Synchronous Forwarders

Marco Carbone[1], Sonia Marin[2], and Carsten Schürmann[1]

[1] IT-University of Copenhagen, Denmark
{carbonem,carsten}@itu.dk
[2] University College London, UK
s.marin@ucl.ac.uk

Abstract. Session types are types for specifying protocols that processes must follow when communicating with each other. Session types are in a propositions-as-types correspondence with linear logic. Previous work has shown that a multiparty session type, a generalisation of session types to protocols of two or more parties, can be modelled as a proof of coherence, a generalisation of linear logic duality. And, protocols expressed as coherence can be simulated by arbiters, processes that act as a middleware by forwarding messages according to the given protocol. In this paper, we generalise the concept of arbiter to that of synchronous forwarder, that is a processes that implements the behaviour of an arbiter in several different ways. In a propositions-as-types fashion, synchronous forwarders form a logic equipped with cut elimination which is a special restriction of classical linear logic. Our main result shows that synchronous forwarders are a characterisation of coherence, i.e., coherence proofs can be transformed into synchronous forwarders and, viceversa, every synchronous forwarder corresponds to a coherence proofs.

Keywords: Forwarders · Session Types · Linear Logic.

1 Introduction

A concurrent system is more than a sum of processes. It also includes the fabric that determines how processes are tied together. Session types, originally proposed by Honda et al. [11], are type annotations that ascribe protocols to processes in a concurrent system and, as a fabric, determine how they behave when communicating with each other. Such type annotations are useful for various reasons. First, they serve as communication blueprints for the entire system and give programmers clear guidance on how to implement communication patterns at each endpoint (process or service). Second, they make implementations of concurrent systems safer, since well-typedness entails basic safety properties of programs such as *lack of communication errors* ("if the protocol says I should receive an integer, I will never receive a boolean"), *session fidelity* ("my programs follow the protocol specification patterns"), and *in-session deadlock freedom* ("the system never gets stuck by running a protocol"). Intuitively, they make sure that the processes are *compatible* and that they exchange messages in the prescribed way for the concurrent system to work correctly. For example,

by preventing messages from being duplicated, as superfluous messages would not be accounted for, and by preventing messages from getting lost, otherwise a process might get stuck, awaiting a message.

In the case of *binary sessions types*, the version of session types that deals only with protocols between two parties, compatibility means for type annotations to be dual to one another: the send action of one party must be matched by a corresponding receive action of the other party, and vice versa. Curiously, binary session types find their logical foundations in linear logic, as identified by Caires and Pfenning [3, 2] and later also by Wadler [20, 21]. They have shown that session types correspond to linear logic propositions, processes to proofs, reductions in the operational semantics to cut reductions in linear logic proofs, and compatibility to the logical notion of duality for linear formulas. Duality, thus, defines the fabric for how two processes communicate in an idealised world, while at the same time abstracting away from all practical details, such as message delay, message order, or message buffering.

The situation is not as direct for *multiparty session types* [12, 13], type annotations for protocols with two or more participants. Carbone et al. [7, 5] extended Wadler's embedding of binary session types into classical linear logic (CLL) to the multiparty setting, by generalising duality to the notion of *coherence*. They observed that the in-between fabric holds the very key to understanding multiparty session types: when forcing the type annotations to be *coherent*, one ensures that sent messages will eventually be collected. Coherence as a deductive system allows one to derive exactly these compatible jugements, while proofs correspond precisely to multiparty protocol specifications. A key result is that coherence proofs can be encoded as well-typed (as proofs in CLL) processes, called *arbiters*, which means that the fabric can actually be formally seen as a process-in-the-middle. However, no precise logical characterisation of what constitutes arbiters was given. In this paper, we continue this line of research and define a subsystem of processes, called *synchronous forwarders*, that provides one possible such characterisation that guarantees coherence.

As the name already suggests, a forwarder is a process that forwards messages, choices, and services from one endpoint to another according to the protocol specification. Intuitively, similarly to an arbiter, a forwarder process mimics the fabric by capturing the message flow. However, when data-dependencies allow, forwarders could, in theory, non-deterministically choose to receive messages from different endpoints, and then forward such messages at a later point. Or, they can also decide to buffer a certain number of messages from a given receiver. In any case, they retransmit messages only after receiving them, without interpreting, modifying, or computing with them.

In this work, synchronous forwarders support buffers of size 1 – only one message can be stored for each endpoint at a given time. This preserves the order of messages from the same sender, i.e., after receiving a message from one party, the forwarder blocks the connection to such party until the message has been delivered to its destination. Synchronous forwarders could be used

to explain communication patterns as they occur in practice, such as message routing, proxy services, and runtime monitors for message flows.

The meta-theoretic study of synchronous forwarders allows us to conclude that there is a proof-as-processes correspondence between synchronous forwarders and coherence proofs. We show that synchronous forwarders can be safely composed through cut elimination, which allows us to combine the fabric between two concurrent systems (figuring arbitrary many processes). With respect to coherence, we prove their *soundness*, namely that any coherence proof can be emulated by a synchronous forwarder simulating the actions of the fabric, and their *completeness*, meaning that every synchronous forwarder guarantees coherence. In particular, arbiters are special instances of forwarders.

Outline and key contributions. The key contributions of this paper include

- a logical characterisation of *synchronous forwarders* (§ 3);
- a reductive operational semantics based on cut-elimination (§ 4);
- a tight correspondence between coherence and synchronous forwarders (§ 5).

Additionally, § 2 recaps the definitions of coherence, processes, and arbiters, while § 6 discusses related and future work. Concluding remarks are in § 8.

2 Coherence, Processes, and Arbiters

To prepare the ground for formally defining synchronous forwarders, we use this section to introduce the basic ingredients: the formal notion of *coherence* [5], the syntax of *processes* [20, 21] used for synchronous forwarders, and the encoding of coherence proofs into special processes known as *arbiters* [5].

2.1 Coherence

Coherence [13, 7, 5], which provides the formal foundation for our results, is a generalisation of the notion of *duality* from CLL [9]. Duality is used by binary session types when composing (two) processes through a communication channel: the two ends of the channel are compatible whenever their types are *dual* to each other, i.e., every output is matched by an input and viceversa. Coherence can be understood as a generalisation of duality, that is, a criterion that decides if two or more parties can agree on who sends what to whom. Clearly, as there can be more than two parties, it is impossible to base this criterion on duality alone. Intuitively, we say that a set of processes are coherent, if each send can be linked to an available receive from another party.

Example 1 (The 2-Buyer Protocol). As a running example throughout this paper, we use the classic *2-buyer protocol* [12, 13], where two buyers intend to buy a book jointly from a seller. The first buyer sends the title of the book to the seller, who, in turn, sends a quote to both buyers. Then, the first buyer decides how much she wishes to contribute and informs the second buyer, who either pays the rest or cancels the transaction by informing the seller.

The three participants are connected through endpoints b_1, b_2, and s respectively. Each endpoint must be used according to its respective session type annotation, expressed as a CLL proposition:

$$b_1 : \mathbf{name} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes \mathbf{1} \quad b_2 : \mathbf{cost}^\perp \parr \mathbf{cost}^\perp \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1})$$
$$s : \mathbf{name}^\perp \parr \mathbf{cost} \otimes \mathbf{cost} \otimes ((\mathbf{addr}^\perp \parr \perp) \,\&\, \perp)$$

The typing above gives a precise description of how each endpoint has to act. For example, $\mathbf{name} \otimes \mathbf{cost}^\perp \parr \mathbf{cost} \otimes \mathbf{1}$ says that buyer b_1 must first send a value of type **name** (the title of the book), then receive a value of type **cost** (the price of the book), then send a value of type **cost** (the amount of money she wishes to contribute), and finally terminate.

Coherence will determine whether the three endpoints above are compatible by establishing for each output which endpoint should receive it. For example, coherence will say that the first output of type **name** at endpoint b_1 must be received by the input of type $\mathbf{name}^\perp$ at s. Then, the output from s of type **cost** can be paired with the input of type $\mathbf{cost}^\perp$ at either b_1 and b_2. And so on, precisely describing what the execution of the 2-buyer protocol should be. □

Types. Following the propositions-as-types approach, *types*, taken to be propositions (formulas) of CLL, are associated to names, denoting the way an endpoint must be used at runtime. Their formal syntax is given as:

$$A, B ::= \quad a \mid a^\perp \mid 1 \mid \perp \mid (A \otimes B) \mid (A \parr B) \mid (A \oplus B) \mid (A \,\&\, B) \mid !A \mid ?A$$

We briefly comment on their interpretation. There is a predefined, finite set of atoms a and their duals $a^\perp$, e.g., **name** and $\mathbf{name}^\perp$. Types $\mathbf{1}$ and $\perp$ denote an endpoint that must be closed by a last synchronisation. A type $A \otimes B$ is assigned to an endpoint that outputs a message of type A and then is used as B. Similarly, an endpoint of type $A \parr B$, receives a message of type A and then continues as B. Types $A \oplus B$ and $A \,\&\, B$ denote branching. The former is the type of an endpoint that may select to go left or right and continues as A or B, respectively. The latter is the type of an endpoint that offers two choices (left or right) and then, based on such choice, continues as A or B. Finally, $!A$ types an endpoint offering a service of type A, while $?A$ types an endpoint of a client invoking some service and behaving as A. Operators can be grouped in pairs of duals that reflect the input-output duality. As a consequence, standard duality $(\cdot)^\perp$ on types is inductively defined as:

$$(a^\perp)^\perp = a \quad \mathbf{1}^\perp = \perp \quad (A \otimes B)^\perp = A^\perp \parr B^\perp \quad (A \oplus B)^\perp = A^\perp \,\&\, B^\perp \quad (!A)^\perp = ?A^\perp$$

In the remainder, for any binary operator $\oslash, \odot \in \{\otimes, \parr, \oplus, \&\}$, we interpret $A \oslash B \odot C$ as $A \oslash (B \odot C)$. Moreover, both ? and ! have higher priority than $\oslash$.

Global Types. Following the standard multiparty session types approach [12, 13], we introduce the syntax of *global types* [5], the language for expressing the protocols that processes must follow when communicating. This is done by the following grammar, starting with a set $\mathcal{N} = \{x, y, z, \ldots\}$ of names, and used for denoting communication endpoints.[3]

$$G, H ::= \quad \tilde{x} \to y \mid \tilde{x} \to y(G).H \mid x \to \tilde{y}.\mathsf{case}(G, H) \mid !x \to \tilde{y}(G) \mid x \leftrightarrow y$$

[3] A list of endpoints $(x_1 \ldots x_n)$ can be abbreviated as $\tilde{x}$.

$$\frac{}{x \leftrightarrow y \vDash x : A, y : A^{\perp}}\ \mathrm{Ax} \qquad \frac{}{\tilde{x} \to y \vDash \{x_i : \mathbf{1}\}_i, y : \bot}\ \mathbf{1}\bot$$

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

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

$$\frac{G \vDash x : A, \{y_i : B_i\}_i}{!x \to \tilde{y}(G) \vDash x : ?A, \{y_i : !B_i\}_i}\ ?!$$

Fig. 1. Coherence

A global type is an "Alice-Bob"-notation for expressing the sequence of interactions that a session (protocol) must follow. The term $\tilde{x} \to y$ expresses a synchronisation between endpoints $x_1 \dots x_n$ and endpoint y, which gathers all the synchronisation messages from $\tilde{x}$. Global type $\tilde{x} \to y(G).H$ is also a gathering operation between $\tilde{x}$ and y, but the processes communicating over endpoints $\tilde{x}$ and y spawn together a new session with global type G before continuing the current one with protocol H. In the term $x \to \tilde{y}.\mathsf{case}(G, H)$, endpoint x broadcasts a choice to endpoints $\tilde{y}$: as a result, the session will follow either protocol G or H. The protocol expressed by $!x \to \tilde{y}(G)$ denotes the composition of a process providing a service that must be invoked through $\tilde{y}$. The term $x \leftrightarrow y$ is just a simple forwarder; it connects two endpoints related via standard duality.

Contexts and Judgements. *Coherence*, denoted by $\vDash$, is defined by the judgement $G \vDash \Delta$. The context Δ contains the types of the endpoints we wish to compose. A coherence proof is used to establish if the types in Δ are compatible, i.e., when the corresponding processes are composed, they will interact (according to protocol G) without raising an error. Formally, *basic contexts* are sets Δ of propositions labelled by endpoints: $\Delta ::= \cdot \mid \Delta, x : A$, where $x : A$ states that endpoint x has type A. Each endpoint occurs at most once.[4]

Global types are the proof terms for coherence, which is defined in Fig. 1. The rules reflect the concept of a generalisation of duality [5]: dual operators can be matched at different endpoints, establishing how communications must be done between them. The axiom rule is identical to that of CLL. Rule $\mathbf{1}\bot$ says that many processes willing to close a session are compatible with a single process waiting for them; as a global type we have the term $\tilde{x} \to y$. For multiplicatives, rule $\otimes\parr$ implements gathering: many processes are outputting an endpoint of type A_i, and another process is gathering such endpoints and establishing a new session; the corresponding global type is $\tilde{x} \to y(G).H$. Rule $\oplus\&$ is for branching where the global type $x \to \tilde{y}.\mathsf{case}(G, H)$ indicates that a process with endpoint x decides which branch to take and communicates that to $\tilde{y}$. Finally, in the exponential rule !? some client with endpoint x invokes many services at $\tilde{y}$.

