Reasonably Programmable Syntax (ICFP 2017 Supplemental Material)

February 27, 2017

Contents

Appendix A

Conventions

A.1 Typographic Conventions

We adopt *PFPL*'s typographic conventions for operational forms throughout the paper [1]. In particular, the names of operators and indexed families of operators are written in `typewriter font`, indexed families of operators specify indices within [braces], and term arguments are grouped arbitrarily (roughly, by sort) using {curly braces} and (rounded braces). We write $p.e$ for expressions binding the variables that appear in the pattern p.

We write $\{i \hookrightarrow \tau_i\}_{i \in L}$ for a sequence of arguments τ_i, one for each $i \in L$, and similarly for arguments of other valences. Operations that are parameterized by label sets, e.g. $\texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$, are identified up to mutual reordering of the label set and the corresponding argument sequence. Similarly, we write $\{i \hookrightarrow J_i\}_{i \in L}$ for the finite set of derivations J_i for each $i \in L$.

We write $\{r_i\}_{1 \leq i \leq n}$ for sequences of $n \geq 0$ rule arguments, and similarly for other finite sequences.

Empty finite sets and finite functions are written $\emptyset$, or omitted entirely within judgements, and non-empty finite sets and finite functions are written as comma-separated sequences identified up to exchange and contraction.

Appendix B

miniVerse$_{\mathbf{S}}$ (and miniVerse$_{\mathbf{SE}}$)

This section defines miniVerse$_{\mathbf{S}}$, the calculus of simple expression and pattern TSMs. For some readers, it might be useful to snip out pattern matching to get a language strictly of expression TSMs. To support that, one can omit the segments typeset in gray backgrounds below to recover miniVerse$_{\mathbf{SE}}$, a calculus of simple expression TSMs. We have included the necessary eliminators below (they are technically redundant with pattern matching, but don't hurt things so they're left in white.)

B.1 Expanded Language (XL)

B.1.1 Syntax

Sort			Operational Form	Description
Typ	τ	::=	t	variable
			$\texttt{parr}(\tau;\tau)$	partial function
			$\texttt{all}(t.\tau)$	polymorphic
			$\texttt{rec}(t.\tau)$	recursive
			$\texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i\in L})$	labeled product
			$\texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i\in L})$	labeled sum
Exp	e	::=	x	variable
			$\texttt{lam}\{\tau\}(x.e)$	abstraction
			$\texttt{ap}(e;e)$	application
			$\texttt{tlam}(t.e)$	type abstraction
			$\texttt{tap}\{\tau\}(e)$	type application
			$\texttt{fold}(e)$	fold
			$\texttt{unfold}(e)$	unfold
			$\texttt{tpl}[L](\{i \hookrightarrow e_i\}_{i\in L})$	labeled tuple
			$\texttt{prj}[\ell](e)$	projection
			$\texttt{inj}[\ell](e)$	injection
			$\texttt{case}[L](e;\{i \hookrightarrow x_i.e_i\}_{i\in L})$	case analysis
			$\texttt{match}[n](e;\{r_i\}_{1\le i\le n})$	match
Rule	r	::=	$\texttt{rule}(p.e)$	rule
Pat	p	::=	x	variable pattern
			$\texttt{wildp}$	wildcard pattern
			$\texttt{foldp}(p)$	fold pattern
			$\texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i\in L})$	labeled tuple pattern
			$\texttt{injp}[\ell](p)$	injection pattern

B.1.2 Statics

Type formation contexts, Δ, are finite sets of hypotheses of the form t type. We write Δ, t type when t type $\notin \Delta$ for Δ extended with the hypothesis t type.

Typing contexts, Γ, are finite functions that map each variable $x \in \mathrm{dom}(\Gamma)$, where $\mathrm{dom}(\Gamma)$ is a finite set of variables, to the hypothesis $x : \tau$, for some τ. We write $\Gamma, x : \tau$, when $x \notin \mathrm{dom}(\Gamma)$, for the extension of Γ with a mapping from x to $x : \tau$, and $\Gamma \cup \Gamma'$ when $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \emptyset$ for the typing context mapping each $x \in \mathrm{dom}(\Gamma) \cup \mathrm{dom}(\Gamma')$ to $x : \tau$ if $x : \tau \in \Gamma$ or $x : \tau \in \Gamma'$. We write $\Delta \vdash \Gamma$ ctx if every type in Γ is well-formed relative to Δ.

Definition B.1 (Typing Context Formation)**.** $\Delta \vdash \Gamma$ ctx *iff for each hypothesis* $x : \tau \in \Gamma$*, we have* $\Delta \vdash \tau$ type.

$\boxed{\Delta \vdash \tau \text{ type}}$ τ is a well-formed type

$$\frac{}{\Delta, t \text{ type} \vdash t \text{ type}} \tag{B.1a}$$

$$\frac{\Delta \vdash \tau_1 \text{ type} \qquad \Delta \vdash \tau_2 \text{ type}}{\Delta \vdash \mathtt{parr}(\tau_1; \tau_2) \text{ type}} \tag{B.1b}$$

$$\frac{\Delta, t \text{ type} \vdash \tau \text{ type}}{\Delta \vdash \mathtt{all}(t.\tau) \text{ type}} \tag{B.1c}$$

$$\frac{\Delta, t \text{ type} \vdash \tau \text{ type}}{\Delta \vdash \mathtt{rec}(t.\tau) \text{ type}} \tag{B.1d}$$

$$\frac{\{\Delta \vdash \tau_i \text{ type}\}_{i \in L}}{\Delta \vdash \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \text{ type}} \tag{B.1e}$$

$$\frac{\{\Delta \vdash \tau_i \text{ type}\}_{i \in L}}{\Delta \vdash \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \text{ type}} \tag{B.1f}$$

$\boxed{\Delta\ \Gamma \vdash e : \tau}$ e is assigned type τ

$$\frac{}{\Delta\ \Gamma, x : \tau \vdash x : \tau} \tag{B.2a}$$

$$\frac{\Delta \vdash \tau \text{ type} \qquad \Delta\ \Gamma, x : \tau \vdash e : \tau'}{\Delta\ \Gamma \vdash \mathtt{lam}\{\tau\}(x.e) : \mathtt{parr}(\tau; \tau')} \tag{B.2b}$$

$$\frac{\Delta\ \Gamma \vdash e_1 : \mathtt{parr}(\tau; \tau') \qquad \Delta\ \Gamma \vdash e_2 : \tau}{\Delta\ \Gamma \vdash \mathtt{ap}(e_1; e_2) : \tau'} \tag{B.2c}$$

$$\frac{\Delta, t \text{ type}\ \Gamma \vdash e : \tau}{\Delta\ \Gamma \vdash \mathtt{tlam}(t.e) : \mathtt{all}(t.\tau)} \tag{B.2d}$$

$$\frac{\Delta\ \Gamma \vdash e : \mathtt{all}(t.\tau) \qquad \Delta \vdash \tau' \text{ type}}{\Delta\ \Gamma \vdash \mathtt{tap}\{\tau'\}(e) : [\tau'/t]\tau} \tag{B.2e}$$

$$\frac{\Delta\ \Gamma \vdash e : [\mathtt{rec}(t.\tau)/t]\tau}{\Delta\ \Gamma \vdash \mathtt{fold}(e) : \mathtt{rec}(t.\tau)} \tag{B.2f}$$

$$\frac{\Delta\ \Gamma \vdash e : \mathtt{rec}(t.\tau)}{\Delta\ \Gamma \vdash \mathtt{unfold}(e) : [\mathtt{rec}(t.\tau)/t]\tau} \tag{B.2g}$$

$$\frac{\{\Delta\ \Gamma \vdash e_i : \tau_i\}_{i \in L}}{\Delta\ \Gamma \vdash \mathtt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L}) : \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})} \tag{B.2h}$$

$$\frac{\Delta\ \Gamma \vdash e : \mathtt{prod}[L,\ell](\{i \hookrightarrow \tau_i\}_{i\in L}; \ell \hookrightarrow \tau)}{\Delta\ \Gamma \vdash \mathtt{prj}[\ell](e) : \tau} \tag{B.2i}$$

$$\frac{\Delta\ \Gamma \vdash e : \tau}{\Delta\ \Gamma \vdash \mathtt{inj}[\ell](e) : \mathtt{sum}[L,\ell](\{i \hookrightarrow \tau_i\}_{i\in L}; \ell \hookrightarrow \tau)} \tag{B.2j}$$

$$\frac{\Delta\ \Gamma \vdash e : \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i\in L}) \qquad \{\Delta\ \Gamma, x_i : \tau_i \vdash e_i : \tau\}_{i\in L}}{\Delta\ \Gamma \vdash \mathtt{case}[L](e; \{i \hookrightarrow x_i.e_i\}_{i\in L}) : \tau} \tag{B.2k}$$

$$\frac{\Delta\ \Gamma \vdash e : \tau \qquad \{\Delta\ \Gamma \vdash r_i : \tau \Rightarrow \tau'\}_{1\le i\le n}}{\Delta\ \Gamma \vdash \mathtt{match}[n](e; \{r_i\}_{1\le i\le n}) : \tau'} \tag{B.2l}$$

$\boxed{\Delta\ \Gamma \vdash r : \tau \Rightarrow \tau'}$ r takes values of type τ to values of type τ'

$$\frac{\Delta \vdash p : \tau \dashv\Vdash \Gamma' \qquad \Delta\ \Gamma \cup \Gamma' \vdash e : \tau'}{\Delta\ \Gamma \vdash \mathtt{rule}(p.e) : \tau \Rightarrow \tau'} \tag{B.3}$$

Rule (B.3) is defined mutually inductively with Rules (B.2).

$\boxed{\Delta \vdash p : \tau \dashv\Vdash \Gamma}$ p matches values of type τ and generates hypotheses Γ

$$\frac{}{\Delta \vdash x : \tau \dashv\Vdash x : \tau} \tag{B.4a}$$

$$\frac{}{\Delta \vdash \mathtt{wildp} : \tau \dashv\Vdash \emptyset} \tag{B.4b}$$

$$\frac{\Delta \vdash p : [\mathtt{rec}(t.\tau)/t]\tau \dashv\Vdash \Gamma}{\Delta \vdash \mathtt{foldp}(p) : \mathtt{rec}(t.\tau) \dashv\Vdash \Gamma} \tag{B.4c}$$

$$\frac{\{\Delta \vdash p_i : \tau_i \dashv\Vdash \Gamma_i\}_{i\in L}}{\Delta \vdash \mathtt{tplp}[L](\{i \hookrightarrow p_i\}_{i\in L}) : \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i\in L}) \dashv\Vdash \cup_{i\in L}\Gamma_i} \tag{B.4d}$$

$$\frac{\Delta \vdash p : \tau \dashv\Vdash \Gamma}{\Delta \vdash \mathtt{injp}[\ell](p) : \mathtt{sum}[L,\ell](\{i \hookrightarrow \tau_i\}_{i\in L}; \ell \hookrightarrow \tau) \dashv\Vdash \Gamma} \tag{B.4e}$$

Metatheory

The rules above are syntax-directed, so we assume an inversion lemma for each rule as needed without stating it separately or proving it explicitly. The following standard lemmas also hold.

The Weakening Lemma establishes that extending the context with unnecessary hypotheses preserves well-formedness and typing.

Lemma B.2 (Weakening)**.**

1. *If* $\Delta \vdash \tau$ type *then* Δ, t type $\vdash \tau$ type*.*
2. (a) *If* $\Delta\ \Gamma \vdash e : \tau$ *then* Δ, t type $\Gamma \vdash e : \tau$*.*
   (b) *If* $\Delta\ \Gamma \vdash r : \tau \Mapsto \tau'$ *then* Δ, t type $\Gamma \vdash r : \tau \Mapsto \tau'$*.*
3. (a) *If* $\Delta\ \Gamma \vdash e : \tau$ *and* $\Delta \vdash \tau''$ type *then* $\Delta\ \Gamma, x : \tau'' \vdash e : \tau$*.*
   (b) *If* $\Delta\ \Gamma \vdash r : \tau \Mapsto \tau'$ *and* $\Delta \vdash \tau''$ type *then* $\Delta\ \Gamma, x : \tau'' \vdash r : \tau \Mapsto \tau'$*.*
4. *If* $\Delta \vdash p : \tau \dashv\mid \Gamma$ *then* $p : \tau \dashv\mid \Gamma$*.*

Proof Sketch.

1. By rule induction over Rules (B.1).
2. By mutual rule induction over Rules (B.2) and Rule (B.3), and part 1.
3. By mutual rule induction over Rules (B.2) and Rule (B.3), and part 1.
4. By rule induction over Rules (B.4).

□

The pattern typing judgement is *linear* in the pattern typing context, i.e. it does *not* obey weakening of the pattern typing context. This is to ensure that the pattern typing context captures exactly those hypotheses generated by a pattern, and no others.

The Substitution Lemma establishes that substitution of a well-formed type for a type variable, or an expanded expression of the appropriate type for an expanded expression variable, preserves well-formedness and typing.

Lemma B.3 (Substitution)**.**

1. *If* Δ, t type $\vdash \tau$ type *and* $\Delta \vdash \tau'$ type *then* $\Delta \vdash [\tau'/t]\tau$ type*.*
2. (a) *If* Δ, t type $\Gamma \vdash e : \tau$ *and* $\Delta \vdash \tau'$ type *then* $\Delta\ [\tau'/t]\Gamma \vdash [\tau'/t]e : [\tau'/t]\tau$*.*
   (b) *If* Δ, t type $\Gamma \vdash r : \tau \Mapsto \tau''$ *and* $\Delta \vdash \tau'$ type *then* $\Delta\ [\tau'/t]\Gamma \vdash [\tau'/t]r : [\tau'/t]\tau \Mapsto [\tau'/t]\tau''$*.*
3. (a) *If* $\Delta\ \Gamma, x : \tau' \vdash e : \tau$ *and* $\Delta\ \Gamma \vdash e' : \tau'$ *then* $\Delta\ \Gamma \vdash [e'/x]e : \tau$*.*
   (b) *If* $\Delta\ \Gamma, x : \tau' \vdash r : \tau \Mapsto \tau''$ *and* $\Delta\ \Gamma \vdash e' : \tau''$ *then* $\Delta\ \Gamma \vdash [e'/x]r : \tau \Mapsto \tau''$*.*

Proof Sketch.

1. By rule induction over Rules (B.1).
2. By mutual rule induction over Rules (B.2) and Rule (B.3).

3. By mutual rule induction over Rules (B.2) and Rule (B.3).

□

The Decomposition Lemma is the converse of the Substitution Lemma.

Lemma B.4 (Decomposition)**.**

1. *If* $\Delta \vdash [\tau'/t]\tau$ type *and* $\Delta \vdash \tau'$ type *then* Δ, t type $\vdash \tau$ type*.*
2. (a) *If* $\Delta\ [\tau'/t]\Gamma \vdash [\tau'/t]e : [\tau'/t]\tau$ *and* $\Delta \vdash \tau'$ type *then* Δ, t type $\Gamma \vdash e : \tau$*.*
   (b) *If* $\Delta\ [\tau'/t]\Gamma \vdash [\tau'/t]r : [\tau'/t]\tau \Rightarrow [\tau'/t]\tau''$ *and* $\Delta \vdash \tau'$ type *then* Δ, t type $\Gamma \vdash r : \tau \Rightarrow \tau''$*.*
3. (a) *If* $\Delta\ \Gamma \vdash [e'/x]e : \tau$ *and* $\Delta\ \Gamma \vdash e' : \tau'$ *then* $\Delta\ \Gamma, x : \tau' \vdash e : \tau$*.*
   (b) *If* $\Delta\ \Gamma \vdash [e'/x]r : \tau \Rightarrow \tau''$ *and* $\Delta\ \Gamma \vdash e' : \tau'$ *then* $\Delta\ \Gamma, x : \tau' \vdash r : \tau \Rightarrow \tau''$*.*

Proof Sketch.

1. By rule induction over Rules (B.1) and case analysis over the definition of substitution. In all cases, the derivation of $\Delta \vdash [\tau'/t]\tau$ type does not depend on the form of τ'.
2. By mutual rule induction over Rules (B.2) and Rule (B.3) and case analysis over the definition of substitution. In all cases, the derivation of $\Delta\ [\tau'/t]\Gamma \vdash [\tau'/t]e : [\tau'/t]\tau$ or $\Delta\ [\tau'/t]\Gamma \vdash [\tau'/t]r : [\tau'/t]\tau \Rightarrow [\tau'/t]\tau''$ does not depend on the form of τ'.
3. By mutual rule induction over Rules (B.2) and Rule (B.3) and case analysis over the definition of substitution. In all cases, the derivation of $\Delta\ \Gamma \vdash [e'/x]e : \tau$ or $\Delta\ \Gamma \vdash [e'/x]r : \tau \Rightarrow \tau''$ does not depend on the form of e'.

□

Lemma B.5 (Pattern Regularity)**.** *If* $\Delta \vdash p : \tau \dashv \Gamma$ *and* $\Delta \vdash \tau$ type *then* $\Delta \vdash \Gamma$ ctx *and* $\mathsf{fv}(p) = dom(\Gamma)$*.*

Proof. By rule induction over Rules (B.4).

Case (B.4a)**.**

(1)	$p = x$	by assumption
(2)	$\Gamma = x : \tau$	by assumption
(3)	$\Delta \vdash \tau$ type	by assumption
(4)	$\Delta \vdash x : \tau$ ctx	by Definition B.1 on (3)
(5)	$\mathsf{fv}(p) = \mathrm{dom}(\Gamma) = \{x\}$	by definition

Case (B.4b)**.**

(1) $p = \texttt{wildp}$ by assumption

(2) $\Gamma = \emptyset$ by assumption

(3) $\Delta \vdash \emptyset\ \mathsf{ctx}$ by Definition B.1

(4) $\mathsf{fv}(p) = \mathrm{dom}(\Gamma) = \emptyset$ by definition

Case (B.4d)**.**

(1) $p = \texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L})$ by assumption

(2) $\tau = \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ by assumption

(3) $\Gamma = \cup_{i \in L} \Gamma_i$ by assumption

(4) $\{\Delta \vdash p_i : \tau_i \dashv\!\mid \Gamma_i\}_{i \in L}$ by assumption

(5) $\Delta \vdash \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})\ \mathsf{type}$ by assumption

(6) $\{\Delta \vdash \tau_i\ \mathsf{type}\}_{i \in L}$ by Inversion of Rule (B.1e) on (5)

(7) $\{\Delta \vdash \Gamma_i\ \mathsf{ctx}\}_{i \in L}$ by IH over (4) and (6)

(8) $\{\mathsf{fv}(p_i) = \mathrm{dom}(\Gamma_i)\}_{i \in L}$ by IH over (4) and (6)

(9) $\Delta \vdash \cup_{i \in L} \Gamma_i\ \mathsf{ctx}$ by Definition B.1 over (7), then Definition B.1 iteratively

(10) $\mathsf{fv}(p) = \mathrm{dom}(\Gamma) = \emptyset$ by definition and (8)

Case (B.4e)**.**

(1) $p = \texttt{injp}[\ell](p')$ by assumption

(2) $\tau = \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')$ by assumption

(3) $\Delta \vdash \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')\ \mathsf{type}$ by assumption

(4) $\Delta \vdash p' : \tau' \dashv\!\mid \Gamma$ by assumption

(5) $\Delta \vdash \tau'\ \mathsf{type}$ by Inversion of Rule (B.1f) on (3)

(6) $\Delta \vdash \Gamma\ \mathsf{ctx}$ by IH on (4) and (5)

(7) $\mathsf{fv}(p') = \mathrm{dom}(\Gamma)$ by IH on (4) and (5)

(8) $\mathsf{fv}(p) = \mathrm{dom}(\Gamma)$ by definition and (7)

□

B.1.3 Structural Dynamics

The *structural dynamics* is specified as a transition system, and is organized around judgements of the following form:

Judgement Form	Description
$e \mapsto e'$	e transitions to e'
e val	e is a value
e matchfail	e raises match failure

We also define auxiliary judgements for *iterated transition*, $e \mapsto^* e'$, and *evaluation*, $e \Downarrow e'$.

Definition B.6 (Iterated Transition)**.** *Iterated transition, $e \mapsto^* e'$, is the reflexive, transitive closure of the transition judgement, $e \mapsto e'$.*

Definition B.7 (Evaluation)**.** *$e \Downarrow e'$ iff $e \mapsto^* e'$ and e'* val*.*

Our subsequent developments do not make mention of particular rules in the dynamics, nor do they make mention of other judgements, not listed above, that are used only for defining the dynamics of the match operator, so we do not produce these details here. Instead, it suffices to state the following conditions.

Condition B.8 (Canonical Forms)**.** *If $\vdash e : \tau$ and e* val *then:*

1. *If $\tau = \texttt{parr}(\tau_1; \tau_2)$ then $e = \texttt{lam}\{\tau_1\}(x.e')$ and $x : \tau_1 \vdash e' : \tau_2$.*
2. *If $\tau = \texttt{all}(t.\tau')$ then $e = \texttt{tlam}(t.e')$ and t* type $\vdash e' : \tau'$.
3. *If $\tau = \texttt{rec}(t.\tau')$ then $e = \texttt{fold}(e')$ and $\vdash e' : [\texttt{rec}(t.\tau')/t]\tau'$ and e'* val*.*
4. *If $\tau = \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ then $e = \texttt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L})$ and $\vdash e_i : \tau_i$ and e_i* val *for each $i \in L$.*
5. *If $\tau = \texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ then for some label set L' and label ℓ and type τ', we have that $L = L', \ell$ and $\tau = \texttt{sum}[L', \ell](\{i \hookrightarrow \tau_i\}_{i \in L'}; \ell \hookrightarrow \tau')$ and $e = \texttt{inj}[\ell](e')$ and $\vdash e' : \tau'$ and e'* val*.*

Condition B.9 (Preservation)**.** *If $\vdash e : \tau$ and $e \mapsto e'$ then $\vdash e' : \tau$.*

Condition B.10 (Progress)**.** *If $\vdash e : \tau$ then either e* val *or e* matchfail *or there exists an e' such that $e \mapsto e'$.*

B.2 Unexpanded Language (UL)

B.2.1 Syntax

Stylized Syntax

Sort			Stylized Form	Description
UTyp	$\hat{\tau}$	::=	$\hat{t}$	identifier
			$\hat{\tau} \rightharpoonup \hat{\tau}$	partial function
			$\forall \hat{t}.\hat{\tau}$	polymorphic
			$\mu \hat{t}.\hat{\tau}$	recursive
			$\langle \{i \hookrightarrow \hat{\tau}_i\}_{i \in L} \rangle$	labeled product
			$[\{i \hookrightarrow \hat{\tau}_i\}_{i \in L}]$	labeled sum
UExp	$\hat{e}$	::=	$\hat{x}$	identifier
			$\hat{e} : \hat{\tau}$	ascription
			$\texttt{let val}\ \hat{x} = \hat{e}\ \texttt{in}\ \hat{e}$	value binding
			$\lambda \hat{x}{:}\hat{\tau}.\hat{e}$	abstraction
			$\hat{e}(\hat{e})$	application
			$\Lambda \hat{t}.\hat{e}$	type abstraction
			$\hat{e}[\hat{\tau}]$	type application
			$\texttt{fold}(\hat{e})$	fold
			$\texttt{unfold}(\hat{e})$	unfold
			$\langle \{i \hookrightarrow \hat{e}_i\}_{i \in L} \rangle$	labeled tuple
			$\hat{e} \cdot \ell$	projection
			$\texttt{inj}[\ell](\hat{e})$	injection
			$\texttt{case}\ \hat{e}\ \{i \hookrightarrow \hat{x}_i.\hat{e}_i\}_{i \in L}$	case analysis
			$\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\tau}\ \texttt{by static}\ e\ \texttt{in}\ \hat{e}$	seTSM definition
			$\hat{a}\ \texttt{`}b\texttt{`}$	seTSM application
			$\texttt{match}\ \hat{e}\ \{\hat{r}_i\}_{1 \leq i \leq n}$	match
			$\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\tau}\ \texttt{for patterns by static}\ e\ \texttt{in}\ \hat{e}$	spTSM definition
URule	$\hat{r}$	::=	$\hat{p} \Rightarrow \hat{e}$	match rule
UPat	$\hat{p}$	::=	$\hat{x}$	identifier pattern
			_	wildcard pattern
			$\texttt{fold}(\hat{p})$	fold pattern
			$\langle \{i \hookrightarrow \hat{p}_i\}_{i \in L} \rangle$	labeled tuple pattern
			$\texttt{inj}[\ell](\hat{p})$	injection pattern
			$\hat{a}\ \texttt{`}b\texttt{`}$	spTSM application

Body Lengths We write $\|b\|$ for the length of b. The metafunction $\|\hat{e}\|$ computes the sum of the lengths of expression literal bodies in $\hat{e}$:

$$
\begin{array}{ll}
\|\hat{x}\| & = 0 \\
\|\hat{e} : \hat{\tau}\| & = \|\hat{e}\| \\
\|\texttt{let val}\ \hat{x} = \hat{e}_1\ \texttt{in}\ \hat{e}_2\| & = \|\hat{e}_1\| + \|\hat{e}_2\| \\
\|\lambda \hat{x}{:}\hat{\tau}.\hat{e}\| & = \|\hat{e}\| \\
\|\hat{e}_1(\hat{e}_2)\| & = \|\hat{e}_1\| + \|\hat{e}_2\| \\
\|\Lambda \hat{t}.\hat{e}\| & = \|\hat{e}\| \\
\|\hat{e}\texttt{[}\hat{\tau}\texttt{]}\| & = \|\hat{e}\| \\
\|\texttt{fold}(\hat{e})\| & = \|\hat{e}\| \\
\|\texttt{unfold}(\hat{e})\| & = \|\hat{e}\| \\
\|\langle \{i \hookrightarrow \hat{e}_i\}_{i \in L} \rangle\| & = \textstyle\sum_{i \in L} \|\hat{e}_i\| \\
\|\ell \cdot \hat{e}\| & = \|\hat{e}\| \\
\|\texttt{inj}[\ell](\hat{e})\| & = \|\hat{e}\| \\
\|\texttt{case}\ \hat{e}\ \{i \hookrightarrow \hat{x}_i.\hat{e}_i\}_{i \in L}\| & = \|\hat{e}\| + \textstyle\sum_{i \in L} \|\hat{e}_i\| \\
\|\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\tau}\ \texttt{by static}\ e_{\text{parse}}\ \texttt{in}\ \hat{e}\| & = \|\hat{e}\| \\
\|\hat{a}\ \texttt{`}b\texttt{`}\| & = \|b\| \\
\|\texttt{match}\ \hat{e}\ \{\hat{r}_i\}_{1 \leq i \leq n}\| & = \|\hat{e}\| + \textstyle\sum_{1 \leq i \leq n} \|r_i\| \\
\|\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\tau}\ \texttt{for patterns by static}\ e_{\text{parse}}\ \texttt{in}\ \hat{e}\| & = \|\hat{e}\|
\end{array}
$$

and $\|\hat{r}\|$ computes the sum of the lengths of expression literal bodies in $\hat{r}$:

$$\|\hat{p} \Rightarrow \hat{e}\| = \|\hat{e}\|$$

Similarly, the metafunction $\|\hat{p}\|$ computes the sum of the lengths of the pattern literal bodies in $\hat{p}$:

$$
\begin{aligned}
\|\hat{x}\| &= 0 \\
\|\texttt{fold(}\hat{p}\texttt{)}\| &= \|\hat{p}\| \\
\|\langle \{i \hookrightarrow \hat{p}_i\}_{i \in L} \rangle\| &= \sum_{i \in L} \|\hat{p}_i\| \\
\|\texttt{inj}[\ell](\hat{p})\| &= \|\hat{p}\| \\
\|\hat{a}\ \texttt{`}b\texttt{`}\| &= \|b\|
\end{aligned}
$$

Common Unexpanded Forms Each expanded form maps onto an unexpanded form. We refer to these as the *common forms*. In particular:

- Each type variable, t, maps onto a unique type identifier, written $\widehat{t}$.

- Each type, τ, maps onto an unexpanded type, $\mathcal{U}(\tau)$, as follows:

$$\begin{aligned}
\mathcal{U}(t) &= \widehat{t} \\
\mathcal{U}(\texttt{parr(}\tau_1\texttt{;}\tau_2\texttt{)}) &= \mathcal{U}(\tau_1) \rightharpoonup \mathcal{U}(\tau_2) \\
\mathcal{U}(\texttt{all(}t\texttt{.}\tau\texttt{)}) &= \forall\widehat{t}.\mathcal{U}(\tau) \\
\mathcal{U}(\texttt{rec(}t\texttt{.}\tau\texttt{)}) &= \mu\widehat{t}.\mathcal{U}(\tau) \\
\mathcal{U}(\texttt{prod}[L]\texttt{(}\{i \hookrightarrow \tau_i\}_{i\in L}\texttt{)}) &= \langle\{i \hookrightarrow \mathcal{U}(\tau_i)\}_{i\in L}\rangle \\
\mathcal{U}(\texttt{sum}[L]\texttt{(}\{i \hookrightarrow \tau_i\}_{i\in L}\texttt{)}) &= [\{i \hookrightarrow \mathcal{U}(\tau_i)\}_{i\in L}]
\end{aligned}$$

- Each expression variable, x, maps onto a unique expression identifier, written $\widehat{x}$.
- Each expanded expression, e, maps onto an unexpanded expression, $\mathcal{U}(e)$, as follows:

$$\begin{aligned}
\mathcal{U}(x) &= \widehat{x} \\
\mathcal{U}(\texttt{lam\{}\tau\texttt{\}(}x\texttt{.}e\texttt{)}) &= \lambda\widehat{x}{:}\mathcal{U}(\tau).\mathcal{U}(e) \\
\mathcal{U}(\texttt{ap(}e_1\texttt{;}e_2\texttt{)}) &= \mathcal{U}(e_1)(\mathcal{U}(e_2)) \\
\mathcal{U}(\texttt{tlam(}t\texttt{.}e\texttt{)}) &= \Lambda\widehat{t}.\mathcal{U}(e) \\
\mathcal{U}(\texttt{tap\{}\tau\texttt{\}(}e\texttt{)}) &= \mathcal{U}(e)\texttt{[}\mathcal{U}(\tau)\texttt{]} \\
\mathcal{U}(\texttt{fold(}e\texttt{)}) &= \texttt{fold}(\mathcal{U}(e)) \\
\mathcal{U}(\texttt{unfold(}e\texttt{)}) &= \texttt{unfold}(\mathcal{U}(e)) \\
\mathcal{U}(\texttt{tpl}[L]\texttt{(}\{i \hookrightarrow e_i\}_{i\in L}\texttt{)}) &= \langle\{i \hookrightarrow \mathcal{U}(e_i)\}_{i\in L}\rangle \\
\mathcal{U}(\texttt{prj}[\ell]\texttt{(}e\texttt{)}) &= \mathcal{U}(e)\cdot\ell \\
\mathcal{U}(\texttt{inj}[\ell]\texttt{(}e\texttt{)}) &= \texttt{inj}[\ell](\mathcal{U}(e)) \\
\mathcal{U}(\texttt{match}[n]\texttt{(}e\texttt{;}\{r_i\}_{1\le i\le n}\texttt{)}) &= \texttt{match}\,\mathcal{U}(e)\ \{\mathcal{U}(r_i)\}_{1\le i\le n}
\end{aligned}$$

- Each expanded rule, r, maps onto an unexpanded rule, $\mathcal{U}(r)$, as follows:

$$\mathcal{U}(\texttt{rule(}p\texttt{.}e\texttt{)}) = \texttt{urule(}\mathcal{U}(p)\texttt{.}\mathcal{U}(e)\texttt{)}$$

- Each expanded pattern, p, maps onto the unexpanded pattern, $\mathcal{U}(p)$, as follows:

$$\begin{aligned}
\mathcal{U}(x) &= \widehat{x} \\
\mathcal{U}(\texttt{wildp}) &= \texttt{uwildp} \\
\mathcal{U}(\texttt{foldp(}p\texttt{)}) &= \texttt{ufoldp(}\mathcal{U}(p)\texttt{)} \\
\mathcal{U}(\texttt{tplp}[L]\texttt{(}\{i \hookrightarrow p_i\}_{i\in L}\texttt{)}) &= \texttt{utplp}[L]\texttt{(}\{i \hookrightarrow \mathcal{U}(p_i)\}_{i\in L}\texttt{)} \\
\mathcal{U}(\texttt{injp}[\ell]\texttt{(}p\texttt{)}) &= \texttt{uinjp}[\ell]\texttt{(}\mathcal{U}(p)\texttt{)}
\end{aligned}$$

Textual Syntax

In addition to the stylized syntax, there is also a context-free textual syntax for the UL. For our purposes, we need only posit the existence of partial metafunctions $\mathsf{parseUTyp}(b)$ and $\mathsf{parseUExp}(b)$ and $\mathsf{parseUPat}(b)$.

Condition B.11 (Textual Representability)**.**

1. *For each* $\hat{\tau}$*, there exists* b *such that* $\mathsf{parseUTyp}(b) = \hat{\tau}$.
2. *For each* $\hat{e}$*, there exists* b *such that* $\mathsf{parseUExp}(b) = \hat{e}$.
3. *For each* $\hat{p}$*, there exists* b *such that* $\mathsf{parseUPat}(b) = \hat{p}$.

We also impose the following technical conditions.

Condition B.12 (Expression Parsing Monotonicity). *If* $\mathsf{parseUExp}(b) = \hat{e}$ *then* $\|\hat{e}\| < \|b\|$.

Condition B.13 (Pattern Parsing Monotonicity). *If* $\mathsf{parseUPat}(b) = \hat{p}$ *then* $\|\hat{p}\| < \|b\|$.

B.2.2 Type Expansion

Unexpanded type formation contexts, $\hat{\Delta}$, are of the form $\langle \mathcal{D}; \Delta \rangle$, i.e. they consist of a *type identifier expansion context*, $\mathcal{D}$, paired with a type formation context, Δ.

A *type identifier expansion context*, $\mathcal{D}$, is a finite function that maps each type identifier $\hat{t} \in \mathrm{dom}(\mathcal{D})$ to the hypothesis $\hat{t} \rightsquigarrow t$, for some type variable t. We write $\mathcal{D} \uplus \hat{t} \rightsquigarrow t$ for the type identifier expansion context that maps $\hat{t}$ to $\hat{t} \rightsquigarrow t$ and defers to $\mathcal{D}$ for all other type identifiers (i.e. the previous mapping is *updated*.)

We define $\hat{\Delta}, \hat{t} \rightsquigarrow t\ \mathsf{type}$ when $\hat{\Delta} = \langle \mathcal{D}; \Delta \rangle$ as an abbreviation of

$$\langle \mathcal{D} \uplus \hat{t} \rightsquigarrow t; \Delta, t\ \mathsf{type} \rangle$$

Definition B.14 (Unexpanded Type Formation Context Formation). $\vdash \langle \mathcal{D}; \Delta \rangle\ \mathsf{utctx}$ *iff for each* $\hat{t} \rightsquigarrow t\ \mathsf{type} \in \mathcal{D}$ *we have* $t\ \mathsf{type} \in \Delta$.

$\boxed{\hat{\Delta} \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type}}$ $\hat{\tau}$ has well-formed expansion τ

$$\frac{}{\hat{\Delta}, \hat{t} \rightsquigarrow t\ \mathsf{type} \vdash \hat{t} \rightsquigarrow t\ \mathsf{type}} \tag{B.5a}$$

$$\frac{\hat{\Delta} \vdash \hat{\tau}_1 \rightsquigarrow \tau_1\ \mathsf{type} \qquad \hat{\Delta} \vdash \hat{\tau}_2 \rightsquigarrow \tau_2\ \mathsf{type}}{\hat{\Delta} \vdash \mathtt{uparr}(\hat{\tau}_1; \hat{\tau}_2) \rightsquigarrow \mathtt{parr}(\tau_1; \tau_2)\ \mathsf{type}} \tag{B.5b}$$

$$\frac{\hat{\Delta}, \hat{t} \rightsquigarrow t\ \mathsf{type} \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type}}{\hat{\Delta} \vdash \mathtt{uall}(\hat{t}.\hat{\tau}) \rightsquigarrow \mathtt{all}(t.\tau)\ \mathsf{type}} \tag{B.5c}$$

$$\frac{\hat{\Delta}, \hat{t} \rightsquigarrow t\ \mathsf{type} \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type}}{\hat{\Delta} \vdash \mathtt{urec}(\hat{t}.\hat{\tau}) \rightsquigarrow \mathtt{rec}(t.\tau)\ \mathsf{type}} \tag{B.5d}$$

$$\frac{\{\hat{\Delta} \vdash \hat{\tau}_i \rightsquigarrow \tau_i\ \mathsf{type}\}_{i \in L}}{\hat{\Delta} \vdash \mathtt{uprod}[L](\{i \hookrightarrow \hat{\tau}_i\}_{i \in L}) \rightsquigarrow \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})\ \mathsf{type}} \tag{B.5e}$$

$$\frac{\{\hat{\Delta} \vdash \hat{\tau}_i \rightsquigarrow \tau_i\ \mathsf{type}\}_{i \in L}}{\hat{\Delta} \vdash \mathtt{usum}[L](\{i \hookrightarrow \hat{\tau}_i\}_{i \in L}) \rightsquigarrow \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L})\ \mathsf{type}} \tag{B.5f}$$

B.2.3 Typed Expression Expansion

Contexts

Unexpanded typing contexts, $\hat{\Gamma}$, are, similarly, of the form $\langle \mathcal{G}; \Gamma \rangle$, where $\mathcal{G}$ is an *expression identifier expansion context*, and Γ is a typing context. An expression identifier expansion context, $\mathcal{G}$, is a finite function that maps each expression identifier $\hat{x} \in \mathrm{dom}(\mathcal{G})$ to the hypothesis $\hat{x} \rightsquigarrow x$, for some expression variable, x. We write $\mathcal{G} \uplus \hat{x} \rightsquigarrow x$ for the expression identifier expansion context that maps $\hat{x}$ to $\hat{x} \rightsquigarrow x$ and defers to $\mathcal{G}$ for all other expression identifiers (i.e. the previous mapping is updated.)

We define $\hat{\Gamma}, \hat{x} \rightsquigarrow x : \tau$ when $\hat{\Gamma} = \langle \mathcal{G}; \Gamma \rangle$ as an abbreviation of

$$\langle \mathcal{G} \uplus \hat{x} \rightsquigarrow x; \Gamma, x : \tau \rangle$$

Definition B.15 (Unexpanded Typing Context Formation)**.** $\Delta \vdash \langle \mathcal{G}; \Gamma \rangle$ uctx *iff* $\Delta \vdash \Gamma$ ctx *and for each* $\hat{x} \rightsquigarrow x \in \mathcal{G}$*, we have* $x \in dom(\Gamma)$*.*

Body Encoding and Decoding

An assumed type abbreviated `Body` classifies encodings of literal bodies, b. The mapping from literal bodies to values of type `Body` is defined by the *body encoding judgement* $b \downarrow_{\mathsf{Body}} e_{\mathrm{body}}$. An inverse mapping is defined by the *body decoding judgement* $e_{\mathrm{body}} \uparrow_{\mathsf{Body}} b$.

Judgement Form	**Description**
$b \downarrow_{\mathsf{Body}} e$	b has encoding e
$e \uparrow_{\mathsf{Body}} b$	e has decoding b

The following condition establishes an isomorphism between literal bodies and values of type `Body` mediated by the judgements above.

Condition B.16 (Body Isomorphism)**.**

1. *For every literal body* b*, we have that* $b \downarrow_{\mathsf{Body}} e_{body}$ *for some* e_{body} *such that* $\vdash e_{body} : \mathtt{Body}$ *and* e_{body} val*.*
2. *If* $\vdash e_{body} : \mathtt{Body}$ *and* e_{body} val *then* $e_{body} \uparrow_{\mathsf{Body}} b$ *for some* b*.*
3. *If* $b \downarrow_{\mathsf{Body}} e_{body}$ *then* $e_{body} \uparrow_{\mathsf{Body}} b$*.*
4. *If* $\vdash e_{body} : \mathtt{Body}$ *and* e_{body} val *and* $e_{body} \uparrow_{\mathsf{Body}} b$ *then* $b \downarrow_{\mathsf{Body}} e_{body}$*.*
5. *If* $b \downarrow_{\mathsf{Body}} e_{body}$ *and* $b \downarrow_{\mathsf{Body}} e'_{body}$ *then* $e_{body} = e'_{body}$*.*
6. *If* $\vdash e_{body} : \mathtt{Body}$ *and* e_{body} val *and* $e_{body} \uparrow_{\mathsf{Body}} b$ *and* $e_{body} \uparrow_{\mathsf{Body}} b'$ *then* $b = b'$*.*

We also assume a partial metafunction, $\mathsf{subseq}(b; m; n)$, which extracts a subsequence of b starting at position m and ending at position n, inclusive, where m and n are natural numbers. The following condition is technically necessary.

Condition B.17 (Body Subsequencing)**.** *If* $\mathsf{subseq}(b; m; n) = b'$ *then* $\|b'\| \leq \|b\|$*.*

Parse Results

The type abbreviated `ParseResultSE`, and an auxiliary abbreviation used below, is defined as follows:

$$L_{\texttt{SE}} \stackrel{\text{def}}{=} \texttt{ParseError}, \texttt{SuccessE}$$
$$\texttt{ParseResultSE} \stackrel{\text{def}}{=} \texttt{sum}[L_{\texttt{SE}}](\texttt{ParseError} \hookrightarrow \langle\rangle, \texttt{SuccessE} \hookrightarrow \texttt{PrExpr})$$

The type abbreviated `ParseResultSP`, and an auxiliary abbreviation used below, is defined as follows:

$$L_{\texttt{SP}} \stackrel{\text{def}}{=} \texttt{ParseError}, \texttt{SuccessP}$$
$$\texttt{ParseResultSE} \stackrel{\text{def}}{=} \texttt{sum}[L_{\texttt{SP}}](\texttt{ParseError} \hookrightarrow \langle\rangle, \texttt{SuccessP} \hookrightarrow \texttt{PrPat})$$

seTSM Contexts

seTSM contexts, $\hat{\Psi}$, are of the form $\langle \mathcal{A}; \Psi \rangle$, where $\mathcal{A}$ is a *TSM identifier expansion context* and Ψ is a *seTSM definition context*.

A *TSM identifier expansion context*, $\mathcal{A}$, is a finite function mapping each TSM identifier $\hat{a} \in \text{dom}(\mathcal{A})$ to the *TSM identifier expansion*, $\hat{a} \rightsquigarrow a$, for some *TSM name*, a. We write $\mathcal{A} \uplus \hat{a} \rightsquigarrow a$ for the TSM identifier expansion context that maps $\hat{a}$ to $\hat{a} \rightsquigarrow a$, and defers to $\mathcal{A}$ for all other TSM identifiers (i.e. the previous mapping is *updated*.)

An *seTSM definition context*, Ψ, is a finite function mapping each TSM name $a \in \text{dom}(\Psi)$ to an *expanded seTSM definition*, $a \hookrightarrow \texttt{setsm}(\tau; e_{\text{parse}})$, where τ is the seTSM's type annotation, and e_{parse} is its parse function. We write $\Psi, a \hookrightarrow \texttt{setsm}(\tau; e_{\text{parse}})$ when $a \notin \text{dom}(\Psi)$ for the extension of Ψ that maps a to $a \hookrightarrow \texttt{setsm}(\tau; e_{\text{parse}})$. We write $\Delta \vdash \Psi$ seTSMs when all the type annotations in Ψ are well-formed assuming Δ, and the parse functions in Ψ are closed and of the appropriate type.

Definition B.18 (seTSM Definition Context Formation)**.** $\Delta \vdash \Psi$ seTSMs *iff for each* $a \hookrightarrow \texttt{setsm}(\tau; e_{parse}) \in \Psi$*, we have* $\Delta \vdash \tau$ type *and* $\emptyset\,\emptyset \vdash e_{parse} : \texttt{parr}(\texttt{Body}; \texttt{ParseResultSE})$*.*

Definition B.19 (seTSM Context Formation)**.** $\Delta \vdash \langle \mathcal{A}; \Psi \rangle$ seTSMctx *iff* $\Delta \vdash \Psi$ seTSMs *and for each* $\hat{a} \rightsquigarrow a \in \mathcal{A}$ *we have* $a \in dom(\Psi)$*.*

We define $\hat{\Psi}, \hat{a} \rightsquigarrow a \hookrightarrow \texttt{setsm}(\tau; e_{\text{parse}})$, when $\hat{\Psi} = \langle \mathcal{A}; \Phi \rangle$, as an abbreviation of

$$\langle \mathcal{A} \uplus \hat{a} \rightsquigarrow a; \Psi, a \hookrightarrow \texttt{setsm}(\tau; e_{\text{parse}}) \rangle$$

spTSM Contexts

spTSM contexts, $\hat{\Phi}$, are of the form $\langle \mathcal{A}; \Phi \rangle$, where $\mathcal{A}$ is a TSM identifier expansion context, defined above, and Φ is a *spTSM definition context*.

An *spTSM definition context*, Φ, is a finite function mapping each TSM name $a \in \mathrm{dom}(\Phi)$ to an *expanded seTSM definition*, $a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}})$, where τ is the spTSM's type annotation, and e_{parse} is its parse function. We write $\Phi, a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}})$ when $a \notin \mathrm{dom}(\Phi)$ for the extension of Φ that maps a to $a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}})$. We write $\Delta \vdash \Phi$ spTSMs when all the type annotations in Φ are well-formed assuming Δ, and the parse functions in Φ are closed and of the appropriate type.

Definition B.20 (spTSM Definition Context Formation). $\Delta \vdash \Phi$ spTSMs *iff for each* $a \hookrightarrow \texttt{sptsm}(\tau; e_{parse}) \in \Phi$, *we have* $\Delta \vdash \tau$ type *and* $\emptyset\ \emptyset \vdash e_{parse} : \texttt{parr}(\texttt{Body}; \texttt{ParseResultSP})$.

Definition B.21 (spTSM Context Formation). $\Delta \vdash \langle \mathcal{A}; \Phi \rangle$ spTSMctx *iff* $\Delta \vdash \Phi$ spTSMs *and for each* $\hat{a} \rightsquigarrow a \in \mathcal{A}$ *we have* $a \in dom(\Phi)$.

