gamma \vdash (e_1, e_2) : t_1 \times t_2}\ (pair)$$

Applying the induction hypothesis twice, we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_i : s_i$ where $s_i \leq t_i$. Then the rule (ALG-PAIR) gives us $\Delta \wr \Gamma \vdash_{\mathscr{A}} (e_1, e_2) : s_1 \times s_2$. Since $s_i \leq t_i$, we deduce $(s_1 \times s_2) \leq (t_1 \times t_2)$.

(*proj*)**:** consider the derivation

$$\frac{\dfrac{\cdots}{\Delta \wr \Gamma \vdash e : (t_1 \times t_2)}}{\Delta \wr \Gamma \vdash \pi_i(e) : t_i}\ (proj)$$

By induction, there exists s such that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : s$ and $s \leq (t_1 \times t_2)$. Clearly we have $s \leq \mathbb{1} \times \mathbb{1}$. Applying (ALG-PROJ), we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} \pi_i(e) : \boldsymbol{\pi}_i(s)$. Moreover, as $s \leq (t_1 \times t_2)$, according to Lemma C.5, we have $\boldsymbol{\pi}_i(s) \leq t_i$. Therefore, the result follows.

(*appl*)**:** consider the derivation

$$\frac{\overset{\cdots}{\Delta \wr \Gamma \vdash e_1 : t_1 \to t_2} \qquad \overset{\cdots}{\Delta \wr \Gamma \vdash e_2 : t_1}}{\Delta \wr \Gamma \vdash e_1 e_2 : t_2}\ (appl)$$

Applying the induction hypothesis twice, we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : t$ and $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s$ where $t \leq t_1 \to t_2$ and $s \leq t_1$. Clearly we have $t \leq \mathbb{0} \to \mathbb{1}$ and $t \leq s \to t_2$ (by contravariance of arrows). From Lemma C.12, we get $s \leq \mathsf{dom}(t)$. So, by applying the rule (ALG-APPL), we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 e_2 : t \cdot s$. Moreover, it is clear that t_2 is a solution for $t \leq s \to s'$. Consequently, it is a super type of $t \cdot s$, that is $t \cdot s \leq t_2$.

(*abstr*)**:** consider the derivation

$$\frac{\forall i \in I, j \in J.\ \overset{\cdots}{\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash e@[\sigma_j] : s_i\sigma_j}}{\Delta \wr \Gamma \vdash \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)}\ (abstr)$$

where $\Delta' = \Delta \cup \mathsf{var}(\bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j))$. By induction, for all $i \in I$ and $j \in J$, there exists s'_{ij} such that $\Delta' \wr \Gamma, (x : t_i\sigma_j) \vdash_{\mathscr{A}} e@[\sigma_j] : s'_{ij}$ and $s'_{ij} \leq s_i\sigma_j$. Then the rule (ALG-ABSTR) gives us $\Delta \wr \Gamma \vdash_{\mathscr{A}} \lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e : \bigwedge_{i\in I, j\in J}(t_i\sigma_j \to s_i\sigma_j)$

(*case*)**:** consider the derivation

$$\frac{\overset{\cdots}{\Delta \wr \Gamma \vdash e : t'} \qquad \begin{cases} t' \not\leq \neg t & \Rightarrow \quad \overset{\cdots}{\Delta \wr \Gamma \vdash e_1 : s} \\ t' \not\leq t & \Rightarrow \quad \overset{\cdots}{\Delta \wr \Gamma \vdash e_2 : s} \end{cases}}{\Delta \wr \Gamma \vdash (e \in t\,?\,e_1 : e_2) : s}\ (case)$$

By induction hypothesis on $\Delta \wr \Gamma \vdash e : t'$, there exists a type t'' such that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t''$ and $t'' \leq t'$. If $t'' \simeq \mathbb{0}$, by (ALG-CASE-NONE), we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : \mathbb{0}$. The result follows straightforwardly. In what follows, we assume that $t'' \not\simeq \mathbb{0}$.

Assume that $t'' \leq t$. Because $t'' \leq t'$, we have $t' \not\leq \neg t$ (otherwise, $t'' \simeq \mathbb{0}$). Therefore the first branch is type-checked, and by induction, there exists a type s_1 such that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : s_1$ and $s_1 \leq s$. Then the rule (ALG-CASE-FST) gives us $\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : s_1$.

Otherwise, $t'' \not\leq t$. In this case, we have $t' \not\leq t$ (otherwise, $t'' \leq t$). Then the second branch is type-checked. By induction, there exists a type s_2 such that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_2 : s_2$ and $s_2 \leq s$. If $t'' \leq \neg t$, then by the rule (ALG-CASE-SND), we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : s_2$. The result follows. Otherwise, we also have $t'' \not\leq \neg t$. Then we also have $t' \not\leq \neg t$ (otherwise, $t'' \leq \neg t$). So the first branch should be type-checked as well. By induction, we have $\Delta \wr \Gamma \vdash_{\mathscr{A}} e_1 : s_1$ where $s_1 \leq s$. By applying (ALG-CASE-BOTH), we get $\Delta \wr \Gamma \vdash_{\mathscr{A}} (e \in t\,?\,e_1 : e_2) : s_1 \vee s_2$. Since $s_1 \leq s$ and $s_2 \leq s$, we deduce that $s_1 \vee s_2 \leq s$. The result follows as well.

(*instinter*)**:** consider the derivation

$$\frac{\overset{\cdots}{\Delta \wr \Gamma \vdash e : t} \qquad \forall j \in J.\ \sigma_j \sharp \Delta}{\Delta \wr \Gamma \vdash e[\sigma_j]_{j\in J} : \bigwedge_{j\in J} t\sigma_j}\ (instinter)$$

By induction, there exists a type s such that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : s$ and $s \leq t$. Then the rule (ALG-INST) gives us that $\Delta \wr \Gamma \vdash_{\mathscr{A}} e[\sigma_j]_{j\in J} : \bigwedge_{j\in J} s\sigma_j$. Since $s \leq t$, we have $\bigwedge_{j\in J} s\sigma_j \leq \bigwedge_{j\in J} t\sigma_j$. Therefore, the result follows.

□

Corollary C.24 (Minimum typing). *Let e be an expression. If $\Delta \wr \Gamma \vdash_{\mathscr{A}} e : t$, then $t = \min\{s \mid \Delta \wr \Gamma \vdash e : s\}$.*

Proof. Consequence of Theorems C.22 and C.23. □

To prove the termination of the type-checking algorithm, we define the *size* of an expression e as follows.

Definition C.25. *Let e be an expression. We define the* size *of e as:*

$$\begin{aligned}
\mathit{size}(c) &= 1 \\
\mathit{size}(x) &= 1 \\
\mathit{size}((e_1, e_2)) &= \mathit{size}(e_1) + \mathit{size}(e_2) + 1 \\
\mathit{size}(\pi_i(e)) &= \mathit{size}(e) + 1 \\
\mathit{size}(e_1 e_2) &= \mathit{size}(e_1) + \mathit{size}(e_2) + 1 \\
\mathit{size}(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e) &= \mathit{size}(e) + 1 \\
\mathit{size}(e \in t\,?\,e_1 : e_2) &= \mathit{size}(e) + \mathit{size}(e_1) + \mathit{size}(e_2) + 1 \\
\mathit{size}(e[\sigma_j]_{j\in J}) &= \mathit{size}(e) + 1
\end{aligned}$$

The relabeling does not enlarge the size of the expression.

Lemma C.26. *Let e be an expression and $[\sigma_j]_{j\in J}$ a set of type substitutions. Then*

$$\mathit{size}(e@[\sigma_j]_{j\in J}) \leq \mathit{size}(e).$$

Proof. By induction on the structure of e. □

Theorem C.27 (Termination). *Let e be an expression. Then the type-inference algorithm for e terminates.*

Proof. By induction on the sizes of the expressions to be checked.

(ALG-CONST): it terminates immediately.
(ALG-VAR): it terminates immediately.
(ALG-PAIR): $\mathrm{size}(e_1) + \mathrm{size}(e_2) < \mathrm{size}((e_1, e_2))$.
(ALG-PROJ): $\mathrm{size}(e') < \mathrm{size}(\pi_1(e'))$.
(ALG-APPL): $\mathrm{size}(e_1) + \mathrm{size}(e_2) < \mathrm{size}(e_1 e_2)$.
(ALG-ABSTR): by Lemma C.26, we have $\mathrm{size}(e'@[\sigma_j]) \leq \mathrm{size}(e')$. Then we get

$$\mathrm{size}(e'@[\sigma_j]) < \mathrm{size}(\lambda^{\wedge_{i\in I} t_i \to s_i}_{[\sigma_j]_{j\in J}} x.e').$$

(ALG-CASE): $\mathrm{size}(e') + \mathrm{size}(e_1) + \mathrm{size}(e_2) < \mathrm{size}(e' \in t \,?\, e_1 : e_2)$.
(ALG-INST): $\mathrm{size}(e') < \mathrm{size}(e'[\sigma_j]_{j\in J})$.

□

D. Evaluation

As described in Section 3, below we assume that polymorphic variables are pairwise distinct and distinct from any monomorphic variables in the expressions under consideration. In particular, when substituting an expression e for a variable x in an expression e', we assume the polymorphic type variables of e' to be distinct from the monomorphic and polymorphic type variables of e. Furthermore, we assume that there are no useless type variables in the domain of any type-substitution and no redundant type-substitutions in any set of type-substitutions (Lemmas B.9 and B.10).

D.1 Equivalence between small-step semantics and big-step semantics

In this section, we prove the equivalence between the small-step semantics in Section A.4 and the big-step semantics for the polymorphic language in Section 5.2 (extended with let-polymorphism in Section 5.4). For clarity, we use v for the values for the small-step semantics and v_p for those of the big-step semantics.