[4] We use $?\Delta$ as shorthand for any $x_1 : ?A_1, \dots, x_n : ?A_n$ and use Δ for the corresponding set $x_1 : A_1, \dots, x_n : A_n$.

Example 2. Returning to Example 1, we can formally attest that the three endpoints b_1, b_2, and s are coherent and can be composed. Overall, a coherence proof can be summarized as a global type. In our example:

$$\begin{array}{c} b_1 \to s(n \leftrightarrow n').\quad s \to b_1(x_1 \leftrightarrow x_1').\quad s \to b_2(x_2 \leftrightarrow x_2').\quad b_1 \to b_2(y \leftrightarrow y').\\ b_2 \to s.\mathsf{case}\quad (\ b_2 \to s(a \leftrightarrow a').(b_1, b_2) \to s,\qquad (b_1, b_2) \to s) \end{array}$$

The global type above corresponds to the following derivation for $\vDash$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{G_6 = (b_1,b_2) \to s \vDash b_1 : \mathbf{1}, b_2 : \mathbf{1}, s : \bot}\,\mathbf{1}\bot}{\quad}\,\otimes\parr \qquad \dfrac{}{(b_1,b_2) \to s \vDash b_1 : \mathbf{1}, b_2 : \mathbf{1}, s : \bot}\,\mathbf{1}\bot}{\begin{array}{l} G_5 = b_2 \to s(b_2 \leftrightarrow s).G_6 \vDash \begin{array}{l} b_1 : \mathbf{1},\\ b_2 : \mathbf{addr} \otimes \mathbf{1},\\ s : \mathbf{addr}^\bot \parr \bot \end{array} \end{array}}\,\oplus\& \\ \hline G_4 = b_2 \to s.\mathsf{case}(G_5, (b_1,b_2) \to s) \vDash b_1 : \mathbf{1}, b_2 : (\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1}, s : (\mathbf{addr}^\bot \parr \bot) \,\&\, \bot \end{array}\,\otimes\parr}{\begin{array}{l} G_3 = b_1 \to b_2(b_1 \leftrightarrow b_2).\ G_4 \vDash \begin{array}{l} b_1 : \mathbf{cost} \otimes \mathbf{1}, b_2 : \mathbf{cost}^\bot \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1}),\\ s : ((\mathbf{addr}^\bot \parr \bot) \,\&\, \bot) \end{array} \end{array}}\,\otimes\parr}{\begin{array}{l} G_2 = s \to b_2(s \leftrightarrow b_2).G_3 \vDash \begin{array}{l} b_1 : \mathbf{cost} \otimes \mathbf{1}, b_2 : \mathbf{cost}^\bot \parr \mathbf{cost}^\bot \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1}),\\ s : \mathbf{cost} \otimes ((\mathbf{addr}^\bot \parr \bot) \,\&\, \bot) \end{array} \end{array}}\,\otimes\parr}{\begin{array}{l} G_1 = s \to b_1(s \leftrightarrow b_1).G_2 \vDash \begin{array}{l} b_1 : \mathbf{cost}^\bot \parr \mathbf{cost} \otimes \mathbf{1},\\ b_2 : \mathbf{cost}^\bot \parr \mathbf{cost}^\bot \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1}),\\ s : \mathbf{cost} \otimes \mathbf{cost} \otimes ((\mathbf{addr}^\bot \parr \bot) \,\&\, \bot) \end{array} \end{array}}\,\otimes\parr$$

$$\begin{array}{l} b_1 \to s(b_1 \leftrightarrow s).G_1 \vDash \begin{array}{l} b_1 : \mathbf{name} \otimes \mathbf{cost}^\bot \parr \mathbf{cost} \otimes \mathbf{1},\\ b_2 : \mathbf{cost}^\bot \parr \mathbf{cost}^\bot \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1}),\\ s : \mathbf{name}^\bot \parr \mathbf{cost} \otimes \mathbf{cost} \otimes ((\mathbf{addr}^\bot \parr \bot) \,\&\, \bot) \end{array} \end{array}$$

For clarity, we elided the left premisses in the applications of $\otimes\parr$ since, in this derivation, they are axiomatic, e.g., $b_1 \leftrightarrow s \vDash b_1 : \mathbf{name}, s : \mathbf{name}^\bot$. □

2.2 Process Terms

We use a language of *processes* to represent the forwarders that decide to whom messages, choices, and services should be delivered, expressing the communications enforced by coherence. For that, we introduce a standard process language which is a variant of the π-calculus [16] with specific communication primitives as usually done for session calculi. Moreover, given that the theory of this paper is based on the proposition-as-types correspondence with CLL, we adopt a syntax akin to that of Wadler [20, 21]:

$P, Q ::=$	$x \leftrightarrow y$	(link)	$(\boldsymbol{\nu} xy)\,(P \mid Q)$	(parallel)
	$x().P$	(wait)	$x[]$	(close)
	$x(y).P$	(input)	$x[y \triangleright P].Q$	(output)
	$x.\mathsf{case}(P, Q)$	(choice)	$x[\mathsf{inl}].P$	(left select)
			$x[\mathsf{inr}].P$	(right select)
	$?x[y].P$	(client request)	$!x(y).P$	(server accept)

We briefly comment on the various production of the grammar above. A link $x \leftrightarrow y$ is a binary forwarder, i.e., a process that forwards any communication between endpoint x and endpoint y. This yields a sort of equality relation on

names: it says that endpoints x and y are equivalent, and communicating something over x is like communicating it over y. The term $(\boldsymbol{\nu} xy)\,(P \mid Q)$ is used for composing processes: it is the parallel composition of P and Q that share a private connection through endpoints x and y. Parallel composition formally define compositionality of processes and will play a key role in deriving the semantics of synchronous forwarders through logic (cf. § 4). Note that we use endpoints instead of channels [19]. The difference is subtle: the restriction $(\boldsymbol{\nu} xy)$ connects the two endpoints x and y, instead of referring to the channel between them. The terms $x().P$ and $x[]$ handle synchronisation (no message passing); $x().P$ can be seen as an empty input on x, while $x[]$ terminates the execution of the process. The term $x[y \triangleright P].Q$ denotes a process that creates a fresh name y, spawns a new process P, and then continues as Q. The intuition behind this communication operation is that P uses y as an interface for dealing with the continuation of the dual primitive (denoted by term $x(y).R$, for some R). We observe that Wadler [20, 21] uses the syntax $x[y].(P \mid Q)$, but we believe that our version is more intuitive and gives a better explanation of why we require two different processes to follow after an output. However, our format is partially more restrictive, since y is forced to be bound in P (which Wadler enforces with typing). Also, note that output messages are always fresh, as for the internal π-calculus [17], hence the output term $x[y \triangleright P].Q$ is a compact version of the π-calculus term $(\nu y)\,\overline{x}y.(P \mid Q)$. Branching computations are handled by $x.\mathsf{case}(P, Q)$, $x[\mathsf{inl}].P$ and $x[\mathsf{inr}].P$. The former denotes a process offering two options (external choice) from which some other process can make a selection with $x[\mathsf{inl}].P$ or $x[\mathsf{inr}].P$ (internal choice). Finally, the term $!x(y).P$ denotes a persistently available service that can be invoked by $?x[z].Q$ at x which will spawn a new session to be handled by a copy of process P.

As shown by Wadler [20, 21], among all of the many process expressions one can write, CLL characterises the subset that is well-behaved, i.e. they satisfy deadlock freedom and session fidelity, and therefore interesting for our work.

2.3 Mapping Coherence into Arbiters

Carbone et al. [5] show that any coherence proof $G \vDash \Delta$ can be transformed into a corresponding CLL proof of $P \vdash_{\mathsf{CLL}} \Delta^{\perp}$, where P is called the *arbiter*. Whenever there is an output on some endpoint x, then the arbiter can input such message and then forward it to the receiver specified by coherence. For example, given a global type $x \to y(G).H$, we can build the arbiter $x'(u).y'[v \triangleright P].Q$ (for some fresh u and v) which inputs from x' (binary endpoint connected to x) and outputs on y' (binary endpoint connected to y), where inductively P and Q are the arbiters corresponding resp. to G and H. The translation of coherence proofs (hereby expressed as global types) into processes is reported in Fig. 2.

Example 3 (2-Buyer Protocol). Recall the 2-buyer protocol from Example 1, where two buyers try to make a joint decision whether to buy a book from

$$
\begin{array}{rcl}
\{\!|x \leftrightarrow y|\!\} & \stackrel{\text{def}}{=} & x' \leftrightarrow y' \\
\{\!|\tilde{x} \to y|\!\} & \stackrel{\text{def}}{=} & x'_1().\cdots x'_n().y'[] \\
\{\!|\tilde{x} \to y(G).H|\!\} & \stackrel{\text{def}}{=} & x'_1(u_1).\cdots x'_n(u_n).y'[v \triangleright \{\!|G|\!\}\{\tilde{u}/\tilde{x}', v/y'\}].\{\!|H|\!\} \\
\{\!|x \to \tilde{y}.\mathsf{case}(G,H)|\!\} & \stackrel{\text{def}}{=} & x.\mathsf{case}(y_1[\mathsf{inl}].\cdots y_n[\mathsf{inl}].\{\!|G|\!\}, y_1[\mathsf{inr}].\cdots y_n[\mathsf{inr}].\{\!|H|\!\}) \\
\{\!|!x \to \tilde{y}(G)|\!\} & \stackrel{\text{def}}{=} & !x(u).?y_1[v_1].\cdots ?y_n[v_n].\{\!|G|\!\}\{u/x, \tilde{v}/\tilde{y}\} \\
 & & \text{where } u, v, \tilde{u}, \tilde{v}, x', y', \tilde{x}' \text{and } \tilde{y}' \text{ are fresh}
\end{array}
$$

Fig. 2. Translation of Global Types into Processes [5]

a seller. The interactions enforced by the global type/coherence proof seen in Example 2 can be expressed by the arbiter process P_1 below:

$$
\begin{array}{l}
P_1 = b'_1(n).s'[n' \triangleright n \leftrightarrow n'].\ s'(x_1).b'_1[x'_1 \triangleright x_1 \leftrightarrow x'_1].\ s'(x_2).b'_2[x'_2 \triangleright x_2 \leftrightarrow x'_2]. \\
\qquad b'_1(y).b'_2[y' \triangleright y \leftrightarrow y'].\ b'_2.\mathsf{case}(\ s'[\mathsf{inl}].b'_2(a).s'[a' \triangleright a \leftrightarrow a'].Q_1\ ,\ s'[\mathsf{inr}].Q_1\ )
\end{array}
$$

Above, b'_1, b'_2, and s' are the endpoints connecting the forwarder P_1 to the endpoints of the two buyers and the seller (resp. b_1, b_2, and s). The process $Q_1 = b'_1().b'_2().s'[]$ closes all endpoints. □

We stress that an arbiter is nothing but a *forwarder*, transmitting messages between the composed peers. However, there are processes that are not exactly arbiters (not in the image of the translation in Fig. 2), but still forwarders and typable in CLL. For example, consider the global type $x \to y(G).z \to y(G').H$, the corresponding arbiter has the form $x'(u).y'[v \triangleright P].z'(s).y'[t \triangleright Q].R$. However, another well-typed process enforcing the protocol could be the process $x'(u).z'(s).y'[v \triangleright P].y'[t \triangleright Q].R$ or the process $z'(s).x'(u).y'[v \triangleright P].y'[t \triangleright Q].R$. In the next section, we define the class of synchronous forwarders which give a much larger class of processes that still correspond to coherence.

3 Synchronous Forwarders

The goal of this section is to present a type system that captures precisely a set of forwarder processes generalising arbiters. Following a proposition-as-types approach, we aim at a restriction of CLL whose type contexts are provable by coherence. Intuitively, we want such restriction to guarantee that messages cannot be opened/used, a received message is always forwarded, forwarded messages have always been previously received, and, the order of messages is preserved between any two endpoints. The latter is a key for abiding to coherence which precisely enforces in which order messages should be exchanged.

In this paper, we restrict our focus to a class of processes that is also synchronous, i.e., a forwarder is ready to receive a message on some endpoint x only after any previous message from x has been forwarded. This corresponds to thinking of a synchronous forwarder as a network queue of size one. The technical device used to enforce this behaviour is a one-size buffer for each endpoint

— while a buffer is full, the forwarder is blocked on that particular endpoint, and can only be unblocked by forwarding the message.

Contexts. To capture this one-size buffer mechanism, we need to introduce new notation. We continue to write $x : A$ for the typing of an unblocked endpoint, which we also call an *active* endpoint, but, when typing a one-size buffer, we write $[y : B]x : A$. Here $x : A$ refers to the blocked endpoint and $y : B$ to the message yet to be forwarded. The notion is also adapted to branching and exponentials. In summary, in this logic, contexts are formed as follows:

$$\Gamma ::= \Delta \quad | \quad \Gamma, [*] \quad | \quad \Gamma, [y : B]x : A \quad | \quad \Gamma, \mathcal{S}_l[\![\Delta]\!]x : A \quad | \quad \Gamma, \mathcal{S}_r[\![\Delta]\!]x : A \quad | \quad \Gamma, \mathcal{Q}[\![\Delta]\!]x : A$$

Δ is the context we defined in the previous section for coherence, i.e., $\Delta ::= \cdot \mid \Delta, x : A$. Intuitively, the extra ingredients in a Γ context are used for bookkeeping messages in-transit. And, we do that by *boxing*. For example, the element $[y : B]x : A$ expresses that some name y of type A has been received from endpoint x and must be later forwarded. Until that is done, endpoint x (that has type B) is frozen. Similarly, $[*]$, $\mathcal{S}_l[\![\Delta]\!]x : A$ and $\mathcal{S}_r[\![\Delta]\!]x : A$, and $\mathcal{Q}[\![\Delta]\!]x : A$ indicate that a request for closing a session, a branching request, or server invocation, respectively, has been received and must be forwarded. In the case of branching (additives) and servers (exponentials), the context Δ contains the endpoints we must forward to.

