Interconnection Networks in Session-based Logical Processes

Bernardo Toninho
Imperial College London
b.toninho@imperial.ac.uk

Nobuko Yoshida
Imperial College London
n.yoshida@imperial.ac.uk

Abstract

In multiparty session types, interconnection networks identify which roles in a session engage in direct communication. If role p is connected to role q, then p exchanges a message with q. In a session-based interpretation of classical linear logic (CLL), this corresponds to the composition, or cut, of dual propositions. This paper shows that well-formed interactions represented in a session-based interpretation of CLL form strictly less expressive interconnection networks than those specified in a multiparty session calculus. To achieve this, we introduce a new compositional synthesis property, dubbed partial multiparty compatibility (PMC), enabling us to build a global type denoting the interactions obtained by iterated composition of well-typed CLL processes. We show that the CLL composition rule induces PMC global types without circular interconnections between three participants. PMC is then used to define a new CLL multicut rule which can form general multiparty interconnections, preserving the deadlock-freedom property of CLL.

Keywords Session Types, Classical Linear Logic, the Pi-Calculus, Multiparty Session Types and Synthesis.

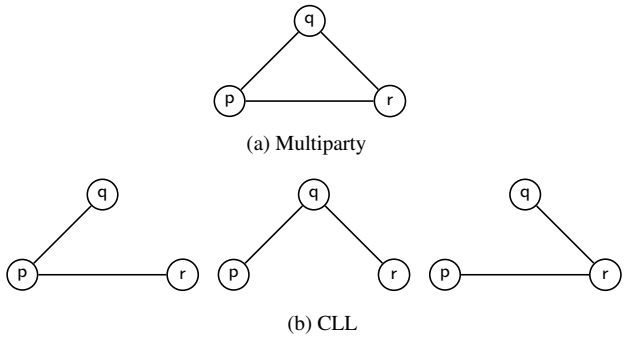
1. Introduction

The discovery of linear logic [12] and the early studies of its connections with concurrent processes [1–3] can be seen as the origin of a Curry-Howard correspondence for linear logic with typed interactive behaviours, which have led to the more recent developments connecting linear logic and (binary) session types [5]. The understanding of linear logic propositions as session types, proofs as concurrent processes and proof simplification as communication has produced new logically-motivated techniques for reasoning about concurrent processes [16, 18, 21, 24, 26, 27], while also offering guidance to clean design and implementations for programming languages based on communication with strong safety guarantees such as *protocol fidelity* and *deadlock freedom* [19, 22, 25].

In linear logic-based session frameworks, processes communicate through a session channel that connects exactly two distinct subsystems typed by dual propositions: when one party sends, the other receives; when one party offers a selection, the other chooses. Sessions may be dynamically exchanged via a session name or created by invocation of replicated servers. A combination of these features enables the modelling of complex behaviours between an arbitrary number of concurrent threads.

However, the linear typing discipline induced by linear logic enforces very strong separation properties on the network topology of communicating processes, to the extent that composition identifies a process by a single of its used *linear* channels, requiring all other linear channels in the composed processes to be disjoint and implemented by strictly *separate* processes. It is from these strong separation properties that deadlock-freedom arises in a rather simple typing discipline, at the cost of disallowing network topologies with more intricate connectedness properties.

This paper provides a fresh look at session-based logical processes, based on concepts originating in *multiparty session types*. Motivated by an industry need [23] to specify protocols with more than two interconnected, interacting parties, the multiparty session types framework [14] develops a methodology where types explicitly describe interconnection networks between *many* communicating processes. The basic idea of the framework consists of taking a *global type* (i.e. a global description of the multiparty interaction), from which we *generate* (or project) *local types* for each communicating party (specifying how a given party interacts with all others) and check that the process implementation of each party adheres to its local type. Once each process is typechecked locally, a group of many participants in a session can interact without deadlock, following a global type.



(a) Multiparty

(b) CLL

Figure 1: Interconnection networks of (1)

In a multiparty session type, interconnection networks identify which roles in a session engage in direct communication. If participant p is connected to another participant q, then p will exchange a message with q. Consider the following 3-party interaction specified as a global type G:

$$G = \mathsf{p} \to \mathsf{q} : (\mathsf{nat}).\mathsf{p} \to \mathsf{r} : (\mathsf{bool}).\mathsf{r} \to \mathsf{q} : (\mathsf{char}).\mathsf{end} \qquad (1)$$

The type G specifies an interaction where role p sends to roles q and r a natural number and a boolean, respectively, followed by r sending to q a string, inducing the interconnection network depicted in Fig. 1a, realisable in a system where each role is implemented by a separate process.

However, the topology of Fig. 1a is *not* realisable in linear logic-based session frameworks, only those of Fig. 1b. We sketch three processes with the behaviour ascribed by G, each implementing one role in the multiparty session which are to then be composed,

$$P \vdash \mathsf{pq}{:}A, \mathsf{pr}{:}B \quad Q \vdash \mathsf{pq}{:}A^{\perp}, \mathsf{qr}{:}C \quad R \vdash \mathsf{pr}{:}B^{\perp}, \mathsf{qr}{:}C^{\perp} \qquad (2)$$

where P is the implementation of p with pq for communication between p and q, and pr for communication between p and r; Q implements role q using channel pq (dually to P) and qr for communication with r, and so on.

While each process is individually typable in linear logic-based session type systems, no 3-way composition is typable. In essence, the multiparty-based approach of identifying processes with roles leads to richer interconnection topologies than the identification of processes with channels during composition, at the cost of requiring global types and projection to ensure deadlock-freedom.

In this work, we make precise the informal argument sketched above, first showing a negative result on the network intercon-

nectability in a session-based interpretation of Classical Linear Logic (CLL), translating it into a multiparty session calculus (MP), which then informs a *conservative* extension of CLL that allows for richer connection topologies. Though a vast amount of works on expressiveness of process calculi have been developed as *encodability* in the literature, our approach differs insofar as our translations focus on a *preservation of connectability* of participants (threads) and typability, without using semantic equivalences (e.g. bisimulations). We show that well-formed interactions in CLL form strictly less expressive interconnection networks than those of MP.

To demonstrate this result, we infer session flows from CLL processes and use them to synthesise a global type. For this, we introduce a new *compositional synthesis property*, dubbed partial multiparty compatibility (PMC), which enables us to build a global type that represents the interactions obtained by iterated composition of CLL processes. We show that the CLL composition rule induces PMC global types without circular interconnections between more than three participants (excluding the topology of Fig. 1a).

The translations from CLL into MP give two positive results: First, CLL can be used to guarantee deadlock-freedom and termination of a (restricted) class of MP-processes with delegation and shared channels, which is *not* normally guaranteed by MP-typing systems. Secondly, PMC enables us to define a new *multicut* rule in CLL which can form general multiparty interconnections, but preserves the deadlock-freedom property of CLL processes. In contrast with [8], our framework does not require any modifications to the syntax of propositions or CLL processes, only requiring the introduction of a multicut rule that is guided by PMC. We also do not require projection from global types, thus preserving an essence of the logical interpretations of session types.

We summarise the contributions of this work:

- A type preserving translation of typed interactions in a session-based interpretation of classical linear logic without channel passing and exponentials (basic CLL) into a *single* multiparty session represented in MP (§ 3).
- A compositional synthesis property, partial multiparty compatibility (PMC) (§ 4), which we use to show that the interconnection networks of CLL processes are less expressive than those of a single multiparty session in MP (§ 5).
- A systematic extension of these results to full CLL (with channel passing and exponentials), showing that full CLL may be encoded into several sessions in MP, still with no interconnection circularity (§ 6,§ 7). As a consequence we obtain a deadlock-free typing discipline for *interleaved* multiparty sessions.
- An extension of CLL with multicut, based on PMC and the developed translations, allowing us to type a range of untypable examples in previous works on MP and its relationship with linear logic [8], still ensuring deadlock-freedom and without modifying the types or syntax of CLL (§ 8).

We discuss related work in § 9. The appendix § A lists omitted definitions and proofs.

2. Processes, types and typing systems

This section introduces the two process calculi and typing disciplines used in our development: the binary session calculus CLL (§ 2.1) which is typed using (a fragment of) the session type interpretation of classical linear logic [7, 27]; and the multiparty session calculus MP [9, 14]. To obtain our first negative result (§ 5), we remove channel passing and replication, addressing these features in § 6 and 7, respectively. Both calculi have a *synchronous* semantics.

$$\frac{}{\mathbf{0} \vdash_{\mathsf{CL}} a{:}\mathbf{1}}\ (\mathbf{1}) \qquad \frac{P \vdash_{\mathsf{CL}} \Delta}{P \vdash_{\mathsf{CL}} a{:}\bot, \Delta}\ (\bot) \qquad \frac{\Psi \vdash M : \tau \quad P \vdash_{\mathsf{CL}} a{:}A, \Delta; \Psi}{a\langle M\rangle.P \vdash_{\mathsf{CL}} a{:}\tau \otimes A, \Delta; \Psi}\ (\otimes)$$

$$\frac{P \vdash_{\mathsf{CL}} a{:}A, \Delta; x{:}\tau, \Psi}{a(x).P \vdash_{\mathsf{CL}} a{:}\tau \parr A, \Delta; \Psi}\ (\parr) \qquad \frac{P \vdash_{\mathsf{CL}} a{:}A_j, \Delta \quad j \in I}{a.l_j; P \vdash_{\mathsf{CL}} a{:} \oplus \{l_i : A_i\}_{i\in I}, \Delta}\ (\oplus)$$

$$(\&)\ \frac{P_1 \vdash_{\mathsf{CL}} a{:}A_1, \Delta \quad \ldots \quad P_n \vdash_{\mathsf{CL}} a{:}A_n, \Delta}{a.\mathsf{case}\{l_i : P_i\}_{i\in I} \vdash_{\mathsf{CL}} a{:}\ \& \{l_i : A_i\}_{i\in I}, \Delta}$$

$$\frac{P \vdash_{\mathsf{CL}} \Delta, a{:}A \quad Q \vdash_{\mathsf{CL}} \Delta', a{:}A^{\bot}}{(\boldsymbol{\nu} a)(P \mid Q) \vdash_{\mathsf{CL}} \Delta, \Delta'}\ (\mathsf{cut}) \qquad \frac{}{[a \leftrightarrow b] \vdash_{\mathsf{CL}} a{:}A, b{:}A^{\bot}}\ (\mathsf{id})$$

Figure 2: CLL Typing Rules

2.1 Classical Linear Logic as binary session types

We give a brief summary of the interpretation of classical linear logic as sessions, consisting of a variant of those of [7, 27]. The syntax of binary session processes P_L, Q_L is given below. Session channels are ranged over with a, b, c. Basic data values (such as strings and integers) are ranged over with M, N:

$P_L, Q_L ::= a\langle M\rangle.P_L$	$a(x).P_L$		Value send and receive
$\mid a.l; P_L$	$a.\mathsf{case}\{l_i : P_L\}_{i\in I}$		Selection and branching
$\mid \mathbf{0}$	$(P_L \mid Q_L)$		Inaction and parallel
$\mid (\boldsymbol{\nu} a)P_L$	$[a \leftrightarrow b]$		Hiding and forwarding

We consider a synchronous process calculus with basic value communication, branching and selection. The forwarder $[a \leftrightarrow b]$ identifies the channels a and b as dual behaviours, implemented in the operational semantics as a renaming operation. The remaining constructs are standard. We write $fn(P_L)$ and $bn(P_L)$ for the free and bound channels of P_L, respectively. We often omit $\mathbf{0}$ and the $\cdot_L$ subscript for readability.

The syntax of (logical) binary session types is given below (where τ ranges over basic data types):

$$A, B \quad ::= \quad \tau \otimes A \quad \mid \quad \tau \parr A \quad \mid \quad \mathbf{1} \mid \bot$$
$$\mid \quad \oplus\{l_i : A_i\}_{i\in I} \quad \mid \quad \&\{l_i : A_i\}_{i\in I}$$

Following [7, 27], $\otimes$ corresponds to output followed by behaviour A, $\parr$ to input followed by A, $\oplus$ and $\&$ to selection and branching behaviours. We note that input and output are restricted to just value communication. The *dual* of A, written $A^{\bot}$, is defined as:

$$\mathbf{1}^{\bot} \triangleq \bot \quad \bot^{\bot} \triangleq \mathbf{1} \quad (\tau \otimes A)^{\bot} \triangleq \tau \parr A^{\bot} \quad (\tau \parr A)^{\bot} \triangleq \tau \otimes A^{\bot}$$
$$(\oplus\{l_i{:}A_i\}_{i\in I})^{\bot} \triangleq \&\{l_i{:}A_i^{\bot}\}_{i\in I} \quad (\&\{l_i{:}A_i\}_{i\in I})^{\bot} \triangleq \oplus\{l_i{:}A_i^{\bot}\}_{i\in I}$$

We define the typing system CLL in Fig. 2, assigning the usage of channels in P_L processes to types A, B. The main typing judgment is written $P_L \vdash_{\mathsf{CL}} \Delta; \Psi$, defined *up to structural congruence* (see § A.1),where Δ is a un-ordered set of hypotheses of the form $a{:}A$, where a stands for a free session channel in P_L and A is a binary session type; Ψ is an un-ordered set of hypotheses of the form $x{:}\tau$, which refer to the values that are sent and received in processes. Throughout the paper, since values are somewhat orthogonal to our development, we often omit the Ψ context region for the sake of readability. The typing judgment thus states that process P_L uses channels according to the session discipline ascribed by Δ. We assume all channel names in Δ are distinct. We write Δ, Δ' for the union of Δ and Δ', only defined when channels in Δ and Δ' are distinct (Appendix A.1 lists explanations of each rule).

The (id) rule gives rise to the following identity process, which is generated inductively on a given type. The identity process is used in the translation from CLL to MP later.

Definition 2.1 (Identity Process)**.** Let A be a session type. We define the process $\mathsf{id}_A(a, b)$, denoting the copycat between channels a and b at type A, typable as: $\mathsf{id}_A(a, b) \vdash_{\mathsf{CL}} a{:}A, b{:}A^{\bot}$, by induction

on the structure of A as follows:

$$\begin{array}{rcl}
\mathsf{id}_{\tau\otimes A}(a,b) & \triangleq & b(n).a\langle n\rangle.\mathsf{id}_A(a,b)\\
\mathsf{id}_{\tau\parr A}(a,b) & \triangleq & a(n).b\langle n\rangle.\mathsf{id}_A(a,b)\\
\mathsf{id}_{\oplus\{l_i:A_i\}_{i\in I}}(a,b) & \triangleq & b.\mathsf{case}\{l_i : (a.l_i;\mathsf{id}_{A_i}(a,b))\}_{i\in I}\\
\mathsf{id}_{\&\{l_i:A_i\}_{i\in I}}(a,b) & \triangleq & a.\mathsf{case}\{l_i : (b.l_i;\mathsf{id}_{A_i}(a,b))\}_{i\in I}\\
\mathsf{id}_{\mathbf{1}}(a,b) \triangleq \mathbf{0} & & \mathsf{id}_{\bot}(a,b) \triangleq \mathbf{0} \qquad \diamond
\end{array}$$

The reduction semantics for CLL is defined by $P_L \to Q_L$ up to structural congruence $\equiv$ (we omit the homomorphic cases):

$$\begin{array}{rcll}
a\langle M\rangle.P \mid a(x).Q & \to & P \mid Q\{M/x\} & \\
a.l_j; P \mid a.\mathsf{case}\{l_i{:}Q_i\}_{i\in I} & \to & P \mid Q_j & (j\in I)\\
(\boldsymbol{\nu} a)([a\leftrightarrow b] \mid P) & \to & P\{b/a\} & (b\notin fn(P))
\end{array}$$

Definition 2.2 (Deadlock-freedom)**.** A closed process P is deadlock-free if for all P' such that $P \to^* P' \not\to$, $P' \equiv \mathbf{0}$. $\diamond$

Proposition 2.1 (Deadlock-freedom in CLL [7, 27])**.** *Suppose $P \vdash_{\mathsf{CL}} \Delta$, where Δ is either empty or only contains $\mathbf{1}$ or $\bot$. P is deadlock-free.*

2.2 Multiparty session calculus

We introduce the MP calculus of multiparty sessions, where processes P, Q use channels annotated by roles of the multiparty session in which they are used.

$$\begin{array}{rcll}
P,Q & ::= & c[\mathsf{p}]\langle M\rangle; P \mid c[\mathsf{p}](x); P & \text{Value send and receive}\\
 & \mid & c[\mathsf{p}] \lhd l; P \mid c[\mathsf{p}] \rhd \{l_i{:}P_i\}_{i\in I} & \text{Selection and branching}\\
 & \mid & \mathbf{0} \mid (P \mid Q) \mid (\boldsymbol{\nu} s)P & \text{Inaction, parallel, hiding}\\
c & ::= & y \mid s[\mathsf{p}] &
\end{array}$$

Role names are identified by $\mathsf{p}, \mathsf{q}, \mathsf{r}$. Channels are ranged over by s, t. As before, we write $fn(P)$ and $bn(P)$ for the free and bound channels of P, respectively.

Role annotated channels $s[\mathsf{p}]$ in MP are assigned local types, ranged over by S, T, denoting the behaviour that role p performs on the given channel. Local types are defined by the grammar:

$$\begin{array}{rcl}
S,T & ::= & \mathsf{p}{\uparrow}(\tau);T \mid \mathsf{p}{\downarrow}(\tau);T\\
 & \mid & \oplus\mathsf{p}\{\mathsf{l}_i{:}T_i\}_{i\in I} \mid \&\mathsf{p}\{\mathsf{l}_i{:}T_i\}_{i\in I} \mid \mathsf{end}
\end{array}$$

The local types $\mathsf{p}{\uparrow}(\tau);T$ and $\mathsf{p}{\downarrow}(\tau);T$ denote output and input to role p of a value of type τ, followed by behaviour T, respectively. Types $\oplus\mathsf{p}\{\mathsf{l}_i{:}T_i\}_{i\in I}$ and $\&\mathsf{p}\{\mathsf{l}_i{:}T_i\}_{i\in I}$ denote the emission (resp. reception) of a label l_i to (resp. from) role p, followed by behaviour T_i. end denotes no further behaviour.

We define the set of roles of local type T, denoted by $\mathsf{roles}(T)$, as the set of all roles occurring in type T. We define the typing system MP in Fig. 3, assigning the usage of role-annotated channels to local types. The judgment $P \vdash_{\mathsf{MP}} \Delta; \Psi$, where Δ is an unordered set of hypotheses of the form $s[\mathsf{p}]{:}T$, denotes that P uses its session channels according to the specification of Δ, with data variables tracked in Ψ (we often omit Ψ for presentation purposes). As before, we assume all bindings in Δ are disjoint and write Δ, Δ' for the union of Δ and Δ' only defined when both contexts are disjoint and we use a judgment $\Psi \vdash M : \tau$ for data values.

The type system relies on a notion of *coherence*, written $\mathsf{co}(\Delta)$, which ensures that Δ contains a local type for each role involved in interactions in Δ. Moreover, coherence ensures that local types of interacting roles contain the necessary dual communication actions. (see § A.2 for a definition of coherence).

We define the dual of a binary type T, written $\overline{T}$ [13] as: $\overline{\uparrow(\tau);T} \triangleq \downarrow(\tau);\overline{T}$, $\overline{\oplus\{l_i{:}T_i\}_{i\in I}} \triangleq \&\{l_i{:}\overline{T_i}\}_{i\in I}$ and $\overline{\mathsf{end}} \triangleq \mathsf{end}$; and the dual rules for $\downarrow$ and $\&$.

The reduction semantics for MP processes are given below (omitting the structural congruence closure and homomorphic rules). They are fundamentally identical to the reduction rules of § 2.1, but where we require not just the session channel to match

$$(\mathsf{vsend})\frac{\Psi \vdash M : \tau \quad P \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}T; \Psi}{s[\mathsf{p}][\mathsf{q}]\langle M\rangle; P \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}\mathsf{q}{\uparrow}(\tau);T; \Psi}$$

$$(\mathsf{vrecv})\frac{P \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}T; x{:}\tau, \Psi}{s[\mathsf{p}][\mathsf{q}](x); P \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}\mathsf{q}{\downarrow}(\tau);T; \Psi}$$

$$(\mathsf{sel})\frac{P \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}T_j \quad j\in I}{s[\mathsf{p}][\mathsf{q}] \lhd l_j; P \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}\oplus\mathsf{q}\{l_i : T_i\}_{i\in I}}$$

$$(\mathsf{branch})\frac{P_1 \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}T_1 \quad \ldots \quad P_n \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}T_n}{s[\mathsf{p}][\mathsf{q}] \rhd \{l_i{:}P_i\}_{i\in I} \vdash_{\mathsf{MP}} \Delta, s[\mathsf{p}]{:}\,\&\,\mathsf{q}\{l_i : T_i\}_{i\in I}}$$

$$(\mathsf{end})\frac{\Delta\ \mathsf{end\ only}}{\mathbf{0} \vdash_{\mathsf{MP}} \Delta} \qquad (\mathsf{comp})\frac{P \vdash_{\mathsf{MP}} \Delta \quad Q \vdash_{\mathsf{MP}} \Delta'}{P \mid Q \vdash_{\mathsf{MP}} \Delta, \Delta'}$$

$$(\mathsf{close})\frac{P \vdash_{\mathsf{MP}} \Delta, s[1] : T_1, \ldots, s[\mathsf{n}] : T_n \quad \mathsf{co}(s[1]{:}T_1, \ldots, s[\mathsf{n}]{:}T_n)}{(\boldsymbol{\nu} s)P \vdash_{\mathsf{MP}} \Delta}$$

Figure 3: MP Typing Rules

but also the role assignment to be consistent:

$$\begin{array}{rcl}
s[\mathsf{p}][\mathsf{q}]\langle M\rangle; P \mid s[\mathsf{q}][\mathsf{p}](x); Q & \to & P \mid Q\{M/x\}\\
s[\mathsf{p}][\mathsf{q}] \lhd l_j; P \mid s[\mathsf{q}][\mathsf{p}] \rhd \{l_i{:}Q_i\}_{i\in I} & \to & P \mid Q_j \quad (j\in I)
\end{array}$$

We highlight that, in contrast to the CLL system, the typing system MP alone does *not* ensure deadlock-freedom.

