Multiparty Session Types as Coherence Proofs

Marco Carbone[1], Fabrizio Montesi[2], Carsten Schürmann[1], and Nobuko Yoshida[3]

1 **IT University of Copenhagen, Denmark**
2 **University of Southern Denmark, Denmark**
3 **Imperial College London, UK**

Abstract

We propose a Curry-Howard correspondence between a language for programming multiparty sessions and a generalisation of Classical Linear Logic (CLL). In this framework, propositions correspond to the local behaviour of a participant in a multiparty session type, proofs to processes, and proof normalisation to executing communications. Our key contribution is generalising duality, from CLL, to a new notion of n-ary compatibility, called *coherence*. Building on coherence as a principle of compositionality, we generalise the cut rule of CLL to a new rule for composing many processes communicating in a multiparty session. We prove the soundness of our model by showing the admissibility of our new rule, which entails deadlock-freedom via our correspondence.

1998 ACM Subject Classification F1.1 Models of Computation

Keywords and phrases Programming languages, Type systems, Session Types, Linear Logic

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2015.412

1 Introduction

Session types are protocols for communications in concurrent systems [13, 22]. A recent line of work investigates Curry-Howard correspondences between the type theory of session types and linear logic, where proofs correspond to processes, propositions to types, and proof normalisation to communications [4, 23]. An important consequence of such correspondences is that several notions that usually require complex additional definitions and proofs, e.g., dependency relations for deadlock-freedom [9, 19], follow for free from the theory of linear logic, yielding a succinct formulation of the formal foundations of sessions.

The aforementioned correspondences cover only session types with exactly two participants, called *binary session types*. In practice, however, protocols often describe the behaviour of multiple participants [21]. *Multiparty Session Types (MPSTs)* have been proposed to capture such protocols, by matching the communications enacted by many participants with a global scenario [14]. Unfortunately, MPSTs are more involved than binary session types, since they include complex analyses on the structure of protocols and a mapping from *global types*, which describe multiparty protocols, to *local types*, which describe the local behaviour of each single participant. So far, it has been unclear whether a succinct logical formulation of MPSTs can be developed, as done for binary session types. Therefore, we ask:

Can we design a proof theory for reasoning about multiparty sessions?

A positive answer to our question would lead to a clearer understanding of the principles that underpin multiparty session programming. The main challenge lies in the foundational notion of duality found in linear logic, which, in a Curry-Howard interpretation of propositions as types, checks whether the session types of two respective participants are compatible. It is an

26th International Conference on Concurrency Theory (CONCUR 2015).
Editors: Luca Aceto and David de Frutos Escrig; pp. 412–426

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

open question how to generalise the notion of type duality to that of "multiparty compatibility" found in MPSTs, which allows to compose an arbitrary number of participants [14, 11, 16]. Therefore, differently from previous work, we are in a situation where the existing logic does not provide us with natural tools for dealing with the types we desire to capture.

The **main contribution** of this work is the development of Multiparty Classical Processes (MCP), a proof theory for reasoning on multiparty communications. The key aspect of MCP is that it generalises Classical Linear Logic (CLL) [12], by building on a new notion of type compatibility, called *coherence*, that replaces duality. Using MCP, we can provide a concise reconstruction of the foundations of MPSTs. In the following, we outline our investigation:

- *Coherence.* We start by formalising a language for *local types* and *global types* (§ 3, Types). As in MPSTs, a local type denotes the I/O actions of a single participant in a session, whereas a global type denotes the desired interactions among all participants in a session. We then present *coherence*, a proof system for determining whether a set of local types follow the scenario denoted by a global type (§ 3, Coherence). We prove the adequacy of coherence by showing that global types are proof terms for coherence proofs (§ 3, Figure 2); equivalences between coherence proofs correspond to the equivalences between global types originally formulated with an auxiliary definition in [6] (§ 3, Proposition 2); and, the coherence proof system yields projection and extraction procedures from global types to local types and vice versa (§ 3, Proposition 3 and Proposition 4). Finally, we show that coherence generalises the notion of duality in CLL (§ 3, Proposition 7). Our extraction procedure is the first not requiring auxiliary conditions (e.g., dependency relations as in [15]) and capturing nested protocols [10].
- *Multiparty Classical Processes.* We present Multiparty Classical Processes (MCP), a proof theory that is in a Curry-Howard correspondence with a language for multiparty sessions (§ 4). The key aspect of MCP is using coherence as a new principle for compositionality in order to generalise the standard cut rule of linear logic, by allowing an arbitrary number of proofs to be composed (§ 4, Figure 6). Such a generalisation allows for the first time to specify cyclic inter-connected networks using (a generalisation of) linear logic whilst preserving its normalisation properties (§ 7). From the proof theory of MCP, we derive logically-founded notions of structural equivalences and reductions for multiparty processes (§ 4, Figure 7 and Figure 8). Driven by the correspondence between processes and proofs, we show that: communications among processes always follow their session types (§ 5, Theorem 10); communications never get stuck (§ 5, Corollary 12), improving on previous techniques for analysing progress with multiparty session types (§ 7); and that protocols used to type processes are always eventually executed (§ 5, Theorem 13).

2 Preview

We give an informal introduction to MCP with the 2-buyer protocol [14], where two buyers buy a book together from a seller. This can be described by the following global type:

$$\begin{array}{ll} 1. & \mathsf{B1} \to \mathsf{S} : \langle \mathbf{str} \rangle;\ \mathsf{S} \to \mathsf{B1} : \langle \mathbf{int} \rangle;\ \mathsf{S} \to \mathsf{B2} : \langle \mathbf{int} \rangle;\ \mathsf{B1} \to \mathsf{B2} : \langle \mathbf{int} \rangle; \\ 2. & \mathsf{B2} \to \mathsf{S} : \&(\mathsf{B2} \to \mathsf{S} : \langle \mathbf{addr} \rangle; \mathsf{end},\quad \mathsf{end}) \end{array} \tag{1}$$

Above, B1 (the first buyer), B2 (the second buyer) and S (the seller) are *roles*. In Line 1, B1 sends the book title to S, then S sends a quote to B1 and B2. At this point, B1 sends to B2 the fraction of the price it wishes to pay. In Line 2, B2 communicates to S whether (&) to proceed with the purchase and, if so, also an address for the delivery.

In multiparty session types, each role in a global type is implemented by a different

process. For example, the following three programs implement the roles in (1):

$$\begin{aligned}
\text{Buyer1} &\stackrel{\text{def}}{=} \quad \overline{x}^{\,\mathsf{B1}\,\mathsf{S}}(\mathit{title});\ x^{\mathsf{B1}\,\mathsf{S}}(\mathit{quote});\ \overline{x}^{\,\mathsf{B1}\,\mathsf{B2}}(\mathit{contr}) \\
\text{Buyer2} &\stackrel{\text{def}}{=} \quad x^{\mathsf{B2}\,\mathsf{S}}(\mathit{quote});\ x^{\mathsf{B2}\,\mathsf{B}_1}(\mathit{contr});\ \big(\ x^{\mathsf{B2}\,\mathsf{S}}.\mathsf{inl};\ \overline{x}^{\,\mathsf{B2}\,\mathsf{S}}(\mathit{addr}) \quad + \quad x^{\mathsf{B2}\,\mathsf{S}}.\mathsf{inr}\ \big) \\
\text{Seller} &\stackrel{\text{def}}{=} \quad x^{\mathsf{S}\,\mathsf{B1}}(\mathit{title});\ \overline{x}^{\,\mathsf{S}\,\mathsf{B1}}(\mathit{quote});\ \overline{x}^{\,\mathsf{S}\,\mathsf{B2}}(\mathit{quote});\ x^{\mathsf{S}\,\mathsf{B2}}\&.\mathsf{case}\big(\ x^{\mathsf{S}\,\mathsf{B2}}(\mathit{addr}),\quad \mathbf{0}\ \big)
\end{aligned}$$

The three processes above are defined in the π-calculus with multiparty sessions [9], and communicate using the session (or channel) x. In term Buyer1, $x^{\mathsf{B1}\,\mathsf{S}}(\mathit{title})$ means "as role B1, send the book *title* over channel x to the process implementing role S"; $\overline{x}^{\,\mathsf{B1}\,\mathsf{S}}(\mathit{quote})$ means "as role B1, receive a quote over channel x from the process implementing role S"; finally, $x^{\mathsf{B1}\,\mathsf{B2}}(\mathit{contr})$ means "as role B1, send to the process implementing role B2, over channel x, the amount the first buyer is willing to contribute with". Note that Buyer2 makes a choice after receiving the contribution from Buyer1, i.e., it either accepts or rejects the purchase by respectively selecting the left or right branch of the case construct in the code of Seller.