In synchronous forwarders, we can have several occurrences of each $\mathcal{S}_l/\mathcal{S}_r/\mathcal{Q}$ in a context, since boxes can occur multiple times, containing potentially different typed endpoints. However, this is not the case for $[*]$: it only acts as a flag, i.e., multiple occurrences are automatically contracted to a single one.

Notation. In the sequel, we silently use equivalences $\mathcal{X}[\![\cdot]\!] \equiv \cdot$ for any $\mathcal{X} \in \{\mathcal{S}_l, \mathcal{S}_r, \mathcal{Q}\}$. Also, we write $\mathcal{S}_\#$ for an unspecified $\mathcal{S}_l$ or $\mathcal{S}_r$. Out of convenience, we write $[\Delta_1]\Delta_2$ for $\Delta_1 = \{y_i : B_i\}$ and $\Delta_2 = \{x_i : A_i\}$ as a slight abuse of notation for the set $\{[y_i : B_i]x_i : A_i\}_i$ assuming it can be inferred from the context how the y_i's and the x_i's are paired. Moreover, we write $\oplus\Delta$ as a shorthand for any set $\{x_i : A_i \oplus B_i\}$ and similarily $?\Delta$ for $\{x_i : ?A_i\}$.

Judgement and Rules. A judgement, denoted by $P \vdash \Gamma$, captures precisely those forwarding processes P that connect the endpoints represented in Γ, buffer at most one message at each endpoint, and preserve order.

In Fig. 3, we report the rules for typing processes: they correspond to the CLL sequent calculus enhanced with process terms (using endpoints [5]), but with some extra restrictions for characterising forwarders. The defining characteristic of $\vdash$ is that it uses $\bot$, $\parr$, $\&$, $!$ as a buffering mechanism for respectively endpoint messages (units and multiplicatives), choices (additives), and requests (exponentials) in order to render them temporarily inaccessible to any other rule. The only way to awaken them (render them accessible again) is to forward to another endpoint, external to the forwarder using $\mathbf{1}$, $\otimes$, $\oplus_\#$, and $?$.

Rule Ax is the axiom of CLL: it denotes a process that interfaces two endpoints of dual type. Rules $\mathbf{1}$ and $\bot$ type forwarding of empty messages. In $x().P$, the typing makes sure that after an empty message is received (of type $\bot$), it is

$$\frac{}{x \leftrightarrow y \vdash x : A^{\perp}, y : A}\ \text{Ax} \qquad \frac{P \vdash \Gamma, [*]}{x().P \vdash \Gamma, x : \bot}\ \bot \qquad \frac{}{x[] \vdash x : \mathbf{1}, [*]}\ \mathbf{1}$$

$$\frac{P \vdash \Delta_1, y : A \quad Q \vdash \Gamma, \Delta_2, x : B}{x[y \triangleright P].Q \vdash \Gamma, [\Delta_1]\Delta_2, x : A \otimes B}\ \otimes \qquad \frac{P \vdash \Gamma, [y : A]x : B}{x(y).P \vdash \Gamma, x : A \parr B}\ \parr$$

$$\frac{P \vdash \Gamma, \mathcal{S}_l[\![\oplus\Delta]\!]x : A \quad Q \vdash \Gamma, \mathcal{S}_r[\![\oplus\Delta]\!]x : B}{x.\mathsf{case}(P, Q) \vdash \Gamma, \oplus\Delta, x : A \,\&\, B}\ \&$$

$$\frac{P \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A}{x[\mathsf{inl}].P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C}\ \oplus_l \qquad \frac{P \vdash \Gamma, \mathcal{S}_r[\![\Delta]\!]z : C, x : B}{x[\mathsf{inr}].P \vdash \Gamma, \mathcal{S}_r[\![\Delta, x : A \oplus B]\!]z : C}\ \oplus_r$$

$$\frac{P \vdash \mathcal{Q}[\![?\Delta]\!]y : A}{!x(y).P \vdash ?\Delta, x : !A}\ ! \qquad \frac{P \vdash \Gamma, \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P \vdash \Gamma, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C}\ ?$$

Fig. 3. Synchronous forwarder logic

forwarded by adding to the typing context for P the object $[*]$ which will make sure that eventually rule $\mathbf{1}$ is used. In fact, rule $\mathbf{1}$ is applicable only if there is $[*]$ in the context, i.e., at least one $\bot$ rule has been encountered before. Note that this corresponds to the gathering operation for units enforced by coherence. Rule $\parr$ types reception of y over x with type $A \parr B$. The received name of type A cannot be used but must be forwarded, therefore it is wrapped as $[y : A]x : B$ for the typing of P. Endpoint x is temporarily blocked, until y is finally forwarded. This is done by rule $\otimes$, which collects the received $[\Delta_1]$ and spawns a new forwarder P of type $\Delta_1, y : A$, freeing $x : B$. As for units, multiplicatives implement gathering, i.e., we forward many sends to a single receiver. Note that, since contexts are sets, we explicitly require $\mathsf{fn}(\Delta)$ and $\mathsf{fn}(\Gamma)$ to be different.

Rules $\oplus_{\#}$ (where $\# \in \{l, r\}$) and $\&$ type branching processes. Unlike the case of units and multiplicatives which use a gathering communication mechanism (many-to-one), additives (and later exponentials) use broadcasting (one-to-many). This is a choice that follows directly from the logic principles and the way each operator are interpreted. Rule $\&$ types the process $x.\mathsf{case}(P, Q)$: in branches P and Q the selected choice, left and right respectively, must be forwarded to some other endpoint. This is enforced by $\mathcal{S}_l[\![\Delta]\!]x : A$ and $\mathcal{S}_r[\![\Delta]\!]x : A$. Besides containing the information on which branch must be forwarded ($\mathcal{S}_l$ or $\mathcal{S}_r$), they also block $x : A$ until the information signaling to pick the left (or right) branch has been forwarded to all active endpoints in $\Delta = \{x_i : A_i \oplus B_i\}$. Similarly, for exponentials, endpoint y is blocked until all other endpoints in $\Delta = \{x_i : ?B\}$ agree (in any order) that A may proceed.

We conclude this subsection with the straightforward result, that embeds $\vdash$ into $\vdash_{\mathsf{CLL}}$, where $\vdash_{\mathsf{CLL}}$ is the typing sequent based on CLL [5].

Proposition 4. *If* $P \vdash \Gamma$*, then* $P \vdash_{\mathsf{CLL}} \ulcorner\Gamma\urcorner$*; embedding* $\ulcorner\cdot\urcorner$ *being defined as:*

$$\ulcorner\Delta\urcorner = \Delta \qquad \ulcorner\Gamma, [*]\urcorner = \ulcorner\Gamma\urcorner$$
$$\ulcorner\Gamma, [y:B]x:A\urcorner = \ulcorner\Gamma\urcorner, y:B, x:A \qquad \ulcorner\Gamma, \mathcal{X}[\![\Delta]\!]x:A\urcorner = \ulcorner\Gamma\urcorner, \Delta, x:A$$

4 Semantics via Cut Elimination

We now turn to the formal semantics of forwarders. Our goal is two-fold: provide a semantics for forwarders and have a way of composing them safely. As a consequence, we obtain a methodology for connecting two forwarders together, define how they can internally communicate, and be sure that after composition the obtained process is still a forwarder. In order to illustrate how this can be used, we begin with of an extension of the 2-buyer protocol example.

Example 5. We extend the 2-buyer example with a second concurrent system. Assume that the second buyer wishes to delegate the decision to buy to two colleagues and only if they both agree, the book will be bought. Here again, we can use a forwarder P_2 to orchestrate the communication between the two colleagues and the second buyer: it forwards the price to the first colleague, and the first buyer's contribution (obtained through the second buyer) to the second colleague. Then, it relays decisions from the two colleagues to the second buyer.

We write this forwarder as process P_2 below, where b_2'', c_1, and c_2 are the endpoints connecting P_2 to the second buyer and the two colleagues. Note that P_1 is connected to the second buyer through endpoint b_2' instead (see Example 3).

$$P_2 = b_2''(y_1).c_1[y_1' \triangleright y_1 \leftrightarrow y_1'].\ b_2''(y_2).c_2[y_2' \triangleright y_2 \leftrightarrow y_2'].\ c_1.\mathsf{case}\left(\begin{array}{l} c_2[\mathsf{inl}].c_2.\mathsf{case}(\ b_2''[\mathsf{inl}].Q_2,\ b_2''[\mathsf{inr}].Q_2\ ), \\ c_2[\mathsf{inr}].c_2.\mathsf{case}(\ b_2''[\mathsf{inl}].Q_2,\ b_2''[\mathsf{inr}].Q_2\ ) \end{array}\right)$$

Process $Q_2 = c_1().c_2().b_2''[]$ closes all endpoints. Intuitively, we wish to combine P_1 and P_2 into a new forwarder P that orchestrates the communications between the seller, the first buyer, and the colleagues, bypassing the second buyer. □

Is a standard cut rule enough? In a propositions-as-types approach, the semantics of processes is obtained from the reductions of proofs (which correspond to processes) given by the *cut elimination* process. In CLL, a cut rule allows to compose two compatible proofs, establishing their compatibility based on duality. Cut elimination is then a procedure for eliminating any occurrence of the cut rule in a proof (also known as normalisation) and it is usually done in small steps called cut reductions. In our process terms, a cut rule corresponds to connecting two endpoints from two parallel processes; and, cut reductions correspond to reductions of processes, thus yielding their semantics.

Ideally, since synchronous forwarders are embedded into CLL, we could use the CLL cut rule to compose them. However, synchronous forwarder have an extended typing context containing extra information on what has to be forwarded. Therefore, the CLL cut rule is not sufficient. In order to understand why, we start

$$\frac{P \vdash \Gamma_1, x : A \quad Q \vdash \Gamma_2, y : A^{\perp}}{(\boldsymbol{\nu} xy)\,(P \mid Q) \vdash \Gamma_1, \Gamma_2}\ \textsc{Cut}$$

$$\frac{P \vdash \Delta_1, u : A \quad Q \vdash \Delta_2, \Gamma_1, x : B \quad R \vdash \Gamma_2, [v : A^{\perp}]y : B^{\perp}}{(\boldsymbol{\nu} x[]y)\,(Q \mid (\boldsymbol{\nu} u[v])\,(P \mid R)) \vdash [\Delta_1]\Delta_2, \Gamma_1, \Gamma_2}\ \textsc{Cut}_{\otimes\parr}$$

$$\frac{P \vdash \Gamma_1, \mathcal{S}_{\#}[\![\Delta_1, x : A \oplus B]\!]z : C \quad Q \vdash \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]y : A^{\perp} \quad R \vdash \Gamma_2, \mathcal{S}_r[\![\Delta_2]\!]y : B^{\perp}}{(\boldsymbol{\nu} xy)\,(P \mid y.\mathsf{case}(Q, R)) \vdash \Gamma_1, \Gamma_2, \mathcal{S}_{\#}[\![\Delta_1, \Delta_2]\!]z : C}\ \textsc{Cut}_{\oplus\&_{1\#}}$$

$$\frac{P \vdash \Gamma_1, \mathcal{S}_{\#}[\![\Delta_1]\!]z : C, x : A \quad Q \vdash \Gamma_2, \mathcal{S}_{\#}[\![\Delta_2]\!]y : A^{\perp}}{(\boldsymbol{\nu} xy)\,(P \mid Q) \vdash \Gamma_1, \Gamma_2, \mathcal{S}_{\#}[\![\Delta_1, \Delta_2]\!]z : C}\ \textsc{Cut}_{\oplus\&_{2\#}}$$

$$\frac{P \vdash \Gamma, \mathcal{Q}[\![\Delta_1, x : ?A]\!]z : C \quad Q \vdash \mathcal{Q}[\![\Delta_2]\!]v : A^{\perp}}{(\boldsymbol{\nu} xy)\,(P \mid !y(v).Q) \vdash \Gamma, \mathcal{Q}[\![\Delta_1, ?\Delta_2]\!]z : C}\ \textsc{Cut}_{!?_1}$$

$$\frac{P \vdash \Gamma_1, \mathcal{Q}[\![\Delta_1]\!]z : C, x : A \quad Q \vdash \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]y : A^{\perp}}{(\boldsymbol{\nu} xy)\,(P \mid Q) \vdash \Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]z : C}\ \textsc{Cut}_{!?_2}$$

Fig. 4. Cut rules for synchronous forwarders

from defining the core rule Cut for synchronous forwarders which, as in CLL, connects two endpoints with dual types:

$$\frac{P \vdash \Gamma_1, x : A \quad Q \vdash \Gamma_2, y : A^{\perp}}{(\boldsymbol{\nu} xy)\,(P \mid Q) \vdash \Gamma_1, \Gamma_2}\ \textsc{Cut}$$

We can read the rule above as follows: we can compose the two processes P and Q willing to communicate on endpoints x and y because their types are dual. And, their parallel composition yields a new process where both x and y have disappeared, since now they have formed an internal channel. Both processes implement input and output operations over the two connected endpoints, but when the two processes start communicating, their state will change into something that may contain messages in transit awaiting to be forwarded. This means that the cut rule may have to involve some of the special elements in a context Γ. In order to deal with such extra constraints on contexts, we require special cut rules that deal with intermediary computation (referred to as *runtime* rules) and associated to new syntactic terms (runtime syntax), depending on the proposition that is being cut (and the corresponding communication operation). Such rules deal with blocked endpoints such as the boxed judgments $[x : A]y : B$, $\mathcal{S}_l[\![\Delta]\!]x : A$, $\mathcal{S}_r[\![\Delta]\!]x : A$, and $\mathcal{Q}[\![\Delta]\!]x : A$. As a consequence, we define six variations of the cut rule for synchronous forwarders, which are reported in Fig. 4.

