# Coherence Generalises Duality: A Logical Explanation of Multiparty Session Types*

**Marco Carbone[1], Sam Lindley[2], Fabrizio Montesi[3], Carsten Schürmann[4], and Philip Wadler[5]**

1 **IT University of Copenhagen**
2 **University of Edinburgh**
3 **University of Southern Denmark**
4 **IT University of Copenhagen**
5 **University of Edinburgh**

**Abstract**

Wadler introduced Classical Processes (CP), a calculus based on a propositions-as-types correspondence between propositions of classical linear logic and session types. Carbone *et al.* introduced Multiparty Classical Processes, a calculus that generalises CP to multiparty session types, by replacing the duality of classical linear logic (relating two types) with a more general notion of coherence (relating an arbitrary number of types). This paper introduces variants of CP and MCP, plus a new intermediate calculus of Globally-governed Classical Processes (GCP). We show a tight relation between these three calculi, giving semantics-preserving translations from GCP to CP and from MCP to GCP. The translation from GCP to CP interprets a coherence proof as an arbiter process that mediates communications in a session, while MCP adds annotations that permit processes to communicate directly without centralised control.

**1998 ACM Subject Classification** F1.2 Modes of Computation: Parallelism and concurrency; F4.1 Mathematical logic: Proof theory

**Keywords and phrases** Multiparty Session Types, Linear Logic, Propositions as Types

**Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2016.33

## 1 Introduction

Session types, introduced by Honda, Vasconcelos, and Kubo [11, 20], are protocols that describe valid communication patterns in process calculi. A correspondence between process calculi and classical linear logic was found by Abramsky [1] and Bellin and Scott [3], and another between session types and intuitionistic linear logic by Caires and Pfenning [6, 7], in both of which channel types correspond to propositions of linear logic, processes to proofs, and communication to proof normalisation. Based on these, Wadler [22] introduced Classical Processes (CP), in which session types correspond to propositions of classical linear logic, processes to proofs, and communication to cut elimination. Key properties of session types such as deadlock freedom follow from key properties of linear logic such as cut elimination.

* Montesi was supported by the CRC project, 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. Lindley and Wadler were supported by the EPSRC grant ABCD (EP/K034413/1). This work was also supported by the EU COST Action IC1201 BETTY.





Last year, Carbone *et al.* [9] introduced Multiparty Classical Processes (MCP), which extends CP to the multiparty session types introduced by Honda, Yoshida, and Carbone [12]. In CP duality is defined between two propositions, whereas in MCP duality is replaced by coherence among multiple propositions. Coherence relates a global type (a description of a multiparty protocol) to many local types (the behaviours of each participant in a session).

MCP came at a cost as compared to CP. First, MCP required annotating logical connectives with roles, and it was unclear how such annotations related to classical linear logic. Second, MCP omitted the axiom and atomic and quantified propositions, losing support for parametric polymorphism. Third, MCP inverted the usual interpretation of connectives $\otimes$ and $\parr$, treating output as input and vice versa, which we no longer believe is a tenable position. This work presents an updated version of MCP that overcomes these shortcomings.

Section 2 introduces our variant of CP. We modify restriction to replace a single channel by two endpoints, yielding a logical reconstruction of the covariable formulation of session types due to Vasconcelos [20]. We partition types into input and output types (a slight variation of positive and negative polarities [13, 17]), which allows us to orient the axiom rule. And, in order to align with the subsequent development, we restrict the cut of axiom against a process to atomic variables, applying $\eta$-expansion to handle the remaining cases.

Section 3 introduces a calculus of Globally-governed Classical Processes (GCP), intermediate between CP and MCP. GCP uses global types to describe multiparty sessions, and coherence to relate the global type to many local types. GCP differs from MCP in not requiring annotations; types in GCP are the standard propositions of CP. We present a semantics-preserving translation from GCP into CP, where a global type is translated into an *arbiter*, an auxiliary process that coordinates communication. Caires and Perez [4] introduce a *medium* which is similar to our arbiter; we compare our work with theirs in Section 5. We show that under our translation GCP is simulated by CP. We also give a translation from CP to GCP and show that it too is a simulation.

Section 4 introduces our variant of MCP. MCP augments GCP with annotations which permit processes to communicate directly without centralised control. Whereas the original MCP refers to sessions and roles, our variant uses a simpler formulation based exclusively on endpoints. Our variant also restores the proper correspondence of $\otimes$ with output and $\parr$ with input, and supports parametric polymorphism. We present a semantics-preserving translation of MCP into GCP, which simply consists of erasing annotations. We show under our translation there is a bisimulation between MCP and GCP.

Section 5 discusses related and future work.

We illustrate our encodings using the classic 2-buyer protocol [12] as a running example. Two buyers, $B_1$ and $B_2$, attempt to buy a book together from a seller $S$. First, $B_1$ sends the title of the book that she wishes to purchase to $S$, who in turn sends a quote to both $B_1$ and $B_2$. Then, $B_1$ decides on how much she wishes to contribute, and informs $B_2$, who either pays the rest or cancels the transaction. A similar example appears in the original paper on MCP [9], but is degenerate in that it represents data using the unit type. There, it is not possible to replace the unit type with a non-degenerate type because of the inversion of $\otimes$ and $\parr$. Here, we present non-degenerate encodings in CP, GCP, and MCP.

## 2 Classical Processes (CP)

We describe our version of Classical Processes (CP), originally introduced by Wadler [22].

**Types.** We start by introducing propositions, which we interpret as session types. Let $A, B, C, D$ range over propositions and $X, Y$ range over atomic propositions.

| | | | | | |
|---|---|---|---|---|---|
| $A, B, C, D$ | ::= | $A \otimes B$ | *(send A, proceed as B)* | $A \parr B$ | *(receive A, proceed as B)* |
| | | $A \oplus B$ | *(select A or B)* | $A \& B$ | *(offer A or B)* |
| | | $0$ | *(unit for $\oplus$)* | $\top$ | *(unit for $\&$)* |
| | | $1$ | *(unit for $\otimes$)* | $\bot$ | *(unit for $\parr$)* |
| | | $?A$ | *(client request)* | $!A$ | *(server accept)* |
| | | $X$ | *(atomic propositions)* | $X^\bot$ | *(dual of atomic proposition)* |
| | | $\exists X.A$ | *(existential)* | $\forall X.A$ | *(universal)* |

We give a behavioural explanation to the types above. Proposition $A \otimes B$ is the type of a channel over which we send a fresh channel of type $A$ and then continue as $B$. Dually, $A \parr B$ is the type of a channel over which we receive a channel of type $A$ and then continue as $B$. Proposition $A \oplus B$ is the type of a channel over which we can select to proceed either with type $A$ or with type $B$. Dually, $A \& B$ is the type of a channel offering a choice of proceeding with either type $A$ or type $B$. Propositions $0$, $\top$, $1$ and $\bot$ are units for $\oplus$, $\&$, $\otimes$ and $\parr$, respectively. Proposition $?A$ is the type of a channel over which a client may request multiple invocations of a server. Dually, $!A$ is the type of a channel over which a server may accept multiple invocations from a client. Atomic propositions $X$ and $X^\bot$, and universal propositions $\exists X.A$ and $\forall X.A$ model polymorphic channels [22].

Each type in the left-hand column is dual to the corresponding type in the right-hand column. We write $A^\bot$ for the dual of $A$. We refer to types on the left as *output* types, and types on the right as *input* types. Our output and input types correspond, respectively, to the standard notions in logic of positive and negative types [13]. The one exception are the exponentials: we classify ? as an output type and ! as an input type, while logicians classify ? as negative and ! as positive. Exponentials are already known to have a less strong correspondence with positive and negative types than other connectives. For instance, negative types have invertible rules while positive types do not, the exception being exponentials, where the rule for ! is invertible while the rules for weakening and contraction are not.