Definition D.1 (Polymorphic language)**.**

$$\begin{array}{lcll} e & ::= & c \mid x \mid \underline{x} \mid \lambda^t_{\sigma_I} x.e \mid ee \mid e \in t \,?\, e : e & \\ & \mid & e\sigma_I \mid \mathtt{let}\ \underline{x} = e\ \mathtt{in}\ e & \\ v & ::= & c \mid \lambda^t_{\sigma_I} x.e & \textit{(for the small-step semantics)} \\ v_\mathsf{p} & ::= & c \mid \langle \lambda^t_{\sigma_I} x.e, \mathscr{E}, \sigma_I \rangle & \textit{(for the big-step semantics)} \end{array}$$

Definition D.2 (Membership)**.** *The membership relation $v_\mathsf{p} \in_\mathsf{p} t$ for polymorphic values is inductively defined as follows:*

$$\begin{array}{rcl} c \in_\mathsf{p} t & \stackrel{\text{def}}{=} & b_c \leq t \\ \langle \lambda^s_{\sigma_I} x.e, \mathscr{E}, \sigma_J \rangle \in_\mathsf{p} t & \stackrel{\text{def}}{=} & s(\sigma_J \circ \sigma_I) \leq t \end{array}$$

Definition D.3. *We define a transformation function $(\!|.|\!)$ from v_p to v as follows:*

$$\begin{array}{rcl} (\!|c|\!) & \stackrel{\text{def}}{=} & c \\ (\!|\langle \lambda^s_{\sigma_I} x.e, \mathscr{E}, \sigma_J \rangle|\!) & \stackrel{\text{def}}{=} & \lambda^s_{\sigma_J \circ \sigma_I} x.e(\!|\mathscr{E}|\!) \end{array}$$

where $(\!|\mathscr{E}|\!)$ is defined as follows:

$$\begin{array}{rcl} (\!|\varnothing|\!) & \stackrel{\text{def}}{=} & \varnothing \\ (\!|\mathscr{E}, x \mapsto v_\mathsf{p}|\!) & \stackrel{\text{def}}{=} & (\!|\mathscr{E}|\!) \cup \{(\!|v_\mathsf{p}|\!)/x\} \\ (\!|\mathscr{E}, \underline{x} \mapsto v_\mathsf{p}|\!) & \stackrel{\text{def}}{=} & (\!|\mathscr{E}|\!) \cup \{(\!|v_\mathsf{p}|\!)/\underline{x}\} \end{array}$$

In the definition of $(\!|\langle \lambda^s_{\sigma_I} x.e, \mathscr{E}, \sigma_J \rangle|\!)$, to avoid unwanted captures, we assume that the polymorphic type variables of $\lambda^s_{\sigma_J \circ \sigma_I} x.e$ are distinct from the monomorphic and polymorphic type variables in $\mathscr{E}$, *ie*, $\mathsf{tv}(\mathscr{E})$ as precisely defined in Definition D.4. This is guaranteed by the assumption we impose on variables.

Definition D.4. *The sets $\mathsf{tv}(v_\mathsf{p})$ and $\mathsf{tv}(\mathscr{E})$ of type variables respectively occurring in v_p and $\mathscr{E}$ are defined as follows:*

$$\begin{array}{rcl} \mathsf{tv}(v_\mathsf{p}) & \stackrel{\text{def}}{=} & \mathsf{tv}((\!|v_\mathsf{p}|\!)) \\ \mathsf{tv}(\mathscr{E}) & \stackrel{\text{def}}{=} & \bigcup_{x\in\mathsf{dom}(\mathscr{E})} \mathsf{tv}(\mathscr{E}(x)) \ \cup \ \bigcup_{\underline{x}\in\mathsf{dom}(\mathscr{E})} \mathsf{tv}(\mathscr{E}(\underline{x})) \end{array}$$

Definition D.5. *Let e be an expression and $\mathscr{E}$ an environment. We write $e \sharp \mathscr{E}$ to mean that the polymorphic type variables in e are distinct from $\mathsf{tv}(\mathscr{E})$. Moreover, we define $\sharp v_\mathsf{p}$ and $\sharp\mathscr{E}$ recursively as follows:*

$$\begin{array}{rcl} \sharp\langle\lambda^t_{\sigma_I} x.e, \mathscr{E}, \sigma_J\rangle & \stackrel{\text{def}}{=} & \sharp\mathscr{E} \text{ and } \exists t', \vdash (\!|\langle\lambda^t_{\sigma_I} x.e, \mathscr{E}, \sigma_J\rangle|\!) : t' \text{ and } (\lambda^t_{\sigma_J\circ\sigma_I} x.e) \sharp \mathscr{E} \\ \sharp c & \stackrel{\text{def}}{=} & \mathtt{true} \\ \sharp\mathscr{E} & \stackrel{\text{def}}{=} & \forall x \in \mathsf{dom}(\mathscr{E}),\ \sharp(\mathscr{E}(x)) \text{ and } \forall \underline{x} \in \mathsf{dom}(\mathscr{E}),\ \sharp(\mathscr{E}(\underline{x})) \end{array}$$

Definition D.6. *Let $\mathbb{i}$ denote a singleton set made of the empty type-substitution $[\{\,\}]$, which is the neutral element of the composition of sets of type-substitutions. We define a transformation function $\mathsf{clos}(\cdot)$ from $(v, \mathscr{E}, \sigma_I)$ to v_p as follows:*

$$\begin{array}{rcl} \mathsf{clos}(c, \mathscr{E}, \sigma_I) & \stackrel{\text{def}}{=} & c \\ \mathsf{clos}(\lambda^t_{\sigma_J} x.e, \mathscr{E}, \sigma_I) & \stackrel{\text{def}}{=} & \langle\lambda^t_{\sigma_J} x.e, \mathscr{E}, \sigma_I\rangle \end{array}$$

For simplicity, we write $\mathsf{clos}(v)$ for $\mathsf{clos}(v, \varnothing, \mathbb{i})$.

Definition D.7. *We define the reflexive and transitive closure $\rightsquigarrow^*$ of a single-step reduction $\rightsquigarrow$ by the following two rules:*

$$\frac{}{e \rightsquigarrow^* e}\ (\textit{refl}) \qquad \frac{e_1 \rightsquigarrow e_2 \qquad e_2 \rightsquigarrow^* e_3}{e_1 \rightsquigarrow^* e_3}\ (\textit{trans})$$

Lemma D.8. *Suppose $e_1 \rightsquigarrow^* e_1'$. Then:*
(1) $e_1 e_2 \rightsquigarrow^* e_1' e_2$.
(2) $e_0 e_1 \rightsquigarrow^* e_0 e_1'$.
(3) $e_1 \in t \ \mathtt{?}\ e_2 \ \mathtt{:}\ e_3 \rightsquigarrow^* e_1' \in t \ \mathtt{?}\ e_2 \ \mathtt{:}\ e_3$.
(4) $\mathtt{let}\ \underline{x} = e_1\ \mathtt{in}\ e_2 \rightsquigarrow^* \mathtt{let}\ \underline{x} = e_1'\ \mathtt{in}\ e_2$.

Proof. By induction on a derivation of $e_1 \rightsquigarrow^* e_1'$. □

Lemma D.9. *If $e_1 \rightsquigarrow^* e_2$ and $e_2 \rightsquigarrow^* e_3$, then $e_1 \rightsquigarrow^* e_3$.*

Proof. By induction on a derivation of $e_1 \rightsquigarrow^* e_2$. □

Lemma D.10. *Let v_p be a polymorphic value. The following properties hold:*
(1) Suppose $v_\mathsf{p} \in_\mathsf{p} t$. If there exists some type t' such that $\vdash (\!|v_\mathsf{p}|\!) : t'$, then $\vdash (\!|v_\mathsf{p}|\!) : t$.
(2) $\vdash (\!|v_\mathsf{p}|\!) : t$ implies $v_\mathsf{p} \in_\mathsf{p} t$.

Proof. (1) By induction on a derivation of $\vdash (\!|v_\mathsf{p}|\!) : t'$. There are three cases to consider.

(*const*): $v_\mathsf{p} = c$ and $t' = b_c$. By definition D.2, $b_c \leq t$ and by the rule (*subsum*), we have $\vdash c : t$.
(*abstr*): $v_\mathsf{p} = \langle\lambda^s_{\sigma_I} x.e, \mathscr{E}, \sigma_J\rangle$, $(\!|v_\mathsf{p}|\!) = \lambda^s_{\sigma_J\circ\sigma_I} x.e(\!|\mathscr{E}|\!)$, and $t' = s(\sigma_J \circ \sigma_I)$.
By Definition D.2, $s(\sigma_J \circ \sigma_I) \leq t$ and by the rule (*subsum*), we have $\vdash (\!|v_\mathsf{p}|\!) : t$.
(*subsum*): We have as assumptions $\vdash (\!|v_\mathsf{p}|\!) : s$ and $s \leq t'$. By induction hypothesis, we have $\vdash (\!|v_\mathsf{p}|\!) : t$.

(2) By induction on a derivation of $\vdash (\!|v_\mathsf{p}|\!) : t$. □

Lemma D.11. *Let v_p be a polymorphic value.*
(1) Suppose $v_\mathsf{p} \notin_\mathsf{p} t$. If there exists some type t' such that $\vdash (\!|v_\mathsf{p}|\!) : t'$, then $\vdash (\!|v_\mathsf{p}|\!) : \neg t$.
(2) $\vdash (\!|v_\mathsf{p}|\!) : \neg t$ implies $v_\mathsf{p} \notin_\mathsf{p} t$.

Proof. (1) By induction on a derivation of $\vdash (\!|v_\mathsf{p}|\!) : t'$. (2) The proof is by contradiction. Suppose $v_\mathsf{p} \in_\mathsf{p} t$. Then, by Lemma D.10, we have $\vdash (\!|v_\mathsf{p}|\!) : t$, which is absurd. □

Lemma D.12. *Suppose (1) $e \sharp \mathscr{E}$ and $\sharp\mathscr{E}$; (2) $\vdash (e@\sigma_I)(\!|\mathscr{E}|\!) : t$; and (3) $\sigma_I; \mathscr{E} \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}$. Then (4) $\sharp v_\mathsf{p}$ and (5) $(e@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_\mathsf{p}|\!)$.*

Proof. By induction on a derivation of $\sigma_I; \mathscr{E} \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}$.

(PE-Const): $\sigma_I; \mathscr{E} \vdash_\mathsf{p} c \Downarrow c$ where $e = c$.
- $\sharp c$ and $(c@\sigma_I)(\!|\mathscr{E}|\!) = c \rightsquigarrow^* c$.

(PE-Var): $\sigma_I; \mathscr{E} \vdash_\mathsf{p} x \Downarrow \mathscr{E}(x)$ where $e = x$.
- From (1), we have $\sharp(\mathscr{E}(x))$.
- By Definition D.3, $(x@\sigma_I)(\!|\mathscr{E}|\!) = x(\!|\mathscr{E}|\!) = x\{(\!|\mathscr{E}(x)|\!)/_x\} = (\!|\mathscr{E}(x)|\!) \rightsquigarrow^* (\!|\mathscr{E}(x)|\!)$.

(PE-Closure): $\sigma_I; \mathscr{E} \vdash_\mathsf{p} \lambda^s_{\sigma_J} x.e_0 \Downarrow \langle\lambda^s_{\sigma_J} x.e_0, \mathscr{E}, \sigma_I\rangle$ where $e = \lambda^s_{\sigma_J} x.e_0$.
- By Definition D.3, $((\lambda^s_{\sigma_J} x.e_0)@\sigma_I)(\!|\mathscr{E}|\!) = \lambda^s_{\sigma_I\circ\sigma_J} x.e_0(\!|\mathscr{E}|\!) = (\!|\langle\lambda^s_{\sigma_J} x.e_0, \mathscr{E}, \sigma_I\rangle|\!) \rightsquigarrow^* (\!|\langle\lambda^s_{\sigma_J} x.e_0, \mathscr{E}, \sigma_I\rangle|\!)$.
- Moreover, from (1) and (2), we have $\sharp\langle\lambda^s_{\sigma_J} x.e_0, \mathscr{E}, \sigma_I\rangle$.