We define $\hat{\Phi}, \hat{a} \rightsquigarrow a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}})$, when $\hat{\Phi} = \langle \mathcal{A}; \Phi \rangle$, as an abbreviation of

$$\langle \mathcal{A} \uplus \hat{a} \rightsquigarrow a; \Phi, a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}}) \rangle$$

Typed Expression Expansion

$\boxed{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau}$ $\hat{e}$ has expansion e of type τ

$$\frac{}{\hat{\Delta}\ \hat{\Gamma}, \hat{x} \rightsquigarrow x : \tau \vdash_{\hat{\Psi};\hat{\Phi}} \hat{x} \rightsquigarrow x : \tau} \tag{B.6a}$$

$$\frac{\hat{\Delta} \vdash \hat{\tau} \rightsquigarrow \tau \text{ type} \qquad \hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} : \hat{\tau} \rightsquigarrow e : \tau} \tag{B.6b}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1 \rightsquigarrow e_1 : \tau_1 \qquad \hat{\Delta}\ \hat{\Gamma}, \hat{x} \rightsquigarrow x : \tau_1 \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_2 \rightsquigarrow e_2 : \tau_2}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \texttt{let val}\ \hat{x} = \hat{e}_1\ \texttt{in}\ \hat{e}_2 \rightsquigarrow \texttt{ap}(\texttt{lam}\{\tau_1\}(x.e_2); e_1) : \tau_2} \tag{B.6c}$$

$$\frac{\hat{\Delta} \vdash \hat{\tau} \rightsquigarrow \tau \text{ type} \qquad \hat{\Delta}\ \hat{\Gamma}, \hat{x} \rightsquigarrow x : \tau \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \lambda \hat{x}{:}\hat{\tau}.\hat{e} \rightsquigarrow \texttt{lam}\{\tau\}(x.e) : \texttt{parr}(\tau; \tau')} \tag{B.6d}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1 \rightsquigarrow e_1 : \texttt{parr}(\tau; \tau') \qquad \hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_2 \rightsquigarrow e_2 : \tau}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1(\hat{e}_2) \rightsquigarrow \texttt{ap}(e_1; e_2) : \tau'} \tag{B.6e}$$

$$\frac{\hat{\Delta}, \hat{t} \rightsquigarrow t \text{ type}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \Lambda \hat{t}.\hat{e} \rightsquigarrow \texttt{tlam}(t.e) : \texttt{all}(t.\tau)} \tag{B.6f}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{all}(t.\tau) \qquad \hat{\Delta} \vdash \hat{\tau}' \rightsquigarrow \tau' \ \mathsf{type}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}[\hat{\tau}'] \rightsquigarrow \mathtt{tap}\{\tau'\}(e) : [\tau'/t]\tau} \tag{B.6g}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [\mathtt{rec}(t.\tau)/t]\tau}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{fold}(\hat{e}) \rightsquigarrow \mathtt{fold}(e) : \mathtt{rec}(t.\tau)} \tag{B.6h}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{rec}(t.\tau)}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{unfold}(\hat{e}) \rightsquigarrow \mathtt{unfold}(e) : [\mathtt{rec}(t.\tau)/t]\tau} \tag{B.6i}$$

$$\frac{\{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_i \rightsquigarrow e_i : \tau_i\}_{i \in L}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \langle \{i \hookrightarrow \hat{e}_i\}_{i \in L} \rangle \rightsquigarrow \mathtt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L}) : \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})} \tag{B.6j}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{prod}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau)}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \cdot \ell \rightsquigarrow \mathtt{prj}[\ell](e) : \tau} \tag{B.6k}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{inj}[\ell](\hat{e}) \rightsquigarrow \mathtt{inj}[\ell](e) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')} \tag{B.6l}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \qquad \{\hat{\Delta}\ \hat{\Gamma}, \hat{x}_i \rightsquigarrow x_i : \tau_i \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_i \rightsquigarrow e_i : \tau\}_{i \in L}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{case}\ \hat{e}\ \{i \hookrightarrow \hat{x}_i.\hat{e}_i\}_{i \in L} \rightsquigarrow \mathtt{case}[L](e; \{i \hookrightarrow x_i.e_i\}_{i \in L}) : \tau} \tag{B.6m}$$

$$\frac{\begin{array}{c}\hat{\Delta} \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \varnothing\,\varnothing \vdash e_{\mathrm{parse}} : \mathtt{parr}(\mathtt{Body}; \mathtt{ParseResultSE}) \\ e_{\mathrm{parse}} \Downarrow e'_{\mathrm{parse}} \qquad \hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi}, \hat{a} \rightsquigarrow a \hookrightarrow \mathtt{setsm}(\tau; e'_{\mathrm{parse}})} \hat{e} \rightsquigarrow e : \tau'\end{array}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{syntax}\ \hat{a}\ \mathtt{at}\ \hat{\tau}\ \mathtt{by}\ \mathtt{static}\ e_{\mathrm{parse}}\ \mathtt{in}\ \hat{e} \rightsquigarrow e : \tau'} \tag{B.6n}$$

$$\frac{\begin{array}{c}\hat{\Psi} = \hat{\Psi}', \hat{a} \rightsquigarrow a \hookrightarrow \mathtt{setsm}(\tau; e_{\mathrm{parse}}) \\ b \downarrow_{\mathsf{Body}} e_{\mathrm{body}} \qquad e_{\mathrm{parse}}(e_{\mathrm{body}}) \Downarrow \mathtt{inj}[\mathtt{SuccessE}](e_{\mathrm{proto}}) \qquad e_{\mathrm{proto}} \uparrow_{\mathsf{PrExpr}} \grave{e} \\ \mathsf{seg}(\grave{e})\ \mathsf{segments}\ b \qquad \varnothing\,\varnothing \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau\end{array}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow e : \tau} \tag{B.6o}$$

$$\frac{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau \qquad \{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r}_i \rightsquigarrow r_i : \tau \Rightarrow \tau'\}_{1 \le i \le n}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{match}\ \hat{e}\ \{\hat{r}_i\}_{1 \le i \le n} \rightsquigarrow \mathtt{match}[n](e; \{r_i\}_{1 \le i \le n}) : \tau'} \tag{B.6p}$$

$$\frac{\begin{array}{c}\hat{\Delta} \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \varnothing\,\varnothing \vdash e_{\mathrm{parse}} : \mathtt{parr}(\mathtt{Body}; \mathtt{ParseResultSP}) \\ e_{\mathrm{parse}} \Downarrow e'_{\mathrm{parse}} \qquad \hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}, \hat{a} \rightsquigarrow a \hookrightarrow \mathtt{sptsm}(\tau; e'_{\mathrm{parse}})} \hat{e} \rightsquigarrow e : \tau'\end{array}}{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{syntax}\ \hat{a}\ \mathtt{at}\ \hat{\tau}\ \mathtt{for}\ \mathtt{patterns}\ \mathtt{by}\ \mathtt{static}\ e_{\mathrm{parse}}\ \mathtt{in}\ \hat{e} \rightsquigarrow e : \tau'} \tag{B.6q}$$

$\boxed{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r} \rightsquigarrow r : \tau \Rrightarrow \tau'}$ $\hat{r}$ has expansion r taking values of type τ to values of type τ'

$$\frac{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \langle \mathcal{G}'; \Gamma' \rangle \qquad \hat{\Delta}\ \langle \mathcal{G} \uplus \mathcal{G}'; \Gamma \cup \Gamma' \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'}{\hat{\Delta}\ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{urule}(\hat{p}.\hat{e}) \rightsquigarrow \mathtt{rule}(p.e) : \tau \Rrightarrow \tau'} \tag{B.7}$$

Typed Pattern Expansion

$\boxed{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Gamma}}$ $\hat{p}$ has expansion p matching against τ generating hypotheses $\hat{\Gamma}$

$$\frac{}{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{x} \rightsquigarrow x : \tau \dashv\!\shortmid \langle \hat{x} \rightsquigarrow x; x : \tau \rangle} \tag{B.8a}$$

$$\frac{}{\hat{\Delta} \vdash_{\hat{\Phi}} \_ \rightsquigarrow \mathtt{wildp} : \tau \dashv\!\shortmid \langle \emptyset; \emptyset \rangle} \tag{B.8b}$$

$$\frac{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : [\mathtt{rec}(t.\tau)/t]\tau \dashv\!\shortmid \hat{\Gamma}}{\hat{\Delta} \vdash_{\hat{\Phi}} \mathtt{fold}(\hat{p}) \rightsquigarrow \mathtt{foldp}(p) : \mathtt{rec}(t.\tau) \dashv\!\shortmid \hat{\Gamma}} \tag{B.8c}$$

$$\frac{\begin{array}{c}\tau = \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \\ \{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p}_i \rightsquigarrow p_i : \tau_i \dashv\!\shortmid \hat{\Gamma}_i\}_{i \in L}\end{array}}{\hat{\Delta} \vdash_{\hat{\Phi}} \langle \{i \hookrightarrow \hat{p}_i\}_{i \in L} \rangle \rightsquigarrow \mathtt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) : \tau \dashv\!\shortmid \uplus_{i \in L} \hat{\Gamma}_i} \tag{B.8d}$$

$$\frac{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Gamma}}{\hat{\Delta} \vdash_{\hat{\Phi}} \mathtt{inj}[\ell](\hat{p}) \rightsquigarrow \mathtt{injp}[\ell](p) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau) \dashv\!\shortmid \hat{\Gamma}} \tag{B.8e}$$

$$\frac{\begin{array}{c}\hat{\Phi} = \hat{\Phi}', \hat{a} \rightsquigarrow a \hookrightarrow \mathtt{sptsm}(\tau; e_{\text{parse}}) \\ b \downarrow_{\mathsf{Body}} e_{\text{body}} \qquad e_{\text{parse}}(e_{\text{body}}) \Downarrow \mathtt{inj}[\mathtt{SuccessP}](e_{\text{proto}}) \qquad e_{\text{proto}} \uparrow_{\mathsf{PrPat}} \hat{p} \\ \mathsf{seg}(\hat{p})\ \mathsf{segments}\ b \qquad \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid^{\hat{\Delta};\hat{\Phi};b} \hat{\Gamma}\end{array}}{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Gamma}} \tag{B.8f}$$

In Rule (B.8d), $\hat{\Gamma}_i$ is shorthand for $\langle \mathcal{G}_i; \Gamma_i \rangle$ and $\uplus_{i \in L} \hat{\Gamma}_i$ is shorthand for

$$\langle \uplus_{i \in L} \mathcal{G}_i; \cup_{i \in L} \Gamma_i \rangle$$

B.3 Proto-Expansion Validation

B.3.1 Syntax of Proto-Expansions

Sort		Operational Form	Stylized Form	Description
PrTyp	$\grave{\tau}$ ::=	t	t	variable
		$\texttt{prparr}(\grave{\tau};\grave{\tau})$	$\grave{\tau} \rightharpoonup \grave{\tau}$	partial function
		$\texttt{prall}(t.\grave{\tau})$	$\forall t.\grave{\tau}$	polymorphic
		$\texttt{prrec}(t.\grave{\tau})$	$\mu t.\grave{\tau}$	recursive
		$\texttt{prprod}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i\in L})$	$\langle\{i \hookrightarrow \grave{\tau}_i\}_{i\in L}\rangle$	labeled product
		$\texttt{prsum}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i\in L})$	$[\{i \hookrightarrow \grave{\tau}_i\}_{i\in L}]$	labeled sum
		$\texttt{splicedt}[m;n]$	$\texttt{splicedt}[m;n]$	spliced type ref.
PrExp	$\grave{e}$ ::=	x	x	variable
		$\texttt{prasc}\{\grave{\tau}\}(\grave{e})$	$\grave{e} : \grave{\tau}$	ascription
		$\texttt{prletval}(\grave{e}; x.\grave{e})$	$\texttt{let val}\ x = \grave{e}\ \texttt{in}\ \grave{e}$	value binding
		$\texttt{prlam}\{\grave{\tau}\}(x.\grave{e})$	$\lambda x{:}\grave{\tau}.\grave{e}$	abstraction
		$\texttt{prap}(\grave{e};\grave{e})$	$\grave{e}(\grave{e})$	application
		$\texttt{prtlam}(t.\grave{e})$	$\Lambda t.\grave{e}$	type abstraction
		$\texttt{prtap}\{\grave{\tau}\}(\grave{e})$	$\grave{e}\texttt{[}\grave{\tau}\texttt{]}$	type application
		$\texttt{prfold}(\grave{e})$	$\texttt{fold}(\grave{e})$	fold
		$\texttt{prunfold}(\grave{e})$	$\texttt{unfold}(\grave{e})$	unfold
		$\texttt{prtpl}\{L\}(\{i \hookrightarrow \grave{e}_i\}_{i\in L})$	$\langle\{i \hookrightarrow \grave{e}_i\}_{i\in L}\rangle$	labeled tuple
		$\texttt{prprj}[\ell](\grave{e})$	$\grave{e} \cdot \ell$	projection
		$\texttt{prinj}[\ell](\grave{e})$	$\texttt{inj}[\ell](\grave{e})$	injection
		$\texttt{prcase}[L](\grave{e};\{i \hookrightarrow x_i.\grave{e}_i\}_{i\in L})$	$\texttt{case}\ \grave{e}\ \{i \hookrightarrow x_i.\grave{e}_i\}_{i\in L}$	case analysis
		$\texttt{splicede}[m;n;\grave{\tau}]$	$\texttt{splicede}[m;n;\grave{\tau}]$	spliced expr. ref.
		$\texttt{prmatch}[n](\grave{e};\{\grave{r}_i\}_{1\le i\le n})$	$\texttt{match}\ \grave{e}\ \{\grave{r}_i\}_{1\le i\le n}$	match
PrRule	$\grave{r}$ ::=	$\texttt{prrule}(p.\grave{e})$	$p \Rightarrow \grave{e}$	rule
PrPat	$\grave{p}$ ::=	$\texttt{prwildp}$	$\_$	wildcard pattern
		$\texttt{prfoldp}(p)$	$\texttt{fold}(p)$	fold pattern
		$\texttt{prtplp}[L](\{i \hookrightarrow \grave{p}_i\}_{i\in L})$	$\langle\{i \hookrightarrow \grave{p}_i\}_{i\in L}\rangle$	labeled tuple pattern
		$\texttt{prinjp}[\ell](\grave{p})$	$\texttt{inj}[\ell](\grave{p})$	injection pattern
		$\texttt{splicedp}[m;n;\grave{\tau}]$	$\texttt{splicedp}[m;n;\grave{\tau}]$	spliced pattern ref.

Common Proto-Expansion Terms

Each expanded term, except variable patterns, maps onto a proto-expansion term. We refer to these as the *common proto-expansion terms*. In particular:

- Each type, τ, maps onto a proto-type, $\mathcal{P}(\tau)$, as follows:

$$\begin{aligned}
\mathcal{P}(t) &= t \\
\mathcal{P}(\texttt{parr}(\tau_1;\tau_2)) &= \texttt{prparr}(\mathcal{P}(\tau_1);\mathcal{P}(\tau_2)) \\
\mathcal{P}(\texttt{all}(t.\tau)) &= \texttt{prall}(t.\mathcal{P}(\tau)) \\
\mathcal{P}(\texttt{rec}(t.\tau)) &= \texttt{prrec}(t.\mathcal{P}(\tau)) \\
\mathcal{P}(\texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i\in L})) &= \texttt{prprod}[L](\{i \hookrightarrow \mathcal{P}(\tau_i)\}_{i\in L}) \\
\mathcal{P}(\texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i\in L})) &= \texttt{prsum}[L](\{i \hookrightarrow \mathcal{P}(\tau_i)\}_{i\in L})
\end{aligned}$$

- Each expanded expression, e, maps onto a proto-expression, $\mathcal{P}(e)$, as follows:

$$\begin{aligned}
\mathcal{P}(x) &= x \\
\mathcal{P}(\texttt{lam\{}\tau\texttt{\}(}x.e\texttt{)}) &= \texttt{prlam\{}\mathcal{P}(\tau)\texttt{\}(}x.\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{ap(}e_1;e_2\texttt{)}) &= \texttt{prap(}\mathcal{P}(e_1);\mathcal{P}(e_2)\texttt{)} \\
\mathcal{P}(\texttt{tlam(}t.e\texttt{)}) &= \texttt{prtlam(}t.\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{tap\{}\tau\texttt{\}(}e\texttt{)}) &= \texttt{prtap\{}\mathcal{P}(\tau)\texttt{\}(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{fold(}e\texttt{)}) &= \texttt{prfold(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{unfold(}e\texttt{)}) &= \texttt{prunfold(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{tpl}[L](\{i \hookrightarrow e_i\}_{i\in L})) &= \texttt{prtpl\{}L\texttt{\}(}\{i \hookrightarrow \mathcal{P}(e_i)\}_{i\in L}\texttt{)} \\
\mathcal{P}(\texttt{inj}[\ell](e)) &= \texttt{prinj}[\ell](\mathcal{P}(e)) \\
\mathcal{P}(\texttt{match}[n](e;\{r_i\}_{1\le i\le n})) &= \texttt{prmatch}[n](\mathcal{P}(e);\{\mathcal{P}(r_i)\}_{1\le i\le n})
\end{aligned}$$

- Each expanded rule, r, maps onto the proto-rule, $\mathcal{P}(r)$, as follows:

$$\mathcal{P}(\texttt{rule(}p.e\texttt{)}) = \texttt{prrule(}p.\mathcal{P}(e)\texttt{)}$$

  Notice that proto-rules bind expanded patterns, not proto-patterns. This is because proto-rules appear in proto-expressions, which are generated by seTSMs. It would not be sensible for an seTSM to splice a pattern out of a literal body.

- Each expanded pattern, p, except for the variable patterns, maps onto a proto-pattern, $\mathcal{P}(p)$, as follows:

$$\begin{aligned}
\mathcal{P}(\texttt{wildp}) &= \texttt{prwildp} \\
\mathcal{P}(\texttt{foldp(}p\texttt{)}) &= \texttt{prfoldp(}\mathcal{P}(p)\texttt{)} \\
\mathcal{P}(\texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i\in L})) &= \texttt{prtplp}[L](\{i \hookrightarrow \mathcal{P}(p_i)\}_{i\in L}) \\
\mathcal{P}(\texttt{injp}[\ell](p)) &= \texttt{prinjp}[\ell](\mathcal{P}(p))
\end{aligned}$$

Proto-Expression Encoding and Decoding

The type abbreviated `PrExpr` classifies encodings of *proto-expressions*. The mapping from proto-expressions to values of type `PrExpr` is defined by the *proto-expression encoding judgement*, $\grave{e} \downarrow_{\mathsf{PrExpr}} e$. An inverse mapping is defined by the *proto-expression decoding judgement*, $e \uparrow_{\mathsf{PrExpr}} \grave{e}$.

Judgement Form	Description
$\grave{e} \downarrow_{\mathsf{PrExpr}} e$	$\grave{e}$ has encoding e
$e \uparrow_{\mathsf{PrExpr}} \grave{e}$	e has decoding $\grave{e}$

Rather than picking a particular definition of `PrExpr` and defining the judgements above inductively against it, we only state the following condition, which establishes an isomorphism between values of type `PrExpr` and proto-expressions.

Condition B.22 (Proto-Expression Isomorphism)**.**

1. *For every $\grave{e}$, we have $\grave{e} \downarrow_{\mathsf{PrExpr}} e_{proto}$ for some e_{proto} such that $\vdash e_{proto} : \mathtt{PrExpr}$ and e_{proto} val.*
2. *If $\vdash e_{proto} : \mathtt{PrExpr}$ and e_{proto} val then $e_{proto} \uparrow_{\mathsf{PrExpr}} \grave{e}$ for some $\grave{e}$.*
3. *If $\grave{e} \downarrow_{\mathsf{PrExpr}} e_{proto}$ then $e_{proto} \uparrow_{\mathsf{PrExpr}} \grave{e}$.*
4. *If $\vdash e_{proto} : \mathtt{PrExpr}$ and e_{proto} val and $e_{proto} \uparrow_{\mathsf{PrExpr}} \grave{e}$ then $\grave{e} \downarrow_{\mathsf{PrExpr}} e_{proto}$.*
5. *If $\grave{e} \downarrow_{\mathsf{PrExpr}} e_{proto}$ and $\grave{e} \downarrow_{\mathsf{PrExpr}} e'_{proto}$ then $e_{proto} = e'_{proto}$.*
6. *If $\vdash e_{proto} : \mathtt{PrExpr}$ and e_{proto} val and $e_{proto} \uparrow_{\mathsf{PrExpr}} \grave{e}$ and $e_{proto} \uparrow_{\mathsf{PrExpr}} \grave{e}'$ then $\grave{e} = \grave{e}'$.*

Proto-Pattern Encoding and Decoding

The type abbreviated `PrPat` classifies encodings of *proto-patterns*. The mapping from proto-patterns to values of type `PrPat` is defined by the *proto-pattern encoding judgement*, $\grave{p} \downarrow_{\mathsf{PrPat}} p$. An inverse mapping is defined by the *proto-expression decoding judgement*, $p \uparrow_{\mathsf{PrPat}} \grave{p}$.

Judgement Form	Description
$\grave{p} \downarrow_{\mathsf{PrPat}} p$	$\grave{p}$ has encoding p
$p \uparrow_{\mathsf{PrPat}} \grave{p}$	p has decoding $\grave{p}$

Again, rather than picking a particular definition of `PrPat` and defining the judgements above inductively against it, we only state the following condition, which establishes an isomorphism between values of type `PrPat` and proto-patterns.

Condition B.23 (Proto-Pattern Isomorphism)**.**

1. *For every $\grave{p}$, we have $\grave{p} \downarrow_{\mathsf{PrPat}} e_{proto}$ for some e_{proto} such that $\vdash e_{proto} : \mathtt{PrPat}$ and e_{proto} val.*
2. *If $\vdash e_{proto} : \mathtt{PrPat}$ and e_{proto} val then $e_{proto} \uparrow_{\mathsf{PrPat}} \grave{p}$ for some $\grave{p}$.*
3. *If $\grave{p} \downarrow_{\mathsf{PrPat}} e_{proto}$ then $e_{proto} \uparrow_{\mathsf{PrPat}} \grave{p}$.*
4. *If $\vdash e_{proto} : \mathtt{PrPat}$ and e_{proto} val and $e_{proto} \uparrow_{\mathsf{PrPat}} \grave{p}$ then $\grave{p} \downarrow_{\mathsf{PrPat}} e_{proto}$.*
5. *If $\grave{p} \downarrow_{\mathsf{PrPat}} e_{proto}$ and $\grave{p} \downarrow_{\mathsf{PrPat}} e'_{proto}$ then $e_{proto} = e'_{proto}$.*

6. *If* $\vdash e_{proto} : \texttt{PrPat}$ *and* e_{proto} val *and* $e_{proto} \uparrow_{\mathsf{PrPat}} \grave{p}$ *and* $e_{proto} \uparrow_{\mathsf{PrPat}} \grave{p}'$ *then* $\grave{p} = \grave{p}'$.

Splice Summaries

The *splice summary* of a proto-expression, $\mathsf{summary}(\grave{e})$, or proto-pattern, $\mathsf{summary}(\grave{p})$, is the finite set of references to spliced types, expressions and patterns that it mentions.

Segmentations

A *segment set*, ψ, is a finite set of pairs of natural numbers indicating the locations of spliced terms. The *segmentation* of a proto-expression, $\mathsf{seg}(\grave{e})$, or proto-pattern, $\mathsf{seg}(\grave{p})$, is the segment set implied by its splice summary.

The predicate ψ segments b checks that each segment in ψ, has non-negative length and is within bounds of b, and that the segments in ψ do not overlap.

B.3.2 Proto-Type Validation

Type splicing scenes, $\mathbb{T}$, are of the form $\hat{\Delta}; b$.

$\boxed{\Delta \vdash^{\mathbb{T}} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type}}$ $\grave{\tau}$ has well-formed expansion τ

$$\frac{}{\Delta, t\ \mathsf{type} \vdash^{\mathbb{T}} t \rightsquigarrow t\ \mathsf{type}} \tag{B.9a}$$

$$\frac{\Delta \vdash^{\mathbb{T}} \grave{\tau}_1 \rightsquigarrow \tau_1\ \mathsf{type} \qquad \Delta \vdash^{\mathbb{T}} \grave{\tau}_2 \rightsquigarrow \tau_2\ \mathsf{type}}{\Delta \vdash^{\mathbb{T}} \texttt{prparr}(\grave{\tau}_1; \grave{\tau}_2) \rightsquigarrow \texttt{parr}(\tau_1; \tau_2)\ \mathsf{type}} \tag{B.9b}$$

$$\frac{\Delta, t\ \mathsf{type} \vdash^{\mathbb{T}} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type}}{\Delta \vdash^{\mathbb{T}} \texttt{prall}(t.\grave{\tau}) \rightsquigarrow \texttt{all}(t.\tau)\ \mathsf{type}} \tag{B.9c}$$

$$\frac{\Delta, t\ \mathsf{type} \vdash^{\mathbb{T}} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type}}{\Delta \vdash^{\mathbb{T}} \texttt{prrec}(t.\grave{\tau}) \rightsquigarrow \texttt{rec}(t.\tau)\ \mathsf{type}} \tag{B.9d}$$

$$\frac{\{\Delta \vdash^{\mathbb{T}} \grave{\tau}_i \rightsquigarrow \tau_i\ \mathsf{type}\}_{i \in L}}{\Delta \vdash^{\mathbb{T}} \texttt{prprod}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L}) \rightsquigarrow \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})\ \mathsf{type}} \tag{B.9e}$$

$$\frac{\{\Delta \vdash^{\mathbb{T}} \grave{\tau}_i \rightsquigarrow \tau_i\ \mathsf{type}\}_{i \in L}}{\Delta \vdash^{\mathbb{T}} \texttt{prsum}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L}) \rightsquigarrow \texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L})\ \mathsf{type}} \tag{B.9f}$$

$$\frac{\mathsf{parseUTyp}(\mathsf{subseq}(b; m; n)) = \hat{\tau} \qquad \langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \Delta \cap \Delta_{\mathrm{app}} = \emptyset}{\Delta \vdash^{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle; b} \texttt{splicedt}[m; n] \rightsquigarrow \tau\ \mathsf{type}} \tag{B.9g}$$

B.3.3 Proto-Expression Validation

Expression splicing scenes, $\mathbb{E}$, are of the form $\hat{\Delta}; \hat{\Gamma}; \hat{\Psi}; \hat{\Phi}; b$. We write $\mathsf{ts}(\mathbb{E})$ for the type splicing scene constructed by dropping unnecessary contexts from $\mathbb{E}$:

$$\mathsf{ts}(\hat{\Delta}; \hat{\Gamma}; \hat{\Psi}; \hat{\Phi}; b) = \hat{\Delta}; b$$

$\boxed{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau}$ $\grave{e}$ has expansion e of type τ

$$\frac{}{\Delta\ \Gamma, x : \tau \vdash^{\mathbb{E}} x \rightsquigarrow x : \tau} \tag{B.10a}$$

$$\frac{\Delta \vdash^{\mathsf{ts}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prasc\{}\grave{\tau}\mathtt{\}(}\grave{e}\mathtt{)} \rightsquigarrow e : \tau} \tag{B.10b}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e}_1 \rightsquigarrow e_1 : \tau_1 \qquad \Delta\ \Gamma, x : \tau_1 \vdash^{\grave{e}_2} e_2 \rightsquigarrow \tau_2 :}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prletval(}\grave{e}_1\mathtt{;}x.\grave{e}_2\mathtt{)} \rightsquigarrow \mathtt{ap(lam\{}\tau_1\mathtt{\}(}x.e_2\mathtt{);}e_1\mathtt{)} : \tau_2} \tag{B.10c}$$

$$\frac{\Delta \vdash^{\mathsf{ts}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \Delta\ \Gamma, x : \tau \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prlam\{}\grave{\tau}\mathtt{\}(}x.\grave{e}\mathtt{)} \rightsquigarrow \mathtt{lam\{}\tau\mathtt{\}(}x.e\mathtt{)} : \mathtt{parr(}\tau\mathtt{;}\tau'\mathtt{)}} \tag{B.10d}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e}_1 \rightsquigarrow e_1 : \mathtt{parr(}\tau\mathtt{;}\tau'\mathtt{)} \qquad \Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e}_2 \rightsquigarrow e_2 : \tau}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prap(}\grave{e}_1\mathtt{;}\grave{e}_2\mathtt{)} \rightsquigarrow \mathtt{ap(}e_1\mathtt{;}e_2\mathtt{)} : \tau'} \tag{B.10e}$$

$$\frac{\Delta, t\ \mathsf{type}\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prtlam(}t.\grave{e}\mathtt{)} \rightsquigarrow \mathtt{tlam(}t.e\mathtt{)} : \mathtt{all(}t.\tau\mathtt{)}} \tag{B.10f}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{all(}t.\tau\mathtt{)} \qquad \Delta \vdash^{\mathsf{ts}(\mathbb{E})} \grave{\tau}' \rightsquigarrow \tau'\ \mathsf{type}}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prtap\{}\grave{\tau}'\mathtt{\}(}\grave{e}\mathtt{)} \rightsquigarrow \mathtt{tap\{}\tau'\mathtt{\}(}e\mathtt{)} : [\tau'/t]\tau} \tag{B.10g}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : [\mathtt{rec(}t.\tau\mathtt{)}/t]\tau}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prfold(}\grave{e}\mathtt{)} \rightsquigarrow \mathtt{fold(}e\mathtt{)} : \mathtt{rec(}t.\tau\mathtt{)}} \tag{B.10h}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{rec(}t.\tau\mathtt{)}}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prunfold(}\grave{e}\mathtt{)} \rightsquigarrow \mathtt{unfold(}e\mathtt{)} : [\mathtt{rec(}t.\tau\mathtt{)}/t]\tau} \tag{B.10i}$$

$$\frac{\begin{array}{c}\tau = \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \\ \{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e}_i \rightsquigarrow e_i : \tau_i\}_{i \in L}\end{array}}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prtpl\{}L\mathtt{\}(}\{i \hookrightarrow \grave{e}_i\}_{i \in L}\mathtt{)} \rightsquigarrow \mathtt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L}) : \tau} \tag{B.10j}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{prod}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau)}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prprj}[\ell](\grave{e}) \rightsquigarrow \mathtt{prj}[\ell](e) : \tau} \tag{B.10k}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prinj}[\ell](\grave{e}) \rightsquigarrow \mathtt{inj}[\ell](e) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')} \tag{B.10l}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \qquad \{\Delta\ \Gamma, x_i : \tau_i \vdash^{\mathbb{E}} \grave{e}_i \rightsquigarrow e_i : \tau\}_{i \in L}}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prcase}[L](\grave{e}; \{i \hookrightarrow x_i.\grave{e}_i\}_{i \in L}) \rightsquigarrow \mathtt{case}[L](e; \{i \hookrightarrow x_i.e_i\}_{i \in L}) : \tau} \tag{B.10m}$$

$$\frac{\begin{array}{c}\varnothing \vdash^{\mathsf{ts}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \mathbb{E} = \langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle;\ \langle \mathcal{G}; \Gamma_{\mathrm{app}} \rangle;\ \hat{\Psi}; \hat{\Phi};\ b \\ \mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{e} \qquad \langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle\ \langle \mathcal{G}; \Gamma_{\mathrm{app}} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{e} \rightsquigarrow e : \tau \\ \Delta \cap \Delta_{\mathrm{app}} = \varnothing \qquad \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma_{\mathrm{app}}) = \varnothing\end{array}}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{splicede}[m; n; \grave{\tau}] \rightsquigarrow e : \tau} \tag{B.10n}$$

$$\frac{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau \qquad \{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{r}_i \rightsquigarrow r_i : \tau \Mapsto \tau'\}_{1 \leq i \leq n}}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prmatch}[n](\grave{e}; \{\grave{r}_i\}_{1 \leq i \leq n}) \rightsquigarrow \mathtt{match}[n](e; \{r_i\}_{1 \leq i \leq n}) : \tau'} \tag{B.10o}$$

$\boxed{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{r} \rightsquigarrow r : \tau \Mapsto \tau'}$ $\grave{r}$ has expansion r taking values of type τ to values of type τ'

$$\frac{p : \tau \dashv \Gamma' \qquad \Delta\ \Gamma \cup \Gamma' \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prrule}(p.\grave{e}) \rightsquigarrow \mathtt{rule}(p.e) : \tau \Mapsto \tau'} \tag{B.11}$$

B.3.4 Proto-Pattern Validation

Pattern splicing scenes, $\mathbb{P}$, are of the form $\hat{\Delta}; \hat{\Phi}; b$.

$\boxed{\grave{p} \rightsquigarrow p : \tau \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}}$ $\grave{p}$ has expansion p matching against τ generating hypotheses $\hat{\Gamma}$

$$\frac{}{\mathtt{prwildp} \rightsquigarrow \mathtt{wildp} : \tau \dashv\!\!\mid^{\mathbb{P}} \langle \varnothing; \varnothing \rangle} \tag{B.12a}$$

$$\frac{\grave{p} \rightsquigarrow p : [\mathtt{rec}(t.\tau)/t]\tau \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}}{\mathtt{prfoldp}(\grave{p}) \rightsquigarrow \mathtt{foldp}(p) : \mathtt{rec}(t.\tau) \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}} \tag{B.12b}$$

$$\frac{\begin{array}{c}\tau = \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \\ \{\grave{p}_i \rightsquigarrow p_i : \tau_i \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}_i\}_{i \in L}\end{array}}{\mathtt{prtplp}[L](\{i \hookrightarrow \grave{p}_i\}_{i \in L}) \rightsquigarrow \mathtt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) : \tau \dashv\!\!\mid^{\mathbb{P}} \uplus_{i \in L} \hat{\Gamma}_i} \tag{B.12c}$$

$$\frac{\grave{p} \rightsquigarrow p : \tau \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}}{\mathtt{prinjp}[\ell](\grave{p}) \rightsquigarrow \mathtt{injp}[\ell](p) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau) \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}} \tag{B.12d}$$

$$\frac{\varnothing \vdash^{\hat{\Delta}; b} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type} \qquad \mathsf{parseUPat}(\mathsf{subseq}(b; m; n)) = \hat{p} \qquad \hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\!\mid \hat{\Gamma}}{\mathtt{splicedp}[m; n; \grave{\tau}] \rightsquigarrow p : \tau \dashv\!\!\mid^{\hat{\Delta}; \hat{\Phi}; b} \hat{\Gamma}} \tag{B.12e}$$

B.4 Metatheory

B.4.1 Type Expansion

Lemma B.24 (Type Expansion). *If* $\langle \mathcal{D}; \Delta \rangle \vdash \hat{\tau} \rightsquigarrow \tau$ type *then* $\Delta \vdash \tau$ type.

Proof. By rule induction over Rules (B.5). In each case, we apply the IH to or over each premise, then apply the corresponding type formation rule in Rules (B.1). □

Lemma B.25 (Proto-Type Validation). *If* $\Delta \vdash^{\langle \mathcal{D}; \Delta_{app} \rangle; b} \grave{\tau} \rightsquigarrow \tau$ type *and* $\Delta \cap \Delta_{app} = \emptyset$ *then* $\Delta \cup \Delta_{app} \vdash \tau$ type.

Proof. By rule induction over Rules (B.9).

Case (B.9a).

(1)	$\Delta = \Delta', t$ type	by assumption
(2)	$\grave{\tau} = t$	by assumption
(3)	$\tau = t$	by assumption
(4)	Δ', t type $\vdash t$ type	by Rule (B.1a)
(5)	Δ', t type $\cup \Delta_{\text{app}} \vdash t$ type	by Lemma B.2 over Δ_{app} to (4)

Case (B.9b).

(1)	$\grave{\tau} = \texttt{prparr}(\grave{\tau}_1; \grave{\tau}_2)$	by assumption
(2)	$\tau = \texttt{parr}(\tau_1; \tau_2)$	by assumption
(3)	$\Delta \vdash^{\langle \mathcal{D}; \Delta_{\text{app}} \rangle; b} \grave{\tau}_1 \rightsquigarrow \tau_1$ type	by assumption
(4)	$\Delta \vdash^{\langle \mathcal{D}; \Delta_{\text{app}} \rangle; b} \grave{\tau}_2 \rightsquigarrow \tau_2$ type	by assumption
(5)	$\Delta \cup \Delta_{\text{app}} \vdash \tau_1$ type	by IH on (3)
(6)	$\Delta \cup \Delta_{\text{app}} \vdash \tau_2$ type	by IH on (4)
(7)	$\Delta \cup \Delta_{\text{app}} \vdash \texttt{parr}(\tau_1; \tau_2)$ type	by Rule (B.1b) on (5) and (6)

Case (B.9c).

(1)	$\grave{\tau} = \texttt{prall}(t.\grave{\tau}')$	by assumption
(2)	$\tau = \texttt{all}(t.\tau')$	by assumption
(3)	Δ, t type $\vdash^{\langle \mathcal{D}; \Delta_{\text{app}} \rangle; b} \grave{\tau}' \rightsquigarrow \tau'$ type	by assumption
(4)	Δ, t type $\cup \Delta_{\text{app}} \vdash \tau'$ type	by IH on (3)

(5) $\Delta \cup \Delta_{\text{app}}, t \text{ type} \vdash \tau' \text{ type}$ — by exchange over Δ_{app} on (4)

(6) $\Delta \cup \Delta_{\text{app}} \vdash \texttt{all}(t.\tau') \text{ type}$ — by Rule (B.1c) on (5)

Case (B.9d).

(1) $\grave{\tau} = \texttt{prrec}(t.\grave{\tau}')$ — by assumption

(2) $\tau = \texttt{rec}(t.\tau')$ — by assumption

(3) $\Delta, t \text{ type} \vdash^{\Delta_{\text{app}};b} \grave{\tau}' \rightsquigarrow \tau' \text{ type}$ — by assumption

(4) $\Delta, t \text{ type} \cup \Delta_{\text{app}} \vdash \tau' \text{ type}$ — by IH on (3)

(5) $\Delta \cup \Delta_{\text{app}}, t \text{ type} \vdash \tau' \text{ type}$ — by exchange over Δ_{app} on (4)

(6) $\Delta \cup \Delta_{\text{app}} \vdash \texttt{rec}(t.\tau') \text{ type}$ — by Rule (B.1d) on (5)

Case (B.9e).

(1) $\grave{\tau} = \texttt{prprod}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L})$ — by assumption

(2) $\tau = \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ — by assumption

(3) $\{\Delta \vdash^{\Delta_{\text{app}};b} \grave{\tau}_i \rightsquigarrow \tau_i \text{ type}\}_{i \in L}$ — by assumption

(4) $\{\Delta \cup \Delta_{\text{app}} \vdash \tau_i \text{ type}\}_{i \in L}$ — by IH over (3)

(5) $\Delta \cup \Delta_{\text{app}} \vdash \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \text{ type}$ — by Rule (B.1e) on (4)

Case (B.9f).

(1) $\grave{\tau} = \texttt{prsum}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L})$ — by assumption

(2) $\tau = \texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ — by assumption

(3) $\{\Delta \vdash^{\Delta_{\text{app}};b} \grave{\tau}_i \rightsquigarrow \tau_i \text{ type}\}_{i \in L}$ — by assumption

(4) $\{\Delta \cup \Delta_{\text{app}} \vdash \tau_i \text{ type}\}_{i \in L}$ — by IH over (3)

(5) $\Delta \cup \Delta_{\text{app}} \vdash \texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \text{ type}$ — by Rule (B.1f) on (4)

Case (B.9g).

(1) $\grave{\tau} = \texttt{splicedt}[m;n]$ — by assumption

(2) $\mathsf{parseUTyp}(\mathsf{subseq}(b; m; n)) = \hat{\tau}$ — by assumption

(3) $\langle \mathcal{D}; \Delta_{\mathsf{app}} \rangle \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type}$ — by assumption

(4) $\Delta \cap \Delta_{\mathsf{app}} = \emptyset$ — by assumption

(5) $\Delta_{\mathsf{app}} \vdash \tau\ \mathsf{type}$ — by Lemma B.24 on (3)

(6) $\Delta \cup \Delta_{\mathsf{app}} \vdash \tau\ \mathsf{type}$ — by Lemma B.2 over Δ on (5) and exchange over Δ

□

B.4.2 Typed Pattern Expansion

Theorem B.26 (Typed Pattern Expansion)**.**

1. *If* $\langle \mathcal{D}; \Delta \rangle \vdash_{\langle \mathcal{A}; \Phi \rangle} \hat{p} \rightsquigarrow p : \tau \dashv\!\mid \langle \mathcal{G}; \Gamma \rangle$ *then* $\Delta \vdash p : \tau \dashv\!\mid \Gamma$.

2. *If* $\hat{p} \rightsquigarrow p : \tau \dashv\!\mid^{\langle \mathcal{D}; \Delta \rangle; \langle \mathcal{A}; \Phi \rangle; b} \langle \mathcal{G}; \Gamma \rangle$ *then* $\Delta \vdash p : \tau \dashv\!\mid \Gamma$.

Proof. By mutual rule induction over Rules (B.8) and Rules (B.12).

1. We induct on the premise. In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta \rangle$ and $\hat{\Gamma} = \langle \mathcal{G}; \Gamma \rangle$ and $\hat{\Phi} = \langle \mathcal{A}; \Phi \rangle$.

   Case (B.8a)**.**

   (1) $\hat{p} = \hat{x}$ — by assumption

   (2) $p = x$ — by assumption

   (3) $\Gamma = x : \tau$ — by assumption

   (4) $\Delta \vdash x : \tau \dashv\!\mid x : \tau$ — by Rule (B.4a)

   Case (B.8b)**.**

   (1) $p = \mathtt{wildp}$ — by assumption

   (2) $\Gamma = \emptyset$ — by assumption

   (3) $\Delta \vdash \mathtt{wildp} : \tau \dashv\!\mid \emptyset$ — by Rule (B.4b)

   Case (B.8c)**.**

   (1) $\hat{p} = \mathtt{fold}(\hat{p}')$ — by assumption

   (2) $p = \mathtt{foldp}(p')$ — by assumption

   (3) $\tau = \mathtt{rec}(t.\tau')$ — by assumption

   (4) $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p}' \rightsquigarrow p' : [\mathtt{rec}(t.\tau')/t]\tau' \dashv\!\mid \hat{\Gamma}$ — by assumption

   (5) $\Delta \vdash p' : [\mathtt{rec}(t.\tau')/t]\tau' \dashv\!\mid \Gamma$ — by IH, part 1 on (4)

(6) $\Delta \vdash \texttt{foldp}(p') : \texttt{rec}(t.\tau') \dashv\!\mid \Gamma$ by Rule (B.4c) on (5)

Case (B.8d).

(1) $\hat{p} = \langle \{i \hookrightarrow \hat{p}_i\}_{i \in L} \rangle$ by assumption
(2) $p = \texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L})$ by assumption
(3) $\tau = \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ by assumption
(4) $\{\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p}_i \rightsquigarrow p_i : \tau_i \dashv\!\mid \langle \mathcal{G}_i; \Gamma_i \rangle\}_{i \in L}$ by assumption
(5) $\Gamma = \cup_{i \in L} \Gamma_i$ by assumption
(6) $\{\Delta \vdash p_i : \tau_i \dashv\!\mid \Gamma_i\}_{i \in L}$ by IH, part 1 over (4)
(7) $\Delta \vdash \texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) : \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \dashv\!\mid \cup_{i \in L} \Gamma_i$ by Rule (B.4d) on (6)

Case (B.8e).

(1) $\hat{p} = \texttt{inj}[\ell](\hat{p}')$ by assumption
(2) $p = \texttt{injp}[\ell](p')$ by assumption
(3) $\tau = \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')$ by assumption
(4) $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p}' \rightsquigarrow p' : \tau' \dashv\!\mid \hat{\Gamma}$ by assumption
(5) $\Delta \vdash p' : \tau' \dashv\!\mid \Gamma$ by IH, part 1 on (4)
(6) $\Delta \vdash \texttt{injp}[\ell](p') : \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau') \dashv\!\mid \Gamma$ by Rule (B.4e) on (5)

Case (B.8f).

(1) $\hat{p} = \hat{a}\ \text{`}b\text{`}$ by assumption
(2) $\mathcal{A} = \mathcal{A}', \hat{a} \rightsquigarrow a$ by assumption
(3) $\Phi = \Phi', a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}})$ by assumption
(4) $b \downarrow_{\mathsf{Body}} e_{\text{body}}$ by assumption
(5) $e_{\text{parse}}(e_{\text{body}}) \Downarrow \texttt{inj}[\texttt{SuccessP}](e_{\text{proto}})$ by assumption
(6) $e_{\text{proto}} \uparrow_{\mathsf{PrPat}} \grave{p}$ by assumption
(7) $\grave{p} \rightsquigarrow p : \tau \dashv\!\mid^{\hat{\Delta}; \langle \mathcal{A}; \Phi \rangle; b} \langle \mathcal{G}; \Gamma \rangle$ by assumption
(8) $\Delta \vdash p : \tau \dashv\!\mid \Gamma$ by IH, part 2 on (7)

2. We induct on the premise. In the following, let $\hat{\Gamma} = \langle \mathcal{G}; \Gamma \rangle$ and $\hat{\Delta} = \langle \mathcal{D}; \Delta \rangle$ and $\hat{\Phi} = \langle \mathcal{A}; \Phi \rangle$.

Case (B.12a).

(1) $p = \texttt{wildp}$ by assumption
(2) $\Gamma = \emptyset$ by assumption
(3) $\Delta \vdash \texttt{wildp} : \tau \dashv\!\mid \emptyset$ by Rule (B.4b)

Case (B.12b).

(1) $\hat{p} = \texttt{prfoldp}(\hat{p}')$ by assumption
(2) $p = \texttt{foldp}(p')$ by assumption
(3) $\tau = \texttt{rec}(t.\tau')$ by assumption
(4) $\hat{p}' \rightsquigarrow p' : [\texttt{rec}(t.\tau')/t]\tau' \dashv\!\vert^{\hat{\Delta};\hat{\Phi};b} \hat{\Gamma}$ by assumption
(5) $\Delta \vdash p' : [\texttt{rec}(t.\tau')/t]\tau' \dashv\!\vert \Gamma$ by IH, part 2 on (4)
(6) $\Delta \vdash \texttt{foldp}(p') : \texttt{rec}(t.\tau') \dashv\!\vert \Gamma$ by Rule (B.4c) on (5)

Case (B.12c).