Following the approach in [23], we can type channel x using CLL propositions (differently from [23], we use $\parr$ to type outputs and $\otimes$ to type inputs, see § 7):

$$\begin{aligned}
&\text{usage of } x \text{ in Buyer1:} && \mathbf{str} \parr \mathbf{int} \otimes \mathbf{int} \parr \mathsf{end} \\
&\text{usage of } x \text{ in Buyer2:} && \mathbf{int} \otimes \mathbf{int} \otimes \big((\mathbf{addr} \parr \mathsf{end}) \oplus \mathsf{end}\big) \\
&\text{usage of } x \text{ in Seller:} && \mathbf{str} \otimes \mathbf{int} \parr \mathbf{int} \parr \big((\mathbf{addr} \otimes \mathsf{end}) \,\&\, \mathsf{end}\big)
\end{aligned} \tag{2}$$

Above, each proposition states how x is used by each process. For instance, Buyer1 outputs ($\parr$) a string, receives ($\otimes$) an integer, sends another integer and finally terminates (end).

CLL cannot compose our three processes using the above specifications, since its composition rule Cut can only compose two processes, which communicate over a same channel x with compatible binary session types A and $A^{\perp}$:

$$\frac{P \vdash \Delta, x{:}A \qquad Q \vdash \Delta', x{:}A^{\perp}}{(\boldsymbol{\nu} x{:}A)\,(P \mid Q) \vdash \Delta, \Delta'}\ \mathsf{Cut}$$

Using the same channel among our three processes is essential for tracking the dependencies expressed by the global type in (1): for example, we need to ensure that Seller sends a quote to Buyer2 only after it has received a request for a book from Buyer1. Such constraints cannot be tracked by binary session types [14]. To overcome this issue, we annotate each connective in propositions with roles. For example, the type of x for Buyer1 would become:

$$\begin{aligned}
&\text{annotated usage of } x \text{ in Buyer1:} && \mathbf{str} \parr^{\mathsf{S}} \mathbf{int} \otimes^{\mathsf{S}} \mathbf{int} \parr^{\mathsf{B2}} \mathsf{end} \\
&\text{annotated usage of } x \text{ in Buyer2:} && \mathbf{int} \otimes^{\mathsf{S}} \mathbf{int} \otimes^{\mathsf{B1}} \big((\mathbf{addr} \parr^{\mathsf{S}} \mathsf{end}) \oplus^{\mathsf{S}} \mathsf{end}\big) \\
&\text{annotated usage of } x \text{ in Seller:} && \mathbf{str} \otimes^{\mathsf{B1}} \mathbf{int} \parr^{\mathsf{B1}} \mathbf{int} \parr^{\mathsf{B2}} \big((\mathbf{addr} \otimes^{\mathsf{B2}} \mathsf{end}) \,\&^{\mathsf{B2}}\, \mathsf{end}\big)
\end{aligned} \tag{3}$$

Annotations identify the dual role for each action, e.g., the usage for Buyer1 now reads: send a string to S ($\parr^{\mathsf{S}}$); receive an integer from S ($\otimes^{\mathsf{S}}$); send an integer to B2 ($\parr^{\mathsf{B2}}$); and, terminate (end). We can then reformulate rule Cut as:

$$\frac{P_i \vdash \Gamma_i, x^{p_i}{:}A_i \qquad G \vDash \{p_i{:}A_i\}_i}{(\boldsymbol{\nu} x{:}G)\,\big(\textstyle\prod_i P_i\big) \vdash \{\Gamma_i\}_i}\ \mathsf{MCut}$$

In our new *multiparty* cut rule MCut, if some processes P_i use session x as role p_i (denoted x^{p_i}), each according to some respective types A_i, and such types coherently follow a global type specification G (formalised by the judgement $G \vDash \{p_i{:}A_i\}_i$), then we can compose them in parallel within the scope of session x, written $(\boldsymbol{\nu} x{:}G)\,(P_1 \mid \ldots \mid P_n)$. In our example,

$$
\begin{array}{rcll}
A, B, \ldots & ::= & 1 \quad \textit{(unit for } \otimes\textit{)} & \mid \quad \bot \quad \textit{(unit for } \parr\textit{)} \\
& \mid & A \parr^{\tilde{p}} B \quad \textit{(send } A \textit{ to } \tilde{p}\textit{, then } B\textit{)} & \mid \quad A \otimes^{p} B \quad \textit{(receive } A \textit{ from } p\textit{, then } B\textit{)} \\
& \mid & A \oplus^{\tilde{p}} B \quad \textit{(select } A \textit{ or } B \textit{ in } \tilde{p}\textit{)} & \mid \quad A \& ^{p} B \quad \textit{(offer } A \textit{ or } B \textit{ to } p\textit{)} \\
& \mid & !A \quad \textit{(client request)} & \mid \quad ?A \quad \textit{(server accept)}
\end{array}
$$

$$
G \quad ::= \quad p \to \tilde{q} : \langle G' \rangle ; G \quad \mid \quad p \to \tilde{q} : \&(G_1, G_2) \quad \mid \quad ?p \to !\tilde{q} : \langle G \rangle \quad \mid \quad \mathsf{end}^{p\tilde{q}}
$$

Figure 1 Local Types ($A, B, \ldots$) and Global Types (G).

for i ranging from 1 to 3, $\{p_i : A_i\}_i$ would correspond to the types in (3), where p_1, p_2 and p_3 would be, respectively, Buyer1, Buyer2 and Seller. In § 6, we will show that such types coherently follow the global type given in (1).

MCP goes beyond the original multiparty session types [14], capturing also multicasting and nested protocols [9, 10]. For example, we can enhance the 2-buyer protocol as:

$$
\begin{array}{ll}
1. & \mathsf{B1} \to \mathsf{S} : \langle \mathbf{str} \rangle ;\ \mathsf{S} \to \mathsf{B1}, \mathsf{B2} : \langle \mathbf{int} \rangle ;\ \mathsf{B1} \to \mathsf{B2} : \langle \mathbf{int} \rangle ; \\
2. & \mathsf{B2} \to \mathsf{B1}, \mathsf{S} : \&\Big(\mathsf{B2} \to \mathsf{S} : \langle \mathbf{addr} \rangle ;\ \mathsf{end}, \quad \mathsf{B1} \to \mathsf{S} : \langle G_{\mathsf{sub}} \rangle ; \mathsf{B1} \to \mathsf{S} : \langle \mathbf{str} \rangle ; \mathsf{B2} \to \mathsf{S} : \langle \mathbf{str} \rangle ; \mathsf{end} \Big)
\end{array}
\tag{4}
$$

Above, S multicasts the price to both B1 and B2; and B2 multicasts its decision to B1 and S. We have also updated the right branch of the choice using a nested protocol G_{sub}, which is private to B1 and S, where B1 tells S whether it wants to purchase the product alone:

$$
G_{\mathsf{sub}} \quad = \quad \mathsf{B1} \to \mathsf{S} : \&\Big(\mathsf{B1} \to \mathsf{S} : \langle \mathbf{addr} \rangle ;\ \mathsf{end}, \quad \mathsf{B1} \to \mathsf{S} : \langle \mathbf{str} \rangle ; \mathsf{end} \Big)
$$

In MCP, nested protocols can proceed in parallel to their originating protocols. For example, the last two communications, where B1 and B2 inform S of their respective reasons for not completing the purchase, can be executed in parallel to G_{sub}. We will formalise this in § 5.

3 Coherence

We give a proof-theoretical reconstruction of coherence, from [14]. Our theory generalises duality, from CLL, to checking the compatibility of multiple types. We define coherence as a proof system for deriving sets of (compatible) *local types*, which describe the local behaviours of participants in a multiparty session. Global types are proof terms for coherence proofs, yielding a correspondence between sets of compatible local types and their global descriptions.

Types. The syntax of local and global types is given in Figure 1, where p, q range over a set of *roles*. Global types are highlighted, to distinguish them as proof terms. Highlighting is also used in our syntax of local types, to show the difference with CLL. We will adopt the same convention in § 4 when we present more terms.