(PE-APPLY): $e = e_1 e_2$.

$$\frac{(a)\ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \Downarrow \langle \lambda^s_{\sigma_K} x.e_0, \mathscr{E}', \sigma_H \rangle \quad s = \wedge_{l \in L} s_l \to s'_l \qquad (b)\ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}0} \\ \sigma_J = \sigma_H \circ \sigma_K \quad (c)\ P = \{j \in J \mid \exists l \in L : v_{\mathsf{p}0} \in_{\mathsf{p}} s_l \sigma_j\} \qquad (d)\ \sigma_P; \mathscr{E}', x \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_0 \Downarrow v_{\mathsf{p}}}{\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 e_2 \Downarrow v_{\mathsf{p}}}$$

- From (2), $(e_1@\sigma_I)(\!|\mathscr{E}|\!)$ and $(e_2@\sigma_I)(\!|\mathscr{E}|\!)$ are also well-typed.
- By IH on (a), $\sharp\langle \lambda^s_{\sigma_K} x.e_0, \mathscr{E}', \sigma_H \rangle$ and $(e_1@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|\langle \lambda^s_{\sigma_K} x.e_0, \mathscr{E}', \sigma_H \rangle|\!) = \lambda^s_{\sigma_H \circ \sigma_K} x.e_0(\!|\mathscr{E}'|\!) = \lambda^s_{\sigma_J} x.e_0(\!|\mathscr{E}'|\!)$ where $\mathsf{dom}(\sigma_J) \cap \mathsf{tv}(\mathscr{E}') = \emptyset$.
- By IH on (b), $\sharp v_{\mathsf{p}0}$ and $(e_2@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}0}|\!)$.
- By Lemmas D.8 and D.9, $((e_1 e_2)@\sigma_I)(\!|\mathscr{E}|\!) = ((e_1@\sigma_I)(\!|\mathscr{E}|\!))((e_2@\sigma_I)(\!|\mathscr{E}|\!)) \rightsquigarrow^* (\lambda^s_{\sigma_J} x.e_0(\!|\mathscr{E}'|\!))(\!|v_{\mathsf{p}0}|\!) \rightsquigarrow ((e_0(\!|\mathscr{E}'|\!))@\sigma_{P'})\{(\!|v_{\mathsf{p}0}|\!)/x\}$ where
$$(e) \quad P' = \{j \in J \mid \exists l \in L, \vdash (\!|v_{\mathsf{p}0}|\!) : s_l \sigma_j\}$$
Furthermore, by Theorem B.15 with (2), $((e_0(\!|\mathscr{E}'|\!))@\sigma_{P'})\{(\!|v_{\mathsf{p}0}|\!)/x\}$ is also well-typed.
- By $\sharp\mathscr{E}'$ and $\sharp v_{\mathsf{p}0}$, we have $\sharp(\mathscr{E}' x \mapsto v_{\mathsf{p}0})$ and by assuming α-conversion, we have $e_0 \sharp (\mathscr{E}' x \mapsto v_{\mathsf{p}0})$.
- By Lemma D.10 with (c) and (e), $P = P'$. Moreover, by Lemma B.5 with $\mathsf{dom}(\sigma_J) \cap \mathsf{tv}(\mathscr{E}') = \emptyset$, we have $(e_0(\!|\mathscr{E}'|\!))@\sigma_{P'} = (e_0(\!|\mathscr{E}'|\!))@\sigma_P = (e_0@\sigma_P)(\!|\mathscr{E}'|\!)$.
- By IH on (d), $(e_0@\sigma_P)(\!|\mathscr{E}', x \mapsto v_{\mathsf{p}0}|\!) = (e_0@\sigma_P)(\!|\mathscr{E}'|\!)\{(\!|v_{\mathsf{p}0}|\!)/x\} = ((e_0(\!|\mathscr{E}'|\!))@\sigma_P)\{(\!|v_{\mathsf{p}0}|\!)/x\} \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$ and $\sharp\, v_{\mathsf{p}}$.
- Finally, by Lemma D.9, $((e_1 e_2)@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$.

(PE-TYPE CASE T): $e = e_1 \in s \,?\, e_2 : e_3$.

$$\frac{(a)\ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}0} \qquad (b)\ v_{\mathsf{p}0} \in_{\mathsf{p}} s \qquad (c)\ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}}{\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \in s \,?\, e_2 : e_3 \Downarrow v_{\mathsf{p}}}$$

- By definition, $((e_1 \in s \,?\, e_2 : e_3)@\sigma_I)(\!|\mathscr{E}|\!) = (e_1@\sigma_I)(\!|\mathscr{E}|\!) \in s \,?\, (e_2@\sigma_I)(\!|\mathscr{E}|\!) : (e_3@\sigma_I)(\!|\mathscr{E}|\!)$, and each sub-expression $(e_i@\sigma_I)(\!|\mathscr{E}|\!)$ $(i = 1, 2, 3)$ is well-typed from the assumption (2).
- By IH on (a), $(e_1@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}0}|\!)$ and $\sharp v_{\mathsf{p}0}$.
- Then, by Lemma D.10 with (b), we have $\vdash (\!|v_{\mathsf{p}0}|\!) : s$.
- By IH on (c), $(e_2@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$ and $\sharp v_{\mathsf{p}}$.
- Finally, by Lemmas D.8 and D.9, we have $(e_1@\sigma_I)(\!|\mathscr{E}|\!) \in s \,?\, (e_2@\sigma_I)(\!|\mathscr{E}|\!) : (e_3@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}0}|\!) \in s \,?\, (e_2@\sigma_I)(\!|\mathscr{E}|\!) : (e_3@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow (e_2@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$, thus proving (5).

(PE-TYPE CASE F): Similar to the case for the rule (PE-TYPE CASE T), except that we exploit Lemma D.11 instead of Lemma D.10.

(PE-INSTANCE): $e = e_0 \sigma_J$.

$$\frac{(a)\ \sigma_I \circ \sigma_J; \mathscr{E} \vdash_{\mathsf{p}} e_0 \Downarrow v}{\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_0 \sigma_J \Downarrow v}$$

- By the assumption (1), we have $e_0 \sharp \mathscr{E}$.
- By IH on (a), $(e_0@(\sigma_I \circ \sigma_J))(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$ and $\sharp v_{\mathsf{p}}$.
- Finally, by definition, we have $((e\sigma_J)@\sigma_I)(\!|\mathscr{E}|\!) = (e@(\sigma_I \circ \sigma_J))(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$, thus proving (5).

(PE-PVAR$_c$): $e = \underline{x}$.

$$\frac{\mathscr{E}(\underline{x}) = c}{\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} \underline{x} \Downarrow c}$$

- $\sharp c$ and $(\underline{x}@\sigma_I)(\!|\mathscr{E}|\!) = c \rightsquigarrow^* c$.

(PE-PVAR$_f$): $e = \underline{x}$.

$$\frac{\mathscr{E}(\underline{x}) = \langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}', \sigma_J \rangle}{\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} \underline{x} \Downarrow \langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}', \sigma_I \circ \sigma_J \rangle}$$

- $(\underline{x}@\sigma_I)(\!|\mathscr{E}|\!) = (\underline{x}[\sigma_i]_{i \in I})(\!|\mathscr{E}|\!) = (\!|\mathscr{E}(\underline{x})|\!)[\sigma_i]_{i \in I} = (\!|\langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}', \sigma_J \rangle|\!)[\sigma_i]_{i \in I} \rightsquigarrow^* \lambda^s_{\sigma_I \circ (\sigma_J \circ \sigma_K)} y.e_0(\!|\mathscr{E}'|\!) = (\!|\langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}', \sigma_I \circ \sigma_J \rangle|\!)$.

(PE-LET): $e = \mathtt{let}\ \underline{x} = e_1\ \mathtt{in}\ e_2$.

$$\frac{(a)\ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}0} \qquad (b)\ \sigma_I; \mathscr{E}, \underline{x} \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}}{\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} \mathtt{let}\ \underline{x} = e_1\ \mathtt{in}\ e_2 \Downarrow v_{\mathsf{p}}}$$

- By IH on (a), $(e_1@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* (\!|v_{\mathsf{p}0}|\!)$.
- By IH on (b), $(e_2@\sigma_I)(\!|\mathscr{E}, \underline{x} \mapsto v_{\mathsf{p}0}|\!) = (e_2@\sigma_I)(\!|\mathscr{E}|\!)\{(\!|v_{\mathsf{p}0}|\!)/\underline{x}\} \rightsquigarrow^* (\!|v_{\mathsf{p}}|\!)$.
- By Lemmas D.8 and D.9, $((\mathtt{let}\ \underline{x} = e_1\ \mathtt{in}\ e_2)@\sigma_I)(\!|\mathscr{E}|\!) = \mathtt{let}\ \underline{x} = (e_1@\sigma_I)(\!|\mathscr{E}|\!)\ \mathtt{in}\ (e_2@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow^* \mathtt{let}\ \underline{x} = (\!|v_{\mathsf{p}0}|\!)\ \mathtt{in}\ (e_2@\sigma_I)(\!|\mathscr{E}|\!) \rightsquigarrow (e_2@\sigma_I)(\!|\mathscr{E}|\!)\{(\!|v_{\mathsf{p}0}|\!)/\underline{x}\} \rightsquigarrow^* v_{\mathsf{p}}$.

□

Definition D.13. *$v_{\mathsf{p}} \equiv_{\mathsf{c}} v_{\mathsf{p}}'$ if and only if $(\!|v_{\mathsf{p}}|\!) = (\!|v_{\mathsf{p}}'|\!)$.*

Lemma D.14. *Suppose $v_{\mathsf{p}} \equiv_{\mathsf{c}} v_{\mathsf{p}}'$. Then $v_{\mathsf{p}} \in_{\mathsf{p}} t$ if and only if $v_{\mathsf{p}}' \in_{\mathsf{p}} t$.*

Proof. Suppose $v_\mathsf{p} = c$. Then ${v_\mathsf{p}}' = c$, which completes the proof. Now suppose $v_\mathsf{p} = \langle \lambda^s_{\sigma_I} x.e, \mathscr{E}, \sigma_J \rangle$. Then ${v_\mathsf{p}}' = \langle \lambda^s_{\sigma_K} x.e', \mathscr{E}', \sigma_H \rangle$ where $\sigma_J \circ \sigma_I = \sigma_H \circ \sigma_K$ and $e(\!|\mathscr{E}|\!) = e'(\!|\mathscr{E}'|\!)$. By the definition of $\in_\mathsf{p}$, we complete the proof. □

Lemma D.15. *Suppose (1)* $(e_1@\sigma_I)(\!|\mathscr{E}_1|\!) = (e_2@\sigma_I)(\!|\mathscr{E}_2|\!)$ *and (2)* $\sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_1 \Downarrow v_\mathsf{p}$. *Then, there exists* ${v_\mathsf{p}}'$ *such that* $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_2 \Downarrow {v_\mathsf{p}}'$ *and* $v_\mathsf{p} \equiv_\mathsf{c} {v_\mathsf{p}}'$.

Proof. By induction on a derivation of $\sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_1 \Downarrow v_\mathsf{p}$.

(PE-CONST): $e_1 = v_\mathsf{p} = c$.
- From (1), e_2 is either c, x, or $\underline{x}$. If $e_2 = c$, then the proof is easy.
- If $e_2 = x$, then ${v_\mathsf{p}}' = \mathscr{E}_2(x)$. From (1) we have $(e_2@\sigma_I)(\!|\mathscr{E}_2|\!) = (x@\sigma_I)(\!|\mathscr{E}_2|\!) = (\!|\mathscr{E}_2(x)|\!) = c$, thus completing the proof. The proof for $e_2 = \underline{x}$ is similar.