(1) $\hat{p} = \texttt{prtplp}[L](\{i \hookrightarrow \hat{p}_i\}_{i \in L})$ by assumption
(2) $p = \texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L})$ by assumption
(3) $\tau = \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})$ by assumption
(4) $\{\hat{p}_i \rightsquigarrow p_i : \tau_i \dashv\!\vert^{\hat{\Delta};\hat{\Phi};b} \langle \mathcal{G}_i; \Gamma_i \rangle\}_{i \in L}$ by assumption
(5) $\Gamma = \cup_{i \in L} \Gamma_i$ by assumption
(6) $\{\Delta \vdash p_i : \tau_i \dashv\!\vert \Gamma_i\}_{i \in L}$ by IH, part 2 over (4)
(7) $\Delta \vdash \texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) : \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \dashv\!\vert \cup_{i \in L} \Gamma_i$ by Rule (B.4d) on (6)

Case (B.12d).

(1) $\hat{p} = \texttt{prinjp}[\ell](\hat{p}')$ by assumption
(2) $p = \texttt{injp}[\ell](p')$ by assumption
(3) $\tau = \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')$ by assumption
(4) $\hat{p}' \rightsquigarrow p' : \tau' \dashv\!\vert^{\hat{\Delta};\hat{\Phi};b} \hat{\Gamma}$ by assumption
(5) $\Delta \vdash p' : \tau' \dashv\!\vert \Gamma$ by IH, part 2 on (4)
(6) $\Delta \vdash \texttt{injp}[\ell](p') : \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau') \dashv\!\vert \Gamma$ by Rule (B.4e) on (5)

Case (B.12e).

(1) $\hat{p} = \texttt{splicedp}[m; n; \hat{\tau}]$ by assumption
(2) $\emptyset \vdash^{\hat{\Delta};b} \hat{\tau} \rightsquigarrow \tau$ type by assumption
(3) $\mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{p}$ by assumption
(4) $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\vert \hat{\Gamma}$ by assumption
(5) $\Delta \vdash p : \tau \dashv\!\vert \Gamma$ by IH, part 1 on (4)

The mutual induction can be shown to be well-founded by showing that the following numeric metric on the judgements that we induct on is decreasing:

$$\|\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!| \, \hat{\Gamma}\| = \|\hat{p}\|$$
$$\|\grave{p} \rightsquigarrow p : \tau \dashv\!|^{\hat{\Delta};\hat{\Phi};b} \, \hat{\Gamma}\| = \|b\|$$

where $\|b\|$ is the length of b and $\|\hat{p}\|$ is the sum of the lengths of the literal bodies in $\hat{p}$, as defined in Sec. B.2.1.

The only case in the proof of part 1 that invokes part 2 is Case (B.8f). There, we have that the metric remains stable:

$$\begin{aligned}&\|\hat{\Delta} \vdash_{\hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow p : \tau \dashv\!| \, \hat{\Gamma}\| \\ =&\|\grave{p} \rightsquigarrow p : \tau \dashv\!|^{\hat{\Delta};\hat{\Phi};b} \, \hat{\Gamma}\| \\ =&\|b\|\end{aligned}$$

The only case in the proof of part 2 that invokes part 1 is Case (B.12e). There, we have that $\mathsf{parseUPat}(\mathsf{subseq}(b;m;n)) = \hat{p}$ and the IH is applied to the judgement $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!| \, \hat{\Gamma}$. Because the metric is stable when passing from part 1 to part 2, we must have that it is strictly decreasing in the other direction:

$$\|\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!| \, \hat{\Gamma}\| < \|\mathtt{splicedp}[m;n;\grave{\tau}] \rightsquigarrow p : \tau \dashv\!|^{\hat{\Delta};\hat{\Phi};b} \, \hat{\Gamma}\|$$

i.e. by the definitions above,

$$\|\hat{p}\| < \|b\|$$

This is established by appeal to Condition B.17, which states that subsequences of b are no longer than b, and the Condition B.13, which states that an unexpanded pattern constructed by parsing a textual sequence b is strictly smaller, as measured by the metric defined above, than the length of b, because some characters must necessarily be used to apply the pattern TSM and delimit each literal body. Combining Conditions B.17 and B.13, we have that $\|\hat{p}\| < \|b\|$ as needed. □

B.4.3 Typed Expression Expansion

Theorem B.27 (Typed Expansion (Strong))**.**

1. (a) *If* $\langle \mathcal{D}; \Delta \rangle \ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau$ *then* $\Delta \ \Gamma \vdash e : \tau$.

   (b) *If* $\langle \mathcal{D}; \Delta \rangle \ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r} \rightsquigarrow r : \tau \Mapsto \tau'$ *then* $\Delta \ \Gamma \vdash r : \tau \Mapsto \tau'$.

2. (a) *If* $\Delta \ \Gamma \vdash^{\langle \mathcal{D};\Delta_{app} \rangle;\langle \mathcal{G};\Gamma_{app} \rangle;\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau$ *and* $\Delta \cap \Delta_{app} = \emptyset$ *and* $dom(\Gamma) \cap dom(\Gamma_{app}) = \emptyset$ *then* $\Delta \cup \Delta_{app} \ \Gamma \cup \Gamma_{app} \vdash e : \tau$.

   (b) *If* $\Delta \ \Gamma \vdash^{\langle \mathcal{D};\Delta_{app} \rangle;\langle \mathcal{G};\Gamma_{app} \rangle;\hat{\Psi};\hat{\Phi};b} \grave{r} \rightsquigarrow r : \tau \Mapsto \tau'$ *and* $\Delta \cap \Delta_{app} = \emptyset$ *and* $dom(\Gamma) \cap dom(\Gamma_{app}) = \emptyset$ *then* $\Delta \cup \Delta_{app} \ \Gamma \cup \Gamma_{app} \vdash r : \tau \Mapsto \tau'$.

Proof. By mutual rule induction over Rules (B.6), Rule (B.7), Rules (B.10) and Rule (B.11).

1. In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta \rangle$ and $\hat{\Gamma} = \langle \mathcal{G}; \Gamma \rangle$.

   (a) **Case** (B.6a).

(1)	$\hat{e} = \hat{x}$	by assumption
(2)	$e = x$	by assumption
(3)	$\Gamma = \Gamma', x : \tau$	by assumption
(4)	$\Delta\ \Gamma', x : \tau \vdash x : \tau$	by Rule (B.2a)

   Case (B.6b).

(1)	$\hat{e} = \hat{e}' : \hat{\tau}$	by assumption
(2)	$\hat{\Delta} \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type}$	by assumption
(3)	$\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}' \rightsquigarrow e : \tau$	by assumption
(4)	$\Delta\ \Gamma \vdash e : \tau$	by IH, part 1(a) on (3)

   Case (B.6c).

(1)	$\hat{e} = \mathtt{let\ val}\ \hat{x} = \hat{e}_1\ \mathtt{in}\ \hat{e}_2$	by assumption
(2)	$e = \mathtt{ap(lam\{}\tau_1\mathtt{\}(}x.e_2\mathtt{);}e_1\mathtt{)}$	by assumption
(3)	$\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1 \rightsquigarrow e_1 : \tau_1$	by assumption
(4)	$\hat{\Delta}\ \hat{\Gamma}, \hat{x} \rightsquigarrow x : \tau_1 \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_2 \rightsquigarrow e_2 : \tau$	by assumption
(5)	$\Delta\ \Gamma \vdash e_1 : \tau_1$	by IH, part 1(a) on (3)
(6)	$\Delta\ \Gamma, x : \tau \vdash e_2 : \tau$	by IH, part 1(a) on (4)
(7)	$\Delta\ \Gamma \vdash \mathtt{lam\{}\tau_1\mathtt{\}(}x.e_2\mathtt{)} : \mathtt{parr(}\tau_1\mathtt{;}\tau\mathtt{)}$	by Rule (B.2b) on (6)
(8)	$\Delta\ \Gamma \vdash \mathtt{ap(lam\{}\tau_1\mathtt{\}(}x.e_2\mathtt{);}e_1\mathtt{)} : \tau$	by Rule (B.2c) on (7) and (5)

   Case (B.6d).

(1)	$\hat{e} = \lambda \hat{x}{:}\hat{\tau}_1.\hat{e}'$	by assumption
(2)	$e = \mathtt{lam\{}\tau_1\mathtt{\}(}x.e'\mathtt{)}$	by assumption
(3)	$\tau = \mathtt{parr(}\tau_1\mathtt{;}\tau_2\mathtt{)}$	by assumption
(4)	$\hat{\Delta} \vdash \hat{\tau}_1 \rightsquigarrow \tau_1\ \mathsf{type}$	by assumption
(5)	$\hat{\Delta}\ \hat{\Gamma}, \hat{x} \rightsquigarrow x : \tau_1 \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}' \rightsquigarrow e' : \tau_2$	by assumption
(6)	$\Delta \vdash \tau_1\ \mathsf{type}$	by Lemma B.24 on (4)
(7)	$\Delta\ \Gamma, x : \tau_1 \vdash e' : \tau_2$	by IH, part 1(a) on (5)
(8)	$\Delta\ \Gamma \vdash \mathtt{lam\{}\tau_1\mathtt{\}(}x.e'\mathtt{)} : \mathtt{parr(}\tau_1\mathtt{;}\tau_2\mathtt{)}$	by Rule (B.2b) on (6) and (7)

   Case (B.6e).

(1)	$\hat{e} = \hat{e}_1(\hat{e}_2)$	by assumption

(2) $e = \texttt{ap}(e_1; e_2)$ — by assumption
(3) $\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1 \rightsquigarrow e_1 : \texttt{parr}(\tau_2; \tau)$ — by assumption
(4) $\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_2 \rightsquigarrow e_2 : \tau_2$ — by assumption
(5) $\Delta\ \Gamma \vdash e_1 : \texttt{parr}(\tau_2; \tau)$ — by IH, part 1(a) on (3)
(6) $\Delta\ \Gamma \vdash e_2 : \tau_2$ — by IH, part 1(a) on (4)
(7) $\Delta\ \Gamma \vdash \texttt{ap}(e_1; e_2) : \tau$ — by Rule (B.2c) on (5) and (6)

Case (B.6f) **through** (B.6m). These cases follow analagously, i.e. we apply Lemma B.24 to or over the type expansion premises and the IH part 1(a) to or over the typed expression expansion premises and then apply the corresponding typing rule in Rules (B.2d) through (B.2k).

Case (B.6n).

(1) $\hat{e} = \texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\tau}'\ \texttt{by}\ \texttt{static}\ e_{\text{parse}}\ \texttt{in}\ \hat{e}'$ — by assumption
(2) $\hat{\Delta} \vdash \hat{\tau}' \rightsquigarrow \tau'\ \mathsf{type}$ — by assumption
(3) $\emptyset\ \emptyset \vdash e_{\text{parse}} : \texttt{parr}(\texttt{Body}; \texttt{ParseResultSE})$ — by assumption
(4) $\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi},\hat{a}\rightsquigarrow a \hookrightarrow \texttt{setsm}(\tau'; e_{\text{parse}});\hat{\Phi}} \hat{e}' \rightsquigarrow e : \tau$ — by assumption
(5) $\Delta \vdash \tau'\ \mathsf{type}$ — by Lemma B.24 to (2)
(6) $\Delta\ \Gamma \vdash e : \tau$ — by IH, part 1(a) on (4)

Case (B.6o).

(1) $\hat{e} = \hat{a}\ \text{`}b\text{`}$ — by assumption
(2) $\mathcal{A} = \mathcal{A}', \hat{a} \rightsquigarrow a$ — by assumption
(3) $\Psi = \Psi', a \hookrightarrow \texttt{setsm}(\tau; e_{\text{parse}})$ — by assumption
(4) $b \downarrow_{\mathsf{Body}} e_{\text{body}}$ — by assumption
(5) $e_{\text{parse}}(e_{\text{body}}) \Downarrow \texttt{inj}[\texttt{SuccessE}](e_{\text{proto}})$ — by assumption
(6) $e_{\text{proto}} \uparrow_{\mathsf{PrExpr}} \grave{e}$ — by assumption
(7) $\emptyset\ \emptyset \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau$ — by assumption
(8) $\emptyset \cap \Delta = \emptyset$ — by finite set intersection
(9) $\emptyset \cap \mathrm{dom}(\Gamma) = \emptyset$ — by finite set intersection
(10) $\emptyset \cup \Delta\ \emptyset \cup \Gamma \vdash e : \tau$ — by IH, part 2(a) on (7), (8), and (9)
(11) $\Delta\ \Gamma \vdash e : \tau$ — by finite set and finite function identity over (10)

Case (B.6p).

(1) $\hat{e} = \texttt{match}\ \hat{e}'\ \{\hat{r}_i\}_{1 \leq i \leq n}$ by assumption
(2) $e = \texttt{match}[n](e'; \{r_i\}_{1 \leq i \leq n})$ by assumption
(3) $\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}' \rightsquigarrow e' : \tau'$ by assumption
(4) $\{\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r}_i \rightsquigarrow r_i : \tau' \Mapsto \tau\}_{1 \leq i \leq n}$ by assumption
(5) $\Delta\ \Gamma \vdash e' : \tau'$ by IH, part 1(a) on (3)
(6) $\{\Delta\ \Gamma \vdash r_i : \tau' \Mapsto \tau\}_{1 \leq i \leq n}$ by IH, part 1(b) over (4)
(7) $\Delta\ \Gamma \vdash \texttt{match}[n](e'; \{r_i\}_{1 \leq i \leq n}) : \tau$ by Rule (B.21) on (5) and (6)

Case (B.6q).

(1) $\hat{e} = \texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\tau}'\ \texttt{for patterns by static}\ e_{\text{parse}}\ \texttt{in}\ \hat{e}'$ by assumption
(2) $\hat{\Delta} \vdash \hat{\tau}' \rightsquigarrow \tau'\ \text{type}$ by assumption
(3) $\emptyset\ \emptyset \vdash e_{\text{parse}} : \texttt{parr(Body;ParseResultSE)}$ by assumption
(4) $\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi},\hat{a} \rightsquigarrow a \hookrightarrow \texttt{sptsm}(\tau'; e_{\text{parse}})} \hat{e}' \rightsquigarrow e : \tau$ by assumption
(5) $\Delta \vdash \tau'\ \text{type}$ by Lemma B.24 to (2)
(6) $\Delta\ \Gamma \vdash e : \tau$ by IH, part 1(a) on (4)

(b) **Case** (B.7).

(1) $\hat{r} = \hat{p} \Rightarrow \hat{e}$ by assumption
(2) $r = \texttt{rule}(p.e)$ by assumption
(3) $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\mid \langle \mathcal{A}'; \Gamma \rangle$ by assumption
(4) $\hat{\Delta}\ \langle \mathcal{A} \uplus \mathcal{A}'; \Gamma \cup \Gamma \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'$ by assumption
(5) $\Delta \vdash p : \tau \dashv\!\mid \Gamma$ by Theorem B.26, part 1 on (3)
(6) $\Delta\ \Gamma \cup \Gamma \vdash e : \tau'$ by IH, part 1(a) on (4)
(7) $\Delta\ \Gamma \vdash \texttt{rule}(p.e) : \tau \Mapsto \tau'$ by Rule (B.3) on (5) and (6)

2. In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta_{\text{app}} \rangle$ and $\hat{\Gamma} = \langle \mathcal{G}; \Gamma_{\text{app}} \rangle$.

(a) **Case** (B.10a).

(1) $\grave{e} = x$ by assumption
(2) $e = x$ by assumption
(3) $\Gamma = \Gamma', x : \tau$ by assumption
(4) $\Delta \cup \Delta_{\text{app}}\ \Gamma', x : \tau \vdash x : \tau$ by Rule (B.2a)

(5) $\Delta \cup \Delta_{\text{app}}\ \Gamma', x : \tau \cup \Gamma_{\text{app}} \vdash x : \tau$ — by Lemma B.2 over Γ_{app} to (4)

Case (B.10d).

(1) $\grave{e} = \texttt{prlam\{}\grave{\tau}_1\texttt{\}(}x.\grave{e}'\texttt{)}$ — by assumption
(2) $e = \texttt{lam\{}\tau_1\texttt{\}(}x.e'\texttt{)}$ — by assumption
(3) $\tau = \texttt{parr(}\tau_1\texttt{;}\tau_2\texttt{)}$ — by assumption
(4) $\Delta \vdash^{\hat{\Delta}_{\text{app}};b} \grave{\tau}_1 \rightsquigarrow \tau_1\ \mathsf{type}$ — by assumption
(5) $\Delta\ \Gamma, x : \tau_1 \vdash^{\hat{\Delta}_{\text{app}};\hat{\Gamma}_{\text{app}};\hat{\Psi};\hat{\Phi};b} \grave{e}' \rightsquigarrow e' : \tau_2$ — by assumption
(6) $\Delta \cap \Delta_{\text{app}} = \emptyset$ — by assumption
(7) $\text{dom}(\Gamma) \cap \text{dom}(\Gamma_{\text{app}}) = \emptyset$ — by assumption
(8) $x \notin \text{dom}(\Gamma_{\text{app}})$ — by identification convention
(9) $\text{dom}(\Gamma, x : \tau_1) \cap \text{dom}(\Gamma_{\text{app}}) = \emptyset$ — by (7) and (8)
(10) $\Delta \cup \Delta_{\text{app}} \vdash \tau_1\ \mathsf{type}$ — by Lemma B.25 on (4) and (6)
(11) $\Delta \cup \Delta_{\text{app}}\ \Gamma, x : \tau_1 \cup \Gamma_{\text{app}} \vdash e' : \tau_2$ — by IH, part 2(a) on (5), (6) and (9)
(12) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}}, x : \tau_1 \vdash e' : \tau_2$ — by exchange over Γ_{app} on (11)
(13) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash \texttt{lam\{}\tau_1\texttt{\}(}x.e'\texttt{)} : \texttt{parr(}\tau_1\texttt{;}\tau_2\texttt{)}$ — by Rule (B.2b) on (10) and (12)

Case (B.10e).

(1) $\grave{e} = \texttt{prap(}\grave{e}_1\texttt{;}\grave{e}_2\texttt{)}$ — by assumption
(2) $e = \texttt{ap(}e_1\texttt{;}e_2\texttt{)}$ — by assumption
(3) $\Delta\ \Gamma \vdash^{\hat{\Delta}_{\text{app}};\hat{\Gamma}_{\text{app}};\hat{\Psi};\hat{\Phi};b} \grave{e}_1 \rightsquigarrow e_1 : \texttt{parr(}\tau_2\texttt{;}\tau\texttt{)}$ — by assumption
(4) $\Delta\ \Gamma \vdash^{\hat{\Delta}_{\text{app}};\hat{\Gamma}_{\text{app}};\hat{\Psi};\hat{\Phi};b} \grave{e}_2 \rightsquigarrow e_2 : \tau_2$ — by assumption
(5) $\Delta \cap \Delta_{\text{app}} = \emptyset$ — by assumption
(6) $\text{dom}(\Gamma) \cap \text{dom}(\Gamma_{\text{app}}) = \emptyset$ — by assumption
(7) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash e_1 : \texttt{parr(}\tau_2\texttt{;}\tau\texttt{)}$ — by IH, part 2(a) on (3), (5) and (6)
(8) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash e_2 : \tau_2$ — by IH, part 2(a) on (4), (5) and (6)
(9) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash \texttt{ap(}e_1\texttt{;}e_2\texttt{)} : \tau$ — by Rule (B.2c) on (7) and (8)

Case (B.10f).

(1) $\grave{e} = \texttt{prtlam(}t.\grave{e}'\texttt{)}$ — by assumption
(2) $e = \texttt{tlam(}t.e'\texttt{)}$ — by assumption

(3) $\tau = \texttt{all}(t.\tau')$ by assumption
(4) $\Delta, t\ \mathsf{type}\ \Gamma \vdash^{\hat{\Delta}_{\text{app}};\hat{\Gamma}_{\text{app}};\hat{\Psi};\hat{\Phi};b} \grave{e}' \rightsquigarrow e' : \tau'$ by assumption
(5) $\Delta \cap \Delta_{\text{app}} = \emptyset$ by assumption
(6) $\text{dom}(\Gamma) \cap \text{dom}(\Gamma_{\text{app}}) = \emptyset$ by assumption
(7) $t\ \mathsf{type} \notin \Delta_{\text{app}}$ by identification convention
(8) $\Delta, t\ \mathsf{type} \cap \Delta_{\text{app}} = \emptyset$ by (5) and (7)
(9) $\Delta, t\ \mathsf{type} \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash e' : \tau'$ by IH, part 2(a) on (4), (8) and (6)
(10) $\Delta \cup \Delta_{\text{app}}, t\ \mathsf{type}\ \Gamma \cup \Gamma_{\text{app}} \vdash e' : \tau'$ by exchange over Δ_{app} on (9)
(11) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash \texttt{tlam}(t.e') : \texttt{all}(t.\tau')$ by Rule (B.2d) on (10)

Case (B.10g) **through** (B.10m). These cases follow analagously, i.e. we apply the IH, part 2(a) to all proto-expression validation judgements, Lemma B.25 to all proto-type validation judgements, the identification convention to ensure that extended contexts remain disjoint, weakening and exchange as needed, and the corresponding typing rule in Rules (B.2e) through (B.2k).

Case (B.10n).
(1) $\grave{e} = \texttt{splicede}[m; n; \grave{\tau}]$ by assumption
(2) $\mathbb{E} = \langle \mathcal{D}; \Delta_{\text{app}} \rangle; \langle \mathcal{G}; \Gamma_{\text{app}} \rangle; \hat{\Psi}; b$ by assumption
(3) $\emptyset \vdash^{\text{ts}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type}$ by assumption
(4) $\text{parseUExp}(\text{subseq}(b; m; n)) = \hat{e}$ by assumption
(5) $\hat{\Delta}_{\text{app}}\ \hat{\Gamma}_{\text{app}} \vdash_{\hat{\Psi}} \hat{e} \rightsquigarrow e : \tau$ by assumption
(6) $\Delta \cap \Delta_{\text{app}} = \emptyset$ by assumption
(7) $\text{dom}(\Gamma) \cap \text{dom}(\Gamma_{\text{app}}) = \emptyset$ by assumption
(8) $\Delta_{\text{app}}\ \Gamma_{\text{app}} \vdash e : \tau$ by IH, part 1 on (5)
(9) $\Delta \cup \Delta_{\text{app}}\ \Gamma \cup \Gamma_{\text{app}} \vdash e : \tau$ by Lemma B.2 over Δ and Γ and exchange on (8)

Case (B.10o).
(1) $\grave{e} = \texttt{prmatch}[n](\grave{e}'; \{\grave{r}_i\}_{1 \leq i \leq n})$ by assumption
(2) $e = \texttt{match}[n](e'; \{r_i\}_{1 \leq i \leq n})$ by assumption
(3) $\Delta\ \Gamma \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{e}' \rightsquigarrow e' : \tau'$ by assumption
(4) $\{\Delta\ \Gamma \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{r}_i \rightsquigarrow r_i : \tau' \Rightarrow \tau\}_{1 \leq i \leq n}$ by assumption
(5) $\Delta \cap \Delta_{\text{app}} = \emptyset$ by assumption

(6)	$\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma_{\mathrm{app}}) = \emptyset$	by assumption
(7)	$\Delta \cup \Delta_{\mathrm{app}}\ \Gamma \cup \Gamma_{\mathrm{app}} \vdash e' : \tau'$	by IH, part 2(a) on (3), (5) and (6)
(8)	$\Delta \cup \Delta_{\mathrm{app}}\ \Gamma \cup \Gamma_{\mathrm{app}} \vdash r : \tau' \Rightarrow \tau$	by IH, part 2(b) on (4), (5) and (6)
(9)	$\Delta \cup \Delta_{\mathrm{app}}\ \Gamma \cup \Gamma_{\mathrm{app}} \vdash \mathtt{match}[n](e'; \{r_i\}_{1 \leq i \leq n}) : \tau$	by Rule (B.21) on (7) and (8)

(b) There is only one case.

Case (B.11).

(1)	$\grave{r} = \mathtt{prrule}(p.\grave{e})$	by assumption
(2)	$r = \mathtt{rule}(p.e)$	by assumption
(3)	$\Delta \vdash p : \tau \dashv\Vdash \Gamma'$	by assumption
(4)	$\Delta\ \Gamma \cup \Gamma' \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau'$	by assumption
(5)	$\Delta \cap \Delta_{\mathrm{app}} = \emptyset$	by assumption
(6)	$\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \emptyset$	by identification convention
(7)	$\mathrm{dom}(\Gamma_{\mathrm{app}}) \cap \mathrm{dom}(\Gamma') = \emptyset$	by identification convention
(8)	$\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma_{\mathrm{app}}) = \emptyset$	by assumption
(9)	$\mathrm{dom}(\Gamma \cup \Gamma') \cap \mathrm{dom}(\Gamma_{\mathrm{app}}) = \emptyset$	by standard finite set definitions and identities on (6), (7) and (8)
(10)	$\Delta \cup \Delta_{\mathrm{app}}\ \Gamma \cup \Gamma' \cup \Gamma_{\mathrm{app}} \vdash e : \tau'$	by IH, part 2(a) on (4), (5) and (9)
(11)	$\Delta \cup \Delta_{\mathrm{app}}\ \Gamma \cup \Gamma_{\mathrm{app}} \cup \Gamma' \vdash e : \tau'$	by exchange of Γ' and Γ_{app} on (10)
(12)	$\Delta \cup \Delta_{\mathrm{app}}\ \Gamma \cup \Gamma_{\mathrm{app}} \vdash \mathtt{rule}(p.e) : \tau \Rightarrow \tau'$	by Rule (B.3) on (3) and (11)

The mutual induction can be shown to be well-founded by showing that the following numeric metric on the judgements that we induct on is decreasing:

$$\|\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau\| = \|\hat{e}\|$$
$$\|\Delta\ \Gamma \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau\| = \|b\|$$

where $\|b\|$ is the length of b and $\|\hat{e}\|$ is the sum of the lengths of the seTSM literal bodies in $\hat{e}$, as defined in Sec. B.2.1.

The only case in the proof of part 1 that invokes part 2 is Case (B.6o). There, we have that the metric remains stable:

$$\begin{aligned}&\|\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow e : \tau\| \\ =&\|\emptyset\ \emptyset \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau\| \\ =&\|b\|\end{aligned}$$

The only case in the proof of part 2 that invokes part 1 is Case (B.10n). There, we have that $\mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{e}$ and the IH is applied to the judgement $\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau$. Because the metric is stable when passing from part 1 to part 2, we must have that it is strictly decreasing in the other direction:

$$\|\hat{\Delta}\ \hat{\Gamma} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau\| < \|\Delta\ \Gamma \vdash^{\hat{\Delta};\hat{\Gamma};\hat{\Psi};\hat{\Phi};b} \texttt{splicede}[m; n; \grave{\tau}] \rightsquigarrow e : \tau\|$$

i.e. by the definitions above,

$$\|\hat{e}\| < \|b\|$$

This is established by appeal to Condition B.17, which states that subsequences of b are no longer than b, and Condition B.12, which states that an unexpanded expression constructed by parsing a textual sequence b is strictly smaller, as measured by the metric defined above, than the length of b, because some characters must necessarily be used to apply a TSM and delimit each literal body. Combining these conditions, we have that $\|\hat{e}\| < \|b\|$ as needed. □

Theorem B.28 (Typed Expression Expansion). *If* $\langle \mathcal{D}; \Delta \rangle\ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau$ *then* $\Delta\ \Gamma \vdash e : \tau$.

Proof. This theorem follows immediately from Theorem B.27, part 1(a). □

B.4.4 Abstract Reasoning Principles

Lemma B.29 (Proto-Type Expansion Decomposition). *If* $\Delta \vdash^{\langle \mathcal{D}; \Delta_{app} \rangle; b} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type}$ *and* $\mathsf{summary}(\grave{\tau}) = \{\texttt{splicedt}[m_i; n_i]\}_{0 \le i < n}$ *then all of the following hold:*

1. $\{\langle \mathcal{D}; \Delta_{app} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \tau_i\ \mathsf{type}\}_{0 \le i < n}$
2. $\tau = [\{\tau_i / t_i\}_{0 \le i < n}]\tau'$ *for some* τ' *and fresh* $\{t_i\}_{0 \le i < n}$ *(i.e.* $\{t_i \notin dom(\Delta)\}_{0 \le i < n}$ *and* $\{t_i \notin dom(\Delta_{app})\}_{0 \le i < n}$*)*
3. $\mathsf{fv}(\tau') \subset dom(\Delta) \cup \{t_i\}_{0 \le i < n}$

Proof. By rule induction over Rules (B.9). In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta_{\text{app}} \rangle$ and $\mathbb{T} = \hat{\Delta}; b$.

Case (B.9a).

(1) $\grave{\tau} = t$ by assumption

(2) $\tau = t$ by assumption

(3) $\Delta = \Delta', t\ \mathsf{type}$ by assumption

(4) $\mathsf{summary}(\check{\tau}) = \emptyset$ by definition

(5) $\mathsf{fv}(t) = \{t\}$ by definition

(6) $\{t\} \subset \mathrm{dom}(\Delta) \cup \emptyset$ by definition

The conclusions hold as follows:

1. This conclusion holds trivially because $n = 0$.
2. Choose $\tau' = t$ and $\emptyset$.
3. (6)

Case (B.9b)**.**

(1) $\check{\tau} = \mathtt{prparr}(\check{\tau}_1; \check{\tau}_2)$ by assumption

(2) $\tau = \mathtt{parr}(\tau_1'; \tau_2')$ by assumption

(3) $\Delta \vdash^{\mathbb{T}} \check{\tau}_1 \rightsquigarrow \tau_1'\ \mathsf{type}$ by assumption

(4) $\Delta \vdash^{\mathbb{T}} \check{\tau}_2 \rightsquigarrow \tau_2'\ \mathsf{type}$ by assumption

(5) $\mathsf{summary}(\check{\tau}) = \mathsf{summary}(\check{\tau}_1) \cup \mathsf{summary}(\check{\tau}_2)$ by definition

(6) $\mathsf{summary}(\check{\tau}_1) = \{\mathtt{splicedt}[m_i; n_i]\}_{0 \le i < n'}$ by definition

(7) $\mathsf{summary}(\check{\tau}_2) = \{\mathtt{splicedt}[m_i; n_i]\}_{n' \le i < n}$ by definition

(8) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \tau_i\ \mathsf{type}\}_{0 \le i < n'}$ by IH on (3) and (6)

(9) $\tau_1' = [\{\tau_i / t_i\}_{0 \le i < n'}]\tau_1''$ for some τ_1'' and fresh $\{t_i\}_{0 \le i < n'}$ by IH on (3) and (6)

(10) $\mathsf{fv}(\tau_1'') \subset \mathrm{dom}(\Delta) \cup \{t_i\}_{0 \le i < n'}$ by IH on (3) and (6)

(11) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \tau_i\ \mathsf{type}\}_{n' \le i < n}$ by IH on (4) and (7)

(12) $\tau_2' = [\{\tau_i / t_i\}_{n' \le i < n}]\tau_2''$ for some τ_2'' and fresh $\{t_i\}_{n' \le i < n}$ by IH on (4) and (7)

(13) $\mathsf{fv}(\tau_2'') \subset \mathrm{dom}(\Delta) \cup \{t_i\}_{n' \le i < n}$ by IH on (4) and (7)

(14) $\{t_i\}_{0 \le i < n'} \cap \{t_i\}_{n' \le i < n} = \emptyset$ by identification convention

(15) $\mathsf{fv}(\tau_1'') \subset \mathrm{dom}(\Delta) \cup \{t_i\}_{0 \le i < n}$ by (10) and (14)

(16) $\mathsf{fv}(\tau_2'') \subset \mathrm{dom}(\Delta) \cup \{t_i\}_{0 \le i < n}$ by (13) and (14)

(17) $\tau_1' = [\{\tau_i / t_i\}_{0 \le i < n}]\tau_1''$ by substitution properties and (9) and (14)

(18) $\tau_2' = [\{\tau_i/t_i\}_{0\leq i<n}]\tau_2''$ — by substitution properties and (12) and (14)

(19) $\texttt{parr}(\tau_1';\tau_2') = [\{\tau_i/t_i\}_{0\leq i<n}]\texttt{parr}(\tau_1'';\tau_2'')$ — by substitution and (17) and (18)

(20) $\mathsf{fv}(\texttt{parr}(\tau_1'';\tau_2'')) = \mathsf{fv}(\tau_1'') \cup \mathsf{fv}(\tau_2'')$ — by definition

(21) $\mathsf{fv}(\texttt{parr}(\tau_1'';\tau_2'')) \subset \mathrm{dom}(\Delta) \cup \{t_i\}_{0\leq i<n}$ — by (20) and (15) and (16)

The conclusions hold as follows:

1. (8) $\cup$ (11)
2. Choosing $\{t_i\}_{0\leq i<n}$ and $\texttt{parr}(\tau_1'';\tau_2'')$, by (19)
3. (21)

Case (B.9c) **through** (B.9f)**.** These cases follow by analagous inductive argument.

Case (B.9g)**.**

(1) $\hat{\tau} = \texttt{splicedt}[m;n]$ — by assumption

(2) $\mathsf{summary}(\texttt{splicedt}[m;n]) = \{\texttt{splicedt}[m;n]\}$ — by definition

(3) $\mathsf{parseUTyp}(\mathsf{subseq}(b;m;n)) = \hat{\tau}$ — by assumption

(4) $\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \hat{\tau} \rightsquigarrow \tau\ \mathsf{type}$ — by assumption

(5) $t \notin \mathrm{dom}(\Delta)$ — by identification convention

(6) $t \notin \mathrm{dom}(\Delta_{\mathrm{app}})$ — by identification

(7) $\tau = [\tau/t]\tau$ — by definition

(8) $\mathsf{fv}(t) \subset \Delta \cup \{t\}$ — by definition

The conclusions hold as follows:

1. (3) and (4)
2. Choosing $\{t\}$ and t, by (5), (6) and (7)
3. (8)

□

Lemma B.30 (Proto-Expression and Proto-Rule Expansion Decomposition)**.**

1. *If* $\Delta\ \Gamma \vdash^{\langle \mathcal{D};\Delta_{app}\rangle;\langle \mathcal{G};\Gamma_{app}\rangle;\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau$ *and* $\mathsf{summary}(\grave{e}) = \{\texttt{splicedt}[m_i';n_i']\}_{0\leq i<n_{ty}} \cup \{\texttt{splicede}[m_i;n_i;\grave{\tau}_i]\}_{0\leq i<n_{exp}}$ *then all of the following hold:*

(a) $\{\langle \mathcal{D}; \Delta_{app} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow \tau'_i \ \mathsf{type}\}_{0 \le i < n_{ty}}$

(b) $\{\emptyset \vdash^{\langle \mathcal{D}; \Delta_{app} \rangle; b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \le i < n_{exp}}$

(c) $\{\langle \mathcal{D}; \Delta_{app} \rangle \ \langle \mathcal{G}; \Gamma_{app} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow e_i : \tau_i\}_{0 \le i < n_{exp}}$

(d) $e = [\{\tau'_i / t_i\}_{0 \le i < n_{ty}}, \{e_i / x_i\}_{0 \le i < n_{exp}}]e'$ *for some* e' *and* $\{t_i\}_{0 \le i < n_{ty}}$ *and* $\{x_i\}_{0 \le i < n_{exp}}$ *such that* $\{t_i\}_{0 \le i < n_{ty}}$ *fresh (i.e.* $\{t_i \notin dom(\Delta)\}_{0 \le i < n_{ty}}$ *and* $\{t_i \notin dom(\Delta_{app})\}_{0 \le i < n_{ty}}$*) and* $\{x_i\}_{0 \le i < n_{exp}}$ *fresh (i.e.* $\{x_i \notin dom(\Gamma)\}_{0 \le i < n_{exp}}$ *and* $\{x_i \notin dom(\Gamma_{app})\}_{0 \le i < n_{ty}}$*)*

(e) $\mathsf{fv}(e') \subset dom(\Delta) \cup dom(\Gamma) \cup \{t_i\}_{0 \le i < n_{ty}} \cup \{x_i\}_{0 \le i < n_{exp}}$

2. *If* $\Delta \ \Gamma \vdash^{\langle \mathcal{D}; \Delta_{app} \rangle; \langle \mathcal{G}; \Gamma_{app} \rangle; \hat{\Psi}; \hat{\Phi}; b} \grave{r} \rightsquigarrow r : \tau \Mapsto \tau'$ *and*

$$\mathsf{summary}(\grave{r}) = \{\texttt{splicedt}[m'_i; n'_i]\}_{0 \le i < n_{ty}} \cup \{\texttt{splicede}[m_i; n_i; \grave{\tau}_i]\}_{0 \le i < n_{exp}}$$

then all of the following hold:

(a) $\{\langle \mathcal{D}; \Delta_{app} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow \tau'_i \ \mathsf{type}\}_{0 \le i < n_{ty}}$

(b) $\{\emptyset \vdash^{\langle \mathcal{D}; \Delta_{app} \rangle; b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \le i < n_{exp}}$

(c) $\{\langle \mathcal{D}; \Delta_{app} \rangle \ \langle \mathcal{G}; \Gamma_{app} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow e_i : \tau_i\}_{0 \le i < n_{exp}}$

(d) $r = [\{\tau'_i / t_i\}_{0 \le i < n_{ty}}, \{e_i / x_i\}_{0 \le i < n_{exp}}]r'$ *for some* e' *and fresh* $\{t_i\}_{0 \le i < n_{ty}}$ *and fresh* $\{x_i\}_{0 \le i < n_{exp}}$

(e) $\mathsf{fv}(r') \subset dom(\Delta) \cup dom(\Gamma) \cup \{t_i\}_{0 \le i < n_{ty}} \cup \{x_i\}_{0 \le i < n_{exp}}$

Proof. By rule induction over Rules (B.10) and Rule (B.11). In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta_{\text{app}} \rangle$ and $\hat{\Gamma} = \langle \mathcal{G}; \Gamma_{\text{app}} \rangle$ and $\mathbb{E} = \hat{\Delta}; \hat{\Gamma}; \hat{\Psi}; \hat{\Phi}; b$.

1. **Case** (B.10a)**.**

(1)	$\grave{e} = x$	by assumption
(2)	$e = x$	by assumption
(3)	$\Gamma = \Gamma', x : \tau$	by assumption
(4)	$\mathsf{summary}(x) = \{\}$	by definition
(5)	$\mathsf{fv}(x) = \{x\}$	by definition
(6)	$\mathsf{fv}(x) \subset \mathrm{dom}(\Gamma)$	by definition
(7)	$\mathsf{fv}(x) \subset \mathrm{dom}(\Gamma) \cup \mathrm{dom}(\Delta)$	by (6) and definition of subset

The conclusions hold as follows:

(a) This conclusion holds trivially because $n_{\text{ty}} = 0$.

(b) This conclusion holds trivially because $n_{\text{exp}} = 0$.

(c) This conclusion holds trivially because $n_{\text{exp}} = 0$.

(d) Choose x, $\emptyset$ and $\emptyset$.

(e) (7)

Case (B.10b) **through** (B.10m)**.** These cases follow by straightforward inductive argument.

Case (B.10n)**.**

(1) $\grave{e} = \texttt{splicede}[m; n; \grave{\tau}]$ by assumption

(2) $\mathsf{summary}(\texttt{splicede}[m; n; \grave{\tau}]) = \mathsf{summary}(\grave{\tau}) \cup \{\texttt{splicede}[m; n; \grave{\tau}]\}$ by definition

(3) $\mathsf{summary}(\grave{\tau}) = \{\texttt{splicedt}[m'_i; n'_i]\}_{0 \leq i < n_{\mathrm{ty}}}$ by definition

(4) $\emptyset \vdash^{\mathsf{ts}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau\ \mathsf{type}$ by assumption

(5) $\mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{e}$ by assumption

(6) $\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle\ \langle \mathcal{G}; \Gamma_{\mathrm{app}} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{e} \rightsquigarrow e : \tau$ by assumption

(7) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow \tau'_i\ \mathsf{type}\}_{0 \leq i < n_{\mathrm{ty}}}$ by Lemma B.29 on (4) and (3)

(8) $x \notin \mathrm{dom}(\Gamma)$ by identification convention

(9) $x \notin \mathrm{dom}(\Gamma_{\mathrm{app}})$ by identification convention

(10) $x \notin \mathrm{dom}(\Delta)$ by identificaiton convention

(11) $x \notin \mathrm{dom}(\Delta_{\mathrm{app}})$ by identification convention

(12) $e = [\{\tau'_i / t_i\}_{0 \leq i < n_{\mathrm{ty}}}, e/x]x$ by definition

(13) $\mathsf{fv}(x) = \{x\}$ by definition

(14) $\mathsf{fv}(x) \subset \mathrm{dom}(\Delta) \cup \mathrm{dom}(\Gamma) \cup \{t_i\}_{0 \leq i < n_{\mathrm{ty}}} \cup \{x\}$ by definition

The conclusions hold as follows:

(a) (7)

(b) {(4)}

(c) {(6)}

(d) Choosing x, $\{t_i\}_{0 \leq i < n_{\mathrm{ty}}}$ and $\{x\}$, by (8), (9), (10), (11) and (12).

(e) (14)

Case (B.10o)**.**

(1) $\grave{e} = \texttt{prmatch}[n](\grave{e}'; \{\grave{r}_i\}_{1 \leq i \leq n})$ by assumption

(2) $e = \texttt{match}[n](\tau; e')\{r_i\}_{1 \leq i \leq n}$ by assumption

(3) $\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'$ by assumption

(4) $\{\Delta\ \Gamma \vdash^{\mathbb{E}} \grave{r}_j \rightsquigarrow r_j : \tau' \Rightarrow \tau\}_{1 \leq j \leq n}$ by assumption

(5) $\mathsf{summary}(\texttt{prmatch}[n](\grave{e}'; \{\grave{r}_i\}_{1 \leq i \leq n})) =$

$\mathsf{summary}(\grave{e}) \cup \bigcup_{0 \le i < n} \mathsf{summary}(\grave{r}_i)$ by definition

(6) $\mathsf{summary}(\grave{e}') = \{\mathtt{splicedt}[m'_i; n'_i]\}_{0 \le i < n'_{\mathrm{ty}}} \cup \{\mathtt{splicede}[m_i; n_i; \grave{\tau}_i]\}_{0 \le i < n'_{\mathrm{exp}}}$ by definition

(7) $\{\mathsf{summary}(\grave{r}_j) = \{\mathtt{splicedt}[m'_{i,j}; n'_{i,j}]\}_{0 \le i < n_{\mathrm{ty},j}} \cup \{\mathtt{splicede}[m_{i,j}; n_{i,j}; \grave{\tau}_{i,j}]\}_{0 \le i < n_{\mathrm{exp},j}}\}_{0 \le j < n}$ by definition

(8) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow \tau'_i \ \mathsf{type}\}_{0 \le i < n'_{\mathrm{ty}}}$ by IH, part 1 on (3) and (6)

(9) $\{\emptyset \vdash^{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle; b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \le i < n'_{\mathrm{exp}}}$ by IH, part 1 on (3) and (6)

(10) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \ \langle \mathcal{G}; \Gamma_{\mathrm{app}} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow e_i : \tau_i\}_{0 \le i < n'_{\mathrm{exp}}}$ by IH, part 1 on (3) and (6)

(11) $e' = [\{\tau'_i / t_i\}_{0 \le i < n'_{\mathrm{ty}}}, \{e_i / x_i\}_{0 \le i < n'_{\mathrm{exp}}}]e''$ for some e'' and fresh $\{t_i\}_{0 \le i < n'_{\mathrm{ty}}}$ and fresh $\{x_i\}_{0 \le i < n'_{\mathrm{exp}}}$ by IH, part 1 on (3) and (6)

(12) $\mathsf{fv}(e'') \subset \mathrm{dom}(\Delta) \cup \mathrm{dom}(\Gamma) \cup \{t_i\}_{0 \le i < n'_{\mathrm{ty}}} \cup \{x_i\}_{0 \le i < n'_{\mathrm{exp}}}$ by IH, part 1 on (3) and (6)

(13) $\{\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_{i,j}; n'_{i,j})) \rightsquigarrow \tau'_{i,j} \ \mathsf{type}\}_{0 \le i < n_{\mathrm{ty},j}}\}_{0 \le j < n}$ by IH, part 2 over (4) and (7)

(14) $\{\{\emptyset \vdash^{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle; b} \grave{\tau}_{i,j} \rightsquigarrow \tau_{i,j} \ \mathsf{type}\}_{0 \le i < n_{\mathrm{exp},j}}\}_{0 \le j < n}$ by IH, part 2 over (4) and (7)

(15) $\{\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \ \langle \mathcal{G}; \Gamma_{\mathrm{app}} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m_{i,j}; n_{i,j})) \rightsquigarrow e_{i,j} : \tau_{i,j}\}_{0 \le i < n_{\mathrm{exp},j}}\}_{0 \le j < n}$ by IH, part 2 over (4) and (7)

(16) $\{r_j = [\{\tau'_{i,j} / t_{i,j}\}_{0 \le i < n_{\mathrm{ty},j}}, \{e_{i,j} / x_{i,j}\}_{0 \le i < n_{\mathrm{exp},j}}]r'_j\}_{0 \le j < n}$ for some $\{r'_j\}_{0 \le j < n}$ and fresh $\{\{t_{i,j}\}_{0 \le i < n_{\mathrm{ty},j}}\}_{0 \le j < n}$ and fresh $\{\{x_{i,j}\}_{0 \le i < n_{\mathrm{exp},j}}\}_{0 \le j < n}$ by IH, part 2 over (4) and (7)

(17) $\{\mathsf{fv}(r'_j) \subset \mathrm{dom}(\Delta) \cup \mathrm{dom}(\Gamma) \cup \{t_{i,j}\}_{0 \le i < n_{\mathrm{ty},j}} \cup \{x_{i,j}\}_{0 \le i < n_{\mathrm{exp},j}}\}_{0 \le j < n}$ by IH, part 2 over (4) and (7)

(18) $(\cup_{0 \le j < n} \{t_{i,j}\}_{0 \le i < n_{\mathrm{ty},j}}) \cap \{t_i\}_{0 \le i < n'_{\mathrm{ty}}} = \emptyset$ by identification convention

(19) $(\cup_{0 \le j < n} \{x_{i,j}\}_{0 \le i < n_{\mathrm{exp},j}}) \cap \{x_i\}_{0 \le i < n'_{\mathrm{exp}}} = \emptyset$ by identification convention

(20) $e' = [\{\tau_i'/t_i\}_{0\le i<n'_{\text{ty}}} \cup_{0\le j<n} \{\tau_{i,j}/t_{i,j}\}_{0\le i<n_{\text{ty},j}}, \{e_i/x_i\}_{0\le i<n_{\text{exp}}} \cup_{0\le j<n} \{\tau_{i,j}/t_{i,j}\}_{0\le i<n_{\text{ty},j}}]e''$ by substitution properties and (11) and (12) and (18) and (19)

(21) $\{r_j = [\{\tau_i'/t_i\}_{0\le i<n'_{\text{ty}}} \cup_{0\le j<n} \{\tau_{i,j}/t_{i,j}\}_{0\le i<n_{\text{ty},j}}, \{e_i/x_i\}_{0\le i<n_{\text{exp}}} \cup_{0\le j<n} \{\tau_{i,j}/t_{i,j}\}_{0\le i<n_{\text{ty},j}}]r_j'\}_{0\le j<n}$ by substitution properties and (16) and (17) and (18) and (19)

(22) $e = [\{\tau_i'/t_i\}_{0\le i<n'_{\text{ty}}} \cup_{0\le j<n} \{\tau_{i,j}/t_{i,j}\}_{0\le i<n_{\text{ty},j}}, \{e_i/x_i\}_{0\le i<n'_{\text{exp}}} \cup_{0\le j<n} \{e_{i,j}/x_{i,j}\}_{0\le i<n_{\text{exp},j}}]\texttt{match}[n](e''; \{r_i'\}_{1\le i\le n})$ by (20) and (21) and definition of substitution

(23) $\text{fv}(e'') \subset \text{dom}(\Delta) \cup \text{dom}(\Gamma) \cup \{t_i\}_{0\le i<n'_{\text{ty}}} \cup_{0\le j<n} \{t_{i,j}\}_{0\le i<n_{\text{ty},j}} \cup \{x_i\}_{0\le i<n'_{\text{exp}}} \cup_{0\le j<n} \{x_{i,j}\}_{0\le i<n_{\text{exp},j}}$ by (12) and (18) and (19)

(24) $\{\text{fv}(r_j') \subset \text{dom}(\Delta) \cup \text{dom}(\Gamma) \cup \{t_i\}_{0\le i<n'_{\text{ty}}} \cup_{0\le j<n} \{t_{i,j}\}_{0\le i<n_{\text{ty},j}} \cup \{x_i\}_{0\le i<n'_{\text{exp}}} \cup_{0\le j<n} \{x_{i,j}\}_{0\le i<n_{\text{exp},j}}\}_{0\le j<n}$ by (17) and (18) and (19)

(25) $\text{fv}(\texttt{match}[n](e''; \{r_i'\}_{1\le i\le n})) \subset \text{dom}(\Delta) \cup \text{dom}(\Gamma) \cup \{t_i\}_{0\le i<n'_{\text{ty}}} \cup_{0\le j<n} \{t_{i,j}\}_{0\le i<n_{\text{ty},j}} \cup \{x_i\}_{0\le i<n'_{\text{exp}}} \cup_{0\le j<n} \{x_{i,j}\}_{0\le i<n_{\text{exp},j}}$ by (23) and (24)

The conclusions hold as follows:

(a) (8) $\cup \bigcup_{0\le j<n}$ (13)$_j$

(b) (9) $\cup \bigcup_{0\le j<n}$ (14)$_j$

(c) (10) $\cup \bigcup_{0\le j<n}$ (15)$_j$

(d) Choose:

i. $\texttt{match}[n](e''; \{r_i'\}_{1\le i\le n})$

ii. $\{t_i\}_{0\le i<n'_{\text{ty}}} \cup \{\{t_{i,j}\}_{0\le i<n_{\text{ty},j}}\}_{0\le j<n}$; and

iii. $\{x_i\}_{0\le i<n'_{\text{exp}}} \cup \{\{x_{i,j}\}_{0\le i<n_{\text{exp},j}}\}_{0\le j<n}$; and

We have $e = [\{\tau_i'/t_i\}_{0\le i<n'_{\text{ty}}} \cup \{\{\tau_{i,j}/t_{i,j}\}_{0\le i<n_{\text{ty},j}}\}_{0\le j<n}, \{e_i/x_i\}_{0\le i<n'_{\text{exp}}} \cup \{\{e_{i,j}/x_{i,j}\}_{0\le i<n_{\text{exp},j}}\}_{0\le j<n}]\texttt{match}[n](e''; \{r_i'\}_{1\le i\le n})$ by (22).