Fact 2.1 (Deadlock in MP)**.** *There exists a dead-locked process P that is typable by the rules of Figure 3, i.e. $P \vdash_{\mathsf{MP}} \emptyset$, does* not *imply that P is deadlock-free.*

Proof. Take $P = s[\mathsf{p}][\mathsf{r}](x); s[\mathsf{p}][\mathsf{q}]\langle 7\rangle, Q = s[\mathsf{q}][\mathsf{p}](x); s[\mathsf{q}][\mathsf{r}]\langle \mathtt{tt}\rangle$, $R = s[\mathsf{r}][\mathsf{q}](x); s[\mathsf{r}][\mathsf{p}]\langle$"$a$"$\rangle$. $(\boldsymbol{\nu} s)(P \mid Q \mid R) \vdash_{\mathsf{MP}} \emptyset$, but $P \mid Q \mid R$ is deadlocked. □

3. Relating the CLL and MP systems

In this section we develop one of our main contributions: the connection between the CLL and MP typing systems. To motivate our approach, consider the following CLL typable processes:

$$\begin{array}{rcl}
P_L & \triangleq & a\langle 7\rangle.b(x).a\langle$"$hello$"$\rangle.\mathbf{0} \vdash_{\mathsf{CL}} a{:}\mathsf{nat} \otimes \mathsf{str} \otimes \mathbf{1}, b{:}\mathsf{nat} \parr \bot\\
P'_L & \triangleq & b(x).a\langle 7\rangle.a\langle$"$hello$"$\rangle.\mathbf{0} \vdash_{\mathsf{CL}} a{:}\mathsf{nat} \otimes \mathsf{str} \otimes \mathbf{1}, b{:}\mathsf{nat} \parr \bot
\end{array}$$

Both P_L and P'_L are typable in the same context, however P_L first outputs on a, then inputs on b and then outputs on a again, whereas P'_L flips the order of the first two actions. By the nature of process composition in CLL, both processes can be safely composed with $R^1_L \vdash_{\mathsf{CL}} a{:}\mathsf{nat} \parr \mathsf{str} \parr \bot$ and $R^2_L \vdash_{\mathsf{CL}} b{:}\mathsf{nat} \otimes \mathbf{1}$. We also observe that, since both P_L and P'_L are typable in the same context, the typing discipline of CLL does not capture *cross-channel sequential dependencies* (i.e. it cannot distinguish the ordering of actions on different channels within a process at the level of types).

We now consider a mapping from CLL to MP. The following processes Q and Q' are idealised translations of P_L and P'_L. Session $s[\mathsf{q}][\mathsf{p}]$ denotes channel s located at participant p which will interact with another participant q:

$$\begin{array}{rcl}
Q & \triangleq & s[\mathsf{q}][\mathsf{p}]\langle 7\rangle; s[\mathsf{q}][\mathsf{r}](x); s[\mathsf{q}][\mathsf{p}]\langle$"$hello$"$\rangle; \mathbf{0}\\
Q' & \triangleq & s[\mathsf{q}][\mathsf{r}](x); s[\mathsf{q}][\mathsf{p}]\langle 7\rangle; s[\mathsf{q}][\mathsf{p}]\langle$"$hello$"$\rangle; \mathbf{0}
\end{array}$$

Morally, the processes P_L and Q above are similar, insofar as both send 7 to a destination (resp. a and role p), followed by an input (resp. on b and from r), followed by an output of "hello" to the initial destination. A similar argument can be made for P'_L and Q'. However, despite P_L and P'_L having the same types, we have that $Q \vdash_{\mathsf{MP}} s[\mathsf{q}]{:}\mathsf{p}{\uparrow}(\mathsf{nat}); \mathsf{r}{\downarrow}(\mathsf{nat}); \mathsf{p}{\uparrow}(\mathsf{str}); \mathsf{end}$, but $Q' \vdash_{\mathsf{MP}} s[\mathsf{q}]{:}\mathsf{r}{\downarrow}(\mathsf{nat}); \mathsf{p}{\uparrow}(\mathsf{nat}); \mathsf{p}{\uparrow}(\mathsf{str}); \mathsf{end}$. By refining channels with role annotations within a multiparty session, MP can distinguish orderings of actions performed on different session *sub-channels*.

Thus, our goal is to find a precise way to not only systematically map process P_L to process Q (and P'_L to Q'), but also generate the corresponding local typing environment in a type preserving way. To relate CLL with MP processes we define a mapping from *session channels* in CLL to role-annotated *session channels* in MP, such that given a cut-free process in CLL (intuitively, a single thread), we consistently map its channels to role-annotated channels in the MP system, capturing the cross-channel sequential dependencies that are not codified at the level of CLL *session types*.

Definition 3.1 (Channel to Role-Indexed Channel Mapping)**.** Let $P_L \vdash_{\mathsf{CL}} \Delta$ such that the typing derivation does not use the cut rule. We define a channel name to role-indexed channel name mapping σ such that for all $a, b \in fn(P_L)$, if $a \neq b$, $\sigma(a) = s[\mathsf{p}][\mathsf{q}]$ and $\sigma(b) = s[\mathsf{p}'][\mathsf{q}']$, then $\mathsf{p} = \mathsf{p}' \wedge \mathsf{q} \neq \mathsf{q}'$. We discard reflexive role assignments of the form $\mathsf{s}[\mathsf{p}][\mathsf{p}]$. We write $P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta$ to denote such a mapping and $\mathsf{c}_\sigma(a)$, $\mathsf{p}_\sigma(a)$ and $\mathsf{d}_\sigma(a)$ to denote the channel, first (principal) and second (destination) roles in the image of a in σ.

A mapping according to Def. 3.1 identifies a single-thread process of CLL with a single role implementation in MP, such that all its channels are mapped to same multiparty session channel s and to the same principal role p, but to different destination roles. Formally, we note that $\forall a, b \in fn(P_L)$, $\mathsf{c}_\sigma(a) = \mathsf{c}_\sigma(b)$ and $\mathsf{p}_\sigma(a) = \mathsf{p}_\sigma(b)$. In the remainder of this paper, we exclude all reflexive role assignments.

Let $P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta$. We write $\sigma(P_L)$ for the process obtained by renaming each free name a in P_L with $\sigma(a)$, where actions in P_L are mapped to their corresponding actions in multiparty session processes and the forwarding construct is mapped to the renamed identity process (Def. 2.1).

Having constructed a syntactic mapping from CLL to MP processes, we must now find a way to generate the appropriate local typings for processes in the image of the translation.

Definition 3.2 (Local Typing Generation)**.** Let $P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta$, Let c_σ and p_σ to refer to the unique channel and principal role of σ. We inductively generate a local type T such that $\sigma(P_L) \vdash_{\mathsf{MP}} \mathsf{c}_\sigma[\mathsf{p}_\sigma]{:}T$ by induction on the structure of typing of P_L, written $[\![P_L]\!]_\sigma$:

$$
\begin{array}{rcl}
[\![\mathbf{0}]\!]_\sigma & \triangleq & \mathsf{end} \\
[\![a\langle M\rangle.P]\!]_\sigma & \triangleq & \mathsf{d}_\sigma(a)\!\uparrow\!(\tau); [\![P]\!]_\sigma \quad \text{with } M : \tau \\
[\![a(y).P]\!]_\sigma & \triangleq & \mathsf{d}_\sigma(a)\!\downarrow\!(\tau); [\![P]\!]_\sigma \quad \text{with } y : \tau \\
[\![a.l_j; P]\!]_\sigma & \triangleq & \oplus \mathsf{d}_\sigma(a)\{l_j : [\![P]\!]_\sigma\} \\
[\![a.\mathsf{case}\{l_i : P_i\}_{i\in I}]\!]_\sigma & \triangleq & \& \mathsf{d}_\sigma(a)\{l_i : [\![P_i]\!]_\sigma\}_{i\in I} \\
[\![[a \leftrightarrow b]]\!]_\sigma & \triangleq & [\![\mathsf{id}_A(a,b)]\!]_\sigma \text{ with } \Delta = a{:}A, b{:}A^{\perp}
\end{array}
$$

Given the definition of a type preserving mapping of *cut-free* processes in CLL to single-thread processes in MP, we account for process composition. We define the judgement $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma$ such that P is an n-ary composition of processes, Γ is a typing context assigning role-indexed channel names to *local* types, Δ is a typing context assigning channel names to *session* types, σ is a free name to role-indexed channel mapping (Def. 3.1) and ρ is a mapping from bound names to role-indexed channels.

Definition 3.3 (Process Composition)**.**

$$
\frac{P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta}{P_L \Vdash^{\sigma}_{\emptyset} \Delta; \mathsf{c}_\sigma[\mathsf{p}_\sigma]{:}[\![P_L]\!]_\sigma}\ (\mathsf{thread})
$$

$$
\frac{P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta, a{:}A \quad Q_L \Vdash^{\sigma'}_{\rho} \Delta', a{:}A^{\perp}; \Gamma \quad \star}{(\boldsymbol{\nu} a)(P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{a\}}_{\rho'} \Delta, \Delta'; \Gamma, \mathsf{c}_\sigma[\mathsf{p}_\sigma]{:}[\![P_L]\!]_\sigma}\ (\mathsf{comp})
$$

$$
\star \begin{cases}
\rho' = \rho \cup (a, \mathsf{c}_\sigma[\mathsf{p}_\sigma(a)][\mathsf{d}_\sigma(a)]) \\
\mathsf{c}_\sigma = \mathsf{c}_{\sigma'} \quad \mathsf{p}_\sigma(a) = \mathsf{d}_{\sigma'}(a) \quad \mathsf{d}_\sigma(a) = \mathsf{p}_{\sigma'}(a) \\
\forall z \in \Delta, y \in \Delta'. \mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \mathsf{d}_\sigma(z), \mathsf{d}_{\sigma'}(y) \notin \rho
\end{cases}
$$

Rule (comp) above defines the composition of a single-thread CLL process with an n-ary composition of CLL processes which can be mapped to MP typed processes. The rule ensures that the resulting mappings are consistent and that the resulting process is well-formed in both CLL and MP. Composition is only defined when both σ and σ' map a to the same multiparty session channel, where the destination role in $\sigma(a)$ matches the principal role in $\sigma'(a)$, and vice-versa. Moreover, since roles are implemented by single threads and composition in CLL only connects a single channel, we enforce that channels in Δ and channels in Δ' cannot have the same destination role (otherwise we would have the same role split across different threads). Finally, role assignments for free names cannot capture previously composed role assignments.

We write $\rho(P)$ for the renaming of bound names in P generated by the following (we omit the congruence cases for conciseness):

$$
\rho((\nu x)(P \mid Q)) = \rho'(P\{\rho(x)/x\}) \mid \rho'(Q\{\overline{\rho(x)}/x\})
$$

where $\rho' = \rho \setminus \{x\}$ and $\overline{\rho(x)}$ denotes $s[\mathsf{q}][\mathsf{p}]$ if $\rho(x) = s[\mathsf{p}][\mathsf{q}]$. We now show that the combination of the type preserving translation of CLL to MP for cut-free processes combined with our composition rule preserves typing in MP.

Proposition 3.1. *If* $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma$*, then* $\rho(\sigma(P)) \vdash_{\mathsf{MP}} \Gamma$.

Example 3.1 (Four Threads)**.** We show how to fully translate the CLL process from the beginning of this section and how to form a system through composition:

$$
\begin{array}{rcl}
P_L & \triangleq & a\langle 7\rangle.b(x).a\langle \text{"}hello\text{"}\rangle.\mathbf{0} \vdash_{\mathsf{CL}} a{:}\mathsf{nat} \otimes \mathsf{str} \otimes \mathbf{1}, b{:}\mathsf{nat} \parr \perp \\
R_L & \triangleq & b\langle 2\rangle.\mathbf{0} \vdash_{\mathsf{CL}} b{:}\mathsf{nat} \otimes \mathbf{1} \qquad S_L \triangleq d(n).\mathbf{0} \vdash_{\mathsf{CL}} d{:}\mathsf{nat} \parr \perp; e{:}\mathbf{1} \\
Q_L & \triangleq & a(n).d\langle 93\rangle.a(s).\mathbf{0} \vdash_{\mathsf{CL}} a{:}\mathsf{nat} \parr \mathsf{str} \parr \perp, d{:}\mathsf{nat} \otimes \mathbf{1}
\end{array}
$$

We define σ, $\sigma_1, \sigma_2, \sigma_3$ such that:

$$
\begin{array}{rcll}
\sigma(P_L) & = & s[\mathsf{p}][\mathsf{q}]\langle 7\rangle; s[\mathsf{p}][\mathsf{r}](a); s[\mathsf{p}][\mathsf{q}]\langle \text{"}hello\text{"}\rangle; \mathbf{0} & \\
\sigma_1(R_L) & = & s[\mathsf{r}][\mathsf{p}]\langle 2\rangle; \mathbf{0} & \sigma_3(S_L) = s[\mathsf{s}][\mathsf{q}](n); \mathbf{0} \\
\sigma_2(Q_L) & = & s[\mathsf{q}][\mathsf{p}](n); s[\mathsf{q}][\mathsf{s}]\langle 93\rangle; s[\mathsf{q}][\mathsf{p}](s); \mathbf{0} &
\end{array}
$$

We thus have:

$$
[\![P_L]\!]_\sigma = \mathsf{q}\!\uparrow\!(\mathsf{nat}); \mathsf{r}\!\downarrow\!(\mathsf{nat}); \mathsf{q}\!\uparrow\!(\mathsf{str}); \mathsf{end} \quad [\![R_L]\!]_{\sigma_1} = \mathsf{p}\!\uparrow\!(\mathsf{nat}); \mathsf{end}
$$
$$
[\![Q_L]\!]_{\sigma_2} = \mathsf{p}\!\downarrow\!(\mathsf{nat}); \mathsf{s}\!\uparrow\!(\mathsf{nat}); \mathsf{p}\!\downarrow\!(\mathsf{str}); \mathsf{end} \quad [\![S_L]\!]_{\sigma_3} = \mathsf{q}\!\downarrow\!(\mathsf{nat}); \mathsf{end}
$$

Let $\Gamma = s[\mathsf{p}]{:}[\![P_L]\!]_\sigma, s[\mathsf{q}]{:}[\![Q_L]\!]_{\sigma_1}, s[\mathsf{r}]{:}[\![R_L]\!]_{\sigma_2}, s[\mathsf{s}]{:}[\![S_L]\!]_{\sigma_3}$. It is then straightforward to see that the following judgment is derivable:

$$
(\boldsymbol{\nu} a, b, c, d)(P_L \mid Q_L \mid R_L \mid S_L) \Vdash^{\emptyset}_{\rho} e{:}\mathbf{1}; \Gamma
$$

Example 3.2 (Choice and Branching)**.** We illustrate how our framework handles choice and branching, by considering the following set of processes. As we discuss in § 4, this typable branching behaviour in CLL is not typable in the previous work on multiparty logic [8]:

$$
\begin{array}{rcl}
P_L & \triangleq & a.\mathsf{case}\{l_1{:}b.l_2; \mathbf{0}, l_3{:}b.l_4; \mathbf{0}\} \qquad Q^1_L \triangleq a.l_1; \mathbf{0} \\
Q^2_L & \triangleq & a.l_3; \mathbf{0} \qquad R_L \triangleq b.\mathsf{case}\{l_2{:}\mathbf{0}, l_4{:}\mathbf{0}\}
\end{array}
$$

with: $P_L \vdash_{\mathsf{CL}} a{:}\,\&\,\{l_1{:}\perp, l_3{:}\perp\}, b{:}\,\oplus\,\{l_2{:}\mathbf{1}, l_4{:}\mathbf{1}\}$
$Q^i_L \vdash_{\mathsf{CL}} a{:}\,\oplus\,\{l_1{:}\mathbf{1}, l_3{:}\mathbf{1}\} \quad R_L \vdash_{\mathsf{CL}} b{:}\,\&\,\{l_2{:}\perp, l_4{:}\perp\}, w{:}\mathbf{1}$

We carry out a mapping from the processes above to MP by defining σ, σ_1 and σ_2 such that:

$$
\begin{array}{rcll}
\sigma(P_L) & = & s[\mathsf{p}][\mathsf{q}] \triangleright \{l_1{:}s[\mathsf{p}][\mathsf{r}] \triangleleft l_2; \mathbf{0}, l_3{:}s[\mathsf{p}][\mathsf{r}] \triangleleft l_4; \mathbf{0}\} & \\
\sigma_1(Q^1_L) & = & s[\mathsf{q}][\mathsf{p}] \triangleleft l_1; \mathbf{0} & \sigma_1(Q^2_L) = s[\mathsf{q}][\mathsf{p}] \triangleleft l_3; \mathbf{0} \\
\sigma_2(R) & = & s[\mathsf{r}][\mathsf{p}] \triangleright \{l_2{:}\mathbf{0}, l_4{:}\mathbf{0}\} &
\end{array}
$$

We can thus generate the following local types:

$$
\begin{array}{rcl}
[\![P_L]\!]_\sigma & = & \&\mathsf{q}\{l_1{:}\oplus\,\mathsf{r}\{l_2{:}\mathsf{end}\}, l_3{:}\oplus\,\mathsf{r}\{l_4{:}\mathsf{end}\}\} \\
[\![Q^1_L]\!]_{\sigma_1} & = & \oplus\mathsf{p}\{l_1{:}\mathsf{end}\} \quad [\![Q^2_L]\!]_{\sigma_1} = \oplus\mathsf{p}\{l_3{:}\mathsf{end}\} \\
[\![R_L]\!]_{\sigma_2} & = & \&\mathsf{p}\{l_2{:}\mathsf{end}, l_4{:}\mathsf{end}\}
\end{array}
$$

Let $\Gamma = s[\mathsf{p}]{:}[\![P_L]\!]_\sigma, s[\mathsf{q}]{:}[\![Q^i_L]\!]_{\sigma_1}, s[\mathsf{r}]{:}[\![R_L]\!]_{\sigma_2}$. Then we have:

$$
(\boldsymbol{\nu} a, b)(P_L \mid Q^i_L \mid R_L) \Vdash^{\emptyset}_{\rho} w{:}\mathbf{1}; \Gamma
$$

4. Partial multiparty compatibility

This section studies a compositional concurrent synthesis property, dubbed *partial multiparty compatibility* (PMC). As illustrated by Fact 2.1, a multiparty session type theory [9, 14] cannot guarantee deadlock-freedom if we do not rely on either (1) a projection from a global type; or (2) a global synthesis property called *multiparty compatibility* [11, 17]. For example, the previous counterexample can be avoided if we start from the following global type:

$$G = \mathsf{p} \to \mathsf{q} : (\mathsf{nat}).\mathsf{q} \to \mathsf{r} : (\mathsf{bool}).\mathsf{r} \to \mathsf{p} : (\mathsf{char}).\mathsf{end} \tag{3}$$

then type each process with the following local types.

$$T_{\mathsf{p}} = \mathsf{q}{\uparrow}(\mathsf{nat}); \mathsf{r}{\downarrow}(\mathsf{char}); \mathsf{end}, \quad T_{\mathsf{q}} = \mathsf{p}{\downarrow}(\mathsf{nat}); \mathsf{r}{\uparrow}(\mathsf{bool}); \mathsf{end}$$

$$T_{\mathsf{r}} = \mathsf{q}{\downarrow}(\mathsf{bool}); \mathsf{p}{\uparrow}(\mathsf{char}); \mathsf{end} \tag{4}$$

or we may build (synthesise) G in (3) from $\{T_{\mathsf{p}}, T_{\mathsf{q}}, T_{\mathsf{r}}\}$ in (4). If we start from a projectable global type or are able to synthesise a global type, the example in Fact 2.1 is no longer typable.

Given that CLL employs a binary form of composition, to prove negative results about expressiveness or define multicut rules in later sections, we extend a global synthesis condition to a binary (partial) relation. We introduce *partial* global types whose arrow $\mathsf{p} \rightsquigarrow \mathsf{q}$ represents a global interaction which has not yet been composed with another party (e.g. it denotes the emission from p to q, not yet composed with the reception by q). When we compose two partial specifications of participant p and q by *fusing* two partial global types, $\mathsf{p} \rightsquigarrow \mathsf{q}$ is changed to $\mathsf{p} \to \mathsf{q}$, preserving the ordering of communications. When we finish composing all participants (thus reconstructing a *complete* global type – one without partial arrows), deadlock-freedom is guaranteed.

4.1 Partial global types and semantics

We first define a partial global type G, consisting of a combination of complete global types and endpoint interactions.

Definition 4.1 (Partial Global Types)**.** The grammar of partial global types (G, G', ...) is defined as:

$$\begin{array}{rcll} G & ::= & \mathsf{end} \mid \mathsf{p} \to \mathsf{q} : (\tau).G & \mid \quad \mathsf{p} \to \mathsf{q} : \{l_j{:}G_j\}_{j\in J} \\ & \mid & \mathsf{p} \rightsquigarrow \mathsf{q} :{\uparrow} (\tau).G & \mid \quad \mathsf{p} \rightsquigarrow \mathsf{q} :{\downarrow} (\tau).G \\ & \mid & \mathsf{p} \rightsquigarrow \mathsf{q} : \oplus\{l_j{:}G\}_{j\in J} & \mid \quad \mathsf{p} \rightsquigarrow \mathsf{q} : \&\{l_j{:}G_j\}_{j\in J} \end{array}$$

We omit $\uparrow, \downarrow, \oplus$ and $\&$ from the partial global types when clear from context or unnecessary. Given a partial global type G, we write $\mathsf{roles}(G)$ to denote the set of role names occurring in G. We write $\mathsf{p} \rightsquigarrow \mathsf{q} \in G$ if $\mathsf{p} \rightsquigarrow \mathsf{q}$ occurs in G. Similarly for $\mathsf{p} \to \mathsf{q} \in G$. We say G is *complete* if $\mathsf{p} \rightsquigarrow \mathsf{q} \notin G$ for all $\mathsf{p}, \mathsf{q} \in \mathsf{roles}(G)$. $\diamond$

The first three constructs of Def. 4.1 represent the standard global types [9, 14]. Global type $\mathsf{p} \to \mathsf{q}{:}(\tau).G$ means that participant p sends a value of type τ to participant q, and the rest of interaction is specified by G. Global type $\mathsf{p} \to \mathsf{q}{:}\{l_j{:}G_j\}_{j\in J}$ means that participant p selects label l_i, then q's i-th branch will be chosen, becoming G_i. On the other hand, *partial* global types (the last four components) denote *half* of a complete global interaction. The modes ($\uparrow, \downarrow, \oplus, \&$) in partial global types indicate which component of the partial global interaction is being satisfied. For instance, $\mathsf{p} \rightsquigarrow \mathsf{q} :{\uparrow} (\tau)$ denotes the contribution of the emission component of the interaction (from a local type $\mathsf{q}{\uparrow}(\tau)$), whereas $\mathsf{p} \rightsquigarrow \mathsf{q} :{\downarrow} (\tau)$ denotes the reception component. A similar reasoning applies to selection and branching.

We introduce a standard notion of projection from global to local types, defined in terms of a merge operation for branchings [11], written $T \sqcup T'$, ensuring that if the locally observable behaviour of the local type is not independent of the chosen branch then it is identifiable via a unique choice/branching label (the merge operator is otherwise undefined) (see Appendix A.3).

