Supplementary Material

ANONYMOUS AUTHOR(S)

1 OVERVIEW

This article supplements the submission "Resource-Aware Session Types for Digital Contracts". The main contributions of the supplementary document are as follows.

- Section 2 presents the type grammar.
- Section 3 presents the process typing rules, concerning the judgment $\Psi \; ; \; \Gamma \; ; \; \Delta \vdash^{q} P :: (x_m : A)$. This judgment types a process in state P providing service of type A along channel x at mode m. Moreover, the process uses functional variables from Ψ, shared channels from Γ and linear channels from Δ. Finally, the process stores potential q.
- Section 4 presents the rules of the operational cost semantics. These discuss the behavior of the semantic objects $\mathsf{proc}(c_m, w, P)$ and $\mathsf{msg}(c_m, w, N)$ defining a process P (or message N) offering along channel c at mode m which has performed work w so far.
- Section 5 presents the rules corresponding to configuration typing and other helper judgments. The configuration typing judgment $\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \; ; \; \Delta)$ describes a well-typed configuration Ω which offers shared channels in Γ and linear channels in Δ.
- Section 6 is the main contribution of the supplementary material. It presents and proves the main theorem of type safety of our language. This is split into a type preservation and a progress theorem. The section also proves the lemmas necessary for the type safety theorems.

2 TYPES

First, I present the grammar for ordinary functional types τ with potential.

$$
\begin{array}{rcl}
\tau & ::= & t \mid \tau \to \tau \mid \tau + \tau \mid \tau \times \tau \\
& \mid & \mathsf{int} \mid \mathsf{bool} \mid L^{q}(\tau) \\
& \mid & \{A_{\mathsf{R}} \leftarrow \overline{A_{\mathsf{R}}}\}_{\mathsf{R}} \mid \{A_{\mathsf{S}} \leftarrow \overline{A_{\mathsf{S}}} \; ; \; \overline{A_{\mathsf{R}}}\}_{\mathsf{S}} \mid \{A_{\mathsf{T}} \leftarrow \overline{A_{\mathsf{S}}} \; ; \; \overline{A}\}_{\mathsf{T}}
\end{array}
$$

Next, I define the purely linear session types.

$$
\begin{array}{rcl}
A_{\mathsf{R}} & ::= & V \mid \oplus\{\ell : A_{\mathsf{R}}\}_{\ell \in L} \mid \&\{\ell : A_{\mathsf{R}}\}_{\ell \in L} \mid A_m \multimap_m A_{\mathsf{R}} \mid A_m \otimes_m A_{\mathsf{R}} \mid \mathbf{1} \\
& \mid & \tau \to A_{\mathsf{R}} \mid \tau \times A_{\mathsf{R}} \mid \triangleright^{r} A_{\mathsf{R}} \mid \triangleleft^{r} A_{\mathsf{R}}
\end{array}
$$

2018. Manuscript submitted to ACM

Next, the shared linear session types.

$$
\begin{array}{rcl}
A_{\mathsf{L}} & ::= & V \mid \oplus\{\ell : A_{\mathsf{L}}\}_{\ell \in L} \mid \&\{\ell : A_{\mathsf{L}}\}_{\ell \in L} \mid A_m \multimap_m A_{\mathsf{L}} \mid A_m \otimes_m A_{\mathsf{L}} \\
& \mid & \tau \to A_{\mathsf{L}} \mid \tau \times A_{\mathsf{L}} \mid \triangleright^r A_{\mathsf{L}} \mid \triangleleft^r A_{\mathsf{L}} \\
& \mid & \downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}}
\end{array}
$$

Finally, the shared session type.

$$A_{\mathsf{S}} \quad ::= \quad \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}$$

The client linear types follow the same grammar as purely linear types. The combined type is represented using A which denotes the type of either a client or contract process in linear mode.

$$
\begin{array}{rcl}
A_{\mathsf{T}} & ::= & A_{\mathsf{R}} \\
A & ::= & A_{\mathsf{T}} \mid A_{\mathsf{L}}
\end{array}
$$

First, the expressions at the functional layer are as follows (usual terms from a functional language).

$$
\begin{array}{rcl}
M, N & ::= & \lambda x : \tau . M_x \mid M\ N \\
& \mid & l \cdot M \mid r \cdot M \mid \mathsf{case}\ M\ (l \hookrightarrow M_l, r \hookrightarrow M_r) \\
& \mid & \langle M, N \rangle \mid M \cdot l \mid M \cdot r \\
& \mid & n \mid \mathsf{true} \mid \mathsf{false} \\
& \mid & [] \mid M :: N \mid \mathsf{match}\ M\ ([] \to M_1, x :: xs \to M_2) \\
& \mid & \{c_{\mathsf{R}} \leftarrow P_{c_{\mathsf{R}}, \overline{a}} \leftarrow \overline{a}\} \mid \{c_{\mathsf{S}} \leftarrow P_{c_{\mathsf{S}}, \overline{a}, \overline{d}} \leftarrow \overline{a}\ ;\ \overline{d}\} \mid \{c_{\mathsf{T}} \leftarrow P_{c_{\mathsf{T}}, \overline{a}, \overline{b}} \leftarrow \overline{a}\ ;\ \overline{b}\}
\end{array}
$$

The processes (proof terms) are as follows.

$$
\begin{array}{rcll}
P, Q & ::= & c \leftarrow M \leftarrow \overline{a}\ ;\ P_c & \text{spawn process computed by } M \text{ and continue with } P_a, \\
& & & \text{both communicating along fresh channel } a \\
& \mid & x \leftarrow y & \text{forward between } x \text{ and } y \\
& \mid & x.l_k\ ;\ P & \text{send label } l_k \text{ along x} \\
& \mid & \mathsf{case}\ x\ (l_i \Rightarrow P) & \text{branch on received label along } x \\
& \mid & \mathsf{send}\ x\ w\ ;\ P & \text{send channel/value } w \text{ along } x \\
& \mid & y \leftarrow \mathsf{recv}\ x\ ;\ P & \text{receive channel/value along } x \text{ and bind it to } y \\
& \mid & \mathsf{close}\ x & \text{close channel } x \\
& \mid & \mathsf{wait}\ x\ ;\ P & \text{wait on closing channel } x \\
& \mid & \mathsf{work}\ \{p\}\ ;\ P & \text{do work } p\text{, continue with } P \\
& \mid & \mathsf{get}\ x\ \{p\}\ ;\ P & \text{get potential } p \text{ on channel } x \\
& \mid & \mathsf{pay}\ x\ \{p\}\ ;\ P & \text{pay potential } p \text{ on channel } x \\
& \mid & x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ;\ P_{x_{\mathsf{L}}} & \text{send an acquire request along } x_{\mathsf{S}} \\
& \mid & x_{\mathsf{L}} \leftarrow \mathsf{accept}\ x_{\mathsf{S}}\ ;\ P_{x_{\mathsf{L}}} & \text{accept an acquire request along } x_{\mathsf{S}} \\
& \mid & x_{\mathsf{S}} \leftarrow \mathsf{detach}\ x_{\mathsf{L}}\ ;\ P_{x_{\mathsf{S}}} & \text{send a detach request along } x_{\mathsf{L}} \\
& \mid & x_{\mathsf{S}} \leftarrow \mathsf{release}\ x_{\mathsf{L}}\ ;\ P_{x_{\mathsf{S}}} & \text{receive a detach request along } x_{\mathsf{L}}
\end{array}
$$

3 TYPE SYSTEM

We first define the judgments we use in our type system.

$\Psi \Vdash^{q} M : \tau$	term M has type τ
	and needs potential q for evaluation
$\Psi \,;\, \Gamma \,;\, \Delta \vDash^{q} P :: (c_m : A)$	process P offers service of type A
	along channel c at mode $m = (\mathsf{S}, \mathsf{L}, \mathsf{T}, \mathsf{R})$
	and uses shared channels from Γ
	and linear channels from Δ
	and functional variables from Ψ
	and stores potential q

Mode S stands for channels in shared mode. Mode L stands for shared channels in their linear mode. Mode T stands for linear channels that internally depend on shared processes. Mode R stands for purely linear channels offered by purely linear processes.

3.1 Monad

First, I present the rules concerning the monad.

Introduction Rules.

$$\frac{\Delta = \overline{d_{\mathsf{R}} : D_{\mathsf{R}}} \qquad \Psi \,;\, \cdot \,;\, \Delta \vDash^{q} P :: (x_{\mathsf{R}} : A_{\mathsf{R}})}{\Psi \Vdash^{q} \{x_{\mathsf{R}} \leftarrow P \leftarrow \overline{d_{\mathsf{R}}}\} : \{A_{\mathsf{R}} \leftarrow \overline{D_{\mathsf{R}}}\}_{\mathsf{R}}} \ \{\}I_{\mathsf{R}}$$

$$\frac{\Gamma = \overline{a_{\mathsf{S}} : A_{\mathsf{S}}} \qquad \Delta = \overline{d_{\mathsf{R}} : D_{\mathsf{R}}} \qquad \Psi \,;\, \Gamma \,;\, \Delta \vDash^{q} P :: (x_{\mathsf{S}} : A)}{\Psi \Vdash^{q} \{x_{\mathsf{S}} \leftarrow P \leftarrow \overline{a_{\mathsf{S}}} \,;\, \overline{d_{\mathsf{R}}}\} : \{A \leftarrow \overline{A_{\mathsf{S}}} \,;\, \overline{D_{\mathsf{R}}}\}_{\mathsf{S}}} \ \{\}I_{\mathsf{S}}$$

$$\frac{\Gamma = \overline{a_{\mathsf{S}} : A_{\mathsf{S}}} \qquad \Delta = \overline{d : D} \qquad \Psi \,;\, \Gamma \,;\, \Delta \vDash^{q} P :: (x_{\mathsf{T}} : A)}{\Psi \Vdash^{q} \{x_{\mathsf{T}} \leftarrow P \leftarrow \overline{a_{\mathsf{S}}} \,;\, \overline{d}\} : \{A \leftarrow \overline{A_{\mathsf{S}}} \,;\, \overline{D}\}_{\mathsf{T}}} \ \{\}I_{\mathsf{T}}$$

Elimination Rules.

$$\frac{\begin{array}{c} r = p + q \qquad \Delta = \overline{d_{\mathsf{R}} : D_{\mathsf{R}}} \qquad \Psi \curlyvee (\Psi_1, \Psi_2) \\ \Psi_1 \Vdash^{p} M : \{A \leftarrow \overline{D_{\mathsf{R}}}\}_{\mathsf{R}} \qquad \Psi_2 \,;\, \Gamma \,;\, \Delta', (x_{\mathsf{R}} : A) \vDash^{q} Q :: (z_m : C) \end{array}}{\Psi \,;\, \Gamma \,;\, \Delta, \Delta' \vDash^{r} x_{\mathsf{R}} \leftarrow M \leftarrow \overline{d_{\mathsf{R}}} \,;\, Q :: (z_m : C)} \ \{\}E_{\mathsf{R}m(=\mathsf{R},\mathsf{S},\mathsf{L},\mathsf{T})}$$

$$\frac{\begin{array}{c} r = p + q \qquad \Gamma \supseteq \overline{a_{\mathsf{S}} : A_{\mathsf{S}}} \qquad \Delta = \overline{d_{\mathsf{R}} : D_{\mathsf{R}}} \qquad (A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Psi \curlyvee (\Psi_1, \Psi_2) \\ \Psi_1 \Vdash^{p} M : \{A \leftarrow \overline{A_{\mathsf{S}}} \,;\, \overline{D_{\mathsf{R}}}\}_{\mathsf{S}} \qquad \Psi_2 \,;\, \Gamma, (x_{\mathsf{S}} : A) \,;\, \Delta' \vDash^{q} Q :: (z_m : C) \end{array}}{\Psi \,;\, \Gamma \,;\, \Delta, \Delta' \vDash^{r} x_{\mathsf{S}} \leftarrow M \leftarrow \overline{d_{\mathsf{R}}} \,;\, Q :: (z_m : C)} \ \{\}E_{\mathsf{S}m(=\mathsf{S},\mathsf{L},\mathsf{T})}$$

$$\frac{r = p + q \qquad \Gamma \supseteq \overline{a_{\mathsf{S}} : A_{\mathsf{S}}} \qquad \Delta = \overline{d : D} \qquad \Psi \curlyvee (\Psi_1, \Psi_2) \quad \Psi_1 \Vdash^{p} M : \{A \leftarrow \overline{A_{\mathsf{S}}} \ ; \ \overline{D}\}_{\mathsf{T}} \qquad \Psi_2 \ ; \ \Gamma \ ; \ (x_{\mathsf{T}} : A), \Delta' \vDash^{q} Q :: (z_m : C)}{\Psi \ ; \ \Gamma \ ; \ \Delta, \Delta' \vDash^{r} x_{\mathsf{T}} \leftarrow M \leftarrow \overline{a_{\mathsf{S}}} \ ; \ \overline{d} \ ; \ Q :: (z_m : C)} \ \{\}E_{\mathsf{T}m(=\mathsf{L},\mathsf{T})}$$

The rest of the rules for expressions in the functional layer are standard. We skip them and discuss the process layer.

3.2 Forwarding

$$\frac{q = 0}{\Psi \ ; \ \Gamma \ ; \ (y_m : A) \vDash^{q} x_m \leftarrow y_m :: (x_m : A)} \ \mathsf{fwd}_{m(=P,\mathsf{T})}$$

3.3 Labels and Branching

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{q} P :: (x_m : A_k) \qquad (k \in L)}{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{q} x_m.k \ ; \ P :: (x_m : \oplus\{\ell : A_\ell\}_{\ell \in L})} \ \oplus R$$

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : A_\ell) \vDash^{q} Q_\ell :: (z_k : C) \qquad (\forall \ell \in L)}{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : \oplus\{\ell : A_\ell\}_{\ell \in L}) \vDash^{q} \mathsf{case}\ x_m\ (\ell \Rightarrow Q_\ell)_{\ell \in L} :: (z_k : C)} \ \oplus L$$

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{q} P :: (x_m : A_\ell) \qquad (\forall \ell \in L)}{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{q} \mathsf{case}\ x_m\ (\ell \Rightarrow P_\ell)_{\ell \in L} :: (x_m : \&\{\ell : A_\ell\}_{\ell \in L})} \ \& R$$

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : A_\ell) \vDash^{q} Q_\ell :: (z_k : C) \qquad (k \in L)}{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : \&\{\ell : A_\ell\}_{\ell \in L}) \vDash^{q} x_m.k \ ; \ P :: (z_k : C)} \ \& L$$

3.4 Linear Channel Communication

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{q} P :: (x_m : B)}{\Psi \ ; \ \Gamma \ ; \ \Delta, (w_n : A) \vDash^{q} \mathsf{send}\ x_m\ w_n \ ; \ P :: (x_m : A \otimes_n B)} \ \otimes_n R$$

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta, (y_n : A), (x_m : B) \vDash^{q} Q :: (z_k : C)}{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : A \otimes_n B) \vDash^{q} y_n \leftarrow \mathsf{recv}\ x_m \ ; \ Q :: (z_k : C)} \ \otimes_n L$$

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta, (y_n : A) \vDash^{q} P :: (x_m : B)}{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{q} y_n \leftarrow \mathsf{recv}\ x_m \ ; \ P :: (x_m : A \multimap B)} \ \multimap_n R$$

$$\frac{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : B) \vDash^{q} Q :: (z_k : C)}{\Psi \ ; \ \Gamma \ ; \ \Delta, (w_n : A), (x_m : A \multimap B) \vDash^{q} \mathsf{send}\ x_m\ w_n \ ; \ Q :: (z_k : C)} \ \multimap_n L$$