**Processes.** In CP, proofs correspond to proof terms, expressed in a $\pi$-calculus with sessions. Let $x$, $y$, and $z$ range over channel endpoints. The syntax of processes is given by the proof terms (denoted in red) shown in Figure 1. In an output operation the sent object is always contained in square brackets $[\ldots]$, and, dually, in an input operation the received variable is always bound in round parentheses $(\ldots)$. As in the internal $\pi$-calculus [18], the object $y$ in a send $x[y].(P \mid Q)$ and in a client request $?x[y].P$ is bound (this is not the case in selection and send type). A link process $x \to y^B$ forwards communications from $x$ to $y$. A restriction $(\boldsymbol{\nu} x^A y)\,(P \mid Q)$ pairs two endpoints $x$ and $y$ into a session (as done by Vasconcelos [20]).

We give the type rules for CP in Figure 1. All rules are standard CLL rules. In the logic, link corresponds to axiom and restriction to cut. Their interpretation as proof terms follows that of Wadler [22], with rules Axiom and Cut adopting the new syntax.

**Semantics.** CLL is equipped with proof transformations that give semantics to CP processes. Figure 2 displays structural equivalence rules, $\eta$-expansions and $\beta$-reductions for processes and cuts. Structural equivalences permit swapping the names in a link and in a restriction, and reassociating two restrictions. The $\eta$-expansions for link do not appear in the original presentation of CP [22], but are standard and appear elsewhere [16]. We reformulate CP to use them, as they prove helpful in defining GCP in the next section. The expansion replaces

$$\frac{}{x \to y^{A} \vdash x : A^{\perp}, y : A}\ \textsc{Axiom} \qquad \frac{P \vdash \Gamma, x : A \quad Q \vdash \Delta, y : A^{\perp}}{(\nu x^{A} y)\,(P \mid Q) \vdash \Gamma, \Delta}\ \textsc{Cut}$$

$$\frac{P \vdash \Gamma, y : A \quad Q \vdash \Delta, x : B}{x[y].(P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B}\ \otimes \qquad \frac{P \vdash \Gamma, y : A, x : B}{x(y).P \vdash \Gamma, x : A \parr B}\ \parr$$

$$\frac{P \vdash \Gamma, x : A}{x[\mathsf{inl}].P \vdash \Gamma, x : A \oplus B}\ \oplus_1 \qquad \frac{P \vdash \Gamma, x : B}{x[\mathsf{inr}].P \vdash \Gamma, x : A \oplus B}\ \oplus_2 \qquad \frac{P \vdash \Gamma, x : A \quad Q \vdash \Gamma, x : B}{x.\mathsf{case}(P, Q) \vdash \Gamma, x : A \,\&\, B}\ \&$$

$$\frac{P \vdash \Gamma, y : A}{?x[y].P \vdash \Gamma, x : ?A}\ ? \qquad \frac{P \vdash\ ?\Gamma, y : A}{!x(y).P \vdash\ ?\Gamma, x : !A}\ ! \qquad \frac{P \vdash \Gamma}{P \vdash \Gamma, x : ?A}\ \textsc{Weaken}$$

$$\frac{P \vdash \Gamma, x : B[A/X]}{x[A].P \vdash \Gamma, x : \exists X.B}\ \exists \qquad \frac{P \vdash \Gamma, x : B \quad X \notin \mathsf{ftv}(\Gamma)}{x(X).P \vdash \Gamma, x : \forall X.B}\ \forall \qquad \frac{P \vdash \Gamma, y : ?A, z : ?A}{P\{x/y, x/z\} \vdash \Gamma, x : ?A}\ \textsc{Contract}$$

$$\frac{}{x[] \vdash x : 1}\ 1 \qquad \frac{P \vdash \Gamma}{x().P \vdash \Gamma, x : \bot}\ \bot \qquad \text{(no rule for 0)} \qquad \frac{}{x.\mathsf{case}() \vdash \Gamma, x : \top}\ \top$$

**Figure 1** CP, Type Rules.

a link, step-by-step, by processes that perform the communications required by its type. The $\beta$-reductions of CP correspond to the cut elimination rules in CLL, and are standard.

We omit commuting conversions, which lift prefixes out of cuts on different endpoints; they are as in [22]. Structural equivalence and reductions apply inside a cut, but not inside a prefix. We use juxtaposition for the composition of relations, write $R^{+}$ for the transitive closure and $R^{*}$ for the transitive reflexive closure of relation $R$. We write $\Longrightarrow$ for $\equiv \longrightarrow \equiv$.

▶ **Theorem 1** (Subject reduction for CP). *If $P \vdash \Gamma$ and $P \Longrightarrow Q$, then $Q \vdash \Gamma$.*

▶ **Theorem 2** (Cut elimination for CP). *If $P \vdash \Gamma$ then there exists a $Q$ such that $P \Longrightarrow^{*} Q$ and $Q$ is not a cut.*

The proof of cut elimination for CP is standard [22]. All other theorems in this paper follow by straightforward induction.

▶ **Example 3.** We now describe the 2-buyer protocol in CP.

First, we proceed by providing a set of suitable types for the endpoints along which $B_1$, $B_2$, and $S$ communicate. We assume that $\mathbf{cost}, \mathbf{name}, \mathbf{addr}$ are atomic propositions. In CP, $B_1$'s endpoint has type $\mathbf{name} \otimes \mathbf{cost}^{\perp} \parr \mathbf{cost} \otimes 1$, $B_2$'s endpoint has type $\mathbf{cost}^{\perp} \parr \mathbf{cost}^{\perp} \parr \big((\mathbf{addr} \otimes 1) \oplus 1\big)$, and $S$'s endpoint has type $\mathbf{name}^{\perp} \parr \mathbf{cost} \otimes \mathbf{cost} \otimes \big((\mathbf{addr}^{\perp} \parr \bot) \& \bot\big)$. In terms of traditional multiparty sessions, these types amount to local types, but with the roles erased.

Second, we define an *arbiter* process over three endpoints, $b_1, b_2, s$, one each for communication with $B_1$, $B_2$, and S, respectively.

$$\begin{aligned} &b_1(b_1').s[s'].(b_1' \to s'^{\,\mathbf{name}} \mid s(s').b_1[b_1'].(s' \to b_1'^{\,\mathbf{cost}} \mid \\ &\quad s(s').b_2[b_2'].(s' \to b_2'^{\,\mathbf{cost}} \mid b_1(b_1').b_2[b_2'].(b_1' \to b_2'^{\,\mathbf{cost}} \mid \\ &\qquad b_2.\mathsf{case}(s[\mathsf{inl}].b_2(b_2').s[s'].(b_2' \to s'^{\,\mathbf{addr}} \mid b_1().b_2().s[]), s[\mathsf{inr}].b_1().b_2().s[])))) \end{aligned}$$

The arbiter process mediates between the parties, incorporating the missing role information that would normally appear in local types. In terms of traditional multiparty sessions, the arbiter corresponds to the dual of a global type.

Finally, an instantiation of the protocol must cut implementations of $B_1$, $B_2$, and $S$ against the arbiter process.