Definition 4.2 (Projection)**.** Let G by a global type. The projection of G for a role p is defined by the function $G \upharpoonright \mathsf{p}$ below. If no side conditions hold then projection is undefined.

$$\mathsf{s} \to \mathsf{r} : (\tau).G' \upharpoonright \mathsf{p} = \begin{cases} \mathsf{r}{\uparrow}(\tau); (G' \upharpoonright \mathsf{p}) & \text{if } \mathsf{p} = \mathsf{s} \\ \mathsf{s}{\downarrow}(\tau); (G' \upharpoonright \mathsf{p}) & \text{if } \mathsf{p} = \mathsf{r} \\ G' \upharpoonright \mathsf{p} & \text{otherwise} \end{cases}$$

$$\mathsf{s} \to \mathsf{r} : \{l_j{:}G_j\}_{j\in J} \upharpoonright \mathsf{p} = \begin{cases} \oplus\mathsf{r}\{l_j{:}G_j \upharpoonright \mathsf{p}\}_{j\in J} & \text{if } \mathsf{p} = \mathsf{s} \\ \&\mathsf{s}\{l_j{:}G_j \upharpoonright \mathsf{p}\}_{j\in J} & \text{if } \mathsf{p} = \mathsf{r} \\ \sqcup_{j\in J} G_j \upharpoonright \mathsf{p} & \text{otherwise} \end{cases}$$

$$\mathsf{end} \upharpoonright \mathsf{p} = \mathsf{end}$$

As an illustration of merging, consider

$$G_m = \mathsf{q} \to \mathsf{p}{:}\{l_1{:}\mathsf{p} \to \mathsf{r}{:}\{l_2{:}\mathsf{end}\}, l_3{:}\mathsf{p} \to \mathsf{r}{:}\{l_4{:}\mathsf{end}\}\} \tag{5}$$

Then we have

$$G_m \upharpoonright \mathsf{p} = \&\mathsf{q}\{l_1{:} \oplus \mathsf{r}\{l_2 : \mathsf{end}\}, l_3{:} \oplus \mathsf{r}\{l_4 : \mathsf{end}\}\}, \tag{6}$$

$$G_m \upharpoonright \mathsf{q} = \oplus\mathsf{p}\{l_1 : \mathsf{end}, l_3 : \mathsf{end}\},\ G_m \upharpoonright \mathsf{r} = \&\mathsf{p}\{l_2{:}\mathsf{end}, l_4{:}\mathsf{end}\}$$

To define the semantics of global and local types, we introduce a swapping relation allowing for the permutation of independent global actions.

Definition 4.3 (Swapping)**.** We define swapping, written $\sim_{\mathsf{sw}}$ as the smallest congruence on complete global types satisfying the following where $\mathsf{p} \neq \mathsf{p}'$ and $\mathsf{q} \neq \mathsf{q}'$:

$$\begin{array}{ll} (\mathsf{ss}) & \mathsf{p} \to \mathsf{q} : (\tau).\mathsf{p}' \to \mathsf{q}' : (\tau').G \sim_{\mathsf{sw}} \mathsf{p}' \to \mathsf{q}' : (\tau').\mathsf{p} \to \mathsf{q} : (\tau).G \\ (\mathsf{sb}) & \mathsf{p} \to \mathsf{q} : (\tau).\mathsf{p}' \to \mathsf{q}' : \{l_i : G_i\}_{i\in I} \\ & \quad \sim_{\mathsf{sw}} \mathsf{p}' \to \mathsf{q}' : \{l_i : \mathsf{p} \to \mathsf{q} : (\tau).G_i\}_{i\in I} \\ (\mathsf{bb}) & \mathsf{p} \to \mathsf{q} : \{l_i : \mathsf{p}' \to \mathsf{q}' : \{l'_j : G_j\}_{j\in J}\}_{i\in I} \\ & \quad \sim_{\mathsf{sw}} \mathsf{p}' \to \mathsf{q}' : \{l'_j : \mathsf{p} \to \mathsf{q} : \{l_i : G_j\}_{i\in I}\}_{j\in J} \end{array}$$

Swapping extends to partial global types in the obvious way. $\diamond$

The operational semantics for local, global types and configurations are given by labelled transition systems (LTS).

Definition 4.4 (LTS for Local Types)**.** We define the syntax of labels as:

$$\ell ::= \mathsf{pq}{\uparrow}(\tau) \mid \mathsf{pq}{\downarrow}(\tau) \mid \mathsf{pq} \triangleleft l \mid \mathsf{pq} \triangleright l$$

We define $\overline{\ell}$ as $\overline{\mathsf{pq}{\uparrow}(\tau)} = \mathsf{qp}{\downarrow}(\tau)$ and $\overline{\mathsf{pq} \triangleleft l} = \mathsf{qp} \triangleright l$ and vice-versa. The transition relation between local types $T \xrightarrow{\ell} T'$ for role p is defined by the following rules (assuming $k \in I$ in $(\mathsf{sel}/\mathsf{br})$):

$$(\mathsf{out}) \quad \mathsf{q} \uparrow (\tau); T \xrightarrow{\mathsf{pq}{\uparrow}(\tau)} T \qquad (\mathsf{in}) \quad \mathsf{q} \downarrow (\tau); T \xrightarrow{\mathsf{pq}{\downarrow}(\tau)} T$$

$$(\mathsf{sel}) \quad \oplus \mathsf{q}\{\mathsf{l}_i{:}T_i\}_{i\in I} \xrightarrow{\mathsf{pq} \triangleleft l_k} T_k \quad (\mathsf{br}) \quad \&\mathsf{q}\{\mathsf{l}_i{:}T_i\}_{i\in I} \xrightarrow{\mathsf{pq} \triangleright l_k} T_k$$

Definition 4.5 (LTS for Global Types)**.** The transition relation between global types $G \xrightarrow{\ell\cdot\ell'} G'$ is defined by the following rules:

$$(\mathsf{io}) \quad \mathsf{p} \to \mathsf{q} : (\tau).G \xrightarrow{\mathsf{pq}{\uparrow}(\tau)\cdot\mathsf{qp}{\downarrow}(\tau)} G$$

$$(\mathsf{sb}) \quad \mathsf{p} \to \mathsf{q} : (l_i : G_i)_{i\in I} \xrightarrow{\mathsf{pq} \triangleleft l_k \cdot \mathsf{qp} \triangleright l_k} G_k$$

$$(\mathsf{sw}) \quad G_1 \sim_{\mathsf{sw}} G'_1 \wedge G'_1 \xrightarrow{\ell\cdot\ell'} G'_2 \wedge G'_2 \sim_{\mathsf{sw}} G_2 \Rightarrow G_1 \xrightarrow{\ell\cdot\ell'} G_2$$

Definition 4.6 (Configurations)**.** Given a set of roles $\mathcal{P}$, we define *a configuration* as $C = (T_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$. The synchronous transition relation between configurations is defined as:

$$(\mathsf{com}) \quad \frac{T_{\mathsf{p}} \xrightarrow{\ell} T'_{\mathsf{p}} \quad T_{\mathsf{q}} \xrightarrow{\overline{\ell}} T'_{\mathsf{q}} \quad T_{\mathsf{r}} = T'_{\mathsf{r}} \quad \mathsf{r} \neq \mathsf{p}, \mathsf{r} \neq \mathsf{q}}{(T_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}} \xrightarrow{\ell\cdot\overline{\ell}} (T'_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}}$$

We write $G \xrightarrow{\vec{\ell}} G_{n-1}$ if $G \xrightarrow{\ell_1\cdot\ell_2} G_1 \cdots \xrightarrow{\ell_{n-1}\cdot\ell_n} G_{n-1}$ and $\vec{\ell} = \ell_1 \cdots \ell_n$ and similarly for T and C. We define *traces of global type* G_0 as $Tr(G_0) = \{\vec{\ell} \mid G_0 \xrightarrow{\vec{\ell}} G_n\ n \geq 0\}$. Similarly for configurations.

Proposition 4.1 (Trace Equivalence)**.** *Suppose G is a well-formed global type and the set of participants in G is $\mathcal{P}$. Assume $C = (T_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}} = (G \upharpoonright \mathsf{p})_{\mathsf{p}\in\mathcal{P}}$. Then $Tr(C) = Tr(G)$.*

4.2 Partial Multiparty Compatibility

In order to define partial multiparty compatibility, we first define multiparty compatibility for synchronous semantics, which is simpler than the one for an asynchronous semantics [11, 17].

Definition 4.7 (Synchronous Multiparty Compatibility)**.** Configuration $C_0 = (T_{0\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$ is *synchronous multiparty compatible* if for all $C_0 \xrightarrow{\vec{\ell}} C = (T_\mathsf{p})_{\mathsf{p}\in\mathcal{P}}$ and $T_\mathsf{p} \xrightarrow{\ell} T'_\mathsf{p}$,

1. if $\ell = \mathsf{pq}{\uparrow}(\tau)$ or $\mathsf{pq} \triangleleft l$, there exists $C \xrightarrow{\vec{\ell'}} C' \xrightarrow{\ell} \xrightarrow{\overline{\ell}} C''$;
2. if $\ell = \mathsf{pq}{\downarrow}(\tau)$, there exists $C \xrightarrow{\vec{\ell'}} C' \xrightarrow{\overline{\ell}} \xrightarrow{\ell} C''$; or
3. if $\ell = \mathsf{pq} \triangleright l$, there exists $\ell_1 = \mathsf{pq} \triangleright l'$, $C \xrightarrow{\vec{\ell'}} C' \xrightarrow{\ell_1} \xrightarrow{\overline{\ell}_1} C''$ where $\vec{\ell'}$ does not include actions from or to p.

One can check that the two sets of local types in (4) and (6) satisfy synchronous multiparty compatibility.

Definition 4.8 (Deadlock-freedom)**.** $C = (T_\mathsf{p})_{\mathsf{p}\in\mathcal{P}}$ *is* deadlock-free *if for all* $C \xrightarrow{\vec{\ell}} C_1$*, there exists* $C' = (T'_\mathsf{p})_{\mathsf{p}\in\mathcal{P}}$ *such that* $C_1 \xrightarrow{\vec{\ell'}} C'$ *and* $T'_\mathsf{p} = \mathsf{end}$ *for all* $\mathsf{p} \in \mathcal{P}$.

Theorem 4.1 (Deadlock-freedom, MC and a Global Type)**.** *The following three statements are equivalent.*

1. (MC) *A configuration C is synchronous multiparty compatible.*
2. (DF) *C is deadlock-free.*
3. (WF) *There exists well-formed G such that $Tr(G) = Tr(C)$.*

Multiparty compatibility is a global property defined using the set of all participants (modelled by communicating automata in [11, 17]). To define a compositional (local) multiparty compatibility, we introduce the composition of two partial global types, which we dub as *fusion*.

Definition 4.9 (Fuse)**.** We define the fusing of two partial global types G_1, G_2, written $\mathsf{fuse}(G_1, G_2)$, inductively on the structure of G_1 and G_2, up to the swapping relation $\sim_{\mathsf{sw}}$:

$$\mathsf{fuse}(\mathsf{end},\mathsf{end}) = \mathsf{end}$$
$$\mathsf{fuse}(\mathsf{p}\rightsquigarrow\mathsf{q}:\uparrow(\tau).G'_1, \mathsf{p}\rightsquigarrow\mathsf{q}:\downarrow(\tau).G'_2) = \mathsf{p}\to\mathsf{q}:(\tau).\mathsf{fuse}(G'_1, G'_2)$$
$$\mathsf{fuse}(\mathsf{p}\rightsquigarrow\mathsf{q}:\oplus\{l:G'_1\}, \mathsf{p}\rightsquigarrow\mathsf{q}:\&\{l:G'_2, \{l_j:G_j\}_{j\in J}\}) = \mathsf{p}\to\mathsf{q}:\{l:\mathsf{fuse}(G'_1,G'_2), \{l_j:G_j\}_{j\in J}\}$$
$$\mathsf{fuse}(\mathsf{p}\rightsquigarrow\mathsf{q}:\uparrow(\tau).G_1, G_2) = \mathsf{p}\rightsquigarrow\mathsf{q}:\uparrow(\tau).\mathsf{fuse}(G_1,G_2) \quad \text{if not } \mathsf{p}\rightsquigarrow\mathsf{q}:\downarrow(\tau).G'_2 \sim_{\mathsf{sw}} G_2$$

The last rule (which does not overlap with the first three rules) can be extended similarly to input, branching and selection in partial and complete global types. We say $\mathsf{fuse}(G_1, G_2) = G$ is well-formed if $\mathsf{p}\to\mathsf{q}\in G$ then $\mathsf{p}\rightsquigarrow\mathsf{q}\notin G$ and $\mathsf{q}\rightsquigarrow\mathsf{p}\notin G$. ◇

We can now define the main contribution of this section: a compositional notion of *partial* multiparty compatibility (PMC).

Definition 4.10 (Partial Multiparty Compatibility)**.** Suppose G_1 and G_2 are partial global types. G_1 and G_2 are partial multiparty compatible iff $\mathsf{fuse}(G_1, G_2)$ is well-formed. ◇

Example 4.1 (Partial Multiparty Compatibility)**. (1)** Consider the following partial global types between two participants which represent a deadlock:

$$G_1 = \mathsf{p}\rightsquigarrow\mathsf{q}:(\mathsf{nat}).\mathsf{q}\rightsquigarrow\mathsf{p}:(\mathsf{bool}).\mathsf{end}$$
$$G_2 = \mathsf{q}\rightsquigarrow\mathsf{p}:(\mathsf{bool}).\mathsf{p}\rightsquigarrow\mathsf{q}:(\mathsf{nat}).\mathsf{end}$$

Then $G_3 = \mathsf{fuse}(G_1, G_2) = \mathsf{p}\rightsquigarrow\mathsf{q}:(\mathsf{nat}).\mathsf{q}\to\mathsf{p}:(\mathsf{bool}).\mathsf{p}\rightsquigarrow\mathsf{q}:(\mathsf{nat}).\mathsf{end}$ which is not well-formed since $\mathsf{p}\rightsquigarrow\mathsf{q}\in G_3$ but $\mathsf{q}\to\mathsf{p}\in G_3$.

(2) Consider (as we mention in § 5, these correspond to the partial global types of Example 3.2. The respective complete global type is untypable in [8]):

$$G_1 = \mathsf{q}\rightsquigarrow\mathsf{p}:\&\{l_1{:}\mathsf{p}\rightsquigarrow\mathsf{r}:\oplus\{l_2{:}\mathsf{end}\}, l_3{:}\mathsf{p}\rightsquigarrow\mathsf{r}:\oplus\{l_4{:}\mathsf{end}\}\}$$
$$G_2 = \mathsf{q}\rightsquigarrow\mathsf{p}:\oplus\{l_1{:}\mathsf{end}\} \quad G_3 = \mathsf{p}\rightsquigarrow\mathsf{r}:\&\{l_2{:}\mathsf{end}, l_4{:}\mathsf{end}\}$$

We then have that:

$$\mathsf{fuse}(G_1,G_2) = \mathsf{q}\to\mathsf{p}:\{l_1{:}\mathsf{p}\rightsquigarrow\mathsf{r}:\oplus\{l_2{:}\mathsf{end}\}, l_3{:}\mathsf{p}\rightsquigarrow\mathsf{r}:\oplus\{l_4{:}\mathsf{end}\}\}$$
$$\mathsf{fuse}(\mathsf{fuse}(G_1,G_2),G_3) = \mathsf{q}\to\mathsf{p}:\{l_1{:}\mathsf{p}\to\mathsf{r}:\{l_2{:}\mathsf{end}\}, l_3{:}\mathsf{p}\to\mathsf{r}:\{l_4{:}\mathsf{end}\}\}$$

Then: $\mathsf{fuse}(\mathsf{fuse}(G_1,G_2),G_3) = \mathsf{fuse}(G_1,\mathsf{fuse}(G_2,G_3))$. ◇

Lemma 4.1. *Suppose* $\mathsf{fuse}(\mathsf{fuse}(G_i,G_j),G_k)$ *with* $\{i,j,k\} = \{1,2,3\}$ *is well-formed. Then* $\mathsf{fuse}(\mathsf{fuse}(G_i,G_j),G_k) \sim_{\mathsf{sw}} \mathsf{fuse}(G_i,\mathsf{fuse}(G_j,G_k))$.

Theorem 4.2 (Compositionality)**.** *Suppose G_1, ...,G_n are partial global types. Assume for all i, j such that $1 \le i \ne j \le n$, G_i and G_j are partial multiparty compatible and $G = \cup_i \mathsf{fuse}(G_k)_{k\in I}$ is a complete global type. Then G is well-formed.*

5. CLL encoded as a single multiparty session

Having defined in § 3 how to translate CLL processes to MP, we study the interconnection networks induced by CLL processes by generating the corresponding *partial global types* (§ 4). We prove a strict inclusion of interconnections of CLL into those of single MP, by *fusing* the partial global types into a *complete* global type.

Definition 5.1 (Generating Partial Global Types)**.** Given P such that $P \vdash^\sigma_{\mathsf{CL}} \Delta$ we generate its partial global type, written $\#_\sigma(T)$ with local type $T = [\![P]\!]_\sigma$ as follows:

$$\begin{array}{lcl}
\#_\sigma(\mathsf{end}) & \triangleq & \mathsf{end}\\
\#_\sigma(\mathsf{q}!(\tau);T) & \triangleq & \mathsf{p}_\sigma\rightsquigarrow\mathsf{q}:\uparrow(\tau).\#_\sigma(T)\\
\#_\sigma(\mathsf{q}?(\tau);T) & \triangleq & \mathsf{q}\rightsquigarrow\mathsf{p}_\sigma:\downarrow(\tau).\#_\sigma(T)\\
\#_\sigma(\oplus\mathsf{q}\{l_i:T_i\}_{i\in I}) & \triangleq & \mathsf{p}_\sigma\rightsquigarrow\mathsf{q}:\oplus\{l_i{:}\#_\sigma(T_i)\}_{i\in I}\\
\#_\sigma(\&\mathsf{q}\{l_i:T_i\}_{i\in I}) & \triangleq & \mathsf{q}\rightsquigarrow\mathsf{p}_\sigma:\&\{l_i{:}\#_\sigma(T_i)\}_{i\in I}
\end{array}$$

We write $\#_\sigma(P)$ for $\#_\sigma([\![P]\!]_\sigma)$.

We then generate a set of partial global types $\mathcal{G}$, following the rules in Definition 3.3. We extend $\Vdash$ judgment with $\mathcal{G}$, written $P \Vdash^\sigma_\rho \Delta;\Gamma;\mathcal{G}$ as follows:

$$(\mathsf{thread}\text{-}\mathcal{G})\quad \frac{P_L \vdash^\sigma_{\mathsf{CL}} \Delta}{P_L \Vdash^\sigma_\emptyset \Delta;\mathsf{c}_\sigma[\mathsf{p}_\sigma]{:}[\![P]\!]_\sigma;\{\#_\sigma(P)\}}$$

$$(\mathsf{comp}\text{-}\mathcal{G})\quad \frac{P_L \vdash^\sigma_{\mathsf{CL}} \Delta, a{:}A \quad Q_L \Vdash^{\sigma'}_\rho \Delta', a{:}A^\perp;\Gamma;\mathcal{G} \quad (\star)\text{ in }(\mathsf{comp})}{(\boldsymbol{\nu} a)(P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{x\}}_{\rho'} \Delta,\Delta';\Gamma,\mathsf{c}_\sigma[\mathsf{p}_\sigma]{:}[\![P]\!]_\sigma;\mathcal{G}\cup\{\#_\sigma(P)\}}$$

Example 5.1 (Four Threads)**.** Using the procedure defined above, we generate the following partial global types for Example 3.1:

$$\begin{array}{lcl}
\#_\sigma(P) & = & \mathsf{p}\rightsquigarrow\mathsf{q}:\uparrow(\mathsf{nat}).\mathsf{r}\rightsquigarrow\mathsf{p}:\downarrow(\mathsf{nat}).\mathsf{p}\rightsquigarrow\mathsf{q}:\uparrow(\mathsf{str}).\mathsf{end}\\
\#_{\sigma_1}(R) & = & \mathsf{r}\rightsquigarrow\mathsf{p}:\uparrow(\mathsf{nat}).\mathsf{end}\\
\#_{\sigma_2}(Q) & = & \mathsf{p}\rightsquigarrow\mathsf{q}:\downarrow(\mathsf{nat}).\mathsf{q}\rightsquigarrow\mathsf{s}:\uparrow(\mathsf{nat}).\mathsf{p}\rightsquigarrow\mathsf{q}:\downarrow(\mathsf{str}).\mathsf{end}\\
\#_{\sigma_3}(S) & = & \mathsf{q}\rightsquigarrow\mathsf{s}:\downarrow(\mathsf{nat}).\mathsf{end}
\end{array}$$

Where applying fuse to the partial global types above produces the global type: $\mathsf{p}\to\mathsf{q}:(\mathsf{nat}).\mathsf{q}\to\mathsf{s}:(\mathsf{nat}).\mathsf{r}\to\mathsf{p}:(\mathsf{nat}).\mathsf{p}\to\mathsf{q}:(\mathsf{str}).\mathsf{end}$. We note that, for instance, adding a message between r and s makes the example not typable in CLL since it introduces a 3-way cycle in the interconnection network topology. ◇

Example 5.2 (Choice and Branching)**.** The partial global types generated for Example 3.2 are $G_1 = \#_\sigma(P)$, $G_2 = \#_{\sigma_1}(Q_1)$ and $G_3 = \#_{\sigma_2}(R)$ in Example 4.1(2).

We now make precise the claims in § 1. We begin by formalising the notion of interconnection network as an un-directed graph.

Definition 5.2 (Interconnection Network)**.** Given a global type G we generate the interconnection network graph for the roles of G where the nodes of the graph are the roles of G and two nodes p,q have an edge between them (written as $\mathsf{p}\leftrightarrow\mathsf{q}$) if $\mathsf{p}\to\mathsf{q}$ or $\mathsf{q}\to\mathsf{p}$ or $\mathsf{p}\rightsquigarrow\mathsf{q}$ or $\mathsf{q}\rightsquigarrow\mathsf{p}$ occurs in G. ◇

We first establish two basic properties of our framework: (1) we can always fuse the partial global types of a well-formed composition; (2) any two such partial types overlaps in at most 2 roles.

Proposition 5.1. *Let $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$ and $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$. There exists a single well-formed global type G such that $G = \mathsf{fuse}(\mathcal{G})$ where $\mathsf{fuse}(\mathcal{G})$ denotes fusion of all partial global types in $\mathcal{G}$.*

Theorem 5.1. *Let $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$. For any distinct $G_1, G_2 \in \mathcal{G}$ we have that $\mathsf{roles}(G_1) \cap \mathsf{roles}(G_2)$ contains at most 2 elements.*

By the above results, we can immediately show the following separation result between general MP global types and those induced by CLL-valid processes.

Theorem 5.2. *Let $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$. Let $G = \mathsf{fuse}(\mathcal{G})$. The interconnection network graph for G is acyclic.*

Finally, we establish that our encoding indeed produces a single multiparty session, that is, the fusing of all partial global types in a complete session is synchronous multiparty compatible.

Theorem 5.3. *Let $P \Vdash^{\emptyset}_{\rho} \Delta; \Gamma; \mathcal{G}$ and $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$. Then we have: (1) $P \to^* \mathbf{0}$; (2) $\mathsf{fuse}(\mathcal{G})$ is well-formed and deadlock-free.*

As a consequence of Prop. 3.1 and Theorem 5.3, the image of the translation into MP is deadlock-free.