A local type A describes the local behaviour of a role in a session. Types 1 and $\bot$ denote session termination, respectively representing the request and the acceptance for closing a session (which were informally abstracted by end in our previous examples). A type $A \parr^{\tilde{p}} B$ denotes a multicast output of a session with type A to roles $\tilde{p}$, with a continuation B. A type $A \otimes^{p} B$ represents an input of a session with type A from role p, with continuation B. Types $A \oplus^{\tilde{p}} B$ and $A \&^{p} B$ denote, respectively, the output of a choice between the continuations A and B to roles $\tilde{p}$ and the input of a choice from role p. The replicated type $!A$ offers behaviour A as many times as requested. Finally, type $?A$ requests the execution of a replicated type and proceeds as A.

$$\frac{G \vDash \Theta,\ p{:}B,\ \{q_i{:}D_i\}_i \qquad G' \vDash p{:}A,\ \{q_i{:}C_i\}_i}{p \to \tilde{q} : \langle G' \rangle ; G \vDash \Theta,\ p{:}A \parr^{\tilde{q}} B,\ \{q_i{:}C_i \otimes^{p} D_i\}_i}\ \otimes\parr \qquad \frac{}{\mathsf{end}^{p\tilde{q}} \vDash p{:}\bot, q_1{:}1, \ldots, q_n{:}1}\ 1\bot$$

$$\frac{G_1 \vDash \Theta, p{:}A, \{q_i{:}C_i\}_i \qquad G_2 \vDash \Theta, p{:}B, \{q_i{:}D_i\}_i}{p \to \tilde{q} : \&(G_1, G_2) \vDash \Theta, p{:}A \oplus^{\tilde{q}} B, \{q_i{:}C_i \&^{p} D_i\}_i}\ \oplus\& \qquad \frac{G \vDash p{:}A, \{q_i{:}B_i\}_i}{?p \to !\tilde{q} : \langle G \rangle \vDash p{:}?A, \{q_i{:}!B_i\}_i}\ !?$$

Figure 2 Coherence.

A global type G describes the behaviour of many participants. In $p \to \tilde{q} : \langle G' \rangle ; G$, role p sends to roles $\tilde{q}$ a message to create a new session of type G', and then the protocol proceeds as G. In $p \to \tilde{q} : \&(G_1, G_2)$, role p communicates to roles $\tilde{q}$ its choice of either branch G_1 or G_2. A type $?p \to !\tilde{q} : \langle G \rangle$ denotes that role p may ask roles $\tilde{q}$ to execute G many times. Finally, in $\mathsf{end}^{p\tilde{q}}$, role p asks roles $\tilde{q}$ to terminate the session (for brevity, we often write end).

Judgements. A *role typing* $p{:}A$ states that role p behaves as specified by type A. Our *judgements* for coherence have the form $G \vDash p_1{:}A_1, \ldots, p_n{:}A_n$ which reads as "the types $A_1, \ldots, A_n$ of the respective roles $p_1, \ldots, p_n$ are compatible and follow the global type G". We use Θ to range over sets of role typings, and make the standard assumption that we can write $\Theta, p{:}A$ only if a role typing for p does not appear in Θ. Given some roles $\tilde{p}$, we use the notation $\{p_i{:}A_i\}_i$ to denote the set of role typings $p_1{:}A_1, \ldots, p_n{:}A_n$, assuming $\tilde{p} = p_1, \ldots, p_n$ and i ranging from 1 to n. Given G, we say that G is *valid* if there exists Θ such that $G \vDash \Theta$. Conversely, given Θ, we say that Θ is *coherent* if there exists G such that $G \vDash \Theta$.

We report the rules for deriving coherence judgements in Figure 2.

Rule $\otimes\parr$ matches the output type from role p to roles $\tilde{q}$ with the input types of roles $\tilde{q}$, whenever (i) the types for the newly created session are coherent and (ii) the types of all continuations are also coherent. Rule $\oplus\&$ checks that both possibilities in a choice are coherent, where all roles participating in the communication are allowed to have different behaviour and the other roles are not (a multicast generalisation of [14]). In rule !?, we check that a client requests the creation of a coherent session only from replicated services. Finally, rule $1\bot$ checks that all participants agree on the termination of a protocol. As in CLL, we interpret type 1 as a terminated process and $\bot$ as a process that has terminated its behaviour in a session and proceeds with other sessions. Therefore, we read rule $1\bot$ as "a protocol terminates when one participant waits (type $\bot$) for the termination of all the others (type 1), which execute in parallel". This design choice simplifies our development; we discuss a generalisation in § 7.

▶ **Example 1** (2-Buyer Protocol). We can revisit the local types for the 2-buyer protocol in § 2 (1), where now data types are abstracted by 1's and $\bot$'s.

$$A \stackrel{\text{def}}{=} \bot \parr^{\mathsf{S}} 1 \otimes^{\mathsf{S}} \bot \parr^{\mathsf{B2}} \bot \qquad B \stackrel{\text{def}}{=} 1 \otimes^{\mathsf{S}} 1 \otimes^{\mathsf{B1}} \left((\bot \parr^{\mathsf{S}} 1) \oplus^{\mathsf{S}} 1 \right) \qquad C \stackrel{\text{def}}{=} 1 \otimes^{\mathsf{B1}} \bot \parr^{\mathsf{B1}} \bot \parr^{\mathsf{B2}} \left((1 \otimes^{\mathsf{B2}} 1) \ \&^{\mathsf{B2}} \ 1 \right)$$

Let G be the global type in (1) with end instead of data types; then, $G \vDash \mathsf{B1}{:}A, \mathsf{B2}{:}B, \mathsf{S}{:}C$.

3.1 Properties of Coherence

Swapping. Immediately from our correspondence between global types and coherence proofs, we can reconstruct the standard notion of swapping $\simeq^{\mathsf{g}}$ for global types from [6]. Intuitively, two communications involving different roles can always be swapped, capturing

$$\frac{\{p,\tilde{q}\} \cap \{r,\tilde{s}\} = \emptyset}{p \to \tilde{q} : \langle G \rangle; r \to \tilde{s} : \langle G' \rangle; G'' \simeq^{\mathsf{g}} r \to \tilde{s} : \langle G' \rangle; p \to \tilde{q} : \langle G \rangle; G''} \ (\to\to)$$

$$\frac{\{p,\tilde{q}\} \cap \{r,\tilde{s}\} = \emptyset}{p \to \tilde{q} : \&(r \to \tilde{s} : \langle G \rangle; G_1, r \to \tilde{s} : \langle G \rangle; G_2) \simeq^{\mathsf{g}} r \to \tilde{s} : \langle G \rangle; p \to \tilde{q} : \&(G_1, G_2)} \ (\to\oplus)$$

$$\frac{\{p,\tilde{q}\} \cap \{r,\tilde{s}\} = \emptyset}{p \to \tilde{q} : \&(r \to \tilde{s} : \&(G_1, G_2), r \to \tilde{s} : \&(G_3, G_4)) \simeq^{\mathsf{g}} r \to \tilde{s} : \&(p \to \tilde{q} : \&(G_1, G_3), p \to \tilde{q} : \&(G_2, G_4))} \ (\oplus\oplus)$$

Figure 3 Swapping relation $\simeq^{\mathsf{g}}$ for global types.

the fact that separate roles execute concurrently. For example, the following coherence proof (for p, q, r, s different):

$$\frac{\dfrac{G \vDash \Theta, p{:}A', q{:}B', r{:}C', s{:}D' \quad G'' \vDash \tilde{\Theta}, r{:}C, s{:}D}{r \to s : \langle G'' \rangle; G \vDash \Theta, p{:}A', q{:}B', r{:}C \parr^{s} C', s{:}D \otimes^{r} D'} \otimes\parr \qquad G' \vDash p{:}A, q{:}B}{p \to q : \langle G' \rangle; r \to s : \langle G'' \rangle; G \vDash \Theta,\ p{:}A \parr^{q} A',\ q{:}B \otimes^{p} B',\ r{:}C \parr^{s} C',\ s{:}D \parr^{r} D',} \otimes\parr$$

is equivalent to $(\simeq^{\mathsf{g}})$

$$\frac{\dfrac{G \vDash \Theta, p{:}A', q{:}B', r{:}C', s{:}D' \quad G' \vDash \tilde{\Theta}, p{:}A, q{:}B}{p \to q : \langle G' \rangle; G \vDash \Theta, p{:}A \parr^{q} A', q{:}B \otimes^{p} B', r{:}C', s{:}D'} \otimes\parr \qquad G'' \vDash r{:}C, s{:}D}{r \to s : \langle G'' \rangle; p \to q : \langle G' \rangle; G \vDash \Theta,\ p{:}A \parr^{q} A',\ q{:}B \otimes^{p} B',\ r{:}C \parr^{s} C',\ s{:}D \parr^{r} D'} \otimes\parr$$

proving that $p \to q : \langle G' \rangle; r \to s : \langle G'' \rangle; G$ is equivalent to $r \to s : \langle G'' \rangle; p \to q : \langle G' \rangle; G$. Figure 3 reports all cases for $\equiv$, derived from the proof system in Figure 2.

