Pirouette - Theory

Shruti Gupta **Prashant Godhwani**

February 2024

1 Syntax

$$
\begin{array}{llcl}
\text{Locations} & \ell & \in & \mathcal{L} \\
\text{Synchronization Labels} & d & ::= & L \mid R \\
\text{Binary Operations} & \odot & ::= & + \mid - \mid * \mid / \mid = \mid <= \mid >= \mid != \mid > \mid < \mid \&\& \mid \| \\
\text{Choreography} & C & ::= & () \mid X \mid \ell.e \mid C \rightsquigarrow \ell \mid \textbf{if } C \textbf{ then } C_1 \textbf{ else } C_2 \\
 & & \mid & \ell_1[d] \rightsquigarrow \ell_2;\ C \mid \textbf{let } 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

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

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

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

2.2 Network Language

$$\textsc{Network - unit}\quad \frac{}{\Gamma; \Delta \vdash (\ ) : \mathsf{unit}}$$

$$\textsc{Network - var}\quad \frac{X : T \in \Delta}{\Gamma; \Delta \vdash X : T}$$

$$\textsc{ret}\quad \frac{\Gamma; \Delta \vdash e : t}{\Gamma; \Delta \vdash \mathsf{ret}\ (e) : \boxed{t}}$$

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

$$\textsc{Network - app}\quad \frac{\Gamma; \Delta \vdash E_1 : T_1 \to T_2 \quad \Gamma; \Delta \vdash E_2 : T_1}{\Gamma; \Delta \vdash E_1 E_2 : T_2}$$

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

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

$$\textsc{Network - send}\quad \frac{\Gamma; \Delta \vdash e : t \quad \Gamma; \Delta \vdash E : T}{\Gamma; \Delta \vdash \mathsf{send}\ e\ \mathsf{to}\ \ell; E : T}$$

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

$$\textsc{Network - choose}\quad \frac{\Gamma; \Delta \vdash E : T}{\Gamma; \Delta \vdash \mathsf{choose}\ d\ \mathsf{for}\ \ell; E : T}$$

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

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

$$\textsc{Network - fst}\quad \frac{\Gamma; \Delta \vdash E : T_1 \times T_2}{\Gamma; \Delta \vdash \mathsf{fst}\ E : T_1}$$

$$\textsc{Network - snd}\quad \frac{\Gamma; \Delta \vdash E : T_1 \times T_2}{\Gamma; \Delta \vdash \mathsf{snd}\ E : T_2}$$

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

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

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

2.3 Choreography

$$\frac{}{\Gamma;\Delta \vdash (\ ) : \mathsf{unit}}\ \text{UNIT} \qquad \frac{X : \tau \in \Delta}{\Gamma;\Delta \vdash X : \tau}\ \text{VAR} \qquad \frac{\Gamma_{|\ell} \vdash e : t}{\Gamma;\Delta \vdash \ell.e : \ell.t}\ \text{DONE} \qquad \frac{\Gamma;\Delta \vdash C : \ell.t}{\Gamma;\Delta \vdash C \rightsquigarrow \ell_2 : \ell_2.t}\ \text{SEND}$$

$$\frac{\Gamma;\Delta \vdash C : \tau}{\Gamma;\Delta \vdash \ell_1[d] \rightsquigarrow \ell_2; C : \tau}\ \text{SYNC} \qquad \frac{\Gamma;\Delta \vdash C_1 : \ell.t \quad \Gamma;\Delta \vdash C_2 : \tau_2 \quad \Gamma;\Delta \vdash C_3 : \tau_2}{\Gamma;\Delta \vdash \mathsf{if}\ C_1\ \mathsf{then}\ C_2\ \mathsf{else}\ C_3 : T_2}\ \text{IF}$$