(e) (25)

2. By rule induction over the rule typing assumption. There is only one case. In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta_{\text{app}} \rangle$ and $\hat{\Gamma} = \langle \mathcal{G}; \Gamma_{\text{app}} \rangle$ and $\mathbb{E} = \hat{\Delta}; \hat{\Gamma}; \hat{\Psi}; \hat{\Phi}; b$.

Case (B.11)**.**

(1) $\grave{r} = \texttt{prrule}(p.\grave{e})$ by assumption
(2) $r = \texttt{rule}(p.e)$ by assumption
(3) $\Delta \vdash p : \tau \dashv\Vdash \Gamma'$ by assumption
(4) $\Delta\ \Gamma \cup \Gamma' \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'$ by assumption
(5) $\mathsf{summary}(\grave{r}) = \mathsf{summary}(\grave{e})$ by definition
(6) $\mathsf{summary}(\grave{e}) = \{\texttt{splicedt}[m_i'; n_i']\}_{0 \le i < n_{\mathrm{ty}}} \cup \{\texttt{splicede}[m_i; n_i; \grave{\tau}_i]\}_{0 \le i < n_{\mathrm{exp}}}$ by definition
(7) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow \tau_i'\ \mathsf{type}\}_{0 \le i < n_{\mathrm{ty}}}$ by IH, part 1 on (4) and (6)
(8) $\{\emptyset \vdash^{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle; b} \grave{\tau}_i \rightsquigarrow \tau_i\ \mathsf{type}\}_{0 \le i < n_{\mathrm{exp}}}$ by IH, part 1 on (4) and (6)
(9) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle\ \langle \mathcal{G}; \Gamma_{\mathrm{app}} \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow e_i : \tau_i\}_{0 \le i < n_{\mathrm{exp}}}$ by IH, part 1 on (4) and (6)
(10) $e = [\{\tau_i'/t_i\}_{0 \le i < n_{\mathrm{ty}}}, \{e_i/x_i\}_{0 \le i < n_{\mathrm{exp}}}]e'$ for some e' and fresh $\{t_i\}_{0 \le i < n_{\mathrm{ty}}}$ and fresh $\{x_i\}_{0 \le i < n_{\mathrm{exp}}}$ by IH, part 1 on (4) and (6)
(11) $\mathsf{fv}(e') \subset \mathrm{dom}(\Delta) \cup \mathrm{dom}(\Gamma) \cup \mathrm{dom}(\Gamma') \cup \{t_i\}_{0 \le i < n_{\mathrm{ty}}} \cup \{x_i\}_{0 \le i < n_{\mathrm{exp}}}$ by IH, part 1 on (4) and (6)
(12) $r = [\{\tau_i'/t_i\}_{0 \le i < n_{\mathrm{ty}}}, \{e_i/x_i\}_{0 \le i < n_{\mathrm{exp}}}]\texttt{rule}(p.e')$ by substitution properties and (10)
(13) $\mathsf{fv}(p) = \mathrm{dom}(\Gamma')$ by Lemma B.5 on (3)
(14) $\mathsf{fv}(\texttt{rule}(p.e')) \subset \mathrm{dom}(\Delta) \cup \mathrm{dom}(\Gamma) \cup \{t_i\}_{0 \le i < n_{\mathrm{ty}}} \cup \{x_i\}_{0 \le i < n_{\mathrm{exp}}}$ by definition of $\mathsf{fv}(r)$ and (11) and (13)

The conclusions hold as follows:

(a) (7)
(b) (8)
(c) (9)
(d) Choosing $\texttt{rule}(p.e')$ and $\{t_i\}_{0 \le i < n_{\mathrm{ty}}}$ and $\{x_i\}_{0 \le i < n_{\mathrm{exp}}}$, by (12)
(e) (14)

□

Theorem B.31 (seTSM Abstract Reasoning Principles)**.** *If* $\langle \mathcal{D}; \Delta \rangle\ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow e : \tau$ *then:*

1. ***(Typing 1)*** $\hat{\Psi} = \hat{\Psi}', \hat{a} \rightsquigarrow a \hookrightarrow \texttt{setsm}(\tau; e_{parse})$ *and* $\Delta\ \Gamma \vdash e : \tau$

2. $b \downarrow_{\mathsf{Body}} e_{body}$
3. $e_{parse}(e_{body}) \Downarrow \mathtt{inj}[\mathtt{SuccessE}](e_{proto})$
4. $e_{proto} \uparrow_{\mathsf{PrExpr}} \grave{e}$
5. ***(Segmentation)*** $\mathsf{seg}(\grave{e})$ segments b
6. $\mathsf{summary}(\grave{e}) = \{\mathtt{splicedt}[m_i'; n_i']\}_{0 \le i < n_{ty}} \cup \{\mathtt{splicede}[m_i; n_i; \grave{\tau}_i]\}_{0 \le i < n_{exp}}$
7. ***(Typing 2)*** $\{\langle \mathcal{D}; \Delta \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow \tau_i' \ \mathsf{type}\}_{0 \le i < n_{ty}}$ *and* $\{\Delta \vdash \tau_i' \ \mathsf{type}\}_{0 \le i < n_{ty}}$
8. ***(Typing 3)*** $\{\emptyset \vdash^{\langle \mathcal{D}; \Delta \rangle; b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \le i < n_{exp}}$ *and* $\{\Delta \vdash \tau_i \ \mathsf{type}\}_{0 \le i < n_{exp}}$
9. ***(Typing 4)*** $\{\langle \mathcal{D}; \Delta \rangle \ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow e_i : \tau_i\}_{0 \le i < n_{exp}}$ *and* $\{\Delta \ \Gamma \vdash e_i : \tau_i\}_{0 \le i < n_{exp}}$
10. ***(Capture Avoidance)*** $e = [\{\tau_i'/t_i\}_{0 \le i < n_{ty}}, \{e_i/x_i\}_{0 \le i < n_{exp}}]e'$ *for some* $\{t_i\}_{0 \le i < n_{ty}}$ *and* $\{x_i\}_{0 \le i < n_{exp}}$ *and* e'
11. ***(Context Independence)*** $\mathsf{fv}(e') \subset \{t_i\}_{0 \le i < n_{ty}} \cup \{x_i\}_{0 \le i < n_{exp}}$

Proof. By rule induction over Rules (B.6). There is only one rule that applies. In the following, let $\hat{\Delta} = \langle \mathcal{D}; \Delta \rangle$ and $\hat{\Gamma} = \langle \mathcal{G}; \Gamma \rangle$.

Case (B.6o)**.**

(1) $\hat{\Psi} = \hat{\Psi}', \hat{a} \rightsquigarrow a \hookrightarrow \mathtt{setsm}(\tau; e_{\text{parse}})$ — by assumption

(2) $\langle \mathcal{D}; \Delta \rangle \ \langle \mathcal{G}; \Gamma \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{a}$ ‘b‘ $\rightsquigarrow e : \tau$ — by assumption

(3) $\Delta \ \Gamma \vdash e : \tau$ — by Theorem B.28 on (2)

(4) $b \downarrow_{\mathsf{Body}} e_{\text{body}}$ — by assumption

(5) $e_{\text{parse}}(e_{\text{body}}) \Downarrow \mathtt{inj}[\mathtt{SuccessE}](e_{\text{proto}})$ — by assumption

(6) $e_{\text{proto}} \uparrow_{\mathsf{PrExpr}} \grave{e}$ — by assumption

(7) $\mathsf{seg}(\grave{e})$ segments b — by assumption

(8) $\emptyset \ \emptyset \vdash^{\hat{\Delta}; \hat{\Gamma}; \hat{\Psi}; \hat{\Phi}; b} \grave{e} \rightsquigarrow e : \tau$ — by assumption

(9) $\mathsf{summary}(\grave{e}) = \{\mathtt{splicedt}[m_i'; n_i']\}_{0 \le i < n_{\text{ty}}} \cup \{\mathtt{splicede}[m_i; n_i; \grave{\tau}_i]\}_{0 \le i < n_{\text{exp}}}$ — by definition

(10) $\{\langle \mathcal{D}; \Delta \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow \tau_i' \ \mathsf{type}\}_{0 \le i < n_{\text{ty}}}$ — by Lemma B.30 on (8) and (9)

(11) $\{\Delta \vdash \tau_i' \ \mathsf{type}\}_{0 \le i < n_{\text{ty}}}$ — by Lemma B.24, part 1 over (10)

(12) $\{\emptyset \vdash^{\langle \mathcal{D};\Delta\rangle;b} \grave{\tau}_i \rightsquigarrow \tau_i \text{ type}\}_{0\le i<n_{\text{exp}}}$ by Lemma B.30 on (8) and (9)

(13) $\emptyset \cap \Delta = \emptyset$ by definition

(14) $\{\Delta \vdash \tau_i \text{ type}\}_{0\le i<n_{\text{exp}}}$ by Lemma B.24, part 2 over (12) and (13)

(15) $\{\langle \mathcal{D};\Delta\rangle\ \langle \mathcal{G};\Gamma\rangle \vdash_{\hat{\Psi};\hat{\Phi}} \text{parseUExp}(\text{subseq}(b;m_i;n_i)) \rightsquigarrow e_i : \tau_i\}_{0\le i<n_{\text{exp}}}$ by Lemma B.30 on (8) and (9)

(16) $\{\Delta\ \Gamma \vdash e_i : \tau_i\}_{0\le i<n_{\text{exp}}}$ by Theorem B.28 over (15)

(17) $e = [\{\tau_i'/t_i\}_{0\le i<n_{\text{ty}}}, \{e_i/x_i\}_{0\le i<n_{\text{exp}}}]e'$ for some e' and fresh $\{t_i\}_{0\le i<n_{\text{ty}}}$ and fresh $\{x_i\}_{0\le i<n_{\text{exp}}}$ by Lemma B.30 on (8) and (9)

(18) $\text{fv}(e') \subset \{t_i\}_{0\le i<n_{\text{ty}}} \cup \{x_i\}_{0\le i<n_{\text{exp}}}$ by Lemma B.30 on (8) and (9)

The conclusions hold as follows:

1. (1) and (3)
2. (4)
3. (5)
4. (6)
5. (7)
6. (9)
7. (10) and (11)
8. (12) and (14)
9. (15) and (16)
10. (17)
11. (18)

□

Lemma B.32 (Proto-Pattern Expansion Decomposition). *If* $\grave{p} \rightsquigarrow p : \tau \dashv\!\vdash^{\hat{\Delta};\hat{\Phi};b} \hat{\Gamma}$ *and*

$$\text{summary}(\grave{p}) = \{\texttt{splicedt}[m_i';n_i']\}_{0\le i<n_{ty}} \cup \{\texttt{splicedp}[m_i;n_i;\grave{\tau}_i]\}_{0\le i<n_{pat}}$$

then all of the following hold:

1. $\{\hat{\Delta} \vdash \text{parseUTyp}(\text{subseq}(b;m_i';n_i')) \rightsquigarrow \tau_i' \text{ type}\}_{0\le i<n_{ty}}$

2. $\{\emptyset \vdash^{\hat{\Delta};b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \leq i < n_{pat}}$
3. $\{\hat{\Delta} \vdash_{\hat{\Phi}} \mathsf{parseUPat}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow p_i : \tau_i \dashv\!| \, \hat{\Gamma}_i\}_{0 \leq i < n_{pat}}$
4. $\hat{\Gamma} = \biguplus_{0 \leq i < n_{pat}} \hat{\Gamma}_i$

Proof. By rule induction over Rules (B.12). In the following, let $\mathbb{P} = \hat{\Delta}; \hat{\Phi}; b$.

Case (B.12a)**.**

(1) $\grave{p} = \texttt{prwildp}$ by assumption
(2) $e = \texttt{wildp}$ by assumption
(3) $\hat{\Gamma} = \langle \emptyset; \emptyset \rangle$ by assumption
(4) $\mathsf{summary}(\texttt{prwildp}) = \emptyset$ by definition

The conclusions hold as follows:

1. This conclusion holds trivially because $n_{\text{ty}} = 0$.
2. This conclusion holds trivially because $n_{\text{pat}} = 0$.
3. This conclusion holds trivially because $n_{\text{pat}} = 0$.
4. This conclusion holds trivially because $\hat{\Gamma} = \emptyset$ and $n_{\text{pat}} = 0$.

Case (B.12b)**.**

(1) $\grave{p} = \texttt{prfoldp}(\grave{p}')$ by assumption
(2) $p = \texttt{foldp}(p')$ by assumption
(3) $\tau = \texttt{rec}(t.\tau')$ by assumption
(4) $\grave{p} \rightsquigarrow p : [\texttt{rec}(t.\tau')/t]\tau' \dashv\!|^{\mathbb{P}} \hat{\Gamma}$ by assumption
(5) $\mathsf{summary}(\texttt{prfoldp}(\grave{p}')) = \mathsf{summary}(\grave{p}')$ by definition
(6) $\mathsf{summary}(\grave{p}') = \{\texttt{splicedt}[m_i'; n_i']\}_{0 \leq i < n_{\text{ty}}} \cup \{\texttt{splicedp}[m_i; n_i; \grave{\tau}_i]\}_{0 \leq i < n_{\text{pat}}}$ by definition
(7) $\{\hat{\Delta} \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow \tau_i' \ \mathsf{type}\}_{0 \leq i < n_{\text{ty}}}$ by IH on (4) and (6)
(8) $\{\emptyset \vdash^{\hat{\Delta};b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \leq i < n_{\text{pat}}}$ by IH on (4) and (6)
(9) $\{\hat{\Delta} \vdash_{\hat{\Phi}} \mathsf{parseUPat}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow p_i : \tau_i \dashv\!| \, \hat{\Gamma}_i\}_{0 \leq i < n_{\text{pat}}}$ by IH on (4) and (6)
(10) $\hat{\Gamma} = \biguplus_{0 \leq i < n_{\text{pat}}} \hat{\Gamma}_i$ by IH on (4) and (6)

The conclusions hold as follows:

1. (7)
2. (8)
3. (9)
4. (10)

Case (B.12c)**.**

(1) $\grave{p} = \texttt{prtplp}[L](\{j \hookrightarrow \grave{p}_j\}_{j \in L})$ by assumption

(2) $p = \texttt{tplp}[L](\{j \hookrightarrow p_j\}_{j \in L})$ by assumption

(3) $\tau = \texttt{prod}[L](\{j \hookrightarrow \tau_j\}_{j \in L})$ by assumption

(4) $\hat{\Gamma} = \uplus_{j \in L} \hat{\Gamma}_j$ by assumption

(5) $\{\grave{p}_j \rightsquigarrow p_j : \tau_j \dashv\!\!\mid^{\mathbb{P}} \hat{\Gamma}_j\}_{j \in L}$ by assumption

(6) $\mathsf{summary}(\texttt{prtplp}[L](\{j \hookrightarrow \grave{p}_j\}_{j \in L})) = \bigcup_{j \in L} \mathsf{summary}(\grave{p}_j)$ by definition

(7) $\{\mathsf{summary}(\grave{p}_j) = \{\texttt{splicedt}[m'_{i,j}; n'_{i,j}]\}_{0 \le i < n_{\mathrm{ty},j}} \cup \{\texttt{splicedp}[m_{i,j}; n_{i,j}; \grave{\tau}_{i,j}]\}_{0 \le i < n_{\mathrm{pat},j}}\}_{j \in L}$ by definition

(8) $n_{\mathrm{pat}} = \Sigma_{j \in L} n_{\mathrm{pat},j}$ by definition

(9) $\{\{\hat{\Delta} \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_{i,j}; n'_{i,j})) \rightsquigarrow \tau'_{i,j}\ \mathsf{type}\}_{0 \le i < n_{\mathrm{ty},j}}\}_{j \in L}$ by IH over (5) and (7)

(10) $\{\{\emptyset \vdash^{\hat{\Delta}; b} \grave{\tau}_{i,j} \rightsquigarrow \tau_{i,j}\ \mathsf{type}\}_{0 \le i < n_{\mathrm{pat},j}}\}_{j \in L}$ by IH over (5) and (7)

(11) $\{\{\hat{\Delta} \vdash_{\hat{\Phi}} \mathsf{parseUPat}(\mathsf{subseq}(b; m_{i,j}; n_{i,j})) \rightsquigarrow p_{i,j} : \tau_{i,j} \dashv\!\!\mid \hat{\Gamma}_{i,j}\}_{0 \le i < n_{\mathrm{pat},j}}\}_{j \in L}$ by IH over (5) and (7)

(12) $\{\hat{\Gamma}_j = \uplus_{0 \le i < n_{\mathrm{pat},j}} \hat{\Gamma}_{i,j}\}_{j \in L}$ by IH over (5) and (7)

(13) $\uplus_{j \in L} \hat{\Gamma}_j = \uplus_{j \in L} \uplus_{i \in n_{\mathrm{pat},j}} \hat{\Gamma}_{i,j}$ by definition and (12)

The conclusions hold as follows:

1. $\bigcup_{j \in L} \bigcup_{i \in n_{\mathrm{ty},j}} (9)_{i,j}$
2. $\bigcup_{j \in L} \bigcup_{i \in n_{\mathrm{pat},j}} (10)_{i,j}$
3. $\bigcup_{j \in L} \bigcup_{i \in n_{\mathrm{pat},j}} (11)_{i,j}$
4. (13)

Case (B.12d)**.**

(1) $\grave{p} = \texttt{prinjp}[\ell](\grave{p}')$ by assumption

(2) $p = \mathtt{injp}[\ell](p')$ by assumption

(3) $\tau = \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')$ by assumption

(4) $\grave{p} \rightsquigarrow p : \tau' \dashv\!\Vert^{\mathbb{P}} \hat{\Gamma}$ by assumption

(5) $\mathsf{summary}(\mathtt{prinjp}[\ell](\grave{p}')) = \mathsf{summary}(\grave{p}')$ by definition

(6) $\mathsf{summary}(\grave{p}') = \{\mathtt{splicedt}[m_i'; n_i']\}_{0 \le i < n_{\mathrm{ty}}} \cup \{\mathtt{splicedp}[m_i; n_i; \grave{\tau}_i]\}_{0 \le i < n_{\mathrm{pat}}}$ by definition

(7) $\{\hat{\Delta} \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow \tau_i' \ \mathsf{type}\}_{0 \le i < n_{\mathrm{ty}}}$ by IH on (4) and (6)

(8) $\{\emptyset \vdash^{\hat{\Delta}; b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \le i < n_{\mathrm{pat}}}$ by IH on (4) and (6)

(9) $\{\hat{\Delta} \vdash_{\hat{\Phi}} \mathsf{parseUPat}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow p_i : \tau_i \dashv\!\Vert \hat{\Gamma}_i\}_{0 \le i < n_{\mathrm{pat}}}$ by IH on (4) and (6)

(10) $\hat{\Gamma} = \biguplus_{0 \le i < n_{\mathrm{pat}}} \hat{\Gamma}_i$ by IH on (4) and (6)

The conclusions hold as follows:

1. (7)
2. (8)
3. (9)
4. (10)

Case (B.12e)**.**

(1) $\grave{p} = \mathtt{splicedp}[m; n; \grave{\tau}]$ by assumption

(2) $\emptyset \vdash^{\hat{\Delta}; b} \grave{\tau} \rightsquigarrow \tau \ \mathsf{type}$ by assumption

(3) $\mathsf{parseUPat}(\mathsf{subseq}(b; m; n)) = \hat{p}$ by assumption

(4) $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\Vert \hat{\Gamma}$ by assumption

(5) $\mathsf{summary}(\mathtt{splicedp}[m; n; \grave{\tau}]) = \mathsf{summary}(\grave{\tau}) \cup \{\mathtt{splicedp}[m; n; \grave{\tau}]\}$ by definition

(6) $\mathsf{summary}(\grave{\tau}) = \{\mathtt{splicedt}[m_i'; n_i']\}_{0 \le i < n_{\mathrm{ty}}}$ by definition

(7) $\{\langle \mathcal{D}; \Delta_{\mathrm{app}} \rangle \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \le i < n}$ by Lemma B.29 on (2) and (6)

The conclusions hold as follows:

1. (7)
2. (2)
3. (3) and (4)

4. This conclusion holds by (4) because $n_{\text{pat}} = 1$.

□

Theorem B.33 (spTSM Abstract Reasoning Principles). *If* $\hat{\Delta} \vdash_{\hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow p : \tau \dashv\!\mid \hat{\Gamma}$ *where* $\hat{\Delta} = \langle \mathcal{D}; \Delta \rangle$ *and* $\hat{\Gamma} = \langle \mathcal{G}; \Gamma \rangle$ *then all of the following hold:*

1. *(**Typing 1**)* $\hat{\Phi} = \hat{\Phi}', \hat{a} \rightsquigarrow a \hookrightarrow \texttt{sptsm}(\tau; e_{parse})$ *and* $\Delta \vdash p : \tau \dashv\!\mid \Gamma$
2. $b \downarrow_{\mathsf{Body}} e_{body}$
3. $e_{parse}(e_{body}) \Downarrow \texttt{inj}[\texttt{SuccessP}](e_{proto})$
4. $e_{proto} \uparrow_{\mathsf{PrPat}} \dot{p}$
5. *(**Segmentation**)* $\mathsf{seg}(\dot{p})$ segments b
6. $\mathsf{summary}(\dot{p}) = \{\texttt{splicedt}[n'_i; m'_i]\}_{0 \le i < n_{ty}} \cup \{\texttt{splicedp}[m_i; n_i; \dot{\tau}_i]\}_{0 \le i < n_{pat}}$
7. *(**Typing 2**)* $\{\hat{\Delta} \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow \tau'_i\ \mathsf{type}\}_{0 \le i < n_{ty}}$ *and* $\{\Delta \vdash \tau'_i\ \mathsf{type}\}_{0 \le i < n_{ty}}$
8. *(**Typing 3**)* $\{\varnothing \vdash^{\hat{\Delta}; b} \dot{\tau}_i \rightsquigarrow \tau_i\ \mathsf{type}\}_{0 \le i < n_{pat}}$ *and* $\{\Delta \vdash \tau_i\ \mathsf{type}\}_{0 \le i < n_{pat}}$
9. *(**Typing 4**)* $\{\hat{\Delta} \vdash_{\hat{\Phi}} \mathsf{parseUPat}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow p_i : \tau_i \dashv\!\mid \hat{\Gamma}_i\}_{0 \le i < n_{pat}}$ *and* $\{\Delta \vdash p_i : \tau_i \dashv\!\mid \Gamma_i\}_{0 \le i < n_{pat}}$
10. *(**No Hidden Bindings**)* $\hat{\Gamma} = \biguplus_{0 \le i < n_{pat}} \hat{\Gamma}_i$

Proof. By rule induction over Rules (B.8). There is only one rule that applies.

Case (B.8f)**.**

(1)	$\hat{\Delta} \vdash_{\hat{\Phi}} \hat{a}\ \text{`}b\text{`} \rightsquigarrow p : \tau \dashv\!\mid \hat{\Gamma}$	by assumption
(2)	$\hat{\Phi} = \hat{\Phi}', \hat{a} \rightsquigarrow a \hookrightarrow \texttt{sptsm}(\tau; e_{\text{parse}})$	by assumption
(3)	$\Delta \vdash p : \tau \dashv\!\mid \Gamma$	by Theorem B.26 on (1)
(4)	$b \downarrow_{\mathsf{Body}} e_{\text{body}}$	by assumption
(5)	$e_{\text{parse}}(e_{\text{body}}) \Downarrow \texttt{inj}[\texttt{SuccessP}](e_{\text{proto}})$	by assumption
(6)	$e_{\text{proto}} \uparrow_{\mathsf{PrPat}} \dot{p}$	by assumption
(7)	$\mathsf{seg}(\dot{p})$ segments b	by assumption
(8)	$\dot{p} \rightsquigarrow p : \tau \dashv\!\mid^{\hat{\Delta}; \hat{\Phi}; b} \hat{\Gamma}$	by assumption
(9)	$\mathsf{summary}(\dot{p}) = \{\texttt{splicedt}[m'_i; n'_i]\}_{0 \le i < n_{\text{ty}}} \cup \{\texttt{splicedp}[m_i; n_i; \}]_{0 \le i < n_{\text{pat}}}$	by definition

(10) $\{\hat{\Delta} \vdash \mathsf{parseUTyp}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow \tau_i' \ \mathsf{type}\}_{0 \leq i < n_{\mathrm{ty}}}$ by Lemma B.32 on (8) and (9)

(11) $\{\Delta \vdash \tau_i' \ \mathsf{type}\}_{0 \leq i < n_{\mathrm{ty}}}$ by Lemma B.24, part 1 over (10)

(12) $\{\emptyset \vdash^{\hat{\Delta}; b} \grave{\tau}_i \rightsquigarrow \tau_i \ \mathsf{type}\}_{0 \leq i < n_{\mathrm{pat}}}$ by Lemma B.32 on (8) and (9)

(13) $\{\Delta \vdash \tau_i \ \mathsf{type}\}_{0 \leq i < n_{\mathrm{pat}}}$ by Lemma B.24, part 2 over (12)

(14) $\{\hat{\Delta} \vdash_{\Phi} \mathsf{parseUPat}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow p_i : \tau_i \dashv\mid \hat{\Gamma}_i\}_{0 \leq i < n_{\mathrm{pat}}}$ by Lemma B.32 on (8) and (9)

(15) $\{\Delta \vdash p_i : \tau_i \dashv\mid \Gamma_i\}_{0 \leq i < n_{\mathrm{pat}}}$ by Theorem B.26 over (14)

(16) $\hat{\Gamma} = \biguplus_{0 \leq i < n_{\mathrm{pat}}} \hat{\Gamma}_i$ by Lemma B.32 on (8) and (9)

The conclusions hold as follows:

1. (2) and (3)
2. (4)
3. (5)
4. (6)
5. (7)
6. (9)
7. (10) and (11)
8. (12) and (13)
9. (14) and (15)
10. (16)

□

Appendix C

miniVerse$_{\mathbf{P}}$

C.1 Expanded Language (XL)

C.1.1 Syntax

Signatures and Module Expressions

Sort			Operational Form	Description
Sig	σ	$::=$	$\texttt{sig}\{\kappa\}(u.\tau)$	signature
Mod	M	$::=$	X	module variable
			$\texttt{struct}(c;e)$	structure
			$\texttt{seal}\{\sigma\}(M)$	seal
			$\texttt{mlet}\{\sigma\}(M;X.M)$	definition

Kinds and Constructions

Sort			Operational Form	Description
Kind	κ	$::=$	k	kind variable
			$\texttt{darr}(\kappa;u.\kappa)$	dependent function
			$\texttt{unit}$	nullary product
			$\texttt{dprod}(\kappa;u.\kappa)$	dependent product
			$\texttt{Type}$	type
			$\texttt{S}(\tau)$	singleton
Con	c,τ	$::=$	u	construction variable
			t	type variable
			$\texttt{abs}(u.c)$	abstraction
			$\texttt{app}(c;c)$	application
			$\texttt{triv}$	trivial
			$\texttt{pair}(c;c)$	pair
			$\texttt{prl}(c)$	left projection
			$\texttt{prr}(c)$	right projection
			$\texttt{parr}(\tau;\tau)$	partial function
			$\texttt{all}\{\kappa\}(u.\tau)$	polymorphic
			$\texttt{rec}(t.\tau)$	recursive
			$\texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i\in L})$	labeled product
			$\texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i\in L})$	labeled sum
			$\texttt{con}(M)$	construction component

Expressions, Rules and Patterns

Sort			Operational Form	Description
Exp	e	$::=$	x	variable
			$\texttt{lam}\{\tau\}(x.e)$	abstraction
			$\texttt{ap}(e;e)$	application
			$\texttt{clam}\{\kappa\}(u.e)$	construction abstraction
			$\texttt{cap}\{\kappa\}(e)$	construction application
			$\texttt{fold}(e)$	fold
			$\texttt{unfold}(e)$	unfold
			$\texttt{tpl}[L](\{i \hookrightarrow e_i\}_{i\in L})$	labeled tuple
			$\texttt{prj}[\ell](e)$	projection
			$\texttt{inj}[\ell](e)$	injection
			$\texttt{match}[n](e;\{r_i\}_{1\leq i\leq n})$	match
			$\texttt{val}(M)$	value component
Rule	r	$::=$	$\texttt{rule}(p.e)$	rule
Pat	p	$::=$	x	variable pattern
			$\texttt{wildp}$	wildcard pattern
			$\texttt{foldp}(p)$	fold pattern
			$\texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i\in L})$	labeled tuple pattern
			$\texttt{injp}[\ell](p)$	injection pattern

C.1.2 Statics

Unified Contexts

A *unified context*, Ω, is an ordered finite function. We write

- $\Omega, X : \sigma$ when $X \notin \mathrm{dom}(\Omega)$ for the extension of Ω with a mapping from X to the hypothesis $X : \sigma$.
- $\Omega, x : \tau$ when $x \notin \mathrm{dom}(\Omega)$ for the extension of Ω with a mapping from x to the hypothesis $x : \tau$.
- $\Omega, u :: \kappa$ when $u \notin \mathrm{dom}(\Omega)$ for the extension of Ω with a mapping from u to the hypothesis $u :: \kappa$.

Signatures and Structures

$\boxed{\Omega \vdash \sigma\ \mathsf{sig}}$ σ is a signature

$$\frac{\Omega \vdash \kappa\ \mathsf{kind} \qquad \Omega, u :: \kappa \vdash \tau :: \mathtt{Type}}{\Omega \vdash \texttt{sig}\{\kappa\}(u.\tau)\ \mathsf{sig}} \tag{C.1}$$

$\boxed{\Omega \vdash \sigma \equiv \sigma'}$ σ and σ' are definitionally equal

$$\frac{\Omega \vdash \kappa \equiv \kappa' \qquad \Omega, u :: \kappa \vdash \tau \equiv \tau' :: \mathtt{Type}}{\Omega \vdash \mathtt{sig}\{\kappa\}(u.\tau) \equiv \mathtt{sig}\{\kappa'\}(u.\tau')} \tag{C.2}$$

$\boxed{\Omega \vdash \sigma <: \sigma'}$ σ is a subsignature of σ'

$$\frac{\Omega \vdash \kappa <:: \kappa' \qquad \Omega, u :: \kappa \vdash \tau <: \tau'}{\Omega \vdash \mathtt{sig}\{\kappa\}(u.\tau) <: \mathtt{sig}\{\kappa'\}(u.\tau')} \tag{C.3}$$

$\boxed{\Omega \vdash M : \sigma}$ M matches σ

$$\frac{\Omega \vdash M : \sigma \qquad \Omega \vdash \sigma <: \sigma'}{\Omega \vdash M : \sigma'} \tag{C.4a}$$

$$\frac{}{\Omega, X : \sigma \vdash X : \sigma} \tag{C.4b}$$

$$\frac{\Omega \vdash c :: \kappa \qquad \Omega \vdash e : [c/u]\tau}{\Omega \vdash \mathtt{struct}(c; e) : \mathtt{sig}\{\kappa\}(u.\tau)} \tag{C.4c}$$

$$\frac{\Omega \vdash \sigma\ \mathsf{sig} \qquad \Omega \vdash M : \sigma}{\Omega \vdash \mathtt{seal}\{\sigma\}(M) : \sigma} \tag{C.4d}$$

$$\frac{\Omega \vdash M : \sigma \qquad \Omega \vdash \sigma'\ \mathsf{sig} \qquad \Omega, X : \sigma \vdash M' : \sigma'}{\Omega \vdash \mathtt{mlet}\{\sigma'\}(M; X.M') : \sigma'} \tag{C.4e}$$

$\boxed{\Omega \vdash M\ \mathsf{mval}}$ M is, or stands for, a module value

$$\frac{}{\Omega \vdash \mathtt{struct}(c; e)\ \mathsf{mval}} \tag{C.5a}$$

$$\frac{}{\Omega, X : \sigma \vdash X\ \mathsf{mval}} \tag{C.5b}$$

Kinds and Constructions

$\boxed{\Omega \vdash \kappa\ \mathsf{kind}}$ κ is a kind

$$\frac{\Omega \vdash \kappa_1\ \mathsf{kind} \qquad \Omega, u :: \kappa_1 \vdash \kappa_2\ \mathsf{kind}}{\Omega \vdash \mathtt{darr}(\kappa_1; u.\kappa_2)\ \mathsf{kind}} \tag{C.6a}$$

$$\frac{}{\Omega \vdash \mathtt{unit}\ \mathsf{kind}} \tag{C.6b}$$

$$\frac{\Omega \vdash \kappa_1 \text{ kind} \qquad \Omega, u :: \kappa_1 \vdash \kappa_2 \text{ kind}}{\Omega \vdash \texttt{dprod}(\kappa_1; u.\kappa_2) \text{ kind}} \tag{C.6c}$$

$$\frac{}{\Omega \vdash \texttt{Type kind}} \tag{C.6d}$$

$$\frac{\Omega \vdash \tau :: \texttt{Type}}{\Omega \vdash \texttt{S}(\tau) \text{ kind}} \tag{C.6e}$$

$\boxed{\Omega \vdash \kappa \equiv \kappa'}$ κ and κ' are definitionally equal

$$\frac{\Omega \vdash \kappa \text{ kind}}{\Omega \vdash \kappa \equiv \kappa} \tag{C.7a}$$

$$\frac{\Omega \vdash \kappa \equiv \kappa'}{\Omega \vdash \kappa' \equiv \kappa} \tag{C.7b}$$

$$\frac{\Omega \vdash \kappa \equiv \kappa' \qquad \Omega \vdash \kappa' \equiv \kappa''}{\Omega \vdash \kappa \equiv \kappa''} \tag{C.7c}$$

$$\frac{\Omega \vdash \kappa_1 \equiv \kappa_1' \qquad \Omega, u :: \kappa_1 \vdash \kappa_2 \equiv \kappa_2'}{\Omega \vdash \texttt{darr}(\kappa_1; u.\kappa_2) \equiv \texttt{darr}(\kappa_1'; u.\kappa_2')} \tag{C.7d}$$

$$\frac{\Omega \vdash \kappa_1 \equiv \kappa_1' \qquad \Omega, u :: \kappa_1 \vdash \kappa_2 \equiv \kappa_2'}{\Omega \vdash \texttt{dprod}(\kappa_1; u.\kappa_2) \equiv \texttt{dprod}(\kappa_1'; u.\kappa_2')} \tag{C.7e}$$

$$\frac{\Omega \vdash c \equiv c' :: \texttt{Type}}{\Omega \vdash \texttt{S}(c) \equiv \texttt{S}(c')} \tag{C.7f}$$

$\boxed{\Omega \vdash \kappa <:: \kappa'}$ κ is a subkind of κ'

$$\frac{\Omega \vdash \kappa \equiv \kappa'}{\Omega \vdash \kappa <:: \kappa'} \tag{C.8a}$$

$$\frac{\Omega \vdash \kappa <:: \kappa' \qquad \Omega \vdash \kappa' <:: \kappa''}{\Omega \vdash \kappa <:: \kappa''} \tag{C.8b}$$

$$\frac{\Omega \vdash \kappa_1' <:: \kappa_1 \qquad \Omega, u :: \kappa_1' \vdash \kappa_2 <:: \kappa_2'}{\Omega \vdash \texttt{darr}(\kappa_1; u.\kappa_2) <:: \texttt{darr}(\kappa_1'; u.\kappa_2')} \tag{C.8c}$$

$$\frac{\Omega \vdash \kappa_1 <:: \kappa_1' \qquad \Omega, u :: \kappa_1 \vdash \kappa_2 <:: \kappa_2'}{\Omega \vdash \texttt{dprod}(\kappa_1; u.\kappa_2) <:: \texttt{dprod}(\kappa_1'; u.\kappa_2')} \tag{C.8d}$$

$$\frac{\Omega \vdash \tau :: \texttt{Type}}{\Omega \vdash \texttt{S}(\tau) <:: \texttt{Type}} \tag{C.8e}$$

$$\frac{\Omega \vdash \tau <: \tau'}{\Omega \vdash \mathtt{S}(\tau) <:: \mathtt{S}(\tau')} \tag{C.8f}$$

$\boxed{\Omega \vdash c :: \kappa}$ c has kind κ

$$\frac{\Omega \vdash c :: \kappa_1 \qquad \Omega \vdash \kappa_1 <:: \kappa_2}{\Omega \vdash c :: \kappa_2} \tag{C.9a}$$

$$\frac{}{\Omega, u :: \kappa \vdash u :: \kappa} \tag{C.9b}$$

$$\frac{\Omega, u :: \kappa_1 \vdash c_2 :: \kappa_2}{\Omega \vdash \mathtt{abs}(u.c_2) :: \mathtt{darr}(\kappa_1; u.\kappa_2)} \tag{C.9c}$$

$$\frac{\Omega \vdash c_1 :: \mathtt{darr}(\kappa_2; u.\kappa) \qquad \Omega \vdash c_2 :: \kappa_2}{\Omega \vdash \mathtt{app}(c_1; c_2) :: [c_1/u]\kappa} \tag{C.9d}$$

$$\frac{}{\Omega \vdash \mathtt{triv} :: \mathtt{unit}} \tag{C.9e}$$

$$\frac{\Omega \vdash c_1 :: \kappa_1 \qquad \Omega \vdash c_2 :: [c_1/u]\kappa_2}{\Omega \vdash \mathtt{pair}(c_1; c_2) :: \mathtt{dprod}(\kappa_1; u.\kappa_2)} \tag{C.9f}$$

$$\frac{\Omega \vdash c :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\Omega \vdash \mathtt{prl}(c) :: \kappa_1} \tag{C.9g}$$

$$\frac{\Omega \vdash c :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\Omega \vdash \mathtt{prr}(c) :: [\mathtt{prl}(c)/u]\kappa_2} \tag{C.9h}$$

$$\frac{\Omega \vdash \tau_1 :: \mathtt{Type} \qquad \Omega \vdash \tau_2 :: \mathtt{Type}}{\Omega \vdash \mathtt{parr}(\tau_1; \tau_2) :: \mathtt{Type}} \tag{C.9i}$$

$$\frac{\Omega \vdash \kappa \ \mathsf{kind} \qquad \Omega, u :: \kappa \vdash \tau :: \mathtt{Type}}{\Omega \vdash \mathtt{all}\{\kappa\}(u.\tau) :: \mathtt{Type}} \tag{C.9j}$$

$$\frac{\Omega, t :: \mathtt{Type} \vdash \tau :: \mathtt{Type}}{\Omega \vdash \mathtt{rec}(t.\tau) :: \mathtt{Type}} \tag{C.9k}$$

$$\frac{\{\Omega \vdash \tau_i :: \mathtt{Type}\}_{1 \le i \le n}}{\Omega \vdash \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) :: \mathtt{Type}} \tag{C.9l}$$

$$\frac{\{\Omega \vdash \tau_i :: \mathtt{Type}\}_{1 \le i \le n}}{\Omega \vdash \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) :: \mathtt{Type}} \tag{C.9m}$$

$$\frac{\Omega \vdash c :: \mathtt{Type}}{\Omega \vdash c :: \mathtt{S}(c)} \tag{C.9n}$$

$$\frac{\Omega \vdash M \ \mathsf{mval} \qquad \Omega \vdash M : \mathtt{sig}\{\kappa\}(u.\tau)}{\Omega \vdash \mathtt{con}(M) :: \kappa} \tag{C.9o}$$