Structural equivalence (Processes)

$$y \to x^{A^\perp} \equiv x \to y^A \qquad (\boldsymbol{\nu} y^{A^\perp} x)\,(Q \mid P) \equiv (\boldsymbol{\nu} x^A y)\,(P \mid Q)$$
$$(\boldsymbol{\nu} w^B z)\,(P \mid (\boldsymbol{\nu} x^A y)\,(Q \mid R)) \equiv (\boldsymbol{\nu} x^A y)\,((\boldsymbol{\nu} w^B z)\,(P \mid Q) \mid R)$$

$\eta$-expansions (Processes)

$$\begin{array}{rcl@{\qquad}rcl} x \to y^{A\otimes B} & \longrightarrow & x(u).y[v].(u \to v^A \mid x \to y^B) & x \to y^1 & \longrightarrow & x().y[] \\ x \to y^{A\oplus B} & \longrightarrow & x.\mathsf{case}(y[\mathsf{inl}].x \to y^A, y[\mathsf{inr}].x \to y^B) & x \to y^0 & \longrightarrow & x.\mathsf{case}() \\ x \to y^{?A} & \longrightarrow & !x(u).?y[v].u \to v^A & x \to y^{\exists X.A} & \longrightarrow & x(X).y[X].x \to y^A \end{array}$$

$\beta$-reductions (Processes)

$$\begin{array}{rcl} (\boldsymbol{\nu} x^X y)\,(w \to x^X \mid Q) & \longrightarrow & Q\{w/y\} \\ (\boldsymbol{\nu} x^{A\otimes B} y)\,(x[u].(P \mid Q) \mid y(v).R) & \longrightarrow & (\boldsymbol{\nu} u^A v)\,(P \mid (\boldsymbol{\nu} x^B y)\,(Q \mid R)) \\ (\boldsymbol{\nu} x^1 y)\,(x[] \mid y().P) & \longrightarrow & P \\ (\boldsymbol{\nu} x^{A\oplus B} y)\,(x[\mathsf{inl}].P \mid x.\mathsf{case}(Q,R)) & \longrightarrow & (\boldsymbol{\nu} x^A y)\,(P \mid Q) \\ (\boldsymbol{\nu} x^{A\oplus B} y)\,(x[\mathsf{inr}].P \mid x.\mathsf{case}(Q,R)) & \longrightarrow & (\boldsymbol{\nu} x^B y)\,(P \mid R) \\ \multicolumn{3}{c}{\text{(no rule for 0 with } \top\text{)}} \\ (\boldsymbol{\nu} x^{?A} y)\,(?x[u].Q \mid !y(v).P) & \longrightarrow & (\boldsymbol{\nu} u^A v)\,(P \mid Q) \\ (\boldsymbol{\nu} x^{?A} y)\,(P \mid !y(v).Q) & \longrightarrow & P \\ (\boldsymbol{\nu} x^{?A} y)\,(P\{x/x', x/x''\} \mid !y(v).Q) & \longrightarrow & (\boldsymbol{\nu} x'^{?A} y')\,(((\boldsymbol{\nu} x''^{?A} y'')\,(P \mid !y'(v).Q)) \mid !y''(v).Q) \\ (\boldsymbol{\nu} x^{\exists X.B} y)\,(x[A].P \mid y(X).Q) & \longrightarrow & (\boldsymbol{\nu} x^{B\{A/X\}} y)\,(P \mid Q\{A/X\}) \end{array}$$

**Figure 2** CP, Structural Equivalence and Reduction Rules.

**Directing link and restriction.** We make a useful observation here that does not appear in [22]. A link or a restriction will always be between an input type and its dual output type. The swap rule $y \to x^{A^\perp} \equiv x \to y^A$ may always be applied to orient a link so that the input type is on the left and the output type on the right, and in this case the flow of information in the link will always be from left to right. Similarly, the swap rule may always be applied to orient a restriction so that the output type is on the left and the input type on the right, and in this case the flow of information in the restriction will always be from left to right. These descriptions are pleasingly similar, but note that input and output swap positions in the description of link and restriction!

**May one invert output and input?** The original presentation of MCP [9] states: "Our work inverts the interpretation of $\otimes$ as output and $\parr$ as input given in [3]. This makes our process terms in line with previous developments of multiparty session types, where communications go from one sender to many receivers [10]." Implicit is the claim that whether one assigns $\otimes$ to output and $\parr$ to input is a convention that may be inverted without harm. Here we argue that such a view is not tenable. As an illustration, consider the derivation in Figure 3; it types a process that inputs a single bit (or, more precisely, offers a choice between two units) and outputs two copies of that bit (or, more precisely, twice makes a selection between two units, both times echoing the choice made on input).

Would it make sense to modify our interpretation of the process terms (in red), so that inputs are considered as outputs and vice versa? Absolutely not! The interpretation of $\&$ as offering a choice and $\oplus$ as making a selection is uncontroversial (it is also accepted in [9]). Once the interpretations of $\&$ and $\oplus$ are fixed then in this example the only way to view $\parr$ is as input and $\otimes$ is as output. Nor is it possible to assign a sensible view if we swap the interpretations of $\&$ and $\oplus$, since then the process would input two bits which must be

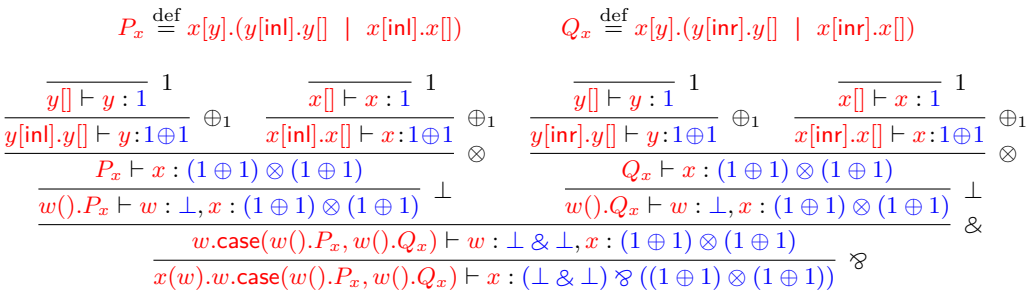


**Figure 3** Example: duplicating a bit.

identical and output one bit which is equal to both inputs; this would eliminate the idea that each channel can have its value chosen independently, and we cannot see how to make sense of the process calculus that would result. More generally, it makes perfect sense when outputting one channel along another channel to assign the behaviour of the output channel and of remaining behaviour of the original channel to two separate processes (as happens when $\otimes$ is interpreted by output), and when inputting one channel along another channel to assign the behaviour of both channels to a single process (as happens when $\parr$ is interpreted by input). But it makes no sense to take the inverse interpretation, and when inputting one channel along another channel assign the behaviour of the input channel and the behaviour of the remainder of the original channel to two separate processes—then the behaviour of the input could have no effect on the behaviour of the remainder of the original channel, which contradicts the notion of how input is intended to behave. (We are grateful to Bob Atkey for this general argument about input and output.)

## 3 Globally-governed Classical Processes (GCP)

In CP, communications between two parties take place over a binary cut. In this section, we introduce GCP, by replacing the *binary* cut in CP with a *multiparty* cut (called coherence cut), where communications among multiple parties are governed by a *global type*.

**Types.** Coherence, introduced in [9], generalises the notion of duality found in classical linear logic. Two propositions $A$ and $B$ are dual if each output type in $A$ is matched by an input type in $B$ and vice versa. Duality ensures that two processes can be composed safely, by connecting two respective endpoints that have compatible interfaces (dual types). In GCP, we wish to also compose more than two processes, and therefore, we need a notion that generalises duality to compatibility among $n$ processes. The notion of coherence serves this purpose and it is given as a proof system whose judgements have the form $G \vDash \Gamma$ where $\Gamma$ is a set of compatible types and $G$, called the *global type*, is the corresponding proof term.