3.5 Value Communication

$$\frac{r = p + q \qquad \Psi \curlyvee (\Psi_1, \Psi_2) \qquad \Psi_1 \Vdash^{p} M : \tau \qquad \Psi_2 \ ; \ \Gamma \ ; \ \Delta \vDash^{q} P :: (x_m : A)}{\Psi \ ; \ \Gamma \ ; \ \Delta \vDash^{r} \mathsf{send}\ x_m\ M \ ; \ P :: (x_m : \tau \times A)} \ \times R$$

$$\frac{\Psi, (y : \tau) \ ; \ \Gamma \ ; \ \Delta, (x_m : A) \vDash^{q} Q :: (z_k : C)}{\Psi \ ; \ \Gamma \ ; \ \Delta, (x_m : \tau \times A) \vDash^{q} y \leftarrow \mathsf{recv}\ x_m \ ; \ Q :: (z_k : C)} \ \times L$$

$$\frac{\Psi, (y : \tau) \;;\; \Gamma \;;\; \Delta \vdash^{q} P :: (x_m : B)}{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} y \leftarrow \mathsf{recv}\; x_m \;;\; P :: (x_m : \tau \to A)} \to R$$

$$\frac{r = p + q \qquad \Psi \curlyvee (\Psi_1, \Psi_2) \qquad \Psi_1 \Vdash^{p} M : \tau \qquad \Psi_2 \;;\; \Gamma \;;\; \Delta, (x_m : A) \vdash^{q} Q :: (z_k : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (x_m : \tau \to A) \vdash^{r} \mathsf{send}\; x_m\; M \;;\; Q :: (z_k : C)} \to L$$

3.6 Termination

$$\frac{q = 0}{\Psi \;;\; \Gamma \;;\; \cdot \vdash^{q} \mathsf{close}\; x_m :: (x_m : \mathbf{1})} \mathbf{1}R \qquad \frac{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} Q :: (z_k : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (x_m : \mathbf{1}) \vdash^{q} \mathsf{wait}\; x_m \;;\; Q :: (z_k : C)} \mathbf{1}L$$

3.7 Potential

$$\frac{q = p + r \qquad \Psi \;;\; \Gamma \;;\; \Delta \vdash^{p} P :: (x_m : A)}{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} \mathsf{tick}\; (r) \;;\; P :: (x_m : A)} \;\mathsf{work}$$

$$\frac{q = p + r \qquad \Psi \;;\; \Gamma \;;\; \Delta \vdash^{p} P :: (x_m : A)}{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} \mathsf{pay}\; x_m\; \{r\} \;;\; P :: (x_m : \triangleright^{r} A)} \triangleright R \qquad \frac{p = q + r \qquad \Psi \;;\; \Gamma \;;\; \Delta, (x_m : A) \vdash^{p} P :: (z_k : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (x_m : \triangleright^{r} A) \vdash^{q} \mathsf{get}\; x_m\; \{r\} \;;\; P :: (z_k : C)} \triangleright L$$

$$\frac{p = q + r \qquad \Psi \;;\; \Gamma \;;\; \Delta \vdash^{p} P :: (x_m : A)}{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} \mathsf{get}\; x_m\; \{r\} \;;\; P :: (x_m : \triangleleft^{r} A)} \triangleleft R \qquad \frac{q = p + r \qquad \Psi \;;\; \Gamma \;;\; \Delta, (x_m : A) \vdash^{p} P :: (z_k : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (x_m : \triangleleft^{r} A) \vdash^{q} \mathsf{pay}\; x_m\; \{r\} \;;\; P :: (z_k : C)} \triangleleft L$$

3.8 Acquiring and Releasing

$$\frac{\Delta\; \mathsf{purelin} \qquad \Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} x_{\mathsf{L}} \leftarrow \mathsf{accept}\; x_{\mathsf{S}} \;;\; P :: (x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})} \uparrow_{\mathsf{L}}^{\mathsf{S}} R$$

$$\frac{\Psi \;;\; \Gamma \;;\; \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}) \vdash^{q} Q :: (z_m : C)}{\Psi \;;\; \Gamma, (x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \;;\; \Delta \vdash^{q} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\; x_{\mathsf{S}} \;;\; Q :: (z_m : C)} \uparrow_{\mathsf{L}}^{\mathsf{S}} L_{m(=\mathsf{L},\mathsf{T})}$$

$$\frac{\Delta\; \mathsf{purelin} \qquad \Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\Psi \;;\; \Gamma \;;\; \Delta \vdash^{q} x_{\mathsf{S}} \leftarrow \mathsf{detach}\; x_{\mathsf{L}} \;;\; P :: (x_{\mathsf{L}} : \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}})} \downarrow_{\mathsf{L}}^{\mathsf{S}} R$$

$$\frac{\Psi \;;\; \Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}) \;;\; \Delta \vdash^{q} Q :: (z_m : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (x_{\mathsf{L}} : \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}}) \vdash^{q} x_{\mathsf{S}} \leftarrow \mathsf{release}\; x_{\mathsf{L}} \;;\; Q :: (z_m : C)} \downarrow_{\mathsf{L}}^{\mathsf{S}} L_{m(=\mathsf{L},\mathsf{T})}$$

4 OPERATIONAL COST SEMANTICS

First, we define the judgments for expressions. The first judgment is a small step semantics for expressions, $M \mapsto M'$ and M val. Finally, we introduce another judgment for processes, $\mathsf{proc}(c_m, w, P) \mapsto \mathsf{proc}(c'_m, w', P')$ and a new predicate $\mathsf{msg}(c_m, w, M)$ to denote a message. Additionally, we define processes with a hole for a compact representation of the cost semantics.

$$P[\cdot] \quad ::= \quad c \leftarrow [\cdot] \leftarrow a_i \ ; \ P_c \quad | \quad \mathsf{send}\ c\ [\cdot]\ ;\ P$$

$$\frac{N \Downarrow V \mid \mu}{\mathsf{proc}(c_m, w, P[N]) \mapsto \mathsf{proc}(c_m, w + \mu, P[V])} \ \mathsf{internal}$$

$$\frac{(c_{\mathsf{R}} \text{ fresh})}{\mathsf{proc}(d_m, w, x_{\mathsf{R}} \leftarrow \{x'_{\mathsf{R}} \leftarrow P_{x'_{\mathsf{R}},\overline{y}} \leftarrow \overline{y}\} \leftarrow \overline{a}\ ;\ Q) \mapsto \mathsf{proc}(c_{\mathsf{R}}, 0, P_{c_{\mathsf{R}},\overline{a}}) \quad \mathsf{proc}(d_m, w, [c_{\mathsf{R}}/x_{\mathsf{R}}]Q)} \ \{\}E_{\mathsf{R}m}$$

$$\frac{(c_{\mathsf{S}} \text{ fresh})}{\begin{array}{c}\mathsf{proc}(d_m, w, x_{\mathsf{S}} \leftarrow \{x'_{\mathsf{S}} \leftarrow P_{x'_{\mathsf{S}},\overline{y},\overline{z}} \leftarrow \overline{y}\ ;\ \overline{z}\} \leftarrow \overline{a}\ ;\ \overline{b}\ ;\ Q) \mapsto \\ \mathsf{proc}(c_{\mathsf{S}}, 0, P_{c_{\mathsf{S}},\overline{a},\overline{b}}) \quad \mathsf{proc}(d_m, w, [c_{\mathsf{S}}/x_{\mathsf{S}}]Q)\end{array}} \ \{\}E_{\mathsf{S}m}$$

$$\frac{(c_{\mathsf{T}} \text{ fresh})}{\begin{array}{c}\mathsf{proc}(d_{\mathsf{T}}, w, x_{\mathsf{T}} \leftarrow \{x'_{\mathsf{T}} \leftarrow P_{x'_{\mathsf{T}},\overline{y},\overline{z}} \leftarrow \overline{y}\ ;\ \overline{z}\} \leftarrow \overline{a}\ ;\ \overline{b}\ ;\ Q) \mapsto \\ \mathsf{proc}(c_{\mathsf{T}}, 0, P_{c_{\mathsf{T}},\overline{a},\overline{b}}) \quad \mathsf{proc}(d_{\mathsf{T}}, w, [c_{\mathsf{T}}/x_{\mathsf{L}}]Q)\end{array}} \ \{\}E_{\mathsf{TT}}$$

$$\frac{}{\mathsf{msg}(d_m, w', M) \quad \mathsf{proc}(c_m, w, c_m \leftarrow d_m) \mapsto \mathsf{msg}(c_m, w + w', [c_m/d_m]M)} \ \mathsf{fwd}^+$$

$$\frac{}{\mathsf{proc}(c_m, w, c_m \leftarrow d_m) \quad \mathsf{msg}(e_l, w', M(c_m)) \mapsto \mathsf{msg}(e_l, w + w', M(d_m))} \ \mathsf{fwd}^-$$

$$\frac{(c_m^+ \text{ fresh})}{\mathsf{proc}(c_m, w, c_m.\ell\ ;\ P) \mapsto \mathsf{proc}(c_m^+, w, [c_m^+/c_m]P) \quad \mathsf{msg}(c_m, 0, c_m.\ell\ ;\ c_m \leftarrow c_m^+)} \ \oplus C_s$$

$$\frac{}{\mathsf{msg}(c_m, w, c_m.\ell\ ;\ c_m \leftarrow c_m^+) \quad \mathsf{proc}(d_k, w', \mathsf{case}\ c_m\ (l \Rightarrow Q_l)_{l \in L}) \mapsto \mathsf{proc}(d_k, w + w', [c_m^+/c_m]Q_\ell)} \ \oplus C_r$$

$$\frac{(c_m^+ \text{ fresh})}{\mathsf{proc}(d_k, w, c_m.\ell\ ;\ P) \mapsto \mathsf{msg}(c_m^+, 0, c_m.\ell\ ;\ c_m^+ \leftarrow c_m) \quad \mathsf{proc}(d_k, w, [c_m^+/c_m]P)} \ \& C_s$$

$$\frac{}{\mathsf{proc}(c_m, w', \mathsf{case}\ c_m\ (l \Rightarrow Q_l)_{l \in L}) \quad \mathsf{msg}(c_m^+, 0, c_m.\ell\ ;\ c_m^+ \leftarrow c_m) \mapsto \mathsf{proc}(c_m^+, w + w', [c_m^+/c_m]Q_\ell)} \ \& C_r$$

$$\frac{(c_m^+ \text{ fresh})}{\mathsf{proc}(c_m, w, \mathsf{send}\ c_m\ e_n\ ;\ P) \mapsto \mathsf{proc}(c_m^+, w, [c_m^+/c_m]P) \quad \mathsf{msg}(c_m, 0, \mathsf{send}\ c_m\ e_n\ ;\ c_m \leftarrow c_m^+)} \ \otimes_n C_s$$

$$\frac{}{\begin{array}{c}\mathsf{msg}(c_m, w, \mathsf{send}\ c_m\ e_n\ ;\ c_m \leftarrow c_m^+) \quad \mathsf{proc}(d_k, w', x_n \leftarrow \mathsf{recv}\ c_m\ ;\ Q) \mapsto \\ \mathsf{proc}(d_k, w + w', [c_m^+/c_m][e_n/x_n]Q)\end{array}} \ \otimes_n C_r$$

$$\frac{(c_m^+ \text{ fresh})}{\text{proc}(d_k, w, \text{send } c_m\ e_n\ ;\ P) \mapsto \text{msg}(c_m^+, 0, \text{send } c_m\ e_n\ ;\ c_m^+ \leftarrow c_m) \quad \text{proc}(d_k, w, [c_m^+/c_m]P)} \multimap_n C_s$$

$$\frac{}{\text{proc}(c_m, w', x_n \leftarrow \text{recv } c_m\ ;\ Q) \quad \text{msg}(c_m^+, w, \text{send } c_m\ e_n\ ;\ c_m^+ \leftarrow c_m) \mapsto \text{proc}(c_m^+, w + w', [c_m^+/c_m][e_n/x_n]Q)} \multimap_n C_r$$

$$\frac{(c_m^+ \text{ fresh}) \qquad N \text{ val}}{\text{proc}(c_m, w, \text{send } c_m\ N\ ;\ P) \mapsto \text{proc}(c_m^+, w, [c_m^+/c_m]P) \quad \text{msg}(c_m, 0, \text{send } c_m\ N\ ;\ c_m \leftarrow c_m^+)} \times C_s$$

$$\frac{}{\text{msg}(c_m, w, \text{send } c_m\ N\ ;\ c_m \leftarrow c_m^+) \quad \text{proc}(d_k, w', x \leftarrow \text{recv } c_m\ ;\ Q) \mapsto \text{proc}(d_k, w + w', [c_m^+/c_m][N/x]Q)} \times C_r$$

$$\frac{(c_m^+ \text{ fresh}) \qquad N \text{ val}}{\text{proc}(d_k, w, \text{send } c_m\ N\ ;\ P) \mapsto \text{msg}(c_m^+, 0, \text{send } c_m\ N\ ;\ c_m^+ \leftarrow c_m) \quad \text{proc}(d_k, w, [c_m^+/c_m]P)} \rightarrow C_s$$

$$\frac{}{\text{proc}(c_m, w', x \leftarrow \text{recv } c_m\ ;\ Q) \quad \text{msg}(c_m^+, w, \text{send } c_m\ N\ ;\ c_m^+ \leftarrow c_m) \mapsto \text{proc}(c_m^+, w + w', [c_m^+/c_m][N/x]Q)} \rightarrow C_r$$

$$\frac{}{\text{proc}(c_m, w, \text{close } c_m) \mapsto \text{msg}(c_m, w, \text{close } c_m)} \mathbf{1} C_s$$

$$\frac{}{\text{msg}(c_m, w, \text{close } c_m) \quad \text{proc}(d_k, w', \text{wait } c_m\ ;\ Q) \mapsto \text{proc}(d_k, w + w', Q)} \mathbf{1} C_r$$

$$\frac{\text{proc}(c_m, w, \text{tick } (\mu)\ ;\ P)}{\text{proc}(c_m, w + \mu, P)} \text{tick}$$

$$\frac{(c_m^+ \text{ fresh})}{\text{proc}(c_m, w, \text{pay } c_m\ \{r\}\ ;\ P) \mapsto \text{proc}(c_m^+, w, [c_m^+/c_m]P) \quad \text{msg}(c_m, 0, \text{pay } c_m\ \{r\}\ ;\ c_m \leftarrow c_m^+)} \triangleright C_s$$

$$\frac{}{\text{msg}(c_m, w, \text{pay } c_m\ \{r\}\ ;\ c_m \leftarrow c_m^+) \quad \text{proc}(d_k, w', \text{get } c_m\ \{r\}\ ;\ Q) \mapsto \text{proc}(d_k, w + w', [c_m^+/c_m]Q)} \triangleright C_r$$

$$\frac{(c_m^+ \text{ fresh})}{\text{proc}(d_k, w, \text{pay } c_m\ \{r\}\ ;\ P) \mapsto \text{msg}(c_m^+, 0, \text{pay } c_m\ \{r\}\ ;\ c_m^+ \leftarrow c) \quad \text{proc}(d_k, w, [c_m^+/c_m]P)} \triangleleft C_s$$

$$\frac{}{\text{proc}(c_m, w', \text{get } c_m\ \{r\}\ ;\ Q) \quad \text{msg}(c_m^+, w, \text{pay } c_m\ \{r\}\ ;\ c_m^+ \leftarrow c_m) \mapsto \text{proc}(c_m, w + w', [c_m^+/c_m]Q)} \triangleleft C_r$$

$$\dfrac{(a_{\mathsf{L}}\ \text{fresh})}{\mathsf{proc}(a_{\mathsf{S}}, w', x_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}}\ ;\ P_{x_{\mathsf{L}}})\quad \mathsf{proc}(c_m, w, x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ;\ Q_{x_{\mathsf{L}}}) \mapsto \atop \mathsf{proc}(a_{\mathsf{L}}, w', P_{a_{\mathsf{L}}})\quad \mathsf{proc}(c_m, w, Q_{a_{\mathsf{L}}})} \uparrow^{\mathsf{S}}_{\mathsf{L}} C$$

$$\dfrac{}{\mathsf{proc}(a_{\mathsf{L}}, w', x_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}}\ ;\ P_{x_{\mathsf{S}}})\quad \mathsf{proc}(c_m, w, x_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}}\ ;\ Q_{x_{\mathsf{S}}}) \mapsto \atop \mathsf{proc}(a_{\mathsf{S}}, w', P_{a_{\mathsf{S}}})\quad \mathsf{proc}(c_m, w, Q_{a_{\mathsf{S}}})} \downarrow^{\mathsf{S}}_{\mathsf{L}} C$$

