Pirouette - Theory

Shruti Gupta **Prashant Godhwani**

September 2023

1 Syntax

$$
\begin{array}{llll}
\text{Locations} & \ell & \in & \mathcal{L} \\
\text{Synchronization Labels} & d & ::= & L \mid R \\
\text{Choreography} & C & ::= & () \mid X \mid \ell.e \mid \ell_1.e \rightsquigarrow \ell_2;\ C \mid \textbf{if } \ell.e \textbf{ then } C_1 \textbf{ else } C_2 \\
 & & \mid & \ell_1[d] \rightsquigarrow \ell_2;\ C \mid \textbf{let } \ell.x ::= C_1 \textbf{ in } C_2 \mid \textbf{fun } X \Rightarrow C \mid C_1\ C_2 \\
 & & \mid & (C_1, C_2) \mid \textbf{fst } C \mid \textbf{snd } C \mid \textbf{left } C \mid \textbf{right } C \\
 & & \mid & \textbf{match } C \textbf{ with left } X \Rightarrow C_1\textbf{; right } Y \Rightarrow C_2 \\
\text{Local Expressions} & e & ::= & () \mid num \mid x \mid e_1\ binop\ e_2 \mid \textbf{let } x = e_1\ in\ e_2 \mid (e_1, e_2) \mid \textbf{fst } e \\
 & & \mid & \textbf{snd } e \mid \textbf{left } e \mid \textbf{right } e \mid \textbf{match } e \textbf{ with left } x \Rightarrow e_1\textbf{; right } y \Rightarrow e_2 \\
\text{Network Expressions} & E & ::= & X \mid () \mid \textbf{fun } X \Rightarrow E \mid E_1\ E_2 \mid \mathsf{ret}(e) \\
 & & \mid & \mathsf{let\ ret}(x) = E_1 \mathsf{\ in\ } E_2 \mid \mathsf{send\ } e \mathsf{\ to\ } \ell;\ E \mid \mathsf{receive\ } x \mathsf{\ from\ } \ell;\ E \\
 & & \mid & \mathsf{if\ } E_1 \mathsf{\ then\ } E_2 \mathsf{\ else\ } E_3 \mid \mathsf{choose\ } d \mathsf{\ for\ } \ell;\ E \\
 & & \mid & \mathsf{allow\ } \ell \mathsf{\ choice\ L} \Rightarrow E_1\ ;\ \mathsf{R} \Rightarrow E_2 \mid (E_1, E_2) \\
 & & \mid & \textbf{fst } E \mid \textbf{snd } E \mid \textbf{left } E \mid \textbf{right } E \\
 & & \mid & \textbf{match } E \textbf{ with left } X \Rightarrow E_1\textbf{; right } Y \Rightarrow E_2 \\
\text{Choreographic Types} & \tau & ::= & \textbf{unit} \mid \ell.t \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2 \mid \tau_1 + \tau_2 \\
\text{Local Types} & t & ::= & \textbf{unit} \mid \textbf{int} \mid \textbf{bool} \mid \textbf{string} \mid t_1 \times t_2 \mid t_1 + t_2 \\
\text{Network Types} & T & ::= & \textbf{unit} \mid \boxed{t} \mid T_1 \to T_2 \mid T_1 \times T_2 \mid T_1 + T_2
\end{array}
$$

2 Type System

2.1 Local Language

$$\textsc{Loc - unit}\quad \frac{}{\Gamma \vdash (\,) : \mathsf{unit}} \qquad \textsc{Loc - var}\quad \frac{x : t \in \Gamma}{\Gamma \vdash x : t} \qquad \textsc{Loc - pair}\quad \frac{\Gamma \vdash e_1 : t_1 \quad \Gamma \vdash e_2 : t_2}{\Gamma \vdash (e_1, e_2) : t_1 \times t_2} \qquad \textsc{Loc - fst}\quad \frac{\Gamma \vdash e : t_1 \times t_2}{\Gamma \vdash \mathsf{fst}\ e : t_1}$$