$\boxed{\Omega \vdash c \equiv c' :: \kappa}$ c and c' are definitionally equal as constructions of kind κ

$$\frac{\Omega \vdash c :: \kappa}{\Omega \vdash c \equiv c :: \kappa} \tag{C.10a}$$

$$\frac{\Omega \vdash c \equiv c' :: \kappa}{\Omega \vdash c' \equiv c :: \kappa} \tag{C.10b}$$

$$\frac{\Omega \vdash c \equiv c' :: \kappa \qquad \Omega \vdash c' \equiv c'' :: \kappa}{\Omega \vdash c \equiv c'' :: \kappa} \tag{C.10c}$$

$$\frac{\Omega, u :: \kappa_1 \vdash c \equiv c' :: \kappa_2}{\Omega \vdash \mathtt{abs}(u.c) \equiv \mathtt{abs}(u.c') :: \mathtt{darr}(\kappa_1; u.\kappa_2)} \tag{C.10d}$$

$$\frac{\Omega \vdash c_1 \equiv c_1' :: \mathtt{darr}(\kappa_2; u.\kappa) \qquad \Omega \vdash c_2 \equiv c_2' :: \kappa_2}{\Omega \vdash \mathtt{app}(c_1; c_2) \equiv \mathtt{app}(c_1'; c_2') :: \kappa} \tag{C.10e}$$

$$\frac{\Omega \vdash \mathtt{abs}(u.c) :: \mathtt{darr}(\kappa_2; u.\kappa) \qquad \Omega \vdash c_2 :: \kappa_2}{\Omega \vdash \mathtt{app}(\mathtt{abs}(u.c); c_2) \equiv [c_2/u]c :: [c_2/u]\kappa} \tag{C.10f}$$

$$\frac{\Omega \vdash c_1 \equiv c_1' :: \kappa_1 \qquad \Omega \vdash c_2 \equiv c_2' :: [c_1/u]\kappa_2}{\Omega \vdash \mathtt{pair}(c_1; c_2) \equiv \mathtt{pair}(c_1'; c_2') :: \mathtt{dprod}(\kappa_1; u.\kappa_2)} \tag{C.10g}$$

$$\frac{\Omega \vdash c \equiv c' :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\Omega \vdash \mathtt{prl}(c) \equiv \mathtt{prl}(c') :: \kappa_1} \tag{C.10h}$$

$$\frac{\Omega \vdash c_1 :: \kappa_1 \qquad \Omega \vdash c_2 :: \kappa_2}{\Omega \vdash \mathtt{prl}(\mathtt{pair}(c_1; c_2)) \equiv c_1 :: \kappa_1} \tag{C.10i}$$

$$\frac{\Omega \vdash c \equiv c' :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\Omega \vdash \mathtt{prr}(c) \equiv \mathtt{prr}(c') :: [\mathtt{prl}(c)/u]\kappa_2} \tag{C.10j}$$

$$\frac{\Omega \vdash c_1 :: \kappa_1 \qquad \Omega \vdash c_2 :: \kappa_2}{\Omega \vdash \mathtt{prr}(\mathtt{pair}(c_1; c_2)) \equiv c_2 :: \kappa_2} \tag{C.10k}$$

$$\frac{\Omega \vdash \tau_1 \equiv \tau_1' :: \mathtt{Type} \qquad \Omega \vdash \tau_2 \equiv \tau_2' :: \mathtt{Type}}{\Omega \vdash \mathtt{parr}(\tau_1; \tau_2) \equiv \mathtt{parr}(\tau_1'; \tau_2') :: \mathtt{Type}} \tag{C.10l}$$

$$\frac{\Omega \vdash \kappa \equiv \kappa' \qquad \Omega, u :: \kappa \vdash \tau \equiv \tau' :: \mathtt{Type}}{\Omega \vdash \mathtt{all}\{\kappa\}(u.\tau) \equiv \mathtt{all}\{\kappa'\}(u.\tau') :: \mathtt{Type}} \tag{C.10m}$$

$$\frac{\Omega, t :: \mathtt{Type} \vdash \tau \equiv \tau' :: \mathtt{Type}}{\Omega \vdash \mathtt{rec}(t.\tau) \equiv \mathtt{rec}(t.\tau') :: \mathtt{Type}} \tag{C.10n}$$

$$\frac{\{\Omega \vdash \tau_i \equiv \tau_i' :: \mathtt{Type}\}_{1 \le i \le n}}{\Omega \vdash \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \equiv \mathtt{prod}[L](\{i \hookrightarrow \tau_i'\}_{i \in L}) :: \mathtt{Type}} \tag{C.10o}$$

$$\frac{\{\Omega \vdash \tau_i \equiv \tau_i' :: \mathtt{Type}\}_{1 \le i \le n}}{\Omega \vdash \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \equiv \mathtt{sum}[L](\{i \hookrightarrow \tau_i'\}_{i \in L}) :: \mathtt{Type}} \tag{C.10p}$$

$$\frac{\Omega \vdash c :: \mathtt{S}(c')}{\Omega \vdash c \equiv c' :: \mathtt{Type}} \tag{C.10q}$$

$$\frac{\Omega \vdash \mathtt{struct}(c; e) : \mathtt{sig}\{\kappa\}(u.\tau)}{\Omega \vdash \mathtt{con}(\mathtt{struct}(c; e)) \equiv c :: \kappa} \tag{C.10r}$$

Expressions, Rules and Patterns

$\boxed{\Omega \vdash \tau <: \tau'}$ τ is a subtype of τ'

$$\frac{\Omega \vdash \tau_1 \equiv \tau_2 :: \mathtt{Type}}{\Omega \vdash \tau_1 <: \tau_2} \tag{C.11a}$$

$$\frac{\Omega \vdash \tau <: \tau' \qquad \Omega \vdash \tau' <: \tau''}{\Omega \vdash \tau <: \tau''} \tag{C.11b}$$

$$\frac{\Omega \vdash \tau_1' <: \tau_1 \qquad \Omega \vdash \tau_2 <: \tau_2'}{\Omega \vdash \mathtt{parr}(\tau_1; \tau_2) <: \mathtt{parr}(\tau_1'; \tau_2')} \tag{C.11c}$$

$$\frac{\Omega \vdash \kappa' <:: \kappa \qquad \Omega, u :: \kappa' \vdash \tau <: \tau'}{\Omega \vdash \mathtt{all}\{\kappa\}(u.\tau) <: \mathtt{all}\{\kappa'\}(u.\tau')} \tag{C.11d}$$

$$\frac{\{\Omega \vdash \tau_i <: \tau_i'\}_{i \in L}}{\Omega \vdash \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) <: \mathtt{prod}[L](\{i \hookrightarrow \tau_i'\}_{i \in L})} \tag{C.11e}$$

$$\frac{\{\Omega \vdash \tau_i <: \tau_i'\}_{i \in L}}{\Omega \vdash \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) <: \mathtt{sum}[L](\{i \hookrightarrow \tau_i'\}_{i \in L})} \tag{C.11f}$$

$\boxed{\Omega \vdash e : \tau}$ e has type τ

$$\frac{\Omega \vdash e : \tau \qquad \Omega \vdash \tau <: \tau'}{\Omega \vdash e : \tau'} \tag{C.12a}$$

$$\frac{}{\Omega, x : \tau \vdash x : \tau} \tag{C.12b}$$

$$\frac{\Omega \vdash \tau :: \mathtt{Type} \qquad \Omega, x : \tau \vdash e : \tau'}{\Omega \vdash \mathtt{lam}\{\tau\}(x.e) : \mathtt{parr}(\tau; \tau')} \tag{C.12c}$$

$$\frac{\Omega \vdash e_1 : \mathtt{parr}(\tau; \tau') \qquad \Omega \vdash e_2 : \tau}{\Omega \vdash \mathtt{ap}(e_1; e_2) : \tau'} \tag{C.12d}$$

$$\frac{\Omega \vdash \kappa\ \mathsf{kind} \qquad \Omega, u :: \kappa \vdash e : \tau}{\Omega \vdash \mathtt{clam\{\kappa\}(u.e)} : \mathtt{all\{\kappa\}(u.\tau)}} \tag{C.12e}$$

$$\frac{\Omega \vdash e : \mathtt{all\{\kappa\}(u.\tau)} \qquad \Omega \vdash c :: \kappa}{\Omega \vdash \mathtt{cap\{c\}(e)} : [c/u]\tau} \tag{C.12f}$$

$$\frac{\Omega \vdash e : [\mathtt{rec}(t.\tau)/t]\tau}{\Omega \vdash \mathtt{fold}(e) : \mathtt{rec}(t.\tau)} \tag{C.12g}$$

$$\frac{\Omega \vdash e : \mathtt{rec}(t.\tau)}{\Omega \vdash \mathtt{unfold}(e) : [\mathtt{rec}(t.\tau)/t]\tau} \tag{C.12h}$$

$$\frac{\{\Omega \vdash e_i : \tau_i\}_{i \in L}}{\Omega \vdash \mathtt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L}) : \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})} \tag{C.12i}$$

$$\frac{\Omega \vdash e : \mathtt{prod}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau)}{\Omega \vdash \mathtt{prj}[\ell](e) : \tau} \tag{C.12j}$$

$$\frac{\Omega \vdash e : \tau}{\Omega \vdash \mathtt{inj}[\ell](e) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau)} \tag{C.12k}$$

$$\frac{\Omega \vdash e : \tau \qquad \{\Omega \vdash r_i : \tau \Mapsto \tau'\}_{1 \le i \le n}}{\Omega \vdash \mathtt{match}[n](e; \{r_i\}_{1 \le i \le n}) : \tau'} \tag{C.12l}$$

$$\frac{\Omega \vdash M\ \mathsf{mval} \qquad \Omega \vdash M : \mathtt{sig\{\kappa\}(u.\tau)}}{\Omega \vdash \mathtt{val}(M) : [\mathtt{con}(M)/u]\tau} \tag{C.12m}$$

$\boxed{\Omega \vdash r : \tau \Mapsto \tau'}$ r takes values of type τ to values of type τ'

$$\frac{\Omega \vdash p : \tau \dashv\mid \Omega' \qquad \Omega \cup \Omega' \vdash e : \tau'}{\Omega \vdash \mathtt{rule}(p.e) : \tau \Mapsto \tau'} \tag{C.13}$$

$\boxed{\Omega \vdash p : \tau \dashv\mid \Omega'}$ p matches values of type τ generating hypotheses Ω'

$$\frac{\Omega \vdash p : \tau \dashv\mid \Omega' \qquad \Omega \vdash \tau <: \tau'}{\Omega \vdash p : \tau' \dashv\mid \Omega'} \tag{C.14a}$$

$$\frac{}{\Omega \vdash x : \tau \dashv\mid x : \tau} \tag{C.14b}$$

$$\frac{}{\Omega \vdash \mathtt{wildp} : \tau \dashv\mid \emptyset} \tag{C.14c}$$

$$\frac{\Omega \vdash p : [\mathtt{rec}(t.\tau)/t]\tau \dashv\mid \Omega'}{\Omega \vdash \mathtt{foldp}(p) : \mathtt{rec}(t.\tau) \dashv\mid \Omega'} \tag{C.14d}$$

$$\frac{\{\Omega \vdash p_i : \tau_i \dashv\Vdash \Omega_i\}_{i \in L}}{\Omega \vdash \mathtt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) : \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \dashv\Vdash \cup_{i \in L} \Omega_i} \tag{C.14e}$$

$$\frac{\Omega \vdash p : \tau \dashv\Vdash \Omega'}{\Omega \vdash \mathtt{injp}[\ell](p) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau) \dashv\Vdash \Omega'} \tag{C.14f}$$

Metatheory

The rules above are syntax-directed, so we assume an inversion lemma for each rule as needed without stating it separately or proving it explicitly. The following standard lemmas also hold, for all basic judgements J above.

Lemma C.1 (Weakening). *If $\Omega \vdash J$ then $\Omega \cup \Omega' \vdash J$.*

Proof Sketch. By straightforward mutual rule induction. □

Definition C.2. *A* substitution, *ω, is a finite function that maps:*

- *each $X \in dom(\omega)$ to a module expression subtitution, M/X;*
- *each $u \in dom(\omega)$ to a construction substitution, c/u; and*
- *each $x \in dom(\omega)$ to an expression substitution, e/x.*

We write $\Omega \vdash \omega : \Omega'$ iff $dom(\omega) = dom(\Omega')$ and:

- *for each $M/X \in \omega$, we have $X : \sigma \in \Omega'$ and $\Omega \vdash M : \sigma$ and $\Omega \vdash M$ mval; and*
- *for each $c/u \in \omega$, we have $u :: \kappa \in \Omega'$ and $\Omega \vdash c :: \kappa$; and*
- *for each $e/x \in \omega$, we have $x : \tau \in \Omega'$ and $\Omega \vdash e : \tau$.*

We simultaneously apply a substitution by placing it in prefix position. For example, $[\omega]e$ applies the substitutions ω simultaneously to e.

Lemma C.3 (Substitution). *If $\Omega \cup \Omega' \cup \Omega'' \vdash J$ and $\Omega \vdash \omega : \Omega'$ then $\Omega \cup [\omega]\Omega'' \vdash [\omega]J$.*

Proof Sketch. By straightforward rule induction. □

Lemma C.4 (Decomposition). *If $\Omega \cup [\omega]\Omega'' \vdash [\omega]J$ and $\Omega \vdash \omega : \Omega'$ then $\Omega \cup \Omega' \cup \Omega'' \vdash J$.*

Proof Sketch. By straightforward rule induction. □

C.1.3 Structural Dynamics

The structural dynamics of modules is defined as a transition system, and is organized around judgements of the following form:

Judgement Form	Description
$M \mapsto M'$	M transitions to M'
M val	M is a module value
M matchfail	M raises match failure

The structural dynamics of expressions is also defined as a transition system, and is organized around judgements of the following form:

Judgement Form	Description
$e \mapsto e'$	e transitions to e'
e val	e is a value
e matchfail	e raises match failure

We also define auxiliary judgements for *iterated transition*, $e \mapsto^* e'$, and *evaluation*, $e \Downarrow e'$ of expressions.

Definition C.5 (Iterated Transition)**.** *Iterated transition, $e \mapsto^* e'$, is the reflexive, transitive closure of the transition judgement, $e \mapsto e'$.*

Definition C.6 (Evaluation)**.** *$e \Downarrow e'$ iff $e \mapsto^* e'$ and e'* val*.*

Similarly, we lift these definitions to the level of module expressions as well.

Definition C.7 (Iterated Module Transition)**.** *Iterated transition, $M \mapsto^* M'$, is the reflexive, transitive closure of the transition judgement, $M \mapsto M'$.*

Definition C.8 (Module Evaluation)**.** *$M \Downarrow M'$ iff $M \mapsto^* M'$ and M'* val*.*

As in miniVerse$_\mathsf{S}$, our subsequent developments do not make mention of particular rules in the dynamics, nor do they make mention of other judgements, not listed above, that are used only for defining the dynamics of the match operator, so we do not produce these details here. Instead, it suffices to state the following conditions.

The Preservation condition ensures that evaluation preserves typing.

Condition C.9 (Preservation)**.**

1. *If $\vdash M : \sigma$ and $M \mapsto M'$ then $\vdash M : \sigma$.*
2. *If $\vdash e : \tau$ and $e \mapsto e'$ then $\vdash e' : \tau$.*

The Progress condition ensures that evaluation of a well-typed expanded expression cannot "get stuck". We must consider the possibility of match failure in this condition.

Condition C.10 (Progress)**.**

1. *If $\vdash M : \sigma$ then either M* val *or M* matchfail *or there exists an M' such that $M \mapsto M'$.*
2. *If $\vdash e : \tau$ then either e* val *or e* matchfail *or there exists an e' such that $e \mapsto e'$.*

C.2 Unexpanded Language (UL)

C.2.1 Syntax

Stylized Syntax – Unexpanded Signatures and Modules

Sort			Stylized Form	Description
USig	$\hat{\sigma}$	::=	$[\![\hat{u} :: \hat{\kappa}; \hat{\tau}]\!]$	signature
UMod	$\hat{M}$	::=	$\hat{X}$	module identifier
			$[\![\hat{c}; \hat{e}]\!]$	structure
			$\hat{M} \upharpoonright \hat{\sigma}$	seal
			$(\texttt{let}\ \hat{X} = \hat{M}\ \texttt{in}\ \hat{M}) : \hat{\sigma}$	definition
			$\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\rho}\ \texttt{for expressions by static}\ e\ \texttt{in}\ \hat{M}$	peTSM definition
			$\texttt{let syntax}\ \hat{a} = \hat{e}\ \texttt{for expressions in}\ \hat{M}$	peTSM binding
			$\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\rho}\ \texttt{for patterns by static}\ e\ \texttt{in}\ \hat{M}$	ppTSM definition
			$\texttt{let syntax}\ \hat{a} = \hat{e}\ \texttt{for patterns in}\ \hat{M}$	ppTSM binding

Stylized Syntax – Unexpanded Kinds and Constructions

Sort			Stylized Form	Description
UKind	$\hat{\kappa}$	::=	$(\hat{u} :: \hat{\kappa}) \to \hat{\kappa}$	dependent function
			$\langle\!\langle\rangle\!\rangle$	nullary product
			$(\hat{u} :: \hat{\kappa}) \times \hat{\kappa}$	dependent product
			$\mathtt{T}$	type
			$[=\hat{\tau}]$	singleton
UCon	$\hat{c}, \hat{\tau}$	::=	$\hat{u}$	construction identifier
			$\hat{t}$	
			$\hat{c} :: \hat{\kappa}$	ascription
			$\lambda \hat{u}.\hat{c}$	abstraction
			$c(c)$	application
			$\langle\!\langle\rangle\!\rangle$	trivial
			$\langle\!\langle \hat{c}, \hat{c} \rangle\!\rangle$	pair
			$\hat{c} \cdot \mathtt{l}$	left projection
			$\hat{c} \cdot \mathtt{r}$	right projection
			$\hat{\tau} \rightharpoonup \hat{\tau}$	partial function
			$\forall(\hat{u} :: \hat{\kappa}).\hat{\tau}$	polymorphic
			$\mu \hat{t}.\hat{\tau}$	recursive
			$\langle \{i \hookrightarrow \hat{\tau}_i\}_{i \in L} \rangle$	labeled product
			$[\{i \hookrightarrow \hat{\tau}_i\}_{i \in L}]$	labeled sum
			$\hat{X} \cdot \mathtt{c}$	construction component

Stylized Syntax – Unexpanded Expressions, Rules and Patterns

Sort			Stylized Form	Description
UExp	$\hat{e}$	::=	$\hat{x}$	identifier
			$\hat{e} : \hat{\tau}$	ascription
			$\texttt{let val}\ \hat{x} = \hat{e}\ \texttt{in}\ \hat{e}$	value binding
			$\lambda\hat{x}{:}\hat{\tau}.\hat{e}$	abstraction
			$\hat{e}(\hat{e})$	application
			$\Lambda\hat{u}{::}\hat{\kappa}.\hat{e}$	construction abstraction
			$\hat{e}[\hat{c}]$	construction application
			$\texttt{fold}(\hat{e})$	fold
			$\texttt{unfold}(\hat{e})$	unfold
			$\langle\{i \hookrightarrow \hat{e}_i\}_{i\in L}\rangle$	labeled tuple
			$\hat{e} \cdot \ell$	projection
			$\texttt{inj}[\ell](\hat{e})$	injection
			$\texttt{match}\ \hat{e}\ \{\hat{r}_i\}_{1\leq i\leq n}$	match
			$\hat{X} \cdot \texttt{v}$	value component
			$\hat{e}$ ‘b‘	peTSM application
URule	$\hat{r}$	::=	$\hat{p} \Rightarrow \hat{e}$	match rule
UPat	$\hat{p}$	::=	$\hat{x}$	identifier pattern
			_	wildcard pattern
			$\texttt{fold}(\hat{p})$	fold pattern
			$\langle\{i \hookrightarrow \hat{p}_i\}_{i\in L}\rangle$	labeled tuple pattern
			$\texttt{inj}[\ell](\hat{p})$	injection pattern
			$\hat{e}$ ‘b‘	ppTSM application

Stylized Syntax – Unexpanded TSM Types and Expressions

Sort			Stylized Form	Description
UMType	$\hat{\rho}$	::=	$\hat{\tau}$	type annotation
			$\forall\hat{X}{:}\hat{\sigma}.\hat{\rho}$	module parameterization
UMExp	$\hat{\epsilon}$	::=	$\hat{a}$	TSM identifier reference
			$\Lambda\hat{X}{:}\hat{\sigma}.\hat{\epsilon}$	module abstraction
			$\hat{\epsilon}(\hat{X})$	module application

Stylized Syntax – TSM Types and Expressions

Sort			Operational Form	Description
MType	ρ	::=	$\texttt{type}(\tau)$	type annotation
			$\texttt{allmods}\{\sigma\}(X.\rho)$	module parameterization
MExp	ϵ	::=	$\texttt{defref}[a]$	TSM definition reference
			$\texttt{absmod}\{\sigma\}(X.\epsilon)$	module abstraction
			$\texttt{apmod}\{M\}(\epsilon)$	module application

Body Lengths

We write $\|b\|$ for the length of b. The metafunction $\|\hat{M}\|$ computes the sum of the lengths of expression literal bodies in $\hat{M}$:

$$\begin{array}{ll}
\|\hat{X}\| & = 0 \\
\|[\![\hat{c}; \hat{e}]\!]\| & = \|\hat{e}\| \\
\|\hat{M} \upharpoonright \hat{\sigma}\| & = \|\hat{M}\| \\
\|(\texttt{let}\ \hat{X} = \hat{M}\ \texttt{in}\ \hat{M}') : \hat{\sigma}\| & = \|\hat{M}\| + \|\hat{M}'\| \\
\|\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\rho}\ \texttt{for expressions by static}\ e\ \texttt{in}\ \hat{M}\| & = \|\hat{M}\| \\
\|\texttt{let syntax}\ \hat{a} = \hat{\epsilon}\ \texttt{for expressions in}\ \hat{M}\| & = \|\hat{M}\| \\
\|\texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\rho}\ \texttt{for patterns by static}\ e\ \texttt{in}\ \hat{M}\| & = \|\hat{M}\| \\
\|\texttt{let syntax}\ \hat{a} = \hat{\epsilon}\ \texttt{for patterns in}\ \hat{M}\| & = \|\hat{M}\|
\end{array}$$

and $\|\hat{e}\|$ computes the sum of the lengths of expression literal bodies in $\hat{e}$:

$$\begin{array}{ll}
\|\hat{x}\| & = 0 \\
\|\lambda\hat{x}{:}\hat{\tau}.\hat{e}\| & = \|\hat{e}\| \\
\|\hat{e}_1(\hat{e}_2)\| & = \|\hat{e}_1\| + \|\hat{e}_2\| \\
\|\Lambda\hat{u}{::}\hat{\kappa}.\hat{e}\| & = \|\hat{e}\| \\
\|\hat{e}[\hat{c}]\| & = \|\hat{e}\| \\
\|\texttt{fold}(\hat{e})\| & = \|\hat{e}\| \\
\|\texttt{unfold}(\hat{e})\| & = \|\hat{e}\| \\
\|\langle\{i \hookrightarrow \hat{e}_i\}_{i \in L}\rangle\| & = \textstyle\sum_{i \in L} \|\hat{e}_i\| \\
\|\ell \cdot \hat{e}\| & = \|\hat{e}\| \\
\|\texttt{inj}[\ell](\hat{e})\| & = \|\hat{e}\| \\
\|\texttt{match}\ \hat{e}\ \{\hat{r}_i\}_{1 \leq i \leq n}\| & = \|\hat{e}\| + \textstyle\sum_{1 \leq i \leq n} \|r_i\| \\
\|\hat{X} \cdot \texttt{v}\| & = 0 \\
\|\hat{\epsilon}\ \texttt{`}b\texttt{`}\| & = \|b\|
\end{array}$$

and $\|\hat{r}\|$ computes the sum of the lengths of expression literal bodies in $\hat{r}$:

$$\|\hat{p} \Rightarrow \hat{e}\| = \|\hat{e}\|$$

Similarly, the metafunction $\|\hat{p}\|$ computes the sum of the lengths of the pattern literal bodies in $\hat{p}$:

$$\begin{array}{rl}
\|\hat{x}\| & = 0 \\
\|\texttt{fold}(\hat{p})\| & = \|\hat{p}\| \\
\|\langle\{i \hookrightarrow \hat{p}_i\}_{i \in L}\rangle\| & = \displaystyle\sum_{i \in L} \|\hat{p}_i\| \\
\|\texttt{inj}[\ell](\hat{p})\| & = \|\hat{p}\| \\
\|\hat{\epsilon}\ \texttt{`}b\texttt{`}\| & = \|b\|
\end{array}$$

Common Unexpanded Forms

Each expanded form, with a few minor exceptions noted below, maps onto an unexpanded form. We refer to these as the *common forms*. In particular:

- Each module variable, X, maps onto a unique module identifier, written $\widehat{X}$.
- Each signature, σ, maps onto an unexpanded signature, $\mathcal{U}(\sigma)$, as follows:

$$\mathcal{U}(\texttt{sig\{}\kappa\texttt{\}(}u.c\texttt{)}) = [\![\widehat{u} :: \mathcal{U}(\kappa); \mathcal{U}(c)]\!]$$

- Each module expression, M, maps onto an unexpanded module expression, $\hat{M}$, as follows:

$$\begin{aligned}
\mathcal{U}(X) &= \widehat{X} \\
\mathcal{U}(\texttt{struct(}\hat{c};\hat{e}\texttt{)}) &= [\![\mathcal{U}(\hat{c}); \mathcal{U}(\hat{e})]\!] \\
\mathcal{U}(\texttt{seal\{}\sigma\texttt{\}(}M\texttt{)}) &= \mathcal{U}(M) \upharpoonright \mathcal{U}(\sigma) \\
\mathcal{U}(\texttt{mlet\{}\sigma\texttt{\}(}M; X.M'\texttt{)}) &= (\texttt{let } \widehat{X} = \mathcal{U}(M) \texttt{ in } \mathcal{U}(M')) : \mathcal{U}(\sigma)
\end{aligned}$$

- Each construction variable, u, maps onto a unique type identifier, written $\widehat{u}$.
- Each kind, κ, maps onto an unexpanded kind, $\mathcal{U}(\kappa)$, as follows:

$$\begin{aligned}
\mathcal{U}(\texttt{darr(}\kappa; u.\kappa'\texttt{)}) &= (\widehat{u} :: \mathcal{U}(\kappa)) \to \mathcal{U}(\kappa') \\
\mathcal{U}(\texttt{unit}) &= \langle\!\langle\rangle\!\rangle \\
\mathcal{U}(\texttt{dprod(}\kappa; u.\kappa'\texttt{)}) &= (\widehat{u} :: \mathcal{U}(\kappa)) \times \mathcal{U}(\kappa') \\
\mathcal{U}(\texttt{Type}) &= \texttt{T} \\
\mathcal{U}(\texttt{S(}\tau\texttt{)}) &= [=\mathcal{U}(\tau)]
\end{aligned}$$

- Each construction, c, except for constructions of the form $\texttt{con(}M\texttt{)}$ where M is not a module variable, maps onto an unexpanded type, $\mathcal{U}(c)$, as follows:

$$\begin{aligned}
\mathcal{U}(u) &= \widehat{u} \\
\mathcal{U}(\texttt{abs(}u.c\texttt{)}) &= \lambda\widehat{u}.\mathcal{U}(c) \\
\mathcal{U}(\texttt{app(}c; c'\texttt{)}) &= \mathcal{U}(c)(\mathcal{U}(c')) \\
\mathcal{U}(\texttt{triv}) &= \langle\!\langle\rangle\!\rangle \\
\mathcal{U}(\texttt{pair(}c; c'\texttt{)}) &= \langle\!\langle\mathcal{U}(c), \mathcal{U}(c')\rangle\!\rangle \\
\mathcal{U}(\texttt{prl(}c\texttt{)}) &= \mathcal{U}(c) \cdot \texttt{l} \\
\mathcal{U}(\texttt{prr(}c\texttt{)}) &= \mathcal{U}(c) \cdot \texttt{r} \\
\mathcal{U}(\texttt{parr(}\tau_1; \tau_2\texttt{)}) &= \mathcal{U}(\tau_1) \rightharpoonup \mathcal{U}(\tau_2) \\
\mathcal{U}(\texttt{all\{}\kappa\texttt{\}(}u.\tau\texttt{)}) &= \forall(\widehat{u} :: \mathcal{U}(\kappa)).\mathcal{U}(\tau) \\
\mathcal{U}(\texttt{rec(}t.\tau\texttt{)}) &= \mu\widehat{t}.\mathcal{U}(\tau)
\end{aligned}$$

$$\begin{aligned}
\mathcal{U}(\texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})) &= \langle \{i \hookrightarrow \mathcal{U}(\tau_i)\}_{i \in L} \rangle \\
\mathcal{U}(\texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L})) &= [\{i \hookrightarrow \mathcal{U}(\tau_i)\}_{i \in L}] \\
\mathcal{U}(\texttt{con}(X)) &= \widehat{X} \cdot \texttt{c}
\end{aligned}$$

- Each expression variable, x, maps onto a unique expression identifier, written $\widehat{x}$.
- Each expanded expression, e, except expressions of the form $\texttt{val}(M)$ where M is not a module variable, maps onto an unexpanded expression, $\mathcal{U}(e)$, as follows:

$$\begin{aligned}
\mathcal{U}(x) &= \widehat{x} \\
\mathcal{U}(\texttt{lam}\{\tau\}(x.e)) &= \lambda \widehat{x}{:}\mathcal{U}(\tau).\mathcal{U}(e) \\
\mathcal{U}(\texttt{ap}(e_1; e_2)) &= \mathcal{U}(e_1)(\mathcal{U}(e_2)) \\
\mathcal{U}(\texttt{clam}\{\kappa\}(u.e)) &= \Lambda \widehat{u}{::}\mathcal{U}(\kappa).\mathcal{U}(e) \\
\mathcal{U}(\texttt{cap}\{c\}(e)) &= \mathcal{U}(e)[\mathcal{U}(c)] \\
\mathcal{U}(\texttt{fold}(e)) &= \texttt{fold}(\mathcal{U}(e)) \\
\mathcal{U}(\texttt{unfold}(e)) &= \texttt{unfold}(\mathcal{U}(e)) \\
\mathcal{U}(\texttt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L})) &= \langle \{i \hookrightarrow \mathcal{U}(e_i)\}_{i \in L} \rangle \\
\mathcal{U}(\texttt{prj}[\ell](e)) &= \mathcal{U}(e) \cdot \ell \\
\mathcal{U}(\texttt{inj}[\ell](e)) &= \texttt{inj}[\ell](\mathcal{U}(e)) \\
\mathcal{U}(\texttt{match}[n](e; \{r_i\}_{1 \le i \le n})) &= \texttt{match}\,\mathcal{U}(e)\ \{\mathcal{U}(r_i)\}_{1 \le i \le n} \\
\mathcal{U}(\texttt{val}(X)) &= \widehat{X} \cdot \texttt{v}
\end{aligned}$$

- Each expanded rule, r, maps onto an unexpanded rule, $\mathcal{U}(r)$, as follows:

$$\mathcal{U}(\texttt{rule}(p.e)) = \texttt{urule}(\mathcal{U}(p).\mathcal{U}(e))$$

- Each expanded pattern, p, maps onto an unexpanded pattern, $\mathcal{U}(p)$, as follows:

$$\begin{aligned}
\mathcal{U}(x) &= \widehat{x} \\
\mathcal{U}(\texttt{wildp}) &= \texttt{uwildp} \\
\mathcal{U}(\texttt{foldp}(p)) &= \texttt{ufoldp}(\mathcal{U}(p)) \\
\mathcal{U}(\texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L})) &= \texttt{utplp}[L](\{i \hookrightarrow \mathcal{U}(p_i)\}_{i \in L}) \\
\mathcal{U}(\texttt{injp}[\ell](p)) &= \texttt{uinjp}[\ell](\mathcal{U}(p))
\end{aligned}$$

Textual Syntax

There is also a context-free textual syntax for the UL. We need only posit the existence of partial metafunctions that satisfy the following condition.

Condition C.11 (Textual Representability)**.**

1. *For each $\hat{\kappa}$, there exists b such that* $\mathsf{parseUKind}(b) = \hat{\kappa}$.

2. *For each $\hat{c}$, there exists b such that* $\mathsf{parseUCon}(b) = \hat{c}$.

3. *For each $\hat{e}$, there exists b such that* $\mathsf{parseUExp}(b) = \hat{e}$.

4. *For each $\hat{p}$, there exists b such that* $\mathsf{parseUPat}(b) = \hat{p}$.

Condition C.12 (Expression Parsing Monotonicity)**.** *If* $\mathsf{parseUExp}(b) = \hat{e}$ *then* $\|\hat{e}\| < \|b\|$.

Condition C.13 (Pattern Parsing Monotonicity)**.** *If* $\mathsf{parseUPat}(b) = \hat{p}$ *then* $\|\hat{p}\| < \|b\|$.

C.2.2 Typed Expansion

Unexpanded Unified Contexts

A *unexpanded unified context*, $\hat{\Omega}$, takes the form $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$, where $\mathcal{M}$ is a *module identifier expansion context*, $\mathcal{D}$ is a *construction identifier expansion context*, $\mathcal{G}$ is an *expression identifier expansion context*, and Ω is a unified context.

A module identifier expansion context, $\mathcal{M}$, is a finite function that maps each module identifier $\hat{X} \in \mathrm{dom}(\mathcal{M})$ to the module identifier expansion $\hat{X} \rightsquigarrow X$. We write $\hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma$ when $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$ as an abbreviation of

$$\langle \mathcal{M} \uplus \hat{X} \rightsquigarrow X; \mathcal{D}; \mathcal{G}; \Omega, X : \sigma \rangle$$

A construction identifier expansion context, $\mathcal{D}$, is a finite function that maps each construction identifier $\hat{u} \in \mathrm{dom}(\mathcal{D})$ to the construction identifier expansion $\hat{u} \rightsquigarrow u$. We write $\hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa$ when $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$ as an abbreviation of

$$\langle \mathcal{M}; \mathcal{D} \uplus \hat{u} \rightsquigarrow u; \mathcal{G}; \Omega, u :: \kappa \rangle$$

An expression identifier expansion context, $\mathcal{G}$, is a finite function that maps each expression identifier $\hat{x} \in \mathrm{dom}(\mathcal{G})$ to the expression identifier expansion $\hat{x} \rightsquigarrow x$. We write $\hat{\Omega}, \hat{x} \rightsquigarrow x : \tau$ when $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$ as an abbreviation of

$$\langle \mathcal{M}; \mathcal{D}; \mathcal{G} \uplus \hat{x} \rightsquigarrow x; \Omega, x : \tau \rangle$$

Body Encoding and Decoding

An assumed type abbreviated `Body` classifies encodings of literal bodies, b. The mapping from literal bodies to values of type `Body` is defined by the *body encoding judgement* $b \downarrow_{\mathsf{Body}} e_{\mathrm{body}}$. An inverse mapping is defined by the *body decoding judgement* $e_{\mathrm{body}} \uparrow_{\mathsf{Body}} b$.

Judgement Form	**Description**
$b \downarrow_{\mathsf{Body}} e$	b has encoding e
$e \uparrow_{\mathsf{Body}} b$	e has decoding b

The following condition establishes an isomorphism between literal bodies and values of type `Body` mediated by the judgements above.

Condition C.14 (Body Isomorphism)**.**

1. *For every literal body b, we have that $b \downarrow_{\mathsf{Body}} e_{body}$ for some e_{body} such that $\vdash e_{body} : \mathtt{Body}$ and e_{body} val.*
2. *If $\vdash e_{body} : \mathtt{Body}$ and e_{body} val then $e_{body} \uparrow_{\mathsf{Body}} b$ for some b.*
3. *If $b \downarrow_{\mathsf{Body}} e_{body}$ then $e_{body} \uparrow_{\mathsf{Body}} b$.*
4. *If $\vdash e_{body} : \mathtt{Body}$ and e_{body} val and $e_{body} \uparrow_{\mathsf{Body}} b$ then $b \downarrow_{\mathsf{Body}} e_{body}$.*
5. *If $b \downarrow_{\mathsf{Body}} e_{body}$ and $b \downarrow_{\mathsf{Body}} e'_{body}$ then $e_{body} = e'_{body}$.*
6. *If $\vdash e_{body} : \mathtt{Body}$ and e_{body} val and $e_{body} \uparrow_{\mathsf{Body}} b$ and $e_{body} \uparrow_{\mathsf{Body}} b'$ then $b = b'$.*

We also assume a partial metafunction, $\mathsf{subseq}(b; m; n)$, which extracts a subsequence of b starting at position m and ending at position n, inclusive, where m and n are natural numbers. The following condition is technically necessary.

Condition C.15 (Body Subsequencing)**.** *If $\mathsf{subseq}(b; m; n) = b'$ then $\|b'\| \leq \|b\|$.*

Parse Results

The type function abbreviated `ParseResult`, and auxiliary abbreviations used below, is defined as follows:

$$L_{\mathtt{P}} \stackrel{\text{def}}{=} \mathtt{ParseError, Success}$$

$$\mathtt{ParseResult} \stackrel{\text{def}}{=} \mathtt{abs}(t.\mathtt{sum}[L_{\mathtt{P}}](\mathtt{ParseError} \hookrightarrow \langle\rangle, \mathtt{Success} \hookrightarrow t))$$

$$\mathtt{ParseResult}(\tau) \stackrel{\text{def}}{=} \mathtt{app}(\mathtt{ParseResult}; \tau)$$

TSM Contexts

peTSM contexts, $\hat{\Psi}$, are of the form $\langle \mathcal{A}; \Psi \rangle$, where $\mathcal{A}$ is a *TSM identifier expansion context* and Ψ is a *peTSM definition context*.

ppTSM contexts, $\hat{\Phi}$, are of the form $\langle \mathcal{A}; \Phi \rangle$, where $\mathcal{A}$ is a TSM identifier expansion context and Φ is a *ppTSM definition context*.

A *TSM identifier expansion context*, $\mathcal{A}$, is a finite function mapping each TSM identifier $\hat{a} \in \mathrm{dom}(\mathcal{A})$ to the *TSM identifier expansion*, $\hat{a} \rightsquigarrow \epsilon$, for some *TSM expression*, ϵ. We write $\mathcal{A} \uplus \hat{a} \rightsquigarrow \epsilon$ for the TSM identifier expansion context that maps $\hat{a}$ to $\hat{a} \rightsquigarrow \epsilon$, and defers to $\mathcal{A}$ for all other TSM identifiers (i.e. the previous mapping is *updated*.)

A *peTSM definition context*, Ψ, is a finite function mapping each TSM name $a \in \mathrm{dom}(\Psi)$ to an *expanded peTSM definition*, $a \hookrightarrow \mathtt{petsm}(\rho; e_{\mathrm{parse}})$, where ρ is the peTSM's type annotation, and e_{parse} is its parse function. We write $\Psi, a \hookrightarrow \mathtt{petsm}(\rho; e_{\mathrm{parse}})$ when $a \notin \mathrm{dom}(\Psi)$ for the extension of Ψ that maps a to $a \hookrightarrow \mathtt{petsm}(\rho; e_{\mathrm{parse}})$. We write $\Omega \vdash \Psi$ peTSMs when all the TSM type annotations in Ψ are well-formed assuming Ω, and the parse functions in Ψ are closed and of the appropriate type.

Definition C.16 (peTSM Definition Context Formation). $\Omega \vdash \Psi$ peTSMs *iff for each* $a \hookrightarrow \texttt{petsm}(\rho; e_{parse}) \in \Psi$, *we have* $\Omega \vdash \rho$ tsmty *and*

$$\varnothing \vdash e_{parse} : \texttt{parr(Body;ParseResult(PPrExpr))}$$

Definition C.17 (peTSM Context Formation). $\Omega \vdash \langle \mathcal{A}; \Psi \rangle$ peTSMctx *iff* $\Omega \vdash \Psi$ peTSMs *and for each* $\hat{a} \rightsquigarrow \epsilon \in \mathcal{A}$ *we have* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *for some* ρ.

A *ppTSM definition context*, Φ, is a finite function mapping each TSM name $a \in \mathrm{dom}(\Phi)$ to an *expanded ppTSM definition*, $a \hookrightarrow \texttt{pptsm}(\rho; e_{\mathrm{parse}})$, where ρ is the ppTSM's type annotation, and e_{parse} is its parse function. We write $\Phi, a \hookrightarrow \texttt{pptsm}(\rho; e_{\mathrm{parse}})$ when $a \notin \mathrm{dom}(\Phi)$ for the extension of Φ that maps a to $a \hookrightarrow \texttt{pptsm}(\rho; e_{\mathrm{parse}})$. We write $\Omega \vdash \Phi$ ppTSMs when all the type annotations in Φ are well-formed assuming Ω, and the parse functions in Φ are closed and of the appropriate type.

Definition C.18 (ppTSM Definition Context Formation). $\Omega \vdash \Phi$ ppTSMs *iff for each* $\hat{a} \hookrightarrow \texttt{pptsm}(\rho; e_{parse}) \in \Phi$, *we have* $\Omega \vdash \rho$ tsmty *and*

$$\varnothing \vdash e_{parse} : \texttt{parr(Body;ParseResult(PPrPat))}$$

Definition C.19 (ppTSM Context Formation). $\Omega \vdash \langle \mathcal{A}; \Phi \rangle$ ppTSMctx *iff* $\Omega \vdash \Phi$ ppTSMs *and for each* $\hat{a} \rightsquigarrow \epsilon \in \mathcal{A}$ *we have* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *for some* ρ.

Signature and Module Expansion

$\boxed{\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma \ \mathsf{sig}}$ $\hat{\sigma}$ has well-formed expansion σ

$$\frac{\hat{\Omega} \vdash \hat{\kappa} \rightsquigarrow \kappa \ \mathsf{kind} \qquad \hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa \vdash \hat{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\hat{\Omega} \vdash [\![\hat{u} :: \hat{\kappa}; \hat{\tau}]\!] \rightsquigarrow \texttt{sig\{}\kappa\texttt{\}(}u.\tau\texttt{)} \ \mathsf{sig}} \tag{C.15}$$

$\boxed{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma}$ $\hat{M}$ has expansion M matching σ

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma \qquad \hat{\Omega} \vdash \sigma <: \sigma'}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma'} \tag{C.16a}$$

$$\frac{}{\hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash_{\hat{\Psi};\hat{\Phi}} \hat{X} \rightsquigarrow X : \sigma} \tag{C.16b}$$

$$\frac{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \kappa \qquad \hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [c/u]\tau}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} [\![\hat{c}; \hat{e}]\!] \rightsquigarrow \texttt{struct(}c;e\texttt{)} : \texttt{sig\{}\kappa\texttt{\}(}u.\tau\texttt{)}} \tag{C.16c}$$

$$\frac{\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma \ \mathsf{sig} \qquad \hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \upharpoonright \hat{\sigma} \rightsquigarrow \texttt{seal\{}\sigma\texttt{\}(}M\texttt{)} : \sigma} \tag{C.16d}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma \qquad \hat{\Omega} \vdash \hat{\sigma}' \rightsquigarrow \sigma' \ \mathsf{sig} \qquad \hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M}' \rightsquigarrow M' : \sigma'}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} (\mathtt{let}\ \hat{X} = \hat{M}\ \mathtt{in}\ \hat{M}') : \hat{\sigma}' \rightsquigarrow \mathtt{mlet\{}\sigma'\mathtt{\}(}M; X.M'\mathtt{)} : \sigma'} \tag{C.16e}$$

$$\frac{\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty} \qquad \varnothing \vdash e_{\mathrm{parse}} : \mathtt{parr(Body; ParseResult(PPrExpr))} \qquad e_{\mathrm{parse}} \Downarrow e'_{\mathrm{parse}} \qquad \hat{\Omega} \vdash_{\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \mathtt{defref[}a\mathtt{]}; \Psi, a \hookrightarrow \mathtt{petsm}(\rho; e'_{\mathrm{parse}}) \rangle;\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash_{\langle \mathcal{A};\Psi \rangle;\hat{\Phi}} \mathtt{syntax}\ \hat{a}\ \mathtt{at}\ \hat{\rho}\ \mathtt{for\ expressions\ by\ static}\ e_{\mathrm{parse}}\ \mathtt{in}\ \hat{M} \rightsquigarrow M : \sigma} \tag{C.16f}$$

$$\frac{\hat{\Omega} \vdash^{\mathsf{Exp}}_{\langle \mathcal{A};\Psi \rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho \qquad \hat{\Omega} \vdash_{\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \epsilon_{\mathrm{normal}};\Psi \rangle;\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash_{\langle \mathcal{A};\Psi \rangle;\hat{\Phi}} \mathtt{let\ syntax}\ \hat{a} = \hat{\epsilon}\ \mathtt{for\ expressions\ in}\ \hat{M} \rightsquigarrow M : \sigma} \tag{C.16g}$$

$$\frac{\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty} \qquad \varnothing \vdash e_{\mathrm{parse}} : \mathtt{parr(Body; ParseResult(PPrPat))} \qquad e_{\mathrm{parse}} \Downarrow e'_{\mathrm{parse}} \qquad \hat{\Omega} \vdash_{\hat{\Psi};\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \mathtt{defref[}a\mathtt{]}; \Phi, a \hookrightarrow \mathtt{pptsm}(\rho; e'_{\mathrm{parse}}) \rangle} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash_{\hat{\Psi};\langle \mathcal{A};\Phi \rangle} \mathtt{syntax}\ \hat{a}\ \mathtt{at}\ \hat{\rho}\ \mathtt{for\ patterns\ by\ static}\ e_{\mathrm{parse}}\ \mathtt{in}\ \hat{M} \rightsquigarrow M : \sigma} \tag{C.16h}$$

$$\frac{\hat{\Omega} \vdash^{\mathsf{Pat}}_{\langle \mathcal{A};\Phi \rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho \qquad \hat{\Omega} \vdash_{\hat{\Psi};\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \epsilon;\Phi \rangle} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash_{\hat{\Psi};\langle \mathcal{A};\Phi \rangle} \mathtt{let\ syntax}\ \hat{a} = \hat{\epsilon}\ \mathtt{for\ patterns\ in}\ \hat{M} \rightsquigarrow M : \sigma} \tag{C.16i}$$

Kind and Construction Expansion

$\boxed{\hat{\Omega} \vdash \hat{\kappa} \rightsquigarrow \kappa\ \mathsf{kind}}$ $\hat{\kappa}$ has well-formed expansion κ

$$\frac{\hat{\Omega} \vdash \hat{\kappa}_1 \rightsquigarrow \kappa_1\ \mathsf{kind} \qquad \hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa_1 \vdash \hat{\kappa}_2 \rightsquigarrow \kappa_2\ \mathsf{kind}}{\hat{\Omega} \vdash (\hat{u} :: \hat{\kappa}_1) \to \hat{\kappa}_2 \rightsquigarrow \mathtt{darr(}\kappa_1; u.\kappa_2\mathtt{)}\ \mathsf{kind}} \tag{C.17a}$$

$$\frac{}{\hat{\Omega} \vdash \langle\!\langle\rangle\!\rangle \rightsquigarrow \mathtt{unit}\ \mathsf{kind}} \tag{C.17b}$$

$$\frac{\hat{\Omega} \vdash \hat{\kappa}_1 \rightsquigarrow \kappa_1\ \mathsf{kind} \qquad \hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa_1 \vdash \hat{\kappa}_2 \rightsquigarrow \kappa_2\ \mathsf{kind}}{\hat{\Omega} \vdash (\hat{u} :: \hat{\kappa}_1) \times \hat{\kappa}_2 \rightsquigarrow \mathtt{dprod(}\kappa_1; u.\kappa_2\mathtt{)}\ \mathsf{kind}} \tag{C.17c}$$

$$\frac{}{\hat{\Omega} \vdash \mathtt{T} \rightsquigarrow \mathtt{Type}\ \mathsf{kind}} \tag{C.17d}$$

$$\frac{\hat{\Omega} \vdash \hat{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\hat{\Omega} \vdash [=\hat{\tau}] \rightsquigarrow \mathtt{S(}\tau\mathtt{)}\ \mathsf{kind}} \tag{C.17e}$$