Corollary 5.1. *If $P \Vdash^{\emptyset}_{\rho} \Delta; \Gamma$ and $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$, then $\rho(\sigma(P)) \vdash_{\mathsf{MP}} \Gamma$ is deadlock-free.*

6. Channel passing

Typing rules and translations We integrate delegation (channel passing) into our framework. In CLL this amounts to replacing the rules for $\otimes$ and $\parr$ of § 2 with:

$$(\otimes)\ \frac{P_L \vdash_{\mathsf{CL}} \Delta, y{:}A \quad Q_L \vdash_{\mathsf{CL}} \Delta', x{:}B}{\overline{x}\langle y\rangle.(P_L \mid Q_L) \vdash_{\mathsf{CL}} \Delta, \Delta', x{:}A \otimes B} \qquad (\parr)\ \frac{P_L \vdash_{\mathsf{CL}} \Delta, y{:}A, x{:}B}{x(y).P_L \vdash_{\mathsf{CL}} \Delta, x{:}A \parr B}$$

Thus, $x{:}A \otimes B$ identifies a channel along which a process sends a fresh y that is used according to type A, and then proceeds as B. The channels of the subprocesses using x and y must be *disjoint*. $A \parr B$ denotes the dual behaviour. In MP, delegation is embodied by the following rules (types are extended in the obvious way):

$$(\mathsf{csend})\ \frac{P \vdash_{\mathsf{MP}} \Delta, c{:}T}{c[\mathsf{q}]\langle c'\rangle; P \vdash_{\mathsf{MP}} \Delta, c'{:}T'; c{:}\mathsf{q}{\uparrow}(T'); T} \qquad (\mathsf{crecv})\ \frac{P \vdash_{\mathsf{MP}} \Delta, c{:}T, x{:}T'}{c[\mathsf{q}](x); P \vdash_{\mathsf{MP}} \Delta, c{:}\mathsf{q}{\downarrow}(T'); T}$$

Unlike in CLL, MP processes delegate a session by performing a *free* output of an ambient session channel. Session input is dual. We assume role identifiers in different sessions are disjoint.

To construct a mapping from CLL to MP with delegation, we relax our single role per CLL thread restriction due to the $\otimes$ typing rule, which splits the context into two distinct regions: one for usage by the emitted session and another for the continuation. If all channels were assigned to the same role, the resulting process would not be typable in MP, which disallows for role implementations across parallel threads.

Thus, we consider a general *bijective* mapping from CLL to role-annotated MP channels, where actions in a cut-free process may map to actions pertaining to *different* MP roles. Moreover, since cut-free processes now have bound as well as free names, we define a mapping η of bound names to variables in MP assigned to a *single* role, denoting the *destination* role for communication along the channel, which to maintain consistency with MP must be a *different* session than that used to send or receive the delegated name.

Definition 6.1 (Name to Role-Indexed Channel Mapping)**.** Let $P_L \vdash_{\mathsf{CL}} \Delta$ without using the cut rule. We define a name to role-indexed channel mapping of P as a pair of mappings (σ, η) such that: For all $x, y \in fn(P_L)$ such that $x \neq y$, if $\sigma(x) = s[\mathsf{p}][\mathsf{q}]$ and $\sigma(y) = s[\mathsf{p}'][\mathsf{q}']$, then $\mathsf{p} \neq \mathsf{q}' \wedge \mathsf{q} \neq \mathsf{p}' \wedge (\mathsf{p} \neq \mathsf{p}' \vee \mathsf{q} \neq \mathsf{q}')$. For all $x, y \in bn(P_L)$ such that $x \neq y$, if $\eta(x) = z[\mathsf{p}]$ and $\eta(y) = z[\mathsf{p}']$ then $\mathsf{p}' \neq \mathsf{p}$. Given $\eta(x) = x[\mathsf{p}]$, we write $\mathsf{d}_\eta(x)$ for p. $\diamond$

The role restrictions in Def. 6.1 guarantee that the renaming is bijective and that in a single thread we cannot implement dual role endpoints. In addition to Def. 6.1, we introduce a well-formedness condition to ensure that instances of the $\otimes$ rule do not induce roles being split across different threads.

Definition 6.2. Let $P_L \vdash_{\mathsf{CL}} \Delta$ without using the cut rule and (σ, η) be a mapping viz. Def 6.1. We say that (σ, η) is well-formed if for each instance of $\otimes$ in the typing derivation of P_L:

$$\frac{P_{1L} \vdash_{\mathsf{CL}} \Delta_1, y{:}A \quad P_{2L} \vdash_{\mathsf{CL}} \Delta_2, x{:}B}{\overline{x}\langle y\rangle.(P_{1L} \mid P_{2L}) \vdash_{\mathsf{CL}} \Delta_1, \Delta_2, x{:}A \otimes B}$$

For all $z \in \Delta_1$ and $z' \in \Delta_2, x{:}$ $\mathsf{c}_\sigma(z) = \mathsf{c}_\sigma(z')$ implies $\mathsf{p}_\sigma(z) \neq \mathsf{p}_\sigma(z')$. $P_L \vdash^{(\sigma,\eta)}_{\mathsf{CL}} \Delta$ denotes (σ, η) is well-formed wrt $P_L \vdash_{\mathsf{CL}} \Delta$.

Since threads can now denote potentially multiple roles, we must generate a local typing *context* from each thread, assigning local types to each role annotated channel. In the cases of selection and branching we collect actions pertaining to the principal role assignment for each branch and combine the result with the context generation for the continuation processes (since it may contain different role assignments). To generate the local type for session output, we produce the delegated session type from the usage of y in subprocess P and proceed inductively, noting that there will be a binding for y in the resulting context (since delegation is a free output in MP). For session input, we dualise the delegated type so that both input and output types match in MP, as needed.

Definition 6.3 (Local Type Generation)**.** Let $P_L \vdash^{(\sigma,\eta)}_{\mathsf{CL}} \Delta$ where all free and bound names are distinct. We generate a local typing context Γ such that $\eta(\sigma(P_L)) \vdash_{\mathsf{MP}} \Gamma$ by induction on the structure of P_L, written $[\![P]\!]^\eta_\sigma$ (we write $\mathsf{c}_\varphi(x)$, $\mathsf{p}_\varphi(x)$ and $\mathsf{d}_\varphi(x)$, where φ stands for σ if $x \in fn(P_L)$ and η otherwise; and c for $\mathsf{c}_\varphi(x)[\mathsf{p}_\varphi(x)]$ if $x \in fn(P_L)$ and for x otherwise; $\overline{T}$ denotes a dual type of T):

$$\begin{array}{lcl}
[\![\mathbf{0}]\!]^\eta_\sigma & \triangleq & \emptyset \\
[\![\overline{x}\langle y\rangle.(P \mid Q)]\!]^\eta_\sigma & \triangleq & c{:}\mathsf{d}_\varphi(x){\uparrow}([\![P]\!]^\eta_\sigma(y)) \uplus ([\![P]\!]^\eta_\sigma \uplus [\![Q]\!]^\eta_\sigma) \\
[\![x(y).P]\!]^\eta_\sigma & \triangleq & c{:}\mathsf{d}_\varphi(x){\downarrow}([\![P]\!]^\eta_\sigma(y)) \uplus ([\![P]\!]^\eta_\sigma \setminus \eta(y)) \\
[\![x.l_j; P]\!]^\eta_\sigma & \triangleq & c{:} \oplus \mathsf{d}_\varphi(x)\{l_j{:}[\![P]\!]^\eta_\sigma(c)\} \uplus [\![P]\!]^\eta_\sigma \\
[\![x.\mathsf{case}\{l_i : P_i\}_{i\in I}]\!]^\eta_\sigma & \triangleq & c{:} \,\&\, \mathsf{d}_\varphi(x)\{l_i{:}[\![P_i]\!]^\eta_\sigma(c)\}_{i\in I} \uplus [\![P]\!]^\eta_\sigma \\
[\![[x \leftrightarrow y]]\!]^\eta_\sigma & \triangleq & [\![\mathsf{id}_A(x,y)]\!]^\eta_\sigma \text{ with } \Delta = x{:}A, y{:}A^\bot
\end{array}$$

In Def. 6.3 above we use an operation $s[\mathsf{p}]{:}T \uplus \Gamma$ defined as $\Gamma, s[\mathsf{p}]{:}T$ if $s[\mathsf{p}] \notin \Gamma$ and $\Gamma \neq \emptyset$. If $\Gamma = \Gamma', s[\mathsf{p}] : T'$, we change T' to $T; T'$ if T is an input our output; or, change T' to T if T is a selection or branching (since T already contains all actions of role p by construction). If Γ is empty we append end to T in the relevant cases. See Appendix A.8 for a full definition.

Proposition 6.1. *If $P_L \vdash^{(\sigma,\eta)}_{\mathsf{CL}} \Delta$ then $\sigma(\eta(P_L)) \vdash_{\mathsf{MP}} [\![P]\!]^\eta_\sigma$. Where $\sigma(P)$ substitutes free names of P by their role-indexed channel mappings and forwarding by the identity process, and $\eta(P)$ replaces non-binding occurrences of bound names by their multiparty session calculus actions with destination given by η.*

As in Def. 6.1, we define the judgment $P \vDash^{\sigma,\eta}_{\sigma} \Delta; \Gamma$ where (σ, η) is a *well-formed* bijective mapping. The key condition ($\star$) in Definition 3.3 is changed to (†) below taking multiple role threads into account and bound names (the red letters highlight differences). We omit the rule for single threads, which just absorbs the $\vdash^{\sigma,\eta}_{\mathsf{CL}}$ judg-

ment and generates the corresponding local types (cf. Def 6.3).

$$\dagger \begin{cases} \rho' = \rho \cup (x, \mathsf{c}_\sigma(x)[\mathsf{p}_\sigma(x)][\mathsf{d}_\sigma(x)]) \\ \mathsf{c}_\sigma(x) = \mathsf{c}_{\sigma'}(x) \quad \mathsf{p}_\sigma(x) = \mathsf{d}_{\sigma'}(x) \quad \mathsf{d}_\sigma(x) = \mathsf{p}_{\sigma'}(x) \\ \forall z \in \Delta, y \in \Delta'. \mathsf{c}_\sigma(x) = \mathsf{c}_{\sigma'}(z) \Rightarrow \mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \mathsf{d}_\sigma(z), \mathsf{d}_{\sigma'}(y) \notin \rho \\ \forall x \in \eta. \forall y \in \eta'. x = y \Rightarrow \eta(x) \neq \eta'(y) \end{cases}$$

The last line states that equal bound names in P and Q must denote distinct destination roles.

(comp_d)

$$\frac{P_L \vdash_{\mathsf{CL}}^{\sigma,\eta} \Delta, x{:}A \qquad Q_L \vDash_{\rho}^{\sigma',\eta'} \Delta', x{:}A^{\perp}; \Gamma \quad (\dagger)}{(\boldsymbol{\nu} x)(P_L \mid Q_L) \vDash_{\rho'}^{(\sigma' \cup \sigma) \setminus \{x\}, \eta \cup \eta'} \Delta, \Delta'; \Gamma, [\![P_L]\!]_\sigma^\eta}$$

$$\forall z \in \Delta; y \in \Delta'. \mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \mathsf{d}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y)$$
$$\forall z \in \Delta, x; y \in \Delta', x. \mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y)$$

We assume, wlog, that if $x \in bn(P_L)$ and $x \in bn(Q_L)$ then the two processes eventually exchange the bound name x. In (comp_d), the additional conditions state that for channels mapped to the same MP session ($\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y)$): (1) since P_L and Q_L may only interact via channel x, σ and σ' cannot map disjoint names to dual role pairings, denoting communication in MP; (2) the same role is not split across different threads.

Proposition 6.2. *If* $P \vDash_\rho^{(\sigma,\eta)} \Delta; \Gamma$ *then* $\sigma(\eta(\rho(P))) \vdash_{\mathsf{MP}} \Gamma$.

Example 6.1. To illustrate the mapping for delegation, consider:

$$P \triangleq \overline{x}(y).(y(n).y(m).z\langle n\rangle.\mathbf{0} \mid \mathbf{0}) \quad Q \triangleq x(y).y\langle 0\rangle.y\langle 1\rangle.\mathbf{0}$$

with the following typings:

$$P \vdash_{\mathsf{CL}} x{:}(\mathsf{nat} \parr \mathsf{nat} \parr \mathbf{1}) \otimes \mathbf{1}, z{:}\mathsf{nat} \otimes \mathbf{1}$$
$$Q \vdash_{\mathsf{CL}} x{:}(\mathsf{nat} \otimes \mathsf{nat} \otimes \perp) \parr \perp, w{:}\mathbf{1}$$

We can produce mappings σ and σ' such that:

$$\begin{array}{lcl} \sigma(P) & = & s[\mathsf{p}][\mathsf{q}]\langle y\rangle; (y[\mathsf{s}](n); y[\mathsf{s}](m); s[\mathsf{t}][\mathsf{v}]\langle n\rangle; \mathbf{0} \mid \mathbf{0}) \\ \sigma'(Q) & = & s[\mathsf{q}][\mathsf{p}](y); y[\mathsf{r}]\langle 0\rangle; y[\mathsf{r}]\langle 1\rangle; \mathbf{0} \end{array}$$

and we have that:

$$(\nu x)(P \mid Q) \vDash^{\sigma'',\eta} z{:}\mathsf{nat} \otimes \mathbf{1}, w{:}\mathbf{1}; s[\mathsf{p}]{:}T_1, y{:}T, s[\mathsf{q}]{:}T_2, s[\mathsf{t}]{:}T_3$$
$$\text{with } T = \mathsf{s}{\downarrow}(\mathsf{nat}); \mathsf{s}{\downarrow}(\mathsf{nat}); \mathsf{end} \quad T_1 = \mathsf{q}{\uparrow}(T); \mathsf{end}$$
$$T_2 = \mathsf{p}{\downarrow}(\mathsf{r}{\downarrow}(\mathsf{nat}); \mathsf{r}{\downarrow}(\mathsf{nat}); \mathsf{end}); \mathsf{end} \quad T_3 = \mathsf{v}{\uparrow}(\mathsf{nat}); \mathsf{end}$$

Note that mapping z to $s[\mathsf{p}][\mathsf{t}]$, for instance, would not allow for a valid composition of the two processes since we would have the role p of session s spread across two threads. Likewise, mapping z to $s[\mathsf{t}][\mathsf{q}]$ would disallow the composition of P and Q since it would require the two processes to share two distinct channel names. ◇

6.1 Interconnection Networks of CLL with Delegation

We carry out a similar development to that of § 5, showing that the topology of the interconnection networks denoted by our encoding cannot form cycles between more than two roles. We extend the syntax of global types with session send (resp. receive) prefixes $\mathsf{p} \rightsquigarrow \mathsf{q}{:}\uparrow(T).G$ (resp. $\mathsf{p} \rightsquigarrow \mathsf{q}{:}\downarrow(T).G$). The fundamental difference is partial global type generation, which is now inductive on the structure of the CLL process, collecting the principal and destination roles from σ, for each MP channel used in the encoding.

Definition 6.4 (Generating Partial Global Types)**.** Given $P \vdash_{\mathsf{CL}}^{(\sigma,\eta)} \Delta$ we generate its partial global type wrt a multiparty session channel s, written $\#_\eta^\sigma(P)(s)$ by induction on the structure of P (the top case is chosen if $x \in \sigma, \mathsf{c}_\sigma(x) = s$, the bottom otherwise):

$$\#_\eta^\sigma(x\langle y\rangle.(P_1 \mid P_2))(s) \triangleq \begin{cases} \mathsf{p}_\sigma(x) \rightsquigarrow \mathsf{d}_\sigma(x) : \uparrow([\![P_1]\!]_\sigma^\eta(y)).\mathsf{fuse}(\#_\eta^\sigma(P_1)(s), \#_\eta^\sigma(P_2)(s)) \\ \mathsf{fuse}(\#_\eta^\sigma(P_1)(s), \#_\eta^\sigma(P_2)(s)) \end{cases}$$

We show only the case for channel passing (see § A.9 for the other cases). Let $\mathcal{C}$ be the set of session channels in the image of σ. We denote by $\#_\eta^\sigma(P)$ the set $\bigcup(\#_\eta^\sigma(P)(s))_{s \in \mathcal{C}}$. ◇

We need to prove that fuse can generate partial global types due to session output involving parallel composition. Extending the $\vDash$ judgment with partial global types to $P \vDash_\rho^{\sigma,\eta} \Delta; \Gamma; \mathcal{G}$ we can then show the following results.

Theorem 6.1. *Let* $P \vDash_\rho^{\sigma,\eta} \Delta; \Gamma; \mathcal{G}$ *and* $\Delta = \emptyset$ *or* Δ *containing only* $\mathbf{1}$ *or* $\perp$*. Then we have:* (1) $P \to^* \mathbf{0}$*;* (2) $\mathsf{fuse}(\mathcal{G})$ *at each session is well-formed and deadlock-free; and* (3) *the interconnection network graph for* $\mathsf{fuse}(\mathcal{G})$ *for each session is acyclic.*

Notice that Theorem 6.1 offers a typing system which guarantees deadlock-freedom in MP with interleaved multiparty sessions (which is *not* usually satisfied in multiparty session calculi [14]).

7. Exponentials (replication)

Typing rules and translation Full classical linear logic has two exponential modalities, written $!A$ and $?A$, where $(!A)^\perp = ?(A^\perp)$ and $(?A)^\perp = !(A^\perp)$, governed by the following rules:

$$(!)\frac{P \vdash_{\mathsf{CL}} \Gamma; y{:}A}{!x(y).P \vdash_{\mathsf{CL}} \Gamma; x{:}!A} \qquad (?)\frac{P \vdash_{\mathsf{CL}} \Gamma, u{:}A; \Delta}{P\{x/u\} \vdash_{\mathsf{CL}} \Gamma; \Delta, x{:}?A}$$

$$(\mathsf{copy})\frac{P \vdash_{\mathsf{CL}} \Gamma, u{:}A; \Delta, y{:}A}{\overline{u}\langle y\rangle.P \vdash_{\mathsf{CL}} \Gamma, u{:}A; \Delta} \qquad (\mathsf{cut}^!)\frac{P \vdash_{\mathsf{CL}} \Gamma; x{:}A \quad Q \vdash_{\mathsf{CL}} \Gamma, u{:}A^\perp; \Delta}{(\boldsymbol{\nu} u)(!u(x).P \mid Q) \vdash_{\mathsf{CL}} \Gamma; \Delta}$$

We extend the typing judgment with an unrestricted context region Γ, subject to contraction and weakening. The process at the root of rule (!) denotes an input-guarded replication on channel x. Note how no linear channels may be used by process P. Rule (?) moves sessions of type $?A$ to the unrestricted context region. Rules (copy) and ($\mathsf{cut}^!$) trigger replicated inputs (which may be used an arbitrary number of times) on fresh sessions and allow for persistent composition, respectively. The operational semantics for input-guarded replication are standard [7] (see § A.1 for a precise definition). In MP, we extend the type syntax with:

$$S, T \quad ::= \quad \tilde{\mathsf{p}}!(T) \mid \mathsf{p}?(T)$$

where $\tilde{\mathsf{p}}!(T)$ denotes a replicated session of type T meant to be used by the set of roles $\tilde{\mathsf{p}}$. The type $\mathsf{p}?(T)$ denotes a client of a replicated session of type T offered by role p. We mirror the typing rules above with the appropriate syntax (where Γ denotes a *shared* context region, also subject to contraction and weakening):

$$(\mathsf{accept})\frac{P \vdash_{\mathsf{MP}} \Gamma; y{:}T}{!c[\tilde{\mathsf{p}}](y); P \vdash_{\mathsf{MP}} \Gamma; c{:}\tilde{\mathsf{p}}!(T)} \qquad (\mathsf{move})\frac{P \vdash_{\mathsf{MP}} \Gamma, u{:}\mathsf{p}?(T); \Delta}{P\{c/u\} \vdash_{\mathsf{MP}} \Gamma; \Delta, c{:}\mathsf{p}?(T)}$$

$$(\mathsf{req})\frac{Q \vdash_{\mathsf{MP}} \Gamma, u{:}\mathsf{p}?(T); \Delta, y{:}T}{?\overline{u}[\mathsf{p}]\langle y\rangle; Q \vdash_{\mathsf{MP}} \Gamma, u{:}\mathsf{p}?(T); \Delta}$$

$$(\mathsf{comp})\frac{P \vdash_{\mathsf{MP}} \Gamma; \Delta_1 \quad Q \vdash_{\mathsf{MP}} \Gamma; \Delta_2 \quad \Gamma; \Delta_1 \asymp \Gamma; \Delta_2}{P \mid Q \vdash_{\mathsf{MP}} \Gamma; \Delta_1, \Delta_2}$$

The $!c[\tilde{\mathsf{p}}](y).P$ construct denotes a replicated input on c meant to be used by by roles $\tilde{\mathsf{p}}$ with the corresponding *fresh* output $(?\overline{c}[\mathsf{q}]\langle y\rangle; Q)$. As in CLL, replicated inputs may be used an arbitrary number of times. The composition rule is extended with a compatibility check on the typing contexts, written $\Gamma; \Delta_1 \asymp \Gamma; \Delta_2$, which checks that the channel role assignment in usages of a replicated session are consistent with the ascribed typings. The compatibility check is defined by the homomorphic extension of the following rule (and undefined otherwise):

$$s[\mathsf{p}]{:}\tilde{\mathsf{q}}!(T) \asymp s[\mathsf{q}_i]{:}\mathsf{p}?(T) \quad \mathsf{q}_i \in \tilde{\mathsf{q}}$$

The operational semantics of replication are identical to CLL.

For free names, we use the same one role per thread restrictions of Def. 3.1, allowing for different session names in the same thread and multiple destination roles for replicated channel names (otherwise the mapping of replicated inputs would be degenerate), and Def. 6.1 for bound names. We note that for the case of replicated input channels, we have that p_σ denotes a single role and d_σ denotes a

set of roles that may use the replicated service. Let $P_L \vdash^{(\sigma,\eta)}_{\mathsf{CL}} \Gamma;\Delta$, with all free and bound names distinct. We generate a local typing context Θ with $\eta(\sigma(P_L)) \vdash_{\mathsf{MP}} \Theta$ by induction on the structure of P_L, written $[\![P]\!]^\eta_\sigma$ (we show only the new cases):

$$\begin{array}{rcl}[\![!x(y).P]\!]^\eta_\sigma & \triangleq & c{:}\mathsf{d}_\varphi(x)!(\overline{[\![P]\!]^\eta_\sigma(y)}) \uplus ([\![P]\!]^\eta_\sigma \setminus \eta(y)) \\ [\![\overline{u}\langle y\rangle.P]\!]^\eta_\sigma & \triangleq & c{:}\mathsf{d}_\varphi(u)?([\![P]\!]^\eta_\sigma(y)) \uplus ([\![P]\!]^\eta_\sigma \setminus \eta(y))\end{array}$$

The rules above are similar to those from § 6, where we compute the local type for the replicated session inductively, by the usage of the bound name y in the continuation process P. We note that in the top rule, d_σ denotes a set of roles. We define the judgment $P \Vdash^{\sigma,\eta}_\rho \Gamma;\Delta;\Theta$ for process composition, now with an additional composition rule to account for *shared* names:

$$\frac{P_L \vdash^{\sigma,\eta}_{\mathsf{CL}} \Gamma;\Delta}{P_L \Vdash^{\sigma,\eta}_{\emptyset} \Gamma;\Delta;[\![P_L]\!]^\eta_\sigma}\ (\mathsf{thread}_r)$$

$$\frac{P_L \vdash^{\sigma,\eta}_{\mathsf{CL}} \Gamma;\Delta,x{:}A \quad Q_L \Vdash^{\sigma',\eta'}_{\rho} \Gamma;\Delta',x{:}A^\perp;\Theta \quad (\dagger)}{(\boldsymbol{\nu} x)(P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{x\},\eta\cup\eta'}_{\rho'} \Gamma;\Delta,\Delta';\Theta,[\![P_L]\!]^\eta_\sigma}\ (\mathsf{comp}_r)$$

$$\frac{P_L \vdash^{\sigma,\eta}_{\mathsf{CL}} \Gamma;x{:}A \quad Q_L \Vdash^{\sigma',\eta'}_{\rho} \Gamma,u{:}A^\perp;\Delta;\Theta \quad (\dagger)}{(\boldsymbol{\nu} u)(!u(x).P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{x,u\},\eta\cup\eta'}_{\rho'} \Gamma;\Delta;\Theta,[\![P_L]\!]^\eta_\sigma}\ (\mathsf{comp}^!_r)$$

Note that the condition of (comp_r) and $(\mathsf{comp}^!_r)$ is $(\dagger)$ in (comp_d) in § 6, where for $(\mathsf{comp}^!_r)$ we consider u instead of x in $\dagger$. Moreover, we require σ and σ' to match for channels in Γ.