$$\textsc{Loc - snd}\quad \frac{\Gamma \vdash e : t_1 \times t_2}{\Gamma \vdash \mathsf{snd}\ e : t_2} \qquad \textsc{Loc - left}\quad \frac{\Gamma \vdash e_1 : t_1}{\Gamma \vdash \mathsf{left}\ e_1 : t_1 + t_2} \qquad \textsc{Loc - right}\quad \frac{\Gamma \vdash e_2 : t_2}{\Gamma \vdash \mathsf{right}\ e_2 : t_1 + t_2}$$

$$\textsc{Loc - match}\quad \frac{\Gamma \vdash e : t_1 + t_2 \quad \Gamma, \mathsf{x} : t_1 \vdash e_1 : t_3 \quad \Gamma, \mathsf{y} : t_2 \vdash e_2 : t_3}{\Gamma \vdash (\mathsf{match\ e\ with\ left\ x} \Rightarrow e_1\ ;\ \mathsf{right\ y} \Rightarrow e_2) : t_3}$$

2.2 Network Language

$$\frac{}{\Gamma; \Delta \vdash (\ ) : \mathsf{unit}}\ \textsc{Network - unit} \qquad \frac{X : T \in \Delta}{\Gamma; \Delta \vdash X : T}\ \textsc{Network - var} \qquad \frac{\Gamma; \Delta \vdash e : t}{\Gamma; \Delta \vdash \mathsf{ret}\ (e) : \boxed{t}}\ \textsc{ret}$$

$$\frac{\Gamma; \Delta,\ X : T_1 \vdash E : T_2}{\Gamma; \Delta \vdash \mathsf{fun}\ X \Rightarrow E : T_1 \to T_2}\ \textsc{Network - fun} \qquad \frac{\Gamma; \Delta \vdash E_1 : T_1 \to T_2 \quad \Gamma; \Delta \vdash E_2 : T_1}{\Gamma; \Delta \vdash E_1 E_2 : T_2}\ \textsc{Network - app}$$

$$\frac{\Gamma; \Delta \vdash E_1 : T_1 \quad \Gamma; \Delta \vdash E_2 : T_2 \quad \Gamma; \Delta \vdash E_3 : T_2}{\Gamma; \Delta \vdash \mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 : T_2}\ \textsc{Network - if}$$

$$\frac{\Gamma; \Delta \vdash E_1 : \boxed{t} \quad \Gamma, x : t; \Delta \vdash E_2 : T_2}{\Gamma; \Delta \vdash \mathsf{let\ ret}\ (x) = E_1\ \mathsf{in}\ E_2 : T_2}\ \textsc{Network - def} \qquad \frac{\Gamma; \Delta \vdash e : t \quad \Gamma; \Delta \vdash E : T}{\Gamma; \Delta \vdash \mathsf{send}\ e\ \mathsf{to}\ \ell; E : T}\ \textsc{Network - send}$$

$$\frac{\Gamma, x : t; \Delta \vdash E : T}{\Gamma; \Delta \vdash \mathsf{receive}\ x\ \mathsf{from}\ \ell; E : T}\ \textsc{Network - rcv} \qquad \frac{\Gamma; \Delta \vdash E : T}{\Gamma; \Delta \vdash \mathsf{choose}\ d\ \mathsf{for}\ \ell; E : T}\ \textsc{Network - choose}$$

$$\frac{\Gamma; \Delta \vdash E_1 : T \quad \Gamma; \Delta \vdash E_2 : T}{\Gamma; \Delta \vdash (\mathsf{allow}\ \ell\ \mathsf{choice}\ \mathsf{L} \Rightarrow E_1\ ;\ \mathsf{R} \Rightarrow E_2) : T}\ \textsc{Network - allow}$$

$$\frac{\Gamma; \Delta \vdash E_1 : T_1 \quad \Gamma; \Delta \vdash E_2 : T_2}{\Gamma; \Delta \vdash (E_1, E_2) : T_1 \times T_2}\ \textsc{Network - pair} \qquad \frac{\Gamma; \Delta \vdash E : T_1 \times T_2}{\Gamma; \Delta \vdash \mathsf{fst}\ E : T_1}\ \textsc{Network - fst} \qquad \frac{\Gamma; \Delta \vdash E : T_1 \times T_2}{\Gamma; \Delta \vdash \mathsf{snd}\ E : T_2}\ \textsc{Network - snd}$$