$\boxed{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \kappa}$ $\hat{c}$ has expansion c of kind κ

$$\frac{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \kappa_1 \qquad \Omega \vdash \kappa_1 <:: \kappa_2}{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \kappa_2} \tag{C.18a}$$

$$\frac{}{\hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa \vdash \hat{u} \rightsquigarrow u :: \kappa} \tag{C.18b}$$

$$\frac{\hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa_1 \vdash \hat{c}_2 \rightsquigarrow c_2 :: \kappa_2}{\hat{\Omega} \vdash \lambda \hat{u}.\hat{c}_2 \rightsquigarrow \mathtt{abs}(u.c_2) :: \mathtt{darr}(\kappa_1; u.\kappa_2)} \tag{C.18c}$$

$$\frac{\hat{\Omega} \vdash \hat{c}_1 \rightsquigarrow c_1 :: \mathtt{darr}(\kappa_2; u.\kappa) \qquad \hat{\Omega} \vdash \hat{c}_2 \rightsquigarrow c_2 :: \kappa_2}{\hat{\Omega} \vdash \hat{c}_1(\hat{c}_2) \rightsquigarrow \mathtt{app}(c_1; c_2) :: [c_1/u]\kappa} \tag{C.18d}$$

$$\frac{}{\hat{\Omega} \vdash \langle\!\langle\rangle\!\rangle \rightsquigarrow \mathtt{triv} :: \mathtt{unit}} \tag{C.18e}$$

$$\frac{\hat{\Omega} \vdash \hat{c}_1 \rightsquigarrow c_1 :: \kappa_1 \qquad \hat{\Omega} \vdash \hat{c}_2 \rightsquigarrow c_2 :: [c_1/u]\kappa_2}{\hat{\Omega} \vdash \langle\!\langle \hat{c}_1, \hat{c}_2 \rangle\!\rangle \rightsquigarrow \mathtt{pair}(c_1; c_2) :: \mathtt{dprod}(\kappa_1; u.\kappa_2)} \tag{C.18f}$$

$$\frac{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\hat{\Omega} \vdash \hat{c} \cdot \mathtt{l} \rightsquigarrow \mathtt{prl}(c) :: \kappa_1} \tag{C.18g}$$

$$\frac{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\hat{\Omega} \vdash \hat{c} \cdot \mathtt{r} \rightsquigarrow \mathtt{prr}(c) :: [\mathtt{prl}(c)/u]\kappa_2} \tag{C.18h}$$

$$\frac{\hat{\Omega} \vdash \hat{\tau}_1 \rightsquigarrow \tau_1 :: \mathtt{Type} \qquad \hat{\Omega} \vdash \hat{\tau}_2 \rightsquigarrow \tau_2 :: \mathtt{Type}}{\hat{\Omega} \vdash \hat{\tau}_1 \rightharpoonup \hat{\tau}_2 \rightsquigarrow \mathtt{parr}(\tau_1; \tau_2) :: \mathtt{Type}} \tag{C.18i}$$

$$\frac{\hat{\Omega} \vdash \hat{\kappa} \rightsquigarrow \kappa \ \mathsf{kind} \qquad \hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa \vdash \hat{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\hat{\Omega} \vdash \forall(\hat{u} :: \hat{\kappa}).\hat{\tau} \rightsquigarrow \mathtt{all}\{\kappa\}(u.\tau) :: \mathtt{Type}} \tag{C.18j}$$

$$\frac{\hat{\Omega}, \hat{t} \rightsquigarrow t :: \mathtt{Type} \vdash \hat{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\hat{\Omega} \vdash \mu \hat{t}.\hat{\tau} \rightsquigarrow \mathtt{rec}(t.\tau) :: \mathtt{Type}} \tag{C.18k}$$

$$\frac{\{\hat{\Omega} \vdash \hat{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{1 \leq i \leq n}}{\hat{\Omega} \vdash \langle \{i \hookrightarrow \hat{\tau}_i\}_{i \in L} \rangle \rightsquigarrow \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) :: \mathtt{Type}} \tag{C.18l}$$

$$\frac{\{\hat{\Omega} \vdash \hat{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{1 \leq i \leq n}}{\hat{\Omega} \vdash [\{i \hookrightarrow \hat{\tau}_i\}_{i \in L}] \rightsquigarrow \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) :: \mathtt{Type}} \tag{C.18m}$$

$$\frac{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \mathtt{Type}}{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \mathtt{S}(c)} \tag{C.18n}$$

$$\frac{}{\hat{\Omega}, \hat{X} \rightsquigarrow X : \mathtt{sig}\{\kappa\}(u.\tau) \vdash \hat{X} \cdot \mathtt{c} \rightsquigarrow \mathtt{con}(X) :: \kappa} \tag{C.18o}$$

Type, Expression, Rule and Pattern Expansion

$\boxed{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau}$ $\hat{e}$ has expansion e of type τ

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau \qquad \Omega \vdash \tau <: \tau'}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'} \tag{C.19a}$$

$$\frac{}{\hat{\Omega}, \hat{x} \rightsquigarrow x : \tau \vdash_{\hat{\Psi};\hat{\Phi}} \hat{x} \rightsquigarrow x : \tau} \tag{C.19b}$$

$$\frac{\hat{\Omega} \vdash \hat{\tau} \rightsquigarrow \tau :: \mathtt{Type} \qquad \hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} : \hat{\tau} \rightsquigarrow e : \tau} \tag{C.19c}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1 \rightsquigarrow e_1 : \tau_1 \qquad \hat{\Omega}, \hat{x} \rightsquigarrow x : \tau_1 \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_2 \rightsquigarrow e_2 : \tau_2}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{let\,val}\ \hat{x} = \hat{e}_1\ \mathtt{in}\ \hat{e}_2 \rightsquigarrow \mathtt{ap(lam\{}\tau_1\mathtt{\}(}x.e_2\mathtt{);}e_1\mathtt{)} : \tau_2} \tag{C.19d}$$

$$\frac{\hat{\Omega} \vdash \hat{\tau}_1 \rightsquigarrow \tau_1 :: \mathtt{Type} \qquad \hat{\Omega}, \hat{x} \rightsquigarrow x : \tau_1 \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau_2}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \lambda \hat{x}{:}\hat{\tau}_1.\hat{e} \rightsquigarrow \mathtt{lam\{}\tau_1\mathtt{\}(}x.e\mathtt{)} : \mathtt{parr(}\tau_1\mathtt{;}\tau_2\mathtt{)}} \tag{C.19e}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1 \rightsquigarrow e_1 : \mathtt{parr(}\tau_2\mathtt{;}\tau\mathtt{)} \qquad \hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_2 \rightsquigarrow e_2 : \tau_2}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_1(\hat{e}_2) \rightsquigarrow \mathtt{ap(}e_1\mathtt{;}e_2\mathtt{)} : \tau} \tag{C.19f}$$

$$\frac{\hat{\Omega} \vdash \hat{\kappa} \rightsquigarrow \kappa\ \mathsf{kind} \qquad \hat{\Omega}, \hat{u} \rightsquigarrow u :: \kappa \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \Lambda \hat{u}{::}\hat{\kappa}.\hat{e} \rightsquigarrow \mathtt{clam\{}\kappa\mathtt{\}(}u.e\mathtt{)} : \mathtt{all\{}\kappa\mathtt{\}(}u.\tau\mathtt{)}} \tag{C.19g}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{all\{}\kappa\mathtt{\}(}u.\tau\mathtt{)} \qquad \hat{\Omega} \vdash \hat{c} \rightsquigarrow c \Rightarrow \kappa}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}\mathtt{[}\hat{c}\mathtt{]} \rightsquigarrow \mathtt{cap\{}c\mathtt{\}(}e\mathtt{)} : [c/t]\tau} \tag{C.19h}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [\mathtt{rec(}t.\tau\mathtt{)}/t]\tau}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{fold}(\hat{e}) \rightsquigarrow \mathtt{fold}(e) : \mathtt{rec}(t.\tau)} \tag{C.19i}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{rec}(t.\tau)}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{unfold}(\hat{e}) \rightsquigarrow \mathtt{unfold}(e) : [\mathtt{rec}(t.\tau)/t]\tau} \tag{C.19j}$$

$$\frac{\{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}_i \rightsquigarrow e_i : \tau_i\}_{i \in L}}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \langle \{i \hookrightarrow \hat{e}_i\}_{i \in L} \rangle \rightsquigarrow \mathtt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L}) : \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})} \tag{C.19k}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \mathtt{prod}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau)}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \cdot \ell \rightsquigarrow \mathtt{prj}[\ell](e) : \tau} \tag{C.19l}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}' \rightsquigarrow e' : \tau'}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{inj}[\ell](\hat{e}) \rightsquigarrow \mathtt{inj}[\ell](e') : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau')} \tag{C.19m}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau \qquad \{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r}_i \rightsquigarrow r_i \Rightarrow \tau \Mapsto \tau'\}_{1 \leq i \leq n}}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathtt{match}\ \hat{e}\ \{\hat{r}_i\}_{1 \leq i \leq n} \rightsquigarrow \mathtt{match}[n](e; \{r_i\}_{1 \leq i \leq n}) : \tau'} \tag{C.19n}$$

$$\frac{}{\hat{\Omega}, \hat{X} \rightsquigarrow X : \mathtt{sig}\{\kappa\}(u.\tau) \vdash_{\hat{\Psi};\hat{\Phi}} \hat{X} \cdot \mathtt{v} \rightsquigarrow \mathtt{val}(X) : [\mathtt{con}(X)/u]\tau} \tag{C.19o}$$

$$\frac{\begin{array}{c}\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle \qquad \hat{\Psi} = \langle \mathcal{A}; \Psi \rangle \\ \hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \mathtt{type}(\tau_{\text{final}}) \qquad \Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon_{\text{normal}} \\ \mathsf{tsmdef}(\epsilon_{\text{normal}}) = a \qquad \Psi = \Psi', a \hookrightarrow \mathtt{petsm}(\rho; e_{\text{parse}}) \\ b \downarrow_{\mathsf{Body}} e_{\text{body}} \qquad e_{\text{parse}}(e_{\text{body}}) \Downarrow \mathtt{inj}[\mathtt{SuccessE}](e_{\text{pproto}}) \qquad e_{\text{pproto}} \uparrow_{\mathsf{PPrExpr}} \dot{e} \\ \Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \looparrowright_{\epsilon_{\text{normal}}} \grave{e}\ ?\ \mathtt{type}(\tau_{\text{proto}}) \dashv \omega : \Omega_{\text{params}} \\ \mathsf{seg}(\grave{e})\ \mathsf{segments}\ b \qquad \Omega_{\text{params}} \vdash^{\omega:\Omega_{\text{params}};\hat{\Omega};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau_{\text{proto}}\end{array}}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}\ \text{`}b\text{`} \rightsquigarrow [\omega]e : [\omega]\tau_{\text{proto}}} \tag{C.19p}$$

$\boxed{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r} \rightsquigarrow r : \tau \Mapsto \tau'}$ $\hat{r}$ has expansion r taking values of type τ to values of type τ'

$$\frac{\begin{array}{c}\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \\ \hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \langle \emptyset; \emptyset; \mathcal{G}'; \Omega' \rangle \qquad \langle \mathcal{M}; \mathcal{D}; \mathcal{G} \uplus \mathcal{G}'; \Omega \cup \Omega' \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'\end{array}}{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{p} \Rightarrow \hat{e} \rightsquigarrow \mathtt{rule}(p.e) : \tau \Mapsto \tau'} \tag{C.20}$$

$\boxed{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Omega}'}$ $\hat{p}$ has expansion p matching against τ generating hypotheses $\hat{\Omega}'$

$$\frac{\begin{array}{c}\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \\ \hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Omega}' \qquad \Omega \vdash \tau <: \tau'\end{array}}{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau' \dashv\!\shortmid \hat{\Omega}'} \tag{C.21a}$$

$$\frac{}{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{x} \rightsquigarrow x : \tau \dashv\!\shortmid \langle \emptyset; \emptyset; \hat{x} \rightsquigarrow x; x : \tau \rangle} \tag{C.21b}$$

$$\frac{}{\hat{\Omega} \vdash_{\hat{\Phi}} \_ \rightsquigarrow \mathtt{wildp} : \tau \dashv\!\shortmid \langle \emptyset; \emptyset; \emptyset; \emptyset \rangle} \tag{C.21c}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : [\mathtt{rec}(t.\tau)/t]\tau \dashv\!\shortmid \hat{\Omega}'}{\hat{\Omega} \vdash_{\hat{\Phi}} \mathtt{fold}(\hat{p}) \rightsquigarrow \mathtt{foldp}(p) : \mathtt{rec}(t.\tau) \dashv\!\shortmid \hat{\Omega}'} \tag{C.21d}$$

$$\frac{\begin{array}{c}\tau = \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \\ \{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p}_i \rightsquigarrow p_i : \tau_i \dashv\!\shortmid \hat{\Omega}_i\}_{i \in L}\end{array}}{\hat{\Omega} \vdash_{\hat{\Phi}} \langle \{i \hookrightarrow \hat{p}_i\}_{i \in L} \rangle \rightsquigarrow \mathtt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) : \tau \dashv\!\shortmid \cup_{i \in L} \hat{\Omega}_i} \tag{C.21e}$$

$$\frac{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Omega}'}{\hat{\Omega} \vdash_{\hat{\Phi}} \mathtt{inj}[\ell](\hat{p}) \rightsquigarrow \mathtt{injp}[\ell](p) : \mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau) \dashv\!\shortmid \hat{\Omega}'} \tag{C.21f}$$

$$\frac{\begin{array}{c}\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle \qquad \hat{\Phi} = \langle \mathcal{A}; \Phi \rangle \\ \hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \texttt{type}(\tau_{\text{final}}) \qquad \Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \Downarrow \epsilon_{\text{normal}} \\ \mathsf{tsmdef}(\epsilon_{\text{normal}}) = a \qquad \Phi = \Phi', a \hookrightarrow \texttt{pptsm}(\rho; e_{\text{parse}}) \\ b \downarrow_{\mathsf{Body}} e_{\text{body}} \qquad e_{\text{parse}}(e_{\text{body}}) \Downarrow \texttt{inj}[\texttt{SuccessP}](e_{\text{pproto}}) \qquad e_{\text{pproto}} \uparrow_{\mathsf{PPrPat}} \dot{p} \\ \Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p} \looparrowright_{\epsilon_{\text{normal}}} \grave{p} \mathbin{?} \texttt{type}(\tau_{\text{proto}}) \dashv \omega : \Omega_{\text{params}} \\ \mathsf{seg}(\grave{p}) \text{ segments } b \qquad \grave{p} \rightsquigarrow p : \tau_{\text{proto}} \dashv\!\!\mid^{\omega : \Omega_{\text{params}}; \hat{\Omega}; \hat{\Phi}; b} \hat{\Omega}'\end{array}}{\hat{\Omega} \vdash_{\hat{\Phi}} \hat{e}\ \text{`}b\text{`} \rightsquigarrow p : [\omega]\tau_{\text{proto}} \dashv\!\!\mid \hat{\Omega}'} \tag{C.21g}$$

TSM Type and Expression Expansion

$\boxed{\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty}}$ $\hat{\rho}$ has well-formed expansion ρ

$$\frac{\hat{\Omega} \vdash \hat{\tau} \rightsquigarrow \tau :: \texttt{Type}}{\hat{\Omega} \vdash \hat{\tau} \rightsquigarrow \texttt{type}(\tau)\ \mathsf{tsmty}} \tag{C.22a}$$

$$\frac{\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma\ \mathsf{sig} \qquad \hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty}}{\hat{\Omega} \vdash \forall \hat{X}{:}\hat{\sigma}.\hat{\rho} \rightsquigarrow \texttt{allmods}\{\sigma\}(X.\rho)\ \mathsf{tsmty}} \tag{C.22b}$$

$\boxed{\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \rho}$ $\hat{e}$ has peTSM expression expansion ϵ at ρ

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho}{\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash^{\mathsf{Exp}}_{\langle \mathcal{A}, \hat{a} \hookrightarrow \epsilon; \Psi \rangle} \hat{a} \rightsquigarrow \epsilon \mathbin{@} \rho} \tag{C.23a}$$

$$\frac{\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma\ \mathsf{sig} \qquad \hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \rho}{\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \Lambda \hat{X}{:}\hat{\sigma}.\hat{e} \rightsquigarrow \texttt{absmod}\{\sigma\}(X.\epsilon) \mathbin{@} \texttt{allmods}\{\sigma\}(X.\rho)} \tag{C.23b}$$

$$\frac{\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \texttt{allmods}\{\sigma\}(X'.\rho) \qquad \hat{\Omega} \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{X} \rightsquigarrow X : \sigma}{\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{e}(\hat{X}) \rightsquigarrow \texttt{apmod}\{X\}(\epsilon) \mathbin{@} [X/X']\rho} \tag{C.23c}$$

$\boxed{\hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Psi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \rho}$ $\hat{e}$ has ppTSM expression expansion ϵ at ρ

$$\frac{\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho}{\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash^{\mathsf{Pat}}_{\langle \mathcal{A}, \hat{a} \hookrightarrow \epsilon; \Phi \rangle} \hat{a} \rightsquigarrow \epsilon \mathbin{@} \rho} \tag{C.24a}$$

$$\frac{\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma\ \mathsf{sig} \qquad \hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \rho}{\hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \Lambda \hat{X}{:}\hat{\sigma}.\hat{e} \rightsquigarrow \texttt{absmod}\{\sigma\}(X.\epsilon) \mathbin{@} \texttt{allmods}\{\sigma\}(X.\rho)} \tag{C.24b}$$

$$\frac{\hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \hat{e} \rightsquigarrow \epsilon \mathbin{@} \texttt{allmods}\{\sigma\}(X'.\rho) \qquad \hat{\Omega} \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{X} \rightsquigarrow X : \sigma}{\hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \hat{e}(\hat{X}) \rightsquigarrow \texttt{apmod}\{X\}(\epsilon) \mathbin{@} [X/X']\rho} \tag{C.24c}$$

Statics of the TSM Language

$\boxed{\Omega \vdash \rho\ \mathsf{tsmty}}$ ρ is a TSM type

$$\frac{\Omega \vdash \tau :: \mathsf{Type}}{\Omega \vdash \mathtt{type}(\tau)\ \mathsf{tsmty}} \tag{C.25a}$$

$$\frac{\Omega \vdash \sigma\ \mathsf{sig} \qquad \Omega, X : \sigma \vdash \rho\ \mathsf{tsmty}}{\Omega \vdash \mathtt{allmods}\{\sigma\}(X.\rho)\ \mathsf{tsmty}} \tag{C.25b}$$

$\boxed{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon\ @\ \rho}$ ϵ is a peTSM expression at ρ

$$\frac{\Omega \vdash \rho\ \mathsf{tsmty}}{\Omega \vdash^{\mathsf{Exp}}_{\Psi, a \hookrightarrow \mathtt{petsm}(\rho; e_{\mathrm{parse}})} \mathtt{defref}[a]\ @\ \rho} \tag{C.26a}$$

$$\frac{\Omega \vdash \sigma\ \mathsf{sig} \qquad \Omega, X : \sigma \vdash^{\mathsf{Exp}}_{\Psi} \epsilon\ @\ \rho}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{absmod}\{\sigma\}(X.\epsilon)\ @\ \mathtt{allmods}\{\sigma\}(X.\rho)} \tag{C.26b}$$

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon\ @\ \mathtt{allmods}\{\sigma\}(X'.\rho) \qquad \Omega \vdash X : \sigma}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{apmod}\{X\}(\epsilon)\ @\ [X/X']\rho} \tag{C.26c}$$

$\boxed{\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon\ @\ \rho}$ ϵ is a ppTSM expression at ρ

$$\frac{\Omega \vdash \rho\ \mathsf{tsmty}}{\Omega \vdash^{\mathsf{Pat}}_{\Phi, a \hookrightarrow \mathtt{pptsm}(\rho; e_{\mathrm{parse}})} \mathtt{defref}[a]\ @\ \rho} \tag{C.27a}$$

$$\frac{\Omega \vdash \sigma\ \mathsf{sig} \qquad \Omega, X : \sigma \vdash^{\mathsf{Pat}}_{\Phi} \epsilon\ @\ \rho}{\Omega \vdash^{\mathsf{Pat}}_{\Phi} \mathtt{absmod}\{\sigma\}(X.\epsilon)\ @\ \mathtt{allmods}\{\sigma\}(X.\rho)} \tag{C.27b}$$

$$\frac{\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon\ @\ \mathtt{allmods}\{\sigma\}(X'.\rho) \qquad \Omega \vdash X : \sigma}{\Omega \vdash^{\mathsf{Pat}}_{\Phi} \mathtt{apmod}\{X\}(\epsilon)\ @\ [X/X']\rho} \tag{C.27c}$$

The following metafunction extracts the TSM name from a TSM expression.

$$\mathsf{tsmdef}(\mathtt{defref}[a]) = a \tag{C.28a}$$

$$\mathsf{tsmdef}(\mathtt{absmod}\{\sigma\}(X.\epsilon)) = \mathsf{tsmdef}(\epsilon) \tag{C.28b}$$

$$\mathsf{tsmdef}(\mathtt{apmod}\{X\}(\epsilon)) = \mathsf{tsmdef}(\epsilon) \tag{C.28c}$$

Dynamics of the TSM Language

$\boxed{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto \epsilon'}$ peTSM expression ϵ transitions to ϵ'

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto \epsilon'}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(}\epsilon\mathtt{)} \mapsto \mathtt{apmod\{}X\mathtt{\}(}\epsilon'\mathtt{)}} \tag{C.29a}$$

$$\frac{}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(absmod\{}\sigma\mathtt{\}(}X'.\epsilon\mathtt{))} \mapsto [X/X']\epsilon} \tag{C.29b}$$

$\boxed{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto \epsilon'}$ ppTSM expression ϵ transitions to ϵ'

$$\frac{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto \epsilon'}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(}\epsilon\mathtt{)} \mapsto \mathtt{apmod\{}X\mathtt{\}(}\epsilon'\mathtt{)}} \tag{C.30a}$$

$$\frac{}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(absmod\{}\sigma\mathtt{\}(}X'.\epsilon\mathtt{))} \mapsto [X/X']\epsilon} \tag{C.30b}$$

$\boxed{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon'}$ peTSM expression ϵ transitions in multiple steps to ϵ'

$$\frac{}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon} \tag{C.31a}$$

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto \epsilon'}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon'} \tag{C.31b}$$

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon' \qquad \Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mapsto^* \epsilon''}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon''} \tag{C.31c}$$

$\boxed{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto^* \epsilon'}$ ppTSM expression ϵ transitions in multiple steps to ϵ'

$$\frac{}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto^* \epsilon} \tag{C.32a}$$

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto \epsilon'}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto^* \epsilon'} \tag{C.32b}$$

$$\frac{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto^* \epsilon' \qquad \Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon' \mapsto^* \epsilon''}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \mapsto^* \epsilon''} \tag{C.32c}$$

$\boxed{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon'}$ peTSM expression ϵ normalizes to ϵ'

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon' \qquad \Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \ \mathsf{normal}}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon'} \tag{C.33}$$

$\boxed{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \Downarrow \epsilon'}$ ppTSM expression ϵ normalizes to ϵ'

$$\frac{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^{*} \epsilon' \qquad \Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \ \mathsf{normal}}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \Downarrow \epsilon'} \tag{C.34}$$

$\boxed{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \ \mathsf{normal}}$ ϵ is a normal peTSM expression

$$\frac{}{\Omega \vdash^{\mathsf{Exp}}_{\Psi, a \hookrightarrow \mathtt{petsm}(\rho; e_{\mathrm{parse}})} \mathtt{defref[}a\mathtt{]} \ \mathsf{normal}} \tag{C.35a}$$

$$\frac{}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{absmod\{}\sigma\mathtt{\}(}X.\epsilon\mathtt{)} \ \mathsf{normal}} \tag{C.35b}$$

$$\frac{\epsilon \neq \mathtt{absmod\{}\sigma\mathtt{\}(}X'.\epsilon'\mathtt{)} \qquad \Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \ \mathsf{normal}}{\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(}\epsilon\mathtt{)} \ \mathsf{normal}} \tag{C.35c}$$

$\boxed{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \ \mathsf{normal}}$ ϵ is a normal ppTSM expression

$$\frac{}{\Omega \vdash^{\mathsf{Pat}}_{\Psi, a \hookrightarrow \mathtt{petsm}(\rho; e_{\mathrm{parse}})} \mathtt{defref[}a\mathtt{]} \ \mathsf{normal}} \tag{C.36a}$$

$$\frac{}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \mathtt{absmod\{}\sigma\mathtt{\}(}X.\epsilon\mathtt{)} \ \mathsf{normal}} \tag{C.36b}$$

$$\frac{\epsilon \neq \mathtt{absmod\{}\sigma\mathtt{\}(}X'.\epsilon'\mathtt{)} \qquad \Omega \vdash^{\mathsf{Pat}}_{\Psi} \epsilon \ \mathsf{normal}}{\Omega \vdash^{\mathsf{Pat}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(}\epsilon\mathtt{)} \ \mathsf{normal}} \tag{C.36c}$$

C.3 Proto-Expansion Validation

C.3.1 Syntax of Proto-Expansions

Syntax – Parameterized Proto-Expressions

Sort			Operational Form	Stylized Form	Description
PPrExpr	$\dot{e}$	::=	`prexp(`$\grave{e}$`)`	$\grave{e}$	proto-expression
			`prbindmod(`$X.\dot{e}$`)`	$\Lambda X.\dot{e}$	module binding

Syntax – Parameterized Proto-Patterns

Sort			Operational Form	Stylized Form	Description
PPrPat	$\dot{p}$	::=	`prpat(`$\grave{p}$`)`	$\grave{p}$	proto-pattern
			`prbindmod(`$X.\dot{p}$`)`	$\Lambda X.\dot{p}$	module binding

Syntax – Proto-Kinds and Proto-Constructions

Sort			Operational Form	Stylized Form	Description
PrKind	$\grave{\kappa}$	::=	$\texttt{prdarr}(\grave{\kappa}; u.\grave{\kappa})$	$(u :: \grave{\kappa}) \to \grave{\kappa}$	dependent function
			$\texttt{prunit}$	$\langle\!\langle\rangle\!\rangle$	nullary product
			$\texttt{prdprod}(\grave{\kappa}; u.\grave{\kappa})$	$(u :: \grave{\kappa}) \times \grave{\kappa}$	dependent product
			$\texttt{prType}$	$\texttt{T}$	type
			$\texttt{prS}(\grave{\tau})$	$[{=}\grave{\tau}]$	singleton
			$\texttt{splicedk}[m; n]$	$\texttt{splicedk}[m; n]$	spliced kind
PrCon	$\grave{c}, \grave{\tau}$	::=	u	u	construction variable
			t	t	type variable
			$\texttt{prabs}(u.\grave{c})$	$\lambda u.\grave{c}$	abstraction
			$\texttt{prapp}(\grave{c}; \grave{c})$	$\grave{c}(\grave{c})$	application
			$\texttt{prtriv}$	$\langle\!\langle\rangle\!\rangle$	trivial
			$\texttt{prpair}(\grave{c}; \grave{c})$	$\langle\!\langle \grave{c}, \grave{c} \rangle\!\rangle$	pair
			$\texttt{prprl}(\grave{c})$	$\grave{c} \cdot \texttt{l}$	left projection
			$\texttt{prprr}(\grave{c})$	$\grave{c} \cdot \texttt{r}$	right projection
			$\texttt{prparr}(\grave{\tau}; \grave{\tau})$	$\grave{\tau} \rightharpoonup \grave{\tau}$	partial function
			$\texttt{prall}\{\grave{\kappa}\}(u.\grave{\tau})$	$\forall(u :: \grave{\kappa}).\grave{\tau}$	polymorphic
			$\texttt{prrec}(t.\grave{\tau})$	$\mu t.\grave{\tau}$	recursive
			$\texttt{prprod}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L})$	$\langle \{i \hookrightarrow \grave{\tau}_i\}_{i \in L} \rangle$	labeled product
			$\texttt{prsum}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L})$	$[\{i \hookrightarrow \grave{\tau}_i\}_{i \in L}]$	labeled sum
			$\texttt{prcon}(X)$	$X \cdot \texttt{c}$	construction component
			$\texttt{splicedc}[m; n; \grave{\kappa}]$	$\texttt{splicedc}[m; n; \grave{\kappa}]$	spliced construction

Syntax – Proto-Expressions and Proto-Rules

Sort			Operational Form	Stylized Form	Description
PrExp	$\grave{e}$	::=	x	x	variable
			$\texttt{prasc}\{\grave{\tau}\}(\grave{e})$	$\grave{e} : \grave{\tau}$	ascription
			$\texttt{prletval}(\grave{e}; x.\grave{e})$	$\texttt{let val}\ x = \grave{e}\ \texttt{in}\ \grave{e}$	value binding
			$\texttt{prlam}\{\grave{\tau}\}(x.\grave{e})$	$\lambda x{:}\grave{\tau}.\grave{e}$	abstraction
			$\texttt{prap}(\grave{e}; \grave{e})$	$\grave{e}(\grave{e})$	application
			$\texttt{prclam}\{\grave{\kappa}\}(u.\grave{e})$	$\Lambda u{::}\grave{\kappa}.\grave{e}$	construction abstraction
			$\texttt{prcap}\{\grave{c}\}(\grave{e})$	$\grave{e}[\grave{c}]$	construction application
			$\texttt{prfold}(\grave{e})$	$\texttt{fold}(\grave{e})$	fold
			$\texttt{prunfold}(\grave{e})$	$\texttt{unfold}(\grave{e})$	unfold
			$\texttt{prtpl}\{L\}(\{i \hookrightarrow \grave{e}_i\}_{i \in L})$	$\langle \{i \hookrightarrow \grave{e}_i\}_{i \in L} \rangle$	labeled tuple
			$\texttt{prprj}[\ell](\grave{e})$	$\grave{e} \cdot \ell$	projection
			$\texttt{prinj}[\ell](\grave{e})$	$\texttt{inj}[\ell](\grave{e})$	injection
			$\texttt{prmatch}[n](\grave{e}; \{\grave{r}_i\}_{1 \le i \le n})$	$\texttt{match}\ \grave{e}\ \{\grave{r}_i\}_{1 \le i \le n}$	match
			$\texttt{prval}(X)$	$X \cdot \texttt{v}$	value component
			$\texttt{splicede}[m; n; \grave{\tau}]$	$\texttt{splicede}[m; n; \grave{\tau}]$	spliced expression
PrRule	$\grave{r}$	::=	$\texttt{prrule}(p.\grave{e})$	$p \Rightarrow \grave{e}$	rule

Syntax – Proto-Patterns

PrPat	$\grave{p}$	::=	$\texttt{prwildp}$	$\_$ wildcard pattern
			$\texttt{prfoldp}(p)$	$\texttt{fold}(p)$ fold pattern
			$\texttt{prtplp}[L](\{i \hookrightarrow \grave{p}_i\}_{i \in L})$	$\langle \{i \hookrightarrow \grave{p}_i\}_{i \in L} \rangle$ labeled tuple pattern
			$\texttt{prinjp}[\ell](\grave{p})$	$\texttt{inj}[\ell](\grave{p})$ injection pattern
			$\texttt{splicedp}[m; n; \grave{\tau}]$	$\texttt{splicedp}[m; n; \grave{\tau}]$ spliced pattern

Common Proto-Expansion Terms

Each expanded term, with a few exceptions noted below, maps onto a proto-expansion term. We refer to these as the *common proto-expansion terms*. In particular:

- Each kind, κ, maps onto a proto-kind, $\mathcal{P}(\kappa)$, as follows:

$$\begin{aligned}
\mathcal{P}(\texttt{darr}(\kappa_1; u.\kappa_2)) &= \texttt{prdarr}(\mathcal{P}(\kappa_1); u.\mathcal{P}(\kappa_2)) \\
\mathcal{P}(\texttt{unit}) &= \texttt{prunit} \\
\mathcal{P}(\texttt{dprod}(\kappa_1; u.\kappa_2)) &= \texttt{prdprod}(\mathcal{P}(\kappa_1); u.\mathcal{P}(\kappa_2)) \\
\mathcal{P}(\texttt{Type}) &= \texttt{prType} \\
\mathcal{P}(\texttt{S}(\tau)) &= \texttt{prS}(\mathcal{P}(\tau))
\end{aligned}$$

- Each construction, c, maps onto a proto-construction, $\mathcal{P}(c)$, as follows:

$$\begin{aligned}
\mathcal{P}(u) &= u \\
\mathcal{P}(\texttt{abs}(u.c)) &= \texttt{prabs}(u.\mathcal{P}(c)) \\
\mathcal{P}(\texttt{app}(c_1; c_2)) &= \texttt{prapp}(\mathcal{P}(c_1); \mathcal{P}(c_2)) \\
\mathcal{P}(\texttt{triv}) &= \texttt{prtriv} \\
\mathcal{P}(\texttt{pair}(c_1; c_2)) &= \texttt{prpair}(\mathcal{P}(c_1); \mathcal{P}(c_2)) \\
\mathcal{P}(\texttt{prl}(c)) &= \texttt{prprl}(\mathcal{P}(c)) \\
\mathcal{P}(\texttt{prr}(c)) &= \texttt{prprr}(\mathcal{P}(c)) \\
\mathcal{P}(\texttt{parr}(\tau_1; \tau_2)) &= \texttt{prparr}(\mathcal{P}(\tau_1); \mathcal{P}(\tau_2)) \\
\mathcal{P}(\texttt{all}(t.\tau)) &= \texttt{prall}(t.\mathcal{P}(\tau)) \\
\mathcal{P}(\texttt{rec}(t.\tau)) &= \texttt{prrec}(t.\mathcal{P}(\tau)) \\
\mathcal{P}(\texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L})) &= \texttt{prprod}[L](\{i \hookrightarrow \mathcal{P}(\tau_i)\}_{i \in L}) \\
\mathcal{P}(\texttt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L})) &= \texttt{prsum}[L](\{i \hookrightarrow \mathcal{P}(\tau_i)\}_{i \in L}) \\
\mathcal{P}(\texttt{con}(X)) &= \texttt{prcon}(X)
\end{aligned}$$

- Each expanded expression, e, except for the value projection of a module expression

that is not of module variable form, maps onto a proto-expression, $\mathcal{P}(e)$, as follows:

$$\begin{aligned}
\mathcal{P}(x) &= x \\
\mathcal{P}(\texttt{lam\{}\tau\texttt{\}(}x.e\texttt{)}) &= \texttt{prlam\{}\mathcal{P}(\tau)\texttt{\}(}x.\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{ap(}e_1\texttt{;}e_2\texttt{)}) &= \texttt{prap(}\mathcal{P}(e_1)\texttt{;}\mathcal{P}(e_2)\texttt{)} \\
\mathcal{P}(\texttt{clam\{}\kappa\texttt{\}(}u.e\texttt{)}) &= \texttt{prclam\{}\mathcal{P}(\kappa)\texttt{\}(}u.\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{cap\{}c\texttt{\}(}e\texttt{)}) &= \texttt{prcap\{}\mathcal{P}(c)\texttt{\}(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{fold(}e\texttt{)}) &= \texttt{prfold(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{unfold(}e\texttt{)}) &= \texttt{prunfold(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{tpl}[L]\texttt{(}\{i \hookrightarrow e_i\}_{i\in L}\texttt{)}) &= \texttt{prtpl\{}L\texttt{\}(}\{i \hookrightarrow \mathcal{P}(e_i)\}_{i\in L}\texttt{)} \\
\mathcal{P}(\texttt{inj}[\ell]\texttt{(}e\texttt{)}) &= \texttt{prinj}[\ell]\texttt{(}\mathcal{P}(e)\texttt{)} \\
\mathcal{P}(\texttt{match}[n]\texttt{(}e\texttt{;}\{r_i\}_{1\leq i\leq n}\texttt{)}) &= \texttt{prmatch}[n]\texttt{(}\mathcal{P}(e)\texttt{;}\{\mathcal{P}(r_i)\}_{1\leq i\leq n}\texttt{)} \\
\mathcal{P}(\texttt{val(}X\texttt{)}) &= \texttt{prval(}X\texttt{)}
\end{aligned}$$

- Each expanded rule, r, maps onto the proto-rule, $\mathcal{P}(r)$, as follows:

  $$\mathcal{P}(\texttt{rule(}p.e\texttt{)}) = \texttt{prrule(}p.\mathcal{P}(e)\texttt{)}$$

  Notice that proto-rules bind expanded patterns, not proto-patterns. This is because proto-rules appear in proto-expressions, which are generated by peTSMs. It would not be sensible for an peTSM to splice a pattern out of a literal body.

- Each expanded pattern, p, except for the variable patterns, maps onto a proto-pattern, $\mathcal{P}(p)$, as follows:

  $$\begin{aligned}
  \mathcal{P}(\texttt{wildp}) &= \texttt{prwildp} \\
  \mathcal{P}(\texttt{foldp(}p\texttt{)}) &= \texttt{prfoldp(}\mathcal{P}(p)\texttt{)} \\
  \mathcal{P}(\texttt{tplp}[L]\texttt{(}\{i \hookrightarrow p_i\}_{i\in L}\texttt{)}) &= \texttt{prtplp}[L]\texttt{(}\{i \hookrightarrow \mathcal{P}(p_i)\}_{i\in L}\texttt{)} \\
  \mathcal{P}(\texttt{injp}[\ell]\texttt{(}p\texttt{)}) &= \texttt{prinjp}[\ell]\texttt{(}\mathcal{P}(p)\texttt{)}
  \end{aligned}$$

Parameterized Proto-Expression Encoding and Decoding

The type abbreviated `PPrExpr` classifies encodings of *parameterized proto-expressions*. The mapping from parameterized proto-expressions to values of type `PPrExpr` is defined by the *parameterized proto-expression encoding judgement*, $\dot{e} \downarrow_{\texttt{PPrExpr}} e$. An inverse mapping is defined by the *parameterized proto-expression decoding judgement*, $e \uparrow_{\texttt{PPrExpr}} \dot{e}$.

Judgement Form	**Description**
$\dot{e} \downarrow_{\texttt{PPrExpr}} e$	$\dot{e}$ has encoding e
$e \uparrow_{\texttt{PPrExpr}} \dot{e}$	e has decoding $\dot{e}$

Rather than picking a particular definition of `PPrExpr` and defining the judgements above inductively against it, we only state the following condition, which establishes an isomorphism between values of type `PPrExpr` and parameterized proto-expressions.

Condition C.20 (Parameterized Proto-Expression Isomorphism)**.**

1. *For every* $\dot{e}$*, we have* $\dot{e} \downarrow_{\mathsf{PPrExpr}} e_{proto}$ *for some* e_{proto} *such that* $\vdash e_{proto} : \mathtt{PPrExpr}$ *and* e_{proto} val.
2. *If* $\vdash e_{proto} : \mathtt{PPrExpr}$ *and* e_{proto} val *then* $e_{proto} \uparrow_{\mathsf{PPrExpr}} \dot{e}$ *for some* $\dot{e}$.
3. *If* $\dot{e} \downarrow_{\mathsf{PPrExpr}} e_{proto}$ *then* $e_{proto} \uparrow_{\mathsf{PPrExpr}} \dot{e}$.
4. *If* $\vdash e_{proto} : \mathtt{PPrExpr}$ *and* e_{proto} val *and* $e_{proto} \uparrow_{\mathsf{PPrExpr}} \dot{e}$ *then* $\dot{e} \downarrow_{\mathsf{PPrExpr}} e_{proto}$.
5. *If* $\dot{e} \downarrow_{\mathsf{PPrExpr}} e_{proto}$ *and* $\dot{e} \downarrow_{\mathsf{PPrExpr}} e'_{proto}$ *then* $e_{proto} = e'_{proto}$.
6. *If* $\vdash e_{proto} : \mathtt{PPrExpr}$ *and* e_{proto} val *and* $e_{proto} \uparrow_{\mathsf{PPrExpr}} \dot{e}$ *and* $e_{proto} \uparrow_{\mathsf{PPrExpr}} \dot{e}'$ *then* $\dot{e} = \dot{e}'$.

Parameterized Proto-Pattern Encoding and Decoding

The type abbreviated `PPrPat` classifies encodings of *parameterized proto-patterns*. The mapping from parameterized proto-patterns to values of type `PPrPat` is defined by the *parameterized proto-pattern encoding judgement*, $\dot{p} \downarrow_{\mathsf{PPrPat}} p$. An inverse mapping is defined by the *parameterized proto-expression decoding judgement*, $p \uparrow_{\mathsf{PPrPat}} \dot{p}$.

Judgement Form	**Description**
$\dot{p} \downarrow_{\mathsf{PPrPat}} p$	$\dot{p}$ has encoding p
$p \uparrow_{\mathsf{PPrPat}} \dot{p}$	p has decoding $\dot{p}$

Again, rather than picking a particular definition of `PPrPat` and defining the judgements above inductively against it, we only state the following condition, which establishes an isomorphism between values of type `PPrPat` and parameterized proto-patterns.

Condition C.21 (Parameterized Proto-Pattern Isomorphism)**.**

1. *For every* $\dot{p}$*, we have* $\dot{p} \downarrow_{\mathsf{PPrPat}} e_{proto}$ *for some* e_{proto} *such that* $\vdash e_{proto} : \mathtt{PPrPat}$ *and* e_{proto} val.
2. *If* $\vdash e_{proto} : \mathtt{PPrPat}$ *and* e_{proto} val *then* $e_{proto} \uparrow_{\mathsf{PPrPat}} \dot{p}$ *for some* $\dot{p}$.
3. *If* $\dot{p} \downarrow_{\mathsf{PPrPat}} e_{proto}$ *then* $e_{proto} \uparrow_{\mathsf{PPrPat}} \dot{p}$.
4. *If* $\vdash e_{proto} : \mathtt{PPrPat}$ *and* e_{proto} val *and* $e_{proto} \uparrow_{\mathsf{PPrPat}} \dot{p}$ *then* $\dot{p} \downarrow_{\mathsf{PPrPat}} e_{proto}$.
5. *If* $\dot{p} \downarrow_{\mathsf{PPrPat}} e_{proto}$ *and* $\dot{p} \downarrow_{\mathsf{PPrPat}} e'_{proto}$ *then* $e_{proto} = e'_{proto}$.
6. *If* $\vdash e_{proto} : \mathtt{PPrPat}$ *and* e_{proto} val *and* $e_{proto} \uparrow_{\mathsf{PPrPat}} \dot{p}$ *and* $e_{proto} \uparrow_{\mathsf{PPrPat}} \dot{p}'$ *then* $\dot{p} = \dot{p}'$.

Splice Summaries

The *splice summary* of a proto-expression, $\mathsf{summary}(\grave{e})$, or proto-pattern, $\mathsf{summary}(\grave{p})$, is the finite set of references to spliced kinds, constructions, expressions and patterns that it mentions.

Segmentations

A *segment set*, ψ, is a finite set of pairs of natural numbers indicating the locations of spliced terms. The *segmentation* of a proto-expression, $\mathsf{seg}(\grave{e})$, or proto-pattern, $\mathsf{seg}(\grave{p})$, is the segment set implied by the splice summary.

The predicate ψ segments b checks that each segment in ψ, has non-negative length and is within bounds of b, and that the segments in ψ do not overlap.