Global types give the flow of communications that sessions must follow. Their syntax is given by the proof terms (in red) in Figure 4. Let $G, H$ range over global types, and write $\tilde{x}$ to abbreviate the sequence $(x_i)_i$. In $\tilde{x} \to y(G).H$, endpoints $\tilde{x}$ each send a message to $y$ to create a new session of type $G$ and then continue as $H$. In $\tilde{x} \to y$, endpoints $\tilde{x}$ each send a message to $y$ to terminate the session. In $x \to \tilde{y}.\mathsf{case}(G, H)$, endpoint $x$ sends a choice to endpoints $\tilde{y}$ on whether to proceed as $G$ or $H$. In $x \to \tilde{y}.\mathsf{case}()$, $x$ sends an empty choice to $\tilde{y}$. In $!x \to \tilde{y}(G)$, client $x$ sends a request to servers $\tilde{y}$ to create a session of global type $G$. In

$$\frac{G \vDash (x_i : A_i)_i, y : C \quad H \vDash \Gamma, (x_i : B_i)_i, y : D}{\tilde{x} \to y(G).H \vDash \Gamma, (x_i : A_i \otimes B_i)_i, y : C \parr D}\ \otimes\parr \qquad \frac{}{\tilde{x} \to y \vDash (x_i : 1)_i, y : \bot}\ 1\bot$$

$$\frac{G \vDash \Gamma, x : A, (y_i : C_i)_i \quad H \vDash \Gamma, x : B, (y_i : D_i)_i}{x \to \tilde{y}.\mathsf{case}(G, H) \vDash \Gamma, x : A \oplus B, (y_i : C_i \,\&\, D_i)_i}\ \oplus\& \qquad \frac{}{x \to \tilde{y}.\mathsf{case}() \vDash \Gamma, x : 0, (y_i : \top)_i}\ 0\top$$

$$\frac{G \vDash x : A, (y_i : B_i)_i}{!x \to \tilde{y}(G) \vDash x : ?A, (y_i : !B_i)_i}\ ?! \qquad \frac{G \vDash \Gamma, x : A, (y_i : B_i)_i \quad X \notin \mathsf{ftv}(\Gamma)}{x \to \tilde{y}.(X)G \vDash \Gamma, x : \exists X.A, (y_i : \forall X.B_i)_i}\ \exists\forall$$

$$\frac{}{x^A \to y \vDash x : A, y : A^\perp}\ \textsc{Axiom} \qquad \frac{(P_i \vdash \Gamma_i, x_i : A_i)_i \quad G \vDash (x_i : A_i)_i}{(\boldsymbol{\nu} \tilde{x}^{\tilde{A}} : G)\,(\tilde{P}) \vdash \tilde{\Gamma}}\ \textsc{CCut}$$

**Figure 4** GCP, Coherence Rules and Coherence Cut.

$x \to \tilde{y}.(X)G$, endpoint $x$ sends a type to endpoints $\tilde{y}$ and the protocol proceeds as $G$. In $x^A \to y$, $x$ is connected to $y$. Thus, a coherence cut in which the global type is an axiom behaves exactly like a binary cut.

Types for endpoints in GCP are identical to those of CP. Then, coherence, denoted by $\vDash$, is defined by the rules given in Figure 4. Rule $\otimes\parr$ says that if we have some endpoints of type $A_i \otimes B_i$ and an endpoint of type $C \parr D$ then a communication can happen (denoted as $\tilde{x} \to y$ in the global type) which will create a new session with endpoints of type $(A_i)_i$ and $C$, and the old session will continue as $\Gamma, (B_i)_i, D$. All other rules are similar.

The rules and the proof terms for GCP are identical to those of CP save that the standard binary cut Cut is replaced by the coherence cut CCut, given in Figure 4, with the proof term $(\boldsymbol{\nu} \tilde{x}^{\tilde{A}} : G)\,(\tilde{P})$. In CCut, the use of coherence becomes clear: the global type $G$ governs all communication between the processes $\tilde{P}$. Each endpoint $x_i$ in each $P_i$ is bound with type $A_i$, and the coherence relation $G \vDash (x_i : A_i)_i$ ensures that such processes can safely communicate on such endpoints. The types $\tilde{A}$ adorning the endpoints $\tilde{x}$ are superfluous, since they are determined by $G$, but will come in handy when we write the translation from GCP to CP. It will follow from the translation, presented below, that if $G \vDash (x_i : A_i)_i$ holds then $\vdash (A_i^\perp)_i$ is derivable in classical linear logic. As in the original formulation of MCP [9], the restriction of coherence to two parties is exactly duality: $G \vDash x : A, y : B$ if and only if $A = B^\perp$. But, as we shall see at the end of this section, the connection between coherence and duality goes deeper than this.

**Semantics.** Figure 5 displays the interesting structural equivalences, $\eta$-expansions, and $\beta$-reductions of GCP. In addition we retain the structural equivalence for axiom and $\eta$-expansions of processes from Figure 2. We omit the straightforward commuting conversions, each one allowing a prefix to be lifted out of a coherence cut; see [9]. The $\beta$-rules are similar to CP, but engage multiple parties and all communication is coordinated by global types. Structural equivalence and reduction applies inside a coherence cut (including inside a global type), but not inside a prefix. Note that $\eta$-expansion on processes is necessary for cut-elimination to hold. For example, the process $(\boldsymbol{\nu} x^{A \otimes B} y^{A^\perp \parr C} z^D : x \to y(G).H)\,(w \to x^{A \otimes B} \mid y(y').P \mid Q)$ can only reduce if we first expand $w \to x^{A \otimes B}$ to $w(w').x[x'].(w' \to x'^A \mid w \to x^B)$.

▶ **Theorem 4** (Subject reduction for GCP). *If $P \vdash \Gamma$ and $P \Longrightarrow Q$, then $Q \vdash \Gamma$.*

▶ **Theorem 5** (Cut elimination for GCP). *If $P \vdash \Gamma$, then there exists $Q$ such that $P \Longrightarrow^* Q$ and $Q$ is not a cut.*

Structural equivalence (Global types and processes)

$$
\begin{aligned}
y^{A^\perp} \to x &\equiv x^A \to y \\
(\boldsymbol{\nu}\tilde{w}, y, x, \tilde{z} : G)\,(\tilde{P} \mid R \mid Q \mid \tilde{S}) &\equiv (\boldsymbol{\nu}\tilde{w}, x, y, \tilde{z} : G)\,(\tilde{P} \mid Q \mid R \mid \tilde{S}) \\
(\boldsymbol{\nu}z, \tilde{w} : H)\,((\boldsymbol{\nu}x, \tilde{y} : G)\,(P \mid \tilde{R}) \mid \tilde{Q}) &\equiv (\boldsymbol{\nu}x, \tilde{y} : G)\,((\boldsymbol{\nu}z, \tilde{w} : H)\,(P \mid \tilde{Q}) \mid \tilde{R})
\end{aligned}
$$

$\eta$-expansions (Global types)

$$
\begin{aligned}
x^{A\otimes B} \to y &\longrightarrow x \to y(x^A \to y).x^B \to y \\
x^{A\oplus B} \to y &\longrightarrow x \to y.\mathsf{case}(x^A \to y, x^B \to y) \\
x^{?A} \to y &\longrightarrow !x \to y(x^A \to y)
\end{aligned}
\qquad
\begin{aligned}
x^1 \to y &\longrightarrow x \to y \\
x^0 \to y &\longrightarrow x \to y.\mathsf{case}() \\
x^{\exists X.A} \to y &\longrightarrow x \to y.(X)x^A \to y
\end{aligned}
$$

$\beta$-reductions (Global types and processes)