$$\frac{\Gamma; \Delta \vdash E_1 : T_1}{\Gamma; \Delta \vdash \mathsf{left}\ E_1 : T_1 + T_2}\ \textsc{Network - left} \qquad \frac{\Gamma; \Delta \vdash E_2 : T_2}{\Gamma; \Delta \vdash \mathsf{right}\ E_2 : T_1 + T_2}\ \textsc{Network - right}$$

$$\frac{\Gamma \vdash E : T_1 + T_2 \quad \Gamma; \Delta, X : T_1 \vdash E_1 : T_3 \quad \Gamma; \Delta, Y : T_2 \vdash E_2 : T_3}{\Gamma \vdash (\mathsf{match\ E\ with\ left\ X} \Rightarrow E_1\ ;\ \mathsf{right\ Y} \Rightarrow E_2) : T_3}\ \textsc{Network - match}$$

2.3 Choreography

$$\frac{}{\Gamma;\Delta \vdash (\ ) : \mathsf{unit}}\ \textsc{Unit} \qquad \frac{X : \tau \in \Delta}{\Gamma;\Delta \vdash X : \tau}\ \textsc{Var} \qquad \frac{\Gamma_{|\ell} \vdash e : t}{\Gamma;\Delta \vdash \ell.e : \ell.t}\ \textsc{Done} \qquad \frac{\Gamma;\Delta \vdash C : \ell.t}{\Gamma;\Delta \vdash C \rightsquigarrow \ell_2 : \ell_2.t}\ \textsc{Send}$$

$$\frac{\Gamma;\Delta \vdash C : \tau}{\Gamma;\Delta \vdash \ell_1[d] \rightsquigarrow \ell_2; C : \tau}\ \textsc{Sync} \qquad \frac{\Gamma;\Delta \vdash C_1 : \tau_1 \quad \Gamma;\Delta \vdash C_2 : \tau_2 \quad \Gamma;\Delta \vdash C_3 : \tau_2}{\Gamma;\Delta \vdash \mathsf{if}\ C_1\ \mathsf{then}\ C_2\ \mathsf{else}\ C_3 : T_2}\ \textsc{If}$$

$$\frac{\Gamma;\Delta \vdash C_1 : \ell.t \quad \Gamma, \ell.x : t;\Delta \vdash C_2 : \tau_2}{\Gamma;\Delta \vdash \mathsf{let}\ \ell.x = C_1\ \mathsf{in}\ C_2 : \tau_2}\ \textsc{Def} \qquad \frac{\Gamma;\Delta, X : \tau_1 \vdash C : \tau_2}{\Gamma;\Delta \vdash \mathsf{fun}\ X \Rightarrow C : \tau_1 \to \tau_2}\ \textsc{Fun}$$

$$\frac{\Gamma;\Delta \vdash C_1 : \tau_1 \to \tau_2 \quad \Gamma;\Delta \vdash C_2 : \tau_1}{\Gamma;\Delta \vdash C_1 C_2 : \tau_2}\ \textsc{App} \qquad \frac{\Gamma;\Delta \vdash C_1 : \tau_1 \quad \Gamma;\Delta \vdash C_2 : \tau_2}{\Gamma;\Delta \vdash (C_1, C_2) : \tau_1 \times \tau_2}\ \textsc{Pair}$$

$$\frac{\Gamma;\Delta \vdash C : \tau_1 \times \tau_2}{\Gamma;\Delta \vdash \mathsf{fst}\ C : \tau_1}\ \textsc{Fst} \qquad \frac{\Gamma;\Delta \vdash C : \tau_1 \times \tau_2}{\Gamma;\Delta \vdash \mathsf{snd}\ C : \tau_2}\ \textsc{Snd}$$

$$\frac{\Gamma;\Delta \vdash C : \tau_1}{\Gamma;\Delta \vdash \mathsf{left}\ C : \tau_1 + \tau_2}\ \textsc{Left} \qquad \frac{\Gamma;\Delta \vdash C : \tau_2}{\Gamma;\Delta \vdash \mathsf{right}\ C : \tau_1 + \tau_2}\ \textsc{Right}$$