5 CONFIGURATION TYPING

$$\dfrac{}{\Gamma_0 \overset{0}{\vDash} (\cdot) :: (\cdot\ ;\ \cdot)}\ \mathsf{emp}$$

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma\ ;\ \Delta, \Delta'_{\mathsf{R}}) \qquad \cdot\ ;\ \cdot\ ;\ \Delta'_{\mathsf{R}} \overset{q}{\vdash} P :: (x_{\mathsf{R}} : A_{\mathsf{R}})}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{proc}(x_{\mathsf{R}}, w, P) :: (\Gamma\ ;\ \Delta, (x_{\mathsf{R}} : A_{\mathsf{R}}))}\ \mathsf{proc}_{\mathsf{R}}$$

$$\dfrac{(x_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0 \qquad (A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma\ ;\ \Delta, \Delta'_{\mathsf{R}}) \qquad \cdot\ ;\ \Gamma_0\ ;\ \Delta'_{\mathsf{R}} \overset{q}{\vdash} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{proc}(x_{\mathsf{S}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}})\ ;\ \Delta)}\ \mathsf{proc}_{\mathsf{S}}$$

$$\dfrac{(x_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0 \qquad (A_{\mathsf{L}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma\ ;\ \Delta, \Delta') \qquad \cdot\ ;\ \Gamma_0\ ;\ \Delta' \overset{q}{\vdash} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{proc}(x_{\mathsf{L}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}})\ ;\ \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}))}\ \mathsf{proc}_{\mathsf{L}}$$

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma\ ;\ \Delta, \Delta') \qquad \cdot\ ;\ \Gamma_0\ ;\ \Delta' \overset{q}{\vdash} P :: (x_{\mathsf{T}} : A_{\mathsf{T}})}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{proc}(x_{\mathsf{T}}, w, P) :: (\Gamma\ ;\ \Delta, (x_{\mathsf{T}} : A_{\mathsf{T}}))}\ \mathsf{proc}_{\mathsf{T}}$$

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma\ ;\ \Delta, \Delta') \qquad \cdot\ ;\ \cdot\ ;\ \Delta' \overset{q}{\vdash} M :: (x_m : A)}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{msg}(x_m, w, M) :: (\Gamma\ ;\ \Delta, (x_m : A))}\ \mathsf{msg}$$

In addition, for a well-typed configuration $\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma\ ;\ \Delta)$, we need the following wellformedness conditions.

- All channels in Γ_0, Γ and Δ are unique.
- $\Gamma \subseteq \Gamma_0$.

5.1 Equi-Synchronizing

$$\frac{(A_\ell, C_\mathsf{S})\ \mathsf{esync}\quad (\forall \ell \in L)}{(\oplus\{\ell : A_\ell\}_{\ell\in L}, C_\mathsf{S})\ \mathsf{esync}}\ \oplus \qquad \frac{(A_\ell, C_\mathsf{S})\ \mathsf{esync}\quad (\forall \ell \in L)}{(\&\{\ell : A_\ell\}_{\ell\in L}, C_\mathsf{S})\ \mathsf{esync}}\ \&$$

$$\frac{(B, C_\mathsf{S})\ \mathsf{esync}}{(A \otimes B, C_\mathsf{S})\ \mathsf{esync}}\ \otimes \qquad \frac{(B, C_\mathsf{S})\ \mathsf{esync}}{(A \multimap B, C_\mathsf{S})\ \mathsf{esync}}\ \multimap$$

$$\frac{(B, C_\mathsf{S})\ \mathsf{esync}}{(\tau \times B, C_\mathsf{S})\ \mathsf{esync}}\ \times \qquad \frac{(B, C_\mathsf{S})\ \mathsf{esync}}{(\tau \rightarrow B, C_\mathsf{S})\ \mathsf{esync}}\ \rightarrow$$

$$\frac{(A, C_\mathsf{S})\ \mathsf{esync}}{(\triangleright^r A, C_\mathsf{S})\ \mathsf{esync}}\ \triangleright \qquad \frac{(A, C_\mathsf{S})\ \mathsf{esync}}{(\triangleleft^r A, C_\mathsf{S})\ \mathsf{esync}}\ \triangleleft$$

$$\frac{(A_\mathsf{L}, \uparrow_\mathsf{L}^\mathsf{S} A_\mathsf{L})\ \mathsf{esync}}{(\uparrow_\mathsf{L}^\mathsf{S} A_\mathsf{L}, \uparrow_\mathsf{L}^\mathsf{S} A_\mathsf{L})\ \mathsf{esync}}\ \uparrow_\mathsf{L}^\mathsf{S} \qquad \frac{(A_\mathsf{S}, A_\mathsf{S})\ \mathsf{esync}}{(\downarrow_\mathsf{L}^\mathsf{S} A_\mathsf{S}, A_\mathsf{S})\ \mathsf{esync}}\ \downarrow_\mathsf{L}^\mathsf{S}$$

5.2 Purely Linear Context

$$\frac{}{\cdot\ \mathsf{purelin}}\ \mathsf{emp} \qquad \frac{x_\mathsf{R} : A_\mathsf{R} \qquad \Delta\ \mathsf{purelin}}{x_\mathsf{R} : A_\mathsf{R}, \Delta\ \mathsf{purelin}}\ \mathsf{step}$$

6 TYPE SAFETY

Lemma 1 (Renaming). *The following renamings are allowed.*

- *If* $\Psi\ ;\ \Gamma, (x_\mathsf{S} : A_\mathsf{S})\ ;\ \Delta \Vdash^{g} P_{x_\mathsf{S}} :: (z_k : C)$ *is well-typed, so is* $\Gamma, (c_\mathsf{S} : A_\mathsf{S})\ ;\ \Delta \Vdash^{g} P_{c_\mathsf{S}} :: (z_k : C)$.
- *If* $\Psi\ ;\ \Gamma\ ;\ \Delta, (x_m : A) \Vdash^{g} P_{x_m} :: (z_k : C)$ *is well-typed, so is* $\Gamma\ ;\ \Delta, (c_m : A) \Vdash^{g} P_{c_m} :: (z_k : C)$.
- *If* $\Psi\ ;\ \Gamma\ ;\ \Delta \Vdash^{g} P_{z_k} :: (z_k : C)$ *is well-typed, so is* $\Gamma\ ;\ \Delta \Vdash^{g} P_{c_k} :: (c_k : C)$.

Lemma 2 (Invariants). *The process typing judgment* $\Psi\ ;\ \Gamma\ ;\ \Delta \Vdash^{g} P :: (x_m : A)$ *preserves the following invariants.*

(R) $\Psi\ ;\ \cdot\ ;\ \Delta_\mathsf{R} \Vdash^{g} P :: (x_\mathsf{R} : A_\mathsf{R})$

(S/L) $\Psi\ ;\ \Gamma\ ;\ \Delta_\mathsf{R} \Vdash^{g} P :: (x_\mathsf{S} : A_\mathsf{S})$ *or* $\Psi\ ;\ \Gamma\ ;\ \Delta \Vdash^{g} P :: (x_\mathsf{L} : A_\mathsf{L})$

(T) $\Psi\ ;\ \Gamma\ ;\ \Delta \Vdash^{g} P :: (x_\mathsf{T} : A_\mathsf{T})$

Proof. The elimination rules preserve the invariant trivially because they can only be applied when the invariant is maintained and the premise in each rule maintains the same invariant.

- Case (E_RR) : This rule can only be applied when the context is purely linear. And then adding x_R to the context will keep it purely linear.
- Case ($E_\mathsf{RS}, E_\mathsf{RL}$) : This rule can only be applied if offering channel is either in S or L mode and the context is purely linear. Hence, adding x_R to the context is allowed.
- Case (E_RT) : The context is mixed linear, hence adding a purely linear channel is valid.

- Case (E_{SS}, E_{SL}, E_{ST}) : The context has shared channels in each case, hence adding another shared channel is valid.
- Case (E_{TT}) : Adding a client linear channel to a mixed context is valid.
- Case (fwd) :

(R) : $\Delta_R = (y_R : A_R)$ which is valid since Δ_R is purely linear and there are no premises.

(S/L) : This rule cannot be applied since the fwd rule applies only when the offering mode is R. Hence, there is a mode mismatch.

(T) : Analogous to (S/L).

- Case ($\oplus R$) :

(R) :

$$\frac{\Psi \,;\, \cdot \,;\, \Delta_R \vdash^{g} P :: (x_R : A_k) \qquad (k \in L)}{\Psi \,;\, \cdot \,;\, \Delta_R \vdash^{g} (x_R.k \,;\, P) :: (x_R : \oplus\{\ell : A_\ell\}_{\ell \in L})} \oplus R$$

The context doesn't change, and the type of the offered channel remains purely linear.

(S/L) :

$$\frac{\Psi \,;\, \Gamma \,;\, \Delta \vdash^{g} P :: (x_L : A_k) \qquad (k \in L)}{\Psi \,;\, \Gamma \,;\, \Delta \vdash^{g} (x_L.k \,;\, P) :: (x_L : \oplus\{\ell : A_\ell\}_{\ell \in L})} \oplus R$$

The context doesn't change, and the type of the offered channel remains shared linear. Also, the mode of x cannot be S because the type doesn't allow that.

(T) :

$$\frac{\Psi \,;\, \Gamma \,;\, \Delta \vdash^{g} P :: (x_T : A_k) \qquad (k \in L)}{\Psi \,;\, \Gamma \,;\, \Delta \vdash^{g} (x_T.k \,;\, P) :: (x_T : \oplus\{\ell : A_\ell\}_{\ell \in L})} \oplus R$$

The context doesn't change, and the type of the offered channel remains client linear.

- Case ($\oplus L$) :

(R) :

$$\frac{\Psi \,;\, \cdot \,;\, \Delta_R, (x_R : A_\ell) \vdash^{g} Q_\ell :: (z_R : C) \qquad (\forall \ell \in L)}{\Psi \,;\, \cdot \,;\, \Delta_R, (x_R : \oplus\{\ell : A_\ell\}_{\ell \in L}) \vdash^{g} \text{case } x_R\ (\ell \Rightarrow Q_\ell)_{\ell \in L} :: (z_R : C)} \oplus L$$

The context remains purely linear, and the offered channel doesn't change.

(S/L) :

$$\frac{\Psi \,;\, \Gamma \,;\, \Delta, (x_m : A_\ell) \vdash^{g} Q_\ell :: (z_k : C) \qquad (\forall \ell \in L)}{\Psi \,;\, \Gamma \,;\, \Delta, (x_m : \oplus\{\ell : A_\ell\}_{\ell \in L}) \vdash^{g} \text{case } x_m\ (\ell \Rightarrow Q_\ell)_{\ell \in L} :: (z_k : C)} \oplus L$$

The mode of x_m doesn't change, and the offered channel doesn't change.

(T) :

$$\frac{\Psi \,;\, \Gamma \,;\, \Delta, (x_m : A_\ell) \vdash^{g} Q_\ell :: (z_T : C) \qquad (\forall \ell \in L)}{\Psi \,;\, \Gamma \,;\, \Delta, (x_m : \oplus\{\ell : A_\ell\}_{\ell \in L}) \vdash^{g} \text{case } x_m\ (\ell \Rightarrow Q_\ell)_{\ell \in L} :: (z_T : C)} \oplus L$$

The mode of the channel x_m doesn't change, and the offered channel doesn't change.

- Case ($\multimap_n R$) :

(R) :

$$\frac{\Psi \;;\; \cdot \;;\; \Delta_{\mathsf{R}}, (y_{\mathsf{R}} : A) \Vdash^{g} P :: (x_{\mathsf{R}} : B)}{\Psi \;;\; \cdot \;;\; \Delta_{\mathsf{R}} \Vdash^{g} y_{\mathsf{R}} \leftarrow \mathsf{recv}\ x_{\mathsf{R}} \;;\; P :: (x_{\mathsf{R}} : A \multimap_{\mathsf{R}} B)} \multimap_{\mathsf{R}} R$$

A process offering a purely linear channel only allows exchanging purely linear channels. This channel gets added to the purely linear context, and the type of the offered channel remains purely linear.

(S/L) :

$$\frac{\Psi \;;\; \Gamma \;;\; \Delta, (y_n : A) \Vdash^{g} P :: (x_{\mathsf{L}} : B)}{\Psi \;;\; \Gamma \;;\; \Delta \Vdash^{g} y_n \leftarrow \mathsf{recv}\ x_{\mathsf{L}} \;;\; P :: (x_{\mathsf{L}} : A \multimap_n B)} \multimap_n R$$

A linear channel gets added to the mixed linear context, and the type of the offered channel remains shared linear. Also, the mode of x cannot be S because the type doesn't allow that.

(T) :

$$\frac{\Psi \;;\; \Gamma \;;\; \Delta, (y_n : A) \Vdash^{g} P :: (x_{\mathsf{T}} : B)}{\Psi \;;\; \Gamma \;;\; \Delta \Vdash^{g} y_n \leftarrow \mathsf{recv}\ x_{\mathsf{T}} \;;\; P :: (x_{\mathsf{T}} : A \multimap_n B)} \multimap_n R$$

A linear channel gets added to the mixed linear context, and the type of the offered channel remains client linear.

- Case ($\multimap_n L$) :

(R) :

$$\frac{\Psi \;;\; \cdot \;;\; \Delta_{\mathsf{R}}, (x_{\mathsf{R}} : B) \Vdash^{g} Q :: (z_{\mathsf{R}} : C)}{\Psi \;;\; \cdot \;;\; \Delta_{\mathsf{R}}, (w_{\mathsf{R}} : A), (x_{\mathsf{R}} : A \multimap_{\mathsf{R}} B) \Vdash^{g} \mathsf{send}\ x_{\mathsf{R}}\ w_{\mathsf{R}} \;;\; Q :: (z_{\mathsf{R}} : C)} \multimap_{\mathsf{R}} L$$

A purely linear channel is allowed in a purely linear context. The context remains purely linear, and the offered channel doesn't change.