We discuss these rules by considering the four possible cases associated to each pair of dual modalities (units, multiplicatives, additives, and exponentials). Later, we will show that the rule Cut and all runtime cut rules are admissible for synchronous forwarders, i.e., we can always create a forwarder that has no par-

allel composition yet behaves as the composition of the two original forwarders. We use $\longrightarrow_\beta$ to denote a rewriting of a proof (and the associated process) that reduces the complexity of the type A in Cut.

Units. This case does not require any additional rule. We can always remove a cut between an application of rule **1** and an application of rule $\bot$:

$$\dfrac{\dfrac{}{x[] \vdash x : \mathbf{1}, [*]}\ \mathbf{1} \quad \dfrac{P \vdash [*], \Gamma}{y().P \vdash y : \bot, \Gamma}\ \bot}{(\boldsymbol{\nu} xy)\,(x[] \mid y().P) \vdash [*], \Gamma}\ \textsc{Cut} \qquad \longrightarrow_\beta \qquad P \vdash [*], \Gamma$$

For processes, we obtain the reduction $(\boldsymbol{\nu} xy)\,(x[] \mid y().P) \longrightarrow_\beta P$ which shows how a final synchronisation closes the connection between endpoints x and y.

Multiplicatives. In the case of $\otimes$ and $\parr$, we get the following principal case:

$$\dfrac{\dfrac{P \vdash \Delta_1, u : A \quad Q \vdash \Delta_2, \Gamma_1, x : B}{x[u \triangleright P].Q \vdash [\Delta_1]\Delta_2, \Gamma_1, x : A \otimes B}\ \otimes \quad \dfrac{R \vdash \Gamma_2, [v : A^\bot]y : B^\bot}{y(v).R \vdash \Gamma_2, y : A^\bot \parr B^\bot}\ \parr}{(\boldsymbol{\nu} xy)\,(x[u \triangleright P].Q \mid y(v).R) \vdash [\Delta_1]\Delta_2, \Gamma_1, \Gamma_2}\ \textsc{Cut}$$

What we would usually do in the proof of cut-elimination for CLL is to replace Cut by two cuts on A and B, respectively. This, however, is not possible in this case, as $v : A^\bot$ is pushed into a buffer that is linked to its sending endpoint y, in the configuration $[v : A^\bot]y : B^\bot$. The two need to remain linked until the message v is forwarded and channel y becomes active again.

This requires us to introduce a new runtime cut rule which handles A and B *at the same time*, despite them splitting into two distinct communications. Yet, they are intertwined and the cut rule needs to capture the *three premisses* together. This is achieved by rule $\textsc{Cut}_{\otimes\parr}$ in Fig. 4. Note that we have also adopted a runtime process syntax for the new rules, yielding $(\boldsymbol{\nu} x[]y)\,(Q \mid (\boldsymbol{\nu} u[v])\,(P \mid R))$ as a new term: the box $[]$ in front of y signals that endpoint y is blocked, while $[v]$ means that v is a message in transit that must be forwarded.

The location of the corresponding sub-derivation ending in $A^\bot$ may be deep in the derivation of R and can only be retrieved by applying several *commuting conversions*. Commuting conversions, denoted by $\longrightarrow_\kappa$, are proof transformations that do not change the size of a proof nor the size of a proposition (type), but only perform rule permutations. We report a full list of commuting conversions (as processes) at the end of this section, in Fig. 6.

Returning to our case, once $\textsc{Cut}_{\otimes\parr}$ meets the send operation ($\otimes$) that frees y, by forwarding $v : A^\bot$ and spawning a new process S, the communication can continue with two basic Cut rules:

$$\dfrac{P \vdash \Delta_1, u : A \qquad Q \vdash \begin{array}{l}\Delta_2, \Gamma_1,\\ x : B\end{array} \qquad \dfrac{S \vdash \begin{array}{l}\Delta_3, v : A^\bot,\\ w : C\end{array} \qquad T \vdash \begin{array}{l}\Gamma_2, y : B^\bot,\\ \Delta_4, z : D\end{array}}{w[z \triangleright S].T \vdash \begin{array}{l}\Gamma_2, [v : A^\bot]y : B^\bot,\\ [\Delta_3]\Delta_4, z : C \otimes D\end{array}}\ \otimes}{(\boldsymbol{\nu} x[]y)\,(Q \mid (\boldsymbol{\nu} u[v])\,(P \mid w[z \triangleright S].T)) \vdash [\Delta_1]\Delta_2, \Gamma_1, \Gamma_2, [\Delta_3]\Delta_4, z : C \otimes D}\ \textsc{Cut}_{\otimes\parr}$$

$$\longrightarrow_\kappa$$

$$\dfrac{\dfrac{P \vdash \Delta_1, u : A \quad S \vdash \Delta_3, v : A^\perp, w : C}{(\boldsymbol{\nu} uv)\,(P \mid S) \vdash \Delta_1, \Delta_3, w : C}\ \textsc{Cut} \qquad \dfrac{Q \vdash \Delta_2, \Gamma_1, x : B \quad T \vdash \Gamma_2, y : B^\perp, \Delta_4, z : D}{(\boldsymbol{\nu} xy)\,(Q \mid T) \vdash \Delta_2, \Gamma_1, \Gamma_2, \Delta_4, z : D}\ \textsc{Cut}}{z[w \triangleright (\boldsymbol{\nu} uv)\,(P \mid S)].(\boldsymbol{\nu} xy)\,(Q \mid T) \vdash [\Delta_1]\Delta_2, \Gamma_1, \Gamma_2, [\Delta_3]\Delta_4, z : C \otimes D}\ \otimes$$

Additives. What would a key reduction look like for the additive connectives in this system? That is, we need to consider the reductions available to a cut with premisses $P \vdash \Gamma_1, x : A \oplus B$ and $Q \vdash \Gamma_2, y : A^\perp \,\&\, B^\perp$. By using commuting conversions, it is always possible to reach the point where the right premiss (that of Q) ends with the rule for which the cut formula is *principal*, that is $Q = y.\mathsf{case}(Q_1, Q_2)$, $\Gamma_2 = \Gamma_2', \oplus\Delta_2$, and the right branch ends as:

$$\dfrac{Q_1 \vdash \Gamma_2', \mathcal{S}_l[\![\oplus\Delta_2]\!]y : A^\perp \quad Q_2 \vdash \Gamma_2', \mathcal{S}_r[\![\oplus\Delta_2]\!]y : B^\perp}{y.\mathsf{case}(Q_1, Q_2) \vdash \Gamma_2', \oplus\Delta_2, y : A^\perp \,\&\, B^\perp}\ \&$$

On the left branch however, reaching the rule for which the cut-formula is principal must be done in two steps as $A \oplus B$ can only be principal after having been chosen for selection. This means that we first need to consider the point at which $P = z.\mathsf{case}(P_1, P_2)$, $\Gamma_1 = \Gamma_1', \oplus\Delta_1, z : C \,\&\, D$ and the left branch of the cut is of the shape:

$$\dfrac{P_1 \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, x : A \oplus B]\!]z : C \quad P_2 \vdash \Gamma_1', \mathcal{S}_r[\![\oplus\Delta_1, x : A \oplus B]\!]z : D}{z.\mathsf{case}(P_1, P_2) \vdash \Gamma_1', \oplus\Delta_1, x : A \oplus B, z : C \,\&\, D}\ \&$$

That places $A \oplus B$ at least in a position to become, higher in the proof, principal. We are therefore in a special configuration of the cut between additives:

$$\dfrac{z.\mathsf{case}(P_1, P_2) \vdash \Gamma_1', \oplus\Delta_1, x : A \oplus B, z : C \,\&\, D \quad y.\mathsf{case}(Q_1, Q_2) \vdash \Gamma_2', \oplus\Delta_2, y : A^\perp \,\&\, B^\perp}{(\boldsymbol{\nu} xy)\,(z.\mathsf{case}(P_1, P_2) \mid y.\mathsf{case}(Q_1, Q_2)) \vdash \Gamma_1', \oplus\Delta_1, z : C \,\&\, D, \Gamma_2', \oplus\Delta_2}$$

We introduce the *runtime* cut $\textsc{Cut}_{\oplus\&1l}$ that puts together the premisses as:

$$\dfrac{P_1 \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, x : A \oplus B]\!]z : C \quad Q_1 \vdash \Gamma_2', \mathcal{S}_l[\![\oplus\Delta_2]\!]y : A^\perp \quad Q_2 \vdash \Gamma_2', \mathcal{S}_r[\![\oplus\Delta_2]\!]y : B^\perp}{R_1 = (\boldsymbol{\nu} xy)\,(P_1 \mid y.\mathsf{case}(Q_1, Q_2)) \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, \oplus\Delta_2]\!]z : C, \Gamma_2'}$$

Similarly, we introduce $\textsc{Cut}_{\oplus\&1r}$ to get $R_2 = (\boldsymbol{\nu} xy)\,(P_2 \mid y.\mathsf{case}(Q_1, Q_2))$ on the right. The two cuts can then be re-combined as follows:

$$\dfrac{R_1 \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, \oplus\Delta_2]\!]z : C, \Gamma_2' \quad R_2 \vdash \Gamma_1', \mathcal{S}_r[\![\oplus\Delta_1, \oplus\Delta_2]\!]z : D, \Gamma_2'}{z.\mathsf{case}(R_1, R_2) \vdash \Gamma_1', \oplus\Delta_1, z : C \,\&\, D, \oplus\Delta_2, \Gamma_2'}\ \&$$

Indeed, this is a special case of a general equivalence (as in CLL) obtained by commuting the rule applied to the side formula $C \,\&\, D$ below the cut:

$$(\boldsymbol{\nu} xy)\,(z.\mathsf{case}(P_1, P_2) \mid Q) \quad \longrightarrow_\kappa \quad z.\mathsf{case}((\boldsymbol{\nu} xy)\,(P_1 \mid Q), (\boldsymbol{\nu} xy)\,(P_2 \mid Q))$$

but in the case where $Q = y.\mathsf{case}(Q_1, Q_2)$.

Then, these $\textsc{Cut}_{\oplus\&_{1\#}}$ will be able to reduce further when meeting the corresponding $\oplus_{\#}$-rule making the cut-formula principal (inside its box), as for instance when $P_1 = x[\mathsf{inl}].P'$:

$$\dfrac{\dfrac{P' \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, x : A}{x[\mathsf{inl}].P' \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, x : A \oplus B]\!]z : C}\,\oplus_1 \quad \begin{array}{l} Q_1 \vdash \Gamma_2', \mathcal{S}_l[\![\oplus\Delta_2]\!]y : A^\perp \\ Q_2 \vdash \Gamma_2', \mathcal{S}_r[\![\oplus\Delta_2]\!]y : B^\perp \end{array}}{(\boldsymbol{\nu} xy)\,(x[\mathsf{inl}].P' \;\mid\; y.\mathsf{case}(Q_1, Q_2)) \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, \oplus\Delta_2]\!]z : C, \Gamma_2'}\,\textsc{Cut}_{\oplus\&_{1l}}$$

$$\longrightarrow_\beta \quad \dfrac{P' \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, x : A \quad Q_1 \vdash \Gamma_2', \mathcal{S}_l[\![\oplus\Delta_2]\!]y : A^\perp}{(\boldsymbol{\nu} xy)\,(P' \;\mid\; Q_1) \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, \oplus\Delta_2]\!]z : C, \Gamma_2'}\,\textsc{Cut}_{\oplus\&_{2l}}$$

In terms of processes, the above models the reduction:

$$(\boldsymbol{\nu} xy)\,(x[\mathsf{inl}].P' \;\mid\; y.\mathsf{case}(Q_1, Q_2)) \longrightarrow_\beta (\boldsymbol{\nu} xy)\,(P' \;\mid\; Q_1)$$

However, this reduction introduces again a new rule $\textsc{Cut}_{\oplus\&_{2\#}}$ since the cut formula on the right branch is blocked until the selection phase is fully over, i.e., until $\oplus\Delta_2$ becomes empty and y becomes active again. This would happen in the following situation, where $Q_1 = w[\mathsf{inl}].Q'$, and the communication then returns to a general $\textsc{Cut}$:

$$\dfrac{\dfrac{P' \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, x : A \quad \dfrac{Q' \vdash \Gamma_2', y : A^\perp, w : B}{w[\mathsf{inl}].Q' \vdash \Gamma_2', \mathcal{S}_l[\![w : B \oplus D]\!]y : A^\perp}\,\oplus_1}{(\boldsymbol{\nu} xy)\,(P' \;\mid\; w[\mathsf{inl}].Q') \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, w : B \oplus D]\!]z : C, \Gamma_2'}\,\textsc{Cut}_{\oplus\&_{2l}}$$

$$\longrightarrow_\kappa \quad \dfrac{\dfrac{P' \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, x : A \quad Q' \vdash \Gamma_2', y : A^\perp, w : B}{(\boldsymbol{\nu} xy)\,(P' \;\mid\; Q') \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, \Gamma_2', w : B}\,\textsc{Cut}}{w[\mathsf{inl}].(\boldsymbol{\nu} xy)\,(P' \;\mid\; Q') \vdash \Gamma_1', \mathcal{S}_l[\![\oplus\Delta_1, w : B \oplus D]\!]z : C, \Gamma_2'}\,\oplus_1$$

Note that here as well this equivalence is not specific to this case; it happens whevener an $\oplus$-rule is permuted with a cut, but in this particular configuration it allows $\textsc{Cut}_{\oplus\&_{2l}}$ to turn into a $\textsc{Cut}$ again.