In general, two global types are proof terms for the same set of local typings if and only if they are equivalent.

▶ **Proposition 2** (Swapping). *Let $G \vDash \Theta$. Then, $G \simeq^{\mathsf{g}} G'$ if and only if $G' \vDash \Theta$.*

Projection and Extraction. The hallmark of the theory of multiparty session types is projection: developers can write protocols as global types, and then automatically project a global type onto a set of local types that can be used to modularly verify the behaviour of each participant. As there is only one possible rule application for each production in the syntax of global types, we can construct an algorithm that traverses the structure of G:

▶ **Proposition 3** (Projection). *For G valid, Θ such that $G \vDash \Theta$ is computable in linear time.*

We can also use coherence for the inverse procedure, i.e., the extraction of a global type from a set of local typings Θ. If Θ is coherent, we can just apply the first applicable coherence rule, noting that the sizes of the local types in the premises always get smaller:

▶ **Proposition 4** (Extraction). *For Θ coherent, G such that $G \vDash \Theta$ is computable.*

▶ **Example 5.** In the 2-buyer protocol, $G \vDash \mathsf{B1}{:}A, \mathsf{B2}{:}B, \mathsf{S}{:}C$ implies: (i) we can infer A, B and C from G (proposition 3) and (ii) we can extract G from $\mathsf{B1}{:}A, \mathsf{B2}{:}B, \mathsf{S}{:}C$ (proposition 4).

Global reductions. We define reductions for global types, denoted $\tilde{G} \rightsquigarrow \tilde{G}'$, where $\tilde{G}$ is a set $\{G_1, \ldots, G_n\}$. *Global type reductions are just a convention* (recalling [6]), which we use in § 5 to concisely formalise how processes follow their protocols. Formally, $\rightsquigarrow$ is the smallest relation satisfying the rules in Figure 4.

Rule $\mathsf{g}_{\otimes\parr}$ models a communication that creates a new session of type G', which will then proceed in parallel to the continuation G. Rule $\mathsf{g}_{1\bot}$ models session termination. Rules

$$(\mathsf{g}_{\otimes\parr}) \quad \{\, p \to \tilde{q} : \langle G' \rangle; G \,\} \rightsquigarrow \{\, G, G' \,\} \qquad (\mathsf{g}_{!C}) \quad \{\, ?p \to !\tilde{q} : \langle G \rangle \,\} \rightsquigarrow \{\, G, ?p \to !\tilde{q} : \langle G \rangle \,\}$$
$$(\mathsf{g}_{\oplus\&1}) \quad \{\, p \to \tilde{q} : \&(G_1, G_2) \,\} \rightsquigarrow \{\, G_1 \,\} \qquad (\mathsf{g}_{\oplus\&2}) \quad \{\, p \to \tilde{q} : \&(G_1, G_2) \,\} \rightsquigarrow \{\, G_2 \,\}$$
$$(\mathsf{g}_{!?}) \quad \{\, ?p \to !\tilde{q} : \langle G \rangle \,\} \rightsquigarrow \{\, G \,\} \qquad (\mathsf{g}_{!W}) \quad \{\, ?p \to !\tilde{q} : \langle G \rangle \,\} \rightsquigarrow \emptyset \qquad (\mathsf{g}_{1\bot}) \quad \{\, \mathsf{end}^{p\tilde{q}} \,\} \rightsquigarrow \emptyset$$
$$(\mathsf{g}_{\mathsf{ctx}}) \quad \tilde{G}_1 \rightsquigarrow \tilde{G}_2 \;\Rightarrow\; \tilde{G} \cup \tilde{G}_1 \rightsquigarrow \tilde{G} \cup \tilde{G}_2 \qquad (\mathsf{g}_{\mathsf{eq}}) \quad \tilde{G}_0 \simeq^{\mathsf{g}} \tilde{G}_1,\ \tilde{G}_1 \rightsquigarrow \tilde{G}_2,\ \tilde{G}_2 \simeq^{\mathsf{g}} \tilde{G}_3 \;\Rightarrow\; \tilde{G}_0 \rightsquigarrow \tilde{G}_3$$

Figure 4 Global Types, reduction semantics.

$\mathsf{g}_{\oplus\&1}$ and $\mathsf{g}_{\oplus\&2}$ model the execution of a choice. In rules $\mathsf{g}_{!?}$, $\mathsf{g}_{!C}$ and $\mathsf{g}_{!W}$, a replicated protocol can be respectively executed exactly once, multiple, or zero times. Rule $\mathsf{g}_{\mathsf{ctx}}$ lifts the behaviour of a protocol to a set of protocols executing concurrently. Finally, rule $\mathsf{g}_{\mathsf{swap}}$ allows for swappings in a global type, where $\tilde{G} \simeq^{\mathsf{g}} \tilde{G}'$ is the point-wise extension of the swapping relation $\simeq^{\mathsf{g}}$ to sets. Our semantics preserves validity:

▶ **Theorem 6** (Coherence Preservation). *If $\tilde{G}$ is valid and $\tilde{G} \rightsquigarrow \tilde{G}'$, then $\tilde{G}'$ is valid.*

▶ Remark. Rule $\mathsf{g}_{!?}$ can be derived from rules $\mathsf{g}_{!C}$ and $\mathsf{g}_{!W}$. Including it simplifies our presentation, since each global type reduction corresponds to a communication in MCP (§ 5).

Coherence as generalised duality. Coherence is a generalisation of duality (from CLL [12]): in the degenerate case of a session with two participants, the two notions coincide. We recall the definition of duality $X^\bot$, defined inductively on the syntax of linear logic propositions:

$$(X \otimes Y)^\bot = X^\bot \parr Y^\bot \qquad (X \parr Y)^\bot = X^\bot \otimes Y^\bot \qquad 1^\bot = \bot \qquad \bot^\bot = 1$$
$$(X \oplus Y)^\bot = X^\bot \& Y^\bot \qquad (X \& Y)^\bot = X^\bot \oplus Y^\bot \qquad (!X)^\bot = ?X^\bot \qquad (?X)^\bot = !X^\bot$$

We define a partial encoding $[\![\cdot]\!]$ from local types into linear logic propositions:

$$[\![1]\!] = 1 \quad [\![\bot]\!] = \bot \qquad [\![!A]\!] = !\,[\![A]\!] \quad [\![?A]\!] = ?\,[\![A]\!] \quad [\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$$
$$[\![A \parr^q B]\!] = [\![A]\!] \parr [\![B]\!] \quad [\![A \oplus^p B]\!] = [\![A]\!] \oplus [\![B]\!] \qquad [\![A \&^p B]\!] = [\![A]\!] \& [\![B]\!]$$

The encoding $[\![\cdot]\!]$ is defined only when $\parr$ and $\oplus$ are annotated with a single role. We get:

▶ **Proposition 7** (Coherence as Duality). *Let A, B be propositions where all subterms of the form $C \parr^{\tilde{p}} D$ or $C \oplus^{\tilde{p}} D$ are such that $\tilde{p} = q$ for some q. Then, $G \vDash p{:}A, q{:}B$ iff $[\![A]\!] = [\![B]\!]^\bot$.*

4 Multiparty Classical Processes

In this section, we present Multiparty Classical Processes (MCP). MCP captures dependencies among actions performed by different participants in a multiparty session, whereas, in previous work, actions among different pairs of participants must be independent [4, 23].

Environments. Let Γ, Δ range over typing environments: $\Gamma, \Delta \;::=\; \cdot \;\mid\; \Gamma, x^p{:}A$. Intuitively, $x^p{:}A$ means that role p in session x follows behaviour A. We write $?\Delta$ whenever Δ contains only types of the form $?A$, and write $\Delta, x^p{:}A$ only when x^p does not appear in Δ.

Processes. We report the syntax of processes in Figure 5.