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

$$\frac{\Gamma;\Delta \vdash C_1 : \tau_1 \to \tau_2 \quad \Gamma;\Delta \vdash C_2 : \tau_1}{\Gamma;\Delta \vdash C_1 C_2 : \tau_2}\ \text{APP} \qquad \frac{\Gamma;\Delta \vdash C_1 : \tau_1 \quad \Gamma;\Delta \vdash C_2 : \tau_2}{\Gamma;\Delta \vdash (C_1, C_2) : \tau_1 \times \tau_2}\ \text{PAIR}$$

$$\frac{\Gamma;\Delta \vdash C : \tau_1 \times \tau_2}{\Gamma;\Delta \vdash \mathsf{fst}\ C : \tau_1}\ \text{FST} \qquad \frac{\Gamma;\Delta \vdash C : \tau_1 \times \tau_2}{\Gamma;\Delta \vdash \mathsf{snd}\ C : \tau_2}\ \text{SND}$$

$$\frac{\Gamma;\Delta \vdash C : \tau_1}{\Gamma;\Delta \vdash \mathsf{left}\ C : \tau_1 + \tau_2}\ \text{LEFT} \qquad \frac{\Gamma;\Delta \vdash C : \tau_2}{\Gamma;\Delta \vdash \mathsf{right}\ C : \tau_1 + \tau_2}\ \text{RIGHT}$$

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

3 Operational Semantics

3.1 Local Language

$$\text{Local values} \quad v \ ::= \ () \mid num \mid v_1 \odot v_2 \mid (v_1, v_2) \mid \mathbf{left}\ v \mid \mathbf{right}\ v$$

$$\frac{e_1 \to_l e_1'}{e_1 \odot e_2 \to_l e_1' \tilde{\odot} e_2}\ \textsc{binop 1} \qquad \frac{e_2 \to_l e_2'}{v \odot e_2 \to_l v \tilde{\odot} e_2'}\ \textsc{binop 2} \qquad \frac{}{v_1 \odot v_2 \to_l v_1 \tilde{\odot} v_2\ (/ \notin \odot)}\ \textsc{binop 3}$$

$$\frac{e_1 \to_l e_1'}{\mathsf{let}\ x := e_1\ \mathsf{in}\ e_2 \to_l \mathsf{let}\ x := e_1'\ \mathsf{in}\ e_2}\ \textsc{Loc - let 1} \qquad \frac{}{\mathsf{let}\ x := v\ \mathsf{in}\ e \to_l e[x \mapsto v]}\ \textsc{Loc - let 2}$$

$$\frac{e_1 \to_l e_1'}{(e_1, e_2) \to_l (e_1', e_2)}\ \textsc{loc - pair 1} \qquad \frac{e_2 \to_l e_2'}{(v, e_2) \to_l (v, e_2')}\ \textsc{loc - pair 2} \qquad \frac{e \to_l e'}{\mathsf{fst}\ e \to_l \mathsf{fst}\ e'}\ \textsc{loc - fst}$$

$$\frac{}{\mathsf{fst}\ (v_1, v_2) \to_l v_1}\ \textsc{loc - pair elim 1}$$

$$\frac{e \to_l e'}{\mathsf{snd}\ e \to_l \mathsf{snd}\ e'}\ \textsc{loc - snd} \qquad \frac{}{\mathsf{snd}\ (v_1, v_2) \to_l v_2}\ \textsc{loc - pair elim 2} \qquad \frac{e \to_l e'}{\mathsf{left}\ e \to_l \mathsf{left}\ e'}\ \textsc{loc - left} \qquad \frac{e \to_l e'}{\mathsf{right}\ e \to_l \mathsf{right}\ e'}\ \textsc{loc - right}$$

$$\frac{e \to_l e'}{(\mathsf{match\ e\ with\ left\ x} \Rightarrow e_1\ ;\ \mathsf{right\ y} \Rightarrow e_2) \to_l (\mathsf{match\ e'\ with\ left\ x} \Rightarrow e_1\ ;\ \mathsf{right\ y} \Rightarrow e_2)}\ \textsc{Loc - match}$$

$$\frac{}{(\mathsf{match\ (left\ v)\ with\ left\ x} \Rightarrow e_1\ ;\ \mathsf{right\ y} \Rightarrow e_2) \to_l e_1\ [x \mapsto v]}\ \textsc{loc - Sum elim 1}$$

$$\frac{}{(\mathsf{match\ (right\ v)\ with\ left\ x} \Rightarrow e_1\ ;\ \mathsf{right\ y} \Rightarrow e_2) \to_l e_2\ [y \mapsto v]}\ \textsc{loc - sum elim 2}$$