Exponentials. In the case of exponentials, we start from an application of rule $\textsc{Cut}$ whose premisses are $P \vdash \Gamma_1, x : ?A$ and $Q \vdash \Gamma_2, y : !A$. As for the additives, it is possible to reach a point where the right branch ends with the rule for which $!A$ is principal. And also in this case, we need to use a two-step strategy in the left branch. First, we observe that a normal cut between $?A$ and $!A$ implies that the proof for $?A$ must contain a !-rule somewhere (which will box it and only then can a ?-rule be applied). Because of that, we must have a case for $\textsc{Cut}$ which will transform it into the runtime $\textsc{Cut}_{!?_1}$:

$$\dfrac{\dfrac{P_1 \vdash \mathcal{Q}[\![?\Delta_1, x : ?A]\!]s : B}{!r(s).P_1 \vdash ?\Delta_1, x : ?A, r : !B}\,! \quad \dfrac{Q_1 \vdash \mathcal{Q}[\![?\Delta_2]\!]z : A^\perp}{!y(z).Q_1 \vdash ?\Delta_2, y : !A^\perp}\,!}{(\boldsymbol{\nu} xy)\,(!r(s).P_1 \;\mid\; !y(z).Q_1) \vdash ?\Delta_1, ?\Delta_2, r : !B}\,\textsc{Cut}$$

$$\longrightarrow_\kappa \quad \dfrac{\dfrac{P_1 \vdash \mathcal{Q}[\![?\Delta_1, x : ?A]\!]s : B \quad Q_1 \vdash \mathcal{Q}[\![?\Delta_2]\!]z : A^\perp}{(\boldsymbol{\nu} xy)\,(P_1 \;\mid\; !y(z).Q_1) \vdash \mathcal{Q}[\![?\Delta_1, ?\Delta_2]\!]s : B}\,\textsc{Cut}_{!?_1}}{!r(s).(\boldsymbol{\nu} xy)\,(P_1 \;\mid\; !y(z).Q_1) \vdash ?\Delta_1, ?\Delta_2, r : !B}\,!$$

Structural equivalence

$$
\begin{array}{lcll}
x \leftrightarrow y & \equiv & y \leftrightarrow x & \\
(\boldsymbol{\nu} yx)\,(Q \mid P) & \equiv & (\boldsymbol{\nu} xy)\,(P \mid Q) & \\
(\boldsymbol{\nu} wz)\,(P \mid (\boldsymbol{\nu} xy)\,(Q \mid R)) & \equiv & (\boldsymbol{\nu} xy)\,((\boldsymbol{\nu} wz)\,(P \mid Q) \mid R) & x, z \in \mathsf{fn}(Q) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,((\boldsymbol{\nu} uv)\,(P_1 \mid P_2) \mid R)) & & & \\
\qquad \equiv & (\boldsymbol{\nu} uv)\,(P_1 \mid (\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P_2 \mid R))) & & x, v \in \mathsf{fn}(P_2) \\
(\boldsymbol{\nu} y[]z)\,((\boldsymbol{\nu} uv)\,(Q_1 \mid Q_2) \mid (\boldsymbol{\nu} x[w])\,(P \mid R)) & & & \\
\qquad \equiv & (\boldsymbol{\nu} uv)\,(Q_1 \mid (\boldsymbol{\nu} y[]z)\,(Q_2 \mid (\boldsymbol{\nu} x[w])\,(P \mid R))) & & y, v \in \mathsf{fn}(Q_2) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid (\boldsymbol{\nu} uv)\,(R_1 \mid R_2))) & & & \\
\qquad \equiv & (\boldsymbol{\nu} uv)\,(R_1 \mid (\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid R_2))) & & z, v \in \mathsf{fn}(R_2)
\end{array}
$$

Key Reductions (β)

$$
\begin{array}{lll}
(\boldsymbol{\nu} xy)\,(x \leftrightarrow w \mid Q) & \longrightarrow_\beta & Q\{w/y\} \\
(\boldsymbol{\nu} xy)\,(x[] \mid y().P) & \longrightarrow_\beta & P \\
(\boldsymbol{\nu} xy)\,(x[u \triangleright P].Q \mid y(v).R) & \longrightarrow_\beta & (\boldsymbol{\nu} x[]y)\,(Q \mid (\boldsymbol{\nu} u[v])\,(P \mid R)) \\
(\boldsymbol{\nu} xy)\,(x[\mathsf{inl}].P \mid y.\mathsf{case}(Q, R)) & \longrightarrow_\beta & (\boldsymbol{\nu} xy)\,(P \mid Q) \\
(\boldsymbol{\nu} xy)\,(x[\mathsf{inr}].P \mid y.\mathsf{case}(Q, R)) & \longrightarrow_\beta & (\boldsymbol{\nu} xy)\,(P \mid R) \\
(\boldsymbol{\nu} xy)\,(?x[u].Q \mid !y(v).P) & \longrightarrow_\beta & (\boldsymbol{\nu} uv)\,(P \mid Q)
\end{array}
$$

Fig. 5. Semantics of Synchronous Forwarders: Equivalences and Reductions

We can now observe a key reduction for ! and ? when the corresponding ?-rule applies to the left premiss of $\textsc{Cut}_{?!_1}$:

$$
\dfrac{\dfrac{P_2 \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1]\!]s : B, w : A}{?x[w].P_2 \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1, x : ?A]\!]s : B}\ ? \qquad Q_1 \vdash \mathcal{Q}[\![?\Delta_2]\!]z : A^\perp}{(\boldsymbol{\nu} xy)\,(?x[w].P_2 \mid !y(z).Q_1) \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1, ?\Delta_2]\!]s : B}\ \textsc{Cut}_{!?_1}
$$

$$
\longrightarrow_\beta \quad \dfrac{P_2 \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1]\!]s : B, w : A \quad Q_1 \vdash \mathcal{Q}[\![?\Delta_2]\!]z : A^\perp}{(\boldsymbol{\nu} wz)\,(P_2 \mid Q_1) \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1, ?\Delta_2]\!]s : B}\ \textsc{Cut}_{!?_2}
$$

This requires to introduce $\textsc{Cut}_{?!_2}$ as the formula $A^\perp$ on the right premiss is blocked by the boxed Δ_2. Similarly to additives, we can now push the cut up on the right premise until we can empty the box in front of $A^\perp$. This is done by:

$$
\dfrac{P_2 \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1]\!]s : B, w : A \quad \dfrac{Q_2 \vdash \Gamma_2, z : A^\perp, v : C}{?u[v].Q_2 \vdash \Gamma_2, \mathcal{Q}[\![u : ?C]\!]z : A^\perp}\ ?}{(\boldsymbol{\nu} wz)\,(P_2 \mid ?u[v].Q_2) \vdash \Gamma_1, \Gamma_2, \mathcal{Q}[\![?\Delta_1, u : ?C]\!]s : B}\ \textsc{Cut}_{?!_2}
$$

$$
\longrightarrow_\kappa \quad \dfrac{\dfrac{P_2 \vdash \Gamma_1, \mathcal{Q}[\![?\Delta_1]\!]s : B, w : A \quad Q_2 \vdash \Gamma_2, z : A^\perp, v : C}{(\boldsymbol{\nu} wz)\,(P_2 \mid Q_2) \vdash \Gamma_1, \Gamma_2, \mathcal{Q}[\![?\Delta_1]\!]s : B, v : C}\ \textsc{Cut}}{?u[v].(\boldsymbol{\nu} wz)\,(P_2 \mid Q_2) \vdash \Gamma_1, \Gamma_2, \mathcal{Q}[\![?\Delta_1, u : ?C]\!]z : B}\ ?
$$

Results and Semantics. In Fig. 5 and Fig. 6, we report the reductions and structural transformation derived for synchronous forwarders. The collection of rules defining essential reductions and commuting conversions are sound and complete, which is summarized by the following cut admissibility theorem.

Theorem 6 (Cut Admissibility). *The cut-rules* $\textsc{Cut}$, $\textsc{Cut}_{\otimes\parr}$, $\textsc{Cut}_{\&\oplus_{1\#}}$, $\textsc{Cut}_{\&\oplus_{2\#}}$, $\textsc{Cut}_{!?_1}$, $\textsc{Cut}_{!?_2}$ *are admissible rules in synchronous forwarder logic.*

$$
\begin{array}{lll}
(\boldsymbol{\nu} xy)\,(u().P \mid Q) & \longrightarrow_\kappa & u().(\boldsymbol{\nu} xy)\,(P \mid Q) \\
(\boldsymbol{\nu} xy)\,(u[v \triangleright P].Q \mid R) & \longrightarrow_\kappa & u[v \triangleright P].(\boldsymbol{\nu} xy)\,(Q \mid R) \\
(\boldsymbol{\nu} xy)\,(u(v).P \mid Q) & \longrightarrow_\kappa & u(v).(\boldsymbol{\nu} xy)\,(P \mid Q) \\
(\boldsymbol{\nu} xy)\,(u[\mathsf{in}\#].P \mid Q) & \longrightarrow_\kappa & u[\mathsf{in}\#].(\boldsymbol{\nu} xy)\,(P \mid Q) \\
(\boldsymbol{\nu} xy)\,(u.\mathsf{case}(P,Q) \mid R) & \longrightarrow_\kappa & u.\mathsf{case}((\boldsymbol{\nu} xy)\,(P \mid R),(\boldsymbol{\nu} xy)\,(Q \mid R)) \\
(\boldsymbol{\nu} xy)\,(!x(z).P \mid !u(v).Q) & \longrightarrow_\kappa & !u(v).(\boldsymbol{\nu} xy)\,(!x(z).P \mid Q) \\
(\boldsymbol{\nu} xy)\,(?u[v].P \mid Q) & \longrightarrow_\kappa & ?u[v].(\boldsymbol{\nu} xy)\,(P \mid Q) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid u().R)) & \longrightarrow_\kappa & u().(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid R)) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid u[v \triangleright R].T)) & \longrightarrow_\kappa & u[v \triangleright (\boldsymbol{\nu} xw)\,(P \mid R)].(\boldsymbol{\nu} yz)\,(Q \mid T) \quad w \in \mathsf{fn}(R) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid u[v \triangleright R].T)) & \longrightarrow_\kappa & u[v \triangleright R].(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid T)) \quad w \in \mathsf{fn}(T) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid u(v).R)) & \longrightarrow_\kappa & u(v).(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid R)) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid u[\mathsf{in}\#].R)) & \longrightarrow_\kappa & u[\mathsf{in}\#].(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid R)) \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid u.\mathsf{case}(R,T))) & & \\
\quad \longrightarrow_\kappa \quad u.\mathsf{case}((\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid R)),(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid T))) & & \\
(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid ?u[v].R)) & \longrightarrow_\kappa & ?u[v].(\boldsymbol{\nu} y[]z)\,(Q \mid (\boldsymbol{\nu} x[w])\,(P \mid R))
\end{array}
$$

Fig. 6. Semantics of Synchronous Forwarders: Commuting Conversions (κ)

From the theorem above we can extend synchronous forwarders with the six admissible cut rules, for which we write $P \vdash_{\text{cut}} \Gamma$. By induction on the number of cut rules in the derivation of a synchronous forwarder, we obtain immediately the main theorem of this section. In the sequel, $\Longrightarrow$ corresponds to $\longrightarrow_\kappa \circ \longrightarrow \circ \longrightarrow_\kappa$ and the $*$ denotes its reflexive and transitive closure.