$$
\begin{aligned}
(\boldsymbol{\nu}\tilde{x}, y, \tilde{z} : \tilde{x} \to y(G).H)\,((x_i[x_i'].(P_i \mid Q_i))_i \mid y(y').R \mid \tilde{S}) &\longrightarrow \\
(\boldsymbol{\nu}\tilde{x}', y' : G\{\tilde{x}'/\tilde{x}, y'/y\})\,(\tilde{P} \mid (\boldsymbol{\nu}\tilde{x}, y, \tilde{z} : H)\,(\tilde{Q} \mid R \mid \tilde{S})) \\
(\boldsymbol{\nu}\tilde{x}, y : \tilde{x} \to y)\,((x_i[])_i \mid y().P) &\longrightarrow P \\
(\boldsymbol{\nu}x, \tilde{y}, \tilde{z} : x \to \tilde{y}.\mathsf{case}(G, H))\,(x[\mathsf{inl}].P \mid (y_i.\mathsf{case}(Q_i, R_i))_i \mid \tilde{S}) &\longrightarrow (\boldsymbol{\nu}x, \tilde{y}, \tilde{z} : G)\,(P \mid \tilde{Q} \mid \tilde{S}) \\
(\boldsymbol{\nu}x, \tilde{y}, \tilde{z} : x \to \tilde{y}.\mathsf{case}(G, H))\,(x[\mathsf{inr}].P \mid (y_i.\mathsf{case}(Q_i, R_i))_i \mid \tilde{S}) &\longrightarrow (\boldsymbol{\nu}x, \tilde{y}, \tilde{z} : H)\,(P \mid \tilde{R} \mid \tilde{S}) \\
(\boldsymbol{\nu}x, \tilde{y} : !x \to \tilde{y}(G))\,(?x[x'].P \mid (!y_i(y_i').Q)_i) &\longrightarrow (\boldsymbol{\nu}x', \tilde{y}' : G\{x'/x, \tilde{y}'/\tilde{y}\})\,(P \mid \tilde{Q}) \\
(\boldsymbol{\nu}x, \tilde{y} : !x \to \tilde{y}(G))\,(P \mid (!y_i(y_i').Q)_i) &\longrightarrow P, \quad \text{if } x \notin \mathsf{fv}(P) \\
(\boldsymbol{\nu}x, \tilde{y} : !x \to \tilde{y}(G))\,(P\{x/w, x/z\} \mid (!y_i(y_i').Q)_i) &\longrightarrow \\
(\boldsymbol{\nu}w, \tilde{y} : !w \to \tilde{y}(G\{w/x\}))\,((\boldsymbol{\nu}z, \tilde{y} : !z \to \tilde{y}(G\{z/x\}))\,(P \mid (!y_i(y_i').Q)_i) \mid (!y_i(y_i').Q)_i) & \\
(\boldsymbol{\nu}x, \tilde{y} : x \to \tilde{y}.(X)G)\,(x[A].P \mid (x_i(X).Q_i)) &\longrightarrow (\boldsymbol{\nu}x, \tilde{y} : G\{A/X\})\,(P \mid (\tilde{Q})\{A/X\}) \\
(\boldsymbol{\nu}x, y : x^X \to y)\,(w \to x^X \mid Q) &\longrightarrow Q\{w/y\} \\
(\boldsymbol{\nu}x, y : y^X \to x)\,(x \to w^X \mid Q) &\longrightarrow Q\{w/y\}
\end{aligned}
$$

**Figure 5** GCP, Structural Equivalence and Reduction Rules.

Cut elimination in GCP depends crucially on the $\eta$-expansions both in processes and in global types. One could prove cut elimination directly for GCP, but instead we will appeal to the standard cut elimination result for CP.

As with the original MCP [9], it is sound to swap independent actions in global types in GCP (and our new variant of MCP). We omit these rules, which are the expected ones [9].

▶ **Example 6.** We continue with the exposition of our running example. To represent the 2-buyer protocol in GCP, it is no longer necessary to construct an unwieldy arbiter process as in Example 3 in CP, but rather construct a global type, which is easily derived in GCP. Let $G$ be the global type:

$$
\begin{aligned}
&B_1 \to S(B_1{}^{\mathbf{name}} \to S).\ S \to B_1(S^{\mathbf{cost}} \to B_1).\\
&\quad S \to B_2(S^{\mathbf{cost}} \to B_2).\ B_1 \to B_2(B_1{}^{\mathbf{cost}} \to B_2).\\
&\qquad B_2 \to S.\mathsf{case}(B_2 \to S(B_2{}^{\mathbf{addr}} \to S).(B_1, B_2) \to S,\ (B_1, B_2) \to S))
\end{aligned}
$$

Then, we can immediately prove the following coherence judgement:

$$
G \vDash \begin{array}{rcl} B_1 & : & \mathbf{name} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes 1, \\ B_2 & : & \mathbf{cost}^\perp \parr \mathbf{cost}^\perp \parr ((\mathbf{addr} \otimes 1) \oplus 1), \\ S & : & \mathbf{name}^\perp \parr \mathbf{cost} \otimes \mathbf{cost}^\perp \otimes ((\mathbf{addr}^\perp \parr \perp) \,\&\, \perp) \end{array}
$$

Using this global type in a CCut, we can compose the three processes for $B_1$, $B_2$, and $S$.

**Translations.** Translations from GCP to CP ($[\![-]\!]$) and from CP to GCP ($(\!|-|\!)$) are given in Figure 6. The function $[\![-]\!]$ from GCP to CP is a homomorphism on all process forms except coherence cut. Coherence cut is translated into a series of binary cuts where the global type

Global cut as binary cut

$$[\![(\boldsymbol{\nu}\tilde{x}^{\tilde{A}} : G)\,(\tilde{P})]\!] \stackrel{\text{def}}{=} (\boldsymbol{\nu}x_1{}^{A_1}y_1)\,([\![P_1]\!] \mid \cdots(\boldsymbol{\nu}x_n{}^{A_n}y_n)\,([\![P_n]\!] \mid [\![G]\!]\{\tilde{y}/\tilde{x}\})\cdots), \qquad \tilde{y} \text{ fresh}$$

Global types as processes

$$\begin{aligned}
[\![\tilde{x} \to y(G).H]\!] &\stackrel{\text{def}}{=} x_1(u_1).\cdots x_n(u_n).y[v].([\![G]\!]\{\tilde{u}/\tilde{x}, v/y\} \mid [\![H]\!]), && \tilde{u}, v \text{ fresh}\\
[\![\tilde{x} \to y]\!] &\stackrel{\text{def}}{=} x_1().\cdots x_n().y[]\\
[\![x \to \tilde{y}.\mathsf{case}(G,H)]\!] &\stackrel{\text{def}}{=} x.\mathsf{case}(y_1[\mathsf{inl}].\cdots y_n[\mathsf{inl}].[\![G]\!], y_1[\mathsf{inr}].\cdots y_n[\mathsf{inr}].[\![H]\!])\\
[\![x \to \tilde{y}.\mathsf{case}()]\!] &\stackrel{\text{def}}{=} x.\mathsf{case}()\\
[\![!x \to \tilde{y}(G)]\!] &\stackrel{\text{def}}{=} !x(u).?y_1[v_1].\cdots?y_n[v_n].[\![G]\!]\{u/x, \tilde{v}/\tilde{y}\}, && u, \tilde{v} \text{ fresh}\\
[\![x \to \tilde{y}.(X)G]\!] &\stackrel{\text{def}}{=} x(X).y_1[X].\cdots y_n[X].[\![G]\!]\\
[\![x^A \to y]\!] &\stackrel{\text{def}}{=} x \to y^A
\end{aligned}$$

Binary cut as global cut

$$(\!|(\boldsymbol{\nu}x^A y)\,(P \mid Q)|\!) = (\boldsymbol{\nu}x, y : y^A \to x)\,((\!|P|\!) \mid (\!|Q|\!))$$

**Figure 6** Translations Between CP and GCP.

becomes an arbiter process that mediates all communication. The function $(\!|-|\!)$ maps CP processes to GCP. It is a homomorphism on all process forms except binary cut. Binary cut is translated into a coherence cut with two processes in which the global type is a link.

**Why $\eta$-expansion?** We now pause to explain how $\eta$-expansion simplifies the system. In the current formulation, the axiom cut rules apply only at atomic types. If we attempt to allow axiom cut at other types, then we need some way of reducing a coherence cut over an axiom process in which the global type is not itself an axiom. For instance, consider $(\boldsymbol{\nu}x_1{}^1, x_2{}^{\perp} : x_1 \to x_2)\,(P \mid x_2 \to w^1)$. Translating to CP, we obtain $(\boldsymbol{\nu}x_1{}^1y_1)\,(P \mid (\boldsymbol{\nu}x_2{}^{\perp}y_2)\,(x_2 \to w^1 \mid y_1().y_2[]))$ which reduces by the unrestricted axiom cut rule to $(\boldsymbol{\nu}x_1{}^1y_1)\,(P \mid y_1().w[])$. An obstacle to mapping this reduction back to GCP is that the substitution of $w$ for $y_2$ occurs inside the arbiter process, that is, the image of a global type. In order to support this substitution we must generalise the CCut rule to allow free variables in global types. This in turn necessitates further reduction rules to prevent free variables in global types from blocking reduction. A more complex system including an unrestricted axiom cut rule is of practical interest as it admits implementations that take advantage of type erasure, but is beyond the scope of this paper.