$$\frac{\Gamma;\Delta \vdash C : \tau_1 + \tau_2 \quad \Gamma;\Delta,\ \mathsf{X} : \tau_1 \vdash C_1 : \tau_3 \quad \Gamma;\Delta,\ \mathsf{Y} : \tau_2 \vdash C_2 : \tau_3}{\Gamma;\Delta \vdash (\mathsf{match\ C\ with\ left\ X} \Rightarrow C_1\ ;\ \mathsf{right\ Y} \Rightarrow C_2) :\ \tau_3}\ \textsc{Match}$$

3 Theorems

Theorem 1. (*Local Progress*): For every choreography $\cdot \vdash e : t$ either $\exists\, e'.\ e \to e'$ or e is a value

Proof. We will start with induction on e

Case e = ()

() is a value and we are done

Case e = num

num is a value and we are done

Case e = e_1 binop e_2

IH1 e_1 is either a value or $\exists\, e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then e_1 binop $e_2 \to e_1'$ binop e_2
if e_1 is a value, IH2 e_2 is either a value or $\exists\, e_2'.\ e_2 \to e_2'$
if e_2 is a value, then e_1 binop $e_2 \to$ a value given e_1 and e_2 are numbers
if $e_2 \to e_2'$ then e_1 binop $e_2 \to e_1$ binop e_2'

Case e = let x = e_1 in e_2 //doubt
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then let x = e_1 in $e_2 \to$ let x = e_1' in e_2
if e_1 is a value, IH2 e_2 is either a value or $\exists\ e_2'.\ e_2 \to e_2'$
if e_2 is a value, let x = e_1 in $e_2 \to e_2$
if $e_2 \to e_2'$ then let x = e_1 in $e_2 \to e_2'[x \mapsto e_1]$

Case e = (e_1, e_2)
IH1 e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then $(e_1, e_2) \to (e_1', e_2)$
if e_1 is a value, IH2 e_2 is either a value or $\exists\ e_2'.\ e_2 \to e_2'$
if e_2 is a value, then (e_1, e_2) is a value
if $e_2 \to e_2'$ then $(e_1, e_2) \to (e_1, e_2')$

Case e = fst e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then fst $e_1 \to$ fst e_1'
if e_1 is a value, this means e_1 is a pair of values as $e_1 : t_1 x t_2$ so if $e_1 = (v_1, v_2)$ then fst $e_1 \to v_1$

Case e = snd e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then snd $e_1 \to$ snd e_1'
if e_1 is a value, this means e_1 is a pair of values as $e_1 : t_1 x t_2$ so if $e_1 = (v_1, v_2)$ then snd $e_1 \to v_2$

Case e = left e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then left $e_1 \to$ left e_1'
if e_1 is a value,then left e_1 is a value

Case e = right e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then right $e_1 \to$ right e_1'
if e_1 is a value,then right e_1 is a value

Case e = match e_1 with left $x \Rightarrow e_2$; right $y \Rightarrow e_3$ //doubt
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to e_1'$
if $e_1 \to e_1'$ then match e_1 with left $x \Rightarrow e_2$; right $y \Rightarrow e_3 \to$ match e_1' with left $x \Rightarrow e_2$; right $y \Rightarrow e_3$
if e_1 is a value

Theorem 2. (*Local Preservation*): If $\Gamma \vdash e : t$ and $e \to e'$ then $\Gamma \vdash e' : t$
Proof. We will start with induction on $\Gamma \vdash e : t$

Case $\Gamma \vdash e : unit$
$\Gamma \vdash ( ) :$ unit. It doesn't take a step and we are done

Case e = x doubt
$\Gamma \vdash$ x : t. It doesn't take a step and we are done

Case $\Gamma \vdash e : t_1 X t_2$
This means e = (e_1, e_2)
IH If $\Gamma \vdash e_1 : t_1$ and $e_1 \to e_1'$ then $\Gamma \vdash e_1' : t_1$
Now we know, $(e_1, e_2) \to (e_1', e_2)$ and $\Gamma \vdash (e_1, e_2) : t_1 X t_2$
Using IH we can say $\Gamma \vdash (e_1', e_2) : t_1 X t_2$