(S/L) :

$$\frac{\Psi \;;\; \Gamma \;;\; \Delta, (x_m : B) \Vdash^{g} Q :: (z_k : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (w_n : A), (x_m : A \multimap_n B) \Vdash^{g} \mathsf{send}\ x_m\ w_n \;;\; Q :: (z_k : C)} \multimap_n L$$

A linear channel is allowed in a mixed linear context. The mode of the channel x_m doesn't change, and the offered channel doesn't change.

(T) :

$$\frac{\Psi \;;\; \Gamma \;;\; \Delta, (x_m : B) \Vdash^{g} Q :: (z_k : C)}{\Psi \;;\; \Gamma \;;\; \Delta, (w_n : A), (x_m : A \multimap_n B) \Vdash^{g} \mathsf{send}\ x_m\ w_n \;;\; Q :: (z_k : C)} \multimap_n L$$

A linear channel is allowed in a mixed linear context. The mode of the channel x_m doesn't change, and the offered channel doesn't change.

- Case ($\uparrow^{\mathsf{S}}_{\mathsf{L}} R$) :

(R) : This rule cannot be applied since the offered channel in this case should be purely linear, which is not the case for $\uparrow^{\mathsf{S}}_{\mathsf{L}} R$ rule.

(S/L) :

$$\frac{\Delta \text{ purelin} \qquad \Psi \,;\, \Gamma \,;\, \Delta \vDash^{g} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Psi \,;\, \Gamma \,;\, \Delta \vDash^{g} x_{\mathsf{L}} \leftarrow \text{accept } x_{\mathsf{S}} \,;\, P :: (x_{\mathsf{S}} : \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}})} \uparrow^{\mathsf{S}}_{\mathsf{L}} R$$

The context doesn't change and the offered channel switches its mode from S to L. Moreover, the rule cannot be applied if the offered channel is in L mode, since there will be a mode mismatch.

(T) : This rule cannot be applied since the offered channel should be in T mode, which doesn't match with S.

- Case ($\downarrow^{\mathsf{S}}_{\mathsf{L}} R$) : Analogous to $\uparrow^{\mathsf{S}}_{\mathsf{L}} R$.
- Case ($\uparrow^{\mathsf{S}}_{\mathsf{L}} L$) :

(R) : This rule cannot be applied since the context should be purely linear, which is not the case for $\uparrow^{\mathsf{S}}_{\mathsf{L}} L$ rule.

(S/L) :

$$\frac{\Psi \,;\, \Gamma \,;\, \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}) \vDash^{g} Q :: (z_{\mathsf{L}} : C)}{\Psi \,;\, \Gamma, (x_{\mathsf{S}} : \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \,;\, \Delta \vDash^{g} x_{\mathsf{L}} \leftarrow \text{acquire } x_{\mathsf{S}} \,;\, Q :: (z_{\mathsf{L}} : C)} \uparrow^{\mathsf{S}}_{\mathsf{L}} L_{\mathsf{L}}$$

A shared linear channel is allowed in a mixed linear context. The mode of the offering channel is unchanged. A shared channel is removed from the shared context, but the new context is still shared.

(T) :

$$\frac{\Psi \,;\, \Gamma \,;\, \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}) \vDash^{g} Q :: (z_{\mathsf{T}} : C)}{\Psi \,;\, \Gamma, (x_{\mathsf{S}} : \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \,;\, \Delta \vDash^{g} x_{\mathsf{L}} \leftarrow \text{acquire } x_{\mathsf{S}} \,;\, Q :: (z_{\mathsf{T}} : C)} \uparrow^{\mathsf{S}}_{\mathsf{L}} L$$

A shared linear channel gets added to the mixed linear context, which is allowed. A shared channel is removed from the shared context, but the new context is still shared. Moreover, the offered channel remains at the same mode.

- Case ($\downarrow^{\mathsf{S}}_{\mathsf{L}} L$) : Analogous to $\uparrow^{\mathsf{S}}_{\mathsf{L}} L$ rule.

□

Lemma 3 (Configuration Weakening). *If we have a well-typed configuration, $\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta)$, then for a shared channel $c_{\mathsf{S}} : B_{\mathsf{S}} \notin \Gamma_0$, we can weaken Γ_0 and get $\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta)$.*

Proof. We case analyze on the configuration typing judgment.

- Case (emp) : We have $\Gamma_0 \overset{0}{\vDash} (\cdot) :: (\cdot \,;\, \cdot)$. But, since there is no premise, we use the emp rule to get $\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \overset{0}{\vDash} (\cdot) :: (\cdot \,;\, \cdot)$.
- Case (proc_{R}) : We have $\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \text{proc}(x_{\mathsf{R}}, w, P) :: (\Gamma \,;\, \Delta, (x_{\mathsf{R}} : A_{\mathsf{R}}))$. Inverting the proc_{R} rule,

$$\frac{\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}}) \qquad \cdot \,;\, \cdot \,;\, \Delta'_{\mathsf{R}} \vDash^{g} P :: (x_{\mathsf{R}} : A_{\mathsf{R}})}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \text{proc}(x_{\mathsf{R}}, w, P) :: (\Gamma \,;\, \Delta, (x_{\mathsf{R}} : A_{\mathsf{R}}))} \text{proc}_{\mathsf{R}}$$

we get $\Gamma_0 \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}})$. By the induction hypothesis, $\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}})$. Applying the $\mathsf{proc}_{\mathsf{R}}$ rule,

$$\frac{\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}}) \qquad \cdot \,;\, \cdot \,;\, \Delta'_{\mathsf{R}} \vdash^{q} P :: (x_{\mathsf{R}} : A_{\mathsf{R}})}{\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{R}}, w, P) :: (\Gamma \,;\, \Delta, (x_{\mathsf{R}} : A_{\mathsf{R}}))}\ \mathsf{proc}_{\mathsf{R}}$$

- Case $(\mathsf{proc}_{\mathsf{S}})$: We have $\Gamma_0 \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{S}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}))$. Inverting the $\mathsf{proc}_{\mathsf{S}}$ rule,

$$\frac{(x_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0 \qquad (A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0 \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}}) \qquad \cdot \,;\, \Delta'_{\mathsf{R}} \vdash^{q} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\Gamma_0 \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{S}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}) \,;\, \Delta)}\ \mathsf{proc}_{\mathsf{S}}$$

we get $\Gamma_0 \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}})$. By the induction hypothesis, $\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}})$. Also, by Lemma 4, we get $\cdot \,;\, \Gamma_0, (c : B_{\mathsf{S}}) \,;\, \Delta'_{\mathsf{R}} \vdash^{q} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})$. Applying the $\mathsf{proc}_{\mathsf{S}}$ rule back,

$$\frac{\begin{array}{c}(x_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \qquad (A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta'_{\mathsf{R}}) \\ \cdot \,;\, \Gamma_0, (c : B_{\mathsf{S}}) \,;\, \Delta'_{\mathsf{R}} \vdash^{q} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})\end{array}}{\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{S}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}) \,;\, \Delta)}\ \mathsf{proc}_{\mathsf{S}}$$

- Case $(\mathsf{proc}_{\mathsf{L}})$: We have $\Gamma_0 \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{L}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}) \,;\, \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}))$. Inverting the $\mathsf{proc}_{\mathsf{L}}$ rule,

$$\frac{(x_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0 \qquad (A_{\mathsf{L}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0 \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta') \qquad \cdot \,;\, \Gamma_0 \,;\, \Delta' \vdash^{q} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Gamma_0 \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{L}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}) \,;\, \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}))}\ \mathsf{proc}_{\mathsf{L}}$$

we get $\Gamma_0 \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta')$. Applying the induction hypothesis, we get $\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta')$. Using Lemma 4, we get $\cdot \,;\, \Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \,;\, \Delta' \vdash^{q} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})$. Applying the $\mathsf{proc}_{\mathsf{L}}$ rule back,

$$\frac{\begin{array}{c}(x_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \qquad (A_{\mathsf{L}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta') \\ \cdot \,;\, \Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \,;\, \Delta' \vdash^{q} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})\end{array}}{\Gamma_0, (c_{\mathsf{S}} : B_{\mathsf{S}}) \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{L}}, w, P) :: (\Gamma, (x_{\mathsf{S}} : A_{\mathsf{S}}) \,;\, \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}))}\ \mathsf{proc}_{\mathsf{L}}$$

- Case $(\mathsf{proc}_{\mathsf{T}})$: We have $\Gamma_0 \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{T}}, w, P) :: (\Gamma \,;\, \Delta, (x_{\mathsf{T}} : A_{\mathsf{T}}))$. Inverting the $\mathsf{proc}_{\mathsf{T}}$ rule,

$$\frac{\Gamma_0 \stackrel{E}{\models} \Omega :: (\Gamma \,;\, \Delta, \Delta') \qquad \cdot \,;\, \Gamma_0 \,;\, \Delta' \vdash^{q} P :: (x_{\mathsf{T}} : A_{\mathsf{T}})}{\Gamma_0 \stackrel{E+q+w}{\models} \Omega, \mathsf{proc}(x_{\mathsf{T}}, w, P) :: (\Gamma \,;\, \Delta, (x_{\mathsf{T}} : A_{\mathsf{T}}))}\ \mathsf{proc}_{\mathsf{T}}$$

we get $\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta')$. By the induction hypothesis, we get $\Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta')$. Also, using Lemma 4, we get $\cdot \,;\, \Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \,;\, \Delta' \overset{q}{\vdash} P :: (x_\mathsf{T} : A_\mathsf{T})$ Applying the proc_T rule back,

$$\frac{\Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta') \qquad \cdot \,;\, \Gamma \,;\, \Delta' \overset{q}{\vdash} P :: (x_\mathsf{T} : A_\mathsf{T})}{\Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \overset{E+q+w}{\vDash} \Omega, \mathsf{proc}(x_\mathsf{T}, w, P) :: (\Gamma \,;\, \Delta, (x_\mathsf{T} : A_\mathsf{T}))} \ \mathsf{proc}_\mathsf{T}$$

- Case (msg) : We have $\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{msg}(x_m, w, M) :: (\Gamma \,;\, \Delta, (x_m : A))$. Inverting the msg rule,

$$\frac{\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta') \qquad \cdot \,;\, \cdot \,;\, \Delta' \overset{q}{\vdash} M :: (x_m : A)}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega, \mathsf{msg}(x_m, w, M) :: (\Gamma \,;\, \Delta, (x_m : A))} \ \mathsf{msg}$$

  we get $\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta')$. By the induction hypothesis, $\Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \overset{E+q+w}{\vDash} \Omega, \mathsf{msg}(x_m, w, M) :: (\Gamma \,;\, \Delta, (x_m : A))$. Applying the msg rule back,

$$\frac{\Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \overset{E}{\vDash} \Omega :: (\Gamma \,;\, \Delta, \Delta') \qquad \cdot \,;\, \cdot \,;\, \Delta' \overset{q}{\vdash} M :: (x_m : A)}{\Gamma_0, (c_\mathsf{S} : B_\mathsf{S}) \overset{E+q+w}{\vDash} \Omega, \mathsf{msg}(x_m, w, M) :: (\Gamma \,;\, \Delta, (x_m : A))} \ \mathsf{msg}$$

□

Lemma 4 (Process Weakening). *For a well-typed process $\Gamma \,;\, \Delta \overset{q}{\vdash} P :: (x_\mathsf{T} : A)$ and for a shared channel $c_\mathsf{S} : A_\mathsf{S} \notin \Gamma$, we have $\Gamma, (c_\mathsf{S} : A_\mathsf{S}) \,;\, \Delta \overset{q}{\vdash} P :: (x_\mathsf{T} : A)$.*

Proof. Analogous to Lemma 3. □

Lemma 5 (Permutation-Message). *Consider a well-typed configuration typed by the judgment $\Gamma_0 \overset{E}{\vDash} \Omega_1, \mathsf{msg}(c_m, w, M), \Omega_2, \mathsf{proc}(d_k, w', P(c_m)) :: (\Gamma \,;\, \Delta)$. Then, the message can be moved right such that the configuration $\Gamma_0 \overset{E}{\vDash} \Omega_1, \Omega_2, \mathsf{msg}(c_m, w, M), \mathsf{proc}(d_k, w', P(c_m)) :: (\Gamma \,;\, \Delta)$ is well-typed.*

Proof. We case analyze on the structure of the message.

- Case $(\otimes_n)$: We have $\Gamma_0 \overset{E}{\vDash} \Omega_1, \mathsf{msg}(c_m, w, \mathsf{send}\ c_m\ e_n \,;\, c_m \leftarrow c_m^+), \Omega_2, \mathsf{proc}(d_k, w', P(c_m)) :: (\Gamma \,;\, \Delta)$. First, we type the message

$$\cdot \,;\, \cdot \,;\, (c_m^+ : B), (e_n : A) \overset{q}{\vdash} \mathsf{send}\ c_m\ e_n \,;\, c_m \leftarrow c_m^+ :: (c_m : A \otimes_n B)$$

  Next, we invert the msg rule,

$$\frac{\begin{array}{c}\Gamma_0 \overset{E}{\vDash} \Omega_1 :: (\Gamma \,;\, \Delta, (c_m^+ : B), (e_n : A)) \\ \cdot \,;\, \cdot \,;\, (c_m^+ : B), (e_n : A) \overset{q}{\vdash} \mathsf{send}\ c_m\ e_n \,;\, c_m \leftarrow c_m^+ :: (c_m : A \otimes_n B)\end{array}}{\Gamma_0 \overset{E+q+w}{\vDash} \Omega_1, \mathsf{msg}(c_m, w, \mathsf{send}\ c_m\ e_n) :: (\Gamma \,;\, \Delta, (c_m : A \otimes_n B))} \ \mathsf{msg}$$

Since the channel c_m is only used by $\mathsf{proc}(d_k, w', P(c_m))$, we know that none of the processes or messages in Ω_2 can use it. Hence, we can move the message just left of the process $\mathsf{proc}(d_k, w', P(c_m))$.

□

Lemma 6 (Permutation-Process). *Consider a well-typed configuration typed by the judgment* $\Gamma_0 \overset{E}{\vDash} \Omega_1, \mathsf{proc}(c_m, w, P), \Omega_2, \mathsf{msg}(c_m^+, w', M(c_m)) :: (\Gamma \,;\, \Delta)$. *Then, the process can be moved right such that the configuration* $\Gamma_0 \overset{E}{\vDash} \Omega_1, \Omega_2, \mathsf{proc}(c_m, w, P), \mathsf{msg}(c_m^+, w', M(c_m)) :: (\Gamma \,;\, \Delta)$ *is well-typed.*

Proof. We case analyze on the structure of the message.

- Case $(\multimap_n)$: We have $\Gamma_0 \overset{E}{\vDash} \Omega_1, \mathsf{proc}(c_m, w, P), \Omega_2, \mathsf{msg}(c_m^+, w', \mathsf{send}\ c_m\ e_n \,;\, c_m^+ \leftarrow c_m) :: (\Gamma \,;\, \Delta)$. First, we type the message

$$\cdot \,;\, \cdot \,;\, (e_n : A), (c_m : A \multimap_n B) \Vdash^{g} \mathsf{send}\ c_m\ e_n \,;\, c_m^+ \leftarrow c_m :: (c_m^+ : B)$$

  Since the message is the only provider of channel c_m offered by $\mathsf{proc}(c_m, w, P)$, we know that none of the processes in Ω_2 can depend on it. Thus, the process can be moved to the without affecting the invariant for any process in Ω_2.