In MCP, both input and output names are bound, as in [23]. Term *(send)* creates a new session y and sends it, as role p, to the processes respectively playing roles $\tilde{q}$ in session x; then, the process proceeds as P. The dual operation *(recv)* receives, as role p in session x, a fresh session y from the process playing role q; the process then proceeds as the parallel

$$
\begin{array}{rcll}
P, Q, R & ::= & \overline{x}^{pq}(y); P & \textit{(send)} \\
& \mid & x^{pq}.\mathsf{inl}; P & \textit{(left sel)} \\
& \mid & x^{pq}.\mathsf{case}(P, Q) & \textit{(case)} \\
& \mid & \mathsf{close}\, x^p & \textit{(close)} \\
& \mid & !x^p(y); P & \textit{(service)} \\
& \mid & P + Q & \textit{(choice)} \\
& \mid & x^{pq}(y); (P \mid Q) & \textit{(recv)} \\
& \mid & x^{pq}.\mathsf{inr}; P & \textit{(right sel)} \\
& \mid & (\boldsymbol{\nu} x{:}G)\left(\prod_i P_i\right) & \textit{(res)} \\
& \mid & \mathsf{wait}\, x^p; P & \textit{(wait)} \\
& \mid & ?x^p(y); P & \textit{(client)}
\end{array}
$$

Figure 5 MCP, syntax of processes.

$$
\frac{P \vdash \Gamma, y^p{:}A \quad Q \vdash \Delta, x^p{:}B}{x^{pq}(y); (P \mid Q) \vdash \Gamma, \Delta, x^p{:}A \otimes^q B}\ \otimes
\qquad
\frac{P \vdash \Gamma, y^p{:}A, x^p{:}B}{\overline{x}^{p\tilde{q}}(y); P \vdash \Gamma, x^p{:}A \parr^{\tilde{q}} B}\ \parr
$$

$$
\frac{P_i \vdash \Gamma_i, x^{p_i}{:}A_i \quad G \vDash \{p_i{:}A_i\}_i}{(\boldsymbol{\nu} x{:}G)\left(\prod_i P_i\right) \vdash \{\Gamma_i\}_i}\ \mathsf{MCut}
\qquad
\frac{P \vdash \Gamma, x^p{:}A \quad Q \vdash \Gamma, x^p{:}B}{x^{pq}.\mathsf{case}(P, Q) \vdash \Gamma, x^p{:}A \,\&^q\, B}\ \&
$$

$$
\frac{P \vdash \Gamma \quad Q \vdash \Gamma}{P + Q \vdash \Gamma}\ +
\qquad
\frac{P \vdash \Gamma, x^p{:}A}{x^{p\tilde{q}}.\mathsf{inl}; P \vdash \Gamma, x^p{:}A \oplus^{\tilde{q}} B}\ \oplus_1
\qquad
\frac{P \vdash \Gamma, x^p{:}B}{x^{p\tilde{q}}.\mathsf{inr}; P \vdash \Gamma, x^p{:}A \oplus^{\tilde{q}} B}\ \oplus_2
$$

$$
\frac{}{\mathsf{close}\, x^p \vdash x^p{:}1}\ 1
\qquad
\frac{P \vdash \Gamma}{\mathsf{wait}\, x^p; P \vdash \Gamma, x^p{:}\bot}\ \bot
\qquad
\frac{P \vdash \Gamma}{P \vdash \Gamma, x^p{:}\ ?A}\ \mathsf{Weaken}
$$

$$
\frac{P \vdash ?\Gamma, y^p{:}A}{!x^p(y); P \vdash ?\Gamma, x^p{:}\ !A}\ !
\qquad
\frac{P \vdash\ \Gamma, y^p{:}A}{?x^p(y); P \vdash\ \Gamma, x^p{:}\ ?A}\ ?
\qquad
\frac{P \vdash\ \Gamma, y^p{:}\ ?A, z^p{:}\ ?A}{P[x/y][x/z] \vdash\ \Gamma, x^p{:}\ ?A}\ \mathsf{Contract}
$$

Figure 6 MCP, typing rules.

composition of P (dedicated to session y) and Q (dedicated to continuing session x). Similarly, terms *(left sel)* and *(right sel)* multicast a selection of a left or right branch respectively to the processes playing roles $\tilde{q}$ in session x, as role p. A selection is received by term *(case)*, which offers the two selectable branches. Terms *(close)* and *(wait)* terminate a session. Term *(choice)* is the standard non-deterministic choice. In a restriction *(res)*, x is bound in the processes P_i; we use the standard type annotation (as in [23]) to show the relation between the semantics of processes and global types in § 5. In term $x^{p\,q}(y); (P \mid Q)$, y is bound in P but not in Q. In terms $\overline{x}^{\,p\,\tilde{q}}(y)P$, $!x^p(y); P$, and $?x^p(y); P$, y is bound in P.

Judgements. Judgements in MCP have the form $P \vdash x_1^{p_1}{:}A_1, \ldots, x_n^{p_n}{:}A_n$, meaning that process P implements roles p_i in the respective session x_i with behaviour A_i.

Rules. We report the rules of MCP in Figure 6. Intuitively, a process is typed with local types; then, we use coherence to check that the local types of composed processes (rule MCut) coherently implement a global type. All rules are defined up to context exchange.

Rule MCut is central: it extends the Cut of CLL to composing in parallel an arbitrary number of P_i that communicate using session x. The rule checks that the composition of the respective local behaviours of the composed processes is coherent ($G \vDash \{p_i{:}A_i\}_i$). In the conclusion, $\{\Gamma_i\}_i$ is the disjoint union of all Γ_i in the premise.

Rule $\otimes$ types an input $x^{pq}(y); (P \mid Q)$, where the subprocess P plays role p with behaviour A in the received multiparty session y; session x then proceeds by following behaviour B for role p in Q. Observe that the $\otimes$ is annotated with the role q that p wishes

to receive from. The multicast output $\overline{x}^{p\tilde{q}}(y); P$ in rule $\parr$ creates a new session y and sends it, as role p in session x, to roles $\tilde{q}$. The new session y is used by P as role p with type A, assuming that the other processes receiving it implement the other roles (this assumption is checked by coherence in MCut, when processes are composed). We discuss in § 7 how to relax the constraint that the role p played in session y is the same.

Rules $\oplus_1$ and $\oplus_2$ type, respectively, the multicast of a left and right selection, by checking that the process continuation follows the expected local type. Similarly, rule & types a branching by checking that the continuations implement the respective expected local types.

Rule + types the nondeterministic process $P + Q$, by checking that both P and Q implement the same local behaviours. Observe that P and Q may still be substantially different, since they may (i) perform different selections on some sessions (as rules $\oplus_1$ and $\oplus_2$ can yield the same typing), and (ii) have different inner compositions of processes whose types have been hidden by rule MCut.

Rules 1 and $\bot$ type, respectively, the request and the acceptance for closing a multiparty session. Rules ! and ? type, respectively, the replicated offering of a service and its repeated usage (a client). Since a service typed by ! may be used multiple times, we require that its continuation does not use any linear behaviour ($?\Delta$). Rules Weaken and Contract type, respectively, the absence of clients or the presence of multiple clients. In rule Contract, sessions y and z are contracted into a single session x with a standard name substitution, provided that they have the same type $?A$.

▶ Remark. Removing proof terms from MCP yields a pure logic that differs from CLL only for rule MCut. Since we will prove in § 5 that MCut is admissible, just like Cut in CLL, the two systems subsume the same set of valid judgements. Nevertheless, as shown in next section, MCP yields a different operational meaning: reductions of MCut correspond to multiparty communications, whereas reductions of Cut in CLL correspond to binary communications.

5 Semantics

In this section, we demonstrate the consistency of MCP, by establishing a cut-elimination result that yields an operational semantics and important properties, e.g., deadlock-freedom.

5.1 Structural Equivalences as Commuting Conversions

MCP supports *commuting conversions*, permutations of applications of MCut that maintain the validity of judgements. As an example, consider the following proof equivalence ($\equiv$):

$$\dfrac{\dfrac{P \vdash \Delta, y^p{:}A, x^p{:}B, z^r{:}C}{\overline{x}^{p\tilde{q}}(y); P \vdash \Delta, x^p{:}A \parr^{\tilde{q}} B, z^r{:}C}\,\parr \qquad Q_i \vdash \Gamma_i, z^{s_i}{:}D_i \qquad G \vDash r{:}C, \{s_i{:}D_i\}_i}{(\boldsymbol{\nu} z{:}G)\left(\overline{x}^{p\tilde{q}}(y); P \mid \prod_i Q_i\right) \vdash \{\Gamma_i\}_i, \Delta, x^p{:}A \parr^{\tilde{q}} B}\,\mathsf{MCut}$$

$$\equiv$$

$$\dfrac{P \vdash \Delta, y^p{:}A, x^p{:}B, z^r{:}C \qquad \dfrac{Q_i \vdash \Gamma_i, z^{s_i}{:}D_i \qquad G \vDash r{:}C, \{s_i{:}D_i\}_i}{(\boldsymbol{\nu} z{:}G)\left(P \mid \prod_i Q_i\right) \vdash \{\Gamma_i\}_i, \Delta, y^p{:}A, x^p{:}B}\,\mathsf{MCut}}{\overline{x}^{p\tilde{q}}(y); (\boldsymbol{\nu} z{:}G)\left(P \mid \prod_i Q_i\right) \vdash \{\Gamma_i\}_i, \Delta, x^p{:}A \parr^{\tilde{q}} B}\,\parr$$

