Coherence Generalises Duality: a logical explanation of multiparty session types

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

1 IT University of Copenhagen
2 University of Edinburgh
3 University of Southern Denmark

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.

1 Introduction

Session types, introduced by Honda, Vasconcelos, and Kubo [13, 22], 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 was found by Caires and Pfenning [6, 7]. Merging these, Wadler [23] introduced Classical Processes (CP), in which session types correspond to propositions of classical linear logic, processes to proofs, and communication to proof normalisation. Key properties of session types such as deadlock freedom follow from key properties of linear logic such as cut elimination. Until last year, this line of work was limited to binary session types where protocols involve exactly two participants, even though in the more general theory of multiparty session types, introduced by Honda, Yoshida, and Carbone [14], protocols may involve many participants.

Last year, Carbone *et al.* introduced a calculus of Multiparty Classical Processes (MCP) [9], which extends CP to multiparty session types. MCP generalises what it means for propositions to be dual. In linear logic 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).

The original version of MCP came at a cost as compared to CP. First, MCP required annotating logical connectives with the 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 $⅋$, 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.

We begin by introducing a 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 [22]. We partition types into input and output types (a slight variation of positive and negative polarities [15, 19]), which allows us to orient the axiom rule. And, in order to align with the

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

subsequent development, we restrict the cut of axiom against a process to atomic variables, applying η-expansion to handle the remaining cases. CP is discussed in Section 2.

Next, we introduce a calculus of Globally-governed Classical Processes (GCP), intermediate between CP and MCP. GCP introduces 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. GCP supports the full expressiveness of CP, including parametric polymorphism. We present a semantics-preserving translation of GCP into CP. A multiparty session among n participants in GCP is encoded by $n+1$ processes in CP. The $n+1$'th process is the *arbiter*, an auxiliary process that coordinates communications among the other n processes, and is synthesised from the global type of the session. We show under our translation that GCP is simulated by CP. We also give a translation from CP to GCP and show that it too is a simulation. GCP is discussed in Section 3.

Finally, we introduce a 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 $⅋$ 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. We discuss MCP in Section 4.

▶ **Example 1.** We illustrate our encodings using the classic 2-buyer protocol [14] 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 then 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 relies on representing data using the unit type. In this paper we present non-degenerate encodings in CP, GCP, and MCP.

2 Classical Processes (CP)

We describe the calculus of Classical Processes (CP), originally introduced by Wadler [23].

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)*	$\mid$	$A ⅋ B$	*(receive A, proceed as B)*
	$\mid$	$A \oplus B$	*(select A or B)*	$\mid$	$A \& B$	*(offer A or B)*
	$\mid$	0	*(unit for $\oplus$)*	$\mid$	$\top$	*(unit for &)*
	$\mid$	1	*(unit for $\otimes$)*	$\mid$	$\bot$	*(unit for ⅋)*
	$\mid$	$?A$	*(client request)*	$\mid$	$!A$	*(server accept)*
	$\mid$	X	*(atomic propositions)*	$\mid$	$X^\perp$	*(dual of atomic proposition)*
	$\mid$	$\exists X.A$	*(existential)*	$\mid$	$\forall X.A$	*(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, $A ⅋ B$ 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 ⅋, 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 $X^\perp$, and universal propositions $\exists X.A$ and $\forall X.A$ model polymorphic channels [23].

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

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

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

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

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

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

Figure 1 CP, Type Rules.

Each type in the left-hand column is dual to the corresponding type in the right-hand column. We write $A^\perp$ 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 [15]. 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.

In CP, proofs correspond to proof terms, expressed in a π-calculus with sessions. Let x, y, and z range over endpoints. The syntax of processes is given by the proof terms (denoted in red) reported in Figure 1. We assume that in a sending operation, the object of the communication is always contained in squared brackets $[\ldots]$, and, dually, the information received in an input operation is in round parenthesis $(\ldots)$. The syntax we adopt corresponds to the standard π-calculus with sessions, but with a few twists. As in the internal π-calculus [20], 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 $(\boldsymbol{\nu} x^A y)\,\big(P \mid Q\big)$ pairs two endpoints x and y into a session (as done by Vasconcelos [22]). The way we formulate restriction is a main difference between our CP and its original version [23].