Case $\Gamma \vdash$ **fst** $e : t_1$
This means $e : t_1 X t_2$
IH If $\Gamma \vdash e : t_1 X t_2$ and $e \to e'$ then $\Gamma \vdash e' : t_1 X t_2$
Now we know, fst $e \to$ fst e' and $\Gamma \vdash$ fst $e : t_1$
Using IH we can say $\Gamma \vdash$ fst $e' : t_1$

Case $\Gamma \vdash$ **snd** $e : t_2$
This means $e : t_1 X t_2$
IH If $\Gamma \vdash e : t_1 X t_2$ and $e \to e'$ then $\Gamma \vdash e' : t_1 X t_2$
Now we know, snd $e \to$ snd e' and $\Gamma \vdash$ snd $e : t_2$
Using IH we can say $\Gamma \vdash$ snd $e' : t_2$

Case $\Gamma \vdash$ **left** $e : t_1 + t_2$
This means $e : t_1$
IH If $\Gamma \vdash e : t_1$ and $e \to e'$ then $\Gamma \vdash e' : t_1$
Now we know, left $e \to$ left e' and $\Gamma \vdash$ left $e : t_1 + t_2$
Using IH we can say $\Gamma \vdash$ left $e' : t_1 + t_2$

Case $\Gamma \vdash$ **right** $e : t_1 + t_2$
This means $e : t_2$
IH If $\Gamma \vdash e : t_2$ and $e \to e'$ then $\Gamma \vdash e' : t_2$
Now we know, right $e \to$ right e' and $\Gamma \vdash$ right $e : t_1 + t_2$
Using IH we can say $\Gamma \vdash$ right $e' : t_1 + t_2$

Case $\Gamma \vdash$ **match** e **with left** $x \Rightarrow e_2$**; right** $y \Rightarrow e_3 : t_3$
This means $e : t_1 + t_2$
IH If $\Gamma \vdash e : t_1 + t_2$ and $e \to e'$ then $\Gamma \vdash e' : t_1 + t_2$
Now we know, match e with left $x \Rightarrow e_2$; right $y \Rightarrow e_3 \to$ match e' with left $x \Rightarrow e_2$; right $y \Rightarrow e_3$ and $\Gamma \vdash$ match e with left $x \Rightarrow e_2$; right $y \Rightarrow e_3 : t_3$
Using IH we can say $\Gamma \vdash$ match e' with left $x \Rightarrow e_2$; right $y \Rightarrow e_3 : t_3$

4 Operational Semantics

4.1 Network Language

$$\mathsf{fst}(E_1, E_2) \to E_1 \qquad\qquad \mathsf{snd}(E_1, E_2) \to E_2$$

$$(\mathsf{match}\ \ \mathsf{inl}\ \ \mathsf{E}\ \ \mathsf{with}\ \ \mathsf{inl}\ \mathsf{X} \Rightarrow E_1;\ \mathsf{inr}\ \mathsf{Y} \Rightarrow E_2) \to E_1\ [X \mapsto E]$$

$$(\mathsf{match}\ \ \mathsf{inr}\ \ \mathsf{E}\ \ \mathsf{with}\ \ \mathsf{inl}\ \mathsf{X} \Rightarrow E_1;\ \mathsf{inr}\ \mathsf{Y} \Rightarrow E_2) \to E_2\ [Y \mapsto E]$$

4.2 Choreography

$$\mathsf{fst}(C_1, C_2) \to C_1 \qquad\qquad \mathsf{snd}(C_1, C_2) \to C_2$$

$$(\mathsf{match}\ \ \mathsf{inl}\ \ \mathsf{C}\ \ \mathsf{with}\ \ \mathsf{inl}\ \mathsf{X} \Rightarrow C_1;\ \mathsf{inr}\ \mathsf{Y} \Rightarrow C_2) \to C_1\ [X \mapsto C]$$

$$(\mathsf{match}\ \ \mathsf{inr}\ \ \mathsf{C}\ \ \mathsf{with}\ \ \mathsf{inl}\ \mathsf{X} \Rightarrow C_1;\ \mathsf{inr}\ \mathsf{Y} \Rightarrow C_2) \to C_2\ [Y \mapsto C]$$