(PE-VAR): $e_1 = x$ and $v_\mathsf{p} = \mathscr{E}_1(x)$.
- From (1), e_2 is either y, c, $\lambda^s_{\sigma_K} z.e$, or $\underline{x}$.
- If $e_2 = c$, then ${v_\mathsf{p}}' = c$. Moreover, from (1) we have $(e_1@\sigma_I)(\!|\mathscr{E}_1|\!) = (x@\sigma_I)(\!|\mathscr{E}_1|\!) = (\!|\mathscr{E}_1(x)|\!) = c$, thus completing the proof.
- If $e_2 = y$, then ${v_\mathsf{p}}' = \mathscr{E}_2(y)$. From (1) we have $\mathscr{E}_1(x) = \mathscr{E}_2(y)$, thus completing the proof.
- If $e_2 = \lambda^s_{\sigma_K} z.e$, then ${v_\mathsf{p}}' = \langle e_2, \mathscr{E}_2, \sigma_I \rangle$. From (1) we have $\mathscr{E}_1(x) = (e_2@\sigma_I)(\!|\mathscr{E}_2|\!) = (\!|{v_\mathsf{p}}'|\!)$, thus completing the proof.
- Now assume $e_2 = \underline{x}$. If $\mathscr{E}_2(\underline{x}) = c$, then from (1) we also have $\mathscr{E}_1(x) = c$, thus completing the proof. Otherwise, $\mathscr{E}_2(\underline{x}) = \langle \lambda^s_{\sigma_K} z.e, \mathscr{E}', \sigma_H \rangle$ and ${v_\mathsf{p}}' = \langle \lambda^s_{\sigma_K} z.e, \mathscr{E}', \sigma_I \circ \sigma_H \rangle$. From (1), we have $\mathscr{E}_1(x) = (\underline{x}@\sigma_I)(\!|\mathscr{E}_2|\!) = (\!|\mathscr{E}_2(\underline{x})|\!)@\sigma_I = (\!|{v_\mathsf{p}}'|\!)$, thus completing the proof.

(PE-CLOSURE): $e_1 = \lambda^s_{\sigma_K} z.e$ and $v_\mathsf{p} = \langle e_1, \mathscr{E}_1, \sigma_I \rangle$.
- From (1), e_2 is either x, $\lambda^s_{\sigma_J} z.e'$, or $\underline{x}$.
- If $e_2 = x$, then the proof is similar to the third case for (PE-VAR).
- If $e_2 = \lambda^s_{\sigma_J} z.e'$, then ${v_\mathsf{p}}' = \langle \lambda^s_{\sigma_J} z.e', \mathscr{E}_2, \sigma_I \rangle$. From (1), we have $(\!|v_\mathsf{p}|\!) = (\!|{v_\mathsf{p}}'|\!)$, thus completing the proof.
- If $e_2 = \underline{x}$, then $\mathscr{E}_2(\underline{x}) = \langle \lambda^s_{\sigma_J} z.e', \mathscr{E}', \sigma_H \rangle$ and ${v_\mathsf{p}}' = \langle \lambda^s_{\sigma_J} z.e', \mathscr{E}', \sigma_I \circ \sigma_H \rangle$. From (1), we have $(\!|v_\mathsf{p}|\!) = (e_1@\sigma_I)(\!|\mathscr{E}_1|\!) = (\underline{x}@\sigma_I)(\!|\mathscr{E}_2|\!) = (\!|\mathscr{E}_2(\underline{x})|\!)@\sigma_I = (\!|{v_\mathsf{p}}'|\!)$, thus completing the proof.

(PE-APPLY): $e_1 = e_{11} e_{12}$.
- From (1), we have $e_2 = e_{21} e_{22}$ and $(e_{1l}@\sigma_I)(\!|\mathscr{E}_1|\!) = (e_{2l}@\sigma_I)(\!|\mathscr{E}_2|\!)$ where $l \in \{1, 2\}$.
- From (2), we have the following assumptions:
  (a) $\sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_{11} \Downarrow \langle \lambda^s_{\sigma_K} x.e, \mathscr{E}, \sigma_H \rangle$ where $s = \wedge_{i \in I} s_i \to s'_i$ and $\mathsf{tv}(\mathscr{E}) \cap \mathsf{dom}(\sigma_H \circ \sigma_K) = \emptyset$;
  (b) $\sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_{12} \Downarrow v_{\mathsf{p}_0}$;
  (c) $\sigma_J = \sigma_H \circ \sigma_K$ and $P = \{j \in J \mid \exists i \in I : v_{\mathsf{p}_0} \in_\mathsf{p} s_i \sigma_j\}$; and
  (d) $\sigma_P; \mathscr{E}, x \mapsto v_{\mathsf{p}_0} \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}$.
- By IH on (a), $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_{21} \Downarrow \langle \lambda^s_{\sigma_{K'}} x.e', \mathscr{E}', \sigma_{H'} \rangle$ where $\sigma_H \circ \sigma_K = \sigma_{H'} \circ \sigma_{K'}$ and $e(\!|\mathscr{E}|\!) = e'(\!|\mathscr{E}'|\!)$. Moreover, $\mathsf{tv}(\mathscr{E}') \cap \mathsf{dom}(\sigma_{H'} \circ \sigma_{K'}) = \emptyset$.
- By IH on (b), $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_{22} \Downarrow {v_{\mathsf{p}_0}}'$ and $v_{\mathsf{p}_0} \equiv_\mathsf{c} {v_{\mathsf{p}_0}}'$.
- By Lemma D.14 with $v_{\mathsf{p}_0} \equiv_\mathsf{c} {v_{\mathsf{p}_0}}'$ and $\sigma_J = \sigma_H \circ \sigma_K = \sigma_{H'} \circ \sigma_{K'}$, we have $P = \{j \in J \mid \exists i \in I : v_{\mathsf{p}_0} \in_\mathsf{p} s_i \sigma_j\} = \{j \in J \mid \exists i \in I : {v_{\mathsf{p}_0}}' \in_\mathsf{p} s_i \sigma_j\}$.
- From $e(\!|\mathscr{E}|\!) = e'(\!|\mathscr{E}'|\!)$ and $v_{\mathsf{p}_0} \equiv_\mathsf{c} {v_{\mathsf{p}_0}}'$ and $\mathsf{tv}(\mathscr{E}) \cap \sigma_P = \emptyset$ and $\mathsf{tv}(\mathscr{E}') \cap \sigma_P = \emptyset$, we have $(e@\sigma_P)(\!|\mathscr{E}, x \mapsto v_{\mathsf{p}_0}|\!) = (e'@\sigma_P)(\!|\mathscr{E}', x \mapsto {v_{\mathsf{p}_0}}'|\!)$.
- Now, by IH on (d), $\sigma_P; \mathscr{E}', x \mapsto {v_{\mathsf{p}_0}}' \vdash_\mathsf{p} e' \Downarrow {v_\mathsf{p}}'$ and $v_\mathsf{p} \equiv_\mathsf{c} {v_\mathsf{p}}'$.
- Finally, we have $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_2 \Downarrow {v_\mathsf{p}}'$ by the rule (PE-APPLY), thus completing the proof.

(PE-TYPE CASE T): $e_1 = e_{11} \in s \,?\, e_{12} : e_{13}$.
- From (1), we have $e_2 = e_{21} \in s \,?\, e_{22} : e_{23}$ and $(e_{1l}@\sigma_I)(\!|\mathscr{E}_1|\!) = (e_{2l}@\sigma_I)(\!|\mathscr{E}_2|\!)$ where $l \in \{1, 2, 3\}$.
- From (2), we have the following assumptions:
  (a) $\sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_{11} \Downarrow v_{\mathsf{p}_0}$ (b) $v_{\mathsf{p}_0} \in_\mathsf{p} s$ (c) $\sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_{12} \Downarrow v_\mathsf{p}$
- By IH on (a), $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_{21} \Downarrow {v_{\mathsf{p}_0}}'$ and $v_{\mathsf{p}_0} \equiv_\mathsf{c} {v_{\mathsf{p}_0}}'$.
- By Lemma D.14 with (b), we have ${v_{\mathsf{p}_0}}' \in_\mathsf{p} s$.
- By IH on (c), $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_{22} \Downarrow {v_\mathsf{p}}'$ and $v_\mathsf{p} \equiv_\mathsf{c} {v_\mathsf{p}}'$.
- Finally, we have $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_2 \Downarrow {v_\mathsf{p}}'$ by the rule (PE-TYPE CASE T), thus completing the proof.

(PE-TYPE CASE F): $e_1 = e_{11} \in s \,?\, e_{12} : e_{13}$.
- Similar to the case for the rule (PE-TYPE CASE F).

(PE-INSTANCE): $e_1 = e\sigma_J$.
- From (1), we have $e_2 = e'\sigma_J$ and $(e@(\sigma_I \circ \sigma_J))(\!|\mathscr{E}_1|\!) = (e'@(\sigma_I \circ \sigma_J))(\!|\mathscr{E}_2|\!)$.
- From (2), we have the following assumption: (a) $\sigma_I \circ \sigma_J; \mathscr{E}_1 \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}$.
- By IH on (a), $\sigma_I \circ \sigma_J; \mathscr{E}_2 \vdash_\mathsf{p} e' \Downarrow {v_\mathsf{p}}'$ and $v_\mathsf{p} \equiv_\mathsf{c} {v_\mathsf{p}}'$.
- Finally, we have $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_2 \Downarrow {v_\mathsf{p}}'$ by the rule (PE-INSTANCE), thus completing the proof.

(PE-PVAR$_c$): $e_1 = \underline{x}$ and $v_\mathsf{p} = c$.

- Similar to the case for (PE-CONST).

(PE-PVAR$_f$): $e_1 = \underline{x}$, $\mathscr{E}_1(\underline{x}) = \langle \lambda^s_{\sigma_K} z.e, \mathscr{E}, \sigma_H \rangle$ and $v_\mathsf{p} = \langle \lambda^s_{\sigma_K} z.e, \mathscr{E}, \sigma_I \circ \sigma_H \rangle$.

- Similar to the case for (PE-VAR).

(PE-LET): $e_1 = \texttt{let}\ \underline{x}\ \texttt{=}\ e_{11}\ \texttt{in}\ e_{12}$.

- From (1), we have $e_2 = \texttt{let}\ \underline{x}\ \texttt{=}\ e_{21}\ \texttt{in}\ e_{22}$. Moreover, $(e_{11}@\sigma_I)[\![\mathscr{E}_1]\!] = (e_{21}@\sigma_I)[\![\mathscr{E}_2]\!]$ and $(e_{12}@\sigma_I)[\![\mathscr{E}_1]\!] = (e_{22}@\sigma_I)[\![\mathscr{E}_2]\!]$.
- From (2), we have the following assumptions:
$$(a)\ \ \sigma_I; \mathscr{E}_1 \vdash_\mathsf{p} e_{11} \Downarrow v_{\mathsf{p}0} \qquad (b)\ \ \sigma_I; \mathscr{E}_1, \underline{x} \mapsto v_{\mathsf{p}0} \vdash_\mathsf{p} e_{12} \Downarrow v_\mathsf{p}$$
- By IH on (a), $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_{21} \Downarrow v_{\mathsf{p}0}'$ and $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$.
- By IH on (b), $\sigma_I; \mathscr{E}_2, \underline{x} \mapsto v_{\mathsf{p}0}' \vdash_\mathsf{p} e_{22} \Downarrow v_\mathsf{p}'$ and $v_\mathsf{p} \equiv_\mathsf{c} v_\mathsf{p}'$.
- We conclude $\sigma_I; \mathscr{E}_2 \vdash_\mathsf{p} e_2 \Downarrow v_\mathsf{p}'$ by the rule (PE-LET).