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 [23] except for rule Cut which now adapts to the new syntax for restriction. As a benefit, we have now that endpoint names x and y are used linearly in processes P and Q respectively.

Semantics CLL is equipped with proof transformations that can be used for giving semantics to CP processes. Figure 2 displays structural equivalence rules, η-expansions and β-reductions. Structural equivalences permit swapping the names in a link and in a restriction, and reassociating two restrictions. The η-expansions for link do not appear in the original presentation of CP [23], but are standard and appear elsewhere [18]. We reformulate CP to use them, as they will prove helpful in defining GCP in the next section. The expansion replaces a link, step-by-step, by processes that perform the communications required by its type. The β-reductions and commuting conversions (which we omit due to lack of space) of CP correspond to the cut elimination rules in CLL, and are standard. Structural equivalence and reductions apply inside a cut, but not inside a prefix.

Structural equivalence

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

η-expansions

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

β-reductions

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

Figure 2 CP, Structural Equivalence and Reduction Rules.

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

▶ **Theorem 3** (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 [23]. All other theorems in this paper follow by straightforward induction.

▶ **Example 4.** We return to Example 1 and describe now the 2-buyer protocol in CP. First, we proceed by deriving a set of suitable types for B_1, B_2, and S. We assume that all communication proceed through a single channel, and furthermore that **cost**, **name**, **addr** are atomic proposition introduced by the $\forall$. In CP, B_1's process has type $\mathbf{name} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes 1$, B_2's process has type $\mathbf{cost}^\perp \parr \mathbf{cost}^\perp \parr \big((\mathbf{addr} \otimes \bot) \oplus \bot\big)$ and S's process has type $\mathbf{name}^\perp \parr \mathbf{cost} \otimes \mathbf{cost} \otimes \big((\mathbf{addr}^\perp \parr 1) \& 1\big)$. Second, we need to determine the type of an arbiter process, otherwise it is impossible to run these three processes in parallel, so that they all follow the 2-buyer protocol. Such an arbiter process, let us call it A, relays every message between B_1, B_2, and S. The type of the arbiter process defines the possible interleavings of messages. We only give one example arbiter type:

$$\mathbf{name}^\perp \parr \mathbf{name} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes \\ \big((\mathbf{addr}^\perp \otimes 1) \oplus 1\big) \parr \big((\mathbf{addr} \parr \bot) \,\&\, \bot\big) \otimes \bot$$

Directing link and restriction We make a useful observation here that does not appear in [23]. A link or a restriction will always be between an input type and its dual output type. The swap rule may always be applied to orient the 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,

$$P_x \stackrel{\text{def}}{=} x[y].(y[\mathsf{inl}].y[\,] \mid x[\mathsf{inl}].x[\,]) \qquad\qquad Q_x \stackrel{\text{def}}{=} x[y].(y[\mathsf{inr}].y[\,] \mid x[\mathsf{inr}].x[\,])$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{y[\,] \vdash y : 1}\,1}{y[\mathsf{inl}].y[\,] \vdash y : 1 \oplus 1}\,\oplus_1 \quad \dfrac{\dfrac{}{x[\,] \vdash x : 1}\,1}{x[\mathsf{inl}].x[\,] \vdash x : 1 \oplus 1}\,\oplus_1}{P_x \vdash x : (1 \oplus 1) \otimes (1 \oplus 1)}\,\otimes}{w().P_x \vdash w : \bot, x : (1 \oplus 1) \otimes (1 \oplus 1)}\,\bot \quad \dfrac{\dfrac{\dfrac{\dfrac{}{y[\,] \vdash y : 1}\,1}{y[\mathsf{inr}].y[\,] \vdash y : 1 \oplus 1}\,\oplus_1 \quad \dfrac{\dfrac{}{x[\,] \vdash x : 1}\,1}{x[\mathsf{inr}].x[\,] \vdash x : 1 \oplus 1}\,\oplus_1}{Q_x \vdash x : (1 \oplus 1) \otimes (1 \oplus 1)}\,\otimes}{w().Q_x \vdash w : \bot, x : (1 \oplus 1) \otimes (1 \oplus 1)}\,\bot}{w.\mathsf{case}(w().P_x, w().Q_x) \vdash w : \bot \mathbin{\&} \bot, x : (1 \oplus 1) \otimes (1 \oplus 1)}\,\&}{x(w).w.\mathsf{case}(w().P_x, w().Q_x) \vdash x : (\bot \mathbin{\&} \bot) \parr ((1 \oplus 1) \otimes (1 \oplus 1))}\,\parr$$

Figure 3 Example: duplicating a bit

the swap rule may always be applied to orient the 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 Figure 3. It illustrates the derivation of 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). The parts of the figure in blue correspond precisely to classical linear logic [12], while the parts in red correspond to CP as originally defined in [23].