Proposition 7.1. *If* $P \Vdash^{(\sigma,\eta)}_\rho \Delta;\Gamma;\Theta$ *then* $\sigma(\eta(\rho(P))) \vdash_{\mathsf{MP}} \emptyset;\Theta$.

Example 7.1 (Replication)**.** The following processes are identical to an example of [8] that is not typable in the global progress type system of [10].

$$\begin{array}{rcl}P & \triangleq & !x(y).y(n).\mathbf{0} \vdash_{\mathsf{CL}} \cdot;x{:}!(\mathsf{int}\parr\mathbf{1}) \\ Q & \triangleq & !z(w).w\langle 1\rangle.\mathbf{0} \vdash_{\mathsf{CL}} \cdot;z{:}!(\mathsf{int}\otimes\mathbf{1}) \\ R & \triangleq & \overline{x}(y).\overline{z}(w).w(n).y\langle n\rangle.\mathbf{0} \vdash_{\mathsf{CL}} \emptyset;x{:}?(\mathsf{int}\otimes\bot),z{:}?(\mathsf{int}\parr\bot),v{:}\mathbf{1}\end{array}$$

We define mappings σ,σ_1,σ_2 and η,η_1,η_2 such that:

$$\begin{array}{rcl}\eta(\sigma(P)) & = & !s[\mathsf{p}][\mathsf{r}](y);y[\mathsf{s}](n);\mathbf{0} \\ \eta_1(\sigma_1(Q)) & = & !s'[\mathsf{q}][\mathsf{b}](w);\underline{w[\mathsf{r}]\langle 1\rangle};\mathbf{0} \\ \eta_2(\sigma_2(R)) & = & ?s[\mathsf{r}][\mathsf{p}]\langle y\rangle;?s'[\mathsf{b}][\mathsf{q}]\langle w\rangle;w[\mathsf{r}'](n);y[\mathsf{s}']\langle n\rangle;\mathbf{0}\end{array}$$

and so produce the following local typings:

$$\begin{array}{rcl}\#^\sigma_\eta(P) & = & s[\mathsf{p}]{:}\mathsf{r}!(\mathsf{s}\uparrow(\mathsf{int})) \quad \#^{\sigma_1}_{\eta_1}(Q) = s'[\mathsf{q}]{:}\mathsf{b}!(\mathsf{r}\downarrow(\mathsf{int})) \\ \#^{\sigma_2}_{\eta_2}(R) & = & s[\mathsf{r}]{:}\mathsf{p}?(\mathsf{s}'\uparrow(\mathsf{int})),s'[\mathsf{b}]{:}\mathsf{q}?(\mathsf{r}'\downarrow(\mathsf{int}))\end{array}$$

Then $(\boldsymbol{\nu} x,z)(P \mid Q \mid R) \Vdash \emptyset;v{:}\mathbf{1};\#^\sigma_\eta(P),\#^{\sigma_1}_{\eta_1}(Q),\#^{\sigma_2}_{\eta_2}(R)$. ◇

7.1 Interconnection Networks of CLL with Replication

Partial global type generation for replication is analogous to that of § 6, insofar as we generate the replicated session type inductively and need to handle potentially multiple session channels in the image of the translation. We extend the syntax of partial global types with a dedicated replication construct $\tilde{\mathsf{p}} \rightsquigarrow \mathsf{q} : !(T).G$, with the corresponding dual $\mathsf{p} \rightsquigarrow \mathsf{q} : ?(T).G$, which fuse to the corresponding $\tilde{\mathsf{p}} \to \mathsf{q} : *(T).G$ global type, which denotes that role q hosts the replicated behaviour T, to be used by roles $\tilde{\mathsf{p}}$ an arbitrary (finite) number of times.

Definition 7.1 (Generating partial global types)**.** Given P such that $P \vdash^{(\sigma,\eta)}_{\mathsf{CL}} \Gamma;\Delta$ we generate its partial global type wrt a multiparty session channel s, written $\#_{(\sigma,\eta)}(P)(s)$ by induction on the structure of P as follows (the top case is chosen if $x \in \sigma, \mathsf{c}_\sigma(x) = s$, the bottom otherwise – we write only the new cases):

$$\begin{array}{rcl}\#^\sigma_\eta(!x(y).P_1)(s) & \triangleq & \begin{cases}\mathsf{d}_\sigma(x) \rightsquigarrow \mathsf{p}_\sigma(x) :!([\![P_1]\!]^\eta_\sigma(y)).\#^\sigma_\eta(P_1)(s) \\ \#^\sigma_\eta(P_1)(s)\end{cases} \\ \#^\sigma_\eta(\overline{u}\langle y\rangle.P_1)(s) & \triangleq & \begin{cases}\mathsf{p}_\sigma(u) \rightsquigarrow \mathsf{d}_\sigma(u) :?([\![P_1]\!]^\eta_\sigma(y)).\#^\sigma_\eta(P_1)(s) \\ \#^\sigma_\eta(P_1)(s)\end{cases}\end{array}$$

We extend to $\Vdash^{\sigma,\eta}_\rho \Gamma;\Delta;\Theta;\mathcal{G}$ using the following extension to the definition of fuse:

$$\begin{array}{l}\mathsf{fuse}(\tilde{\mathsf{p}} \rightsquigarrow \mathsf{q} :!(T).\mathsf{end},\mathsf{end}) = \tilde{\mathsf{p}} \to \mathsf{q} : *(T).\mathsf{end} \\ \mathsf{fuse}(\tilde{\mathsf{p}} \to \mathsf{q} : *(T).\mathsf{end},\mathsf{end}) = \tilde{\mathsf{p}} \to \mathsf{q} : *(T).\mathsf{end} \\ \mathsf{fuse}(\tilde{\mathsf{p}} \rightsquigarrow \mathsf{q} :!(T).G_1,\mathsf{p} \rightsquigarrow \mathsf{q} :?(T).G_2) \\ \qquad = \mathsf{fuse}(\tilde{\mathsf{p}} \rightsquigarrow \mathsf{q} :!(T).G_1,G_2) \\ \mathsf{fuse}(\tilde{\mathsf{p}} \to \mathsf{q} : *(T).G_1,\mathsf{p} \rightsquigarrow \mathsf{q} :?(T).G_2) \\ \qquad = \mathsf{fuse}(\tilde{\mathsf{p}} \to \mathsf{q} : *(T).G_1,G_2)\end{array}$$

Definition 7.2 (Live Process)**.** A process P is live, written $live(P)$ iff $P \equiv (\boldsymbol{\nu}\tilde{a})(\pi.Q \mid R)$, for some R, sequences of names $\tilde{a}$ and a *non-replicated* guarded process $\pi.Q$. ◇

Theorem 7.1. *Let* $P \Vdash^{\emptyset,\eta}_\rho \Gamma;\Delta;\Theta;\mathcal{G}$ *and* $\Delta = \emptyset$ *or* Δ *contains only* $\mathbf{1}$ *or* $\bot$*. We have:* (1) *If* $live(P)$ *then* $P \to P'$*;* (2) $\mathsf{fuse}(\mathcal{G})$ *for each session is well-formed and deadlock-free; and* (3) *the interconnection net. graph for* $\mathsf{fuse}(\mathcal{G})$ *for each session is acyclic.*

8. Multicut in CLL

As discussed in § 1, and made precise throughout our development, the inability to compose processes that interact using more than one channel – dubbed *multicut* – significantly limits the admissible interconnection network topologies. Logically, such a form of unrestricted multicut is *unsound* since it destroys the crucial cut elimination property. At the operational level, this results in the loss of deadlock freedom in a process interpretation of logic.

In this section we make use of partial multiparty compatibility (PMC) to develop a new multicut rule for CLL *without delegation*. While we allow for composed processes to share multiple dual channels, we restrict composition by requiring the induced partial global types to be *fuseable* (or PMC). With this restriction in place we recover the deadlock freedom property. Using the mapping of § 3, identifying cut-free processes with the same role, we redefine the judgment $P \Vdash^\sigma_\rho \Delta;\Gamma$ as follows (where $\mathsf{fuse}(\Gamma_1,\Gamma_2)$ is defined as $\mathsf{fuse}(\cdots(\mathsf{fuse}(\#_{\sigma_1}(T_1),\#_{\sigma_2}(T_2)),\#_{\sigma_3}(T_3)))\cdots,\#_{\sigma_n}(T_n))$ where $\mathsf{c}_{\sigma_i}[\mathsf{p}_{\sigma_i}]{:}T_i \in \Gamma_1 \cup \Gamma_2$:

$$\frac{P_L \Vdash^\sigma_\rho \Delta,x_1{:}A_1,\ldots,x_n{:}A_n;\Gamma_1 \quad Q_L \Vdash^{\sigma'}_{\rho'} \Delta',x_1{:}A_1^\perp,\ldots,x_n{:}A_n^\perp;\Gamma_2 \quad \mathsf{fuse}(\Gamma_1,\Gamma_2)\ \text{defined} \quad \rho' \cap \rho = \emptyset \quad (\ddagger)}{(\boldsymbol{\nu} x_1,\ldots,x_n)(P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{x_1,\ldots,x_n\}}_{\rho''} \Delta,\Delta';\Gamma_1,\Gamma_2}\ (\mathsf{Mcomp})$$

where $(\ddagger)$ extends $(\star)$ in Def. 3.3 as follows (the red letters highlight differences).

$$\ddagger \begin{cases}\rho'' = \rho \cup \rho' \cup_i (x_i, \mathsf{c}_\sigma[\mathsf{p}_\sigma(x_i)][\mathsf{d}_\sigma(x_i)]) \\ \mathsf{c}_\sigma = \mathsf{c}_{\sigma'} \quad \mathsf{p}_\sigma(x_i) = \mathsf{d}_{\sigma'}(x_i) \quad \mathsf{d}_\sigma(x_i) = \mathsf{p}_{\sigma'}(x_i) \\ \forall y \in \Delta.z \in \Delta'.\mathsf{d}_\sigma(z),\mathsf{d}_\sigma(y) \notin \rho,\rho' \\ \wedge \mathsf{p}_\sigma(y) \neq \mathsf{p}_\sigma(z) \wedge \mathsf{p}_\sigma(y) \neq \mathsf{d}_{\sigma'}(z) \wedge \mathsf{p}_{\sigma'}(z) \neq \mathsf{d}_\sigma(y)\end{cases}$$

Unlike in the previous sections, rule (Mcomp) is *symmetric*, matching a generalised cut rule. Beyond the basic restrictions such as lack of overlapping bound names in the two processes and that the role assignments for each of the composed x_i channels must be dual, we just maintain basic invariants akin to those of composition for § 6: non-composed channels cannot have dual role assignments, the same role may not be split across different threads and role assignments cannot capture those of previously bound names. To enable multicut, the condition $\mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y)$ is dropped.

Theorem 8.1 (Single Participant per Thread)**.** *Let* $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$ *and* $\Delta = \emptyset$ *or* Δ *contains only* $\mathbf{1}$ *or* $\bot$*. Then we have:* (1) $P \rightarrow^{*} \mathbf{0}$*; and* (2) $\mathsf{fuse}(\mathcal{G})$ *is well-formed and deadlock-free.*

(1) above is by the fact that the calculus preserves a single multiparty session [14, § 5]; (2) is by construction. The system (and Theorem 8.1) above is only defined in a setting *without* delegation. In the presence of delegation (and thus interleaved multiparty sessions) we can construct a well-typed (and fuseable) composition of processes that deadlock. Intuitively, local types (and partial global types) fail to capture the linear cross-*session* dependencies, allowing us to interleave sessions in a non-deadlock free way.

However, in the presence of interleaved *shared* sessions (represented in our framework through replication), multicut preserves deadlock-freedom. By considering the channel mapping of § 7 we can define a multicut rule that accounts for replication by generalising the judgment above to $P \Vdash^{\sigma,\eta}_{\rho} \Gamma; \Delta; \Theta$ and adding the restriction to the η mapping from † to the (Mcomp) rule. (1) below follows due to the calculus satisfying the well-linked property in [14, § 5]; (2) is again by construction.

Theorem 8.2 (Exponential)**.** *Let* $P \Vdash^{\emptyset,\eta}_{\rho} \Gamma; \Delta; \Theta; \mathcal{G}$ *and* $\Delta = \emptyset$ *or* Δ *contains only* $\mathbf{1}$ *or* $\bot$*. Then we have:* (1) *if* $live(P)$ *then* $P \rightarrow P'$ (2) $\mathsf{fuse}(\mathcal{G})$ *is well-formed and deadlock-free.*

Example 8.1 (Two Buyer Protocol [14])**.** The 2-buyer protocol aims to define the coordinated interactions of two Buyer agents seeking to buy from a Seller agent.

$$\begin{array}{lcl} \mathsf{Seller} & \triangleq & x(s).x\langle 32\rangle.y\langle 32\rangle.y.\mathsf{case}\{ok{:}y(s).\mathbf{0}, nok{:}\mathbf{0}\} \\ \mathsf{Buyer}_1 & \triangleq & x\langle\text{"xpto"}\rangle.x(n).z\langle n/2\rangle.\mathbf{0} \\ \mathsf{Buyer}_2 & \triangleq & y(n).z(m).y.ok; y\langle\text{"icl"}\rangle.\mathbf{0} \end{array}$$

The process Seller uses channels x and y to interact with Buyer_1 and Buyer_2, respectively. Process Buyer_1 uses channel z to interact with Buyer_2. Consider σ, σ_1 and σ_2 such that:

$$\begin{array}{lcl} \sigma(\mathsf{Seller}) & = & s[\mathsf{S}][\mathsf{B1}](s); s[\mathsf{S}][\mathsf{B1}]\langle 32\rangle; s[\mathsf{S}][\mathsf{B2}]\langle 32\rangle; \\ & & s[\mathsf{S}][\mathsf{B2}] \triangleright \{ok{:}s[\mathsf{S}][\mathsf{B2}](s); \mathbf{0}, nok{:}\mathbf{0}\} \\ \sigma_1(\mathsf{Buyer}_1) & = & s[\mathsf{B1}][\mathsf{S}]\langle\text{"xpto"}\rangle; s[\mathsf{B1}][\mathsf{S}](n); s[\mathsf{B1}][\mathsf{B2}]\langle n/2\rangle; \mathbf{0} \\ \sigma_2(\mathsf{Buyer}_2) & = & s[\mathsf{B2}][\mathsf{S}](n); s[\mathsf{B2}][\mathsf{B1}](m); s[\mathsf{B2}][\mathsf{S}] \triangleleft ok; \\ & & s[\mathsf{B2}][\mathsf{S}]\langle\text{"icl"}\rangle; \mathbf{0}. \end{array}$$

While in the previous development the processes above would *not* be composable, we can now compose them via multicut given that:

$$\begin{array}{lcl} \#_{\sigma}(\mathsf{Seller}) & = & \mathsf{B1} \rightsquigarrow \mathsf{S} :\downarrow (\mathsf{str}).\mathsf{S} \rightsquigarrow \mathsf{B1} :\uparrow (\mathsf{int}). \\ & & \mathsf{S} \rightsquigarrow \mathsf{B2} :\uparrow (\mathsf{int}).\mathsf{B2} \rightsquigarrow \mathsf{S} : \\ & & \&\{ok{:}\mathsf{B2} \rightsquigarrow \mathsf{S} :\downarrow (\mathsf{str}).\mathsf{end}, nok{:}\mathsf{end}\} \\ \#_{\sigma_1}(\mathsf{Buyer}_1) & = & \mathsf{B1} \rightsquigarrow \mathsf{S} :\uparrow (\mathsf{str}).\mathsf{S} \rightsquigarrow \mathsf{B1} :\downarrow (\mathsf{int}). \\ & & \mathsf{B1} \rightsquigarrow \mathsf{B2} :\uparrow (\mathsf{int}).\mathsf{end} \\ \#_{\sigma_2}(\mathsf{Buyer}_2) & = & \mathsf{S} \rightsquigarrow \mathsf{B2} :\downarrow (\mathsf{int}).\mathsf{B1} \rightsquigarrow \mathsf{B2} :\downarrow (\mathsf{int}). \\ & & \mathsf{B2} \rightsquigarrow \mathsf{S} : \oplus\{ok : \mathsf{B2} \rightsquigarrow \mathsf{S} :\uparrow (\mathsf{int}).\mathsf{end}\} \end{array}$$

Let $G_1 = \#_{\sigma}(\mathsf{Seller}), G_2 = \#_{\sigma_1}(\mathsf{Buyer}_1)$ and $G_3 = \#_{\sigma_2}(\mathsf{Buyer}_2)$. We can see that:

$$\begin{array}{c} \mathsf{fuse}(\mathsf{fuse}(G_1, G_2), G_3) = \mathsf{fuse}(\mathsf{fuse}(G_1, G_3), G_2) \\ = \mathsf{fuse}(\mathsf{fuse}(G_2, G_3), G_1) = \\ \mathsf{B1} \rightarrow \mathsf{S} : (\mathsf{str}).\mathsf{S} \rightarrow \mathsf{B1} : (\mathsf{int}).\mathsf{S} \rightarrow \mathsf{B2} : (\mathsf{int}).\mathsf{B1} \rightarrow \mathsf{B2} : (\mathsf{int}). \\ \mathsf{B2} \rightarrow \mathsf{S} : (ok : \mathsf{B2} \rightarrow \mathsf{S} : (\mathsf{str}), nok : \mathsf{end}) \end{array}$$

9. Related and future work

The seminal work of [1] studies acyclicity of proofs in a computational interpretation of linear logic where names are used exactly once. With session types, channel names can be reused *multiple* times in a cyclic way (even in CLL defined in § 2.1), insofar as two processes may both send and receive along the same channel. This feature, combined with dynamic name creation in CLL, makes the study of interconnection networks (and deadlock-freedom in general) more challenging. The term "interconnection networks" in this paper originates from [2], which studies categorical models that enable cyclic networks corresponding to multicut rules. A similar discussion can be found in [27]. In our work, we make their informal observations precise, defining interconnection networks and proving the negative results in a session typed setting.

The works of [8] and [6] are the most related to our own. The work [8] proposed a typing system for CLL extended to multiparty session primitives. The main differences from our work are: (1) the work in [8] does not (aim to) study interconnection networks; (2) the projection-based approach in [8] relies on a special form of global types annotated by modalities, and linear logic propositions annotated by participants. Our synthesis-based approach requires no change to the syntax of processes, types, nor typing rules (except multicut) of CLL; (3) the multicut rule is applied to a complete set of processes in a multiparty session, directly using information of global types, while ours is a cut between two processes; and (4) the projection-based cut rule [8] is more limited than our synthesis-based rule. This is because a global type built by PMC soundly and completely characterises all deadlock-free traces observable from local type configurations (Theorem 4.1). For example, our system can type Example 3.2, which is untypable in [8], noting that the traces of local types generated in Example 5.2 are deadlock-free.

Caires and Pérez [6] show that the binary session interpretation of intuitionistic linear logic can encode the behaviour of multiparty sessions (up to a typed *barbed congruence*). While the goals of this paper are quite different, focusing on interconnection network topologies of CLL processes, we note that their work does not contradict our negative results. The encoding of [6] consists of adding *another* participant (i.e. a centralised coordinator), that mediates all possible interactions between role implementations. Thus, a topology such as that of Fig. 1a is realised by disconnecting all participants in the network, adding a new (fourth) participant c, and then connecting each participant solely with c (i.e. a tree topology). In this sense, their encoding does not preserve interconnection network topologies of global types, which is key in our criteria.

Type systems for progress in concurrent processes are a vast area of research. Usually such a typing system requires sophisticated information on channel usages (e.g. [15]) or causality among channels (e.g. [10]). While the main emphasis of this work is *not* the development of a type system for progress, our encodings provide a discipline for progress in restricted interleaved multiparty sessions. See Example 7.1 which is typable in our system but not in a typing system for progress in multiparty sessions [10]. Our development of multicut, which is achieved as a simple composition rule guided by PMC, ensures deadlock-freedom in the presence of interleaved *shared* sessions. We plan to investigate a notion of multicut for CLL that ensures progress of interleaved *linear* sessions. This feature would be handled by using global types with session channels [14] and synthesising a single global type consisting of several sessions, with the additional linearity checking from [14].

Multiparty compatibility (MC) properties are studied in [4, 11, 17] where a global type is synthesised from communicating automata, assuming all participants are initially present. These global synthesis methods are not applicable to CLL where the typing rule for composition is binary. Investigations of partial compatibility for choreographies [17] and timers [4] would allow us to capture larger classes of connectability (with timing information) in the CLL framework.

References

[1] S. Abramsky. Computational interpretations of linear logic. *Theoret. Comput. Sci.*, 111(1–2):3–57, 1993.

[2] S. Abramsky, S. J. Gay, and R. Nagarajan. Specification structures and propositions-as-types for concurrency. In *Logics for Conc.*:5–40, 1995.

[3] G. Bellin and P. Scott. On the π-calculus and linear logic. *Theoret. Comput. Sci.*, 135(1):11–65, 1994.

[4] L. Bocchi, J. Lange and N. Yoshida, Meeting Deadlines Together, In *CONCUR'15*:283–296, 2015.