▶ **Theorem 7** (Type preservation from GCP to CP).

1. *If $P \vdash \Gamma$ in GCP, then $[\![P]\!] \vdash \Gamma$ in CP.*
2. *If $G \vDash \Gamma$ in GCP, then $[\![G]\!] \vdash \Gamma^{\perp}$ in CP.*

A coherence judgement $G \vDash \Gamma$ translates to a judgement $[\![G]\!] \vdash \Gamma^{\perp}$, where $[\![G]\!]$ is an arbiter processes acting as an intermediary between the processes of a global session. It is typed in the dual of the environment in which we type the global type $G$. Write $\longrightarrow_{\eta}$ for $\eta$-expansion.

▶ **Theorem 8** (Simulation of GCP in CP).

1. *If $P \vdash \Gamma$ and $P \equiv Q$ in GCP, then $[\![P]\!] \equiv [\![Q]\!]$ in CP.*
2. *If $P \vdash \Gamma$ and $P \longrightarrow_{\eta} Q$ in GCP, then $[\![P]\!] \longrightarrow_{\eta} [\![Q]\!]$ in CP.*
3. *If $G \vDash \Gamma$ and $G \longrightarrow_{\eta} H$ in GCP, then $[\![G]\!] \longrightarrow_{\eta} [\![H]\!]$ in CP.*
4. *If $P \vdash \Gamma$ and $P \longrightarrow Q$ in GCP, then $[\![P]\!] \Longrightarrow^{+} [\![Q]\!]$ in CP.*

Each reduction in GCP is simulated by one or more reductions in CP. For instance, a cut involving a global type $\tilde{x} \to y(G).H$ performs a series of sends to the arbiter along $\tilde{x}$ followed by a receive from the arbiter along $y$. Theorem 8 shows that GCP is strongly normalising, by strong normalisation of CP (a standard result for classical linear logic).

▶ **Theorem 9** (Reflection of CP in GCP). *If $P \vdash \Gamma$ in GCP and $[\![P]\!] \longrightarrow Q'$ in CP, then there exists $Q$ such that $P \Longrightarrow Q$ in GCP and $Q' \Longrightarrow^* [\![Q]\!]$ in CP.*

Theorem 9 shows that cut-elimination (Theorem 5), and hence deadlock-freedom, holds for GCP, by cut-elimination for CP. For if $P$ does not reduce, then $[\![P]\!]$ must not reduce either, which means it is not a cut, which in turn means that $P$ is not a cut.

▶ **Theorem 10** (Type preservation from CP to GCP). *If $P \vdash \Gamma$, then $(\!|P|\!) \vdash \Gamma$.*

▶ **Theorem 11** (Simulation of CP in GCP).
1. *If $P \vdash \Gamma$ and $P \equiv Q$ in CP, then $(\!|P|\!) \equiv (\!|Q|\!)$ in GCP.*
2. *If $P \vdash \Gamma$ and $P \longrightarrow Q$ in CP, then $(\!|P|\!) \longrightarrow^+ (\!|Q|\!)$ in GCP.*

Structural equivalence in CP is simulated by structural equivalence in GCP. Each reduction in CP is simulated by one or two reductions in GCP. Before performing the main reduction, it is sometimes necessary to $\eta$-expand a global link.

**Coherence generalises duality.** It is well known that axiom at any type is admissible given axiom at atomic types, and that the proof of admissibility corresponds to $\eta$-expansion [19, Chapter 6]. If we restrict axioms to atomic propositions, then every cut-free derivation of a judgement $\vdash A^{\perp}, A$ in CLL corresponds to the $\eta$-expansion of a link process $x \to y^A$. Note the close relation between $\eta$-expansion and the translation of global types. For instance, compare the $\eta$-expansion of $\parr\otimes$ in CP with the translation of the global type for $\parr\otimes$ in GCP. For CP, we have that $x \to y^{A \otimes B}$ expands to

$$x(u).y[v].(P \mid Q)$$

where $P = u \to v^A$ and $Q = x \to y^B$, while for GCP, we have that $\tilde{x} \to y(G).H$ translates to

$$x_1(u_1).\cdots x_n(u_n).y[v].(P \mid Q)$$

where $P = [\![G]\!]\{\tilde{u}/\tilde{x}, v/y\}$ and $Q = [\![H]\!]$. The relation is similarly close for each of the other logical connectives. Hence, duality corresponds to $\eta$-expansion and coherence corresponds to a straightforward generalisation of $\eta$-expansion.

## 4 Multiparty Classical Processes (MCP)

The semantics of GCP is governed, unlike in standard multiparty session types [12]. For example, process $x[w].(P \mid Q)$ in GCP does not say to which other endpoint the message should be routed; a communication can happen only under a restriction with a global type that pairs such output action with an input action, e.g., when in a context such as in $(\boldsymbol{\nu} xy\tilde{z} : x \to y(G).H)\,(x[w].(P \mid Q) \mid \tilde{R})$. Thus, a global type is a central point of control. In standard multiparty session types, this is avoided by annotating actions with the endpoint that they should interact with. In our example, the process becomes $x^y[w].(P \mid Q)$, meaning that this output can synchronise only with an input performed at endpoint $y$ (which, dually, has to express that it intends to synchronise with an output from $x$). In this section, we define a variant of the calculus of Multiparty Classical Processes (MCP) [9], which follows the standard methodology of multiparty session types, simply by annotating types and processes of GCP with endpoints. Formally, MCP is defined as GCP with the following modifications.

Coherence rules

$$\frac{G \vDash (x_i{:}A_i)_i,\ y{:}C \quad H \vDash \Gamma,\ (x_i{:}B_i)_i,\ y{:}D}{\tilde{x} \to y(G).H \vDash \Gamma,\ (x_i{:}A_i \otimes^{y} B_i)_i,\ y{:}C \parr^{\tilde{x}} D}\ \otimes\parr \qquad \frac{}{\tilde{x} \to y \vDash (x_i{:}1^{y})_i, y{:}\bot^{\tilde{x}}}\ 1\bot$$

$$\frac{G_1 \vDash \Gamma, x{:}A, (y_i{:}C_i)_i \quad G_2 \vDash \Gamma, x{:}B, (y_i{:}D_i)_i}{x \to \tilde{y}.\mathsf{case}(G_1, G_2) \vDash \Gamma, x{:}A \oplus^{\tilde{y}} B, (y_i{:}C_i \& {}^{x} D_i)_i}\ \oplus\& \qquad \frac{G \vDash x{:}A, (y_i{:}B_i)_i}{!x \to \tilde{y}(G) \vDash x{:}?^{\tilde{y}}A, (y_i{:}!^{x}B_i)_i}\ !?$$

$$\frac{}{x \to \tilde{y}.\mathsf{case}() \vDash \Gamma, x{:}0^{\tilde{y}}, (y_i{:}\top^{x})_i}\ 0\top \qquad \frac{G \vDash \Gamma, x{:}A, (y_i{:}B_i)_i \quad X \notin \mathsf{ftv}(\Gamma)}{x \to \tilde{y}.(X)G \vDash \Gamma, x{:}\exists^{\tilde{y}}X.A, (y_i{:}\forall^{x}X.B_i)_i}\ \exists\forall$$

$$\frac{|A|^{\perp} = |B|}{x^{A} \to y^{B} \vdash x : A, y : B}\ \textsc{Axiom}$$