5 Glossary

$$\ell \text{ involved in } \tau = \quad \ell \in locs(\tau)$$

$\ell \in locs(\tau) =$ getLoc is a function that recursively traverses over τ to construct $locs(\tau)$

$$locs(\tau) = \begin{cases} \phi & \text{if } \tau = \textbf{unit} \\ \{\ell\} & \text{if } \tau = \ell.e \\ \mathsf{getLoc}\ \tau_1 \cup \mathsf{getLoc}\ \tau_2 & \text{if } \tau = \tau_1 \to \tau_2 \text{ or } \tau_1 + \tau_2 \text{ or } \tau_1 \times \tau_2 \end{cases}$$

6 Endpoint Projection

$$[\![(C_1,\ C_2)]\!]_\ell = \begin{cases} ([\![C_1]\!]_\ell,\ [\![C_2]\!]_\ell) & \text{if } (C_1, C_2) : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C_1, C_2, \tau_1,\ \tau_2 \\ \begin{array}{l}\ \mathsf{let}\ \mathsf{x} = [\![C_1]\!]_\ell \ \mathsf{in} \\ \ \mathsf{let}\ \_ = [\![C_2]\!]_\ell \ \mathsf{in}\ \mathsf{x}\end{array} & \text{if } (C_1, C_2) : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C_1, C_2 \text{ and } \tau_1 \text{ but not in } \tau_2 \\ [\![C_1]\!]_\ell; [\![C_2]\!]_\ell & \text{if } (C_1, C_2) : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C_1 \text{ or } C_2 \\ () & \text{otherwise} \end{cases}$$

$$[\![\textbf{fst}\ C]\!]_\ell = \begin{cases} \textbf{fst}\ [\![C]\!]_\ell & \text{if } C : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C, \tau_1 \text{ and } \tau_2 \\ [\![C]\!]_\ell & \text{if } C : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C \\ () & \text{otherwise} \end{cases}$$

$$[\![\textbf{snd} C]\!]_\ell = \begin{cases} \textbf{snd}\ [\![C]\!]_\ell & \text{if } C : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C, \tau_1 \text{ and } \tau_2 \\ [\![C]\!]_\ell & \text{if } C : \tau_1 \times \tau_2 \text{ and } \ell \text{ is involved in } C \\ () & \text{otherwise} \end{cases}$$

$$[\![\textbf{inl } C]\!]_\ell = \begin{cases} \textbf{inl } [\![C]\!]_\ell & \text{if } C : \tau_1 \text{ and } \ell \text{ is involved in } C \text{ and } \tau_1 \\ [\![C]\!]_\ell & \text{if } C : \tau_1 \text{ and } \ell \text{ is involved in } C \text{ but not in } \tau_1 \\ () & \text{if } C : \tau_1 \text{ and } \ell \text{ is not involved in } C \text{ and } \tau_1 \end{cases}$$

$$[\![\textbf{inr } C]\!]_\ell = \begin{cases} \textbf{inr } [\![C]\!]_\ell & \text{if } C : \tau_2 \text{ and } \ell \text{ is involved in } C \text{ and } \tau_2 \\ [\![C]\!]_\ell & \text{if } C : \tau_2 \text{ and } \ell \text{ is involved in } C \text{ but not in } \tau_2 \\ () & \text{if } C : \tau_2 \text{ and } \ell \text{ is not involved in } C \text{ and } \tau_2 \end{cases}$$

$$[\![\textbf{match } C \textbf{ in inl X } \Rightarrow C_1\textbf{; inr Y } \Rightarrow C_2]\!]_\ell$$

$$= \begin{cases} \textbf{match } [\![C]\!]_\ell \textbf{ in inl X } \Rightarrow [\![C_1]\!]_\ell\textbf{; inr Y } \Rightarrow [\![C_2]\!]_\ell & \text{if } C : \tau_1 + \tau_2 \text{ and } \ell \text{ is involved in both } \tau_1 \text{ and } \tau_2 \\ [\![C]\!]_\ell\textbf{; } [\![C_1]\!]_\ell \sqcup [\![C_2]\!]_\ell & \text{if } C : \tau_1 + \tau_2 \text{ and } \ell \text{ is involved in } \tau_1 \text{ or } \tau_2 \text{ or } C \\ [\![C_1]\!]_\ell \sqcup [\![C_2]\!]_\ell & \text{if } C : \tau_1 + \tau_2 \text{ and } \ell \text{ is not involved in } C,\ \tau_1 \text{ and } \tau_2 \end{cases}$$