□

Lemma 7 (Permutation-Acquire). *Consider a well-typed configuration typed by the judgment* $\Gamma_0 \overset{E}{\vDash} \Omega_1, \mathsf{proc}(c_m, w', a_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}} \,;\, Q), \Omega_2, \mathsf{proc}(a_{\mathsf{S}}, w, a_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}} \,;\, P), \Omega_3 :: (\Gamma \,;\, \Delta)$. *Then, the acquiring process can be moved right such that the configuration* $\Gamma_0 \overset{E}{\vDash} \Omega_1, \Omega_2, \mathsf{proc}(a_{\mathsf{S}}, w, a_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}} \,;\, P)$, $\mathsf{proc}(c_m, w', a_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}} \,;\, Q), \Omega_3 :: (\Gamma \,;\, \Delta)$ *is well-typed.*

Proof. Due to independence, we know that $\mathsf{proc}(a_{\mathsf{S}}, w, a_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}} \,;\, P)$ can only depend on any channels at mode S or R. On the other hand, m can only be T or L. In particular, the shared process cannot depend on channel c_m, thus the acquiring process can be moved to the right of the shared process. □

Lemma 8 (Permutation-Release). *Consider a well-typed configuration typed by the judgment* $\Gamma_0 \overset{E}{\vDash} \Omega_1, \mathsf{proc}(c_m, w', a_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}} \,;\, Q), \Omega_2, \mathsf{proc}(a_{\mathsf{L}}, w, a_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}} \,;\, P), \Omega_3 :: (\Gamma \,;\, \Delta)$. *Then, the releasing process can be moved right such that the configuration* $\Gamma_0 \overset{E}{\vDash} \Omega_1, \Omega_2, \mathsf{proc}(a_{\mathsf{L}}, w, a_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}} \,;\, P)$, $\mathsf{proc}(c_m, w', a_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}} \,;\, Q), \Omega_3 :: (\Gamma \,;\, \Delta)$ *is well-typed.*

Proof. Due to independence, we know that $\mathsf{proc}(a_{\mathsf{L}}, w, a_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}} \,;\, P)$ can only depend on any channels at mode S or R. On the other hand, m can only be T or L. In particular, the shared process cannot depend on channel c_m, thus the releasing process can be moved to the right of the detaching process. □

Lemma 9 (Shared-Substitution). *If the process* $\Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}}) \,;\, \Delta \Vdash^{g} P_{x_{\mathsf{S}}} :: (z_m : C)$ *is well-typed, then* $\Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}) \,;\, \Delta \Vdash^{g} P_{b_{\mathsf{S}}} :: (z_m : C)$ *is also well-typed.*

Proof. We apply induction on the process typing judgment.

- Case ($\{\}E_{\mathsf{TT}}$) :

$$\frac{r = p + q \qquad \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}}) \supseteq \overline{a_{\mathsf{S}} : A} \qquad \Delta = \overline{d : D} \\ \Psi \Vdash^{p} M : \{A \leftarrow \overline{A}\ ;\ \overline{D}\}_{\mathsf{T}} \qquad \Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta', (y_{\mathsf{T}} : A) \vdash^{g} Q_{x_{\mathsf{S}}} :: (z_{\mathsf{T}} : C)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, \Delta' \vdash^{r} y_{\mathsf{T}} \leftarrow M \leftarrow a_{\mathsf{S}}\ ;\ d\ ;\ Q_{x_{\mathsf{S}}} :: (z_{\mathsf{T}} : C)} \ \{\}E_{\mathsf{TT}}$$

By the induction hypothesis, $\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta', (y_{\mathsf{T}} : A) \vdash^{g} Q_{b_{\mathsf{S}}} :: (z_{\mathsf{T}} : C)$. We simply substitute b_{S} for x_{S} in $\overline{a_{\mathsf{S}} : A}$. Hence, $\Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}) \supseteq [b_{\mathsf{S}}/x_{\mathsf{S}}]\overline{a_{\mathsf{S}} : A}$. Applying the $\{\}E_{\mathsf{TT}}$ rule back

$$\frac{r = p + q \qquad \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}) \supseteq [b_{\mathsf{S}}/x_{\mathsf{S}}]\overline{a_{\mathsf{S}} : A} \qquad \Delta = \overline{d : D} \\ \Psi \Vdash^{p} M : \{A \leftarrow \overline{A}\ ;\ \overline{D}\}_{\mathsf{T}} \qquad \Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta', (y_{\mathsf{T}} : A) \vdash^{g} Q_{b_{\mathsf{S}}} :: (z_{\mathsf{T}} : C)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, \Delta' \vdash^{r} y_{\mathsf{T}} \leftarrow M \leftarrow [b_{\mathsf{S}}/x_{\mathsf{S}}]a_{\mathsf{S}}\ ;\ d\ ;\ Q_{x_{\mathsf{S}}} :: (z_{\mathsf{T}} : C)} \ \{\}E_{\mathsf{TT}}$$

- Case (fwd) :

$$\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ (y_k : A) \vdash^{g} z_m \leftarrow y_k :: (z_m : A)$$

Here, the lemma holds trivially since x_{S} doesn't occur in $P_{x_{\mathsf{S}}}$. Therefore, $P_{x_{\mathsf{S}}} = P_{b_{\mathsf{S}}}$ and

$$\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ (y_k : A) \vdash^{g} z_m \leftarrow y_k :: (z_m : A)$$

- Case ($\multimap_n R$) :

$$\frac{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (y_n : A) \vdash^{g} P_{x_{\mathsf{S}}} :: (z_m : B)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta \vdash^{g} y_n \leftarrow \mathsf{recv}\ z_m\ ;\ P_{x_{\mathsf{S}}} :: (z_m : A \multimap_n B)} \ \multimap_n R$$

By the induction hypothesis, $\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (y_n : A) \vdash^{g} P_{b_{\mathsf{S}}} :: (z_m : B)$. Applying the $\multimap R$ rule,

$$\frac{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (y_n : A) \vdash^{g} P_{b_{\mathsf{S}}} :: (z_m : B)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta \vdash^{g} y_n \leftarrow \mathsf{recv}\ z_m\ ;\ P_{b_{\mathsf{S}}} :: (z_m : A \multimap_n B)} \ \multimap_n R$$

- Case ($\multimap_n L$) :

$$\frac{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (y_k : B) \vdash^{g} Q_{x_{\mathsf{S}}} :: (z_m : C)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}}), (x_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (w_n : A), (y_k : A \multimap B) \vdash^{g} \mathsf{send}\ y_k\ w_n\ ;\ Q_{x_{\mathsf{S}}} :: (z_m : C)} \ \multimap L$$

By the induction hypothesis, $\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (y_k : B) \vdash^{g} Q_{b_{\mathsf{S}}} :: (z_m : C)$. Applying the $\multimap_n L$ rule,

$$\frac{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (y_k : B) \vdash^{g} Q_{b_{\mathsf{S}}} :: (z_m : C)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : B_{\mathsf{S}})\ ;\ \Delta, (w_n : A), (y_k : A \multimap_n B) \vdash^{g} \mathsf{send}\ y_k\ w_n\ ;\ Q_{b_{\mathsf{S}}} :: (z_m : C)} \ \multimap_n L$$

- Case ($\uparrow^{\mathsf{S}}_{\mathsf{L}} L$) :

$$\frac{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}})\ ;\ \Delta, (x_{\mathsf{L}} : A_{\mathsf{L}}) \vdash^{g} Q :: (z_m : C)}{\Psi\ ;\ \Gamma, (b_{\mathsf{S}} : \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}), (x_{\mathsf{S}} : \uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}})\ ;\ \Delta \vdash^{g} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ;\ Q :: (z_m : C)} \ \uparrow^{\mathsf{S}}_{\mathsf{L}} L$$

The lemma holds trivially since x_S doesn't occur in Q. Hence, $[b_S/x_S]Q = Q$. Applying the $\uparrow^{S}_{L} L$ rule,

$$\dfrac{\Psi \ ; \ \Gamma, (b_S :\uparrow^{S}_{L} A_L) \ ; \ \Delta, (x_L : A_L) \vdash^{q} Q :: (z_m : C)}{\Psi \ ; \ \Gamma, (b_S :\uparrow^{S}_{L} A_L) \ ; \ \Delta \vdash^{q} x_L \leftarrow \text{acquire } b_S \ ; \ Q :: (z_m : C)} \ \uparrow^{S}_{L} L$$

- Case $(\downarrow^{S}_{L} L)$:

$$\dfrac{\Psi \ ; \ \Gamma, (b_S : B_S), (x_S : B_S), (y_S : A_S) \ ; \ \Delta \vdash^{q} Q_{x_S} :: (z_m : C)}{\Psi \ ; \ \Gamma, (b_S : B_S), (x_S : B_S) \ ; \ \Delta, (y_L :\downarrow^{S}_{L} A_S) \vdash^{q} y_S \leftarrow \text{release } y_L \ ; \ Q_{x_S} :: (z_m : C)} \ \downarrow^{S}_{L} L$$

By the induction hypothesis, $\Psi \ ; \ \Gamma, (b_S : A_S), (y_S : A_S) \ ; \ \Delta \vdash^{q} Q_{b_S} :: (z_m : C)$. Applying the $\downarrow^{S}_{L} L$ rule,

$$\dfrac{\Psi \ ; \ \Gamma, (b_S : A_S), (y_S : A_S) \ ; \ \Delta \vdash^{q} Q_{b_S} :: (z_m : C)}{\Psi \ ; \ \Gamma, (b_S : B_S) \ ; \ \Delta, (y_L :\downarrow^{S}_{L} A_S) \vdash^{q} y_S \leftarrow \text{release } y_L \ ; \ Q_{b_S} :: (z_m : C)} \ \downarrow^{S}_{L} L$$

□

Lemma 10 (Variable Substitution). *To substitute value for a variable from the functional context, we need the following two lemmas.*

- *If* V val *and* $\cdot \Vdash^{p} V : \tau$ *and* $\Psi, (x : \tau) \Vdash^{q} M : \sigma$*, then* $\Psi \Vdash^{p+q} [V/x]M : \sigma$.
- *If* V val *and* $\cdot \Vdash^{p} V : \tau$ *and* $\Psi, (x : \tau) \ ; \ \Gamma \ ; \ \Delta \Vdash^{q} P :: (c : A)$*, then* $\Psi \ ; \ \Gamma \ ; \ \Delta \Vdash^{p+q} [V/x]P :: (c : A)$

Theorem 1 (Expression Preservation). *If a well-typed expression* $\cdot \Vdash^{q} N : \tau$ *takes a step, i.e.,* $N \Downarrow V \mid \mu$*, then* V val *and* $q \geq \mu$ *and* $\cdot \Vdash^{q-\mu} V : \tau$.

Theorem 2 (Process Preservation). *Consider a closed well-formed and well-typed configuration* Ω *such that* $\Gamma_0 \overset{E}{\vDash} \Omega :: (\Gamma \ ; \ \Delta)$*. If the configuration takes a step, i.e.* $\Omega \mapsto \Omega'$*, then there exist* Γ_0', Γ' *such that* $\Gamma_0' \overset{E}{\vDash} \Omega' :: (\Gamma' \ ; \ \Delta)$*, i.e., the resulting configuration is well-typed.*

Proof. We case analyze on the semantics.

- Case (internal) : $\Omega = \mathcal{D}, \text{proc}(c_m, w, P[N])$ and $\Omega' = \mathcal{D}, \text{proc}(c_m, w + \mu, P[V])$. We case analyze on $P[N]$.
  - Case ($\rightarrow$ send) : $P[N] = \text{send } d_k \ N \ ; \ P$ and $P[V] = \text{send } d_k \ V \ ; \ P$, where $N \Downarrow V \mid \mu$. Suppose, $\Gamma_0 \overset{E+r+w}{\vDash} \mathcal{D}, \text{proc}(c_m, w, \text{send } d_k \ N \ ; \ P) :: (\Gamma \ ; \ \Delta, (c_m : C))$. Inverting the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \ ; \ \Delta_1, (d_k : \tau \rightarrow A), \Delta) \qquad \dfrac{r = p + q \quad \cdot \Vdash^{p} N : \tau \quad \cdot \ ; \ \Gamma_0 \ ; \ \Delta, (d_k : A) \vdash^{q} P :: (c_m : C)}{\cdot \ ; \ \Gamma_0 \ ; \ \Delta_1, (d_k : \tau \rightarrow A) \vdash^{r} \text{send } d_k \ N \ ; \ P :: (c_m : C)} \rightarrow L}{\Gamma_0 \overset{E+r+w}{\vDash} \mathcal{D}, \text{proc}(c_m, w, \text{send } d_k \ N \ ; \ P) :: (\Gamma \ ; \ \Delta, (c_m : C))} \ \text{proc}_m$$

By Theorem 1, we get that $\cdot \Vdash^{p-\mu} V : \tau$. Finally, we apply the same derivation again to get

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta_1, (d_k : \tau \to A), \Delta) \qquad \dfrac{r' = p - \mu + q \qquad \cdot \Vdash^{p-\mu} V : \tau \qquad \cdot \; ; \; \Gamma_0 \; ; \; \Delta, (d_k : A) \vdash^{q} P :: (c_m : C)}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, (d_k : \tau \to A) \vdash^{r'} \mathsf{send}\; d_k \; V \; ; \; P :: (c_m : C)} \to L}{\Gamma_0 \overset{E + r' + w + \mu}{\vDash} \mathsf{proc}(c_m, w + \mu, \mathsf{send}\; d_k \; N \; ; \; P), \mathcal{D} :: (\Gamma \; ; \; \Delta, (c_m : C))} \; \mathsf{proc}_m$$

and the proof succeeds since $r' + w + \mu = p - \mu + q + w + \mu = p + q + w = r + w$.

- Case $(\times\mathsf{send})$: Analogous to $\to \mathsf{send}$.
- Case $(E_{\mathsf{S}m})$: $\Omega = \mathcal{D}, \mathcal{D}, \mathsf{proc}(c_m, w, d_{\mathsf{S}} \leftarrow N \leftarrow \overline{a_{\mathsf{S}}} \; ; \; \overline{a_{\mathsf{R}}} \; ; \; Q)$ and $\Omega' = \mathcal{D}, \mathsf{proc}(c_m, w + \mu, d_{\mathsf{S}} \leftarrow V \leftarrow \overline{a_{\mathsf{S}}} \; ; \; \overline{a_{\mathsf{R}}} \; ; \; Q)$ where $N \Downarrow V \mid \mu$. Inverting the $\mathsf{proc_m}$ rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1, \Delta_2) \qquad \dfrac{r = p + q \quad \Gamma_0 \supseteq \overline{a_{\mathsf{S}} : A} \quad \Delta_1 = \overline{a_{\mathsf{R}} : D} \quad \cdot \Vdash^{p} N : \{A_{\mathsf{S}} \leftarrow \overline{A} \; ; \; \overline{D}\}_{\mathsf{S}} \quad \cdot \; ; \; \Gamma_0, (d_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta_2 \vdash^{q} Q :: (c_m : C)}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, \Delta_2 \vdash^{r} d_{\mathsf{S}} \leftarrow N \leftarrow \overline{a_{\mathsf{S}}} \; ; \; \overline{a_{\mathsf{R}}} \; ; \; Q :: (c_m : C)} \; E_{\mathsf{S}m}}{\Gamma_0 \overset{E + r + w}{\vDash} \mathcal{D}, \mathsf{proc}(c_m, w, d_{\mathsf{S}} \leftarrow N \leftarrow \overline{a_{\mathsf{S}}} \; ; \; \overline{a_{\mathsf{R}}} \; ; \; Q) :: (\Gamma \; ; \; \Delta, c_m : C)} \; \mathsf{proc}_m$$

By Theorem 1, $\cdot \Vdash^{p-\mu} V : \{A_{\mathsf{S}} \leftarrow \overline{D}\}_{\mathsf{S}}$. Applying the same derivation back,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1, \Delta_2) \qquad \dfrac{r' = p - \mu + q \quad \Gamma_0 \supseteq \overline{a_{\mathsf{S}} : A} \quad \Delta_1 = \overline{a_{\mathsf{R}} : D} \quad \cdot \Vdash^{p-\mu} V : \{A_{\mathsf{S}} \leftarrow \overline{A} \; ; \; \overline{D}\}_{\mathsf{S}} \quad \cdot \; ; \; \Gamma_0, (d_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta_2 \vdash^{q} Q :: (c_m : C)}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, \Delta_2 \vdash^{r'} d_{\mathsf{S}} \leftarrow V \leftarrow \overline{a_{\mathsf{S}}} \; ; \; \overline{a_{\mathsf{R}}} \; ; \; Q :: (c_m : C)} \; E_{\mathsf{S}m}}{\Gamma_0 \overset{E + r' + w + \mu}{\vDash} \mathcal{D}, \mathsf{proc}(c_m, w + \mu, d_{\mathsf{S}} \leftarrow V \leftarrow \overline{a_{\mathsf{S}}} \; ; \; \overline{a_{\mathsf{R}}} \; ; \; Q) :: (\Gamma \; ; \; \Delta, c_m : C)} \; \mathsf{proc}_m$$

and the proof succeeds since $r' + w + \mu = p - \mu + q + w + \mu = p + q + w = r + w$.