Would it make sense to modify our interpretation of the parts 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 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 of 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.)

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

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

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

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

Figure 4 GCP, Coherence Rules and Coherence Cut.

3 Globally-governed Classical Processes (GCP)

In CP, communications between two parties take place over a binary cut. In this section, we introduce GCP, which is the same as CP except that *binary* cut is replaced by a *multiparty* cut construct (dubbed coherence cut), in which communications among n parties are governed by a global session 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 and input type in B and vice versa. Duality ensure that two composed processes communicate safely, that is they have compatible interfaces (session fidelity) and cannot get stuck (deadlock freedom). 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 Γ is a set of compatible types and G, called *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) given in Figure 4. 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 $x \to \tilde{y}.(X)G$, endpoint x sends a type to endpoints $\tilde{y}$ and the protocol proceeds as G. In $x \to y^B$, x is connected to y. Thus, a coherence cut in which the global type is an axiom behaves exactly like a binary cut.

In order to define coherence for GCP, we observe that types of 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 $A \otimes B$ 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. We could make rule $\otimes\parr$ more general by splitting the context between G and H. We deliberately avoid splitting the context because it is unclear how to implement this behaviour using multiparty session types without keeping hold of the global type at runtime.

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. 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

Structural equivalence

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

η-expansions

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

β-reductions

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

Figure 5 GCP, Structural Equivalence and Reduction Rules.

classical linear logic. As in the original formulation of MCP [9], the restiction 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 Reduction and structural equivalence rules for GCP are given in Figure 5. The structural equivalence rules are straightforward. The η-expansions in processes are identical to those of Fig. 2. We also define a notion of η-expansion on global types. Due to lack of space, we omit the straightforward commuting conversions, each one allowing a prefix to be lifted out of a cut [9]. The β-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 η-expansions are necessary for cut-elimination to hold. For example, the process $(\boldsymbol{\nu}x^{A\otimes B}y^{A^\perp \invamp C}z^D : x \to y(G).H)\,(w \to x^{A\otimes B} \mid y(y').P \mid Q)$ 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 5** (Subject reduction for GCP). *If $P \vdash \Gamma$ and $P \Longrightarrow Q$, then $Q \vdash \Gamma$.*

▶ **Theorem 6** (Cut elimination for GCP). *If $P \vdash \Gamma$, then there exists Q such that $P \Longrightarrow^* Q$ and Q is not a cut.*

Cut elimination in GCP depends crucially on the η-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.

▶ **Example 7.** 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 4 in CP, but rather construct a global type, which is easily derived in GCP. Let G be the global type:

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

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

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

Binary cut as global cut

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

Figure 6 Translations Between CP and GCP.

Then, we can immediately prove the following coherence judgement:

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

Using this global type in the side condition of the CCut rule, we can compose the three processes for B_1, B_2, and S directly.

Translations Translations from GCP to CP ($[\![-]\!]$) and from CP to GCP ($(\!|-|\!)$) are presented in Figure 6. The function $[\![-]\!]$ maps GCP processes and global types to CP processes. It is a homomorphism on all process forms except global cut. Global cut is translated into a series of binary cuts in which the global type is translated into 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 global cut with two processes in which the global type is a link.

▶ **Theorem 8** (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 and hence it is typed in the dual of the environment in which we type the global type G (where duality on type environments is defined pointwise on the types of the variables). We write $\longrightarrow_\eta$ for η-expansion, and $\Longrightarrow_\eta$ for $\equiv\longrightarrow_\eta\equiv$.

▶ **Theorem 9** (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 9 shows that GCP is strongly normalising, by strong normalisation of CP (a standard result for classical linear logic).