Theorem 7 (Cut elimination). *If $P \vdash_{cut} \Gamma$ then there exists a cut-free Q such that $P \Longrightarrow^* Q$ and $Q \vdash \Gamma$.*

As mentioned in the introduction, we can deduce the following corollaries

Corollary 8 (Subject reduction). *If $P \vdash \Gamma$ and $P \Longrightarrow Q$, then $Q \vdash \Gamma$.*

Corollary 9 (Deadlock Freedom). *If $P \vdash \Gamma$, then there exists a restriction-free Q such that $P \Longrightarrow^* Q$ and $Q \vdash \Gamma$.*

Example 10. We revisit the two buyers example and its extension (Examples 3 and 5). Using compositionality, we can combine forwarders P_1 and P_2 into $P = (\boldsymbol{\nu} b_2' b_2'')\,(P_1 | P_2)$. By cut reductions, we can transform P into a new process P' that does not appeal to endpoints b_2' and b_2'', such that:

$$
\begin{array}{l}
P' \vdash \quad b_1 : \mathbf{name}^\perp \parr \mathbf{cost} \otimes \mathbf{cost}^\perp \parr \bot, c_1 : \mathbf{cost} \parr ((\mathbf{addr}^\perp \parr \bot) \& \bot), \\
c_2 : \mathbf{cost} \parr ((\mathbf{addr}^\perp \parr \bot) \& \bot), s : \mathbf{name} \otimes \mathbf{cost}^\perp \parr \mathbf{cost}^\perp \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1})
\end{array}
$$

And, by cut elimination, we know that the resulting process is a forwarder. □

5 From coherence to synchronous forwarders (and back)

In this section, we establish a strong connection between coherence and synchronous forwarders. First, we show that any coherence proof can be transformed

into a synchronous forwarder (soundness). Then, we show the opposite, i.e., every synchronous forwarder corresponds to a coherence proof (completeness).

5.1 Soundness

We begin with the soundness result, which builds on the already mentioned result from Carbone et al. [5], where a translation to arbiters in CLL is provided. The following theorem shows a stronger result since coherence can be translated into synchronous forwarders, which is a proper fragment CLL.

Theorem 11 (Soundness). *If $G \vDash \Delta$ then there exists a P, such that $P \vdash \Delta^{\perp}$*

Proof. By induction on the derivation of G, we show how to construct $P \vdash \Delta^{\perp}$. The construction, denoted by $\{\!|\cdot|\!\}$, is identical to that of [5] and is reported in Figure 2. Below, we report the case of $\otimes\parr$. The other cases are similar.

Suppose the derivation of G ends with

$$\frac{G_1 \vDash \{x_i : A_i\}_i, x : C \quad G_2 \vDash \Gamma, \{x_i : B_i\}_i, x : D}{G = x_1 \dots x_n \to y(G_1).G_2 \vDash \Gamma, \{x_i : A_i \otimes B_i\}_i, x : C \parr D} \otimes\parr$$

Then by i.h. on $G_1[y_i/x_i, y/x]$, there exists $P_1 \vdash \{y_i : A_i^{\perp}\}_i, y : C^{\perp}$; and by i.h. on G_2 there exists $P_2 \vdash \Gamma^{\perp}, \{x_i : B_i^{\perp}\}_i, x : D^{\perp}$. Therefore, we can construct:

$$\dfrac{\dfrac{P_1 \vdash \{y_i : A_i^{\perp}\}_i, y : C^{\perp} \quad P_2 \vdash \Gamma^{\perp}, \{x_i : B_i^{\perp}\}_i, x : D^{\perp}}{x[y \triangleright P_1].P_2 \vdash \Gamma^{\perp}, \{[y_i : A_i^{\perp}]x_i : B_i^{\perp}\}_i, x : C^{\perp} \otimes D^{\perp}} \otimes}{\vdots}\parr$$
$$x_1(y_1).\ \dots x_n(y_n).\ x[y \triangleright P_1].P_2 \vdash \Gamma^{\perp}, \{x_i : A_i^{\perp} \parr B_i^{\perp}\}_i, x : C^{\perp} \otimes D^{\perp}$$

which is indeed a synchronous forwarder derivation for

$$\{\!|G|\!\} \vdash (\Gamma, \{x_i : A_i \otimes B_i\}_i, x : C \parr D)^{\perp}$$

as requested. □

The proof of soundness shows not only that any coherence judgement $G \vDash \Delta$ can be proven as $\{\!|G|\!\} \vdash \Delta^{\perp}$, but also provides a constructive algorithm for it.

5.2 Completeness

We now move to proving the opposite of Theorem 11, i.e., given a synchronous forwarder $P \vdash \Delta$, we can always build a coherence proof $G \vDash \Delta^{\perp}$. Intuitively, the procedure for transforming a synchronous forwarder into a coherence proof consists of permuting rules within the derivation of $P \vdash \Delta$ in order to synthetise rules into coherence-like blocks. Additives and exponentials are treated by permuting $\oplus$s and ?s *down* to the matching & and ! respectively; while, in contrast, multiplicatives and units get regrouped by pushing $\parr$s and $\perp$s *up* to their corresponding $\otimes$ and **1** respectively, without changing the structure of the proof (apart from the rules moved down/up). This is because of the different communication patterns modelled by coherence: additives and exponentials use

broadcasting (one endpoint broadcasts a message to many endpoints); on the other hand, for multiplicatives and units, communication consists of *gathering* (many endpoints send a message to a single collector).

Additives and Exponentials. Our first step shows that any application of $\oplus_\#$ (for $\# \in \{l, r\}$) and any application of ? can be permuted down to the bottom of a proof if the proposition it introduces is still in the context.

Lemma 12 ($\oplus_\#$/? Invertibility).

1. *Let* $P \vdash \Gamma, \mathcal{S}_\#[\![\Delta, x : A_l \oplus A_r]\!]z : C$. *Then, there exists* P' *such that*
$$\frac{P' \vdash \Gamma, \mathcal{S}_\#[\![\Delta]\!]z : C, x : A_\#}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_\#[\![\Delta, x : A_l \oplus A_r]\!]z : C} \oplus_\#$$
2. *Let* $P \vdash \Gamma, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C$. *Then, there exists* P' *such that*
$$\frac{P' \vdash \Gamma, \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P' \vdash \Gamma, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C} ?$$

Proof. We show the result for $\oplus_l$; the ones for $\oplus_r$ and ? follow the same methodology, by induction on the size of the proof, and a case analysis on the last rule applied in the typing derivation of P. Note that besides permuting down the rule introducing $A_l \oplus A_r$, we do not change the proof structure.

If the last applied rule is $\oplus_l$ and it is on endpoint x then we are done (base case). Otherwise, rule $\oplus_l$ may concern either formulas in Δ or in some other box. In both cases, we proceed by a simple induction. Below, we look at the case where the formula is in another box.
$$\frac{P_1 \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, \mathcal{S}_l[\![\Delta']\!]y : F, w : D}{P = w[\mathsf{inl}].P_1 \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!]y : F} \oplus_l$$
By induction hypothesis, there exists P_1' such that
$$\frac{P_1' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, \mathcal{S}_l[\![\Delta']\!]y : F, w : D}{x[\mathsf{inl}].P_1' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, \mathcal{S}_l[\![\Delta']\!]y : F, w : D} \oplus_l$$
We can then reorganise the derivation to obtain $P' = w[\mathsf{inl}].P_1'$ with:
$$\frac{\dfrac{P_1' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, \mathcal{S}_l[\![\Delta']\!]y : F, w : D}{w[\mathsf{inl}].P_1' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!]y : F} \oplus_l}{x[\mathsf{inl}].w[\mathsf{inl}].P_1' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!]y : F} \oplus_l$$
The rest of the case analysis is similar. □

The first step for showing how to transform a synchronous forwarder into a coherence proof is to use Lemma 12 to build a forwarder that has immediately above every & all the associated $\oplus$s, and immediately above every ! all the associated ?s. For example, for the additives, it gives the following structure:
$$\frac{\dfrac{\dfrac{P_n \vdash \Gamma, x : A, \{y_i : C_i\}_{1 \le i \le n}}{\vdots}}{P_1 \vdash \Gamma, \mathcal{S}_l[\![\{y_i : C_i \oplus D_i\}_{2 \le i}]\!]x : A, y_1 : C_1} \quad \dfrac{\dfrac{Q_n \vdash \Gamma, x : A, \{y_i : D_i\}_{1 \le i \le n}}{\vdots}}{Q_1 \vdash \Gamma, \mathcal{S}_l[\![\{y_i : C_i \oplus D_i\}_{2 \le i}]\!]x : B, y_1 : D_1}}{\dfrac{P_0 \vdash \Gamma, \mathcal{S}_l[\![\{y_i : C_i \oplus D_i\}_i]\!]x : A}{x.\mathsf{case}(P_0, Q_0) \vdash \Gamma, \{y_i : C_i \oplus D_i\}_{1 \le i \le n}, x : A \,\&\, B} \quad \dfrac{Q_0 \vdash \Gamma, \mathcal{S}_r[\![\{y_i : C_i \oplus D_i\}_i]\!]x : B}{}}$$

To make this more precise, we extend the relation $\vdash$ to a new relation, dubbed $\vdash_{p_1}$, with the following two derivable rules (where index p stands for partial):

$$\frac{P \vdash_{p_1} \Gamma, \mathcal{S}_l[\![\oplus\Delta_1]\!]x : A, \Delta_2 \quad Q \vdash_{p_1} \Gamma, \mathcal{S}_r[\![\oplus\Delta_1]\!]x : B, \Delta_3}{x.\mathsf{case}(y_1[\mathsf{inl}].\ldots y_n[\mathsf{inl}].P, y_1[\mathsf{inr}].\ldots y_n[\mathsf{inr}].Q) \vdash_{p_1} \Gamma, \oplus\Delta_1, \Delta_2 \oplus \Delta_3, x : A \,\&\, B}\ \oplus\&_p^\perp$$

where $\Delta_2 = \{y_i : C_i\}_i$, $\Delta_3 = \{y_i : D_i\}_i$ and $\Delta_2 \oplus \Delta_3 = \{y_i : C_i \oplus D_i\}_i$;

$$\frac{P \vdash_{p_1} \mathcal{Q}[\![?\Delta_1]\!]y : A, \Delta_2}{!x(y).?x_1[y_1].\ldots?x_n[y_n].P \vdash_{p_1} ?\Delta_1, ?\Delta_2, x : !A}\ ?!_p^\perp$$

where $?\Delta_2 = \{x_i : ?B_i\}_i$ and $\Delta_2 = \{y_i : B_i\}_i$.

Rules $\oplus\&_p^\perp$ and $?!_p^\perp$ simulate the back-and-forth interaction of rules $\oplus/\&$ and $?/!$, respectively. In that respect, note that the process terms are obtained exactly in the form of arbiters. The following Lemma shows that, given a forwarder with no additive or exponential message in transit, additive and exponential rules become admissible in the presence of the new rules above. We say that $\&\oplus_p^\perp$ and $?!_p^\perp$ are *full*, written $\&\oplus^\perp$ and $?!^\perp$, whenever $\oplus\Delta_1$, or respectively $?\Delta_1$, is empty. We write $\mathcal{D} :: P \vdash_{p_1} \Gamma$ when $\mathcal{D}$ is a derivation of the judgement $P \vdash_{p_1} \Gamma$.

Lemma 13 (&/! Elimination). *Let $\mathcal{D} :: P \vdash_{p_1} \Gamma$ such that Γ has no $\mathcal{S}_l/\mathcal{S}_r/\mathcal{Q}$ box. There exists P' with $\mathcal{D}' :: P' \vdash_{p_1} \Gamma$ and $\mathcal{D}'$ is free from !, ?, &, $\oplus_l$ and $\oplus_r$.*

Proof. The first step of the proof is to replace all the &- and the !-rules, with $\&\oplus_p^\perp$ and $!?_p^\perp$ respectively, where Δ_2 and Δ_3 are empty. The proof then proceeds by induction on the height of $\mathcal{D}$, by doing a case analysis of the last applied rule in $\mathcal{D}$. We consider only the two cases where the last rule is $\&\oplus_p^\perp$ or $!?_p^\perp$, the other cases are obtained straightforwardly by permutation.

If $\mathcal{D}$ ends with $\&\oplus_p^\perp$, $P = z.\mathsf{case}(y_1[\mathsf{inl}].\ldots y_n[\mathsf{inl}].P_1, y_1[\mathsf{inr}].\ldots y_n[\mathsf{inr}].P_2)$ with:

$$\frac{P_1 \vdash_{p_1} \Gamma, \mathcal{S}_l[\![\oplus\Delta_1, x : A \oplus B]\!]z : D, \Delta_2 \quad P_2 \vdash_{p_1} \Gamma, \mathcal{S}_r[\![\oplus\Delta_1, x : A \oplus B]\!]z : C, \Delta_3}{P \vdash_{p_1} \Gamma, \oplus\Delta_1, x : A \oplus B, \Delta_2 \oplus \Delta_3, z : C \,\&\, D}\ \&\oplus_p^\perp$$

where Γ is free of $\mathcal{S}_l/\mathcal{S}_r/\mathcal{Q}$ boxes. By Lemma 12 (applicable here since a proof in $\vdash_{p_1}$ can always be expanded into a proof in $\vdash$), we know that there exist P_1' and P_2' such that:

$$\frac{P_1' \vdash \Gamma, \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, x : A, \Delta_2}{x[\mathsf{inl}].P_1' \vdash \Gamma, \mathcal{S}_l[\![\oplus\Delta_1, x : A \oplus B]\!]z : C, \Delta_2}\ \oplus_l$$

$$\frac{P_2' \vdash \Gamma, \mathcal{S}_r[\![\oplus\Delta_1]\!]z : D, x : B, \Delta_3}{x[\mathsf{inr}].P_2' \vdash \Gamma, \mathcal{S}_r[\![\oplus\Delta_1, x : A \oplus B]\!]z : D, \Delta_3}\ \oplus_r$$

We can take $P' = z.\mathsf{case}(y_1[\mathsf{inl}].\ldots y_n[\mathsf{inl}].x[\mathsf{inl}].P_1', y_1[\mathsf{inr}].\ldots y_n[\mathsf{inr}].x[\mathsf{inr}].P_2')$ as

$$\frac{P_1' \vdash_{p_1} \Gamma, \mathcal{S}_l[\![\oplus\Delta_1]\!]z : C, x : A, \Delta_2 \quad P_2' \vdash_{p_1} \Gamma, \mathcal{S}_r[\![\oplus\Delta_1]\!]z : D, x : B, \Delta_3}{P' \vdash_{p_1} \Gamma, \oplus\Delta_1, x : A \oplus B, \Delta_2 \oplus \Delta_3, z : C \,\&\, D}\ \oplus\&_p^\perp$$

Recursively, we repeat this process until all elements of $\oplus\Delta_1$ are exhausted and $\oplus\&_p^\perp$ becomes full. Then, the result follows by induction on the premisses, as they become free of exponential/additive boxes (and so is Γ).