Typing rules

$$\frac{|A|^{\perp} = |B|}{x^{A} \to y^{B} \vdash x : A, y : B}\ \textsc{Axiom} \qquad \frac{(P_i \vdash \Gamma_i, x_i : A_i)_i \quad G \vDash (x_i : A_i)_i}{(\boldsymbol{\nu}\tilde{x}^{\tilde{A}} : G)\,(\tilde{P}) \vdash \tilde{\Gamma}}\ \textsc{CCut}$$

$$\frac{P \vdash \Gamma, y : A \quad Q \vdash \Delta, x : B}{x^{z}[y].(P \mid Q) \vdash \Gamma, \Delta, x : A \otimes^{z} B}\ \otimes \qquad \frac{P \vdash \Gamma, y : A, x : B}{x^{\tilde{z}}(y).P \vdash \Gamma, x : A \parr^{\tilde{z}} B}\ \parr$$

$$\frac{P \vdash \Gamma, x : A}{x^{\tilde{z}}[\mathsf{inl}].P \vdash \Gamma, u : A \oplus^{\tilde{z}} B}\ \oplus_1 \qquad \frac{P \vdash \Gamma, x : B}{x^{\tilde{z}}[\mathsf{inr}].P \vdash \Gamma, x : A \oplus^{\tilde{z}} B}\ \oplus_2 \qquad \frac{P \vdash \Gamma, x : A \quad Q \vdash \Gamma, x : B}{x^{z}.\mathsf{case}(P, Q) \vdash \Gamma, x : A \& {}^{z} B}\ \&$$

$$\frac{P \vdash\ ?\Gamma, y : A}{!x^{z}(y).P \vdash\ ?\Gamma, x : !^{z}A}\ ! \qquad \frac{P \vdash \Gamma, y : A}{?x^{\tilde{z}}[y].P \vdash \Gamma, x : ?^{\tilde{z}}A}\ ? \qquad \frac{P \vdash \Gamma}{P \vdash \Gamma, x : ?^{\tilde{z}}A}\ \textsc{Weaken}$$

$$\frac{P \vdash \Gamma, x : B[A/X]}{x^{\tilde{z}}[A].P \vdash \Gamma, x : \exists^{\tilde{z}}X.B}\ \exists \qquad \frac{P \vdash \Gamma, x : B \quad X \notin \mathsf{ftv}(\Gamma)}{x^{z}(X).P \vdash \Gamma, x : \forall^{z}X.B}\ \forall \qquad \frac{P \vdash \Gamma, y : ?^{\tilde{w}}A, z : ?^{\tilde{w}}A}{P[x/y, x/z] \vdash \Gamma, x : ?^{\tilde{w}}A}\ \textsc{Contract}$$

$$\frac{}{x^{z}[] \vdash x : 1^{z}}\ 1 \qquad \frac{P \vdash \Gamma}{x^{\tilde{z}}().P \vdash \Gamma, x : \bot^{\tilde{z}}}\ \bot \qquad \text{no rule for } 0 \qquad \frac{}{x^{z}.\mathsf{case}() \vdash \Gamma, x : \top^{z}}\ \top$$

**Figure 7** MCP, Coherence Rules and Typing Rules.

**Types.** The coherence relation for MCP, given in Figure 7, is identical to that of GCP, except from the type connectives, which are now annotated with the names of the endpoints with which they are supposed to interact. By an abuse of notation, we now let $A, B, C, D$ range over types with annotations. Rule Axiom is also slightly different then that of GCP. In its premise, we use the operation $|A|$, which removes all annotations from a given proposition $A$. For instance, $|B_1 \otimes^{x} B_2| = |B_1| \otimes |B_2|$. The syntax of global types is the same as that given for GCP, with the exception of the extra annotation in the term for the axiom.

Endpoint annotations restrict which derivations we can use to prove that some types are coherent. As an example, for some $A_i$, $B_i$, and $C$, consider the following annotated types:

$$x{:}A_1 \otimes^{y} B_1, \quad y{:}A_2 \parr^{x} B_2, \quad z{:}A_3 \parr^{w} B_3, \quad w : C$$

In MCP, it is necessary to eventually apply rule $\otimes\parr$ to $x{:}A_1 \otimes^{y} B_1$ and $y{:}A_2 \parr^{x} B_2$. However, in GCP, this would not be necessary: there may also be a coherence proof in which we apply rule $\otimes\parr$ to $x{:}A_1 \otimes^{y} B_1$ and $z{:}A_3 \parr^{w} B_3$ instead (after removing annotations).

The typing rules for MCP processes are given in Figure 7. As for coherence, the typing rules of MCP are those of GCP, but now with annotations. Importantly, the annotation of each communication action must be the same as that of the corresponding type construct. E.g., the send process is now written as $x^{z}[y].(P \mid Q)$, meaning that endpoint $x$ is sending $y$ to endpoint $z$, and the corresponding $\otimes$ connective in the type is annotated with the same $z$. Again, by an abuse of notation, we now let $P, Q, R$ range over processes with annotations.

**Semantics.** The semantics of MCP is the same as that of GCP, extended with the expected endpoint annotations. The consequence of annotations is that MCP enjoys an ungoverned semantics; it is fully distributed, as in the original theories of multiparty session types [12] and MCP [9]. As an example, here is the $\eta$-expansion rule for $\parr$ and $\otimes$:

$$x^{A \parr^{\tilde{w}} B} \to y^{C \otimes^{z} D} \longrightarrow x^{\tilde{w}}(u).y^{z}[v].(u^{A} \to v^{C} \mid x^{B} \to y^{D})$$

Similarly, here is the $\beta$-reduction rule for $\otimes$ and $\parr$:

$$\begin{aligned}(\boldsymbol{\nu}\tilde{z}, \tilde{x}, y : \tilde{x} \to y(G).H)\,(\tilde{S} \mid (x_i{}^{y}[x_i'].(P_i \mid Q_i))_i \mid y^{\tilde{x}}(y').R) \longrightarrow \\ (\boldsymbol{\nu}\tilde{x}', y' : G\{\tilde{x}'/\tilde{x}, y'/y\})\,(\tilde{S} \mid \tilde{P} \mid (\boldsymbol{\nu}\tilde{z}, \tilde{x}, y : H)\,(\tilde{Q} \mid R))\end{aligned}$$

In the reduction above (and all other reductions of MCP), the global type of the session is unnecessary for the communicating processes to know which others processes are involved in the communication; that information is instead taken from the endpoint annotations of their respective actions. Type preservation is ensured by the annotations used to type the processes, since that guarantees that coherence continues to hold for the reductum.

▶ **Theorem 12** (Subject reduction for MCP). *If $P \vdash \Gamma$ and $P \Longrightarrow Q$, then $Q \vdash \Gamma$.*

▶ **Theorem 13** (Cut elimination for MCP). *If $P \vdash \Gamma$, then there exists $Q$ such that $P \Longrightarrow^{*} Q$ and $Q$ is not a cut.*

Cut elimination for MCP follows from that for GCP (by Theorems 15, 16, 17, and 18).

▶ **Example 14.** We revisit the 2-buyer protocol from Example 6 in MCP. The global type is exactly the same (except for the axioms, which have the extra annotation on the left). However, the types of each endpoint are now appropriately annotated.

$$\begin{aligned}&B_1 : \mathbf{name} \otimes^{S} \mathbf{cost}^{\perp} \parr^{S} \mathbf{cost} \otimes^{B_2} 1^{S},\\ &B_2 : \mathbf{cost}^{\perp} \parr^{S} \mathbf{cost}^{\perp} \parr^{B_1} ((\mathbf{addr} \otimes^{S} 1^{S}) \oplus^{S} 1^{S}),\\ &S : \mathbf{name}^{\perp} \parr^{B_1} (\mathbf{cost} \otimes^{B_1} (\mathbf{cost} \otimes^{B_2} ((\mathbf{addr}^{\perp} \parr^{B_2} \perp^{B_1,B_2}) \,\&^{B_2}\, \perp^{B_1,B_2})))\end{aligned}$$

The annotations on the connectives say that, e.g., $B_1$ first sends a name to an implementation of endpoint $S$, and then, she receives a quote from $S$, before sending her bid to $B_2$.

**Translations.** Proofs in MCP translate to proofs in GCP, by erasing annotations. We write $|P|$ for the erasure of annotations from $P$. We obtain the following type preservation results.