□

Lemma D.16. *Let v be a well-typed closed value such that $\mathsf{dom}(\sigma_I) \cap \mathsf{tv}(v) = \emptyset$. Suppose (1) $\sigma_I; \mathscr{E} \vdash_\mathsf{p} (e@\sigma_L)\{v/x\} \Downarrow v_\mathsf{p}$. Then $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}'$ and $v_\mathsf{p} \equiv_\mathsf{c} v_\mathsf{p}'$.*

Proof. By induction on a derivation of $\sigma_I; \mathscr{E} \vdash_\mathsf{p} (e@\sigma_L)\{v/x\} \Downarrow v_\mathsf{p}$.

(PE-CONST): $e = c$ and $v_\mathsf{p} = v_\mathsf{p}' = c$.

(PE-VAR): There are two cases to consider.

- Suppose $e = x$. Then $(e@\sigma_L)\{v/x\} = v$ and $v_\mathsf{p} = \mathsf{clos}(v, \mathscr{E}, \sigma_I) = \mathsf{clos}(v) = v_\mathsf{p}'$.
- Now suppose $(e = y \ \wedge\ y \neq x)$. Then $(e@\sigma_L)\{v/x\} = y$ and $v_\mathsf{p} = v_\mathsf{p}' = \mathscr{E}(y)$.

(PE-CLOSURE): $e = \lambda^s_{\sigma_K} y.e_0$, $(e@\sigma_L)\{v/x\} = \lambda^s_{\sigma_L \circ \sigma_K} y.(e_0\{v/x\})$, and $v_\mathsf{p} = \langle \lambda^s_{\sigma_L \circ \sigma_K} y.(e_0\{v/x\}), \mathscr{E}, \sigma_I \rangle$. Moreover, $v_\mathsf{p}' = \langle e, (\mathscr{E}, x \mapsto \mathsf{clos}(v)), \sigma_I \circ \sigma_L \rangle$ and thus $v_\mathsf{p} \equiv_\mathsf{c} v_\mathsf{p}'$.

(PE-APPLY): $e = e_1 e_2$ and $(e@\sigma_L)\{v/x\} = ((e_1@\sigma_L)\{v/x\})((e_2@\sigma_L)\{v/x\})$.

- From (1), we have the following assumptions:
  (a) $\sigma_I; \mathscr{E} \vdash_\mathsf{p} (e_1@\sigma_L)\{v/x\} \Downarrow \langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}_0, \sigma_H \rangle$ where $s = \wedge_{i \in I} s_i \to s_i'$ and $\mathsf{tv}(\mathscr{E}_0) \cap \mathsf{dom}(\sigma_H \circ \sigma_K) = \emptyset$;
  (b) $\sigma_I; \mathscr{E} \vdash_\mathsf{p} (e_2@\sigma_L)\{v/x\} \Downarrow v_{\mathsf{p}0}$;
  (c) $\sigma_J = \sigma_H \circ \sigma_K$ and $P = \{j \in J \mid \exists i \in I : v_{\mathsf{p}0} \in_\mathsf{p} s_i\sigma_j\}$; and
  (d) $\sigma_P; \mathscr{E}_0, y \mapsto v_{\mathsf{p}0} \vdash_\mathsf{p} e_0 \Downarrow v_\mathsf{p}$.
- By IH on (a), $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_1 \Downarrow \langle \lambda^s_{\sigma_{K'}} y.e_0', \mathscr{E}_0', \sigma_{H'} \rangle$ where $\sigma_H \circ \sigma_K = \sigma_{H'} \circ \sigma_{K'}$ and $e_0[\![\mathscr{E}_0]\!] = e_0'[\![\mathscr{E}_0']\!]$. Moreover, $\mathsf{tv}(\mathscr{E}_0') \cap \mathsf{dom}(\sigma_{H'} \circ \sigma_{K'}) = \emptyset$.
- By IH on (b), $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_2 \Downarrow v_{\mathsf{p}0}'$ and $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$.
- By Lemma D.14 with $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$ and $\sigma_H \circ \sigma_K = \sigma_{H'} \circ \sigma_{K'}$, we have $P = \{j \in J \mid \exists i \in I : v_{\mathsf{p}0} \in_\mathsf{p} s_i\sigma_j\} = \{j \in J \mid \exists i \in I : v_{\mathsf{p}0}' \in_\mathsf{p} s_i\sigma_j\}$.
- From $e_0[\![\mathscr{E}_0]\!] = e_0'[\![\mathscr{E}_0']\!]$ and $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$ and $\mathsf{tv}(\mathscr{E}_0) \cap \sigma_P = \emptyset$ and $\mathsf{tv}(\mathscr{E}_0') \cap \sigma_P = \emptyset$, we have $(e_0@\sigma_P)[\![\mathscr{E}_0, y \mapsto v_{\mathsf{p}0}]\!] = (e_0'@\sigma_P)[\![\mathscr{E}_0', y \mapsto v_{\mathsf{p}0}']\!]$.
- By Lemma D.15 with (d), we have $\sigma_P; \mathscr{E}_0', y \mapsto v_{\mathsf{p}0}' \vdash_\mathsf{p} e_0' \Downarrow v_\mathsf{p}'$ and $v_\mathsf{p} \equiv_\mathsf{c} v_\mathsf{p}'$.
- By the rule (PE-APPLY), we have $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}'$, thus completing the proof.

(PE-TYPE CASE T): $e = e_1 \in s\ \texttt{?}\ e_2\ \texttt{:}\ e_3$.

- $(e@\sigma_L)\{v/x\} = (e_1@\sigma_L)\{v/x\} \in s\ \texttt{?}\ (e_2@\sigma_L)\{v/x\}\ \texttt{:}\ (e_3@\sigma_L)\{v/x\}$.
- From (1), we have the following assumptions:
$$(a)\ \ \sigma_I; \mathscr{E} \vdash_\mathsf{p} (e_1@\sigma_L)\{v/x\} \Downarrow v_{\mathsf{p}0} \qquad (b)\ \ v_{\mathsf{p}0} \in_\mathsf{p} s \qquad (c)\ \ \sigma_I; \mathscr{E} \vdash_\mathsf{p} (e_2@\sigma_L)\{v/x\} \Downarrow v_\mathsf{p}$$
- By IH on (a), $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_1 \Downarrow v_{\mathsf{p}0}'$ and $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$.
- By Lemma D.14 with (b) and $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$, we have $v_{\mathsf{p}0}' \in_\mathsf{p} s$.
- By IH on (c), $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_2 \Downarrow v_\mathsf{p}'$ and $v_\mathsf{p} \equiv_\mathsf{c} v_\mathsf{p}'$.
- We conclude $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e \Downarrow v_\mathsf{p}'$ by the rule (PE-TYPE CASE T).

(PE-TYPE CASE F): Similar to the case for the rule (PE-TYPE CASE T).

(PE-INSTANCE): $e = e_0\sigma_J$ and $e' = ((e_0\sigma_J)@\sigma_L)\{v/x\} = (e_0@(\sigma_L \circ \sigma_J))\{v/x\}$.

- From (1), we have $\sigma_I; \mathscr{E} \vdash_\mathsf{p} (e_0@(\sigma_L \circ \sigma_J))\{v/x\} \Downarrow v_\mathsf{p}$.
- By IH, we have $\sigma_I \circ (\sigma_L \circ \sigma_J); \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_0 \Downarrow v_\mathsf{p}'$ and $v_\mathsf{p} \equiv_\mathsf{c} v_\mathsf{p}'$.
- We conclude $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_0\sigma_J \Downarrow v_\mathsf{p}'$ by the rule (PE-INSTANCE).

(PE-VAR$_c$): $e = \underline{x}$ and $v_\mathsf{p} = v_\mathsf{p}' = c$.

(PE-VAR$_f$): $e = \underline{x}$, $\mathscr{E}(\underline{x}) = \langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}', \sigma_H \rangle$, and $v_\mathsf{p} = v_\mathsf{p}' = \langle \lambda^s_{\sigma_K} y.e_0, \mathscr{E}', (\sigma_I \circ \sigma_L) \circ \sigma_H \rangle$.

(PE-LET): $e = \texttt{let}\ \underline{x}\ \texttt{=}\ e_1\ \texttt{in}\ e_2$.

- $(e@\sigma_L)\{v/x\} = \texttt{let}\ \underline{x}\ \texttt{=}\ (e_1@\sigma_L)\{v/x\}\ \texttt{in}\ (e_2@\sigma_L)\{v/x\}$.
- From (1), we have the following assumptions:
$$(a)\ \ \sigma_I; \mathscr{E} \vdash_\mathsf{p} (e_1@\sigma_L)\{v/x\} \Downarrow v_{\mathsf{p}0} \qquad (b)\ \ \sigma_I; \mathscr{E}, \underline{x} \mapsto v_{\mathsf{p}0} \vdash_\mathsf{p} (e_2@\sigma_L)\{v/x\} \Downarrow v_\mathsf{p}$$
- By IH on (a), $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_\mathsf{p} e_1 \Downarrow v_{\mathsf{p}0}'$ and $v_{\mathsf{p}0} \equiv_\mathsf{c} v_{\mathsf{p}0}'$.

- By IH on (b), $\sigma_I \circ \sigma_L; \mathscr{E}, \underline{x} \mapsto v_{\mathsf{p}0}, x \mapsto \mathsf{clos}(v) \vdash_{\mathsf{p}} e_2 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- We conclude $\sigma_I \circ \sigma_L; \mathscr{E}, x \mapsto \mathsf{clos}(v) \vdash_{\mathsf{p}} e \Downarrow {v_{\mathsf{p}}}'$ by the rule (PE-LET).

□

Lemma D.17. *Let v be a well-typed closed value such that $\mathsf{dom}(\sigma_I) \cap \mathsf{tv}(v) = \emptyset$. Suppose (1) $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e\{v/\underline{x}\} \Downarrow v_{\mathsf{p}}$. Then $\sigma_I; \mathscr{E}, \underline{x} \mapsto \mathsf{clos}(v) \vdash_{\mathsf{p}} e \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.*

Proof. Similar to the proof for Lemma D.16. □

Lemma D.18. *If (1) $\vdash e : t$, (2) $e \rightsquigarrow e'$, and (3) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e' \Downarrow v_{\mathsf{p}}$, then $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.*

Proof. By induction on a derivation of $e \rightsquigarrow e'$.