- Case $(E_{\mathsf{R}m}, E_{\mathsf{TT}})$: Analogous to $E_{\mathsf{S}m}$.

• Case $(\{\}E_{\mathsf{ST}})$: $\Omega = \mathcal{D}, \mathsf{proc}(d_{\mathsf{T}}, w, x_{\mathsf{S}} \leftarrow \{x'_{\mathsf{S}} \leftarrow P_{x'_{\mathsf{S}}, \overline{y}, \overline{z}} \leftarrow \overline{y} \; ; \; \overline{z}\} \leftarrow \overline{a} \; ; \; \overline{b} \; ; \; Q)$ and $\Omega' = \mathcal{D}, \mathsf{proc}(c_{\mathsf{S}}, 0, P_{c_{\mathsf{S}}, \overline{a}, \overline{b}}), \mathsf{proc}(d_{\mathsf{T}}, w, [c_{\mathsf{S}}/x_{\mathsf{S}}]Q)$. Inverting the $\mathsf{proc_T}$ rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1, \Delta_2) \qquad \dfrac{\dfrac{\Gamma_y = \overline{y : A} \qquad \Delta_z = \overline{z : D} \qquad \cdot \; ; \; \Gamma_y \; ; \; \Delta_z \vdash^{p} P_{x'_{\mathsf{S}}, \overline{y}, \overline{z}} :: (x'_{\mathsf{S}} : A_{\mathsf{S}})}{\cdot \Vdash^{p} \{x'_{\mathsf{S}} \leftarrow P_{x'_{\mathsf{S}}, \overline{y}, \overline{z}} \leftarrow \overline{y} \; ; \; \overline{z}\} : \{A_{\mathsf{S}} \leftarrow \overline{A} \; ; \; \overline{D}\}_{\mathsf{S}}} \{\}I_{\mathsf{S}} \quad r = p + q \quad \Gamma_0 \supseteq \overline{a : A} \quad \Delta_1 = \overline{b : D} \quad (A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \quad \cdot \; ; \; \Gamma_0, (x_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta_2 \vdash^{q} Q :: (d_{\mathsf{T}} : A_{\mathsf{T}})}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, \Delta_2 \vdash^{r} x_{\mathsf{S}} \leftarrow \{x'_{\mathsf{S}} \leftarrow P_{x'_{\mathsf{S}}, \overline{y}, \overline{z}} \leftarrow \overline{y} \; ; \; \overline{z}\} \leftarrow \overline{a} \; ; \; \overline{b} \; ; \; Q :: (d_{\mathsf{T}} : A_{\mathsf{T}})} \{\}E_{SC}}{\Gamma_0 \overset{E + r + w}{\vDash} \mathcal{D}, \mathsf{proc}(d_{\mathsf{T}}, w, x_{\mathsf{S}} \leftarrow \{x'_{\mathsf{S}} \leftarrow P_{x'_{\mathsf{S}}, \overline{y}, \overline{z}} \leftarrow \overline{y} \; ; \; \overline{z}\} \leftarrow \overline{a} \; ; \; \overline{b} \; ; \; Q) :: (\Gamma \; ; \; \Delta, (d_{\mathsf{T}} : A_{\mathsf{T}}))} \; \mathsf{proc}_{\mathsf{T}}$$

The premise for $\{\}I_{\mathsf{S}}$ gives us $\cdot \; ; \; \Gamma_y \; ; \; \Delta_z \vdash^{p} P_{x'_{\mathsf{S}}, \overline{y}, \overline{z}} :: (x'_{\mathsf{S}} : A_{\mathsf{S}})$, which by Lemma 1, gives us $\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \vdash^{p} P_{c_{\mathsf{S}}, \overline{a}, \overline{b}} :: (c_{\mathsf{S}} : A_{\mathsf{S}})$. Then, by Lemma 4, we get $\cdot \; ; \; \Gamma_0, (c_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta_1 \vdash^{p} P_{c_{\mathsf{S}}, \overline{a}, \overline{b}} :: (c_{\mathsf{S}} : A_{\mathsf{S}})$

Similarly, we get $\cdot\ ;\ \Gamma_0, (c_S : A_S)\ ;\ \Delta_2 \overset{g}{\Vdash} [c_S/x_S]Q :: (d_T : A_T)$. First, using Lemma 3, we get $\Gamma_0, (c_S : A_S) \overset{E}{\models} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, \Delta_2)$. Next, apply the proc_S rule,

$$\frac{\Gamma_0, (c_S : A_S) \overset{E}{\models} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, \Delta_2) \qquad \cdot\ ;\ \Gamma_0, (c_S : A_S)\ ;\ \Delta_1 \overset{p}{\Vdash} P_{c_S, \overline{a}, \overline{b}} :: (c_S : A_S)}{\Gamma_0, (c_S : A_S) \overset{E+p+0}{\models} \mathcal{D}, \mathsf{proc}(c_S, 0, P_{c_S, \overline{a}, \overline{b}}) :: (\Gamma, (c_S : A_S)\ ;\ \Delta, \Delta_2)}\ \mathsf{proc}_S$$

Call this new configuration $\mathcal{D}'$. Now, apply the proc_T rule.

$$\frac{\Gamma_0, (c_S : A_S) \overset{E+p+0}{\models} \mathcal{D}' :: (\Gamma, (c_S : A_S)\ ;\ \Delta, \Delta_2) \qquad \cdot\ ;\ \Gamma, (c_S : A_S)\ ;\ \Delta_2 \overset{g}{\Vdash} [c_S/x_S]Q :: (d_T : A_T)}{\Gamma_0, (c_S : A_S) \overset{E+p+q+w}{\models} \mathcal{D}', \mathsf{proc}(d_T, w, [c_S/x_S]Q) :: (\Gamma, (c_S : A_S)\ ;\ \Delta, (d_T : A_T))}\ \mathsf{proc}_T$$

where $E+p+q+w = E+r+w$ since $r = p+q$. Hence, in this case $\Gamma_0' = \Gamma_0, (c_S : A_S)$ and $\Gamma' = \Gamma, (c_S : A_S)$.

- Case $(\{\}E_{TT}) : \Omega = \mathcal{D}, \mathsf{proc}(d_T, w, x_T \leftarrow \{x_T' \leftarrow P_{x_T', \overline{y}, \overline{z}} \leftarrow \overline{y}\ ;\ \overline{z}\} \leftarrow a_S\ ;\ d\ ;\ Q)$ and $\Omega' = \mathcal{D}, \mathsf{proc}(c_T, 0, P_{c_T, \overline{a_S}, \overline{d}}), \mathsf{proc}(d_T, w, [c_T/x_T]Q)$. Inverting the proc_T rule

$$\frac{\dfrac{r = p+q \qquad \Gamma_0 \supseteq \overline{a_S : A} \qquad \Delta_1 = \overline{d : D} \qquad \dfrac{\Gamma_y = \overline{y : A} \qquad \Delta_z = \overline{z : D} \qquad \cdot\ ;\ \Gamma_y\ ;\ \Delta_z \overset{p}{\Vdash} P_{x_T', \overline{y}, \overline{z}} :: (x_T' : A)}{\cdot \overset{p}{\Vdash} \{x_T' \leftarrow P_{x_T', \overline{y}, \overline{z}} \leftarrow \overline{y}\ ;\ \overline{z}\} : \{A \leftarrow \overline{A}\ ;\ \overline{D}\}_T}\ \{\}I_T \qquad \cdot\ ;\ \Gamma_0\ ;\ \Delta_2, (x_T : A) \overset{g}{\Vdash} Q :: (d_T : C)}{\cdot\ ;\ \Gamma_0\ ;\ \Delta_1, \Delta_2 \overset{r}{\Vdash} x_T \leftarrow \{x_T' \leftarrow P_{x_T', \overline{y}, \overline{z}} \leftarrow \overline{y}\ ;\ \overline{z}\} \leftarrow \overline{a_S}\ ;\ \overline{d}\ ;\ Q :: (d_T : C)}\ \{\}E_{TT} \qquad \Gamma_0 \overset{E}{\models} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, \Delta_2)}{\Gamma_0 \overset{E+r+w}{\models} \mathcal{D}, \mathsf{proc}(d_T, w, x_T \leftarrow \{x_T' \leftarrow P_{x_T', \overline{y}, \overline{z}} \leftarrow \overline{y}\ ;\ \overline{z}\} \leftarrow \overline{a_S}\ ;\ \overline{d}\ ;\ Q) :: (\Gamma\ ;\ \Delta, (d_T : C))}\ \mathsf{proc}_T$$

We contract all multiple occurrences of the same channel in $\overline{a_S : A}$. Let the resulting vector be $\Gamma' = \overline{a_S' : A'}$. We know, by Lemma 9 that $\cdot\ ;\ \Gamma'\ ;\ \Delta' \overset{p}{\Vdash} P_{x_T', \overline{a_S'}, \overline{z}} :: (x_T' : A)$ is well-typed. Next, by Lemma 1, we get $\Gamma'\ ;\ \Delta_1 \overset{p}{\Vdash} P_{c_T, \overline{a_S'}, \overline{d}} :: (c_T : A)$. Finally, we weaken Γ' using Lemma 4 to get $\cdot\ ;\ \Gamma_0\ ;\ \Delta_1 \overset{p}{\Vdash} P_{c_T, \overline{a_S'}, \overline{d}} :: (c_T : A)$. Also, note that since $\overline{a_S'}$ is a refinement of $\overline{a_S}$ by eliminating duplicates, $P_{c_T, \overline{a_S'}, \overline{d}} = P_{c_T, \overline{a_S}, \overline{d}}$. Hence, we apply the proc_T rule,

$$\frac{\Gamma_0 \overset{E}{\models} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, \Delta_2) \qquad \cdot\ ;\ \Gamma_0\ ;\ \Delta_1 \overset{p}{\Vdash} P_{c_T, \overline{a_S}, \overline{d}} :: (c_T : A)}{\Gamma_0 \overset{E+p+0}{\models} \mathcal{D}, \mathsf{proc}(c_T, 0, P_{c_T, \overline{a_S}, \overline{d}}) :: (\Gamma\ ;\ \Delta, \Delta_2, (c_T : A))}\ \mathsf{proc}_T$$

Call this new configuration $\mathcal{D}'$. Also, applying renaming using Lemma 1, we get $\cdot\ ;\ \Gamma_0\ ;\ \Delta_2, (c_T : A) \overset{g}{\Vdash} [c_T/x_T]Q :: (d_T : C)$. Again, applying the proc_T rule, we get

$$\frac{\Gamma_0 \overset{E+p+0}{\models} \mathcal{D}' :: (\Gamma\ ;\ \Delta, \Delta_2, (c_T : A)) \qquad \cdot\ ;\ \Gamma_0\ ;\ \Delta_2, (c_T : A) \overset{g}{\Vdash} [c_T/x_T]Q :: (d_T : C)}{\Gamma_0 \overset{E+p+q+w}{\models} \mathcal{D}', \mathsf{proc}(d_T, w, [c_T/x_T]Q) :: (\Gamma\ ;\ \Delta, (d_T : C))}\ \mathsf{proc}_T$$

where $E + p + q + w = E + r + w$ since $r = p + q$.

- Case (fwd^+) : $\Omega = \mathcal{D}, \text{msg}(d_k, w', M), \text{proc}(c_m, w, c_m \leftarrow d_k)$ and $\Omega' = \text{msg}(c_m, w + w', [c_m/d_k]M)$. First, inverting the msg rule,

$$\frac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Omega \;;\; \Delta, \Delta_1) \qquad \cdot \;;\; \cdot \;;\; \Delta_1 \vdash^{g} M :: (d_k : A)}{\Gamma_0 \overset{E+q+w'}{\vDash} \mathcal{D}, \text{msg}(d_k, w', M) :: (\Gamma \;;\; \Delta, (d_k : A))}\;\text{msg}$$

Call this new configuration $\mathcal{D}'$. Next, inverting the proc_m rule

$$\frac{\Gamma_0 \overset{E+q+w'}{\vDash} \mathcal{D}' :: (\Gamma \;;\; \Delta, (d_k : A)) \qquad \cdot \;;\; \Gamma_0 \;;\; (d_k : A) \vdash^{0} c_m \leftarrow d_k :: (c_m : A)}{\Gamma_0 \overset{E+q+w'+0+w}{\vDash} \mathcal{D}', \text{proc}(c_m, w, c_m \leftarrow d_k) :: (\Gamma \;;\; \Delta, (c_m : A))}\;\text{proc}_m$$

Using Lemma 1, we get $\cdot \;;\; \cdot \;;\; \Delta_1 \vdash^{g} [c_m/d_k]M :: (c_m : A)$. Applying the msg rule,

$$\frac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Omega \;;\; \Delta, \Delta_1) \qquad \cdot \;;\; \cdot \;;\; \Delta_1 \vdash^{g} [c_m/d_k]M :: (c_m : A)}{\Gamma_0 \overset{E+q+w'+w}{\vDash} \mathcal{D}, \text{msg}(c_m, w', [c_m/d_k]M) :: (\Gamma \;;\; \Delta, (c_m : A))}\;\text{msg}$$

- Case (fwd^-) : $\Omega = \mathcal{D}, \text{proc}(c_m, w, c_m \leftarrow d_k), \text{msg}(e_l, w', M(c_m))$ and $\Omega' = \text{msg}(e_l, w + w', M(d_k))$. First, inverting on the proc_m rule

$$\frac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \;;\; \Delta, \Delta_1, (d_k : A)) \qquad \cdot \;;\; \Gamma_0 \;;\; (d_k : A) \vdash^{0} c_m \leftarrow d_k :: (c_m : A)}{\Gamma_0 \overset{E+0+w}{\vDash} \mathcal{D}, \text{proc}(c_m, w, c_m \leftarrow d_k) :: (\Gamma \;;\; \Delta, \Delta_1, (c_m : A))}\;\text{proc}_m$$