Above, an output is moved out of a restriction of a different session (or in it, reading in the other direction), as in [23]. In this example, the output process is the first in the parallel under the restriction; in general, this is not always the case since the process may be any of those in the parallel composition. In order to represent equivalences independently of the position of processes in a parallel, we use process contexts [20]. A context, denoted by $\mathcal{C}$, is a parallel composition with a hole: $\mathcal{C}[\cdot] ::= \cdot \mid \mathcal{C}[\cdot] \mid P \mid P \mid \mathcal{C}[\cdot]$. All equivalences are reported in Figure 7.

$$
\begin{array}{lrcll}
(\kappa_{\mathsf{par}}) & (\boldsymbol{\nu} z{:}G)\left(\prod_{i\in\tilde{k}} P_i\right) & \equiv & (\boldsymbol{\nu} z{:}G)\left(\prod_{j\in\tilde{k'}} P_j\right) & (\tilde{k} \text{ is a permutation of } \tilde{k'}) \\
(\kappa_{\mathsf{cut}}) & (\boldsymbol{\nu} x{:}G)\left(\mathcal{C}\left[(\boldsymbol{\nu} y{:}G')\,\mathcal{C}'[P]\right]\right) & \equiv & (\boldsymbol{\nu} y{:}G')\left(\mathcal{C}'\left[(\boldsymbol{\nu} x{:}G)\,\mathcal{C}[P]\right]\right) & (\text{if } x, y \in \mathsf{fn}(P)) \\
(\kappa_{\parr}) & (\boldsymbol{\nu} z{:}G)\left(\mathcal{C}\left[\overline{x}^{pq}(y); P\right]\right) & \equiv & \overline{x}^{pq}(y); (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P] & \\
(\kappa_{\otimes}) & (\boldsymbol{\nu} z{:}G)\left(\mathcal{C}\left[x^{pq}(y); (P \mid Q)\right]\right) & \equiv & x^{pq}(y); (P \mid (\boldsymbol{\nu} z{:}G)\,(\mathcal{C}[Q])) & (\text{if } z \notin \mathsf{fn}(P)) \\
(\kappa_{\oplus 1}) & (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[x^{pq}.\mathsf{inl}; P] \equiv x^{pq}.\mathsf{inl}; (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P] & & (\kappa_{\oplus 2}) & (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[x^{pq}.\mathsf{inr}; P] \equiv x^{pq}.\mathsf{inr}; (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P] \\
(\kappa_{\&}) & (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[x^{pq}.\mathsf{case}(P, Q)] & \equiv & x^{pq}.\mathsf{case}((\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P], (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[Q]) & \\
(\kappa_{!}) & (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[!x^{p}(y); P] \equiv !x^{p}(y); (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P] & & (\kappa_{?}) & (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[?x^{p}(y); P] \equiv ?x^{p}(y); (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P] \\
(\kappa_{\bot}) & (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[\mathsf{wait}\, x^{p}; P] & \equiv & \mathsf{wait}\, x^{p}; (\boldsymbol{\nu} z{:}G)\,\mathcal{C}[P] &
\end{array}
$$

Figure 7 MCP, Structural Equivalences.

$$
\begin{array}{ll}
(\beta_{\otimes\parr}) & (\boldsymbol{\nu} x{:}p\text{->}\tilde{q} : \langle G'\rangle;G)\left(\prod_i x^{q_i\,p}(y); (P_i \mid Q_i) \mid \overline{x}^{p\tilde{q}}(y); R \mid \prod_j P_j\right) \\
& \quad\to\quad (\boldsymbol{\nu} y{:}G')\left(\prod_i P_i \mid (\boldsymbol{\nu} x{:}G)\,(\prod_i Q_i \mid R \mid \prod_j P_j)\right) \\
(\beta_{\oplus\&1}) & (\boldsymbol{\nu} x{:}p\text{->}\tilde{q} : \&(G_1, G_2))\left(x^{p\tilde{q}}.\mathsf{inl}; P \mid \prod_i x^{q_i p}.\mathsf{case}(Q_i, R_i) \mid \prod_j P_j\right) \to (\boldsymbol{\nu} x{:}G_1)\left(P \mid \prod_i Q_i \mid \prod_j P_j\right) \\
(\beta_{\oplus\&2}) & (\boldsymbol{\nu} x{:}p\text{->}\tilde{q} : \&(G_1, G_2))\left(x^{p\tilde{q}}.\mathsf{inr}; P \mid \prod_i x^{q_i p}.\mathsf{case}(Q_i, R_i) \mid \prod_j P_j\right) \to (\boldsymbol{\nu} x{:}G_2)\left(P \mid \prod_i R_i \mid \prod_j P_j\right) \\
(\beta_{!?}) & (\boldsymbol{\nu} x{:}?p\text{->}!\tilde{q} : \langle G\rangle)\left(?x^{p}(y); P \mid \prod_i !x^{q_i}(y); Q_i\right) \to (\boldsymbol{\nu} y{:}G)\left(P \mid \prod_i Q_i\right) \\
(\beta_{!W}) & (\boldsymbol{\nu} x{:}?p\text{->}!\tilde{q} : \langle G\rangle)\left(\prod_i !x^{q_i}(y); Q_i \mid P\right) \to P \qquad \text{if } x \notin \mathsf{fn}(P) \\
(\beta_{!C}) & (\boldsymbol{\nu} x{:}?p\text{->}!\tilde{q} : \langle G\rangle)\left(\prod_i !x^{q_i}(w); Q_i \mid P[x/y][x/z]\right) \\
& \quad\to\quad (\boldsymbol{\nu} y{:}?p\text{->}!\tilde{q} : \langle G\rangle)\left(\prod_i !y^{q_i}(w); Q_i \mid (\boldsymbol{\nu} z{:}?p\text{->}!\tilde{q} : \langle G\rangle)\left(\prod_i !z^{q_i}(w); Q_i \mid P\right)\right) \\
(\beta_{1\bot}) & (\boldsymbol{\nu} x{:}\mathsf{end}^{p\tilde{q}})\left(\mathsf{wait}\, x^{p}; P \mid \prod_i \mathsf{close}\, x^{q_i}\right) \to P \\
(\beta_{+}) & (\boldsymbol{\nu} x{:}G)\left((P_1 + P_2) \mid \prod_i Q_i\right) \to (\boldsymbol{\nu} x{:}G)\left(P_j \mid \prod_i Q_i\right) \qquad j \in \{1, 2\}
\end{array}
$$

Figure 8 MCP, Cut Reductions.

The equivalence κ_{par} permutes processes in a parallel, since the premises of rule MCut can be in any order. In κ_{cut}, we can swap two restrictions, which corresponds to swapping two applications of rule MCut. The equivalence $\kappa_{\parr}$ shows that a restriction can always be swapped with an output on a different session. Similarly, the equivalence $\kappa_{\otimes}$ swaps a restriction with an input, requiring that the restricted name (z in this case) occurs free in P. In the case of $\oplus$, we have two equivalences, corresponding to the right and left selection respectively. For $\kappa_{\&}$, we can move a restriction to each branch of a case construct, also duplicating the context $\mathcal{C}$. Equivalences $\kappa_{!}$ and $\kappa_{?}$ allow to swap a restriction with a service and a client respectively. Finally, $\kappa_{\bot}$ is the case for $\mathsf{wait}\, x^{p}$. There is no equivalence for the process $\mathsf{close}\, x^{p}$ since it is only typable with the axiom 1.

5.2 Process Reductions as MCut Reductions

As for equivalences, we use our proof theory to derive reductions for processes, given in Figure 8.

In the reduction $\beta_{\otimes\parr}$, the output from role p to roles $\tilde{q}$ on session x is matched with the inputs at such roles, creating a new session y, following the global type of x. Reductions $\beta_{\oplus\&1}$ and $\beta_{\oplus\&2}$ capture the left and right multicast selection of a branching, respectively. In $\beta_{!?}$, a set of services with a single client is reduced to the composition of the bodies of such

services with that of the client; the type $?p \texttt{->} !\tilde{q} : \langle G \rangle$ of x is correspondingly reduced to G. Reduction $\beta_{!W}$ garbage collects a set of unused services. In $\beta_{!C}$, instead, a set of services is replicated to handle multiple clients. Finally, reduction $\beta_{1\perp}$ terminates a session x.