▶ **Theorem 15** (Type preservation from MCP to GCP).
1. *If $P \vdash \Gamma$ in MCP, then $|P| \vdash |\Gamma|$ in GCP.*
2. *If $G \vDash \Gamma$ in MCP, then $|G| \vDash |\Gamma|$ in GCP.*

▶ **Theorem 16** (Type preservation from GCP to MCP).
1. *If $P \vdash \Gamma$ in GCP, then there exist $Q, \Delta$ such that $|Q| = P$, $|\Delta| = \Gamma$, and $Q \vdash \Delta$ in MCP.*
2. *If $G \vDash \Gamma$ in GCP, then there exist $H, \Delta$ such that $|H| = G$, $|\Delta| = \Gamma$, and $H \vDash \Delta$ in MCP.*

We also obtain a lockstep bisimulation between MCP and GCP.

▶ **Theorem 17** (Simulation of MCP in GCP). *Assume $P \vdash \Gamma$ and $G \vDash \Gamma$ in MCP.*
1. *If $P \equiv Q$ in MCP, then $|P| \equiv |Q|$ in GCP.*
2. *If $P \longrightarrow Q$ in MCP, then $|P| \longrightarrow |Q|$ in GCP.*
3. *If $G \longrightarrow H$ in MCP, then $|G| \longrightarrow |H|$ in GCP.*

▶ **Theorem 18** (Reflection of GCP in MCP). *Assume* $P \vdash \Gamma$ *and* $G \vDash \Gamma$ *in MCP.*

1. *If* $|P| \equiv Q$ *in GCP, then there exists* $R$ *such that* $|R| = Q$ *and* $P \equiv R$ *in MCP.*
2. *If* $|P| \longrightarrow Q'$ *in GCP, then there exists* $Q$ *such that* $|Q| = Q'$ *and* $P \longrightarrow Q$ *in MCP.*
3. *If* $|G| \longrightarrow H'$ *in GCP, then there exists* $H$ *such that* $|H| = H'$ *and* $G \longrightarrow H$ *in MCP.*

Combining these results with those for the translation from GCP to CP, we obtain an end-to-end translation of distributed multiparty sessions into arbitrated binary sessions.

## 5 Related and Future Work

**Arbiters.** Caires and Perez [4] show how to translate multiparty sessions by translating a global type to a process, which they call a *medium*. Their medium is similar to our arbiter, and their translation of a global type to a medium is similar to our translation of GCP to CP, which takes a global type to an arbiter. Their system, like ours, guarantees fidelity and deadlock freedom via the translation; and, like us, they extend their system to support polymorphism. However there are several differences. Our work is based on classical linear logic, whereas theirs is based on intuitionistic linear logic; we suspect that their approach could be adopted to classical logic, and ours to intuitionistic. More importantly, where we use coherence, they use the standard definition of projection [12]. Projection is closer to the original formulation of multiparty session types [12], but lacks the tighter connection to logic offered by coherence, in particular the way in which coherence is organised around pairs of logical connectives, generalising duality. Unlike us, they do not consider replication and nesting in global types. Their medium process imposes global governance, similar to our GCP, but they offer no decentralised system based on direct communication like our MCP.

**Multiparty session types.** Coherence logically reconstructs the notion of well-formedness found in multiparty session types [12, 10], in the context of synchronous communication [2]. Polymorphism for binary session types is considered in a proposition-as-types setting by Wadler [21] and Caires *et al.* [5]; we have generalised this notion to the multiparty case. Our new coherence proof system extends the one presented in the original MCP [9] with rules for polymorphism. Since coherence yields an algorithm for extracting a global type from a set of types [9], ours is also the first work dealing with the extraction of polymorphic global types.

**Issues with axiom.** In the original formulation of CP [22], axiom reduction applies at all types and all reductions are independent of types. In the current formulation, axiom reduction applies only at atomic types and other instances are handled by $\eta$-rules that depend upon types. As a result, the implementation of type instantiation may be problematic. Alternative implementations may seek to restore a version of axiom that applies at all types. We have explored one variant that does so, but the formalism is more complicated as it requires free variables in global types. We leave further exploration to future work.

**Coherence.** Coherence in GCP may be generalised. In $\otimes\parr$, the context $\Gamma$ in the conclusion may be distributed to the two premises. Similarly, in !? a context !$\Gamma$ may be added to the premise and conclusion of the rule, splitting channels between those that retain ! in the premise and those that lose it. While such splits are straightforward in GCP, it is unclear how to add them without centralised control in MCP. We also leave this to future work.

**Choreography.** Carbone *et al.* [8] search for a propositions-as-types correspondence for the calculus of compositional choreographies of Montesi and Yoshida [15]. This work inspired the notion of coherence in the original MCP [9] — typing choreographies requires a similar handling of multiple connectives. However, it was limited to binary sessions, whereas the original theory by Montesi and Yoshida supported multiparty sessions. This work may close the circle: our generalisation of CP to GCP seems applicable to choreographies, and would yield an expressive choreography language with parametric polymorphism.

**Relationship with standard multiparty session types.** MCP is directly connected to classical linear logic, while also bearing a close resemblance to traditional multiparty session types. However, there remain important differences between the two. First, we do not handle recursive behaviour [14]. Second, multiparty session types support broadcast from one sender to many receivers, while our types gather information from many senders to one receiver – a choice dictated by our desire to translate MCP to CP. We look forward to further studying the relation between MCP, GCP, CP, and other systems with multiparty session types.

## References


1 Samson Abramsky. Proofs as processes. *Theor. Comput. Sci.*, 135(1):5–9, 1994.
2 Andi Bejleri and Nobuko Yoshida. Synchronous multiparty session types. *ENTCS*, 241:3–33, 2009.
3 Gianluigi Bellin and Philip J. Scott. On the pi-calculus and linear logic. *TCS*, 135(1):11–65, 1994.
4 Luís Caires and Jorge Perez. Multiparty session types within a canonical binary theory, and beyond. In *FORTE*, 2016.
5 Luís Caires, Jorge A. Pérez, Frank Pfenning, and Bernardo Toninho. Behavioral polymorphism and parametricity in session-based communication. In *ESOP*, pages 330–349, 2013.
6 Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In *CONCUR*, pages 222–236, 2010.
7 Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *MSCS*, 26(3):367–423, 2016.
8 Marco Carbone, Fabrizio Montesi, and Carsten Schürmann. Choreographies, logically. In *CONCUR*, pages 47–62, 2014.
9 Marco Carbone, Fabrizio Montesi, Carsten Schürmann, and Nobuko Yoshida. Multiparty session types as coherence proofs. In *CONCUR*, pages 412–426, 2015.
10 Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida, and Luca Padovani. Global progress for dynamically interleaved multiparty sessions. *MSCS*, 760:1–65, 2015.
11 Kohei Honda, Vasco Vasconcelos, and Makoto Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP*, pages 22–138, 1998.
12 Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *JACM*, 63(1):9, 2016. Also: POPL, 2008, pages 273–284.
13 Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *TCS*, 410(46):4747–4768, 2009.
14 Sam Lindley and Garrett Morris. Talking bananas: structural recursion for session types. In *ICFP*. ACM, 2016. To appear.
15 Fabrizio Montesi and Nobuko Yoshida. Compositional choreographies. In *CONCUR*, pages 425–439, 2013.
16 Jennifer Paykin and Steve Zdancewic. Linear $\lambda\mu$ is CP (more or less). In *A List of Successes That Can Change The World*, pages 273–291, 2016.

**17** Frank Pfenning and Dennis Griffith. Polarized substructural session types. In *Foundations of Software Science and Computation Structures*, pages 3–22. Springer, 2015.

**18** Davide Sangiorgi. Pi-calculus, internal mobility, and agent-passing calculi. *TCS*, 167(1&2):235–274, 1996.

**19** A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory (2nd Ed.)*. Cambridge University Press, 2000.

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

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

**22** Philip Wadler. Propositions as sessions. *JFP*, 24(2–3):384–418, 2014. Also: ICFP, pages 273–286, 2012.