3.2 NetIR

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

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

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

3.3 Choreography

Choreographic Values $\quad V \quad ::= \quad () \mid \ell.v \mid (V_1, V_2) \mid \textbf{left}\ V \mid \textbf{right}\ V \mid \textbf{fun}\ X \Rightarrow C$

$$\textsc{assoc}\ \frac{e \rightarrow_l e'}{\ell.e \rightarrow_g \ell.e'}$$

$$\textsc{send 1}\ \frac{C \rightarrow_g C'}{C \rightsquigarrow \ell \rightarrow_g C' \rightsquigarrow \ell}$$

$$\textsc{send 2}\ \frac{}{\ell.v \rightsquigarrow \ell' \rightarrow_g \ell'.v}$$

$$\textsc{sync}\ \frac{}{\ell_1[d] \rightsquigarrow \ell_2; C \rightarrow_g C}$$

$$\textsc{let 1}\ \frac{C_1 \rightarrow_g C_1'}{\mathsf{let}\ X := C_1\ \mathsf{in}\ C_2 \rightarrow_g \mathsf{let}\ X := C_1'\ \mathsf{in}\ C_2}$$

$$\textsc{let 2}\ \frac{}{\mathsf{let}\ X := V\ \mathsf{in}\ C \rightarrow_g C[X \mapsto V]}$$

$$\textsc{If 1}\ \frac{C \rightarrow_g C'}{\mathsf{if}\ C\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \rightarrow_g \mathsf{if}\ C'\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2}$$

$$\textsc{If 2}\ \frac{}{\mathsf{if}\ \ell.true\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \rightarrow_g C_1}$$

$$\textsc{If 3}\ \frac{}{\mathsf{if}\ \ell.false\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \rightarrow_g C_2}$$

$$\textsc{App 1}\ \frac{C_1 \rightarrow_g C_1'}{C_1 C_2 \rightarrow_g C_1' C_2}$$

$$\textsc{app 2}\ \frac{C_2 \rightarrow_g C_2'}{(\mathsf{fun}\ X \Rightarrow \mathsf{C})\ C_2 \rightarrow_g (\mathsf{fun}\ X \Rightarrow \mathsf{C})\ C_2'}$$

$$\textsc{app 3}\ \frac{}{(\mathsf{fun}\ X \Rightarrow \mathsf{C})\ V \rightarrow_g C[X \mapsto V]}$$

$$\textsc{pair 1}\ \frac{C_1 \rightarrow_g C_1'}{(C_1, C_2) \rightarrow_g (C_1', C_2)}$$

$$\textsc{pair 2}\ \frac{C_2 \rightarrow_g C_2'}{(V, C_2) \rightarrow_g (V, C_2')}$$

$$\textsc{fst}\ \frac{C \rightarrow_g C'}{\mathsf{fst}\ C \rightarrow_g \mathsf{fst}\ C'}$$

$$\textsc{pair elim 1}\ \frac{}{\mathsf{fst}\ (V_1, V_2) \rightarrow_g V_1}$$

$$\textsc{snd}\ \frac{C \rightarrow_g C'}{\mathsf{snd}\ C \rightarrow_g \mathsf{snd}\ C'}$$

$$\textsc{pair elim 2}\ \frac{}{\mathsf{snd}\ (V_1, V_2) \rightarrow_g V_2}$$

$$\textsc{left}\ \frac{C \rightarrow_g C'}{\mathsf{left}\ C \rightarrow_g \mathsf{left}\ C'}$$