Call this new configuration $\mathcal{D}'$. Next, inverting on the msg rule,

$$\frac{\Gamma_0 \overset{E+w}{\vDash} \mathcal{D}' :: (\Gamma \;;\; \Delta, \Delta_1, (c_m : A)) \qquad \cdot \;;\; \cdot \;;\; \Delta_1, (c_m : A) \vdash^{g} M(c_m) :: (e_l : C)}{\Gamma_0 \overset{E+w+q+w'}{\vDash} \mathcal{D}', \text{msg}(e_l, w', M(c_m)) :: (\Gamma \;;\; \Delta, (e_l : C))}\;\text{msg}$$

Using Lemma 1, we get $\cdot \;;\; \cdot \;;\; \Delta_1, (d_k : A) \vdash^{g} M(d_k) :: (e_l : C)$. Reapplying the msg rule,

$$\frac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \;;\; \Delta, \Delta_1, (d_k : A)) \qquad \cdot \;;\; \cdot \;;\; \Delta_1, (d_k : A) \vdash^{g} M(d_k) :: (e_l : C)}{\Gamma_0 \overset{E+q+w+w'}{\vDash} \mathcal{D}, \text{msg}(e_l, w', M(d_k)) :: (\Gamma \;;\; \Delta, (e_l : C))}\;\text{msg}$$

- Case $(\oplus C_s) : \Omega = \mathcal{D}, \text{proc}(c_m, w, c_m.\ell \; ; \; P)$ and $\Omega' = \mathcal{D}, \text{proc}(c_m^+, w, [c_m^+/c_m]P), \text{msg}(c_m, 0, c_m.\ell \; ; \; c_m \leftarrow c_m^+)$. First, inverting on the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1) \qquad \dfrac{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \vdash^{g} P :: (c_m : A_\ell)}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \vdash^{g} c_m.\ell \; ; \; P :: (c_m : \oplus\{l : A_l\}_{l \in L})} \oplus R}{\Gamma_0 \overset{E+q+w}{\vDash} \mathcal{D}, \text{proc}(c_m, w, c_m.\ell \; ; \; P) :: (\Gamma \; ; \; \Delta, (c_m : \oplus\{l : A_l\}_{l \in L}))} \; \text{proc}_m$$

Using Lemma 1, we get $\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \vdash^{g} [c_m^+/c_m]P :: (c_m^+ : A_\ell)$. Now, applying the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1) \qquad \cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \vdash^{g} [c_m^+/c_m]P :: (c_m^+ : A_\ell)}{\Gamma_0 \overset{E+q+w}{\vDash} \mathcal{D}, \text{proc}(c_m, w, c_m.\ell \; ; \; P) :: (\Gamma \; ; \; \Delta, (c_m^+ : A_\ell))} \; \text{proc}_m$$

Next, typing the message

$$\cdot \; ; \; \cdot \; ; \; (c_m^+ : A_\ell) \vdash^{0} c_m.\ell \; ; \; c_m \leftarrow c_m^+ :: (c_m : \oplus\{l : A_l\}_{l \in L})$$

Call this new configuration $\mathcal{D}'$. Applying the msg rule next

$$\dfrac{\Gamma_0 \overset{E+q+w}{\vDash} \mathcal{D}' :: (\Gamma \; ; \; \Delta, (c_m : A_\ell)) \qquad \cdot \; ; \; \cdot \; ; \; (c_m^+ : A_\ell) \vdash^{0} c_m.\ell \; ; \; c_m \leftarrow c_m^+ :: (c_m : \oplus\{l : A_l\}_{l \in L})}{\Gamma_0 \overset{E+q+w}{\vDash} \mathcal{D}', \text{msg}(c_m, 0, c_m.\ell \; ; \; c_m \leftarrow c_m^+) :: (\Gamma \; ; \; \Delta, (c_m : \oplus\{l : A_l\}_{l \in L}))} \; \text{msg}$$

- Case $(\oplus C_r) : \Omega = \mathcal{D}, \text{msg}(c_m, w, c_m.\ell \; ; \; c_m \leftarrow c_m^+), \text{proc}(d_k, w', \text{case } c_m \; (l \Rightarrow Q_l)_{l \in L})$ and $\Omega' = \mathcal{D}, \text{proc}(d_k, w + w', [c_m^+/c_m]Q_\ell)$. First, inverting the msg rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1, (c_m^+ : A_\ell)) \qquad \cdot \; ; \; \cdot \; ; \; (c_m^+ : A_\ell) \vdash^{0} c_m.\ell \; ; \; c_m \leftarrow c_m^+ :: (c_m : \oplus\{l : A_l\}_{l \in L})}{\Gamma_0 \overset{E+0+w}{\vDash} \mathcal{D}, \text{msg}(c_m, w, c_m.\ell \; ; \; c_m \leftarrow c_m^+) :: (\Gamma \; ; \; \Delta, \Delta_1, (c_m : \oplus\{l : A_l\}_{l \in L}))} \; \text{msg}$$

Call this new configuration $\mathcal{D}'$. Next, inverting the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E+0+w}{\vDash} \mathcal{D}' :: (\Gamma \; ; \; \Delta, \Delta_1, (c_m : \oplus\{l : A_l\}_{l \in L})) \qquad \dfrac{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, (c_m : A_l) \vdash^{g} Q_l :: (d_k : C)}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, (c_m : \oplus\{l : A_l\}_{l \in L}) \vdash^{g} \text{case } c_m \; (l \Rightarrow Q_l)_{l \in L} :: (d_k : C)} \oplus R}{\Gamma_0 \overset{E+0+w+q+w'}{\vDash} \mathcal{D}', \text{proc}(d_k, w', \text{case } c_m \; (l \Rightarrow Q_l)_{l \in L}) :: (\Gamma \; ; \; \Delta, (d_k : C))} \; \text{proc}_m$$

Renaming using Lemma 1, we get $\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, (c_m^+ : A_\ell) \vdash^{g} [c_m^+/c_m]Q_\ell :: (d_k : C)$. Next, we apply the proc_m rule

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma \; ; \; \Delta, \Delta_1, (c_m^+ : A_\ell)) \qquad \cdot \; ; \; \Gamma_0 \; ; \; \Delta_1, (c_m^+ : A_\ell) \vdash^{g} [c_m^+/c_m]Q_\ell :: (d_k : C)}{\Gamma_0 \overset{E+q+w+w'}{\vDash} \mathcal{D}', \text{proc}(d_k, w + w', [c_m^+/c_m]Q_\ell) :: (\Gamma \; ; \; \Delta, (d_k : C))} \; \text{proc}_m$$

- Case $(\multimap_n C_s)$: $\Omega = \mathcal{D}, \mathsf{proc}(d_k, w, \mathsf{send}\ c_m\ e_n\ ;\ P)$ and $\Omega' = \mathcal{D}, \mathsf{msg}(c_m^+, 0, \mathsf{send}\ c_m\ e_n\ ;\ c_m^+ \leftarrow c_m), \mathsf{proc}(d_k, w, [c_m^+/c_m]P)$. First, we invert the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, (e_{\mathsf{R}} : A), (c_m : A \multimap B)) \qquad \dfrac{\cdot\ ;\ \Gamma\ ;\ \Delta_1, (c_m : B) \vdash^{g} P :: (d_k : C)}{\cdot\ ;\ \Gamma\ ;\ \Delta_1, (e_{\mathsf{R}} : A), (c_m : A \multimap B) \vdash^{g} \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ P :: (d_k : C)} \multimap L}{\Gamma_0 \overset{E+q+w}{\vDash} \mathcal{D}, \mathsf{proc}(d_k, w, \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ P) :: (\Gamma\ ;\ \Delta, (d_k : C))} \ \mathsf{proc}_m$$

Using renaming (Lemma 1), we get $\Gamma\ ;\ \Delta_1, (c_m^+ : B) \vdash^{g} [c_m^+/c_m]P :: (d_k : C)$. Next, we type the message

$$\cdot\ ;\ \Gamma\ ;\ (e_{\mathsf{R}} : A), (c_m : A \multimap B) \vdash^{0} \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m :: (c_m^+ : B)$$

Next, we apply the msg rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, (e_{\mathsf{R}} : A), (c_m : A \multimap B)) \qquad \cdot\ ;\ \Gamma\ ;\ (e_{\mathsf{R}} : A), (c_m : A \multimap B) \vdash^{0} \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m :: (c_m^+ : B)}{\Gamma_0 \overset{E}{\vDash} \mathcal{D}, \mathsf{msg}(c_m^+, 0, \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m) :: (\Gamma\ ;\ \Delta, \Delta_1, (c_m^+ : B))} \ \mathsf{msg}$$

Call this new configuration $\mathcal{D}'$. Next, we apply the proc_m rule

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D}' :: (\Gamma\ ;\ \Delta, \Delta_1, (c_m^+ : B)) \qquad \cdot\ ;\ \Gamma\ ;\ \Delta_1, (c_m^+ : B) \vdash^{g} [c_m^+/c_m]P :: (d_k : C)}{\Gamma_0 \overset{E+q+w}{\vDash} \mathcal{D}', \mathsf{proc}(d_k, w, [c_m^+/c_m]P) :: (\Gamma\ ;\ \Delta, (d_k : C))} \ \mathsf{proc}_m$$

- Case $(\multimap C_r)$: $\Omega = \mathcal{D}, \mathsf{proc}(c_m, w', x_{\mathsf{R}} \leftarrow \mathsf{recv}\ c_m\ ;\ Q), \mathsf{msg}(c_m^+, w, \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m)$ and $\Omega' = \mathcal{D}, \mathsf{proc}(c_m^+, w + w', [c_m^+/c_m][e_{\mathsf{R}}/x_{\mathsf{R}}]Q)$. First, inverting the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, (e_{\mathsf{R}} : A)) \qquad \dfrac{\cdot\ ;\ \Gamma\ ;\ \Delta_1, (x_{\mathsf{R}} : A) \vdash^{g} Q :: (c_m : B)}{\cdot\ ;\ \Gamma\ ;\ \Delta_1 \vdash^{g} x_{\mathsf{R}} \leftarrow \mathsf{recv}\ c_m\ ;\ Q :: (c_m : A \multimap B)} \multimap R}{\Gamma_0 \overset{E+q+w'}{\vDash} \mathcal{D}, \mathsf{proc}(c_m, w', x_{\mathsf{R}} \leftarrow \mathsf{recv}\ c_m\ ;\ Q) :: (\Gamma\ ;\ \Delta, (e_{\mathsf{R}} : A), (c_m : A \multimap B))} \ \mathsf{proc}_m$$

Call this new configuration $\mathcal{D}'$. Next, we type the message.

$$\cdot\ ;\ \Gamma\ ;\ (e_{\mathsf{R}} : A), (c_m : A \multimap B) \vdash^{0} \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m :: (c_m^+ : B)$$

Inverting the msg rule,

$$\dfrac{\Gamma_0 \overset{E+q+w'}{\vDash} \mathcal{D}' :: (\Gamma\ ;\ \Delta, (e_{\mathsf{R}} : A), (c_m : A \multimap B)) \qquad \cdot\ ;\ \Gamma\ ;\ (e_{\mathsf{R}} : A), (c_m : A \multimap B) \vdash^{0} \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m :: (c_m^+ : B)}{\Gamma_0 \overset{E+q+w'+0+w'}{\vDash} \mathcal{D}', \mathsf{msg}(c_m^+, w, \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m) :: (\Gamma\ ;\ \Delta, (c_m^+ : B))} \ \mathsf{msg}$$

By renaming using Lemma 1, $\cdot\ ;\ \Gamma\ ;\ \Delta_1, (e_{\mathsf{R}} : A) \vdash^{g} [c_m^+/c_m][e_{\mathsf{R}}/x_{\mathsf{R}}]Q :: (c_m^+ : B)$. Now, applying the proc_m rule,

$$\frac{\Gamma_0 \overset{E}{\vDash} \mathcal{D} :: (\Gamma\ ;\ \Delta, \Delta_1, (e_{\mathsf{R}} : A)) \qquad \cdot\ ;\ \Gamma\ ;\ \Delta_1, (e_{\mathsf{R}} : A) \vdash^{g} [c_m^+/c_m][e_{\mathsf{R}}/x_{\mathsf{R}}]Q :: (c_m^+ : B)}{\Gamma_0 \overset{E+q+w'}{\vDash} \mathcal{D}, \mathsf{proc}(c_m^+, w + w', [c_m^+/c_m][e_{\mathsf{R}}/x_{\mathsf{R}}]Q) :: (\Gamma\ ;\ \Delta, (c_m^+ : B))}\ \mathsf{proc}_m$$

- Case $(\uparrow_{\mathsf{L}}^{\mathsf{S}} C) : \Omega = \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{S}}, w', x_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}}\ ;\ P_{x_{\mathsf{L}}}), \mathsf{proc}(c_m, w, x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ;\ Q_{x_{\mathsf{L}}})$ and $\Omega' = \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{L}}, w', P_{a_{\mathsf{L}}}), \mathsf{proc}(c_m, w, Q_{a_{\mathsf{L}}})$. Applying the $\mathsf{proc}_{\mathsf{S}}$ rule first,

$$\frac{(a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \in \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \quad (\uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}, \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ \mathsf{esync} \quad \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \overset{E}{\vDash} \mathcal{D}_1 :: (\Gamma\ ;\ \Delta, \Delta_1, \Delta_2) \quad \mathcal{E}}{\Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \overset{E+p+w'}{\vDash} \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{S}}, w', x_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}}\ ;\ P_{x_{\mathsf{L}}}) :: (\Gamma, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta, \Delta_2)}\ \mathsf{proc}_{\mathsf{S}}$$

where $\mathcal{E}$ is

$$\frac{\cdot\ ;\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta_1 \vdash^{p} P_{x_{\mathsf{L}}} :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\cdot\ ;\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta_1 \vdash^{p} x_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}}\ ;\ P_{x_{\mathsf{L}}} :: (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})}\ \uparrow_{\mathsf{L}}^{\mathsf{S}} R$$

Call this new configuration $\mathcal{D}_1'$. Applying the proc_m rule next,

$$\frac{\Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \overset{E'}{\vDash} \mathcal{D}_1' :: (\Gamma, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta, \Delta_2) \qquad \dfrac{\cdot\ ;\ \Gamma_0\ ;\ \Delta_2, (x_{\mathsf{L}} : A_{\mathsf{L}}) \vdash^{g} Q_{x_{\mathsf{L}}} :: (c_m : C)}{\cdot\ ;\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta_2 \vdash^{g} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ;\ Q_{x_{\mathsf{L}}} :: (c_m : C)}\ \uparrow_{\mathsf{L}}^{\mathsf{S}} L}{\Gamma_0 \overset{E'+q+w}{\vDash} \mathcal{D}_1', \mathsf{proc}(c_m, w, x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ;\ Q_{x_{\mathsf{L}}}) :: (\Gamma, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta, (c_m : C))}\ \mathsf{proc}_m$$