C.3.2 Deparameterization

$\boxed{\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \leadsto_{\epsilon} \grave{e} \mathbin{?} \rho \dashv \omega : \Omega_{\text{params}}}$ When applying peTSM ϵ, $\dot{e}$ has deparameterization $\grave{e}$ leaving ρ with parameter substitution ω

$$\frac{\Omega_{\text{app}} \vdash \rho\ \mathsf{tsmty}}{\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi, a \hookrightarrow \mathtt{petsm}(\rho; e_{\text{parse}})} \mathtt{prexp(}\grave{e}\mathtt{)} \leadsto_{\mathtt{defref[}a\mathtt{]}} \grave{e} \mathbin{?} \rho \dashv \emptyset : \emptyset} \tag{C.37a}$$

$$\frac{\begin{array}{c}\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \leadsto_{\epsilon} \grave{e} \mathbin{?} \mathtt{allmods\{}\sigma\mathtt{\}(}X.\rho\mathtt{)} \dashv \omega : \Omega \\ \Omega_{\text{app}} \vdash X' : \sigma \qquad X \notin \mathrm{dom}(\Omega_{\text{app}})\end{array}}{\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{prbindmod(}X.\dot{e}\mathtt{)} \leadsto_{\mathtt{apmod\{}X'\mathtt{\}(}\epsilon\mathtt{)}} \grave{e} \mathbin{?} \rho \dashv (\omega, X'/X) : (\Omega, X : \sigma)} \tag{C.37b}$$

$\boxed{\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p} \leadsto_{\epsilon} \grave{p} \mathbin{?} \rho \dashv \omega : \Omega_{\text{params}}}$ When applying ppTSM ϵ, $\dot{p}$ has deparameterization $\grave{p}$ leaving ρ with parameter substitution ω

$$\frac{\Omega_{\text{app}} \vdash \rho\ \mathsf{tsmty}}{\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi, a \hookrightarrow \mathtt{pptsm}(\rho; e_{\text{parse}})} \mathtt{prpat(}\grave{p}\mathtt{)} \leadsto_{\mathtt{defref[}a\mathtt{]}} \grave{p} \mathbin{?} \rho \dashv \emptyset : \emptyset} \tag{C.38a}$$

$$\frac{\begin{array}{c}\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p} \leadsto_{\epsilon} \grave{p} \mathbin{?} \mathtt{allmods\{}\sigma\mathtt{\}(}X.\rho\mathtt{)} \dashv \omega : \Omega \\ \Omega_{\text{app}} \vdash X' : \sigma \qquad X \notin \mathrm{dom}(\Omega_{\text{app}})\end{array}}{\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \mathtt{prbindmod(}X.\dot{p}\mathtt{)} \leadsto_{\mathtt{apmod\{}X'\mathtt{\}(}\epsilon\mathtt{)}} \grave{p} \mathbin{?} \rho \dashv (\omega, X'/X) : (\Omega, X : \sigma)} \tag{C.38b}$$

C.3.3 Proto-Expansion Validation

Splicing Scenes

Expression splicing scenes, $\mathbb{E}$, are of the form $\omega : \Omega_{\text{params}}; \hat{\Omega}; \hat{\Psi}; \hat{\Phi}; b$, *construction splicing scenes*, $\mathbb{C}$, are of the form $\omega : \Omega_{\text{params}}; \hat{\Omega}; b$, and *pattern splicing scenes*, $\mathbb{P}$, are of the form $\omega : \Omega_{\text{params}}; \hat{\Omega}; \hat{\Phi}; b$. We write $\mathsf{cs}(\mathbb{E})$ for the construction splicing scene constructed by dropping the TSM contexts from $\mathbb{E}$:

$$\mathsf{cs}(\omega : \Omega_{\text{params}}; \hat{\Omega}; \hat{\Psi}; \hat{\Phi}; b) = \omega : \Omega_{\text{params}}; \hat{\Omega}; b$$

Proto-Kind and Proto-Construction Validation

$\boxed{\Omega \vdash^{\mathbb{C}} \grave{\kappa} \rightsquigarrow \kappa \text{ kind}}$ $\grave{\kappa}$ has well-formed expansion κ

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{\kappa}_1 \rightsquigarrow \kappa_1 \text{ kind} \qquad \Omega, u :: \kappa_1 \vdash^{\mathbb{C}} \grave{\kappa}_2 \rightsquigarrow \kappa_2 \text{ kind}}{\Omega \vdash^{\mathbb{C}} \mathtt{prdarr}(\grave{\kappa}_1; u.\grave{\kappa}_2) \rightsquigarrow \mathtt{darr}(\kappa_1; u.\kappa_2) \text{ kind}} \tag{C.39a}$$

$$\frac{}{\Omega \vdash^{\mathbb{C}} \mathtt{prunit} \rightsquigarrow \mathtt{unit} \text{ kind}} \tag{C.39b}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{\kappa}_1 \rightsquigarrow \kappa_1 \text{ kind} \qquad \Omega, u :: \kappa_1 \vdash^{\mathbb{C}} \grave{\kappa}_2 \rightsquigarrow \kappa_2 \text{ kind}}{\Omega \vdash^{\mathbb{C}} \mathtt{prdprod}(\grave{\kappa}_1; u.\grave{\kappa}_2) \rightsquigarrow \mathtt{dprod}(\kappa_1; u.\kappa_2) \text{ kind}} \tag{C.39c}$$

$$\frac{}{\Omega \vdash^{\mathbb{C}} \mathtt{prType} \rightsquigarrow \mathtt{Type} \text{ kind}} \tag{C.39d}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\Omega \vdash^{\mathbb{C}} \mathtt{prS}(\grave{\tau}) \rightsquigarrow \mathtt{S}(\tau) \text{ kind}} \tag{C.39e}$$

$$\frac{\mathsf{parseUKind}(\mathsf{subseq}(b; m; n)) = \hat{\kappa} \qquad \hat{\Omega} \vdash \hat{\kappa} \rightsquigarrow \kappa \text{ kind} \qquad \hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle \qquad \mathrm{dom}(\Omega) \cap \mathrm{dom}(\Omega_{\text{app}}) = \emptyset}{\Omega \vdash^{\omega:\Omega_{\text{params}}; \hat{\Omega}; b} \mathtt{splicedk}[m; n] \rightsquigarrow \kappa \text{ kind}} \tag{C.39f}$$

$\boxed{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \kappa}$ $\grave{c}$ has expansion c of kind κ

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \kappa_1 \qquad \Omega \vdash \kappa_1 <:: \kappa_2}{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \kappa_2} \tag{C.40a}$$

$$\frac{}{\Omega, u :: \kappa \vdash^{\mathbb{C}} u \rightsquigarrow u :: \kappa} \tag{C.40b}$$

$$\frac{\Omega, u :: \kappa_1 \vdash^{\mathbb{C}} \grave{c}_2 \rightsquigarrow c_2 :: \kappa_2}{\Omega \vdash^{\mathbb{C}} \mathtt{prabs}(u.\grave{c}_2) \rightsquigarrow \mathtt{abs}(u.c_2) :: \mathtt{darr}(\kappa_1; u.\kappa_2)} \tag{C.40c}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{c}_1 \rightsquigarrow c_1 :: \mathtt{darr}(\kappa_2; u.\kappa) \qquad \Omega \vdash^{\mathbb{C}} \grave{c}_2 \rightsquigarrow c_2 :: \kappa_2}{\Omega \vdash^{\mathbb{C}} \mathtt{prapp}(\grave{c}_1; \grave{c}_2) \rightsquigarrow \mathtt{app}(c_1; c_2) :: [c_1/u]\kappa} \tag{C.40d}$$

$$\frac{}{\Omega \vdash^{\mathbb{C}} \mathtt{prtriv} \rightsquigarrow \mathtt{triv} :: \mathtt{unit}} \tag{C.40e}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{c}_1 \rightsquigarrow c_1 :: \kappa_1 \qquad \Omega \vdash^{\mathbb{C}} \grave{c}_2 \rightsquigarrow c_2 :: [c_1/u]\kappa_2}{\Omega \vdash^{\mathbb{C}} \mathtt{prpair}(\grave{c}_1; \grave{c}_2) \rightsquigarrow \mathtt{pair}(c_1; c_2) :: \mathtt{dprod}(\kappa_1; u.\kappa_2)} \tag{C.40f}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\Omega \vdash^{\mathbb{C}} \mathtt{prprl}(\grave{c}) \rightsquigarrow \mathtt{prl}(c) :: \kappa_1} \tag{C.40g}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \mathtt{dprod}(\kappa_1; u.\kappa_2)}{\Omega \vdash^{\mathbb{C}} \mathtt{prprr}(\grave{c}) \rightsquigarrow \mathtt{prr}(c) :: [\mathtt{prl}(c)/u]\kappa_2} \tag{C.40h}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{\tau}_1 \rightsquigarrow \tau_1 :: \mathtt{Type} \qquad \Omega \vdash^{\mathbb{C}} \grave{\tau}_2 \rightsquigarrow \tau_2 :: \mathtt{Type}}{\Omega \vdash^{\mathbb{C}} \mathtt{prparr}(\grave{\tau}_1; \grave{\tau}_2) \rightsquigarrow \mathtt{parr}(\tau_1; \tau_2) :: \mathtt{Type}} \tag{C.40i}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{\kappa} \rightsquigarrow \kappa \ \mathsf{kind} \qquad \Omega, u :: \kappa \vdash^{\mathbb{C}} \grave{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\Omega \vdash^{\mathbb{C}} \mathtt{prall}\{\grave{\kappa}\}(u.\grave{\tau}) \rightsquigarrow \mathtt{all}\{\kappa\}(u.\tau) :: \mathtt{Type}} \tag{C.40j}$$

$$\frac{\Omega, t :: \mathtt{Type} \vdash^{\mathbb{C}} \grave{\tau} \rightsquigarrow \tau :: \mathtt{Type}}{\Omega \vdash^{\mathbb{C}} \mathtt{prrec}(t.\grave{\tau}) \rightsquigarrow \mathtt{rec}(t.\tau) :: \mathtt{Type}} \tag{C.40k}$$

$$\frac{\{\Omega \vdash^{\mathbb{C}} \grave{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{1 \leq i \leq n}}{\Omega \vdash^{\mathbb{C}} \mathtt{prprod}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L}) \rightsquigarrow \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) :: \mathtt{Type}} \tag{C.40l}$$

$$\frac{\{\Omega \vdash^{\mathbb{C}} \grave{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{1 \leq i \leq n}}{\Omega \vdash^{\mathbb{C}} \mathtt{prsum}[L](\{i \hookrightarrow \grave{\tau}_i\}_{i \in L}) \rightsquigarrow \mathtt{sum}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) :: \mathtt{Type}} \tag{C.40m}$$

$$\frac{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \mathtt{Type}}{\Omega \vdash^{\mathbb{C}} \grave{c} \rightsquigarrow c :: \mathtt{S}(c)} \tag{C.40n}$$

$$\frac{}{\Omega, X : \mathtt{sig}\{\kappa\}(u.\tau) \vdash^{\mathbb{C}} \mathtt{prcon}(X) \rightsquigarrow \mathtt{con}(X) :: \kappa} \tag{C.40o}$$

$$\frac{\begin{array}{c}\mathbb{C} = \omega : \Omega_{\mathsf{params}}; \hat{\Omega}; b \qquad \Omega_{\mathsf{params}} \vdash^{\mathbb{C}} \grave{\kappa} \rightsquigarrow \kappa \ \mathsf{kind} \\ \mathsf{parseUCon}(\mathsf{subseq}(b; m; n)) = \hat{c} \qquad \hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: [\omega]\kappa \\ \hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\mathsf{app}} \rangle \qquad \mathrm{dom}(\Omega) \cap \mathrm{dom}(\Omega_{\mathsf{app}}) = \emptyset\end{array}}{\Omega \vdash^{\mathbb{C}} \mathtt{splicedc}[m; n; \grave{\kappa}] \rightsquigarrow c :: \kappa} \tag{C.40p}$$

Proto-Expression and Proto-Rule Validation

$\boxed{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau}$ $\grave{e}$ has expansion e of type τ

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau \qquad \Omega \vdash \tau <: \tau'}{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'} \tag{C.41a}$$

$$\frac{}{\Omega, x : \tau \vdash^{\mathbb{E}} x \rightsquigarrow x : \tau} \tag{C.41b}$$

$$\frac{\Omega \vdash^{\mathsf{cs}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau :: \mathtt{Type} \qquad \Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau}{\Omega \vdash^{\mathbb{E}} \mathtt{prasc}\{\grave{\tau}\}(\grave{e}) \rightsquigarrow e : \tau} \tag{C.41c}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e}_1 \rightsquigarrow e_1 : \tau_1 \qquad \Omega, x : \tau_1 \vdash^{\grave{e}_2} e_2 \rightsquigarrow \tau_2 :}{\Omega \vdash^{\mathbb{E}} \mathtt{prletval}(\grave{e}_1; x.\grave{e}_2) \rightsquigarrow \mathtt{ap}(\mathtt{lam}\{\tau_1\}(x.e_2); e_1) : \tau_2} \tag{C.41d}$$

$$\frac{\Omega \vdash^{\mathsf{cs}(\mathbb{E})} \grave{\tau}_1 \rightsquigarrow \tau_1 :: \mathtt{Type} \qquad \Omega, x : \tau_1 \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau_2}{\Omega \vdash^{\mathbb{E}} \mathtt{prlam\{\grave{\tau}_1\}(x.\grave{e})} \rightsquigarrow \mathtt{lam\{\tau_1\}(x.e)} : \mathtt{parr}(\tau_1; \tau_2)} \tag{C.41e}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e}_1 \rightsquigarrow e_1 : \mathtt{parr}(\tau_2; \tau) \qquad \Omega \vdash^{\mathbb{E}} \grave{e}_2 \rightsquigarrow e_2 : \tau_2}{\Omega \vdash^{\mathbb{E}} \mathtt{prap}(\grave{e}_1; \grave{e}_2) \rightsquigarrow \mathtt{ap}(e_1; e_2) : \tau} \tag{C.41f}$$

$$\frac{\Omega \vdash^{\mathsf{cs}(\mathbb{E})} \grave{\kappa} \rightsquigarrow \kappa \ \mathsf{kind} \qquad \Omega, u :: \kappa \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau}{\Delta\ \Gamma \vdash^{\mathbb{E}} \mathtt{prclam\{\grave{\kappa}\}(u.\grave{e})} \rightsquigarrow \mathtt{clam\{\kappa\}(u.e)} \Rightarrow \mathtt{all\{\kappa\}(u.\tau)}} \tag{C.41g}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{all\{\kappa\}(u.\tau)} \qquad \Omega \vdash^{\mathsf{cs}(\mathbb{E})} \grave{c} \rightsquigarrow c :: \kappa}{\Omega \vdash^{\mathbb{E}} \mathtt{prcap\{\grave{c}\}(\grave{e})} \rightsquigarrow \mathtt{cap\{c\}(e)} : [c/u]\tau} \tag{C.41h}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : [\mathtt{rec}(t.\tau)/t]\tau}{\Omega \vdash^{\mathbb{E}} \mathtt{prfold}(\grave{e}) \rightsquigarrow \mathtt{fold}(e) : \mathtt{rec}(t.\tau)} \tag{C.41i}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{rec}(t.\tau)}{\Omega \vdash^{\mathbb{E}} \mathtt{prunfold}(\grave{e}) \rightsquigarrow \mathtt{unfold}(e) : [\mathtt{rec}(t.\tau)/t]\tau} \tag{C.41j}$$

$$\frac{\begin{array}{c}\tau = \mathtt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \\ \{\Omega \vdash^{\mathbb{E}} \grave{e}_i \rightsquigarrow e_i : \tau_i\}_{i \in L}\end{array}}{\Omega \vdash^{\mathbb{E}} \mathtt{prtpl\{L\}(\{i \hookrightarrow \grave{e}_i\}_{i \in L})} \rightsquigarrow \mathtt{tpl}[L](\{i \hookrightarrow e_i\}_{i \in L}) : \tau} \tag{C.41k}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \mathtt{prod}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau)}{\Omega \vdash^{\mathbb{E}} \mathtt{prprj}[\ell](\grave{e}) \rightsquigarrow \mathtt{prj}[\ell](e) : \tau} \tag{C.41l}$$

$$\frac{\begin{array}{c}\mathtt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau') \\ \Omega \vdash^{\mathbb{E}} \grave{e}' \rightsquigarrow e' : \tau'\end{array}}{\Omega \vdash^{\mathbb{E}} \mathtt{prinj}[\ell](\grave{e}') \rightsquigarrow \mathtt{inj}[\ell](e') : \tau} \tag{C.41m}$$

$$\frac{\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau \qquad \{\Omega \vdash^{\mathbb{E}} \grave{r}_i \rightsquigarrow r_i : \tau \Rightarrow \tau'\}_{1 \leq i \leq n}}{\Omega \vdash^{\mathbb{E}} \mathtt{prmatch}[n](\grave{e}; \{\grave{r}_i\}_{1 \leq i \leq n}) \rightsquigarrow \mathtt{match}[n](e; \{r_i\}_{1 \leq i \leq n}) : \tau'} \tag{C.41n}$$

$$\frac{}{\Omega, X : \mathtt{sig\{\kappa\}(u.\tau)} \vdash^{\mathbb{E}} \mathtt{prval}(X) \rightsquigarrow \mathtt{val}(X) : [\mathtt{con}(X)/u]\tau} \tag{C.41o}$$

$$\frac{\begin{array}{c}\mathbb{E} = \omega : \Omega_{\text{params}}; \hat{\Omega}; \hat{\Psi}; \hat{\Phi}; b \qquad \Omega_{\text{params}} \vdash^{\mathsf{cs}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau :: \mathtt{Type} \\ \mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{e} \qquad \hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [\omega]\tau \\ \hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle \qquad \mathrm{dom}(\Omega) \cap \mathrm{dom}(\Omega_{\text{app}}) = \emptyset\end{array}}{\Omega \vdash^{\mathbb{E}} \mathtt{splicede}[m; n; \grave{\tau}] \rightsquigarrow e : \tau} \tag{C.41p}$$

$\boxed{\Omega \vdash^{\mathbb{E}} \grave{r} \rightsquigarrow r : \tau \Rightarrow \tau'}$ $\grave{r}$ has expansion r taking values of type τ to values of type τ'

$$\frac{\Omega \vdash p : \tau \dashv\!\mid \Omega' \qquad \Omega \cup \Omega' \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'}{\Omega \vdash^{\mathbb{E}} \mathtt{prrule}(p.\grave{e}) \rightsquigarrow \mathtt{rule}(p.e) : \tau \Rightarrow \tau'} \tag{C.42}$$

Proto-Pattern Validation

$\boxed{\grave{p} \rightsquigarrow p : \tau \dashv\!\|^{\mathbb{P}} \hat{\Omega}}$ $\grave{p}$ has expansion p matching against τ generating hypotheses $\hat{\Omega}$

$$\frac{}{\texttt{prwildp} \rightsquigarrow \texttt{wildp} : \tau \dashv\!\|^{\mathbb{P}} \langle \emptyset; \emptyset; \emptyset; \emptyset \rangle} \tag{C.43a}$$

$$\frac{\grave{p} \rightsquigarrow p : [\texttt{rec}(t.\tau)/t]\tau \dashv\!\|^{\mathbb{P}} \hat{\Omega}}{\texttt{prfoldp}(\grave{p}) \rightsquigarrow \texttt{foldp}(p) : \texttt{rec}(t.\tau) \dashv\!\|^{\mathbb{P}} \hat{\Omega}} \tag{C.43b}$$

$$\frac{\grave{p} = \texttt{prtplp}[L](\{i \hookrightarrow \grave{p}_i\}_{i \in L}) \qquad p = \texttt{tplp}[L](\{i \hookrightarrow p_i\}_{i \in L}) \qquad \{\grave{p}_i \rightsquigarrow p_i : \tau_i \dashv\!\|^{\mathbb{P}} \hat{\Gamma}_i\}_{i \in L}}{\grave{p} \rightsquigarrow p : \texttt{prod}[L](\{i \hookrightarrow \tau_i\}_{i \in L}) \dashv\!\|^{\mathbb{P}} \uplus_{i \in L} \hat{\Omega}_i} \tag{C.43c}$$

$$\frac{\grave{p} \rightsquigarrow p : \tau \dashv\!\|^{\mathbb{P}} \hat{\Omega}}{\texttt{prinjp}[\ell](\grave{p}) \rightsquigarrow \texttt{injp}[\ell](p) : \texttt{sum}[L, \ell](\{i \hookrightarrow \tau_i\}_{i \in L}; \ell \hookrightarrow \tau) \dashv\!\|^{\mathbb{P}} \hat{\Omega}} \tag{C.43d}$$

$$\frac{\Omega_{\text{params}} \vdash^{\omega:\Omega_{\text{params}}; \hat{\Omega}; b} \grave{\tau} \rightsquigarrow \tau :: \texttt{Type} \qquad \mathsf{parseUPat}(\mathsf{subseq}(b; m; n)) = \hat{p} \qquad \hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : [\omega]\tau \dashv\! | \, \hat{\Omega}'}{\texttt{splicedp}[m; n; \grave{\tau}] \rightsquigarrow p : \tau \dashv\!\|^{\omega:\Omega_{\text{params}}; \hat{\Omega}; \hat{\Phi}; b} \hat{\Omega}'} \tag{C.43e}$$

C.4 Metatheory

C.4.1 TSM Expressions

Lemma C.22 (peTSM Regularity). *If* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *then* $\Omega \vdash \rho$ tsmty.

Proof. By rule induction over Rules (C.26).

Case (C.26a).

(1) $\Omega \vdash \rho$ tsmty — by assumption

Case (C.26b).

(1) $\epsilon = \texttt{absmod}\{\sigma\}(X.\epsilon')$ — by assumption
(2) $\rho = \texttt{allmods}\{\sigma\}(X.\rho')$ — by assumption
(3) $\Omega, X : \sigma \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} \rho'$ — by assumption
(4) $\Omega \vdash \sigma$ sig — by assumption
(5) $\Omega, X : \sigma \vdash \rho'$ tsmty — by IH on (3)
(6) $\Omega \vdash \texttt{allmods}\{\sigma\}(X.\rho')$ tsmty — by Rule (C.25b) on (4) and (5)

Case (C.26c)**.**

(1)	$\epsilon = \texttt{apmod}\{X\}(\epsilon')$	by assumption
(2)	$\rho = [X/X']\rho'$	by assumption
(3)	$\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \texttt{allmods}\{\sigma\}(X'.\rho')$	by assumption
(4)	$\Omega \vdash X : \sigma$	by assumption
(5)	$\Omega \vdash \texttt{allmods}\{\sigma\}(X'.\rho')$ tsmty	by IH on (3)
(6)	$\Omega, X' : \sigma \vdash \rho'$ tsmty	by Inversion of Rule (C.25b) on (5)
(7)	$\Omega \vdash [X'/X]\rho'$ tsmty	by Substitution Lemma C.3 on (4) and (6)

□

Lemma C.23 (ppTSM Regularity)**.** *If* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *then* $\Omega \vdash \rho$ tsmty*.*

Proof. By rule induction over Rules (C.27). The proof is nearly identical to the proof of Lemma C.22, differing only in that ppTSM contexts and the corresponding judgements are mentioned. □

Lemma C.24 (peTSM Unicity)**.** *If* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho'$ *then* $\rho = \rho'$*.*

Proof. By rule induction over Rules (C.26). The rules are syntax-directed, so the proof is by straightforward observations of syntactic contradictions. □

Lemma C.25 (ppTSM Unicity)**.** *If* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho'$ *then* $\rho = \rho'$*.*

Proof. By rule induction over Rules (C.27). The rules are syntax-directed, so the proof is by straightforward observations of syntactic contradictions. □

Theorem C.26 (peTSM Preservation)**.** *If* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto \epsilon'$ *then* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} \rho$*.*

Proof. By rule induction over Rules (C.29).

Case (C.29a)**.** By rule induction over Rules (C.26). There is only one rule that applies.

Case (C.26c)**.**

(1)	$\epsilon = \texttt{apmod}\{X\}(\epsilon'')$	by assumption
(2)	$\epsilon' = \texttt{apmod}\{X\}(\epsilon''')$	by assumption
(3)	$\rho = [X/X']\rho'$	by assumption
(4)	$\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'' \mapsto \epsilon'''$	by assumption
(5)	$\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'' \mathbin{@} \texttt{allmods}\{\sigma\}(X'.\rho')$	by assumption

(6) $\Omega \vdash X : \sigma$ by assumption

(7) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon''' \mathbin{@} \texttt{allmods\{}\sigma\texttt{\}(}X'.\rho'\texttt{)}$ by IH on (5) and (4)

(8) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{apmod\{}X\texttt{\}(}\epsilon'''\texttt{)} \mathbin{@} [X/X']\rho$ by Rule (C.26c) on (7) and (6)

Case (C.29b)**.** By rule induction over Rules (C.26). There is only one rule that applies.

Case (C.26c)**.**

(1) $\epsilon = \texttt{apmod\{}X\texttt{\}(absmod\{}\sigma\texttt{\}(}X'.\epsilon''\texttt{))}$ by assumption

(2) $\epsilon' = [X/X']\epsilon''$ by assumption

(3) $\rho = [X/X']\rho'$ by assumption

(4) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{absmod\{}\sigma\texttt{\}(}X'.\epsilon''\texttt{)} \mathbin{@} \texttt{allmods\{}\sigma\texttt{\}(}X'.\rho'\texttt{)}$ by assumption

(5) $\Omega \vdash X : \sigma$ by assumption

(6) $\Omega, X' : \sigma \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'' \mathbin{@} \rho'$ by Inversion Lemma for Rule (C.26b) on (4)

(7) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} [X/X']\epsilon'' \mathbin{@} [X/X']\rho'$ by Substitution Lemma C.3 on (6)

□

Corollary C.27 (peTSM Preservation (Multistep))**.** *If* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto^* \epsilon'$ *then* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} \rho$*.*

Proof. The multistep relation is the reflexive, transitive closure of the single step relation, so the proof follows by applying Theorem C.26 over each step. □

Corollary C.28 (peTSM Preservation (Evaluation))**.** *If* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon'$ *then* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} \rho$*.*

Proof. The evaluation relation is the multistep relation with an additional requirement, so the proof follows directly from Corollary C.27. □

Theorem C.29 (ppTSM Preservation)**.** *If* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mapsto \epsilon'$ *then* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon' \mathbin{@} \rho$*.*

Proof. The proof is nearly identical to the proof of Theorem C.26, differing only in that ppTSM contexts and the corresponding judgements are mentioned. □

Corollary C.30 (ppTSM Preservation (Multistep))**.** *If* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mapsto^* \epsilon'$ *then* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon' \mathbin{@} \rho$*.*

Proof. The multistep relation is the reflexive, transitive closure of the single step relation, so the proof follows by applying Theorem C.29 over each step. □

Corollary C.31 (ppTSM Preservation (Evaluation)). *If* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *and* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \Downarrow \epsilon'$ *then* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon' \mathbin{@} \rho$.

Proof. The evaluation relation is the multistep relation with an additional requirement, so the proof follows directly from Corollary C.30. □

Theorem C.32 (peTSM Progress). *If* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ *then either* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mapsto \epsilon'$ *for some* ϵ' *or* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon$ normal.

Proof. By rule induction over Rules (C.26).

Case (C.26a).

(1) $\epsilon = \texttt{defref}[a]$ — by assumption
(2) $\Psi = \Psi', a \hookrightarrow \texttt{petsm}(\rho; e_{\text{parse}})$ — by assumption
(3) $\Omega \vdash^{\mathsf{Exp}}_{\Psi', a \hookrightarrow \texttt{petsm}(\rho; e_{\text{parse}})} \texttt{defref}[a]$ normal — by Rule (C.35a)

Case (C.26b).

(1) $\epsilon = \texttt{absmod}\{\sigma\}(X.\epsilon')$ — by assumption
(2) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{absmod}\{\sigma\}(X.\epsilon')$ normal — by Rule (C.35b)

Case (C.26c).

(1) $\epsilon = \texttt{apmod}\{X\}(\epsilon')$ — by assumption
(2) $\rho = [X/X']\rho'$ — by assumption
(3) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} \texttt{allmods}\{\sigma\}(X'.\rho')$ — by assumption
(4) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mapsto \epsilon''$ for some ϵ'' or $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'$ normal — by IH on (3)

Proceed by cases on (4).

Case $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mapsto \epsilon''$.

(5) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mapsto \epsilon''$ — by assumption
(6) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{apmod}\{X\}(\epsilon') \mapsto \texttt{apmod}\{X\}(\epsilon'')$ — by Rule (C.29a) on (5)

Case $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'$ normal. Proceed by rule induction over Rules (C.35).

Case (C.35a).

(7) $\epsilon' = \texttt{defref}[a]$ — by assumption
(8) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'$ normal — by assumption
(9) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{apmod}\{X\}(\epsilon')$ normal — by Rule (C.35c) on (8)

Case (C.35b).

(10) $\epsilon' = \texttt{absmod}\{\sigma\}(X.\epsilon'')$ by assumption

(11) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{apmod}\{X\}(\texttt{absmod}\{\sigma\}(X'.\epsilon'')) \mapsto [X/X']\epsilon''$ by Rule (C.29b)

Case (C.35c).

(12) $\epsilon' = \texttt{apmod}\{X''\}(\epsilon'')$ by assumption

(13) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'$ normal by assumption

(14) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \texttt{apmod}\{X\}(\epsilon')$ normal by Rule (C.35c) on (8)

□

Theorem C.33 (ppTSM Progress). *If* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mathbin{@} \rho$ *then either* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \mapsto \epsilon'$ *for some* ϵ' *or* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon$ normal.

Proof. The proof is nearly identical to the proof of Theorem C.32, differing only in that ppTSM contexts and the corresponding judgements are mentioned. □

C.4.2 Typed Expansion

Kinds, Constructions and Signatures

Theorem C.34 (Kind and Construction Expansion).

1. *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash \hat{\kappa} \rightsquigarrow \kappa$ kind *then* $\Omega \vdash \kappa$ kind.
2. *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash \hat{c} \rightsquigarrow c :: \kappa$ *then* $\Omega \vdash c :: \kappa$.

Proof. By mutual rule induction over Rules (C.17) and Rules (C.18). In each case, we apply the IH to each premise and then apply the corresponding kind formation rule from Rules (C.6) or kinding rule from Rules (C.9). □

Theorem C.35 (Signature Expansion). *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash \hat{\sigma} \rightsquigarrow \sigma$ sig *then* $\Omega \vdash \sigma$ sig.

Proof. By rule induction over Rule (C.15). Apply Theorem C.34 to each premise, then apply Rule (C.1). □

TSM Types and Expressions

Theorem C.36 (TSM Type Expansion). *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash \hat{\rho} \rightsquigarrow \rho$ tsmty *then* $\Omega \vdash \rho$ tsmty.

Proof. By rule induction over Rules (C.22).

Case (C.22a).

(1) $\hat{\rho} = \hat{\tau}$ by assumption

(2) $\rho = \texttt{type}(\tau)$ by assumption

(3) $\hat{\Omega} \vdash \hat{\tau} \rightsquigarrow \tau :: \texttt{Type}$ — by assumption

(4) $\Omega \vdash \tau :: \texttt{Type}$ — by Theorem C.34 on (3)

(5) $\Omega \vdash \texttt{type}(\tau)$ tsmty — by Rule (C.25a) on (4)

Case (C.22b).

(1) $\hat{\rho} = \forall \hat{X}{:}\hat{\sigma}.\hat{\rho}'$ — by assumption

(2) $\rho = \texttt{allmods}\{\sigma\}(X.\rho')$ — by assumption

(3) $\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma$ sig — by assumption

(4) $\hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash \hat{\rho}' \rightsquigarrow \rho'$ tsmty — by assumption

(5) $\Omega \vdash \sigma$ sig — by Theorem C.35 on (3)

(6) $\Omega, X : \sigma \vdash \rho'$ tsmty — by IH on (4)

(7) $\Omega \vdash \texttt{allmods}\{\sigma\}(X.\rho')$ tsmty — by Rule (C.25b) on (5) and (6)

□

Theorem C.37 (peTSM Expression Expansion). *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash^{\mathsf{Exp}}_{\langle \mathcal{A}; \Psi \rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho$ *then* $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$.

Proof. By rule induction over Rules (C.23). In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$ and $\hat{\Psi} = \langle \mathcal{A}; \Psi \rangle$.

Case (C.23a).

(1) $\hat{\epsilon} = \hat{a}$ — by assumption

(2) $\mathcal{A} = \mathcal{A}', \hat{a} \hookrightarrow \epsilon$ — by assumption

(3) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \rho$ — by assumption

Case (C.23b).

(1) $\hat{\epsilon} = \Lambda \hat{X}{:}\hat{\sigma}.\hat{\epsilon}'$ — by assumption

(2) $\epsilon = \texttt{absmod}\{\sigma\}(X.\epsilon')$ — by assumption

(3) $\rho = \texttt{allmods}\{\sigma\}(X.\rho')$ — by assumption

(4) $\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma$ sig — by assumption

(5) $\hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{\epsilon}' \rightsquigarrow \epsilon' \mathbin{@} \rho'$ — by assumption

(6) $\Omega, X : \sigma \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} \rho'$ — by IH on (4) and (5)

(7) $\Omega \vdash \sigma\ \mathsf{sig}$ — by Theorem C.35 on (4)

(8) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{absmod\{}\sigma\mathtt{\}(}X.\epsilon'\mathtt{)}\ @\ \mathtt{allmods\{}\sigma\mathtt{\}(}X.\rho'\mathtt{)}$ — by Rule (C.26b) on (7) and (6)

Case (C.23c).

(1) $\hat{\epsilon} = \hat{\epsilon}'(\hat{X})$ — by assumption

(2) $\epsilon = \mathtt{apmod\{}X\mathtt{\}(}\epsilon'\mathtt{)}$ — by assumption

(3) $\rho = [X/X']\rho'$ — by assumption

(4) $\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{\epsilon}' \rightsquigarrow \epsilon'\ @\ \mathtt{allmods\{}\sigma\mathtt{\}(}X'.\rho'\mathtt{)}$ — by assumption

(5) $\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{X} \rightsquigarrow X : \sigma$ — by assumption

(6) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon'\ @\ \mathtt{allmods\{}\sigma\mathtt{\}(}X'.\rho'\mathtt{)}$ — by IH on (4)

(7) $\Omega \vdash X : \sigma$ — by Theorem C.43 on (5)

(8) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \mathtt{apmod\{}X\mathtt{\}(}\epsilon'\mathtt{)}\ @\ [X/X']\rho'$ — by Rule (C.26c) on (6) and (7)

□

Theorem C.38 (ppTSM Expression Expansion). *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash^{\mathsf{Pat}}_{\langle \mathcal{A}; \Phi \rangle} \hat{\epsilon} \rightsquigarrow \epsilon\ @\ \rho$ *then* $\Omega \vdash^{\mathsf{Pat}}_{\Phi} \epsilon\ @\ \rho$.

Proof. The proof is nearly identical to the proof of Theorem C.37, differing only in that ppTSM contexts and the corresponding judgements are mentioned. □

Patterns

Lemma C.39 (Proto-Pattern Deparameterization). *If* $\Omega_{app} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p}\ \mathcal{Y}\!\!\mapsto_{\epsilon} \hat{p}\ ?\ \rho \dashv \omega : \Omega_{params}$ *then* $dom(\Omega_{app}) \cap dom(\Omega_{params}) = \emptyset$ *and* $\Omega_{app} \vdash \omega : \Omega_{params}$ *and* $\Omega_{app} \vdash^{\mathsf{Pat}}_{\Phi} \epsilon\ @\ [\omega]\rho$.

Proof. By rule induction over Rules (C.38).

Case (C.38a). We have:

(1) $\epsilon = \mathtt{defref[}a\mathtt{]}$ — by assumption

(2) $\omega = \emptyset$ — by assumption

(3) $\Phi = \Phi', a \hookrightarrow \mathtt{pptsm}(\rho; e_{\text{parse}})$ — by assumption

(4) $\Omega_{\text{params}} = \emptyset$ — by assumption

(5) $\Omega_{\text{app}} \vdash \rho\ \mathsf{tsmty}$ — by assumption

(6) $\mathrm{dom}(\Omega_{\text{app}}) \cap \mathrm{dom}(\emptyset) = \emptyset$ — by definition

(7) $\Omega_{\text{app}} \vdash \emptyset : \emptyset$ — by definition

(8) $[\emptyset]\rho = \rho$ — by definition

(9) $\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi', a \hookrightarrow \texttt{pptsm}(\rho; e_{\text{parse}})} a \mathbin{@} \rho$ — by Rule (C.27a) on (5)

Case (C.38b)**.** We have:

(1) $\epsilon = \texttt{apmod\{}X\texttt{\}(}\epsilon'\texttt{)}$ — by assumption

(2) $\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p} \rightarrowtail_{\epsilon'} \hat{p}\ \texttt{?}\ \texttt{allmods\{}\sigma\texttt{\}(}X'.\rho\texttt{)} \dashv \omega' : \Omega'$ — by assumption

(3) $\Omega_{\text{app}} \vdash X : \sigma$ — by assumption

(4) $X' \notin \mathrm{dom}(\Omega_{\text{app}})$ — by assumption

(5) $\omega = \omega', X/X'$ — by assumption

(6) $\Omega_{\text{params}} = \Omega', X' : \sigma$ — by assumption

(7) $\mathrm{dom}(\Omega_{\text{app}}) \cap \mathrm{dom}(\Omega') = \emptyset$ — by IH on (2)

(8) $\Omega_{\text{app}} \vdash \omega' : \Omega'$ — by IH on (2)

(9) $\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \epsilon' \mathbin{@} [\omega']\texttt{allmods\{}\sigma\texttt{\}(}X'.\rho\texttt{)}$ — by IH on (2)

(10) $\mathrm{dom}(\Omega_{\text{app}}) \cap \mathrm{dom}(\Omega', X' : \sigma)$ — by (4) and (7) and definition of finite set intersection

(11) $\Omega_{\text{app}} \vdash \omega', X/X' : \Omega', X' : \sigma$ — by Definition C.2 on (3)

(12) $\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \texttt{apmod\{}X\texttt{\}(}\epsilon'\texttt{)} \mathbin{@} (\omega', X/X')\rho$ — by Rule (C.27c) on (9) and (3)

□

Theorem C.40 (Typed Pattern Expansion)**.**

1. *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{app} \rangle \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid \langle \mathcal{M}'; \mathcal{D}'; \mathcal{G}'; \Omega' \rangle$ *then* $\mathcal{M}' = \emptyset$ *and* $\mathcal{D}' = \emptyset$ *and* $\Omega_{app} \vdash p : \tau \dashv\!\shortmid \Omega'$.

2. *If* $\hat{p} \rightsquigarrow p : \tau \dashv\!\shortmid^{\omega : \Omega_{params}; \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{app} \rangle; \hat{\Phi}; b} \langle \mathcal{M}'; \mathcal{D}'; \mathcal{G}'; \Omega' \rangle$ *and* $dom(\Omega_{params}) \cap dom(\Omega_{app}) = \emptyset$ *then* $\mathcal{M}' = \emptyset$ *and* $\mathcal{D}' = \emptyset$ *and* $\Omega_{params} \cup \Omega_{app} \vdash p : \tau \dashv\!\shortmid \Omega'$.

Proof. My mutual rule induction over Rules (C.21) and Rules (C.43).

1. In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle$ and $\hat{\Omega}' = \langle \mathcal{M}'; \mathcal{D}'; \mathcal{G}'; \Omega' \rangle$.

   Case (C.21a) **through** (C.21f)**.** These cases follow by applying the IH, part 1 and applying the corresponding pattern typing rule in Rules (C.14).

Case (C.21g)**.** We have:

(1) $\hat{p} = \hat{\epsilon}\ \text{`}b\text{`}$ — by assumption
(2) $\hat{\Phi} = \langle \mathcal{A}; \Phi \rangle$ — by assumption
(3) $\hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \hat{\epsilon} \rightsquigarrow \epsilon \ @\ \mathtt{type}(\tau_{\text{final}})$ — by assumption
(4) $\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \epsilon \Downarrow \epsilon_{\text{normal}}$ — by assumption
(5) $\mathsf{tsmdef}(\epsilon_{\text{normal}}) = a$ — by assumption
(6) $\Phi = \Phi', a \hookrightarrow \mathtt{pptsm}(\rho; e_{\text{parse}})$ — by assumption
(7) $b \downarrow_{\mathsf{Body}} e_{\text{body}}$ — by assumption
(8) $e_{\text{parse}}(e_{\text{body}}) \Downarrow \mathtt{inj}[\mathtt{SuccessP}](e_{\text{proto}})$ — by assumption
(9) $e_{\text{proto}} \uparrow_{\mathsf{PPrPat}} \dot{p}$ — by assumption
(10) $\Omega_{\text{app}} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p} \hookrightarrow_{\epsilon_{\text{normal}}} \dot{p}\ ?\ \mathtt{type}(\tau_{\text{proto}}) \dashv \omega : \Omega_{\text{params}}$ — by assumption
(11) $\dot{p} \rightsquigarrow p : \tau_{\text{proto}} \dashv\!\vert^{\omega:\Omega_{\text{params}}; \hat{\Omega}; \hat{\Phi}; b} \hat{\Omega}'$ — by assumption
(12) $\tau = [\omega]\tau_{\text{proto}}$ — by assumption
(13) $\mathrm{dom}(\Omega_{\text{params}}) \cap \mathrm{dom}(\Omega_{\text{app}}) = \emptyset$ — by Lemma C.39 on (10)
(14) $\Omega_{\text{app}} \vdash \omega : \Omega_{\text{params}}$ — by Lemma C.39 on (10)
(15) $\mathcal{M}' = \emptyset$ and $\mathcal{D}' = \emptyset$ — by IH, part 2 on (11) and (13)
(16) $\Omega_{\text{params}} \cup \Omega_{\text{app}} \vdash p : \tau_{\text{proto}} \dashv\!\vert \Omega'$ — by IH, part 2 on (11) and (13)
(17) $\Omega_{\text{app}} \vdash p : [\omega]\tau_{\text{proto}} \dashv\!\vert \Omega'$ — by Substitution Lemma C.3 on (14) and (16)

2. We induct on the premise. In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle$ and $\hat{\Omega}' = \langle \mathcal{M}'; \mathcal{D}'; \mathcal{G}'; \Omega' \rangle$.

Case (C.43a) **through** (C.43d)**.** These cases follow by applying the IH, part 2 and then applying the corresponding pattern rule in Rules (C.14).

Case (C.43e)**.**

(1) $\dot{p} = \mathtt{splicedp}[m; n; \dot{\tau}]$ — by assumption
(2) $\tau = [\omega]\tau'$ — by assumption
(3) $\Omega_{\text{params}} \vdash^{\omega:\Omega_{\text{params}}; \hat{\Omega}; b} \dot{\tau} \rightsquigarrow \tau' :: \mathtt{Type}$ — by assumption
(4) $\mathsf{parseUPat}(\mathsf{subseq}(b; m; n)) = \hat{p}$ — by assumption
(5) $\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : [\omega]\tau' \dashv\!\vert \hat{\Omega}'$ — by assumption
(6) $\mathcal{M}' = \emptyset$ and $\mathcal{D}' = \emptyset$ — by IH, part 1 on (5)
(7) $\Omega_{\text{app}} \vdash p : [\omega]\tau' \dashv\!\vert \Omega'$ — by IH, part 1 on (5)

(8) $\Omega_{\text{params}} \cup \Omega_{\text{app}} \vdash p : [\omega]\tau' \dashv\mid \Omega'$ by Weakening on (7)

The mutual induction can be shown to be well-founded by an argument analagous to that in the proof of Theorem B.26, appealing to Condition C.13 and Condition C.15.

□

Expressions and Rules

Lemma C.41 (Proto-Expression Deparameterization). *If* $\Omega_{app} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \leftarrow\!\!\!\!\rightarrow_{\epsilon} \grave{e} \; ? \; \rho \dashv \omega : \Omega_{params}$ *then* $dom(\Omega_{app}) \cap dom(\Omega_{params}) = \emptyset$ *and* $\Omega_{app} \vdash \omega : \Omega_{params}$ *and* $\Omega_{app} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} [\omega]\rho$.

Proof. By rule induction over Rules (C.37).

Case (C.37a)**.** We have:

(1) $\epsilon = \texttt{defref[}a\texttt{]}$ by assumption
(2) $\omega = \emptyset$ by assumption
(3) $\Psi = \Psi', a \hookrightarrow \texttt{petsm}(\rho; e_{\text{parse}})$ by assumption
(4) $\Omega_{\text{params}} = \emptyset$ by assumption
(5) $\Omega_{\text{app}} \vdash \rho$ tsmty by assumption
(6) $\text{dom}(\Omega_{\text{app}}) \cap \text{dom}(\emptyset) = \emptyset$ by definition
(7) $\Omega_{\text{app}} \vdash \emptyset : \emptyset$ by definition
(8) $[\emptyset]\rho = \rho$ by definition
(9) $\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi', a \hookrightarrow \texttt{petsm}(\rho; e_{\text{parse}})} a \mathbin{@} \rho$ by Rule (C.26a) on (5)

Case (C.37b)**.** We have:

(1) $\epsilon = \texttt{apmod\{}X\texttt{\}(}\epsilon'\texttt{)}$ by assumption
(2) $\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \leftarrow\!\!\!\!\rightarrow_{\epsilon'} \grave{e} \; ? \; \texttt{allmods\{}\sigma\texttt{\}(}X'.\rho\texttt{)} \dashv \omega' : \Omega'$ by assumption
(3) $\Omega_{\text{app}} \vdash X : \sigma$ by assumption
(4) $X' \notin \text{dom}(\Omega_{\text{app}})$ by assumption
(5) $\omega = \omega', X/X'$ by assumption
(6) $\Omega_{\text{params}} = \Omega', X' : \sigma$ by assumption
(7) $\text{dom}(\Omega_{\text{app}}) \cap \text{dom}(\Omega') = \emptyset$ by IH on (2)
(8) $\Omega_{\text{app}} \vdash \omega' : \Omega'$ by IH on (2)
(9) $\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon' \mathbin{@} [\omega']\texttt{allmods\{}\sigma\texttt{\}(}X'.\rho\texttt{)}$ by IH on (2)

(10) $\mathrm{dom}(\Omega_{\text{app}}) \cap \mathrm{dom}(\Omega', X' : \sigma)$ — by (4) and (7) and definition of finite set intersection

(11) $\Omega_{\text{app}} \vdash \omega', X/X' : \Omega', X' : \sigma$ — by Definition C.2 on (3)

(12) $\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \texttt{apmod\{}X\texttt{\}(}\epsilon'\texttt{)} \mathbin{@} (\omega', X/X')\rho$ — by Rule (C.26c) on (9) and (3)

□

Theorem C.42 (Typed Expression and Rule Expansion)**.**

1. (a) *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau$ *then* $\Omega \vdash e : \tau$.

   (b) *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{r} \rightsquigarrow r : \tau \Rrightarrow \tau'$ *then* $\Omega \vdash r : \tau \Rrightarrow \tau'$.

2. (a) *If* $\Omega \vdash^{\omega:\Omega_{params};\langle \mathcal{M};\mathcal{D};\mathcal{G};\Omega_{app}\rangle;\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau$ *and* $dom(\Omega) \cap dom(\Omega_{app}) = \emptyset$ *then* $\Omega \cup \Omega_{app} \vdash e : \tau$.

   (b) *If* $\Omega \vdash^{\omega:\Omega_{params};\langle \mathcal{M};\mathcal{D};\mathcal{G};\Omega_{app}\rangle;\hat{\Psi};\hat{\Phi};b} \grave{r} \rightsquigarrow r : \tau \Rrightarrow \tau'$ *and* $dom(\Omega) \cap dom(\Omega_{app}) = \emptyset$ *then* $\Omega \cup \Omega_{app} \vdash r : \tau \Rrightarrow \tau'$.

Proof. By mutual rule induction over Rules (C.19), Rule (C.20), Rules (C.41) and Rule (C.42).

1. (a) In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$.

   Case (C.19a)**.**

   (1) $\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'$ — by assumption

   (2) $\Omega \vdash \tau <: \tau'$ — by assumption

   (3) $\Omega \vdash e : \tau'$ — by IH, part 1(a) on (1)

   (4) $\Omega \vdash e : \tau$ — by Rule (C.12a) on (3) and (2)

   Case (C.19b) **through** (C.19o)**.** In each of these cases, we apply the IH, part 1(a) or 1(b), over the premises and then apply the corresponding typing rule in Rules (C.12) and weakening as needed.

   Case (C.19p)**.**

   (1) $\hat{e} = \hat{\epsilon}\ \text{`}b\text{`}$ — by assumption

   (2) $e = [\omega]e'$ — by assumption

   (3) $\tau = [\omega]\tau_{\text{proto}}$ — by assumption

   (4) $\hat{\Psi} = \langle \mathcal{A}; \Psi \rangle$ — by assumption

   (5) $\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \texttt{type(}\tau_{\text{final}}\texttt{)}$ — by assumption

   (6) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon_{\text{normal}}$ — by assumption

(7) $\mathsf{tsmdef}(\epsilon_{\text{normal}}) = a$ — by assumption
(8) $\Psi = \Psi', a \hookrightarrow \texttt{petsm}(\rho; e_{\text{parse}})$ — by assumption
(9) $b \downarrow_{\mathsf{Body}} e_{\text{body}}$ — by assumption
(10) $e_{\text{parse}}(e_{\text{body}}) \Downarrow \texttt{inj}[\texttt{SuccessE}](e_{\text{pproto}})$ — by assumption
(11) $e_{\text{pproto}} \uparrow_{\mathsf{PPrExpr}} \dot{e}$ — by assumption
(12) $\Omega \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \looparrowright_{\epsilon_{\text{normal}}} \grave{e}\ ?\ \texttt{type}(\tau_{\text{proto}}) \dashv \omega : \Omega_{\text{params}}$ — by assumption
(13) $\Omega_{\text{params}} \vdash^{\omega:\Omega_{\text{params}}; \hat{\Omega}; \hat{\Psi}; \hat{\Phi}; b} \grave{e} \rightsquigarrow e' : \tau_{\text{proto}}$ — by assumption
(14) $\Omega \vdash \omega : \Omega_{\text{params}}$ — by Lemma C.41 on (12)
(15) $\Omega \cup \Omega_{\text{params}} \vdash e' : \tau_{\text{proto}}$ — by IH, part 2(a) on (13)
(16) $\Omega \vdash [\omega]e' : [\omega]\tau_{\text{proto}}$ — by Substitution Lemma C.3 on (14) and (15)

(b) In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$.