▶ **Theorem 10** (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 10 shows that cut-elimination (Theorem 6), 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 11** (Type preservation from CP to GCP). *If $P \vdash \Gamma$, then $(\!|P|\!) \vdash \Gamma$.*

▶ **Theorem 12** (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 (usually) necessary to η-expand a global link.

Coherence generalises duality It is well known that duality corresponds to η-expansion. If we restict axioms to atomic propositions, then every cut-free derivation of a judgement $\vdash A^{\perp}, A$ in CLL corresponds to the η-expansion of a link process $x \to y^{A}$. Note the close relation between η-expansion and the translation of global types. For instance, compare the η-expansion of $⅋\otimes$ in CP with the translation of the global type for $⅋\otimes$ 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 η-expansion and coherence corresponds to a straightforward generalisation of η-expansion. This observation justifies our proclamation that coherence generalises duality.

4 Multiparty Classical Processes (MCP)

An important difference between GCP and standard multiparty session types [14] is that the semantics of the former is globally governed by global types, whereas that of the latter depends only on the local actions performed by processes. 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 G that pairs such output action with an input action, e.g., when put in a context such as in $(\boldsymbol{\nu} xyz : \tilde{x} \to y(G).H)\,(x[w].(P \mid Q) \mid \tilde{R})$. This makes the global type a central point of control for the entire session. In standard multiparty sessions, this problem is avoided by annotating actions with the endpoint that they are intended to interact with. In our example, the send action becomes $x^{y}[w].(P \mid Q)$, meaning that this output can synchronise only with an input action performed by endpoint y (which, dually, also 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 modifications reported in the following.

Coherence rules

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

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

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

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

Typing rules

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

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

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

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

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

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

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 name of the endpoint(s) they are supposed to interact with. With an abuse of notation, we denote annotated propositions with letters A, B, C and D, as in CP and GCP. 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 proof 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 set of 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, for the above to be coherent, 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 endpoint annotations. Importantly, the annotation of each communication action must be the same as that of the corresponding type construct. For example, the send process is now written as $x^z[y].(P \mid Q)$, meaning that endpoint x is sending y over to endpoint z, and the corresponding $\otimes$ connective in the type is annotated with the same z. Also for processes, we retain the same notation as in CP and GCP by using P, Q, and R.

Semantics The semantics of MCP is the same as that of GCP, except for added endpoint annotations. The consequence of adopting endpoint annotations is that MCP enjoys an ungoverned semantics; it is fully distributed, as in the original theory of multiparty session types [14] and the original presentation of MCP [9].

Decorating the reductions with annotations is straightforward, since the annotations are determined by the types. As an example, here is the η-expansion rule for $⅋$ and $\otimes$:

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

Similarly, here is the β-reduction rule for $\otimes$ and $⅋$:

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

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

▶ **Theorem 14** (Cut elimination for MCP)**.** *If $P \vdash \Gamma$, then there exists Q such that $P \Longrightarrow^{*} Q$ and Q is not a cut.*

As we shall see in the next section, cut elimination for MCP follows directly from that for GCP.

▶ **Example 15.** We revisit Example 7 and express the 2-buyer protocol in MCP. The global type derived by coherence is exactly the same as that derived in Example 7 (except for the axioms, which have the extra annotation on the left). However, the types of the participating processes are slightly different as they are augmented by the appropriate endpoint annotations.

$$B_1 : \mathbf{name}^{\perp} \otimes^{S} \mathbf{cost} ⅋^{S} \mathbf{cost}^{\perp} \otimes^{B_2} 1^{B_1},$$
$$B_2 : \mathbf{cost} ⅋^{S} \mathbf{cost} ⅋^{B_1} ((\mathbf{addr}^{\perp} \otimes^{S} \perp^{B_2}) \oplus^{S} \perp^{B_2}),$$
$$S : \mathbf{name} ⅋^{B_1} (\mathbf{cost}^{\perp} \otimes^{B_1} (\mathbf{cost}^{\perp} \otimes^{B_2} ((\mathbf{addr} ⅋^{B_2} 1^{S}) \mathbin{\&}^{B_2} 1^{S})))$$

We only explain the type of B_1. The annotations on the connectives say that B_1 first sends a name to a session implementing endpoint S. Then, she receives a quote from S, before sending her bid to B_2.