From the first premise, we get by Lemma 1, $\cdot\ ;\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta_1 \vdash^{p} P_{a_{\mathsf{L}}} :: (a_{\mathsf{L}} : A_{\mathsf{L}})$ while from the second premise, we get by Lemma 1 and Lemma 4, $\cdot\ ;\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta_2, (a_{\mathsf{L}} : A_{\mathsf{L}}) \vdash^{g} Q_{a_{\mathsf{L}}} :: (c_m : C)$. Reapplying the $\mathsf{proc}_{\mathsf{L}}$ rule,

$$\frac{\begin{array}{c}(a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \in \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \qquad (A_{\mathsf{L}}, \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ \mathsf{esync} \\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \overset{E}{\vDash} \mathcal{D}_1 :: (\Gamma\ ;\ \Delta, \Delta_1, \Delta_2) \quad \cdot\ ;\ \Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta_1 \vdash^{p} P_{a_{\mathsf{L}}} :: (a_{\mathsf{L}} : A_{\mathsf{L}})\end{array}}{\Gamma_0, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}) \overset{E+p+w'}{\vDash} \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{L}}, w', P_{a_{\mathsf{L}}}) :: (\Gamma, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta, \Delta_2, (a_{\mathsf{L}} : A_{\mathsf{L}}))}\ \mathsf{proc}_{\mathsf{L}}$$

Call this new configuration $\mathcal{D}_1''$. Reapplying the proc_m rule,

$$\dfrac{\begin{array}{c}\Gamma_0, (a_{\mathsf{S}} :\uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \overset{E'}{\vDash} \mathcal{D}_1'' :: (\Gamma, (a_{\mathsf{S}} :\uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \; ; \; \Delta, \Delta_2, (a_{\mathsf{L}} : A_{\mathsf{L}})) \\ \cdot \; ; \; \Gamma_0, (a_{\mathsf{S}} :\uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \; ; \; \Delta_2, (a_{\mathsf{L}} : A_{\mathsf{L}}) \overset{q}{\vdash} Q_{a_{\mathsf{L}}} :: (c_m : C)\end{array}}{\Gamma_0, (a_{\mathsf{S}} :\uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \overset{E'+q+w}{\vDash} \mathcal{D}_1'', \mathsf{proc}(c_m, w, Q_{a_{\mathsf{L}}}) :: (\Gamma', (a_{\mathsf{S}} :\uparrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{L}}) \; ; \; \Delta', (c_m : C))} \; \mathsf{proc}_m$$

- Case $(\downarrow^{\mathsf{S}}_{\mathsf{L}} C) : \Omega = \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{L}}, w', x_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}} \; ; \; P_{x_{\mathsf{S}}}), \mathsf{proc}(c_{\mathsf{T}}, w, x_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}} \; ; \; Q_{x_{\mathsf{S}}})$ and $\Omega' = \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{S}}, w', P_{a_{\mathsf{S}}}), \mathsf{proc}(c_{\mathsf{L}}, w, Q_{a_{\mathsf{S}}})$. Applying the $\mathsf{proc}_{\mathsf{L}}$ rule first,

$$\dfrac{(a_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0 \qquad (\downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0 \overset{E}{\vDash} \mathcal{D}_1 :: (\Gamma \; ; \; \Delta, \Delta_1, \Delta_2) \qquad \mathcal{E}}{\Gamma_0 \overset{E+p+w'}{\vDash} \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{L}}, w', x_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}} \; ; \; P_{x_{\mathsf{S}}}) :: (\Gamma, (a_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta, \Delta_2, (a_{\mathsf{L}} :\downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}}))} \; \mathsf{proc}_{\mathsf{L}}$$

where $\mathcal{E}$ is

$$\dfrac{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \overset{p}{\vdash} P_{x_{\mathsf{S}}} :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \overset{p}{\vdash} x_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}} \; ; \; P_{x_{\mathsf{S}}} :: (a_{\mathsf{L}} :\downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}})} \downarrow^{\mathsf{S}}_{\mathsf{L}} R$$

Call this configuration $\mathcal{D}_1'$. Applying the proc_m rule,

$$\dfrac{\dfrac{\cdot \; ; \; \Gamma_0, (x_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta_2 \overset{q}{\vdash} Q_{x_{\mathsf{S}}} :: (c_m : C)}{\cdot \; ; \; \Gamma_0 \; ; \; \Delta_2, (a_{\mathsf{L}} :\downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}}) \overset{q}{\vdash} x_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}} \; ; \; Q_{x_{\mathsf{S}}} :: (c_m : C)} \downarrow^{\mathsf{S}}_{\mathsf{L}} L \qquad \Gamma_0 \overset{E'}{\vDash} \mathcal{D}_1' :: (\Gamma, (a_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta, \Delta_2, (a_{\mathsf{L}} :\downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}}))}{\Gamma_0 \overset{E'+q+w}{\vDash} \mathcal{D}_1', \mathsf{proc}(c_{\mathsf{T}}, w, x_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}} \; ; \; Q_{x_{\mathsf{S}}}) :: (\Gamma, (a_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta, (c_m : C))} \; \mathsf{proc}_m$$

From the first premise, we get by Lemma 1, $\cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \overset{p}{\vdash} P_{a_{\mathsf{S}}} :: (a_{\mathsf{S}} : A_{\mathsf{S}})$. From the second premise, by Lemma 9 (contracting $a_{\mathsf{S}} : A_{\mathsf{S}}$ and $x_{\mathsf{S}} : A_{\mathsf{S}}$), we get $\cdot \; ; \; \Gamma_0 \; ; \; \Delta_2 \overset{q}{\vdash} Q_{a_{\mathsf{S}}} :: (c_m : C)$. Finally, applying the $\mathsf{proc}_{\mathsf{S}}$ rule,

$$\dfrac{(a_{\mathsf{S}} : A_{\mathsf{S}}) \in \Gamma_0 \qquad (A_{\mathsf{S}}, A_{\mathsf{S}})\ \mathsf{esync} \qquad \Gamma_0 \overset{E}{\vDash} \mathcal{D}_1 :: (\Gamma \; ; \; \Delta, \Delta_1, \Delta_2) \qquad \cdot \; ; \; \Gamma_0 \; ; \; \Delta_1 \overset{p}{\vdash} P_{a_{\mathsf{S}}} :: (a_{\mathsf{S}} : A_{\mathsf{S}})}{\Gamma_0 \overset{E+p+w'}{\vDash} \mathcal{D}_1, \mathsf{proc}(a_{\mathsf{S}}, w', P_{a_{\mathsf{S}}}) :: (\Gamma, (a_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta, \Delta_2)} \; \mathsf{proc}_{\mathsf{S}}$$

Call this new configuration $\mathcal{D}_1''$. Applying the proc_m rule,

$$\dfrac{\Gamma_0 \overset{E'}{\vDash} \mathcal{D}_1'' :: (\Gamma, (a_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta, \Delta_2) \qquad \cdot \; ; \; \Gamma_0 \; ; \; \Delta_2 \overset{q}{\vdash} Q_{a_{\mathsf{S}}} :: (c_m : C)}{\Gamma_0 \overset{E'+q+w}{\vDash} \mathcal{D}_1'', \mathsf{proc}(c_m, w, Q_{a_{\mathsf{S}}}) :: (\Gamma, (a_{\mathsf{S}} : A_{\mathsf{S}}) \; ; \; \Delta, (c_m : C))} \; \mathsf{proc}_{\mathsf{T}}$$

□

Definition 1. *A process* $\mathsf{proc}(c_m, w, P)$ *is said to be poised if it is trying to receive a message on* c_m*. A message* $\mathsf{msg}(c_m, w, M)$ *is said to be poised if it is trying to send a message along* c_m*. A configuration* Ω *is said to be poised if all the processes and messages in* Ω *are poised. Concretely, the following processes are poised.*

- $\mathsf{proc}(c_m, w, c_m \overset{-}{\leftarrow} d_m)$
- $\mathsf{proc}(c_m, w, \mathsf{case}\ c_m\ (l_i \Rightarrow P_i)_{i \in I})$
- $\mathsf{proc}(c_m, w, x_{\mathsf{R}} \leftarrow \mathsf{recv}\ c_m\ ;\ P)$
- $\mathsf{proc}(c_m, w, x \leftarrow \mathsf{recv}\ c_m\ ;\ P)$
- $\mathsf{proc}(c_{\mathsf{S}}, w, c_{\mathsf{L}} \leftarrow \mathsf{accept}\ c_{\mathsf{S}}\ ;\ P)$
- $\mathsf{proc}(c_{\mathsf{L}}, w, c_{\mathsf{S}} \leftarrow \mathsf{detach}\ c_{\mathsf{L}}\ ;\ P)$
- $\mathsf{proc}(c_m, w, \mathsf{get}\ c_m\ \{r\}\ ;\ P)$

Similarly, the following messages are poised.

- $\mathsf{msg}(c_m, w, c_m.l_k\ ;\ P)$
- $\mathsf{msg}(c_m, w, \mathsf{send}\ c_m\ e_n\ ;\ P)$
- $\mathsf{msg}(c_m, w, \mathsf{send}\ c_m\ N\ ;\ P)$
- $\mathsf{msg}(c_m, w, \mathsf{close}\ c_m)$
- $\mathsf{msg}(c_m, w, \mathsf{pay}\ c_m\ \{r\}\ ;\ P)$

Theorem 3 (Process Progress). *Consider a closed well-formed and well-typed configuration* Ω *such that* $\Gamma_0 \overset{E}{\models} \Omega :: (\Gamma\ ;\ \Delta)$*. Either* Ω *is poised, or it can take a step, i.e.,* $\Omega \mapsto \Omega'$*, or some process in* Ω *is blocked along* a_{S} *for some shared channel* a_{S} *and there is a process* $\mathsf{proc}(a_{\mathsf{L}}, w, P) \in \Omega$*.*

Proof. Either $\Omega = \Omega_1, \mathsf{proc}(c_m, w, P)$ or $\Omega = \Omega_1, \mathsf{msg}(c_m, w, M)$. In either case, either $\Omega_1 \mapsto \Omega_1'$, in which case we are done. Or there is a process in Ω_1 blocked along a_{S} in which case, we are also done. Hence, in the final case, we get Ω_1 is poised and there is no process in Ω_1 blocked along a_{S}. Now, we case analyze on the structure of the process or message. We start with processes.

- Case $(\{\}E_{mn})$: In each case, the process spontaneously steps by spawning another process.
- Case $(\mathsf{fwd}^+ : \mathsf{proc}(c_m, w, c_m \overset{+}{\leftarrow} d_k))$:

$$\cdot\ ;\ \Gamma\ ;\ (d_k : A) \vdash^{0} c_m \overset{+}{\leftarrow} d_k :: (c_m : A)$$

  Since Ω_1 is poised, there must be a message in Ω_1 offering along $d_m : A$. We use Lemma 5 to move the message just left of the process, and then apply the fwd^+ rule. Hence, Ω can step.
- Case $(\mathsf{fwd}^- : \mathsf{proc}(c_m, w, c_m \overset{-}{\leftarrow} d_m))$: This process is poised, hence Ω is poised.
- Case $(\oplus R : \mathsf{proc}(c_m, w, c_m.k\ ;\ P))$: Ω steps using $\oplus C_s$ rule.
- Case $(\oplus L : \mathsf{proc}(d_k, w, \mathsf{case}\ c_m\ (l \Rightarrow Q_l)_{l \in L}))$:

$$\cdot\ ;\ \Gamma\ ;\ (c_m : \oplus\{l : A_l\}_{l \in L}) \vdash^{q} \mathsf{case}\ c_m\ (l \Rightarrow Q_l)_{l \in L} :: (d_k : C)$$

Since Ω_1 is poised, there must be a message in Ω_1 offering along $c_m : \oplus\{l : A_l\}_{l \in L}$. We use Lemma 5 to move the message just left of the process, and then apply the $\oplus C_r$ rule. Hence, Ω can step.

- Case ($\multimap R$: $\mathsf{proc}(c_m, w, x_n \leftarrow \mathsf{recv}\ c_m\ ;\ P)$) : This process is poised, hence Ω is poised.
- Case ($\multimap L$: $\mathsf{proc}(c_m, w, \mathsf{send}\ c_m\ e_n\ ;\ Q)$) : Ω steps using $\multimap C_s$ rule.
- Case ($\uparrow_{\mathsf{L}}^{\mathsf{S}} R$: $\mathsf{proc}(c_{\mathsf{S}}, c_{\mathsf{L}} \leftarrow \mathsf{accept}\ c_{\mathsf{S}}\ ;\ P)$) : This process is poised, hence Ω is poised.
- Case ($\uparrow_{\mathsf{L}}^{\mathsf{S}} L$: $\mathsf{proc}(c_m, w, a_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ;\ Q)$) :

$$\cdot\ ;\ \Gamma, (a_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})\ ;\ \Delta \vDash^{q} a_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ;\ Q :: (c_m : C)$$

There must be some process in Ω_1 that offers on a_{S}. Either this process is in shared mode or linear mode. If the process is in shared mode, and since Ω_1 is poised, the process must be $\mathsf{proc}(a_{\mathsf{S}}, w', a_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}}\ ;\ P)$ in which case, we can use Lemma 7 to move the two processes next to each other and Ω can step using $\uparrow_{\mathsf{L}}^{\mathsf{S}} C$ rule. Or the process is in linear mode in which case the acquiring process is blocked and there is some $\mathsf{proc}(a_{\mathsf{L}}, w', P)$ in Ω.

- Case ($\downarrow_{\mathsf{L}}^{\mathsf{S}} R$: $\mathsf{proc}(c_{\mathsf{S}}, c_{\mathsf{L}} \leftarrow \mathsf{detach}\ c_{\mathsf{S}}\ ;\ P)$: This process is poised, hence Ω is poised.
- Case ($\downarrow_{\mathsf{L}}^{\mathsf{S}} L$: $\mathsf{proc}(c_{\mathsf{T}}, w, a_{\mathsf{L}} \leftarrow \mathsf{release}\ a_{\mathsf{S}}\ ;\ Q)$) :

$$\cdot\ ;\ \Gamma\ ;\ \Delta, (a_{\mathsf{L}} : \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}}) \vDash^{q} a_{\mathsf{L}} \leftarrow \mathsf{release}\ a_{\mathsf{S}}\ ;\ Q :: (c_m : C)$$

There must be some process in Ω_1 that offers along a_{L}. Since Ω_1 is poised, this process must be $\mathsf{proc}(a_{\mathsf{L}}, w', a_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}}\ ;\ P)$ in which case we use Lemma 8 to move the releasing process next to the detaching process and Ω can step using $\downarrow_{\mathsf{L}}^{\mathsf{S}} C$ rule.

That completes the cases where the last predicate is a process. Now, we consider the cases where the last predicate is a message.

- Case (fwd^- : $\mathsf{msg}(e_k, w, M(c_m))$) : There must be some process in Ω_1 that offers along d_m. Since Ω_1 is poised, if there is a forwarding process $\mathsf{proc}(c_m, w', c_m \xleftarrow{-} d_m)$ in Ω_1, then Ω steps using fwd^- rule. Hence, in the following cases, we assume that the offering process used by the message will not be a forwarding process.
- Case ($\oplus$: $\mathsf{msg}(c_m, c_m.k\ ;\ M)$) : This message is poised, hence Ω is poised.
- Case ($\multimap$: $\mathsf{msg}(c_m^+, \mathsf{send}\ c_m\ e_{\mathsf{R}}\ ;\ c_m^+ \leftarrow c_m)$) : There must be a process in Ω_1 that offers along c_m. Since Ω_1 is poised, this process must be $\mathsf{proc}(c_m, x_n \leftarrow \mathsf{recv}\ c_m\ ;\ P)$. We move the process to the left of this message using Lemma 6. And then, Ω can step using $\multimap C_r$ rule.

□