[5] L. Caires and F. Pfenning. Session types as intuitionistic linear propositions. In *CONCUR'10*: 222–236, 2010.
[6] L. Caires and J. A. Pérez. A typeful characterization of multiparty structured conversations based on binary sessions. *CoRR*, abs/1407.4242, 2014.
[7] L. Caires, F. Pfenning, and B. Toninho. Linear logic propositions as session types. *MSCS*, avail. on CJO 2014:1-57, 2014.
[8] M. Carbone, F. Montesi, C. Schürmann, and N. Yoshida. Multiparty session types as coherence proofs. In *CONCUR'15*:412–426, 2015.
[9] M. Coppo, M. Dezani-Ciancaglini, L. Padovani, and N, Yoshida. A gentle introduction to multiparty asynchronous session types. In *SFM-15:MP*:146–178, 2015.
[10] M. Coppo, M. Dezani-Ciancaglini, N. Yoshida, and L. Padovani. Global Progress for Dynamically Interleaved Multiparty Sessions. *MSCS*, 760:1–65, 2015.
[11] P.-M. Deniélou and N. Yoshida. Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types. In *ICALP'13*:174–186, 2013.
[12] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–102, 1987.
[13] K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*:22–138, 1998.
[14] K. Honda, N. Yoshida, and M. Carbone. Multiparty Asynchronous Session Types. In *POPL'08*, 273–284, 2008. A full version in *JACM*.
[15] A. Igarashi, N. Kobayashi. A generic type system for the pi-calculus. *Theoret. Comput. Sci.*, 311(1-3):121–163, 2004.
[16] L. Jia, H. Gommerstadt, and F. Pfenning. Monitors and blame assignment for higher-order session types. In *POPL'16*:582–594, 2016.
[17] J. Lange, E. Tuosto, and N. Yoshida. From Communicating Machines to Graphical Choreographies. In *POPL'15*:221–232, 2015.
[18] S. Lindley and J. G. Morris. A semantics for propositions as sessions. In *ESOP'15*:560–584, 2015.
[19] Links homepage. `http://groups.inf.ed.ac.uk/links/`.
[20] D. Orchard and N. Yoshida. Effects as sessions, sessions as effects. *POPL'16*:568–581, 2016.
[21] J. A. Pérez, L. Caires, F. Pfenning, and B. Toninho. Linear logical relations for session-based concurrency. In *ESOP'12*:539–558, 2012.
[22] F. Pfenning and D. Griffith. Polarized substructural session types. In *FoSSaCs'15*:3–22, 2015.
[23] Scribble Project Homepage, `www.scribble.org`.
[24] B. Toninho, L. Caires, and F. Pfenning. Functions as session-typed processes. In *FoSSaCs'12*, 2012.
[25] B. Toninho, L. Caires, and F. Pfenning. Higher-order processes, functions, and sessions: A monadic integration. In *ESOP'13*:350–369, 2013.
[26] B. Toninho, L. Caires, and F. Pfenning. Corecursion and non-divergence in session-typed processes. In *TGC'14*:159–175, 2014.
[27] P. Wadler. Propositions as sessions. In *ICFP'12*:273–286, 2012. a full version in *J. Funct. Program.*

Appendix

The structure of this appendix is as follows:

- § A contains proofs and additional definitions
  - § A.1 explains in detail the typing rules and reduction semantics for both CLL and MP systems. § A.1.2 also contains additional definitions necessary to precisely define typing for MP and projections from global types.
  - § A.2 contains proofs of results from § 3.
  - § A.3 contains proofs of results from § 4.
  - § A.4 contains proofs of results (and additional theorems) from § 5.
  - § A.5 contains proofs of results (and additional theorems) from § 6.
  - § A.6 contains proofs of results (and additional theorems) from § 7.
  - § A.7 contains proofs of results from § 8.
- § A.8 contains the precise definition of the $\biguplus$ function, used throughout our development, that combines a local type with a typing environment.
- § A.9 lists the full definition of partial global type generation for CLL with delegation.
- § B lists the extended $\vDash$ judgement used in Section 6.
- § C gives the multicut rule for replication.
- Finally, § D gives a detailed presentation (and corresponding proofs) of an alternate mapping from CLL into MP where a thread may implement multiple roles. We prove that such a renaming does not add to the expressiveness of the topologies of the resulting interconnection network.

A. Proofs and Additional Definitions

A.1 Explanation of typing rules and reduction semantics

This section explains the typing rules and semantics for the CLL and MP calculi. In the latter case, we also include some additional definitions that are needed to precisely define the typing system, namely *partial projection*, *merging* of local and binary types and *coherence*.

A.1.1 The CLL System

Recalling that the rules of Figure 2 define the judgement $P_L \vdash_{\mathsf{CL}} \Delta; \Psi$, defined *modulo the structural congruence* relation $\equiv$ of Figure 4, where Δ is a set of assigning distinct channel names of P_L to session types, we give an explanation of each rule: Rule ($\mathbf{1}$) types the inactive process with an arbitrary session channel assigned type $\mathbf{1}$; rule ($\bot$) types the dual behaviour, which just discards the no longer used name; rule ($\otimes$) accounts for the value output behaviour, typing a channel a with $\tau \otimes A$ if the process outputs along a a value M of type τ and then proceeds by using a as A; dually, rule ($\parr$) types a channel a with $\tau \parr A$ if the process performs an input on a of a value of type τ and then uses a according to type A (in § 6 we generalise both $\otimes$ and $\parr$ to emission and reception of channels, respectively, with the expected multiplicative decomposition of the context for $\otimes$); rule ($\oplus$) types channel a with $\oplus\{l_i{:}A_i\}_{i\in I}$ by having the process emit a label l_j with $j \in I$, and then using the channel a according to the type A_j in the corresponding branch; dually, rule ($\&$) types processes that wait for a choice on channel a, with type $\&\{l_i{:}A_i\}_{i\in I}$, if the process is prepared to account for any of the possible choice labels and corresponding behaviours in the

$$
\begin{array}{rcl}
a\langle M\rangle.P \mid a(x).Q & \to & P \mid Q\{M/x\} \\
a.l_j; P \mid a.\mathsf{case}\{l_i{:}Q_i\}_{i\in I} & \to & P \mid Q_j \quad (j \in I) \\
(\boldsymbol{\nu} a)([a \leftrightarrow b] \mid P) & \to & P\{b/a\} \quad (b \notin fn(P)) \\
P \equiv P' \wedge P' \to Q' \wedge Q' \equiv Q & \Rightarrow & P \to Q' \\
\overline{x}\langle y\rangle.P \mid x(y).Q & \to & (\boldsymbol{\nu} y)(P \mid Q) \\
!x(y).P \mid \overline{x}\langle y\rangle.Q & \to & !x(y).P \mid (\boldsymbol{\nu} y)(P \mid Q) \\
P \to Q & \Rightarrow & (\boldsymbol{\nu} a)P \to (\boldsymbol{\nu} a)Q
\end{array}
$$

$$
P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R)
$$

$$
(\boldsymbol{\nu} a)(P \mid Q) \equiv (\boldsymbol{\nu} a)P \mid Q \quad a \notin fn(Q) \quad P \equiv_\alpha Q \Rightarrow P \equiv Q
$$

$$
(\boldsymbol{\nu} x)(!x(y).P \mid Q) \equiv Q \quad x \notin fn(Q)
$$

Figure 4: Reduction Semantics for CLL Processes

type. Thus, the case branching construct must contain one process P_i using a according to behaviour A_i for each label in the type. Note the additive nature of the rule, where the context Δ is the same in all premises.

Finally, the (cut) rule composes two processes P and Q that use channel a with dual types A and $A^\perp$, respectively, by hiding the name a in the composed process in the conclusion of the rule (since no other process may use a). We note that Δ and Δ' are disjoint, thus the only common channel between P and Q is exactly a. Rule (id) types the forwarding construct, identifying two channel names a and b that denote dual behaviours.

Reduction Semantics The reduction semantics for CLL processes is given in Fig. 4, including the rules for delegation (§ 6) and replication (§ 7). We omit the standard compatible closure cases. The reduction rules for delegation and replication entail the generation of a *fresh* name y.

A.1.2 The MP System

To precisely define the typing system for the MP calculus, we require the following auxiliary definitions.

Definition A.1 (Partial Projection)**.** Given a local type T, we define its partial projection onto a participant p, written $T \upharpoonright \mathsf{p}$ by induction on the structure of T according to the rules of Figure 5. In all cases, if no side condition applies then partial projection is undefined.

Partial projection is defined in terms of a binary type merge operation for selections, where the *merge* $T \sqcup T'$ of T and T' is defined by $T \sqcup T \triangleq T$; and with $T = \oplus\{\mathsf{l}_i : T_i\}_{i\in I}$ and $T' = \oplus\{\mathsf{l}'_j : T'_j\}_{j\in J}$,

$$
T \sqcup T' \triangleq \oplus(\{\mathsf{l}_h : T_h\}_{h\in I\setminus J} \cup \{\mathsf{l}'_h : T'_h\}_{h\in J\setminus I} \cup \{\mathsf{l}_h : T_h \sqcup T'_h\}_{h\in I\cap J})
$$

if $\mathsf{l}_h = \mathsf{l}'_h$ for each $h \in I \cap J$; and homomorphic for other types (i.e. $\mathcal{T}[T_1] \sqcup \mathcal{T}[T_2] = \mathcal{T}[T_1 \sqcup T_2]$ where $\mathcal{T}$ is a context of local types). $T \sqcup T'$ is undefined otherwise.

Coherence, written $\mathsf{co}(\Delta)$, ensures that Δ contains a local type for each role involved in interactions in Δ. Moreover, coherence ensures that local types of interacting roles contain the necessary dual communication actions.

Definition A.2 (Coherence)**.** We say Δ is coherent (denoted by $\mathsf{co}(\Delta)$) iff for all $s[\mathsf{p}]{:}T_1$, there exists $s[\mathsf{q}]{:}T_2 \in \Delta$ such that $T_1 \upharpoonright \mathsf{q} = \overline{T_2 \upharpoonright \mathsf{p}}$; and if $s[\mathsf{p}]{:}T \in \Delta$, then for all $\mathsf{q} \in \mathsf{roles}(T)$, $s[\mathsf{q}]{:}T' \in \Delta$. ◇

The typing rules for the MP calculus are given in Figure 3, defining the judgement $P \vdash_{\mathsf{MP}} \Delta; \Psi$. Rule (vsend) types the emission of M of type τ from role p to role q, assigning $s[\mathsf{p}]$ the local type $\mathsf{q} \uparrow (\tau); T$, provided the continuation P uses $s[\mathsf{p}]$ according to type T. Dually, rule (vrecv) types the reception of a value of type

$$
\begin{array}{rcl}
(\mathsf{r}\uparrow(\tau); T) \upharpoonright \mathsf{p} & = & \begin{cases} \uparrow(\tau); (T \upharpoonright \mathsf{p}) & \text{if } \mathsf{p} = \mathsf{r} \\ T \upharpoonright \mathsf{p} & \text{otherwise} \end{cases} \\
(\mathsf{r}\downarrow(\tau); T) \upharpoonright \mathsf{p} & = & \begin{cases} \downarrow(\tau); (T \upharpoonright \mathsf{p}) & \text{if } \mathsf{p} = \mathsf{r} \\ T \upharpoonright \mathsf{p} & \text{otherwise} \end{cases} \\
(\oplus\mathsf{r}\{l_i{:}T_i\}_{i\in I}) \upharpoonright \mathsf{p} & = & \begin{cases} \oplus\{l_i{:}(T_i \upharpoonright \mathsf{p})\}_{i\in I} & \text{if } \mathsf{p} = \mathsf{r} \\ \sqcup_{i\in I}(T_i \upharpoonright \mathsf{p}) & \text{otherwise} \end{cases} \\
(\&\mathsf{r}\{l_i{:}T_i\}_{i\in I}) \upharpoonright \mathsf{p} & = & \begin{cases} \&\{l_i{:}(T_i \upharpoonright \mathsf{p})\}_{i\in I} & \text{if } \mathsf{p} = \mathsf{r} \\ \sqcup_{i\in I}(T_i \upharpoonright \mathsf{p}) & \text{otherwise} \end{cases} \\
\mathsf{end} \upharpoonright \mathsf{p} & = & \mathsf{end}
\end{array}
$$

Figure 5: Partial Projection.

$$
\begin{array}{rcl}
s[\mathsf{p}][\mathsf{q}]\langle M\rangle; P \mid s[\mathsf{q}][\mathsf{p}](x); Q & \to & P \mid Q\{M/x\} \\
s[\mathsf{p}][\mathsf{q}] \triangleleft l_j; P \mid s[\mathsf{q}][\mathsf{p}] \triangleright \{l_i{:}Q_i\}_{i\in I} & \to & P \mid Q_j \quad (j \in I) \\
s[\mathsf{p}][\mathsf{q}]\langle s'[\mathsf{p}']\rangle; P \mid s[\mathsf{q}][\mathsf{p}](x); Q & \to & P \mid Q\{s'[\mathsf{p}']/x\} \\
!s[\mathsf{p}][\tilde{\mathsf{q}}](y); P \mid ?\overline{s[\mathsf{q}][\mathsf{p}]}\langle y\rangle; Q & \to & !s[\mathsf{p}][\tilde{\mathsf{q}}](y); P \mid \\
 & & (\boldsymbol{\nu} s')(P\{s'/y\} \mid \\
 & & Q\{s'/y\}) \\
P \equiv P' \wedge P' \to Q' \wedge Q' \equiv Q & \Rightarrow & P \to Q'
\end{array}
$$

Figure 6: Reduction Semantics for MP Processes

τ, bound to x in the continuation P, received by role p and sent by q, with type $\mathsf{q} \downarrow (\tau); T$, provided P uses $s[\mathsf{p}]$ according to T. Rules (sel) and (branch) are the MP counterparts of rules ($\oplus$) and ($\&$) from CLL(Fig. 2), respectively, with the former typing the emission a label from p to q and the latter typing the reception of a label by p from q. Rule (end) types the inactive process in a session context containing only terminated sessions. Rule (comp) types parallel composition of processes with disjoint session contexts Δ and Δ'. Finally, rule (close) types a *complete* multiparty session s by hiding the session channel, provided that process P uses $s[1]{:}T_1, \ldots, s[\mathsf{n}]{:}T_n$ and the corresponding role indices and local types form a coherent typing context (Def. A.2).

The following definition is used in § 4, to define the projection of a global type onto a participant.

Definition A.3 (Merge between Local Types)**.** The *merge* $T \sqcup T'$ of T and T' is defined by $T \sqcup T \triangleq T$; and with $T = \&\mathsf{r}\{\mathsf{l}_i : T_i\}_{i\in I}$ and $T' = \&\mathsf{r}\{\mathsf{l}'_j : T'_j\}_{j\in J}$,

$$
T \sqcup T' \triangleq \&\mathsf{r}(\{\mathsf{l}_h : T_h\}_{h\in I\setminus J} \cup \{\mathsf{l}'_h : T'_h\}_{h\in J\setminus I} \cup \{\mathsf{l}_h : T_h \sqcup T'_h\}_{h\in I\cap J})
$$

if $\mathsf{l}_h = \mathsf{l}'_h$ for each $h \in I \cap J$; and homomorphic for other types (i.e. $\mathcal{T}[T_1] \sqcup \mathcal{C}[T_2] = \mathcal{T}[T_1 \sqcup T_2]$ where $\mathcal{T}$ is a context of local types). $T \sqcup T'$ is undefined otherwise. ◇

Reduction Semantics The semantics of MP processes are given in Figure 6, including the reduction rules for delegation (§ 6) and replication (§ 7), but omitting the standard structural congruence rules and compatible closure cases.

A.1.3 Proof of Proposition 2.1

Proof. By the results in [7, 27]. □

A.2 Proofs from § 3 – Relating the CLL and MP systems

A.2.1 Proof of Proposition 3.1

Proof. The prefix case is straightforward by Definition 3.1 and (thread) in Definition 3.3; and the parallel composition uses

(comp) in Definition 3.3. Both cases are mechanical by induction on P. □

A.3 Proofs from § 4 – Partial Multiparty Compatibility

A.3.1 Proof of Proposition 4.1

Proof. By definition of the projection and the LTSs. We use the swapping rules defined in Definition 4.3. □

A.3.2 Proof of Theorem 4.1

Proof. **(DF)⇒(MC):** (a) Suppose configuration C_0 is deadlock-free. Then by DF, for all $C_0 \xrightarrow{\vec{\ell}} C$, there exists $C \xrightarrow{\ell_1 \cdot \overline{\ell_1}} C_1 \cdots C_n \xrightarrow{\ell_n \cdot \overline{\ell_n}} C'$ such that C' contains only end.

The base case $C_n = C_0 = C$ is obvious. Suppose $T_{1\mathsf{r}} \xrightarrow{\ell'_k} T_{k+1\mathsf{r}}$ and we are at C_k. In the case ℓ'_k is an output $\ell'_k = \mathsf{rq}\uparrow(\tau)$ or a selection $\ell'_k = \mathsf{rq} \triangleleft l$, there are traces such that $C_k \xrightarrow{\vec{\ell'}} C'_k$ which does not include the action from/to the participant p. Hence at C'_k, we have $T_{1\mathsf{r}} \xrightarrow{\ell'_k} T_{k+1\mathsf{r}}$ and $C'_k \xrightarrow{\ell'_k \cdot \overline{\ell_k}'} C''_k$. This matches with Definition 4.7(1). The case of the input $\ell'_k = \mathsf{rq}\downarrow(\tau)$ is similar, and matches with Definition 4.7(2). In the case ℓ'_k is a branching such that $\ell'_k = \mathsf{rq} \triangleright l$, we can reach C' which only contains end if and only if there exists $\ell''_k = \mathsf{rq} \triangleright l'$ such that $T_{1\mathsf{r}} \xrightarrow{\ell''_k} T'_{k+1\mathsf{r}}$ and $C_k \xrightarrow{\vec{\ell''}} C'_k \xrightarrow{\overline{\ell_k}'' \cdot \ell''_k} C''_k$. This matches with Definition 4.7(3).
(SC)⇒(WF) By [11].
(WF)⇒(DF) By definition of projection. □

A.3.3 Proof of Lemma 4.1

Proof. Mechanical by induction on G_i, G_j and G_k. □

A.3.4 Proof of Theorem 4.2

Proof. By Lemma 4.1, we need only show the case where $G'_{n-1} = \mathsf{fuse}(\cdots(\mathsf{fuse}(G_1, G_2), G_3), \cdots, G_{n-1})$ and $\mathsf{fuse}(G'_{n-1}, G_n)$ is complete. We proceed by induction on n.

The case for $n = 2$ is obvious.

Suppose G'_{n-1} contains n participants and there are only partial arrows from p_n or to p_n and $\mathsf{fuse}(G'_{n-1}, G_n)$ is complete. Then the partial arrows in G'_{n-1} form (possibly more than one) chains such that $\mathsf{p}_1 \rightsquigarrow \mathsf{p}'_1 \cdots \mathsf{p}_m \rightsquigarrow \mathsf{p}'_m$ where either p_i or p'_i is p_n. To obtain a complete global type, we must have dual chains $\mathsf{p}_1 \rightsquigarrow \mathsf{p}'_1 \cdots \mathsf{p}_m \rightsquigarrow \mathsf{p}'_m$ in G_n. Assuming the completed $n-1$ participants in G'_{n-1} form a well-formed global type, applying fuse rules one by one from the head, we see that $\mathsf{fuse}(G'_{n-1}, G_n)$ is well-formed. □

A.4 Proofs from § 5 – CLL encoded as a single multiparty session

A.4.1 Proof of Proposition 5.1

Proof. Since Δ is empty or contains only $\mathbf{1}$ or $\bot$ we have that P is an n-ary composition of (cut-free) processes. If $n = 1$ then $P = \mathbf{0}$ and its corresponding global type is just $\mathbf{end}$. The interesting case is when $n > 1$. Since the context is either empty or contains only $\mathbf{1}$ or $\bot$, we have that P is of the form $(\boldsymbol{\nu}\tilde{a})(P_1 \mid \cdots \mid P_n)$ where all free names $a_j{:}A_j$ of each of the P_i processes are cut with some other $P_{i'}$ using $a_j{:}A_j^{\bot}$. Thus, by construction of $\Vdash$ we have that for each bound name a of P we have $\mathsf{c}_\rho(a)[\mathsf{p}_\rho(a)]{:}T \in \Gamma$ and $\mathsf{c}_\rho(a)[\mathsf{d}_\rho(a)]{:}T' \in \Gamma$ with $T \upharpoonright \mathsf{d}_\rho(a) = \overline{T' \upharpoonright \mathsf{p}_\rho(a)}$ and thus for each action between two roles in a partial global type in $\mathcal{G}$ in we can always find a matching action in another partial global type in $\mathcal{G}$, therefore we can fuse all partial global types in $\mathcal{G}$ into a single global type. □

A.4.2 Proof of Theorem 5.1

Proof. We proceed by induction on the derivation $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$, showing that each case preserves the specified invariant of at most 2 elements in the intersection of $\mathsf{roles}(G_1) \cap \mathsf{roles}(G_2)$, for any $G_1, G_2 \in \mathcal{G}$. The only interesting case is when the last rule in the derivation is (comp-$\mathcal{G}$):

(comp-$\mathcal{G}$)
$$\frac{P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta, a{:}A \quad Q_L \Vdash^{\sigma'}_{\rho} \Delta', a{:}A^{\bot}; \Gamma; \mathcal{G} \quad (\star) \text{ in } (\mathsf{comp})}{(\boldsymbol{\nu} a)(P_L \mid Q_L) \Vdash^{(\sigma' \cup \sigma) \setminus \{x\}}_{\rho'} \Delta, \Delta'; \Gamma, \mathsf{c}_\sigma[\mathsf{p}_\sigma]{:}[\![P]\!]_\sigma; \mathcal{G} \cup \{\#_\sigma(P)\}}$$

By the i.h. we have that for any $G'_1, G'_2 \in \mathcal{G}_1$, $\mathsf{roles}(G'_1) \cap \mathsf{roles}(G'_2)$ contains at most 2 elements. By construction we know that roles in $\mathcal{G}$ must either appear in σ' (corresponding to role assignments to channels in Δ' and a) or ρ (corresponding to role assignments to bound names).

By inversion we know that $\forall z \in \Delta, y \in \Delta'.\mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y)$, thus there are no common d_σ role assignments between σ and σ' to free names of the two processes beyond those for a. We also know that $\mathsf{p}_\sigma(a) = \mathsf{d}_{\sigma'}(a)$ and $\mathsf{d}_\sigma(a) = \mathsf{p}_{\sigma'}(a)$. By the definition of $\#_\sigma(T_1)$ there are at least two common role names with each endpoint interaction in $\mathcal{G}_1$ coming from σ and σ' (i.e. role assignments to free names), which are p_σ and $\mathsf{p}_{\sigma'}$. Since p_σ is invariant and $\forall z \in \Delta, y \in \Delta'.\mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y)$, we have that free names in Δ cannot share any additional roles. We need now only consider ρ. By construction, we know that $\forall z \in \Delta, y \in \Delta'.\mathsf{d}_\sigma(z) \notin \rho \wedge \mathsf{d}_{\sigma'}(y) \notin \rho$, thus $\mathsf{d}_\sigma(z)$ cannot appear in $\mathcal{G}_1$ due to ρ. The only remaining possibilities are $\mathsf{p}_\sigma(x)$ and $\mathsf{d}_\sigma(x)$ which are already accounted for from the argument above. Thus we preserve the invariant and conclude the proof. □

A.4.3 Proof of Theorem 5.2

Proof. Each endpoint interaction sequence in $\mathcal{G}$ denotes the contribution of a single endpoint role in the global conversation. By Theorem 5.1 we have that any distinct pair of partial global types in $\mathcal{G}$ shares at most 2 role names. This means that for any distinct roles $\mathsf{p}, \mathsf{q}, \mathsf{r}$, if $\mathsf{p} \leftrightarrow \mathsf{q} \in G$ and $\mathsf{p} \leftrightarrow \mathsf{r} \in G$ then neither $\mathsf{q} \rightarrow \mathsf{r}$ nor $\mathsf{r} \rightarrow \mathsf{q}$ or $\mathsf{q} \rightsquigarrow \mathsf{r}$ nor $\mathsf{r} \rightsquigarrow \mathsf{q}$ in G. Hence, in the connection graph of G we know that we cannot have triangles of the form $(\mathsf{p}, \mathsf{q}), (\mathsf{p}, \mathsf{r}), (\mathsf{r}, \mathsf{q})$ as edges.