$$\textsc{right}\ \frac{C \rightarrow_g C'}{\mathsf{right}\ C \rightarrow_g \mathsf{right}\ C'}$$

$$\textsc{match}\ \frac{C \rightarrow_g C'}{(\mathsf{match\ C\ with\ left\ X} \Rightarrow C_1\ ;\ \mathsf{right\ Y} \Rightarrow C_2) \rightarrow_g (\mathsf{match\ C'\ with\ left\ X} \Rightarrow C_1\ ;\ \mathsf{right\ Y} \Rightarrow C_2)}$$

$$\textsc{sum elim 1}\ \frac{}{(\mathsf{match\ (left\ V)\ with\ left\ X} \Rightarrow C_1\ ;\ \mathsf{right\ Y} \Rightarrow C_2) \rightarrow_g C_1\ [X \mapsto V]}$$

$$\textsc{sum elim 2}\ \frac{}{(\mathsf{match\ (right\ V)\ with\ left\ X} \Rightarrow C_1\ ;\ \mathsf{right\ Y} \Rightarrow C_2) \rightarrow_g C_2\ [Y \mapsto V]}$$

4 Theorems

Theorem 1. (*Local Progress*): For every expression $\cdot \vdash e : t$ either $\exists\ e'.\ e \to_l e'$ 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 = $e_1 \odot e_2$
IH1 e_1 is either a value or $\exists\ e_1'.\ e_1 \to_l e_1'$
if $e_1 \to_l e_1'$ then $e_1 \odot e_2 \to_l e_1' \tilde{\odot}\ e_2$ using binop 1 rule
if e_1 is a value v_1, IH2 e_2 is either a value or $\exists\ e_2'.\ e_2 \to_l e_2'$
if e_2 is a value v_2, then $v_1 \odot v_2 \to_l v_1 \tilde{\odot}\ v_2$, which is a value using the binop 3 rule
if $e_2 \to_l e_2'$ then $v_1 \odot e_2 \to_l v_1 \tilde{\odot}\ e_2'$ using binop 2 rule

Case e = let $x := e_1$ **in** e_2
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to_l e_1'$
if $e_1 \to_l e_1'$ then let $x := e_1$ in $e_2 \to_l$ let $x := e_1'$ in e_2 using loc - let 1 rule
if e_1 is a value v, then let $x := v$ in $e_2 \to_l e_2[x \mapsto v]$ using loc - let 2 rule

Case e = (e_1, e_2)
IH1 e_1 is either a value or $\exists\ e_1'.\ e_1 \to_l e_1'$
if $e_1 \to_l e_1'$ then $(e_1, e_2) \to_l (e_1', e_2)$ using loc - pair 1 rule
if e_1 is a value v_1, IH2 e_2 is either a value or $\exists\ e_2'.\ e_2 \to_l e_2'$
if e_2 is a value v_2, then (v_1, v_2) is a value
if $e_2 \to_l e_2'$ then $(v_1, e_2) \to_l (v_1, e_2')$ using loc - pair 2 rule

Case e = fst e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to_l e_1'$
if $e_1 \to_l e_1'$ then fst $e_1 \to_l$ fst e_1' using loc - fst rule
if e_1 is a value v, this means $v = (v_1, v_2)$ then fst $(v_1, v_2) \to_l v_1$ using loc - pair elim 1 rule

Case e = snd e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to_l e_1'$
if $e_1 \to_l e_1'$ then snd $e_1 \to_l$ snd e_1' using loc - snd rule
if e_1 is a value v, this means $v = (v_1, v_2)$ then snd $(v_1, v_2) \to_l v_2$ using loc - pair elim 2 rule

Case e = left e_1
IH e_1 is either a value or $\exists\ e_1'.\ e_1 \to_l e_1'$

if $e_1 \rightarrow_l e_1'$ then left $e_1 \rightarrow_l$ left e_1' using loc - left rule
if e_1 is a value v, then left v is a value