7 Type Projection

$$[\![\textbf{unit}]\!]_\ell = \textbf{unit}$$

$$[\![\ell_1.t]\!]_{\ell_2} = \begin{cases} t & \text{if } \ell_1 = \ell_2 \\ \textbf{unit} & \text{otherwise} \end{cases}$$

$$[\![\tau_1 \to \tau_2]\!]_\ell = \begin{cases} [\![\tau_1]\!]_\ell \to [\![\tau_2]\!]_\ell & \text{if } \ell \text{ is involved in } \tau_1 \text{ or } \tau_2 \text{ or both} \\ \textbf{unit} & \text{otherwise} \end{cases}$$

$$[\![\tau_1 + \tau_2]\!]_\ell = \begin{cases} [\![\tau_1]\!]_\ell + [\![\tau_2]\!]_\ell & \text{if } \ell \text{ is involved in } \tau_1 \text{ or } \tau_2 \text{ or both} \\ \textbf{unit} & \text{otherwise} \end{cases}$$

$$[\![\tau_1 \times \tau_2]\!]_\ell = \begin{cases} [\![\tau_1]\!]_\ell \times [\![\tau_2]\!]_\ell & \text{if } \ell \text{ is involved in } \tau_1 \text{ and } \tau_2 \\ [\![\tau_1]\!]_\ell & \text{if } \ell \text{ is involved in } \tau_1 \text{ but not } \tau_2 \\ [\![\tau_2]\!]_\ell & \text{if } \ell \text{ is involved in } \tau_2 \text{ but not } \tau_1 \\ \textbf{unit} & \text{otherwise} \end{cases}$$

8 Lemmas

Lemma 1 : If ℓ is not involved in τ then $[\![\tau]\!]_\ell = \mathbf{unit}$
Proof : By Induction on τ

Case $\tau = \mathbf{unit}$:
From type projection we know, $[\![\mathbf{unit}]\!]_\ell = \text{unit}$

Case $\tau = \ell.t$:
Here we know that, $\ell_1 \neq \ell$ as ℓ is not involved in τ
so using type projection for $[\![\ell_1.t]\!]_\ell$
we can say, $[\![\ell_1.t]\!]_\ell = \mathbf{unit}$

Case $\tau = [\![\tau_1 \rightarrow \tau_2]\!]_\ell$:
By IH, ℓ is not involved in τ_1 and ℓ is not involved in τ_2
so using type projection for $[\![\tau_1 \rightarrow \tau_2]\!]_\ell$
we can say, $[\![\tau_1 \rightarrow \tau_2]\!]_\ell = \mathbf{unit}$

Case $\tau = [\![\tau_1 + \tau_2]\!]_\ell$:
By IH, ℓ is not involved in τ_1 and ℓ is not involved in τ_2
so using type projection for $[\![\tau_1 + \tau_2]\!]_\ell$
we can say, $[\![\tau_1 + \tau_2]\!]_\ell = \mathbf{unit}$

Case $\tau = [\![\tau_1 \times \tau_2]\!]_\ell$:
By IH, ℓ is not involved in τ_1 and ℓ is not involved in τ_2
so using type projection for $[\![\tau_1 \times \tau_2]\!]_\ell$
we can say, $[\![\tau_1 * \tau_2]\!]_\ell = \mathbf{unit}$

Lemma 2 : If $\ell \notin locs(C)$ then $[\![C]\!]_\ell = ()$

Lemma 3 : If $\vdash C : \tau,$ then $\vdash [\![C]\!]_\ell : [\![\tau]\!]_\ell$