5.3 Properties

In the remainder, we abuse the notation $P \to P'$ to refer to process reductions closed up to our structural equivalence $\equiv$, as in standard process calculi.

Processes and Types. Since both equivalences and reductions are derived from judgement-preserving proof transformations, we immediately obtain the following two properties:

▶ **Theorem 8** (Subject Congruence). *$P \vdash \Delta$ and $P \equiv Q$ imply that $Q \vdash \Delta$.*

▶ **Theorem 9** (Subject Reduction). *$P \vdash \Delta$ and $P \to Q$ imply that $Q \vdash \Delta$.*

In Figure 8, global type annotations should not be mistaken for a requirement of our reductions; they are rather a guarantee given by our proof theory: if a process is reducible, then its sessions are surely typed with the respective global types reported in the rule. We use this property to reconstruct the result of session fidelity from multiparty session types [14]. In the following, $\mathsf{gt}(P)$ denotes the set of global types used in the restrictions inside P.

▶ **Theorem 10** (Session Fidelity). *$P \vdash \Delta$ and $P \to P'$ imply that either $\mathsf{gt}(P) \rightsquigarrow \mathsf{gt}(P')$ or $\mathsf{gt}(P) \simeq^{\mathsf{g}} \mathsf{gt}(P')$.*

Deadlock Freedom. Processes in MCP are guaranteed to be deadlock-free. We use the standard methodology from [4, 23]. First, we prove that the MCut rule in MCP is admissible:

▶ **Theorem 11** (MCut Admissibility). *$P_i \vdash \Gamma_i, x^{p_i} : A_i$, for $i \in [1, n]$, and $G \vDash \{p_i : A_i\}_i$ imply that there exists Q such that $Q \vdash \{\Gamma_i\}_i$.*

The admissibility of MCut gives us a methodology for removing cuts from a proof, corresponding to executing communications in a process until all restrictions are eliminated:

▶ **Corollary 12** (Deadlock-Freedom). *$P \vdash \Delta$ and P has a restriction imply $P \to Q$ for some Q.*

Protocol Progress. Our correspondence between process and global type reductions goes both ways, a novel result for Multiparty Session Types. Below, $\to^+$ denotes one or more applications of $\to$:

▶ **Theorem 13** (Protocol Progress). *$P \vdash \Delta$ and $\mathsf{gt}(P) \rightsquigarrow \tilde{G}$ imply $P \to^+ P'$ and $\tilde{G} = \mathsf{gt}(P')$.*

6 The 2-Buyer Protocol Example

We now formalise the 2-buyer protocol from § 2 and expand it further.

Processes and Types. Roles B1, B2 and S are implemented as the processes:

$$\overline{x}^{\mathsf{B1\,S}}(title);\ \mathsf{wait}\ title^{\mathsf{B1}};\ x^{\mathsf{B1\,S}}(quote);\Big(\mathsf{close}\ quote^{\mathsf{B1}}\ \mid\ \overline{x}^{\mathsf{B1\,B2}}(contrib);\mathsf{wait}\ contrib^{\mathsf{B1}};\mathsf{wait}\ x^{\mathsf{B1}};\mathsf{close}\ z^{\mathsf{Z}}\Big)$$
$$x^{\mathsf{B2\,S}}(quote);\ \Big(\mathsf{close}\ quote^{\mathsf{B2}}\ \mid\ x^{\mathsf{B2\,B1}}(contrib);\ \big(\mathsf{close}\ contrib^{\mathsf{B2}}\ \mid\ P_{B2}\big)\Big)$$
$$x^{\mathsf{S\,B1}}(title);\ \big(\mathsf{close}\ title^{\mathsf{S}}\ \mid\ \overline{x}^{\mathsf{S\,B1}}(quote);\ \mathsf{wait}\ quote^{\mathsf{S}};\ \overline{x}^{\mathsf{S\,B2}}(quote);\ \mathsf{wait}\ quote^{\mathsf{S}};\ \ P_S\big)\Big)$$

The first process is the first buyer Buyer1. In the second process, the second buyer Buyer2, subterm P_{B2} implements the choice of whether to accept or reject the purchase:

$$\left(x^{\mathsf{B2\,S}}.\mathsf{inl};\ \overline{x}^{\,\mathsf{B2\,S}}(addr);\ \mathsf{wait}\ addr^{\mathsf{S}};\ \mathsf{close}\ x^{\mathsf{B2}}\right)\ +\ \left(x^{\mathsf{B2\,S}}.\mathsf{inr};\mathsf{close}\ x^{\mathsf{B2}}\right)$$

Finally, in the third process, the implementation of the seller, P_S is the process:

$$x^{\mathsf{S\,B2}}.\mathsf{case}\ (\ x^{\mathsf{S\,B2}}(addr);\ \left(\mathsf{close}\ addr^{\mathsf{S}}\ \mid\ \mathsf{close}\ x^{\mathsf{S}}\right),\qquad \mathsf{close}\ x^{\mathsf{S}}\ )$$

At the level of types, the local types in Example 1 from § 3 can be used to type the three processes above: Buyer1 $\vdash x^{\mathsf{B1}}{:}A, z^{\mathsf{Z}}{:}1$, Buyer2 $\vdash x^{\mathsf{B2}}{:}B$ and Seller $\vdash x^{\mathsf{S}}{:}C$. If we apply our new cut rule, we obtain $(\boldsymbol{\nu} x : G)\left(\text{Buyer1}\ \mid\ \text{Buyer2}\ \mid\ \text{Seller}\right) \vdash z^{\mathsf{Z}}{:}1$ where the global type G, corresponding to equation (1) in § 2, is such that $G\ \vDash\ \mathsf{B1}{:}A,\ \mathsf{B2}{:}B,\ \mathsf{S}{:}C$.

Nested Multiparty Sessions. We can extend the example above by implementing the global type (4) in § 2, where the first buyer creates a sub-session with the seller if the second buyer decides not to contribute to the purchase. Below, we give an excerpt of the new seller:

$$\ldots\overline{x}^{\mathsf{S\,B1,B2}}(quote);\mathsf{wait}\ quote^{\mathsf{S}};x^{\mathsf{S\,B2}}.\mathsf{case}\left(\ldots,x^{\mathsf{S\,B1}}(y);\left(P_{\mathsf{sub}}\mid x^{\mathsf{S\,B1}}(why);\left(\mathsf{close}\ why^{\mathsf{S}}\mid x^{\mathsf{S\,B2}}(why);\ldots\right)\right)\right)$$

where $P_{\mathsf{sub}} = y^{\mathsf{S\,B1}}.\mathsf{case}\left(y^{\mathsf{S\,B1}}(addr);(\mathsf{close}\ addr^{\mathsf{S}}\ \mid\ \mathsf{close}\ y^{\mathsf{S}}),\quad y^{\mathsf{S\,B1}}(why);(\mathsf{close}\ why^{\mathsf{S}}\ \mid\ \mathsf{close}\ y^{\mathsf{S}})\right)$. Hence, the type of channel x, from the seller's viewpoint, becomes:

$$1\otimes^{\mathsf{B1}}\bot\ \invamp^{\mathsf{B1,B2}}\left((1\otimes^{\mathsf{B2}}1)\ \ \&^{\mathsf{B2}}\ \ \left(\left((1\otimes^{\mathsf{B1}}1)\ \ \&^{\mathsf{B1}}\ \ (1\otimes^{\mathsf{B1}}1)\right)\otimes^{\mathsf{B1}}1\otimes^{\mathsf{B1}}1\otimes^{\mathsf{B2}}1\right)\right)$$

We can then use coherence to infer the global type (4) in § 2.