Case e = right e_1
IH e_1 is either a value or $\exists\, e_1'.\ e_1 \rightarrow_l e_1'$
if $e_1 \rightarrow_l e_1'$ then right $e_1 \rightarrow_l$ right e_1' using loc - right rule
if e_1 is a value v, then right v is a value

Case e = match e_1 **with left x** $\Rightarrow e_2$**; right y** $\Rightarrow e_3$
IH e_1 is either a value or $\exists\, e_1'.\ e_1 \rightarrow_l e_1'$
if $e_1 \rightarrow_l e_1'$ then match e_1 with left x $\Rightarrow e_2$; right y $\Rightarrow e_3 \rightarrow_l$ match e_1' with left x $\Rightarrow e_2$; right y $\Rightarrow e_3$ using loc - match rule
if e_1 is a value, then e_1 is either left v or right v
if $e_1 =$ left v then match (left v) with left x $\Rightarrow e_2$; right y $\Rightarrow e_3 \rightarrow_l e_2[x \mapsto v]$ using loc - sum elim 1 rule
if $e_1 =$ right v then match (right v) with left x $\Rightarrow e_2$; right y $\Rightarrow e_3 \rightarrow_l e_3[y \mapsto v]$ using loc - sum elim 2 rule

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

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

Case x : t
$\Gamma \vdash$ x : t. x 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 $e_1 \rightarrow e_1'$ then $\Gamma \vdash e_1' : t_1$
Now we know, $(e_1, e_2) \rightarrow (e_1', e_2)$ 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 $e \rightarrow e'$ then $\Gamma \vdash e' : t_1 X t_2$
Now we know, fst $e \rightarrow$ 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 $e \rightarrow e'$ then $\Gamma \vdash e' : t_1 X t_2$
Now we know, snd $e \rightarrow$ 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 $\Gamma; x : t_1, y : t_2 \vdash$ **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 $y \Rightarrow e_3 : t_3$

Theorem 3. (*Progress*): For every choreography $\cdot \vdash C : \tau$ either $\exists\, C'.\ C \to_g C'$ or C is a value
Proof. We will start with induction on C

Case C = ()
() is a value and we are done

Case C = $\ell.e$
we know from local progress that e is either a value or $\exists\, e'.\ e \to_l e'$
So, if e is a value v, $\ell.v$ is a value and we are done
If $\exists\, e'.\ e \to_l e'$ then, $\ell.e \to_g \ell.e'$ using assoc rule

Case C = $C_1 \rightsquigarrow \ell$
IH C_1 is either a value or $\exists\, C_1'.\ C_1 \to_g C_1'$
if $C_1 \to_g C_1'$ then $C_1 \rightsquigarrow \ell \to_g C_1' \rightsquigarrow \ell$ using the send 1 rule
if C_1 is a value, $C_1 = \ell'.v$ and $\ell'.v \rightsquigarrow \ell \to_g \ell.v$ using send 2 rule

Case C = $\ell_1[d] \rightsquigarrow \ell_2; C_1$
we know that, $\ell_1[d] \rightsquigarrow \ell_2; C_1 \to_g C_1$ using sync rule

Case C = if C_1 **then** C_2 **else** C_3
IH1 C_1 is either a value or $\exists\, C_1'.\ C_1 \to_g C_1'$
if $C_1 \to C_1'$ then if C_1 then C_2 else $C_3 \to_g$ if C_1' then C_2 else C_3 using if 1 rule
if C_1 is a value, then C_1 is either $\ell.true$ or $\ell.false$
if $C_1 = \ell.true$ then if C_1 then C_2 else $C_3 \to_g C_2$ using if 2 rule

if $C_1 = \ell.false$ then if C_1 then C_2 else $C_3 \rightarrow_g C_3$ using if 3 rule

Case C = let $X := C_1$ **in** C_2
IH C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then let $X := C_1$ in $C_2 \rightarrow_g$ let $X := C_1'$ in C_2 using let 1 rule
if C_1 is a value V, then let $X := V$ in $C_2 \rightarrow_g C_2[X \mapsto V]$ using let 2 rule