Case (C.20).

(1) $\hat{r} = \hat{p} \Rightarrow \hat{e}$ — by assumption
(2) $r = \texttt{rule}(p.e)$ — by assumption
(3) $\hat{\Omega} \vdash_{\hat{\Phi}} \hat{p} \rightsquigarrow p : \tau \dashv\Vdash \langle \emptyset; \emptyset; \mathcal{G}'; \Omega' \rangle$ — by assumption
(4) $\langle \mathcal{M}; \mathcal{D}; \mathcal{G} \uplus \mathcal{G}'; \Omega \cup \Omega' \rangle \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{e} \rightsquigarrow e : \tau'$ — by assumption
(5) $\Omega \vdash p : \tau \dashv\Vdash \Omega'$ — by Theorem C.40 on (3)
(6) $\Omega \cup \Omega' \vdash e : \tau'$ — by IH, part 1(a) on (4)
(7) $\Omega \vdash r : \tau \Rrightarrow \tau'$ — by Rule (C.13) on (5) and (6)

2. (a) In the following, let $\mathbb{E} = \omega : \Omega_{\text{params}};\ \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle;\ \hat{\Psi};\ \hat{\Phi};\ b$.

Case (C.41a).

(1) $\Omega \vdash^{\mathbb{E}} \grave{e} \rightsquigarrow e : \tau'$ — by assumption
(2) $\Omega \vdash \tau' <: \tau$ — by assumption
(3) $\Omega \cup \Omega_{\text{app}} \vdash e : \tau'$ — by IH, part 2(a) on (1)
(4) $\Omega \cup \Omega_{\text{app}} \vdash e : \tau$ — by Rule (C.12a) on (3) and (2)

Case (C.41b) **through** (C.41o). In each of these cases, we apply the IH, part 2(a) or 2(b), over the premises and then apply the corresponding typing rule in Rules (C.12) and weakening as needed.

Case (C.41p).

(1) $\grave{e} = \texttt{splicede}[m; n; \grave{\tau}]$ — by assumption

(2) $\tau = [\omega]\tau'$ by assumption
(3) $\Omega_{\text{params}} \vdash^{\mathsf{cs}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau' :: \mathtt{Type}$ by assumption
(4) $\mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{e}$ by assumption
(5) $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [\omega]\tau'$ by assumption
(6) $\Omega_{\text{app}} \vdash e : [\omega]\tau'$ by IH, part 1(a) on (5)
(7) $\Omega_{\text{app}} \cup \Omega \vdash e : [\omega]\tau'$ by Weakening on (6)

(b) **Case** (C.42).

(1) $\grave{r} = \mathtt{prrule}(p.\grave{e})$ by assumption
(2) $r = \mathtt{rule}(p.e)$ by assumption
(3) $\Omega \vdash p : \tau \dashv\!| \Omega'$ by assumption
(4) $\Omega \cup \Omega' \vdash^{\omega:\Omega_{\text{params}};\langle \mathcal{M};\mathcal{D};\mathcal{G};\Omega_{\text{app}}\rangle;\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e : \tau'$ by assumption
(5) $(\Omega \cup \Omega') \cap \Omega_{\text{app}} = \emptyset$ by identification convention
(6) $\Omega \cup \Omega' \cup \Omega_{\text{app}} \vdash e : \tau'$ by IH, part 2(a) on (4) and (5)
(7) $\Omega \cup \Omega_{\text{app}} \vdash p : \tau \dashv\!| \Omega'$ by Weakening on (3)
(8) $\Omega \cup \Omega_{\text{app}} \vdash r : \tau \Mapsto \tau'$ by Rule (C.13) on (7) and (6)

The mutual induction can be shown to be well-founded by an argument analagous to that in the proof of Theorem B.27, appealing to Condition C.12 and Condition C.15. □

Modules

Theorem C.43 (Module Expansion). *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma$ *then* $\Omega \vdash M : \sigma$.

Proof. By rule induction over Rules (C.16). In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$.

Case (C.16a).

(1) $\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma'$ by assumption
(2) $\hat{\Omega} \vdash \sigma' <: \sigma$ by assumption
(3) $\Omega \vdash M : \sigma'$ by IH on (1)
(4) $\Omega \vdash M : \sigma$ by Rule (C.4a) on (3) and (2)

Case (C.16b) **through** (C.16e). In each of these cases, we apply the IH over each premise then apply the corresponding signature matching rule in Rules (C.4) and weakening as needed.

Case (C.16f) **through** (C.16i). In each of these cases, we apply the IH to the module expansion premise.

□

C.4.3 Abstract Reasoning Principles

Lemma C.44 (Proto-Construction and Proto-Kind Decomposition)**.**

1. *If* $\Omega \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{c} \rightsquigarrow c :: \kappa$ *and* $\mathsf{summary}(\grave{c}) = \{\texttt{splicedk}[m_i; n_i]\}_{0\leq i<n_{kind}} \cup \{\texttt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0\leq i<n_{con}}$ *then*
   (a) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i\ \mathsf{kind}\}_{0\leq i<n_{kind}}$
   (b) $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i\ \mathsf{kind}\}_{0\leq i<n_{con}}$
   (c) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0\leq i<n_{con}}$
   (d) $c = [\{\kappa_i/k_i\}_{0\leq i<n_{kind}}, \{c_i/u_i\}_{0\leq i<n_{con}}, \omega]c'$ *for some* c' *and fresh* $\{k_i\}_{0\leq i<n_{kind}}$ *and fresh* $\{u_i\}_{0\leq i<n_{con}}$
   (e) $\mathsf{fv}(c') \subset \{k_i\}_{0\leq i<n_{kind}} \cup \{u_i\}_{0\leq i<n_{con}} \cup dom(\Omega_{params})$
2. *If* $\Omega \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\kappa} \rightsquigarrow \kappa\ \mathsf{kind}$ *and* $\mathsf{summary}(\grave{\kappa}) = \{\texttt{splicedk}[m_i; n_i]\}_{0\leq i<n_{kind}} \cup \{\texttt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0\leq i<n_{con}}$ *then*
   (a) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i\ \mathsf{kind}\}_{0\leq i<n_{kind}}$
   (b) $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i\ \mathsf{kind}\}_{0\leq i<n_{con}}$
   (c) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0\leq i<n_{con}}$
   (d) $\kappa = [\{\kappa_i/k_i\}_{0\leq i<n_{kind}}, \{c_i/u_i\}_{0\leq i<n_{con}}, \omega]\kappa'$ *for some* κ' *and fresh* $\{k_i\}_{0\leq i<n_{kind}}$ *and fresh* $\{u_i\}_{0\leq i<n_{con}}$
   (e) $\mathsf{fv}(\kappa') \subset \{k_i\}_{0\leq i<n_{kind}} \cup \{u_i\}_{0\leq i<n_{con}} \cup dom(\Omega_{params})$

Proof. By mutual rule induction over Rules (C.40) and Rules (C.39).

1. **Case** (C.40a)**.** This case follows by applying the IH.

   Case (C.40b) **through** (C.40o)**.** These cases follow by applying the IH,gathering together the sets of conclusions and invoking the identification convention as necessary.

   Case (C.40p)**.** Letting $\mathbb{C} = \omega : \Omega_{\text{params}}; \hat{\Omega}; b$,

   (1) $\grave{c} = \texttt{splicedc}[m; n; \grave{\kappa}]$ — by assumption
   (2) $\Omega_{\text{params}} \vdash^{\mathbb{C}} \grave{\kappa} \rightsquigarrow \kappa\ \mathsf{kind}$ — by assumption
   (3) $\mathsf{parseUCon}(\mathsf{subseq}(b; m; n)) = \hat{c}$ — by assumption
   (4) $\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: [\omega]\kappa$ — by assumption
   (5) $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle$ — by assumption
   (6) $\text{dom}(\Omega) \cap \text{dom}(\Omega_{\text{app}}) = \emptyset$ — by assumption
   (7) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i\ \mathsf{kind}\}_{0\leq i<n_{\text{kind}}}$ — by IH, part 2 on (2)
   (8) $\{\Omega_{\text{params}} \vdash^{\omega:\Omega_{\text{params}};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i\ \mathsf{kind}\}_{0\leq i<n_{\text{con}}-1}$ — by IH, part 2 on (2)

(9) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m_i'; n_i')) \rightsquigarrow c_i :: [\omega]\kappa_i'\}_{0 \le i < n_{\mathrm{con}}-1}$ by IH, part 2 on (2)

(10) $\kappa = [\{\kappa_i/k_i\}_{0 \le i < n_{\mathrm{kind}}}, \{c_i/u_i\}_{0 \le i < n_{\mathrm{con}}-1}, \omega]\kappa'$ for some κ' and fresh $\{k_i\}_{0 \le i < n_{\mathrm{kind}}}$ and fresh $\{u_i\}_{0 \le i < n_{\mathrm{con}}}$ by IH, part 2 on (2)

(11) $\mathsf{fv}(\kappa') \subset \{k_i\}_{0 \le i < n_{\mathrm{kind}}} \cup \{u_i\}_{0 \le i < n_{\mathrm{con}}-1} \cup \mathrm{dom}(\Omega_{\mathrm{params}})$ by IH, part 2 on (2)

The conclusions hold as follows:

(a) (7)
(b) (2) and (8)
(c) (3), (4) and (9)
(d) Choose $c' = u$ for fresh u. Then $c = [\omega', c/u]u$ for any ω'.
(e) $\mathsf{fv}(u) = u$ and $u \subset \{u\} \cup \{k_i\}_{0 \le i < n_{\mathrm{kind}}} \cup \{u_i\}_{0 \le i < n_{\mathrm{con}}-1} \cup \mathrm{dom}(\Omega_{\mathrm{params}})$ by definition.

2. **Case** (C.39a) **through** (C.39e)**.** These cases follow by applying the IH and gathering together the sets in the conclusion, invoking the identification convention as necessary.

Case (C.39f)**.** Letting $\mathbb{C} = \omega : \Omega_{\mathrm{params}}; \hat{\Omega}; b$

(1) $\hat{\kappa} = \texttt{splicedk}[m; n]$ by assumption
(2) $\mathsf{parseUKind}(\mathsf{subseq}(b; m; n)) = \hat{\kappa}$ by assumption
(3) $\hat{\Omega} \vdash \hat{\kappa} \rightsquigarrow \kappa$ kind by assumption
(4) $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\mathrm{app}} \rangle$ by assumption
(5) $\mathrm{dom}(\Omega) \cap \mathrm{dom}(\Omega_{\mathrm{app}}) = \emptyset$ by assumption
(6) $n_{\mathrm{con}} = 0$ by definition of summary

The conclusions hold as follows:

(a) (2) and (3)
(b) (6)
(c) (6)
(d) Choose $\kappa' = k$ for fresh k. Then $\kappa = [\omega', \kappa/k]k$ for any ω'.
(e) $\mathsf{fv}(k) = k$ and $k \subset \{k\} \cup \mathrm{dom}(\Omega_{\mathrm{params}})$ by definition.

□

Lemma C.45 (Proto-Expression and Proto-Rule Decomposition)**.**

1. *If* $\Omega \vdash^{\omega:\Omega_{params}; \hat{\Omega}; \hat{\Psi}; \hat{\Phi}; b} \grave{e} \rightsquigarrow e : \tau_{proto}$ *and* $\mathsf{summary}(\grave{e}) = \{\texttt{splicedk}[m_i; n_i]\}_{0 \le i < n_{kind}} \cup \{\texttt{splicedc}[m_i'; n_i'; \grave{\kappa}_i']\}_{0 \le i < n_{con}} \cup \{\texttt{splicede}[m_i''; n_i''; \grave{\tau}_i]\}_{0 \le i < n_{exp}}$ *then*

(a) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i \text{ kind}\}_{0 \le i < n_{kind}}$

(b) $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i \ \mathsf{kind}\}_{0\le i<n_{con}}$

(c) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0\le i<n_{con}}$

(d) $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{0\le i<n_{exp}}$

(e) $\{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m''_i; n''_i)) \rightsquigarrow e_i : [\omega]\tau_i\}_{0\le i<n_{exp}}$

(f) $e = [\{\kappa_i/k_i\}_{0\le i<n_{kind}}, \{c_i/u_i\}_{0\le i<n_{con}}, \{e_i/x_i\}_{0\le i<n_{exp}}, \omega]e''$ *for some* e'' *and fresh* $\{k_i\}_{0\le i<n_{kind}}$ *and fresh* $\{u_i\}_{0\le i<n_{con}}$ *and fresh* $\{x_i\}_{0\le i<n_{exp}}$

(g) $\mathsf{fv}(e'') \subset \{k_i\}_{0\le i<n_{kind}} \cup \{u_i\}_{0\le i<n_{con}} \cup \{x_i\}_{0\le i<n_{exp}} \cup dom(\Omega_{params})$

2. *If* $\Omega \vdash^{\omega:\Omega_{params};\hat{\Omega};\hat{\Psi};\hat{\Phi};b} \grave{r} \rightsquigarrow r : \tau \Mapsto \tau'$ *and* $\mathsf{summary}(\grave{r}) = \{\mathtt{splicedk}[m_i; n_i]\}_{0\le i<n_{kind}} \cup \{\mathtt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0\le i<n_{con}} \cup \{\mathtt{splicede}[m''_i; n''_i; \grave{\tau}_i]\}_{0\le i<n_{exp}}$ *then*

(a) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i \ \mathsf{kind}\}_{0\le i<n_{kind}}$

(b) $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i \ \mathsf{kind}\}_{0\le i<n_{con}}$

(c) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0\le i<n_{con}}$

(d) $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{0\le i<n_{exp}}$

(e) $\{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m''_i; n''_i)) \rightsquigarrow e_i : [\omega]\tau_i\}_{0\le i<n_{exp}}$

(f) $r = [\{\kappa_i/k_i\}_{0\le i<n_{kind}}, \{c_i/u_i\}_{0\le i<n_{con}}, \{e_i/x_i\}_{0\le i<n_{exp}}, \omega]r'$ *for some* r' *and fresh* $\{k_i\}_{0\le i<n_{kind}}$ *and fresh* $\{u_i\}_{0\le i<n_{con}}$ *and fresh* $\{x_i\}_{0\le i<n_{exp}}$

(g) $\mathsf{fv}(r') \subset \{k_i\}_{0\le i<n_{kind}} \cup \{u_i\}_{0\le i<n_{con}} \cup \{x_i\}_{0\le i<n_{exp}} \cup dom(\Omega_{params})$

Proof. By mutual rule induction over Rules (C.41) and Rule (C.42).

1. **Case** (C.41a)**.** This case follows by applying the IH.

   Case (C.41b) **through** (C.41o)**.** These cases follow by applying the IH or Lemma C.44, gathering together the sets of conclusions and invoking the identification convention as necessary.

   Case (C.41p)**.** Letting $\mathbb{E} = \omega : \Omega_{\text{params}};\ \hat{\Omega};\ \hat{\Psi};\ \hat{\Phi};\ b$,

   (1) $\grave{e} = \mathtt{splicede}[m; n; \grave{\tau}]$ — by assumption
   (2) $\Omega_{\text{params}} \vdash^{\mathsf{cs}(\mathbb{E})} \grave{\tau} \rightsquigarrow \tau :: \mathtt{Type}$ — by assumption
   (3) $\mathsf{parseUExp}(\mathsf{subseq}(b; m; n)) = \hat{e}$ — by assumption
   (4) $\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [\omega]\tau$ — by assumption
   (5) $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle$ — by assumption
   (6) $\mathrm{dom}(\Omega) \cap \mathrm{dom}(\Omega_{\text{app}}) = \emptyset$ — by assumption
   (7) $\mathsf{summary}(\grave{c}) = \{\mathtt{splicedk}[m_i; n_i]\}_{0\le i<n_{\text{kind}}} \cup \{\mathtt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0\le i<n_{\text{con}}}$ — by definition

(8) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i \ \mathsf{kind}\}_{0 \leq i < n_{\text{kind}}}$ by Lemma C.44 on (2) and (7)

(9) $\{\Omega_{\text{params}} \vdash^{\omega:\Omega_{\text{params}}; \hat{\Omega}; b} \check{\kappa}'_i \rightsquigarrow \kappa'_i \ \mathsf{kind}\}_{0 \leq i < n_{\text{con}}}$ by Lemma C.44 on (2) and (7)

(10) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0 \leq i < n_{\text{con}}}$ by Lemma C.44 on (2) and (7)

The conclusions hold as follows:

(a) (8)
(b) (9)
(c) (10)
(d) (2)
(e) (3) and (4)
(f) Choose $e'' = x$ for fresh x. Then $e = [\omega', e/x]x$ for any ω'.
(g) $\mathsf{fv}(x) = x$ and $x \subset \{x\} \cup \{k_i\}_{0 \leq i < n_{\text{kind}}} \cup \{u_i\}_{0 \leq i < n_{\text{con}}} \cup \mathrm{dom}(\Omega_{\text{params}})$ by definition.

2. There is only one case. The conclusions follow immediately after applying the IH, part 1 on the branch expression.

□

Theorem C.46 (peTSM Abstract Reasoning Principles). *If* $\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e}$ ‘b‘ $\rightsquigarrow e : \tau$ *then:*

1. $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{app} \rangle$
2. $\hat{\Psi} = \langle \mathcal{A}; \Psi \rangle$
3. ***(Typing 1)*** $\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{e} \rightsquigarrow \epsilon$ @ $\mathtt{type}(\tau)$ *and* $\Omega_{app} \vdash e : \tau$
4. $\Omega_{app} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon_{normal}$
5. $\mathsf{tsmdef}(\epsilon_{normal}) = a$
6. $\Psi = \Psi', a \hookrightarrow \mathtt{petsm}(\rho; e_{parse})$
7. $b \downarrow_{\mathsf{Body}} e_{body}$
8. $e_{parse}(e_{body}) \Downarrow \mathtt{inj}[\mathtt{SuccessE}]\,(e_{pproto})$
9. $e_{pproto} \uparrow_{\mathsf{PPrExpr}} \dot{e}$
10. $\Omega_{app} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \leftarrow\!\!\!\shortmid_{\epsilon_{normal}} \grave{e}$? $\mathtt{type}(\tau_{proto}) \dashv \omega : \Omega_{params}$
11. ***(Segmentation)*** $\mathsf{seg}(\grave{e})$ segments b

12. $\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e' : \tau_{proto}$
13. $e = [\omega]e'$
14. $\tau = [\omega]\tau_{proto}$
15. $\mathsf{summary}(\grave{e}) = \{\texttt{splicedk}[m_i; n_i]\}_{0 \le i < n_{kind}} \cup \{\texttt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0 \le i < n_{con}} \cup \{\texttt{splicede}[m''_i; n''_i; \grave{\tau}_i]\}_{0 \le i < n_{exp}}$
16. ***(Kinding 1)*** $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i\ \mathsf{kind}\}_{0 \le i < n_{kind}}$ *and* $\{\Omega_{app} \vdash \kappa_i\ \mathsf{kind}\}_{0 \le i < n_{kind}}$
17. ***(Kinding 2)*** $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i\ \mathsf{kind}\}_{0 \le i < n_{con}}$ *and* $\{\Omega_{app} \vdash [\omega]\kappa'_i\ \mathsf{kind}\}_{0 \le i < n_{con}}$
18. ***(Kinding 3)*** $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0 \le i < n_{con}}$ *and* $\{\Omega_{app} \vdash c_i :: [\omega]\kappa'_i\}_{0 \le i < n_{con}}$
19. ***(Kinding 4)*** $\{\Omega_{params} \vdash^{\omega:\Omega_{params};\hat{\Omega};b} \grave{\tau}_i \rightsquigarrow \tau_i :: \texttt{Type}\}_{0 \le i < n_{exp}}$ *and* $\{\Omega_{app} \vdash [\omega]\tau_i :: \texttt{Type}\}_{0 \le i < n_{exp}}$
20. ***(Typing 2)*** $\{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m''_i; n''_i)) \rightsquigarrow e_i : [\omega]\tau_i\}_{0 \le i < n_{exp}}$ *and* $\{\Omega_{app} \vdash e_i : [\omega]\tau_i\}_{0 \le i < n_{exp}}$
21. ***(Capture Avoidance)*** $e = [\{\kappa_i/k_i\}_{0 \le i < n_{kind}}, \{c_i/u_i\}_{0 \le i < n_{con}}, \{e_i/x_i\}_{0 \le i < n_{exp}}, \omega]e''$ *for some* e'' *and fresh* $\{k_i\}_{0 \le i < n_{kind}}$ *and fresh* $\{u_i\}_{0 \le i < n_{con}}$ *and fresh* $\{x_i\}_{0 \le i < n_{exp}}$
22. ***(Context Independence)***

$$\mathsf{fv}(e'') \subset \{k_i\}_{0 \le i < n_{kind}} \cup \{u_i\}_{0 \le i < n_{con}} \cup \{x_i\}_{0 \le i < n_{exp}} \cup dom(\Omega_{params})$$

Proof. By rule induction over Rules (C.19). The only rule that applies is Rule (C.19p).

Case (C.19p)**.**

(1)	$\hat{e} = \hat{\epsilon}\ \text{`}b\text{`}$	by assumption
(2)	$e = [\omega]e'$	by assumption
(3)	$\tau = [\omega]\tau_{\text{proto}}$	by assumption
(4)	$\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{\text{app}} \rangle$	by assumption
(5)	$\hat{\Psi} = \langle \mathcal{A}; \Psi \rangle$	by assumption
(6)	$\hat{\Omega} \vdash^{\mathsf{Exp}}_{\hat{\Psi}} \hat{\epsilon} \rightsquigarrow \epsilon\ @\ \texttt{type}(\tau_{\text{final}})$	by assumption
(7)	$\Omega_{\text{app}} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \Downarrow \epsilon_{\text{normal}}$	by assumption
(8)	$\mathsf{tsmdef}(\epsilon_{\text{normal}}) = a$	by assumption
(9)	$\Psi = \Psi', a \hookrightarrow \texttt{petsm}(\rho; e_{\text{parse}})$	by assumption

(10) $b \downarrow_{\mathsf{Body}} e_{\mathrm{body}}$ — by assumption

(11) $e_{\mathrm{parse}}(e_{\mathrm{body}}) \Downarrow \mathtt{inj}[\mathtt{SuccessE}](e_{\mathrm{pproto}})$ — by assumption

(12) $e_{\mathrm{pproto}} \uparrow_{\mathsf{PPrExpr}} \dot{e}$ — by assumption

(13) $\Omega_{\mathrm{app}} \vdash^{\mathsf{Exp}}_{\Psi} \dot{e} \hookrightarrow_{\epsilon_{\mathrm{normal}}} \grave{e} \mathbin{?} \mathtt{type}(\tau_{\mathrm{proto}}) \dashv \omega : \Omega_{\mathrm{params}}$ — by assumption

(14) $\mathsf{seg}(\grave{e})$ segments b — by assumption

(15) $\Omega_{\mathrm{params}} \vdash^{\omega:\Omega_{\mathrm{params}};\hat{\Omega};\hat{\Psi};\hat{\Phi};b} \grave{e} \rightsquigarrow e' : \tau_{\mathrm{proto}}$ — by assumption

(16) $\Omega_{\mathrm{app}} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon \mathbin{@} \mathtt{type}(\tau_{\mathrm{final}})$ — by Theorem C.37 on (6)

(17) $\Omega_{\mathrm{app}} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon_{\mathrm{normal}} \mathbin{@} \mathtt{type}(\tau_{\mathrm{final}})$ — by Corollary C.28 on (16) and (7)

(18) $\Omega_{\mathrm{app}} \vdash^{\mathsf{Exp}}_{\Psi} \epsilon_{\mathrm{normal}} \mathbin{@} [\omega]\mathtt{type}(\tau_{\mathrm{proto}})$ — by Lemma C.41 on (13)

(19) $\mathtt{type}(\tau_{\mathrm{final}}) = [\omega]\mathtt{type}(\tau_{\mathrm{proto}})$ — by Theorem C.24 on (17) and (18)

(20) $\tau_{\mathrm{final}} = [\omega]\tau_{\mathrm{proto}} = \tau$ — by definition of substitution and (19) and (3)

(21) $\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{\epsilon}$ ‘b‘ $\rightsquigarrow e : \tau$ — by assumption

(22) $\Omega_{\mathrm{app}} \vdash e : \tau$ — by Theorem C.42 on (21)

(23) $\mathsf{summary}(\grave{e}) = \{\mathtt{splicedk}[m_i; n_i]\}_{0 \le i < n_{\mathrm{kind}}} \cup \{\mathtt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0 \le i < n_{\mathrm{con}}} \cup \{\mathtt{splicede}[m''_i; n''_i; \grave{\tau}_i]\}_{0 \le i < n_{\mathrm{exp}}}$ — by definition

(24) $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i\ \mathsf{kind}\}_{0 \le i < n_{\mathrm{kind}}}$ — by Lemma C.45 on (15) and (23)

(25) $\{\Omega_{\mathrm{params}} \vdash^{\omega:\Omega_{\mathrm{params}};\hat{\Omega};b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i\ \mathsf{kind}\}_{0 \le i < n_{\mathrm{con}}}$ — by Lemma C.45 on (15) and (23)

(26) $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0 \le i < n_{\mathrm{con}}}$ — by Lemma C.45 on (15) and (23)

(27) $\{\Omega_{\mathrm{params}} \vdash^{\omega:\Omega_{\mathrm{params}};\hat{\Omega};b} \grave{\tau}_i \rightsquigarrow \tau_i :: \mathtt{Type}\}_{0 \le i < n_{\mathrm{exp}}}$ — by Lemma C.45 on (15) and (23)

(28) $\{\hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \mathsf{parseUExp}(\mathsf{subseq}(b; m''_i; n''_i)) \rightsquigarrow e_i : [\omega]\tau_i\}_{0 \le i < n_{\mathrm{exp}}}$ — by Lemma C.45 on (15) and (23)

(29) $e = [\{\kappa_i/k_i\}_{0\le i<n_{\text{kind}}}, \{c_i/u_i\}_{0\le i<n_{\text{con}}}, \{e_i/x_i\}_{0\le i<n_{\text{exp}}}, \omega]e''$ for some e'' and fresh $\{k_i\}_{0\le i<n_{\text{kind}}}$ and fresh $\{u_i\}_{0\le i<n_{\text{con}}}$ and fresh $\{x_i\}_{0\le i<n_{\text{exp}}}$
by Lemma C.45 on (15) and (23)

(30) $\text{fv}(e'') \subset \{k_i\}_{0\le i<n_{\text{kind}}} \cup \{u_i\}_{0\le i<n_{\text{con}}} \cup \{x_i\}_{0\le i<n_{\text{exp}}} \cup \text{dom}(\Omega_{\text{params}})$
by Lemma C.45 on (15) and (23)

The conclusions hold as follows:

1. (4)
2. (5)
3. (6) and (20) and (22)
4. (7)
5. (8)
6. (9)
7. (10)
8. (11)
9. (12)
10. (13)
11. (14)
12. (15)
13. (2)
14. (3)
15. (23)
16. (24)
17. (25)
18. (26)
19. (27)
20. (28)
21. (29)
22. (30)

□

Theorem C.47 (ppTSM Abstract Reasoning Principles). *If* $\hat{\Omega} \vdash_{\hat{\Phi}} \hat{e}$ ‘b‘ $\rightsquigarrow p : \tau \dashv\!\shortmid \hat{\Omega}'$ *then:*

1. $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega_{app} \rangle$

2. $\hat{\Phi} = \langle \mathcal{A}; \Phi \rangle$
3. ***(Typing 1)*** $\hat{\Omega} \vdash^{\mathsf{Pat}}_{\hat{\Phi}} \hat{e} \rightsquigarrow e \mathbin{@} \texttt{type}(\tau)$ *and* $\Omega_{app} \vdash p : \tau \dashv\!\!\mid \hat{\Omega}'$
4. $\Omega_{app} \vdash^{\mathsf{Pat}}_{\Phi} e \Downarrow e_{normal}$
5. $\mathsf{tsmdef}(e_{normal}) = a$
6. $\Phi = \Phi', a \hookrightarrow \texttt{pptsm}(\rho; e_{parse})$
7. $b \downarrow_{\mathsf{Body}} e_{body}$
8. $e_{parse}(e_{body}) \Downarrow \texttt{inj}[\mathtt{SuccessP}]\,(e_{proto})$
9. $e_{proto} \uparrow_{\mathsf{PPrPat}} \dot{p}$
10. $\Omega_{app} \vdash^{\mathsf{Pat}}_{\Phi} \dot{p} \looparrowright_{e_{normal}} \grave{p}$? $\texttt{type}(\tau_{proto}) \dashv \omega : \Omega_{params}$
11. ***(Segmentation)*** $\mathsf{seg}(\dot{p})$ segments b
12. $\mathsf{summary}(\grave{e}) = \{\texttt{splicedk}[m_i; n_i]\}_{0 \le i < n_{kind}} \cup \{\texttt{splicedc}[m'_i; n'_i; \grave{\kappa}'_i]\}_{0 \le i < n_{con}} \cup \{\texttt{splicedp}[m''_i; n''_i; \grave{\tau}_i]\}_{0 \le i < n_{pat}}$
13. ***(Kinding 1)*** $\{\hat{\Omega} \vdash \mathsf{parseUKind}(\mathsf{subseq}(b; m_i; n_i)) \rightsquigarrow \kappa_i\ \mathsf{kind}\}_{0 \le i < n_{kind}}$ *and* $\{\Omega_{app} \vdash \kappa_i\ \mathsf{kind}\}_{0 \le i < n_{kind}}$
14. ***(Kinding 2)*** $\{\Omega_{params} \vdash^{\omega : \Omega_{params}; \hat{\Omega}; b} \grave{\kappa}'_i \rightsquigarrow \kappa'_i\ \mathsf{kind}\}_{0 \le i < n_{con}}$ *and* $\{\Omega_{app} \vdash [\omega]\kappa'_i\ \mathsf{kind}\}_{0 \le i < n_{con}}$
15. ***(Kinding 3)*** $\{\hat{\Omega} \vdash \mathsf{parseUCon}(\mathsf{subseq}(b; m'_i; n'_i)) \rightsquigarrow c_i :: [\omega]\kappa'_i\}_{0 \le i < n_{con}}$ *and* $\{\Omega_{app} \vdash c_i :: [\omega]\kappa'_i\}_{0 \le i < n_{con}}$
16. ***(Kinding 4)*** $\{\Omega_{params} \vdash^{\omega : \Omega_{params}; \hat{\Omega}; b} \grave{\tau}_i \rightsquigarrow \tau_i :: \texttt{Type}\}_{0 \le i < n_{pat}}$ *and* $\{\Omega_{app} \vdash [\omega]\tau_i :: \texttt{Type}\}_{0 \le i < n_{pat}}$
17. ***(Typing 3)*** $\{\hat{\Omega} \vdash_{\hat{\Phi}} \mathsf{parseUPat}(\mathsf{subseq}(b; m''_i; n''_i)) \rightsquigarrow p_i : [\omega]\tau_i \dashv\!\!\mid \hat{\Omega}'\}_{0 \le i < n_{pat}}$
18. ***(No Hidden Bindings)*** $\hat{\Omega}' = \biguplus_{0 \le i < n_{pat}} \hat{\Omega}_i$

Proof Sketch. By rule induction over Rules (C.21). (There are no details about ppTSMs in the paper, so we leave out further details here as well. However, see Theorem B.33 for a proof of the corresponding theorem for spTSMs. The same approach, tweaked as in the above developments to incorporate parameters, should work here.) □

Appendix D

miniVerse$_{\mathbf{PH}}$

We will now formalize the mechanisms outlined in Sec. 6 of the paper by developing a reduced calculus, miniVerse$_{\mathbf{PH}}$. This calculus is defined identically to miniVerse$_{\mathbf{P}}$ with the exception of the syntax and semantics of unexpanded module expressions, $\hat{M}$, so we assume all of the definitions that were given in Appendix C without restating them.

D.1 Syntax of Unexpanded Modules

The syntax of unexpanded modules is defined in Figure D.1. The parts of this figure that differ from miniVerse$_{\mathbf{P}}$ are highlighted in yellow. Each binding form has a *phase* annotation, φ, and parse functions are now unexpanded expressions, $\hat{e}$, rather than expanded expressions, e. In the textual syntax, the phase annotation `standard` is assumed when no phase annotation has been given.

D.2 Module Expansion

The module expansion judgement in miniVerse$_{\mathbf{PH}}$ takes the following form:

Judgement Form	**Description**
$\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma$	$\hat{M}$ has expansion M matching σ

The difference here is that there is now a *static environment*, Σ. Static environments take the form $\omega : \hat{\Omega}; \hat{\Psi}; \hat{\Phi}$, where ω is a substitution.

The static environment passes opaquely through the subsumption rule and the rules governing module identifiers, structures and sealing:

$$\frac{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma \qquad \hat{\Omega} \vdash \sigma <: \sigma'}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma'} \tag{D.1a}$$

$$\frac{}{\hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{X} \rightsquigarrow X : \sigma} \tag{D.1b}$$

Sort		Stylized Form	Description
Phase	φ ::=	`standard`	standard phase
		`static`	static phase
UMod	$\hat{M}$ ::=	$\hat{X}$	module identifier
		$[\![\hat{c};\hat{e}]\!]$	structure
		$\hat{M} \upharpoonright \hat{\sigma}$	seal
		$(\varphi\ \texttt{let}\ \hat{X} = \hat{M}\ \texttt{in}\ \hat{M}) : \hat{\sigma}$	definition
		$\varphi\ \texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\rho}\ \texttt{for expressions by static}\ \hat{e}\ \texttt{in}\ \hat{M}$	peTSM definition
		$\varphi\ \texttt{let syntax}\ \hat{a} = \hat{e}\ \texttt{for expressions in}\ \hat{M}$	peTSM binding
		$\varphi\ \texttt{syntax}\ \hat{a}\ \texttt{at}\ \hat{\rho}\ \texttt{for patterns by static}\ \hat{e}\ \texttt{in}\ \hat{M}$	ppTSM definition
		$\varphi\ \texttt{let syntax}\ \hat{a} = \hat{e}\ \texttt{for patterns in}\ \hat{M}$	ppTSM binding

Figure D.1: Syntax of unexpanded modules in miniVerse$_{\mathbf{PH}}$

$$\frac{\hat{\Omega} \vdash \hat{c} \rightsquigarrow c :: \kappa \qquad \hat{\Omega} \vdash_{\hat{\Psi};\hat{\Phi}} \hat{e} \rightsquigarrow e : [c/u]\tau}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} [\![\hat{c};\hat{e}]\!] \rightsquigarrow \texttt{struct}(c;e) : \texttt{sig}\{\kappa\}(u.\tau)} \tag{D.1c}$$

$$\frac{\hat{\Omega} \vdash \hat{\sigma} \rightsquigarrow \sigma\ \mathsf{sig} \qquad \hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M} \upharpoonright \hat{\sigma} \rightsquigarrow \texttt{seal}\{\sigma\}(M) : \sigma} \tag{D.1d}$$

Each binding form in the syntax of $\hat{M}$ is qualified with a *phase*, φ, which is either `standard` or `static`. The static environment passes opaquely through the standard phase module let binding construct:

$$\frac{\begin{array}{c}\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma \qquad \hat{\Omega} \vdash \hat{\sigma}' \rightsquigarrow \sigma'\ \mathsf{sig} \\ \hat{\Omega}, \hat{X} \rightsquigarrow X : \sigma \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \hat{M}' \rightsquigarrow M' : \sigma'\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} (\texttt{standard let}\ \hat{X} = \hat{M}\ \texttt{in}\ \hat{M}') : \hat{\sigma}' \rightsquigarrow \texttt{mlet}\{\sigma'\}(M; X.M') : \sigma'} \tag{D.1e}$$

The rule for the `static` phase module let binding construct, on the other hand, calls for the module expression being bound to be evaluated to a module value under the current environment. The substitution and corresponding unexpanded context is then extended with this module value:

$$\frac{\begin{array}{c}\Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \\ \hat{\Omega}_S \vdash^{\Sigma}_{\hat{\Psi}_S;\hat{\Phi}_S} \hat{M} \rightsquigarrow M : \sigma \qquad [\omega]M \Downarrow M' \\ \hat{\Omega} \vdash \hat{\sigma}' \rightsquigarrow \sigma'\ \mathsf{sig} \qquad \hat{\Omega} \vdash^{\omega, M'/X:\hat{\Omega}_S, \hat{X} \rightsquigarrow X:\sigma; \hat{\Psi}_S; \hat{\Phi}_S}_{\hat{\Psi};\hat{\Phi}} \hat{M}' \rightsquigarrow M' : \sigma'\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} (\texttt{static let}\ \hat{X} = \hat{M}\ \texttt{in}\ \hat{M}') : \hat{\sigma}' \rightsquigarrow M' : \sigma'} \tag{D.1f}$$

The standard peTSM definition construct is governed by the following rule:

$$\frac{\begin{array}{c}\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty} \qquad \Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \\ \hat{\Omega}_S \vdash_{\hat{\Psi}_S;\hat{\Phi}_S} \hat{e}_{\text{parse}} \rightsquigarrow e_{\text{parse}} : \texttt{parr(Body;ParseResult(PPrExpr))} \\ [\omega]e_{\text{parse}} \Downarrow e'_{\text{parse}} \qquad \hat{\Omega} \vdash^{\Sigma}_{\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \texttt{defref[}a\texttt{]};\Psi, a \hookrightarrow \texttt{petsm}(\rho;e'_{\text{parse}})\rangle;\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\langle \mathcal{A};\Psi\rangle;\hat{\Phi}} \texttt{standard syntax } \hat{a} \texttt{ at } \hat{\rho} \texttt{ for expressions by static } \hat{e}_{\text{parse}} \texttt{ in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1g}$$

The difference here is that the parse function is an unexpanded (rather than an expanded) expression. It is expanded under the static environment's unified context, $\hat{\Omega}_S$. Then the substitution, ω, is applied to the resulting expanded parse function before it is added to the peTSM context.

The static peTSM definition construct operates similarly, differing only in that the static environment's peTSM context is extended, rather than the standard peTSM context:

$$\frac{\begin{array}{c}\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty} \qquad \Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \qquad \hat{\Psi}_S = \langle \mathcal{A}_S; \Psi_S\rangle \\ \hat{\Omega}_S \vdash_{\hat{\Psi}_S;\hat{\Phi}_S} \hat{e}_{\text{parse}} \rightsquigarrow e_{\text{parse}} : \texttt{parr(Body;ParseResult(PPrExpr))} \\ [\omega]e_{\text{parse}} \Downarrow e'_{\text{parse}} \qquad \hat{\Omega} \vdash^{\omega:\hat{\Omega}_S;\langle \mathcal{A}_S \uplus \hat{a} \hookrightarrow \texttt{defref[}a\texttt{]};\Psi_S, a \hookrightarrow \texttt{petsm}(\rho;e'_{\text{parse}})\rangle;\hat{\Phi}_S}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \texttt{static syntax } \hat{a} \texttt{ at } \hat{\rho} \texttt{ for expressions by static } \hat{e}_{\text{parse}} \texttt{ in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1h}$$

The static environment passes opaquely through the standard peTSM abbreviation construct:

$$\frac{\hat{\Omega} \vdash^{\mathsf{Exp}}_{\langle \mathcal{A};\Psi\rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho \qquad \hat{\Omega} \vdash^{\Sigma}_{\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \epsilon;\Psi\rangle;\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash^{\Sigma}_{\langle \mathcal{A};\Psi\rangle;\hat{\Phi}} \texttt{standard let syntax } \hat{a} = \hat{\epsilon} \texttt{ for expressions in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1i}$$

The static peTSM abbreviation construct updates the static peTSM identifier expansion context, $\mathcal{A}_S$:

$$\frac{\begin{array}{c}\hat{\Omega} \vdash^{\mathsf{Exp}}_{\langle \mathcal{A};\Psi\rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho \\ \Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \qquad \hat{\Psi}_S = \langle \mathcal{A}_S; \Psi_S\rangle \\ \hat{\Omega} \vdash^{\omega:\Omega_S;\langle \mathcal{A}_S \uplus \hat{a} \hookrightarrow \epsilon;\Psi_S\rangle;\hat{\Phi}_S}_{\hat{\Psi};\hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\hat{\Phi}} \texttt{static let syntax } \hat{a} = \hat{\epsilon} \texttt{ for expressions in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1j}$$

The rules governing ppTSM definitions and abbreviations are analagous:

$$\frac{\begin{array}{c}\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho\ \mathsf{tsmty} \qquad \Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \\ \hat{\Omega}_S \vdash_{\hat{\Psi}_S;\hat{\Phi}_S} \hat{e}_{\text{parse}} \rightsquigarrow e_{\text{parse}} : \texttt{parr(Body;ParseResult(PPrPat))} \\ [\omega]e_{\text{parse}} \Downarrow e'_{\text{parse}} \qquad \hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\langle \mathcal{A} \uplus \hat{a} \hookrightarrow \texttt{defref[}a\texttt{]};\Phi, a \hookrightarrow \texttt{pptsm}(\rho;e'_{\text{parse}})\rangle} \hat{M} \rightsquigarrow M : \sigma\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi};\langle \mathcal{A};\Phi\rangle} \texttt{standard syntax } \hat{a} \texttt{ at } \hat{\rho} \texttt{ for patterns by static } \hat{e}_{\text{parse}} \texttt{ in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1k}$$

$$\frac{\begin{array}{c}\hat{\Omega} \vdash \hat{\rho} \rightsquigarrow \rho \ \mathsf{tsmty} \qquad \Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \qquad \hat{\Phi}_S = \langle \mathcal{A}_S; \Phi_S \rangle \\ \hat{\Omega}_S \vdash_{\hat{\Psi}_S; \hat{\Phi}_S} \hat{e}_{\text{parse}} \rightsquigarrow e_{\text{parse}} : \texttt{parr}(\texttt{Body}; \texttt{ParseResult}(\texttt{PPrPat})) \\ [\omega] e_{\text{parse}} \Downarrow e'_{\text{parse}} \qquad \hat{\Omega} \vdash^{\omega:\hat{\Omega}_S; \hat{\Psi}_S; \langle \mathcal{A}_S \uplus \hat{a} \hookrightarrow \texttt{defref[}a\texttt{]}; \Phi_S, a \hookrightarrow \texttt{pptsm}(\rho; e_{\text{parse}}) \rangle}_{\hat{\Psi}; \hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi}; \hat{\Phi}} \texttt{static syntax } \hat{a} \texttt{ at } \hat{\rho} \texttt{ for patterns by static } \hat{e}_{\text{parse}} \texttt{ in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1l}$$

$$\frac{\hat{\Omega} \vdash^{\mathsf{Pat}}_{\langle \mathcal{A}; \Phi \rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho \qquad \hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi}; \langle \mathcal{A} \uplus \hat{a} \hookrightarrow \epsilon; \Phi \rangle} \hat{M} \rightsquigarrow M : \sigma}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi}; \langle \mathcal{A}; \Phi \rangle} \texttt{standard let syntax } \hat{a} = \hat{\epsilon} \texttt{ for patterns in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1m}$$

$$\frac{\begin{array}{c}\hat{\Omega} \vdash^{\mathsf{Pat}}_{\langle \mathcal{A}; \Phi \rangle} \hat{\epsilon} \rightsquigarrow \epsilon \mathbin{@} \rho \\ \Sigma = \omega : \hat{\Omega}_S; \hat{\Psi}_S; \hat{\Phi}_S \qquad \hat{\Phi}_S = \langle \mathcal{A}_S; \Phi_S \rangle \\ \hat{\Omega} \vdash^{\omega:\hat{\Omega}_S; \hat{\Psi}_S; \langle \mathcal{A}_S \uplus \hat{a} \hookrightarrow \epsilon; \Phi_S \rangle}_{\hat{\Psi}; \hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma\end{array}}{\hat{\Omega} \vdash^{\Sigma}_{\hat{\Psi}; \hat{\Phi}} \texttt{static let syntax } \hat{a} = \hat{\epsilon} \texttt{ for patterns in } \hat{M} \rightsquigarrow M : \sigma} \tag{D.1n}$$

D.3 Metatheory

The metatheorem having to do with unexpanded module expressions was the Module Expansion theorem, Theorem C.43. This theorem continues to hold in miniVerse$_{\mathbf{PH}}$.

Theorem D.1 (Module Expansion)**.** *If* $\langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle \vdash^{\Sigma}_{\hat{\Psi}; \hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma$ *then* $\Omega \vdash M : \sigma$.

Proof. By rule induction over Rules (D.1). In the following, let $\hat{\Omega} = \langle \mathcal{M}; \mathcal{D}; \mathcal{G}; \Omega \rangle$.

Case (D.1a)**.**

(1) $\hat{\Omega} \vdash_{\hat{\Psi}; \hat{\Phi}} \hat{M} \rightsquigarrow M : \sigma'$	by assumption
(2) $\hat{\Omega} \vdash \sigma' <: \sigma$	by assumption
(3) $\Omega \vdash M : \sigma'$	by IH on (1)
(4) $\Omega \vdash M : \sigma$	by Rule (C.4a) on (3) and (2)

Case (D.1b) **through** (D.1f)**.** In each of these cases, we apply the IH over each premise then apply the corresponding signature matching rule in Rules (C.4) and weakening as needed.

Case (D.1g) **through** (D.1n)**.** In each of these cases, we apply the IH to the module expansion premise.

□

The rest of the metatheory is identical to that of miniVerse$_{\mathbf{P}}$.

Bibliography

[1] Robert Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, 2nd edition, 2016. URL https://www.cs.cmu.edu/~rwh/plbook/2nded.pdf. A.1