($e = e_1 e_2$):

(I) $\dfrac{e_1 \rightsquigarrow e_1'}{e_1 e_2 \rightsquigarrow e_1' e_2}$ where $e' = e_1' e_2$:
- From (3), we have the following assumptions:
  (a) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1' \Downarrow \langle \lambda^s_{\sigma_K} x.e_0, \mathscr{E}, \sigma_H \rangle$ where $s = \wedge_{l \in L} s_l \to s_l'$ and $\mathsf{tv}(\mathscr{E}) \cap \mathsf{dom}(\sigma_H \circ \sigma_K) = \emptyset$;
  (b) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}0}$;
  (c) $\sigma_J = \sigma_H \circ \sigma_K$ and $P = \{j \in J \mid \exists l \in L : v_{\mathsf{p}0} \in_{\mathsf{p}} s_l \sigma_j\}$; and
  (d) $\sigma_P; \mathscr{E}, x \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_0 \Downarrow v_{\mathsf{p}}$.
- By IH on $e_1 \rightsquigarrow e_1'$ with (a), $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow \langle \lambda^s_{\sigma_{K'}} x.e_0', \mathscr{E}', \sigma_{H'} \rangle$ where $\sigma_H \circ \sigma_K = \sigma_{H'} \circ \sigma_{K'}$ and $\mathsf{tv}(\mathscr{E}') \cap \mathsf{dom}(\sigma_{H'} \circ \sigma_{K'}) = \emptyset$ and $e_0 (\!| \mathscr{E} |\!) = e_0' (\!| \mathscr{E}' |\!)$.
- From $e_0 (\!| \mathscr{E} |\!) = e_0' (\!| \mathscr{E}' |\!)$, $\mathsf{tv}(\mathscr{E}) \cap \mathsf{dom}(\sigma_P) = \emptyset$, and $\mathsf{tv}(\mathscr{E}') \cap \mathsf{dom}(\sigma_P) = \emptyset$, we have $(e_0 @ \sigma_P)(\!| \mathscr{E}, x \mapsto v_{\mathsf{p}0} |\!) = (e_0' @ \sigma_P)(\!| \mathscr{E}', x \mapsto v_{\mathsf{p}0} |\!)$.
- By Lemma D.15 with (d), $\sigma_P; \mathscr{E}', x \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_0' \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- Finally, by the rule (PE-APPLY), we conclude $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 e_2 \Downarrow {v_{\mathsf{p}}}'$.

(II) $\dfrac{e_2 \rightsquigarrow e_2'}{e_1 e_2 \rightsquigarrow e_1 e_2'}$ where $e' = e_1 e_2'$:
- Similar to the case for (I).

(III) $e_1 = \lambda^{\wedge_{i \in I} s_i \to s_i'}_{\sigma_J} x.e_0$, $e_2 = v$, $e' = (e_0 @ \sigma_P)\{v/x\}$, and $P = \{j \in J \mid \exists i \in I, \vdash v : s_i \sigma_j\}$:
- We have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow \mathsf{clos}(e_1)$ and $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} v \Downarrow \mathsf{clos}(v)$.
- By Lemma D.10, $P = \{j \in J \mid \exists i \in I, \vdash v : s_i \sigma_j\} = \{j \in J \mid \exists i \in I,\ \mathsf{clos}(v) \in_{\mathsf{p}} s_i \sigma_j\}$.
- By Lemma D.16 with (3), we have $\sigma_P; x \mapsto \mathsf{clos}(v) \vdash_{\mathsf{p}} e_0 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- Finally, by the rule (PE-APPLY), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 e_2 \Downarrow {v_{\mathsf{p}}}'$.

($e = e_1 \in s \,?\, e_2 : e_3$):

(I) $\dfrac{e_1 \rightsquigarrow e_1'}{e \rightsquigarrow e_1' \in s \,?\, e_2 : e_3}$ where $e' = e_1' \in s \,?\, e_2 : e_3$:
- From (3), there are two cases to consider. Here we consider only the case for (PE-TYPE CASE T). The case for (PE-TYPE CASE F) is similar.
- We have the following assumptions:
  (a) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1' \Downarrow v_{\mathsf{p}0}$ (b) $v_{\mathsf{p}0} \in_{\mathsf{p}} s$ (c) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}$
- By IH on $e_1 \rightsquigarrow e_1'$ with (a), $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow {v_{\mathsf{p}0}}'$ and $v_{\mathsf{p}0} \equiv_{\mathsf{c}} {v_{\mathsf{p}0}}'$.
- By Lemma D.14 with (b), we have ${v_{\mathsf{p}0}}' \in_{\mathsf{p}} s$.
- Finally, by the rule (PE-TYPE CASE T), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$.

(II) $\dfrac{e_2 \rightsquigarrow e_2'}{e \rightsquigarrow e_1 \in s \,?\, e_2' : e_3}$ where $e' = e_1 \in s \,?\, e_2' : e_3$:
- From (3), there are two cases to consider.
- First, consider the case for the rule (PE-TYPE CASE T). Then, we have the following assumptions:
  (a) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}0}$ (b) $v_{\mathsf{p}0} \in_{\mathsf{p}} s$ (c) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_2' \Downarrow v_{\mathsf{p}}$
  By IH on $e_2 \rightsquigarrow e_2'$ with (c), we have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_2 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$. Then, by the rule (PE-TYPE CASE T), we complete the proof.
- Now consider the case for the rule (PE-TYPE CASE F). Then, we have the following assumptions:
  (a) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}0}$ (b) $v_{\mathsf{p}0} \notin_{\mathsf{p}} s$ (c) $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_3 \Downarrow v_{\mathsf{p}}$
  Then, we just complete the proof by the rule (PE-TYPE CASE F).

(III) $\dfrac{e_3 \rightsquigarrow e_3'}{e \rightsquigarrow e_1 \in s \,?\, e_2 : e_3'}$ where $e' = e_1 \in s \,?\, e_2 : e_3'$:
- Similar to the case for (II).

(IV) $e_1 = v$, $\vdash v : s$, and $e' = e_2$:
- We have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow \mathsf{clos}(v)$.
- By Lemma D.10 with $\vdash v : s$, we have $\mathsf{clos}(v) \in_{\mathsf{p}} s$.

- Then, by the rule (PE-TYPE CASE T) with (3), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$.

(V) $e_1 = v$, $\not\forall\ v : s$, and $e' = e_3$:

- We have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow \mathsf{clos}(v)$.
- By Lemma D.11 with $\not\forall\ v : s$, we have $\mathsf{clos}(v) \not\in_{\mathsf{p}} s$.
- Then, by the rule (PE-TYPE CASE F) with (3), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$.

$(e = e_0\sigma_J)$: $e' = e_0@\sigma_J$.

- By Lemma D.16 with (3), we have $\sigma_J; \varnothing \vdash_{\mathsf{p}} e_0 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- By the rule (PE-INSTANCE), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_0\sigma_J \Downarrow {v_{\mathsf{p}}}'$.

$(e = \mathtt{let}\ \underline{x}\ \mathtt{=}\ e_1\ \mathtt{in}\ e_2)$:

(I) $\dfrac{e_1 \rightsquigarrow e_1'}{e \rightsquigarrow \mathtt{let}\ \underline{x}\ \mathtt{=}\ e_1'\ \mathtt{in}\ e_2}$ where $e' = \mathtt{let}\ \underline{x}\ \mathtt{=}\ e_1'\ \mathtt{in}\ e_2$:

- From (3), we have the following assumptions:
$$(a)\ \ \mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1' \Downarrow v_{\mathsf{p}0} \qquad (b)\ \ \mathbb{i}; \underline{x} \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}$$
- By IH on $e_1 \rightsquigarrow e_1'$ with (a), we have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow {v_{\mathsf{p}0}}'$ and $v_{\mathsf{p}0} \equiv_{\mathsf{c}} {v_{\mathsf{p}0}}'$.
- By Lemma D.15 with (b), we have $\mathbb{i}; \underline{x} \mapsto {v_{\mathsf{p}0}}' \vdash_{\mathsf{p}} e_2 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- Then, by the rule (PE-LET), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow {v_{\mathsf{p}}}'$.

(II) $\dfrac{e_2 \rightsquigarrow e_2'}{e \rightsquigarrow \mathtt{let}\ \underline{x}\ \mathtt{=}\ e_1\ \mathtt{in}\ e_2'}$ where $e' = \mathtt{let}\ \underline{x}\ \mathtt{=}\ e_1\ \mathtt{in}\ e_2'$:

- From (3), we have the following assumptions:
$$(a)\ \ \mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}0} \qquad (b)\ \ \mathbb{i}; \underline{x} \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_2' \Downarrow v_{\mathsf{p}}$$
- By IH on $e_2 \rightsquigarrow e_2'$ with (b), we have $\mathbb{i}; \underline{x} \mapsto v_{\mathsf{p}0} \vdash_{\mathsf{p}} e_2 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- Then, by the rule (PE-LET), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow {v_{\mathsf{p}}}'$.

(III) $e_1 = v$ and $e' = e_2\{{}^{v}/_{\underline{x}}\}$:

- We have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow \mathsf{clos}(v)$.
- By Lemma D.17 with (3), we have $\mathbb{i}; \underline{x} \mapsto \mathsf{clos}(v) \vdash_{\mathsf{p}} e_2 \Downarrow {v_{\mathsf{p}}}'$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$.
- Finally, by the rule (PE-LET), we conclude $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow {v_{\mathsf{p}}}'$.

□

Lemma D.19. *If* $\vdash e : t$ *and* $e \rightsquigarrow^* v$, *then* $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$ *and* $v = (\!|v_{\mathsf{p}}|\!)$.

Proof. By induction on $e \rightsquigarrow^* v$. Suppose $e = v$. Then $v_{\mathsf{p}} = \mathsf{clos}(v)$ and $(\!|v_{\mathsf{p}}|\!) = v$.

Now suppose $e \rightsquigarrow e_1 \rightsquigarrow^* v$. By induction hypothesis on $e_1 \rightsquigarrow^* v$, we have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e_1 \Downarrow {v_{\mathsf{p}}}'$ and $v = (\!|{v_{\mathsf{p}}}'|\!)$. Then, by Lemma D.18, we have $\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$ and $v_{\mathsf{p}} \equiv_{\mathsf{c}} {v_{\mathsf{p}}}'$, and therefore $v = (\!|v_{\mathsf{p}}|\!)$. □

Theorem D.20. *Let* e *be a well-typed closed explicitly-typed expression* ($\vdash e : t$). *Then*
$$\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v \iff e \rightsquigarrow^* (\!|v|\!)$$

Proof. Follows from Lemmas D.12 and D.19. □

D.2 Compilation of the polymorphic language into the intermediate language

In this section, we prove the adequacy of the compilation of the polymorphic language into the intermediate language defined in Section 5.3 (extended with let-polymorphism in Section 5.4). For clarity, in this section, we write respectively e_{o}, v_{o}, and $\mathscr{E}_{\mathsf{o}}$ for the expressions, values, and environments for the intermediate language.