Case C = fun $X \Rightarrow C_1$
fun $X \Rightarrow C_1$ is a value and we are done

Case C = $C_1 C_2$
IH1 C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then $C_1 C_2 \rightarrow_g C_1' C_2$ using app 1 rule
if C_1 is a value, fun $X \Rightarrow C'$, IH2 C_2 is either a value or $\exists\, C_2'.\ C_2 \rightarrow_g C_2'$
if C_2 is a value V, then (fun $X \Rightarrow C'$)$V \rightarrow_g C'[X \mapsto V]$ using app 3 rule
if $C_2 \rightarrow_g C_2'$ then (fun $X \Rightarrow C'$)$C_2 \rightarrow_g$ (fun $X \Rightarrow C'$)C_2' using app 2 rule

Case C = (C_1, C_2)
IH1 C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then $(C_1, C_2) \rightarrow_g (C_1', C_2)$ using pair 1 rule
if C_1 is a value V_1, IH2 C_2 is either a value or $\exists\, C_2'.\ C_2 \rightarrow_g C_2'$
if C_2 is a value V_2, then (V_1, V_2) is a value
if $C_2 \rightarrow_g C_2'$ then $(V_1, C_2) \rightarrow (V_1, C_2')$ using pair 2 rule

Case C = fst C_1
IH C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then fst $C_1 \rightarrow_g$ fst C_1' using fst rule
if C_1 is a value V, this means $V = (V_1, V_2)$ then fst $(V_1, V_2) \rightarrow_g V_1$ using pair elim 1 rule

Case C = snd C_1
IH C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then snd $C_1 \rightarrow_g$ snd C_1' using snd rule
if C_1 is a value V, this means $V = (V_1, V_2)$ then snd $(V_1, V_2) \rightarrow_g V_2$ using pair elim 2 rule

Case C = left C_1
IH C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then left $C_1 \rightarrow_g$ left C_1' using left rule
if C_1 is a value V,then left V is a value

Case C = right C_1
IH C_1 is either a value or $\exists\, C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then right $C_1 \rightarrow_g$ right C_1' using right rule
if C_1 is a value V,then right V is a value

Case C = match C_1 with left X $\Rightarrow C_2$; right Y $\Rightarrow C_3$
IH C_1 is either a value or $\exists\ C_1'.\ C_1 \rightarrow_g C_1'$
if $C_1 \rightarrow_g C_1'$ then match C_1 with left X $\Rightarrow C_2$; right Y $\Rightarrow C_3 \rightarrow_g$ match C_1' with left X $\Rightarrow C_2$; right Y $\Rightarrow C_3$ using match rule
if C_1 is a value, then C_1 is either left V or right V
if C_1 = left V then match (left V) with left X $\Rightarrow C_2$; right Y $\Rightarrow C_3 \rightarrow_g C_2[X \mapsto V]$ using sum elim 1 rule
if C_1 = right V then match (right V) with left X $\Rightarrow C_2$; right Y $\Rightarrow C_3 \rightarrow_g C_3[Y \mapsto V]$ using sum elim 2 rule

Theorem 4. (*Preservation*): If $\Gamma; \Delta \vdash C : \tau$ and $C \rightarrow C'$ then $\Gamma; \Delta \vdash C' : \tau$
Proof. We will start with induction on $\Gamma; \Delta \vdash C : \tau$

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

Case $\Gamma; \Delta \vdash X : \tau$
$\Gamma; \Delta \vdash X : \tau$. X doesn't take a step and we are done

Case $\Gamma; \Delta \vdash C : \ell.t$
This means C = $\ell.e$
Using local preservation we can say that, if $\Gamma \vdash e : t$ and $e \rightarrow e'$ then $\Gamma \vdash e' : t$
Now we know, $\ell.e \rightarrow \ell.e'$ and $\Gamma; \Delta; \Delta \vdash C : \ell.t$
Using local preservation, $\Gamma; \Delta \vdash C' : \ell.t$ where $C' = \ell.e'$