If the last applied rule in $\mathcal{D}$ is $!?^{\perp}_{p}$, then $P = !u(v).?x_1[y_1].\ldots.?x_n[y_n].P_1$ with:

$$\frac{P_1 \vdash_{p_1} \mathcal{Q}[\![?\Delta_1, x : ?B]\!]v : A, \Delta_2}{P \vdash_{p_1} ?\Delta_1, x : ?B, ?\Delta_2, u : !A}\ ?!^{\perp}_{p}$$

By Lemma 12 (applicable for the same reason), there exists P_1' such that:

$$\frac{P_1' \vdash \mathcal{Q}[\![?\Delta_1]\!]v : A, y : B, \Delta_2}{?x[y].P_1' \vdash \mathcal{Q}[\![?\Delta_1, x : ?B]\!]v : A, \Delta_2}\ ?$$

which allows us to define $P' = !u(v).?x_1[y_1].\ldots.?x_n[y_n].?x[y].P_1'$ justified by

$$\frac{P_1' \vdash_{p_1} \mathcal{Q}[\![?\Delta_1]\!]v : A, y : B, \Delta_2}{P' \vdash_{p_1} ?\Delta_1, x : ?B, ?\Delta_2, u : !A}\ ?!^{\perp}_{p}$$

Recursively, we repeat this procedure until all of $?\Delta_1$ is exhausted, obtaining a full rule. The rest follows by induction. □

Multiplicatives and Units. We are now ready to undertake the last step of our transformation, i.e., dealing with multiplicatives and units. In such cases, given that the nature of the communication is gathering, we need to push all $\parr$s up to their corresponding $\otimes$ and all $\perp$s up to their corresponding $\mathbf{1}$. For this purpose, we change $\vdash_{p_1}$ to a new jugement, dubbed $\vdash_{p_2}$, where we remove rules $\&$, $\oplus_l$, $\oplus_r$, $?$ and $!$, replace them by $\&\oplus^{\perp}$ and $?!^{\perp}$, and add the rules:

$$\frac{P \vdash_{p_2} \Delta_1, \{u_i : A_i\}_i, v : C \quad Q \vdash_{p_2} \Gamma, \Delta_2, \{x_i : B_i\}_i, y : D}{x_1(u_1).\ldots.x_n(u_n).y[v \triangleright P].Q \vdash_{p_2} \Gamma, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr^{\perp}_{p}$$

and

$$\frac{}{x_1().\ldots.x_n().y[] \vdash_{p_2} \{x_i : \perp\}_i, y : \mathbf{1}, [*]}\ \mathbf{1}\perp^{\perp}_{p}$$

Rules $\otimes\parr^{\perp}_{p}$ and $\mathbf{1}\perp^{\perp}_{p}$ simulate the interplay of rules $\otimes/\parr$ and $\mathbf{1}/\perp$ respectively. Ultimately, we will use the *full* versions of these, written $\otimes\parr^{\perp}$ and $\mathbf{1}\perp^{\perp}$, that designate whenever $[\Delta_1]\Delta_2$ is empty in the former, and whenever $[*]$ is absent in the latter. Similarly to what we did for additives and exponentials, we show now that we can replace the remaning rules from $\vdash$ by compound ones.

Lemma 14 ($\parr/\perp$ Admissibility).

1. *Let $\mathcal{D} :: P \vdash_{p_2} \Gamma, [y : A]x : B$ such that $\mathcal{D}$ is $\parr$-free. Then, there exist a forwarder Q and a $\parr$-free proof $\mathcal{E}$ such that $\mathcal{E} :: Q \vdash_{p_2} \Gamma, x : A \parr B$.*
2. *Let $\mathcal{D} :: P \vdash_{p_2} \Gamma, [*]$ such that $\mathcal{D}$ is $\perp$-free. Then, there exists a $\perp$-free proof $\mathcal{E}$ and Q such that $\mathcal{E} :: Q \vdash_{p_2} \Gamma, x : \perp$, for some x.*

Proof. We proceed by induction on the size of $\mathcal{D}$ and a case analysis on the last applied rule. We only report on case in the proof of item 1 when the last applied rule in $\mathcal{D}$ is $\otimes\parr^{\perp}_{p}$, that is, $P = x_1(u_1).\ldots.x_n(u_n).z[v \triangleright P_1].P_2$ and we have two possible subcases. (The other ones are simpler.)

If $[y : A]x : B$ is not touched by $\otimes\parr^{\perp}_{p}$, we have:

$$\frac{P_1 \vdash_{p_2} \Delta_1, \{u_i : A_i\}_i, v : C \quad P_2 \vdash_{p_2} \Gamma, [y : A]x : B, \Delta_2, \{x_i : B_i\}_i, z : D}{P \vdash_{p_2} \Gamma, [y : A]x : B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, z : C \otimes D}\ \otimes\parr^{\perp}_{p}$$

By induction hypothesis, since our proof is $\parr$-free, there exists Q_2 such that $Q_2 \vdash_{p_2} \Gamma, x : A \parr B, \Delta_2, \{x_i : B_i\}_i, z : D$. By applying $\otimes\parr^{\perp}_{p}$ again, we obtain $Q = x_1(u_1).\ldots x_n(u_n).z[v \triangleright P_1].Q_2$ and:

$$\frac{P_1 \vdash_{p_2} \Delta_1, \{u_i : A_i\}_i, v : C \quad Q_2 \vdash_{p_2} \Gamma, x : A \parr B, \Delta_2, \{x_i : B_i\}_i, z : D}{Q \vdash_{p_2} \Gamma, x : A \parr B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, z : C \otimes D} \otimes\parr^{\perp}_{p}$$

If $[y : A]x : B$ is indeed modified by $\otimes\parr^{\perp}_{p}$, then we have a base case:

$$\frac{P_1 \vdash_{p_1} y : A, \Delta_1, \{u_i : A_i\}_i, v : C \quad P_2 \vdash_{p_1} \Gamma, x : B, \Delta_2, \{x_i : B_i\}_i, z : D}{P \vdash_{p_1} \Gamma, [y : A]x : B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, z : C \otimes D} \otimes\parr^{\perp}_{p}$$

We can take $Q = x(y).P$ and simply change the rule $\otimes\parr^{\perp}_{p}$ obtaining:

$$\frac{P_1 \vdash_{p_1} y : A, \Delta_1, \{x_i : A_i\}_i, y : C \quad P_2 \vdash_{p_1} \Gamma, x : B, \Delta_2, \{x_i : B_i\}_i, y : D}{Q \vdash_{p_1} \Gamma, x : A \parr B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D} \otimes\parr^{\perp}_{p}$$

The proof for item 2 is similar. Note that in the base case, we can have

$$\mathcal{D} = \frac{}{P = x_1().\ldots x_n().y[] \vdash_{p_1} \{x_i : \perp\}_i, y : \mathbf{1}, [*]}\ \mathbf{1}\perp^{\perp}_{p}$$

which gives us $\mathcal{E} = \dfrac{}{Q = x().x_1().\ldots x_n().y[] \vdash_{p_1} \{x_i : \perp\}_i, y : \mathbf{1}, x : \perp}\ \mathbf{1}\perp^{\perp}$. □

As a corollary, we can always eliminate all $\otimes$ with their corresponding $\parr$s and all the $\mathbf{1}$ with their corresponding $\perp$s.

Lemma 15 ($\otimes$/1 Elimination). *Let $\mathcal{D} :: P \vdash_{p_2} \Gamma$. Then, there exist a forwarder Q with $\mathcal{E} :: Q \vdash_{p_2} \Gamma$, and $\mathcal{E}$ is free from $\parr$, $\otimes$, $\perp$, and $\mathbf{1}$.*

Proof. It follows from replacing any instance of $\otimes$ and $\mathbf{1}$ with $\otimes\parr^{\perp}_{p}$ (with empty Δ_1 and Δ_2) and $\mathbf{1}\perp^{\perp}_{p}$, respectively; then, applying the previous Lemma repeatedly to the top-most instances of $\parr$ or $\perp$ first.

Theorem 16 (Completeness). *If $P \vdash \Delta$, then there exists a global type G, s.t. $G \vDash \Delta^{\perp}$.*

Proof. It follows from the previous results, noting that in order to get coherence, all rules from $\vdash_{p_2}$ must be full, which is the case since our context is a basic Δ. We observe that in the case of multiplicatives, we need to perform a name substitution in order to obtain a valid coherence proof. Below, let $\{\!\{P\}\!\}$ be the function that transforms a process term P corresponding to a proof in $\vdash_{p_2}$ with only full rules to a global type. Then,

$$\frac{P \vdash_{p_2} \{u_i : A_i\}_i, v : C \quad Q \vdash_{p_2} \Delta, \{x_i : B_i\}_i, y : D}{x_1(u_1).\ldots x_n(u_n).y[v \triangleright P].Q \vdash_{p_2} \Delta, \{x_i : A_i \parr B_i\}_i, y : C \otimes D} \otimes\parr^{\perp}$$

is transformed into

$$\frac{\{\!\{P'\}\!\} \vDash \{x_i : A_i\}_i, y : C \quad \{\!\{Q\}\!\} \vDash \Delta, \{x_i : B_i\}_i, y : D}{\tilde{x} \to y(\{\!\{P'\}\!\}).\{\!\{Q\}\!\} \vDash \Delta, \{x_i : A_i \otimes B_i\}_i, y : C \parr D} \otimes\parr$$

such that $P' = P\{y/v, \tilde{x}/\tilde{u}\}$. □

Example 17. Let us consider a variation of process P_1 from Example 3:

$$(\text{as } P_1)\ldots \quad b'_1(y).\ s'(x_2).\ b'_2[x'_2 \triangleright x_2 \leftrightarrow x'_2].\ b'_2[y' \triangleright y \leftrightarrow y'].\ \ldots (\text{as } P_1)$$

The process above still enforces the same protocol, we can transform it into P_1 by permuting some actions, and then into the coherence proof in Example 2. □

6 Related Work

Our work takes [5] as a starting point. Guided by CLL, we set out to explore if coherence can be broken down into more elementary logical rules which led to the discovery of synchronous forwarders. Caires and Perez [2] also study multiparty session types in the context of intuitionistic linear logic by translating global types to processes, called *mediums*. Their work does not start from a logical account of global types (their global types are just syntactic terms). But, as in this paper and previous work [5], they do generate arbiters as linear logic proofs. In this work, we also achieve the converse: from a forwarder process, we provide a procedure for generating a global type (coherence proof).

Sangiorgi [17], probably the first to treat forwarders for the π-calculus, uses binary forwarders, i.e., processes that only forward between two channels, which is equivalent to our $x \leftrightarrow y$. We attribute our result to the line of work that originated in 2010 by Caires and Pfenning [3], where forwarders à la Sangiorgi were introduced as processes to be typed by the axiom rule in linear logic. Van den Heuvel and Perez [18] have recently developed a version of linear logic that encompasses both classical and intuitionistic logic, presenting a unified view on binary forwarders in both logics.

Gardner et al. [8] study the expressivity of the linear forwarder calculus, by encoding the asynchronous π-calculus (since it can encode distributed choice). The linear forwarder calculus is a variant of the (asynchronous) π-calculus that has binary forwarders and a restriction on the input $x(y).P$ such that y cannot be used for communicating (but only forwarded). Such a restriction is similar to the intuition behind synchronous forwarders, with the key difference that it would not work for some of our session-based primitives.

Barbanera and Dezani [1] study multiparty session types as gateways which are basically forwarders that work as a medium among many interacting parties, forwarding communications between two multiparty sessions. Such mechanism reminds us of our forwarder composition: indeed, in their related work discussion they do mention that their gateways could be modeled by a "connection-cut".

Recent work [14, 10] proposes an extension of linear logic that models *identity providers*, a sort of monitoring mechanisms that are basically forwarders between two channels in the sense of Sangiorgi, but asynchronous, i.e., they allow unbounded buffering of messages before forwarding.

Our forwarder mechanism may be confused with that of locality [15], which is discussed from a logical point of view by Caires et al. [4]. Locality only requires that received channels cannot be used for inputs (that can only be done at the location where the channel was created). In our case instead, we do not allow received channels to be used at all until a new forwarder is created.

The transformations between coherence and synchronous forwarders are related to those of projection and extraction for choreographies. A choreography is basically a global description of a the sequence of interactions (communications) that must happen in a distributed system (like a global type). Carbone et al. [6] give a characterisation of this in intuitionistic linear logic, by using hypersequents to represent both choreographies and the processes corresponding to

those choreographies: through proof transformations they show how to go from choreographies to processes and vice versa. Although we also transform choreographies (coherence) intro processes, our forwarder is a single point of control while they deal with a distributed implementation.