Services. We extend the example to support multiple clients on a replicated session a:

$$(\boldsymbol{\nu} a{:}?\mathsf{B1}\ \texttt{->}\ !\mathsf{B2},\mathsf{S}:\langle G\rangle)\ (\ \text{Buyers}\ \mid\ !a^{\mathsf{B2}}(x);\text{Buyer2}\ \mid\ !a^{\mathsf{S}}(x);\text{Seller}\ )$$

where Buyers consists of two buyers: $(\boldsymbol{\nu} z:\mathsf{end})\ (\ ?a^{\mathsf{B1}}(x);\text{Buyer1}\ \mid\ ?a^{\mathsf{B1}}(x);\text{Buyer1}')$. Process Buyer1′ initially behaves as Buyer1, but we replace $\mathsf{close}\ z^{\mathsf{Z}}$ with $\mathsf{wait}\ z^{\mathsf{Z}};\mathsf{close}\ w^{\mathsf{W}}$. By applying $\beta_{!C}$ once, $\beta_{!?}$ twice, and commuting conversions, the process above can be reduced to the parallel composition of two sessions that follow the 2-buyer protocol:

$$(\boldsymbol{\nu} x{:}G)\left(\text{Buyer2}\ \mid\ \text{Seller}\ \mid\ (\boldsymbol{\nu} z:\mathsf{end})\left(\text{Buyer1}\ \mid\ (\boldsymbol{\nu} x{:}G)\,(\text{Buyer2}\ \mid\ \text{Seller}\ \mid\ \text{Buyer1'})\right)\right)$$

7 Related Work and Discussion

Curry-Howard correspondences for session types. The works closest to ours are the Curry-Howard correspondences between binary session types and linear logic [4, 23]. We extended this line of work considerably by introducing multiparty sessions, which required generalising the notion of type compatibility in linear logic to address multiple types (coherence). Coherence reconstructs the standard relationship between the global and local views found in multiparty session types. We then used coherence to develop a new proof theory that conservatively extends linear logic to capture multiparty interactions (all derivable judgements in linear logic are derivable also in our framework, and vice versa). Furthermore, our work provides, for the first time, a notion of session fidelity in the context of a Curry-Howard

correspondence between linear logic and session types (§ 5, Theorem 10). In this work we have not treated polymorphism and existential/universal quantification, which we believe can be naturally added to MCP following the lines presented in [23, 3] for binary sessions.

Our work inverts the interpretation of $\otimes$ as output and $\parr$ as input given in [2]. This makes our process terms in line with previous developments of multiparty session types, where communications go from one sender to many receivers [9]. Using the standard interpretation would yield a join mechanism where multiple senders synchronise with a single receiver; note that there would be no need to re-prove our results, since the proof theory would not change.

The standard cut rule in CLL forces the graph of connections among processes to be a tree [1], a known sufficient condition for deadlock-freedom in session types [5]. A multi-cut rule is proposed in [1] to allow two processes to share multiple channels. This enables reasoning on networks with cyclic inter-connections, but breaks the deadlock-freedom property guaranteed by linear logic, since duality is no longer a sufficient condition when multiple resources are involved (also noted in [23]). For the first time, MCP processes can have cyclic inter-connections (e.g., our example in § 2), but they are still guaranteed to be deadlock-free. The key twist is to use coherence as a principle to check that the inter-connections are safely resolved by communications. This suggests that coherence may be useful also in other settings related to linear logic, for reasoning about the sharing of resources among multiple entities (in our case, sessions). We leave this investigation as interesting future work.

Multiparty Session Types (MPSTs). Our work concisely unifies many of the ideas found in separate developments of multiparty session types. Our global types with multicasting are inspired from [9], to which we added nested and replicated types; both additions arise naturally from our proof theory. Our nesting of global types can be seen as a logical reconstruction of (a simplification of) those originally presented in [10], while repetitions in global types reconstruct the concept presented in [8].

Our proof system for coherence is inspired by the notion of well-formedness found in MPSTs [14, 9]. Since coherence is a proof system, projection and extraction are derived from proof equivalences, rather than being defined separately as in [14, 15]. A benefit is that our projection and extraction are guaranteed to be correct by construction, whereas in previous works they have to be proven correct separately wrt the auxiliary notion of well-formedness.

In [9], MPSTs are combined with an ordering on session names to guarantee deadlock-freedom. Our deadlock-freedom result, instead, is based on the structure of our proofs. In some cases, our technique is more precise; for example, consider the deadlock-free system:

$$?a^{p}(x_a);\ ?b^{r}(x_b);\ \overline{x_a}^{\,pq}(w_1);\ {x_b}^{rs}(w_2);\ (\ \overline{x_a}^{\,pt}(w_3);\ P_1\ \mid\ P_2\ )$$
$$!a^{q}(x_a);\ {x_a}^{qp}(w_1);\ (P_3\ \mid\ P_4) \qquad !a^{t}(x_a);\ {x_a}^{tp}(w_3);\ (P_5\ \mid\ P_6) \qquad !b^{s}(x_b);\ \overline{x_b}^{\,sr}(w_2);\ P_7$$

If we compose these processes in parallel, restricting sessions a and b accordingly, we obtain a typable MCP process. Instead, the system in [9] rejects it, since the actions performed by the first process create a cycle between the names x_a and x_b. In [19], the approach in [9] is refined to type processes such as the one above by ordering the I/O actions of each session.

We conjecture that MCP can be used to naturally extend the work in [7], where linear logic is used to type choreography programs, obtaining a Curry-Howard correspondence for the calculus of compositional choreographies typed with multiparty session types [18].

Coherence. Coherence can be generalised, e.g., in Figure 2: (i) rule !? could allow for more than one client; (ii) similarly, rule $1\bot$ could be relaxed to allow for more than one $\bot$ type; (iii) rule $\otimes\parr$ could allow the involved participants to play different roles in the nested session they

create, as in [10] (adding such roles as an extra annotation to each type respectively). We leave these extensions as interesting future work. Point (ii) influences greatly the complexity of the cut admissibility proof for MCP (Theorem 11), because it would imply that the cut reduction of a terminated session could lead to having more than one process in the reductum (all the processes typed with $\bot$), whereas now we have only one. This means that we would have to type a parallel composition of processes without restriction, requiring to extend our framework in the fashion of the logic presented in [7]. While extending the proof theory of MCP would be easy, (extending coherence to allow for missing participants to be added later, as in [18]), it would also cause an explosion in the number of cases to consider in the proof [7]. As future work, we will investigate how our rule MCut and the notion of coherence can affect the mapping from the functional language GV [23, 17].

Acknowledgements. Montesi was supported by CRC, grant no. DFF–4005-00304 from the Danish Council for Independent Research. Schürmann was partly supported by DemTech, grant no. 10-092309 from the Danish Council for Strategic Research. Yoshida was partially sponsored by the EPSRC EP/K011715/1, EP/K034413/1, EP/L00058X/1, and EU project FP7-612985 UpScale. This work is also supported by the COST Action IC1201 BETTY.

References

1 Samson Abramsky, Simon J. Gay, and Rajagopal Nagarajan. Interaction categories and the foundations of typed concurrent programming. In *NATO ASI DPD*, pages 35–113, 1996.

2 Gianluigi Bellin and Philip J. Scott. On the pi-calculus and linear logic. *Theor. Comput. Sci.*, 135(1):11–65, 1994.

3 Luís Caires, Jorge A. Pérez, Frank Pfenning, and Bernardo Toninho. Behavioral polymorphism and parametricity in session-based communication. In *ESOP*, pages 330–349, 2013.

4 Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In *CONCUR*, pages 222–236, 2010.

5 Marco Carbone and Søren Debois. A graphical approach to progress for structured communication in web services. In *Proc. of ICE'10*, 2010.

6 Marco Carbone and Fabrizio Montesi. Deadlock-freedom-by-design: multiparty asynchronous global programming. In *POPL*, pages 263–274, 2013.

7 Marco Carbone, Fabrizio Montesi, and Carsten Schürmann. Choreographies, logically. In *CONCUR*, pages 47–62, 2014.

8 Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, and Luca Padovani. On global types and multi-party session. *LMCS*, 8(1), 2012.

9 Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida, and Luca Padovani. Global progress for dynamically interleaved multiparty sessions. *MSCS*, 760:1–65, 2015.

10 Romain Demangeon and Kohei Honda. Nested protocols in session types. In *CONCUR*, pages 272–286, 2012.

11 Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP (2)*, pages 174–186, 2013.

12 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.

13 Kohei Honda, Vasco Vasconcelos, and Makoto Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP*, pages 22–138, 1998.

14 Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In *Proc. of POPL*, volume 43(1), pages 273–284. ACM, 2008.

15 Julien Lange and Emilio Tuosto. Synthesising choreographies from local session types. In *CONCUR*, pages 225–239, 2012.

16 Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From communicating machines to graphical choreographies. In *POPL 2015*, pages 221–232. ACM, 2015.

17 Sam Lindley and J. Garrett Morris. A semantics for propositions as sessions. In *ESOP*, pages 560–584, 2015.

18 Fabrizio Montesi and Nobuko Yoshida. Compositional choreographies. In *CONCUR*, pages 425–439, 2013.

19 Luca Padovani, Vasco Thudichum Vasconcelos, and Hugo Torres Vieira. Typing liveness in multiparty communicating systems. In *COORDINATION*, pages 147–162, 2014.

20 D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

21 Scribble project home page. `http://www.scribble.org`.

22 Vasco T. Vasconcelos. Fundamentals of session types. *Inf. Comput.*, 217:52–70, 2012.

23 Philip Wadler. Propositions as sessions. In *ICFP*, pages 273–286, 2012.