Case $\Gamma; \Delta \vdash C : \tau_1$ **x** τ_2
This means C = (C_1, C_2)
IH If $\Gamma; \Delta \vdash C_1 : \tau_1$ and $C_1 \rightarrow C_1'$ then $\Gamma; \Delta \vdash C_1' : \tau_1$
Now we know, $(C_1, C_2) \rightarrow (C_1', C_2)$ and $\Gamma; \Delta \vdash (C_1, C_2) : \tau_1$ x τ_2
Using IH we can say $\Gamma; \Delta \vdash (C_1', C_2) : \tau_1$ x τ_2

Case $\Gamma; \Delta \vdash$ **fst** $C : \tau_1$
This means $C : \tau_1$ x τ_2
IH If $\Gamma; \Delta \vdash C : \tau_1$ x τ_2 and $C \rightarrow c'$ then $\Gamma; \Delta \vdash C' : \tau_1$ x τ_2
Now we know, fst $C \rightarrow$ fst C' and $\Gamma; \Delta \vdash$ fst $C : \tau_1$
Using IH we can say $\Gamma; \Delta \vdash$ fst $C' : \tau_1$

Case $\Gamma; \Delta \vdash$ **snd** $C : \tau_2$
This means $C : \tau_1$ x τ_2
IH If $\Gamma; \Delta \vdash C : \tau_1$ x τ_2 and $C \rightarrow C'$ then $\Gamma; \Delta \vdash C' : \tau_1$ x τ_2
Now we know, snd $C \rightarrow$ snd C' and $\Gamma; \Delta \vdash$ snd $C : \tau_2$
Using IH we can say $\Gamma; \Delta \vdash$ snd $C' : \tau_2$

Case $\Gamma; \Delta \vdash$ **left** $C : \tau_1 + \tau_2$
This means $C : \tau_1$

IH If $\Gamma; \Delta \vdash C : \tau_1$ and $C \to C'$ then $\Gamma; \Delta \vdash C' : \tau_1$
Now we know, left $C \to$ left C' and $\Gamma; \Delta \vdash$ left $C : \tau_1 + \tau_2$
Using IH we can say $\Gamma; \Delta \vdash$ left $C' : \tau_1 + \tau_2$

Case $\Gamma; \Delta \vdash$ **right** $C : \tau_1 + \tau_2$
This means $C : \tau_2$
IH If $\Gamma; \Delta \vdash C : \tau_2$ and $C \to C'$ then $\Gamma; \Delta \vdash C' : \tau_2$
Now we know, right $C \to$ right C' and $\Gamma; \Delta \vdash$ right $C : \tau_1 + \tau_2$
Using IH we can say $\Gamma; \Delta \vdash$ right $C' : \tau_1 + \tau_2$

Case $\Gamma; \Delta \vdash$ **match** C **with left** $X \Rightarrow C_2$**; right** $Y \Rightarrow C_3 : \tau_3$
This means $C : \tau_1 + \tau_2$
IH If $\Gamma; \Delta \vdash C : \tau_1 + \tau_2$ and $C \to C'$ then $\Gamma; \Delta \vdash C' : \tau_1 + \tau_2$
Now we know, match C with left $X \Rightarrow C_2$; right $Y \Rightarrow C_3 \to$ match C' with left $X \Rightarrow C_2$; right $Y \Rightarrow C_3$ and $\Gamma; \Delta \vdash$ match C with left $X \Rightarrow C_2$; right $Y \Rightarrow C_3 : \tau_3$
Using IH we can say, $\Gamma; \Delta \vdash$ match C' with left $X \Rightarrow C_2$; right $Y \Rightarrow C_3 : \tau_3$

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 $locs(\tau)$

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

6 Endpoint Projection

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

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

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

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

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

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

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

7 Type Projection

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

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

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

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

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

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

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 $\vdash [\![C]\!]_\ell : [\![\tau]\!]_\ell$