We can then see that no cycles can be formed through a "diamond" – a sequence of edges of the form $(\mathsf{p}, \mathsf{q}), (\mathsf{p}, \mathsf{r}), (\mathsf{r}, \mathsf{t}), (\mathsf{q}, \mathsf{t})$ – in the graph by the fact that at each composition step, processes can only share one free name ($\Delta \cup \Delta' = \emptyset$) and role assignments ($\forall z \in \Delta, y \in \Delta'.\mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y)$, similarly for ρ). If we could form a "diamond" cycle in the graph, then either we have to able to eventually compose processes sharing more than one name or with different names mapping to the same roles (to close the "diamond"). □

A.4.4 Auxiliary Lemmas for Theorem 5.3

Lemma A.1. *Let* $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$. $\mathsf{co}(\Gamma)$ *implies* $\Delta = \emptyset$ *or* Δ *contains only* $\mathbf{1}$ *or* $\bot$ *and* $\sigma = \emptyset$.

Proof. Assume to the contrary that $\Delta \neq \emptyset$ and doesn't contain only $\mathbf{1}$ and $\bot$, or $\sigma \neq \emptyset$. Then it must be the case that P has some free name $x{:}A \in \Delta$ where $\sigma(x) = s[\mathsf{p}][\mathsf{q}]$, for some $s, \mathsf{p}, \mathsf{q}$ with $A \neq \mathbf{1}$ or $\bot$. By construction it must necessarily be the case that $s[\mathsf{p}]{:}T \in \Gamma$. Since x is free in P, we cannot have $s[\mathsf{q}]{:}T' \in \Gamma$ with $T \upharpoonright \mathsf{q} = \overline{T' \upharpoonright \mathsf{p}}$: single thread σ-renamings are invariant on the p role name, so for $s[\mathsf{q}]$ to occur in Γ it must've arose due to composition on x, which is impossible since x is free, or on some other (now) bound name y that mapped to $s[\mathsf{q}][\mathsf{r}]$, for some r. However, since $\sigma(x) = s[\mathsf{p}][\mathsf{q}]$, by construction we know that $\mathsf{q} \notin \rho$. This is

contradictory with the assumption of coherence and so we conclude the proof. □

Lemma A.2. *Let $P \Vdash^{\emptyset}_{\rho} \Delta; \Gamma; \mathcal{G}$ with $\Delta = \emptyset$ or Δ containing only $\mathbf{1}$ or $\bot$. We have that $\mathsf{co}(\Gamma)$.*

Proof. Assume to the contrary that Γ is not coherent. So (1) there exists $s[\mathsf{p}]{:}T, s[\mathsf{q}]{:}T'$ in Γ such that $T \upharpoonright \mathsf{q} \neq \overline{T' \upharpoonright \mathsf{p}}$ or (2) $s[\mathsf{p}] : T \in \Gamma$ such that $q \in \mathsf{rset}(T)$ and $s[\mathsf{q}] : T' \notin \Gamma$.

For (1) to be the case, either $T \upharpoonright \mathsf{q}$ contains an action unmatched by $T' \upharpoonright \mathsf{p}$ or vice-versa. Assume wlog that $T \upharpoonright \mathsf{q}$ contains an unmatched action. Since both $s[\mathsf{p}]{:}T \in \Gamma$ and $s[\mathsf{q}]{:}T' \in \Gamma$ we know that there exists $(x, s[\mathsf{p}][\mathsf{q}]) \in \rho$ where some subprocess of P uses $x{:}A$ for some A and some other subprocess of P uses $\overline{x}{:}A^{\bot}$. The duality of $x{:}A$ and $\overline{x}{:}A^{\bot}$ contradicts the existence of an unmatched action in $T \upharpoonright \mathsf{q}$. The argument for an unmatched action in $T' \upharpoonright \mathsf{p}$ is identical. For (2) to be the case, since $s[\mathsf{p}]{:}T \in \Gamma$ and $s[\mathsf{q}]{:}T' \notin \Gamma$, we know that there exists $(x, s[\mathsf{p}][\mathsf{r}]) \in \rho$, with $\mathsf{r} \neq \mathsf{q}$, and that $(z, s[\mathsf{q}][\mathsf{s}]) \notin \rho$. Since $q \in \mathsf{roles}(T)$ then it must be the case that P uses a channel y mapped to $s[\mathsf{p}][\mathsf{q}]$ that is free, which contradicts our assumptions. □

A.4.5 Proof of Theorem 5.3

Proof. By Propositions 2.1 and 5.1 and Theorem 4.1, together with Lemmas A.1 and A.2. □

A.4.6 Proof of Corollary 5.1

Proof. By Proposition 3.1 and Theorem 5.3. □

A.5 Proofs from § 6 – Encoding Delegation

For the results in this section we extend coherence in the natural way, requiring an output of a channel of type T to match with the appropriate input of a channel of type T.

A.5.1 Proof of Propositions 6.1 and 6.2

Proof. Mechanical by induction on P. □

A.5.2 Auxiliary Lemmas for Theorem 6.1

Lemma A.3. *Let $P \vDash^{\emptyset,\eta}_{\rho} \Delta; \Gamma; \mathcal{G}$. $\mathsf{co}(\Gamma)$ implies $\Delta = \emptyset$ or Δ containing only $\mathbf{1}$ or $\bot$.*

Proof. Identical to Lemma D.1, noting that the renamings ensure that the two endpoints of an interaction cannot be implemented by the same single-thread process and that bound-names involved in delegation denote linear interactions along different session channels. □

Lemma A.4. *Let $P \vDash^{\emptyset,\eta}_{\rho} \Delta; \Gamma; \mathcal{G}$ with $\Delta = \emptyset$ or Δ containing only $\mathbf{1}$ or $\bot$. We have that $\mathsf{co}(\Gamma)$.*

Proof. Identical to Lemma D.2, noting that in the case of delegation the sent channel has a dual behaviour to the received channel (but generate compatible endpoint types in Γ). □

Proposition A.1. *Let $P \vDash^{\sigma,\eta}_{\rho} \Delta; \Gamma; \mathcal{G}$ and $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$. There exists a single well-formed global type G for each session s, such that $G = \mathsf{fuse}(\mathcal{G})$ where $\mathsf{fuse}(\mathcal{G})$ denotes fusion of all partial global types for session s in $\mathcal{G}$.*

Proof. Since Δ is empty or contains only $\mathbf{1}$ or $\bot$ we have that P is an n-ary composition of (cut-free) processes. If $n = 1$ then $P = \mathbf{0}$ and its corresponding global type is just $\mathbf{end}$. The interesting case is when $n > 1$. Since the context is either empty or contains only $\mathbf{1}$ or $\bot$, we have that P is of the form $(\boldsymbol{\nu}\tilde{a})(P_1 \mid \cdots \mid P_n)$ where all free names $a_j{:}A_j$ of each of the P_i processes are cut with some other $P_{i'}$ using $a_j{:}A_j^{\bot}$. Thus, by construction of $\vDash$ we have that for each bound name a of P we have $\mathsf{c}_{\rho}(a)[\mathsf{p}_{\rho}(a)]{:}T \in \Gamma$ and $\mathsf{c}_{\rho}(a)[\mathsf{d}_{\rho}(a)]{:}T' \in \Gamma$ with $T \upharpoonright \mathsf{d}_{\rho}(a) = \overline{T' \upharpoonright \mathsf{p}_{\rho}(a)}$ and thus for each action between two roles in a partial global type in $\mathcal{G}$ in we can always find a matching action in another partial global type in $\mathcal{G}$, therefore we can fuse all partial global types in $\mathcal{G}$ into a single global type. □

Lemma A.5. *Let $P \vDash^{\sigma,\eta}_{\rho} \Delta; \Gamma; \mathcal{G}$ and $\Delta = \emptyset$ or Δ containing only $\mathbf{1}$ or $\bot$. The interconnection network graph for $\mathsf{fuse}(\mathcal{G})$ for each session is acyclic.*

Proof. The proof is identical to that of Theorem D.1. We observe that delegation does not add to the connection graph since delegation denotes a distinct multiparty session (in fact, a binary session).

We note that forwarding combined with delegation also do not add to the connection graph, since a forwarder just "ping-pongs" between two channels. For instance in $x(y).[y \leftrightarrow x]$, the forwarder is implemented by a process that will perform an action on y and a dual action on x. Thus if x is mapped to $s[\mathsf{p}][\mathsf{q}]$ and y instantiated with $s'[\mathsf{r}][\mathsf{t}]$ we already had an edge between p and q in the connection graph due to x and one between r and t (in a graph for s') due to y.

Similarly, in $\overline{x}\langle y\rangle.([z \leftrightarrow y] \mid P)$, the forwarder simply ping pongs between z and y, mediating between the process implementing a session on z with the process receiving y. □

A.5.3 Proof of Theorem 6.1

Proof. (1) follows from [7, 27]. (2) follows from Prop. A.1. (3) follows from Lemma A.5. □

A.6 Proofs from § 7 – Encoding Replication

For the results in this section we extend coherence in the natural way, requiring a replicated input to match with the appropriate replicated output.

A.6.1 Proof of Proposition 7.1

Proof. Mechanical by induction on P (similar with Propositions 6.1 and 6.2). □

A.6.2 Auxiliary Lemmas and Definitions for Theorem 7.1

Lemma A.6. *Let $P \Vdash^{\sigma,\eta}_{\rho} \Gamma; \Delta; \Theta; \mathcal{G}$. $\mathsf{co}(\Theta)$ implies $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$ and $\sigma = \emptyset$.*

Proof. As Lemma A.3. □

Lemma A.7. *Let $P \Vdash^{\emptyset,\eta}_{\rho} \Gamma; \Delta; \Theta; \mathcal{G}$ with $\Delta = \emptyset$ or Δ containing only $\mathbf{1}$ or $\bot$. We have that $\mathsf{co}(\Theta)$.*

Proof. As Lemma A.4. □

Theorem A.1. *Let $P \Vdash^{\sigma,\eta}_{\rho} \Gamma; \Delta; \Theta; \mathcal{G}$. $\mathsf{co}(\Theta)$ iff $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$ and $\sigma = \emptyset$.*

Proof. From Lemmas A.6 and A.7. □

Extensions to Exponential in MP The system which contains exponential as sessions is studied in [20]. We extend several definitions for the exponential constructs based on [7] as follows. Note that [20] uses $*!$ for ? and $*?$ for ! in our paper.

1. The syntax of T includes $\tilde{\mathsf{p}}!(T)$ and $\mathsf{p}?(T)$.
2. The duality of binary types is extended to $\overline{!(T)} = ?(T)$ and $\overline{?(T)} = !(T)$.

3. A projection of T into the binary types are defined as an input and an output.
4. $G = \tilde{\mathsf{s}} \to \mathsf{r} : *(T).\mathsf{end}$ is projected to
   - $G\upharpoonright \mathsf{p} = \mathsf{r}?(T)$ if $\mathsf{p} \in \tilde{\mathsf{s}}$;
   - $G\upharpoonright \mathsf{p} = \tilde{\mathsf{s}}!(T)$ if $\mathsf{p} = \mathsf{r}$;
   - $G\upharpoonright \mathsf{p} = (G'\upharpoonright \mathsf{p})$ otherwise.

   Note that the projection from the global types fused from the translations is always defined and $*$ does not have a continuation.
5. The labels for LTSs are extended with $\mathsf{p}\tilde{\mathsf{q}}!(T)$ and $\mathsf{pq}?(T)$, and the LTS rules of the local types are defined as:

$$(\mathsf{que}) \quad \mathsf{q}?(T) \xrightarrow{\mathsf{pq}?(T)} \mathsf{end} \quad (\mathsf{bang}) \quad \tilde{\mathsf{q}}!T \xrightarrow{\mathsf{p}\tilde{\mathsf{q}}!(T)} \tilde{\mathsf{q}}!(T)$$

   We extend the duality of labels as $\overline{\mathsf{p}\tilde{\mathsf{q}}!(T)} = \mathsf{q}_i\mathsf{p}?(T)$ and $\overline{\mathsf{q}_i\mathsf{p}?(T)} = \mathsf{p}\tilde{\mathsf{q}}!(T)$ with $\mathsf{q}_i \in \tilde{\mathsf{q}}$. Then the transitions of the configurations do not change.
6. We extend Definition 4.7 by adding the following cases.
   4. if $\ell = \mathsf{pq}?(T)$ there exists $C \xrightarrow{\vec{\ell'}} C' \xrightarrow{\ell} \xrightarrow{\bar{\ell}} C''$;
   5. if $\ell = \mathsf{p}\tilde{\mathsf{q}}!(T)$ for all $C \xrightarrow{\vec{\ell'}} C''$ such that $C'' = (T'_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$, $T'_{\mathsf{p}} \xrightarrow{\ell} T''_{\mathsf{p}}$.

Theorem 4.1 is updated replacing (DF) by the following liveness property:

Definition A.4 (Live). $C = (T_{0\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$ is *live* if for all $C \xrightarrow{\vec{\ell}} C_1 = (T_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$, if $T_{\mathsf{p}} \xrightarrow{\ell} T'_{\mathsf{p}}$ and ℓ is not !, there exists $C' = (T''_{\mathsf{p}})_{\mathsf{p}\in\mathcal{P}}$ such that $C_1 \xrightarrow{\vec{\ell'}} C'$ and (1) $C' \xrightarrow{\ell\cdot\bar{\ell}} C''$ if ℓ is an output or a selection or ?-output; (2) $C' \xrightarrow{\bar{\ell}\cdot\ell} C''$ if ℓ is an input; or (3) $C' \xrightarrow{\vec{\ell'}\cdot\ell'} C''$ if $\ell = \mathsf{pq} \triangleright l$ with some $\ell' = \mathsf{pq} \triangleright l'$. $\diamond$

Then the rest is proved as Theorem 4.1.

Proposition A.2. *Let $P \Vdash^{\sigma,\eta}_{\rho} \Gamma;\Delta;\Theta;\mathcal{G}$ and $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$. There exists a single well-formed global type G for each session s, such that $G = \mathsf{fuse}(\mathcal{G})$ where $\mathsf{fuse}(\mathcal{G})$ denotes fusion of all partial global types for session s in $\mathcal{G}$.*

Proof. Since Δ is empty or contains only $\mathbf{1}$ or $\bot$ we have that P is an n-ary composition of (cut-free) processes. If $n = 1$ then $P = \mathbf{0}$ and its corresponding global type is just $\mathbf{end}$. The interesting case is when $n > 1$. Since the context is either empty or contains only $\mathbf{1}$ or $\bot$, we have that P is of the form $(\boldsymbol{\nu}\tilde{a})(P_1 \mid \cdots \mid P_n)$ where all free names $a_j{:}A_j$ of each of the P_i processes are cut with some other $P_{i'}$ using $a_j{:}A_j^{\perp}$. Thus, by construction of $\Vdash$ we have that for each bound name a of P we have $\mathsf{c}_\rho(a)[\mathsf{p}_\rho(a)]{:}T \in \Gamma$ and $\mathsf{c}_\rho(a)[\mathsf{d}_\rho(a)]{:}T' \in \Gamma$ with $T \upharpoonright \mathsf{d}_\rho(a) = \overline{T' \upharpoonright \mathsf{p}_\rho(a)}$ and thus for each action between two roles in a partial global type in $\mathcal{G}$ we can always find a matching action in another partial global type in $\mathcal{G}$, therefore we can fuse all partial global types in $\mathcal{G}$ into a single global type. □

Theorem A.2. *Let $P \Vdash^{\sigma,\eta}_{\rho} \Gamma;\Delta;\Theta;\mathcal{G}$. If $\Delta = \emptyset$ or containing only $\mathbf{1}$ or $\bot$, the interconnection net. graph for $\mathsf{fuse}(\mathcal{G})$ for each session is acyclic.*

Proof. As before, noting that it is not possible for different threads to use the same replicated channel.

We note that even if we had a MIX-like rule which did enable for such a process to be typable, each instance of the replicated server uses a distinct session channel and thus denotes distinct global types and connection graphs. □

A.6.3 Proof of Theorem 7.1

Proof. (1) follows from [7, 27]. (2) follows from Prop. A.2. (3) follows from Theorem A.2. □

A.7 Proofs from § 8 – Multicut

A.7.1 Proof of Theorem 8.1

Proof. (1) By the same proposition as Proposition 3.1, we obtain if $P \Vdash^{\sigma}_{\rho} \Delta;\Gamma$, then $\rho(\sigma(P)) \vdash_{\mathsf{MP}} \Gamma$. By the definition, Γ only contains a single multiparty session where each prefix is simple [14, Definition 5.25 in JACM]. Since $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\bot$, Γ is coherent. By the result of [14, § 5], $\rho(\sigma(P)) \to^* \mathbf{0}$. Then by the operational correspondence between MP and CLL, $P \to^* \mathbf{0}$. (2) is by construction of (Mcomp). □

A.7.2 Proof of Theorem 8.2

Proof. (1) By the same reasoning as above, noting that as the image of CLL, ! and ? names (shared names below) satisfy the following well-linked condition from [14][Definition 5.27 in JACM] when ?-name is active:

> We say P is well-linked when for each $P \to^* Q$, whenever Q has an active prefix whose subject is a (free or bound) shared name, then it is always part of a redex.

Hence the translation into MP satisfies the liveness property. Since if $P \Vdash^{(\sigma,\eta)}_{\rho} \Delta;\Gamma$ then $\sigma(\eta(\rho(P))) \vdash_{\mathsf{MP}} \Gamma$ and the operational correspondence between MP and CLL, CLL satisfies the liveness. (2) is by construction. □

A.8 Definition of $\biguplus$

$$
\begin{array}{lcl}
s[\mathsf{p}]{:}\mathsf{q}\uparrow(\tau) \biguplus (\Gamma', s[\mathsf{p}]{:}T') & \triangleq & \Gamma', s[\mathsf{p}]{:}\mathsf{q}\uparrow(\tau); T' \\
s[\mathsf{p}]{:}\mathsf{q}\uparrow(\tau) \biguplus \emptyset & \triangleq & s[\mathsf{p}]{:}\mathsf{q}\uparrow(\tau); \mathsf{end} \\
s[\mathsf{p}]{:}\mathsf{q}\downarrow(\tau) \biguplus (\Gamma', s[\mathsf{p}]{:}T') & \triangleq & \Gamma', s[\mathsf{p}]{:}\mathsf{q}\downarrow(\tau); T' \\
s[\mathsf{p}]{:}\mathsf{q}\downarrow(\tau) \biguplus \emptyset & \triangleq & s[\mathsf{p}]{:}\mathsf{q}\downarrow(\tau); \mathsf{end} \\
s[\mathsf{p}]{:} \oplus \mathsf{q}\{l_j : T_j\} \biguplus (\Gamma', s[\mathsf{p}]{:}T') & \triangleq & \Gamma', s[\mathsf{p}]{:} \oplus \mathsf{q}\{l_j : T_j\} \\
s[\mathsf{p}]{:} \oplus \mathsf{q}\{l_j : T_j\} \biguplus \emptyset & \triangleq & s[\mathsf{p}]{:} \oplus \mathsf{q}\{l_j : T_j\} \\
s[\mathsf{p}]{:} \& \mathsf{q}\{l_j : T_j\}_{j\in J} \biguplus (\Gamma', s[\mathsf{p}]{:}T') & \triangleq & \Gamma', s[\mathsf{p}]{:} \& \mathsf{q}\{l_j : T_j\}_{j\in J} \\
s[\mathsf{p}]{:} \& \mathsf{q}\{l_j : T_j\}_{j\in J} \biguplus \emptyset & \triangleq & s[\mathsf{p}]{:} \& \mathsf{q}\{l_j : T_j\}_{j\in J}
\end{array}
$$

A.9 Partial Global Type Generation for Delegation

Figure 7 lists the partial global type generation for CLL for delegation.