Definition D.21 (Intermediate language)**.**
$$\begin{array}{lcl} e_{\mathsf{o}} & ::= & c \mid x \mid x_\Sigma \mid \lambda^t_\Sigma x.e_{\mathsf{o}} \mid e_{\mathsf{o}}e_{\mathsf{o}} \mid e_{\mathsf{o}} \in t\ \mathtt{?}\ e_{\mathsf{o}}\ \mathtt{:}\ e_{\mathsf{o}} \mid \mathtt{let}\ \underline{x}\ \mathtt{=}\ e_{\mathsf{o}}\ \mathtt{in}\ e_{\mathsf{o}} \\ v_{\mathsf{o}} & ::= & c \mid \langle \lambda^t_\Sigma x.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}} \rangle \\ \Sigma & ::= & \sigma_I \mid \mathtt{comp}(\Sigma, \Sigma') \mid \mathtt{sel}(x, t, \Sigma) \mid \langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle \end{array}$$

In this definition, we added a new symbolic substitution of the form $\langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle$, which is absent from Section 5.3. Furthermore, we use the following new evaluation rule for polymorphic let-variables:
$$\text{(OE-PVAR}_f\text{)}\quad \frac{\mathscr{E}_{\mathsf{o}}(\underline{x}) = \langle \lambda^t_{\Sigma'} y.e_{\mathsf{o}}, {\mathscr{E}_{\mathsf{o}}}' \rangle}{\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} x_\Sigma \Downarrow \langle \lambda^t_{\mathtt{comp}(\langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle, \Sigma')} y.e_{\mathsf{o}}, {\mathscr{E}_{\mathsf{o}}}' \rangle}$$

The difference from the previous rule is that in this rule we use in the decoration $\langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle$ instead of Σ. The main reason is that Σ in x_Σ may contain a symbolic substitution of the form $\mathtt{sel}(y, t, \Sigma_0)$ such that $y \notin \mathsf{dom}({\mathscr{E}_{\mathsf{o}}}')$. Such a symbolic substitution as $\mathtt{sel}(y, t, \Sigma_0)$ may be generated if the let-variable $\underline{x}$ is used inside λ-abstraction in the body of the let-expression. However, in practice, only type-substitutions for the type variables introduced before or in the let-binding may be applied to the polymorphic let-variable, which are already recorded in the closure for the let-variable. Therefore, for the implementation, it is safe to ignore such substitutions as $\langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle$ without evaluating them. Still, introducing this new extra symbolic substitution makes the intermediate language and its evaluation semantics clearer.

Definition D.22 (Compilation)**.**

$$\begin{array}{rcl}
[\![c]\!]_\Sigma & = & c \\
[\![x]\!]_\Sigma & = & x \\
[\![\underline{x}]\!]_\Sigma & = & x_\Sigma \\
[\![\lambda^t_{\sigma_I} x.e]\!]_\Sigma & = & \lambda^t_{\mathtt{comp}(\Sigma,\sigma_I)} x.[\![e]\!]_{\mathtt{sel}(x,t,\mathtt{comp}(\Sigma,\sigma_I))} \\
[\![e_1 e_2]\!]_\Sigma & = & [\![e_1]\!]_\Sigma [\![e_2]\!]_\Sigma \\
[\![e\sigma_I]\!]_\Sigma & = & [\![e]\!]_{\mathtt{comp}(\Sigma,\sigma_I)} \\
[\![e_1 \in t\ \mathtt{?}\ e_2\ \mathtt{:}\ e_3]\!]_\Sigma & = & [\![e_1]\!]_\Sigma \in t\ \mathtt{?}\ [\![e_2]\!]_\Sigma\ \mathtt{:}\ [\![e_3]\!]_\Sigma \\
[\![\mathtt{let}\ \underline{x} = e_1\ \mathtt{in}\ e_2]\!]_\Sigma & = & \mathtt{let}\ \underline{x} = [\![e_1]\!]_\Sigma\ \mathtt{in}\ [\![e_2]\!]_\Sigma
\end{array}$$

Definition D.23 (Membership)**.**

$$\begin{array}{rcl}
c \in_{\mathsf{o}} t & \overset{\text{def}}{\iff} & b_c \le t \\
\langle \lambda^s_\Sigma x.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}} \rangle \in_{\mathsf{o}} t & \overset{\text{def}}{\iff} & s(\mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma)) \le t
\end{array}$$

where the evaluation of the symbolic set of type-substitutions is inductively defined as

$$\begin{array}{rcl}
\mathit{eval}(\mathscr{E}_{\mathsf{o}}, \sigma_I) & = & \sigma_I \\
\mathit{eval}(\mathscr{E}_{\mathsf{o}}, \mathtt{comp}(\Sigma, \Sigma')) & = & \mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma) \circ \mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma') \\
\mathit{eval}(\mathscr{E}_{\mathsf{o}}, \mathtt{sel}(x, \bigwedge_{i\in I} t_i \to s_i, \Sigma)) & = & [\ \sigma_j \in \mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma) \mid \exists i \in I : \mathscr{E}_{\mathsf{o}}(x) \in_{\mathsf{o}} t_i\sigma_j\ ] \\
\mathit{eval}(\mathscr{E}_{\mathsf{o}}, \langle \mathscr{E}_{\mathsf{o}}', \Sigma \rangle) & = & \mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma)
\end{array}$$

Definition D.24. *Let $\mathscr{E}$ and v_{p} be an environment and a value for the polymorphic language and $\mathscr{E}_{\mathsf{o}}$ and v_{o} for the intermediate language. We define an equivalence relation $\equiv_{\mathsf{o}}$ between them as follows:*

$$\begin{array}{rcl}
\mathscr{E} \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}} & \overset{\text{def}}{\iff} & \mathsf{dom}(\mathscr{E}) = \mathsf{dom}(\mathscr{E}_{\mathsf{o}}) \textit{ and } \forall x \in \mathsf{dom}(\mathscr{E}), \mathscr{E}(x) \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}(x) \\
 & & \textit{and } \forall \underline{x} \in \mathsf{dom}(\mathscr{E}), \mathscr{E}(\underline{x}) \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}(\underline{x}) \\
\langle \lambda^t_{\sigma_K} x.e, \mathscr{E}, \sigma_H \rangle \equiv_{\mathsf{o}} \langle \lambda^t_\Sigma x.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}} \rangle & \overset{\text{def}}{\iff} & \sigma_H \circ \sigma_K = \mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma) \textit{ and } e_{\mathsf{o}} = [\![e]\!]_{\mathtt{sel}(x,t,\Sigma)} \textit{ and } \mathscr{E} \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}
\end{array}$$

Moreover, for any constant c, by definition $c \equiv_{\mathsf{o}} c$.

Note that if $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$, then $(\!|v_{\mathsf{p}}|\!) = (\!|v_{\mathsf{o}}|\!)$, but not vice versa.

Lemma D.25. *Let $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$. Then:*
(1) $v_{\mathsf{p}} \in_{\mathsf{p}} t$ if and only if $v_{\mathsf{o}} \in_{\mathsf{o}} t$.
(2) $v_{\mathsf{p}} \notin_{\mathsf{p}} t$ if and only if $v_{\mathsf{o}} \notin_{\mathsf{o}} t$.

Proof. If $v_{\mathsf{p}} = v_{\mathsf{o}} = c$, the proof is trivial. Suppose $v_{\mathsf{p}} = \langle \lambda^s_{\sigma_K} x.e, \mathscr{E}, \sigma_H \rangle$ and $v_{\mathsf{o}} = \langle \lambda^s_\Sigma x.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}} \rangle$. By definition, $v_{\mathsf{p}} \in_{\mathsf{p}} t$ if and only if $s(\sigma_H \circ \sigma_K) \le t$ and $v_{\mathsf{o}} \in_{\mathsf{o}} t$ if and only if $s(\mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma)) \le t$. From $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$, we have $\sigma_H \circ \sigma_K = \mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma)$, thus completing the proof. □

Lemma D.26. *Suppose $\mathscr{E} \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}$ and $\sigma_I = \mathit{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma)$ and $\vdash (e@\sigma_I)(\!|\mathscr{E}|\!) : t$. Then, $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$ if and only if $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e]\!]_\Sigma \Downarrow v_{\mathsf{o}}$ where $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$.*

Proof. (Only-if part) By induction on a derivation of $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$.

(PE-CONST)**:** $e = [\![e]\!]_\Sigma = v_{\mathsf{p}} = v_{\mathsf{o}} = c$.
(PE-VAR)**:** $e = [\![e]\!]_\Sigma = x$ and $v_{\mathsf{p}} = \mathscr{E}(x)$ and $v_{\mathsf{o}} = \mathscr{E}_{\mathsf{o}}(x)$. From $\mathscr{E} \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}$, we have $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$.
(PE-CLOSURE)**:** $e = \lambda^t_{\sigma_K} x.e'$ and $[\![e]\!]_\Sigma = \lambda^t_{\mathtt{comp}(\Sigma,\sigma_K)} x.[\![e']\!]_{\mathtt{sel}(x,t,\mathtt{comp}(\Sigma,\sigma_K))}$.

- $v_{\mathsf{p}} = \langle \lambda^t_{\sigma_K} x.e', \mathscr{E}, \sigma_I \rangle$.
- $v_{\mathsf{o}} = \langle \lambda^t_{\mathtt{comp}(\Sigma,\sigma_K)} x.[\![e']\!]_{\mathtt{sel}(x,t,\mathtt{comp}(\Sigma,\sigma_K))}, \mathscr{E}_{\mathsf{o}} \rangle$.
- $\mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \mathtt{comp}(\Sigma, \sigma_K)) = \mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma) \circ \mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \sigma_K) = \sigma_I \circ \sigma_K$ and therefore $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$ by Definition D.24.

(PE-APPLY)**:** $e = e_1 e_2$ and $[\![e]\!]_\Sigma = [\![e_1]\!]_\Sigma [\![e_2]\!]_\Sigma$.

- From $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$, we have the following assumptions:
  (*a*) $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \Downarrow \langle \lambda^s_{\sigma_K} x.e', \mathscr{E}', \sigma_H \rangle$ where $s = \wedge_{i\in I} s_i \to s_i'$;
  (*b*) $\sigma_I; \mathscr{E}_1 \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}'$;
  (*c*) $\sigma_J = \sigma_H \circ \sigma_K$ and $P = \{j \in J \mid \exists i \in I : v_{\mathsf{p}}' \in_{\mathsf{p}} s_i\sigma_j\}$; and
  (*d*) $\sigma_P; \mathscr{E}', x \mapsto v_{\mathsf{p}}' \vdash_{\mathsf{p}} e' \Downarrow v_{\mathsf{p}}$.
- By IH on (a), $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e_1]\!]_\Sigma \Downarrow \langle \lambda^s_{\Sigma'} x.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}}' \rangle$ where $\mathsf{eval}(\mathscr{E}_{\mathsf{o}}', \Sigma') = \sigma_H \circ \sigma_K = \sigma_J$ and $\mathscr{E}' \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}'$ and $e_{\mathsf{o}} = [\![e']\!]_{\mathtt{sel}(x,s,\Sigma')}$.
- By IH on (b), $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e_2]\!]_\Sigma \Downarrow v_{\mathsf{o}}'$ where $v_{\mathsf{p}}' \equiv_{\mathsf{o}} v_{\mathsf{o}}'$.
- From $\mathscr{E}' \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}'$ and $v_{\mathsf{p}}' \equiv_{\mathsf{o}} v_{\mathsf{o}}'$, we have $(\mathscr{E}', x \mapsto v_{\mathsf{p}}') \equiv_{\mathsf{o}} (\mathscr{E}_{\mathsf{o}}', x \mapsto v_{\mathsf{o}}')$.
- $\mathsf{eval}((\mathscr{E}_{\mathsf{o}}', x \mapsto v_{\mathsf{o}}'), \mathtt{sel}(x, s, \Sigma')) = [\ \sigma_j \in \mathsf{eval}(\mathscr{E}_{\mathsf{o}}', \Sigma') \mid \exists i \in I : v_{\mathsf{o}}' \in_{\mathsf{o}} s_i\sigma_j\ ] = [\ \sigma_j \in \sigma_J \mid \exists i \in I : v_{\mathsf{o}}' \in_{\mathsf{o}} s_i\sigma_j\ ]$, which is equal to σ_P by Lemma D.25.
- By IH on (d), $\mathscr{E}_{\mathsf{o}}', x \mapsto v_{\mathsf{o}}' \vdash_{\mathsf{o}} e_{\mathsf{o}} \Downarrow v_{\mathsf{o}}$ and $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$.
- Finally, by the rule (OE-APPLY), we conclude $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e]\!]_\Sigma \Downarrow v_{\mathsf{o}}$.