Translations Proofs in MCP are easily translated into proofs in GCP, simply by erasing annotations. We write $|P|$ for the erasure of endpoint annotations from P. We obtain the following type preservation results.

▶ **Theorem 16** (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 17** (Type preservation from GCP to MCP)**.**
1. *If $P \vdash \Gamma$ in GCP, then there exist Q and Δ such that $|Q| = P$, $|\Delta| = \Gamma$, and $Q \vdash \Delta$ in MCP.*
2. *If $G \vDash \Gamma$ in GCP, then there exist H and Δ such that $|H| = G$, $|\Delta| = \Gamma$, and $H \vDash \Delta$ in MCP.*

We also obtain a lockstep bisimulation between MCP and GCP.

▶ **Theorem 18** (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 19** (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 obtained for the translation from GCP to CP, our development yields an end-to-end translation of distributed multiparty sessions to arbitrated binary sessions.

5 Related Work

Here we summarise related work not cited elsewhere in the paper.

Arbiters Similar to our translation of GCP to CP, Caires and Perez [4] show how to translate multiparty sessions using an arbiter process. They do not give a propositions-as-types correspondence, and instead of coherence use a projection operation closer to the original formulation of multiparty session types [14].

Polymorphism Polymorphism for session types is considered in a proposition-as-types setting by Caires *et al.* [5]. Simmons [21] studies suspended propositions that arise in focused logics with atomic propositions. His findings are related to our definition of coherence.

Multiparty session types Coherence corresponds to the notion of well-formedness found in multiparty session types [14, 10], in the context of synchronous communication [2]. Our nested constructs for global types are inspired by those of Demangeon and Honda [11].

6 Future Work

Issues with axiom In the original formulation of CP [23], axiom applies at all types and all reductions are independent of types. In the current formulation, axiom applies only at atomic types and other instances are handled by η-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.

Generalised forms of coherence Our definition of coherence for GCP may be generalised. In $\otimes ⅋$, the context Γ 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 which retain $!$ in the premise and those which 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 [17]. This work inspired the notion of coherence in the original version of MCP [9] — choreographies require a similar handling of multiple connectives in typing rules. However, it was limited to binary sessions, whereas the original theory by Montesi and Yoshida supported multiparty sessions. We conjecture that our work may finally close the circle: our generalisation of CP to GCP seems straightforward to apply to choreographies, and would yield an expressive choreography language with parametric polymorphism.

Relationship with standard multiparty session types MCP has a direct connection 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, though preliminary results extending CP to support recursion look promising [16]. Second, multiparty session types support broadcast from one sender to many receivers, while our global 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 study of 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 Luis Caires and Jorge Perez. Multiparty session types within a canonical binary theory, and beyond. In *FORTE*, 2016. To appear.
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 Romain Demangeon and Kohei Honda. Nested protocols in session types. In *CONCUR*, pages 272–286, 2012.
12 Jean-Yves Girard. Linear logic. *TCS*, 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. *JACM*, 63(1):9, 2016. Also: POPL, 2008, pages 273–284.
15 Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *TCS*, 410(46):4747–4768, 2009.
16 Sam Lindley and Garret Morris. Talking bananas, 2016. draft.
17 Fabrizio Montesi and Nobuko Yoshida. Compositional choreographies. In *CONCUR*, pages 425–439, 2013.
18 Jennifer Paykin and Steve Zdancewic. Linear $\lambda\mu$ is CP (more or less). In *A List of Successes That Can Change The World*, volume 9600 of *LNCS*, pages 273–291. Springer, 2016.
19 Frank Pfenning and Dennis Griffith. Polarized substructural session types. In *Foundations of Software Science and Computation Structures*, pages 3–22. Springer, 2015.
20 Davide Sangiorgi. Pi-calculus, internal mobility, and agent-passing calculi. *TCS*, 167(1&2):235–274, 1996.
21 Robert J. Simmons. Structural focalization. *ACM Trans. Comput. Log.*, 15(3):21:1–21:33, 2014.
22 Vasco T. Vasconcelos. Fundamentals of session types. *Inf. Comput.*, 217:52–70, 2012.
23 Philip Wadler. Propositions as sessions. *JFP*, 24(2–3):384–418, 2014. Also: ICFP, pages 273–286, 2012.