B. Extended $\vDash$ judgement

$$\frac{P_L \vdash^{\sigma,\eta}_{\mathsf{CL}} \Delta \quad \rho = \emptyset}{P_L \vDash^{\sigma,\eta}_{\rho} \Delta; [\![P_L]\!]^{\eta}_{\sigma}; \#_{(\sigma,\eta)}(P)} \ (\mathsf{thread})$$

$$\frac{\begin{array}{c} P_L \vdash^{\sigma;\eta}_{\mathsf{CL}} \Delta, x{:}A \qquad Q_L \vDash^{\sigma',\eta'}_{\rho} \Delta', x{:}A^{\perp}; \Gamma; \mathcal{G} \quad (\dagger) \\ \forall z \in \Delta; y \in \Delta'.\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \\ \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \mathsf{d}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y) \\ \forall z \in \Delta, x; y \in \Delta', x.\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y) \end{array}}{(\boldsymbol{\nu}x)(P_L \mid Q_L) \vDash^{(\sigma'\cup\sigma)\setminus\{x\},\eta\cup\eta'}_{\rho'} \Delta, \Delta'; \Gamma, [\![P_L]\!]^{\eta}_{\sigma}; \mathcal{G} \cup \#^{\sigma}_{\eta}(P)} \ (\mathsf{comp}_d)$$

$$
\begin{array}{rcl}
\#^{\sigma}_{\eta}(\mathbf{0})(s) & \triangleq & \mathsf{end} \\
\#^{\sigma}_{\eta}(x\langle y\rangle.(P_1 \mid P_2))(s) & \triangleq & \begin{cases} \mathsf{p}_\sigma(x) \rightsquigarrow \mathsf{d}_\sigma(x) :\uparrow (\llbracket P_1 \rrbracket^{\eta}_{\sigma}(y)).\mathsf{fuse}(\#^{\sigma}_{\eta}(P_1)(s), \#^{\sigma}_{\eta}(P_2)(s)) \\ \mathsf{fuse}(\#^{\sigma}_{\eta}(P_1)(s), \#^{\sigma}_{\eta}(P_2)(s)) \end{cases} \\
\#^{\sigma}_{\eta}(x(y).P_1)(s) & \triangleq & \begin{cases} \mathsf{d}_\sigma(x) \rightsquigarrow \mathsf{p}_\sigma(x) :\downarrow (\overline{\llbracket P_1 \rrbracket^{\eta}_{\sigma}(y)}).\#^{\sigma}_{\eta}(P_1)(s) \\ \#^{\sigma}_{\eta}(P_1)(s) \end{cases} \\
\#^{\sigma}_{\eta}(x.l_i; P_1)(s) & \triangleq & \begin{cases} \mathsf{p}_\sigma(x) \rightsquigarrow \mathsf{d}_\sigma(x); \oplus\{\#^{\sigma}_{\eta}(P_1)(s)\} \\ \#^{\sigma}_{\eta}(P_1)(s) \end{cases} \\
\#^{\sigma}_{\eta}(x.\mathsf{case}(l_i{:}P_i)_{i\in I})(s) & \triangleq & \begin{cases} \mathsf{d}_\sigma(x) \rightsquigarrow \mathsf{p}_\sigma(x); \&\{\#^{\sigma}_{\eta}(P_i)(s)\}_{i\in I} \\ \#^{\sigma}_{\eta}(P_1)(s) \end{cases}
\end{array}
$$

Figure 7: Partial Global Types for Delegation

C. Multicut for Replication

Using the name to role-indexed channel mapping of § 7 we redefine the judgement $P \Vdash^{\sigma,\eta}_{\rho} \Gamma; \Delta; \Theta$ as follows:

$$
\text{(Mcomp)}\quad \frac{\begin{array}{c} P_L \Vdash^{\sigma,\eta}_{\rho} \Gamma; \Delta, x_1{:}A_1, ..., x_n{:}A_n; \Theta_1 \\ Q_L \Vdash^{\sigma',\eta'}_{\rho'} \Gamma; \Delta', x_1{:}A_1^{\perp}, ..., x_n{:}A_n^{\perp}; \Theta_2 \\ \mathsf{fuse}(\Theta_1, \Theta_2) \text{ defined} \quad \rho' \cap \rho = \emptyset \quad (\diamond) \end{array}}{(\boldsymbol{\nu} x_1, ..., x_n)(P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{x_1,...,x_n\},\eta\cup\eta'}_{\rho''} \Gamma; \Delta, \Delta'; \Theta_1, \Theta_2}
$$

where $(\diamond)$ – key changes from $(\dagger)$ are highlighted in red:

$$
\diamond \begin{cases} \rho'' = \rho \cup \rho' \cup_i (x_i, \mathsf{c}_\sigma(x_i)[\mathsf{p}_\sigma(x_i)][\mathsf{d}_\sigma(x_i)]) \\ \mathsf{c}_\sigma(x_i) = \mathsf{c}_{\sigma'}(x_i) \quad \mathsf{p}_\sigma(x_i) = \mathsf{d}_{\sigma'}(x_i) \quad \mathsf{d}_\sigma(x) = \mathsf{p}_{\sigma'}(x) \\ \forall z \in \Delta, y \in \Delta'.\mathsf{c}_\sigma(x) = \mathsf{c}_{\sigma'}(z) \Rightarrow \mathsf{d}_\sigma(z), \mathsf{d}_{\sigma'}(y) \notin \rho \\ \wedge \mathsf{p}_\sigma(y) \neq \mathsf{p}_\sigma(z) \wedge \mathsf{p}_\sigma(y) \neq \mathsf{d}_{\sigma'}(z) \wedge \mathsf{p}_{\sigma'}(z) \neq \mathsf{d}_\sigma(y) \\ \forall x \in \eta.\forall y \in \eta'.x = y \Rightarrow \eta(x) \neq \eta'(y) \\ \sigma(\Gamma) = \sigma'(\Gamma) \end{cases}
$$

D. CLL threads as multirole MP threads

The results developed in our may seem to hinge on the fact that we map CLL cut-free processes (threads) to single role processes in MP. In this section we prove this is not the case by showing that if we consider a generalized *bijective* mapping from CLL channels to role-annottated MP channels (i.e.where actions in a cut-free process may map to actions pertaining to *different* MP roles), we may reconstruct identical results.

Definition D.1 (Bijective Channel to Role-Indexed Channel Mapping)**.** Let $P_L \vdash_{\mathsf{CL}} \Delta$ not using the cut rule. We define a bijective channel name to role-indexed channel name mapping σ such that for all $a \in fn(P_L)$, $\sigma(a) = s[\mathsf{p}][\mathsf{q}]$, for some $s, \mathsf{p}, \mathsf{q}$, where $b \in fn(P_L)$ with $b \neq a$ implies $\sigma(b) = s[\mathsf{p}'][\mathsf{q}']$ such that $(\mathsf{p} \neq \mathsf{q}' \wedge \mathsf{q} \neq \mathsf{p}'), (\mathsf{p} \neq \mathsf{p}' \vee \mathsf{q} \neq \mathsf{q}')$. We write $P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta$ for such a mapping and $\mathsf{c}_\sigma(a)$, $\mathsf{p}_\sigma(a)$ and $\mathsf{d}_\sigma(a)$ to denote the channel, first and second roles in the image of a in σ. $\diamond$

The role restrictions in the definition guarantee that the renaming is bijective and that in a single thread we cannot implement dual role endpoints. Since threads now denote potentially multiple roles, we must generate a local typing *context* from each thread, assigning local types to each role annotated channel. In the cases of selection and branching we collect actions pertaining to the principal role assignment for each branch and combine the result with the context generation for the continuation processes (since it may contain different role assignments).

Definition D.2 (Local Typing)**.** We generate a local typing context Γ such that $\sigma(P_L) \vdash_{\mathsf{MP}} \Gamma$ by induction on the structure of P_L, written $\llbracket P \rrbracket_\sigma$ (we write $\llbracket P \rrbracket_\sigma(c)$ for the type binding for c in $\llbracket P \rrbracket_\sigma$ or end if no such binding exists and c for $\mathsf{c}_\sigma(a)[\mathsf{p}_\sigma(x)]$):

$$
\begin{array}{lcl}
\llbracket \mathbf{0} \rrbracket_\sigma & \triangleq & \emptyset \\
\llbracket a\langle M\rangle.P \rrbracket_\sigma & \triangleq & c{:}\mathsf{d}_\sigma(a)\uparrow(\tau) \uplus \llbracket P \rrbracket_\sigma \quad \text{with } M : \tau \\
\llbracket a(y).P \rrbracket_\sigma & \triangleq & c{:}\mathsf{d}_\sigma(a)\downarrow(\tau) \uplus \llbracket P \rrbracket_\sigma \quad \text{with } y : \tau \\
\llbracket a.l_j; P \rrbracket_\sigma & \triangleq & c{:} \oplus \mathsf{d}_\sigma(a)\{l_j{:}\llbracket P \rrbracket_\sigma(c)\} \uplus \llbracket P \rrbracket_\sigma \\
\llbracket a.\mathsf{case}\{l_i{:}P_i\}_{i\in I} \rrbracket_\sigma & \triangleq & c{:} \& \mathsf{d}_\sigma(a)\{l_i{:}\llbracket P_i \rrbracket_\sigma(c)\}_{i\in I} \uplus \llbracket P_i \rrbracket_\sigma \\
\llbracket [a \leftrightarrow b] \rrbracket_\sigma & \triangleq & \llbracket \mathsf{id}_A(a, b) \rrbracket_\sigma \text{ with } \Delta = a{:}A, b{:}A^{\perp}
\end{array}
$$

We define the judgment $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma$ as before, but where σ is a mapping according to Definition D.1 and $\blacklozenge$ is defined as:

$$
\blacklozenge \begin{cases} \rho' = \rho \cup (x, \mathsf{c}_\sigma(x)[\mathsf{p}_\sigma(x)][\mathsf{d}_\sigma(x)]) \\ \mathsf{c}_\sigma(x) = \mathsf{c}_{\sigma'}(x) \quad \mathsf{p}_\sigma(x) = \mathsf{d}_{\sigma'}(x) \quad \mathsf{d}_\sigma(x) = \mathsf{p}_{\sigma'}(x) \\ \forall z \in \Delta, y \in \Delta'.\mathsf{c}_\sigma(x) = \mathsf{c}_{\sigma'}(z) \Rightarrow \\ \qquad \mathsf{d}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \mathsf{d}_\sigma(z), \mathsf{d}_{\sigma'}(y) \notin \rho \end{cases}
$$

$$
\frac{P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta \quad \rho = \emptyset}{P_L \Vdash^{\sigma}_{\rho} \Delta; \llbracket P_L \rrbracket_\sigma} \ (\mathsf{thread}_b)
$$

$$
\text{(comp}_b\text{)}\quad \frac{\begin{array}{c} P_L \vdash^{\sigma,\eta}_{\mathsf{CL}} \Delta, x{:}A \qquad Q_L \Vdash^{\sigma'}_{\rho} \Delta', x{:}A^{\perp}; \Gamma \quad (\blacklozenge) \\ \forall z \in \Delta; y \in \Delta'.\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \\ \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \qquad \mathsf{d}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y) \\ \forall z \in \Delta, x; y \in \Delta', x.\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y) \end{array}}{(\boldsymbol{\nu} x)(P_L \mid Q_L) \Vdash^{(\sigma'\cup\sigma)\setminus\{x\}}_{\rho'} \Delta, \Delta'; \Gamma, \llbracket P_L \rrbracket^{\eta}_{\sigma}}
$$

Proposition D.1. *If* $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma$ *then* $\rho(\sigma(P)) \vdash_{\mathsf{MP}} \Gamma$

D.1 CLL bijective mapping as a single MP session

We replay the development of § 5 for our bijective mapping.

Definition D.3 (Generating Partial Global Types)**.** *Given* P *such that* $P \vdash^{\sigma}_{\mathsf{CL}} \Delta$ *we generate its partial global type, written* $\#_\sigma(P)$ *by induction on the structure of* P *as follows:*

$$
\begin{array}{lcll}
\#_\sigma(\mathbf{0}) & \triangleq & \mathsf{end} & \\
\#_\sigma(x\langle M\rangle.P) & \triangleq & \mathsf{p}_\sigma(x) \rightsquigarrow \mathsf{d}_\sigma(x) :\uparrow (\tau); \#_\sigma(P) & \text{with } M : \tau \\
\#_\sigma(x(y).P) & \triangleq & \mathsf{d}_\sigma(x) \rightsquigarrow \mathsf{p}_\sigma(x) :\downarrow (\tau); \#_\sigma(P) & \text{with } y : \tau \\
\#_\sigma(x.l_i; P) & \triangleq & \mathsf{p}_\sigma(x) \rightsquigarrow \mathsf{d}_\sigma(x); \oplus\{l_i{:}\#_\sigma(P)\} & \\
\#_\sigma(x.\mathsf{case}(l_i{:}P_i)_{i\in I}) & \triangleq & \mathsf{d}_\sigma(x) \rightsquigarrow \mathsf{p}_\sigma(x); \&\{l_i{:}\#_\sigma(P_i)\}_{i\in I} &
\end{array}
$$

Exactly as before, we generalize the $\Vdash$ judgment with a set of partial global types, written $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$. We may then replicate

the results of § 5 uniformly.

$$(\mathsf{thread}_b\text{-}\mathcal{G})\quad \frac{P_L \vdash^{\sigma}_{\mathsf{CL}} \Delta \quad \rho = \emptyset}{P_L \Vdash^{\sigma}_{\rho} \Delta; [\![P_L]\!]_\sigma; \{\#_\sigma(P_L)\}}$$

$$(\mathsf{comp}_b\text{-}\mathcal{G})\quad \frac{\begin{array}{c} P_L \vdash^{\sigma,\eta}_{\mathsf{CL}} \Delta, x{:}A \qquad Q_L \Vdash^{\sigma'}_{\rho} \Delta', x{:}A^{\perp}; \Gamma; \mathcal{G} \quad (\blacklozenge) \\ \forall z \in \Delta; y \in \Delta'.\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \\ \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{d}_{\sigma'}(y) \wedge \mathsf{d}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y) \\ \forall z \in \Delta, x; y \in \Delta', x.\mathsf{c}_\sigma(z) = \mathsf{c}_{\sigma'}(y) \Rightarrow \mathsf{p}_\sigma(z) \neq \mathsf{p}_{\sigma'}(y) \end{array}}{(\boldsymbol{\nu} x)(P_L \mid Q_L) \Vdash^{(\sigma' \cup \sigma)\setminus\{x\}}_{\rho'} \Delta, \Delta'; \Gamma, [\![P_L]\!]^{\eta}_{\sigma}; \mathcal{G} \cup \{\#_\sigma(P_L)\}}$$

Proposition D.2. *Let $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$ and $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\perp$. There exists a single well-formed global type G such that $G = \mathsf{fuse}(\mathcal{G})$ where $\mathsf{fuse}(\mathcal{G})$ denotes fusion of all partial global types in $\mathcal{G}$.*

Proof. Since Δ is empty or contains only $\mathbf{1}$ or $\perp$ we have that P is an n-ary composition of (cut-free) processes. If $n = 1$ then $P = \mathbf{0}$ and its corresponding global type is just end. The interesting case is when $n > 1$. Since the context is either empty or contains only $\mathbf{1}$ or $\perp$, we have that P is of the form $(\boldsymbol{\nu}\tilde{a})(P_1 \mid \cdots \mid P_n)$ where all free names $a_j{:}A_j$ of each of the P_i processes are cut with some other $P_{i'}$ using $a_j{:}A_j^{\perp}$. Thus, by construction of $\Vdash$ we have that for each bound name a of P we have $\mathsf{c}_\rho(a)[\mathsf{p}_\rho(a)]{:}T \in \Gamma$ and $\mathsf{c}_\rho(a)[\mathsf{d}_\rho(a)]{:}T' \in \Gamma$ with $T \upharpoonright \mathsf{d}_\rho(a) = \overline{T' \upharpoonright \mathsf{p}_\rho(a)}$ and thus for each action between two roles in a partial global type in $\mathcal{G}$ in we can always find a matching action in another partial global type in $\mathcal{G}$, therefore we can fuse all partial global types in $\mathcal{G}$ into a single global type. □

Theorem D.1. *Let $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$. Let G be the fusion of all partial global types in $\mathcal{G}$ into a single global type. If $\Delta = \emptyset$ or containing only $\mathbf{1}$ or $\perp$, The interconnection network graph for G is acyclic.*

Proof. Assume to the contrary that the connection graph for G has a cycle. We have two kinds of cycles: a sequence of edges of the form $(\mathsf{p},\mathsf{q}),(\mathsf{p},\mathsf{r}),(\mathsf{q},\mathsf{r})$ – a triangle – or a sequence of edges of the form $(\mathsf{p},\mathsf{q}),(\mathsf{p},\mathsf{r}),(\mathsf{q},\mathsf{t}_1),\ldots,(\mathsf{t}_n,\mathsf{s}),(\mathsf{r},\mathsf{v}_1),\ldots(\mathsf{v}_m,\mathsf{s})$ – a diamond. We show that Δ cannot be empty or contain only $\mathbf{1}$ or $\perp$ in either case, deriving a contradiction.

Assume the connection graph for G contains a triangle. Since (p,q) is in the connection graph, we know that roles p and q cannot be implemented in the same process thread. Similarly for p and r and q and r. Thus, we must have at least three process threads P_1, P_2, P_3, one for each role. Without loss of generality, assume $P_1 \vdash^{\sigma_0}_{\mathsf{CL}} x{:}A, y{:}B$ with $\sigma_0(x) = s[\mathsf{p}][\mathsf{q}]$ and $\sigma(y) = s[\mathsf{p}][\mathsf{r}]$. $P_2 \vdash^{\sigma_1}_{\mathsf{CL}} x : A^{\perp}, z{:}C$ with $\sigma_1(x) = s[\mathsf{q}][\mathsf{p}]$ and $\sigma_1(z) = s[\mathsf{q}][\mathsf{r}]$. It is then immediate that we cannot find any P_3 implementing r (to fully empty the context) since it would have to share two channel names with the composition of P_1 and P_2, which is not a well-formed composition.

Assume the connection graph for G contains a diamond. We already know by Theorem 5.2 that when all roles are implemented by separate process threads that we cannot form a diamond. We also know that only unconnected roles in the graph may be implemented by the same process thread by Definition D.1. Thus the remaining possibility is for unconnected roles in the connection graph to be implemented by the same process thread. We note that if q and r are implemented by the same process thread, then we cannot find a closing instance of p (since we would need to compose two processes sharing two channel names).

We proceed by case analysis on (n, m): when $n = 0$ and $m = 0$ we see that we cannot find a closed instance of the network since either we are in the one-role-per-thread case (Theorem 5.2) or p and s are implemented by the same thread. If this were the case, we must have some $P_1 \vdash^{\sigma_0}_{\mathsf{CL}} x{:}A, y{:}B, z{:}C, w{:}D$ such that $\sigma_0(x) = s[\mathsf{p}][\mathsf{q}]$, $\sigma_0(y) = s[\mathsf{p}][\mathsf{r}]$, $\sigma_0(z) = s[\mathsf{s}][\mathsf{r}]$, $\sigma_0(w) = s[\mathsf{s}][\mathsf{q}]$. This is impossible by Definition D.1.

When $n = 0$ and $m = m' + 1$: we have established that p and q, p and r and q and r cannot be implemented by the same thread. If q and v_1 were implemented by the same thread, we can easily see that we cannot find a closed instance of this network since we'd need to compose with the (distinct) implementations of p and r which are themselves connected and thus we'd need to compose a process mapped to $s[\mathsf{q}][\mathsf{p}]; s[\mathsf{v}_1][\mathsf{r}]; s[\mathsf{v}_1][\mathsf{v}_2]$ with a process mapped to $s[\mathsf{p}][\mathsf{q}]; s[\mathsf{p}][\mathsf{r}]$ and another mapped to $s[\mathsf{r}][\mathsf{p}]; s[\mathsf{r}][\mathsf{v}_1]$. If we compose the first with the second, we cannot compose with the third since they would share two channel/role assignment pairs: $s[\mathsf{r}][\mathsf{p}]$ and $s[\mathsf{r}][\mathsf{v}_1]$. A similar reasoning applies to the other ways of composing the processes. It is also easy to see that the same reasoning applies if q and v_i were implemented by the same thread.

When $n = n' + 1$ and $m = m' + 1$ we begin with the case where q and v_1 are implemented by the same thread and r and t_1 are implemented by the same thread. It is easy to see we cannot compose such processes. The same reasoning applies as we increase n and m. If t_i and v_i are implemented by the same process we must have processes sharing two distinct channels, which we cannot. If t_n and v_m are the same process, then the implementation of s must share two channels with this process, which is also a contradiction. Finally, if p and s are the same process, we are in the same situation as that described in the $n = 0$ and $m = 0$ case. □

Lemma D.1. *Let $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$. $\mathsf{co}(\Gamma)$ implies $\Delta = \emptyset$ or Δ contains only $\mathbf{1}$ or $\perp$ and $\sigma = \emptyset$.*

Proof. Assume to the contrary that $\Delta \neq \emptyset$ and doesn't contain only $\mathbf{1}$ and $\perp$, or $\sigma \neq \emptyset$. Then it must be the case that P has some free name $x{:}A \in \Delta$ where $\sigma(x) = s[\mathsf{p}][\mathsf{q}]$, for some $s, \mathsf{p}, \mathsf{q}$ with $A \neq \mathbf{1}$ or $\perp$. By construction it must necessarily be the case that $s[\mathsf{p}]{:}T_1 \in \Gamma$. However, since x is free in P we have that if $s[\mathsf{q}]{:}T_2 \in \Gamma$ then $T_1 \upharpoonright \mathsf{q} \neq T_2 \upharpoonright \mathsf{p}$. For T_2 to have the corresponding actions with role p there must exist a free name y in P such that $\sigma(y) = s[\mathsf{q}][\mathsf{p}]$. If both x and y are in the same cut-free sub-process this contradicts Definition D.1. If they are in different sub-processes, this contradicts the premise of the composition rule. The only remaining possibility is for a bound name of P to have been mapped to $s[\mathsf{q}][\mathsf{p}]$, which also contradicts the premise of the composition rule. This arguments contradicts the assumption of coherence and so we conclude the proof. □

Lemma D.2. *Let $P \Vdash^{\emptyset}_{\rho} \Delta; \Gamma; \mathcal{G}$ with $\Delta = \emptyset$ or Δ containing only $\mathbf{1}$ or $\perp$. We have that $\mathsf{co}(\Gamma)$.*

Proof. Assume to the contrary that Γ is not coherent. Then this means there exists $s[\mathsf{p}]{:}T, s[\mathsf{q}]{:}T'$ in Γ such that $T \upharpoonright \mathsf{q} \neq \overline{T' \upharpoonright \mathsf{p}}$. For this to be the case, either $T \upharpoonright \mathsf{q}$ contains an action unmatched by $T' \upharpoonright \mathsf{p}$ or vice-versa. Assume that $T \upharpoonright \mathsf{q}$ contains an unmatched action. Since $s[\mathsf{p}]{:}T \in \Gamma$ and $s[\mathsf{q}]{:}T' \in \Gamma$ we know that there exists $(x, s[\mathsf{p}][\mathsf{q}]) \in \rho$ – this is necessarily the case due to the construction of the $\Vdash$ judgment, insofar as if both $s[\mathsf{q}]{:}T'$ and $s[\mathsf{p}]{:}T \in \Gamma$ with $\mathsf{q} \in \mathsf{rset}(T)$ then we cannot introduce $s[\mathsf{q}]$ in the context unless with an instance of the composition rule for the corresponding channel x – where some subprocess of P uses $x{:}A$ for some A and some other subprocess of P uses $x{:}A^{\perp}$. The duality of $x{:}A$ and $\overline{x}{:}A^{\perp}$ contradicts the existence of an unmatched action in $T \upharpoonright \mathsf{q}$. The argument for an unmatched action in $T' \upharpoonright \mathsf{p}$ is identical.

The other possibility is for $s[\mathsf{p}] : T \in \Gamma$ such that $q \in \mathsf{roles}(T)$ and $s[\mathsf{q}] : T' \notin \Gamma$. If this is the case, then we must have x such that $\sigma(x) = s[\mathsf{p}][\mathsf{q}]$, with $x{:}A'$ in Δ and $A' \neq \mathbf{1}, \perp$ which is a contradiction. □

Theorem D.2. *Let* $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}.\ \mathsf{co}(\Gamma)$ *iff* $\Delta = \emptyset$ *or* Δ *contains only* $\mathbf{1}$ *or* $\bot$ *and* $\sigma = \emptyset$.

Proof. By Lemmas D.1 and D.2 □

Theorem D.3. *Let* $P \Vdash^{\sigma}_{\rho} \Delta; \Gamma; \mathcal{G}$ *and* $\Delta = \emptyset$ *or* Δ *containing only* $\mathbf{1}$ *or* $\bot$. *Then we have:* (1) $P \to^* \mathbf{0}$*;* (2) $\mathsf{fuse}(\mathcal{G})$ *at each session is well-formed and deadlock-free; and* (3) *the interconnection network graph for* $\mathsf{fuse}(\mathcal{G})$ *is acyclic.*

Proof. (1) follows from [7, 27]. (2) follows from Prop. D.2. (3) follows from Theorem D.1. □