(PE-TYPE CASE T)**:** $e = e_1 \in t\ \mathtt{?}\ e_2\ \mathtt{:}\ e_3$ and $[\![e]\!]_\Sigma = [\![e_1]\!]_\Sigma \in t\ \mathtt{?}\ [\![e_2]\!]_\Sigma\ \mathtt{:}\ [\![e_3]\!]_\Sigma$.

- From $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$, we have the following assumptions:
$$(a)\ \ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}}' \qquad (b)\ \ v_{\mathsf{p}}' \in_{\mathsf{p}} t \qquad (c)\ \ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}$$
- By IH on (a), $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e_1]\!]_\Sigma \Downarrow v_{\mathsf{o}}'$ and $v_{\mathsf{p}}' \equiv_{\mathsf{o}} v_{\mathsf{o}}'$.
- By Lemma D.25 with (b), we have $v_{\mathsf{o}}' \in_{\mathsf{o}} t$.
- By IH on (c), $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e_2]\!]_\Sigma \Downarrow v_{\mathsf{o}}$ and $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$.
- Finally, by the rule (OE-TYPE CASE T), we conclude $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e]\!]_\Sigma \Downarrow v_{\mathsf{o}}$.

(PE-TYPE CASE F): Similar to the case for (PE-TYPE CASE T).

(PE-INSTANCE): $e = e_0\sigma_J$ and $[\![e]\!]_\Sigma = [\![e_0]\!]_{\mathsf{comp}(\Sigma,\sigma_J)}$.
- From $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$, we have the following assumption:
$$(a)\ \ \sigma_I \circ \sigma_J; \mathscr{E} \vdash_{\mathsf{p}} e_0 \Downarrow v_{\mathsf{p}}$$
- $\mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \mathsf{comp}(\Sigma, \sigma_J)) = \mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma) \circ \mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \sigma_J) = \sigma_I \circ \sigma_J$.
- By IH on (a), $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e_0]\!]_{\mathsf{comp}(\Sigma,\sigma_J)} \Downarrow v_{\mathsf{o}}$ and $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$.
- Then, from $[\![e]\!]_\Sigma = [\![e_0]\!]_{\mathsf{comp}(\Sigma,\sigma_J)}$, we complete the proof.

(PE-VAR$_c$): Similar to the case for (PE-CONST).

(PE-VAR$_f$): $e = \underline{x}$ and $[\![e]\!]_\Sigma = x_\Sigma$ and $\mathscr{E}(\underline{x}) = \langle \lambda^t_{\sigma_K} y.e_0, \mathscr{E}', \sigma_H \rangle$.
- From $\mathscr{E} \equiv_{\mathsf{o}} \mathscr{E}_{\mathsf{o}}$, we have $\mathscr{E}_{\mathsf{o}}(\underline{x}) = \langle \lambda^t_{\Sigma'} y.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}}' \rangle$ where $\mathsf{eval}(\mathscr{E}_{\mathsf{o}}', \Sigma') = \sigma_H \circ \sigma_K$ and $\mathscr{E}' = \mathscr{E}_{\mathsf{o}}'$ and $e_{\mathsf{o}} = [\![e_0]\!]_{\mathtt{sel}(y,t,\Sigma')}$.
- Then, $v_{\mathsf{p}} = \langle \lambda^t_{\sigma_K} y.e_0, \mathscr{E}', \sigma_I \circ \sigma_H \rangle$ and $v_{\mathsf{o}} = \langle \lambda^t_{\mathsf{comp}(\langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle, \Sigma')} y.e_{\mathsf{o}}, \mathscr{E}_{\mathsf{o}}' \rangle$.
- $\mathsf{eval}(\mathscr{E}_{\mathsf{o}}', \mathsf{comp}(\langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle, \Sigma')) = \mathsf{eval}(\mathscr{E}_{\mathsf{o}}', \langle \mathscr{E}_{\mathsf{o}}, \Sigma \rangle) \circ \mathsf{eval}(\mathscr{E}_{\mathsf{o}}', \Sigma') = \mathsf{eval}(\mathscr{E}_{\mathsf{o}}, \Sigma) \circ (\sigma_H \circ \sigma_K) = \sigma_I \circ (\sigma_H \circ \sigma_K)$ and therefore $v_{\mathsf{p}} \equiv v_{\mathsf{o}}$.

(PE-LET): $e = \mathtt{let}\ \underline{x} = e_1\ \mathtt{in}\ e_2$ and $[\![e]\!]_\Sigma = \mathtt{let}\ \underline{x} = [\![e_1]\!]_\Sigma\ \mathtt{in}\ [\![e_2]\!]_\Sigma$.
- From $\sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e \Downarrow v_{\mathsf{p}}$, we have the following assumptions:
$$(a)\ \ \sigma_I; \mathscr{E} \vdash_{\mathsf{p}} e_1 \Downarrow v_{\mathsf{p}}' \qquad (b)\ \ \sigma_I; \mathscr{E}, \underline{x} \mapsto v_{\mathsf{p}}' \vdash_{\mathsf{p}} e_2 \Downarrow v_{\mathsf{p}}$$
- By IH on (a), $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e_1]\!]_\Sigma \Downarrow v_{\mathsf{o}}'$ and $v_{\mathsf{p}}' \equiv_{\mathsf{o}} v_{\mathsf{o}}'$.
- By IH on (b), $\mathscr{E}_{\mathsf{o}}, \underline{x} \mapsto v_{\mathsf{o}}' \vdash_{\mathsf{o}} [\![e_1]\!]_\Sigma \Downarrow v_{\mathsf{o}}$ and $v_{\mathsf{p}} \equiv_{\mathsf{o}} v_{\mathsf{o}}$.
- Finally, by the rule (OE-LET), we conclude $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e]\!]_\Sigma \Downarrow v_{\mathsf{o}}$.

(If part) By induction on a derivation of $\mathscr{E}_{\mathsf{o}} \vdash_{\mathsf{o}} [\![e]\!]_\Sigma \Downarrow v_{\mathsf{o}}$. The proof is similar to the proof for the (only-if part). □

Theorem D.27. *Let e be a well-typed closed explicitly-typed expression ($\vdash_{\mathscr{A}} e : t$). Then*
$$\mathbb{i}; \varnothing \vdash_{\mathsf{p}} e \Downarrow v \iff \vdash_{\mathsf{o}} [\![e]\!]_{\mathbb{i}} \Downarrow v'$$
with $(\!|v|\!) = (\!|v'|\!)$.

Proof. Follows from Lemma D.26. □

Corollary D.28 (Adequacy of the compilation)**.** *Let e be a well-typed closed explicitly-typed expression ($\vdash_{\mathscr{A}} e : t$). Then*
$$\vdash_{\mathsf{o}} [\![e]\!]_{\mathbb{i}} \Downarrow v_{\mathsf{o}} \iff e \rightsquigarrow^* (\!|v|\!)$$

Proof. Follows from Theorems D.20 and D.27. □

E. Syntactic sugar for type-case

In order to make type-case closer to pattern matching, it is handy to define an extension of the type-case expression that features binding, which we write $(x{=}e){\in}t\,?\,e_1 : e_2$, and that can be encoded as:
$$(\lambda^{((s\wedge t)\to t_1)\wedge((s\wedge\neg t)\to t_2)} x.x{\in}t\,?\,e_1 : e_2)e$$
where s is the type of e, t_1 the type of e_1, and t_2 the type of e_2. We add another twist to this construct and define a particular (purely) syntactic sugar: $x\ \boldsymbol{\epsilon}\ t\,?\,e_1 : e_2$ (notice the boldface "belongs to" symbol) which stands for $(x{=}x){\in}t\,?\,e_1 : e_2$. The reader may wonder what is the interest of binding a variable to itself. Actually, the two occurrences of x in $(x{=}x){\in}t$ denote two distinct variables: the one on the right is recorded in the environment with some type s; this variable does not occur either in e_1 or e_2 because it is hidden by the x on the left; this binds the occurrences of x in e_1 and e_2 but with different types, $s{\wedge}t$ in e_1 and $s{\wedge}\neg t$ in e_2. This allows the system to use different type assumptions for x in each branch, as stated by the (algorithmic) typing rule directly derived from the encoding:
$$\frac{\begin{array}{l}(\textit{case-var})\\ t_i \not\simeq \mathbb{0} \qquad \Rightarrow \qquad \Delta \,;\, \Gamma, (x : t_i) \vdash e_i : s_i\end{array}}{\Delta \,;\, \Gamma \vdash x\ \boldsymbol{\epsilon}\ t\,?\,e_1 : e_2 \ :\ \bigvee_{t_i \not\simeq \mathbb{0}} s_i} \quad \begin{array}{l} t_1 = \Gamma(x) \wedge t \\ t_2 = \Gamma(x) \wedge \neg t \end{array}$$

Note that x is defined in Γ but its type $\Gamma(x)$ is overridden in the premises to type the branches. We already silently used this construction in the body of the definition of `map` in Section 2 which should have been written with the boldface "belongs to" symbol:

$$\lambda^{[\alpha]\to[\beta]}\ell\,.\,\ell\,\boldsymbol{\epsilon}\,\texttt{nil}\,\texttt{?}\,\texttt{nil}\,\texttt{:}\,(f(\pi_1\ell),mf(\pi_2\ell)),$$

since the presence of the π_i's in the second branch needs the hypothesis $\ell : [\alpha]\backslash\texttt{nil}$ (the expression $\pi_i\ell$ is well-typed only if ℓ is a product, that is, in this case it a non-empty list). If we do not use the boldface belong symbol, then each branch (in particular, the second one) is typed under the hypothesis $\ell : [\alpha]$ and, thus, type-check fails. In practice, any real programming language will implement just $\boldsymbol{\epsilon}$ and not $\in$ whenever the tested expression is a variable.