7 Discussion and Future Work

Coherence Compositionality. The results of this paper give us compositionality (cut) and cut elimination also for coherence proofs. In fact, we can always transform two coherence proofs into synchronous forwarders, compose and normalise them, and finally translate them back to coherence.

Process Language. Our process language is based on that of [21] with some omissions. For the sake of presentation, we have left out polymorphic communications. We believe that these communication primitives, together with polymorphic types $\exists X.A$ and $\forall X.A$, can be added to synchronous forwarders. Moreover, our process language does not support recursion for coherence nor for synchronous forwarders. We leave these points as future work.

Classical vs Intuitionistic Linear Logic. In this paper, we have chosen to base our theory on CLL for two main reasons. Coherence is indeed defined by Carbone et al. [5] in terms of CLL and therefore our results can immediately be related to theirs without further investigations. We would like to remark that an earlier version of synchronous forwarders was based on intuitionistic linear logic, but moving to CLL required many fewer rules and greatly improved the presentation of our results. Nevertheless, our results can be easily reproduced in intuitionistic linear logic, including a straightforward adaptation of coherence.

Exponentials, Weakening and Contraction. In our work, we do not allow synchronous forwarders to harness the full power of exponentials, because we disallow the use of weakening and contraction. In some sense exponentials are used linearly. Weakening allows us to extend the context by fresh channels provided that they are ?-quantified. Weakening is also useful when composing a process offering some service of type $!A$ with some process that does not wish to use such service. Contraction, on the other hand, models server duplication, i.e., creating a copy of a server for every possible client. Guided by the given definition of coherence, we are sure that neither weakening nor contraction reflect our intuition of synchronous forwarders. In fact, we would like to remark that adding these rules to synchronous forwarders would invalidate the completeness result. This is because coherence is apparently incompatible with weakening and contraction of assumptions in the context. We leave a further investigation of how and if to add weakening and contraction to future work.

Unlimited-Size Buffers. Synchronous forwarders guarantee that the order of messages between two endpoints is preserved. This is achieved by preventing the sending endpoint from sending further messages until the previous message has been forwarded. As future work, we wish to consider buffers of any size, i.e., a sender can keep on sending messages that can be stored in the forwarder and then

forwarded at a later time. Our idea is to generalise, e.g., a boxing $[y : A]x : B$ to allow for more messages to be stored as in $[y_1 : A_1 \dots y_n : A_n]x : B$, and at the same time allowing $x : B$ to be used. At first this may seem a simple extension of synchronous forwarders. However, it has major implications in the proof of cut elimination that we must better understand. Note that the system of forwarders by Gommerstadt et al. [14, 10] does have unlimited-size buffers, but it is restricted to binary forwarders. In this case, the proof of cut elimination is standard. Unfortunately, adding just a third endpoint having unbounded queues breaks the standard structural proof ($\otimes$ and Cut do not commute as in page 14).

Complete Interleaving of Synchronous Forwarders. In synchronous forwarders, the rule for $\&$ requires that a non-empty set of formulas $\Delta = \{A_i \oplus B_i\}$ is selected from the context and boxed, effectively forcing the processes to interact with the choice. This design decision was necessary to achieve cut elimination as well as our completeness theorem for mapping synchronous forwarders to coherence. But it comes at a price: it restricts the number of proofs, and consequently, there are processes that are still forwarders, implement a 1-size buffer, their CLL type is coherent, but are not typable in $\vdash$. This is because the order of communications interferes with the $\&$ rule. For example, the process $z(y).x.\mathsf{case}(\ z[\mathsf{inl}].P,\ z[\mathsf{inr}].Q)$ (for some adequate P and Q) is a synchronous forwarder. However, the slightly different version $x.\mathsf{case}(\ z(y).z[\mathsf{inl}].P,\ z(y).z[\mathsf{inr}].Q)$ is typable in CLL but is not a synchronous forwarder, despite the input on z is totally unrelated to the branching. And its typing context is also provable in coherence. Unfortunately, all attempts to generalise the $\&$ rule have broken cut elimination. We leave a further investigation to future work.

Variants of Coherence. Our results show that synchronous forwarders are also coherent. As a follow-up, we would like to investigate in future work, whether generalised variants of forwarders also induce interesting generalised notions of coherence, and, as a consequence, generalisations of global types.

8 Conclusions

To our knowledge, this work is the first to give characterisation of coherence in terms of forwarders which generalise the concept of arbiter. We have developed a proof system based on linear logic that models a class of forwarders, called synchronous forwarders, that preserve message order. Well typed-forwarders are shown to be compositional. We show that synchronous forwarders provide a sound and complete characterisation of coherence and therefore provide a logic of global types and a protocol language for describing distributed protocols.

References

1. Franco Barbanera and Mariangiola Dezani-Ciancaglini. Open multiparty sessions. In Massimo Bartoletti, Ludovic Henrio, Anastasia Mavridou, and Alceste Scalas, editors, *Proceedings 12th Interaction and Concurrency Experience, ICE 2019, Copenhagen, Denmark, 20-21 June 2019*, volume 304 of *EPTCS*, pages 77–96, 2019.
2. Luís Caires and Jorge A. Pérez. Multiparty session types within a canonical binary theory, and beyond. In Elvira Albert and Ivan Lanese, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 36th IFIP WG 6.1 International Conference, FORTE 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*, volume 9688 of *Lecture Notes in Computer Science*, pages 74–95. Springer, 2016.
3. Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In *CONCUR*, pages 222–236, 2010.
4. Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *Math. Struct. Comput. Sci.*, 26(3):367–423, 2016.
5. Marco Carbone, Sam Lindley, Fabrizio Montesi, Carsten Schürmann, and Philip Wadler. Coherence generalises duality: A logical explanation of multiparty session types. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*, pages 33:1–33:15, Germany, 2016. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.
6. Marco Carbone, Fabrizio Montesi, and Carsten Schürmann. Choreographies, logically. In *CONCUR*, pages 47–62, 2014.
7. Marco Carbone, Fabrizio Montesi, Carsten Schürmann, and Nobuko Yoshida. Multiparty session types as coherence proofs. In *CONCUR*, pages 412–426, 2015.
8. Philippa Gardner, Cosimo Laneve, and Lucian Wischik. Linear forwarders. *Inf. Comput.*, 205(10):1526–1550, 2007.
9. Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
10. Hannah Gommerstadt, Limin Jia, and Frank Pfenning. Session-typed concurrent contracts. In Amal Ahmed, editor, *Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10801 of *Lecture Notes in Computer Science*, pages 771–798. Springer, 2018.
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. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 273–284. ACM, 2008.
13. Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *JACM*, 63(1):9, 2016. Also: POPL, 2008, pages 273–284.
14. Limin Jia, Hannah Gommerstadt, and Frank Pfenning. Monitors and blame assignment for higher-order session types. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on*

Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016, pages 582–594. ACM, 2016.
15. Massimo Merro and Davide Sangiorgi. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science*, 14(5):715–767, 2004.
16. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, September 1992.
17. Davide Sangiorgi. pi-calculus, internal mobility, and agent-passing calculi. *Theor. Comput. Sci.*, 167(1&2):235–274, 1996.
18. Bas van den Heuvel and Jorge A. Pérez. Session type systems based on linear logic: Classical versus intuitionistic. In Stephanie Balzer and Luca Padovani, editors, *Proceedings of the 12th International Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software, PLACES@ETAPS 2020, Dublin, Ireland, 26th April 2020*, volume 314 of *EPTCS*, pages 1–11, 2020.
19. Vasco T. Vasconcelos. Fundamentals of session types. *Inf. Comput.*, 217:52–70, 2012.
20. Philip Wadler. Propositions as sessions. In *ICFP*, pages 273–286, 2012.
21. Philip Wadler. Propositions as sessions. *Journal of Functional Programming*, 24(2–3):384–418, 2014. Also: ICFP, pages 273–286, 2012.

A CLL [20] with simple exponentials

$$\frac{}{x \to y^A \vdash x : A^\perp, y : A}\ \textsc{Axiom} \qquad \frac{P \vdash \Gamma}{x().P \vdash \Gamma, x : \bot}\ \bot \qquad \frac{}{x[] \vdash x : \mathbf{1}}\ \mathbf{1}$$

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

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

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

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

B Cut Elimination Proof for Synchronous Forwarders

Theorem 18 (Cut-admissibility).

1. *If* $\mathcal{D} :: \Gamma_1, A$ *and* $\mathcal{E} :: \Gamma_2, A^\perp$ *then* Γ_1, Γ_2.
2. *If* $\mathcal{D} :: \Delta_1, A$ *and* $\mathcal{E} :: \Delta_2, \Gamma_2, B$ *and* $\mathcal{F} :: \Gamma_3, [A^\perp]B^\perp$ *then* $[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3$.
3. *If* $\mathcal{D} :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C$ *and* $\mathcal{E} :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\perp$ *and* $\mathcal{F} :: \Gamma_2, \mathcal{S}_r[\![\Delta_2]\!]B^\perp$ *then* $\Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$.
4. *If* $\mathcal{D} :: \Gamma_1, \mathcal{S}_l[\![\Delta_1]\!]C, A$ *and* $\mathcal{E} :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\perp$ *then* $\Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$.
5. *If* $\mathcal{D} :: \Gamma_1, \mathcal{S}_r[\![\Delta_1, A \oplus B]\!]C$ *and* $\mathcal{E} :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\perp$ *and* $\mathcal{F} :: \Gamma_2, \mathcal{S}_r[\![\Delta_2]\!]B^\perp$ *then* $\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_1, \Delta_2]\!]C$.
6. *If* $\mathcal{D} :: \Gamma_1, \mathcal{S}_r[\![\Delta_1]\!]C, B$ *and* $\mathcal{E} :: \Gamma_2, \mathcal{S}_r[\![\Delta_2]\!]B^\perp$ *then* $\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_1, \Delta_2]\!]C$.
7. *If* $\mathcal{D} :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C$ *and* $\mathcal{E} :: \mathcal{Q}[\![\Delta_2]\!]A^\perp$ *then* $\Gamma_1, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$.
8. *If* $\mathcal{D} :: \Gamma_1, \mathcal{Q}[\![\Delta_1]\!]C, A$ *and* $\mathcal{E} :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp$ *then* $\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$.

Proof. by induction over the cut formula and the left and right derivation.

1. By induction on $\mathcal{D}$
   Impossible Cases:
   Axiom Case:
   $$\mathcal{D} = \frac{}{A^\perp, A}\ \text{Ax}$$
   $A^\perp, \Gamma_2$ by $\mathcal{E}$
   Key Case:
   $$\mathcal{D} = \frac{}{\mathbf{1}, [*]^n}\ \mathbf{1}$$
   and
   $$\mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma_2, [*]}{\Gamma_2, \bot}\ \bot$$
   $\Gamma_2, [*]$ by $\mathcal{E}_1$
   Key Case:
   $$\mathcal{D} = \frac{\mathcal{D}_1 :: \Delta_1, A \quad \mathcal{D}_2 :: \Gamma_1, \Delta_2, B}{\Gamma_1, [\Delta_1]\Delta_2, A \otimes B}\ \otimes$$
   and
   $$\mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma_2, [A^\perp]B^\perp}{\Gamma_2, A^\perp \parr B^\perp}\ \parr$$
   $[\Delta_1]\Delta_2, \Gamma_1, \Gamma_2$ by i.h. 2 on $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{E}_1$
   Key Case:
   $$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![A \oplus B, \oplus\Delta_1]\!]C \quad \mathcal{D}_2 :: \Gamma_1, \mathcal{S}_r[\![A \oplus B, \oplus\Delta_1]\!]D}{\Gamma_1, A \oplus B, \oplus\Delta_1, C \,\&\, D}\ \&$$
   and
   $$\mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\oplus\Delta_2]\!]A^\perp \quad \mathcal{E}_2 :: \Gamma_2, \mathcal{S}_r[\![\oplus\Delta_2]\!]B^\perp}{\Gamma_2, \oplus\Delta_2, A^\perp \,\&\, B^\perp}\ \&$$

$\mathcal{G}_1 :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\oplus\Delta_1, \oplus\Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$ and $\mathcal{E}_1$ and $\mathcal{E}_2$
$\mathcal{G}_2 :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\oplus\Delta_1, \oplus\Delta_2]\!]D$ by i.h. 5 on $\mathcal{D}_2$ and $\mathcal{E}_1$ and $\mathcal{E}_2$
$\Gamma_1, \Gamma_2, \oplus\Delta_1, \oplus\Delta_2, C \,\&\, D$ by $\&$ on $\mathcal{G}_1$ and $\mathcal{G}_2$

Key Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \mathcal{Q}[\![?\Delta_1, ?A]\!]C}{?\Delta_1, ?A, !C}\ !$$

and

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \mathcal{Q}[\![?\Delta_2]\!]A^{\perp}}{?\Delta_2, !A^{\perp}}\ !$$

$\mathcal{G} :: \mathcal{Q}[\![?\Delta_1, ?\Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}_1$
$?\Delta_1, ?\Delta_2, !C$ by ! on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, A, [*]}{\Gamma_1, A, \perp}\ \perp$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [*]$ by i.h. 1 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, \perp$ by $\perp$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Delta_1, D \quad \mathcal{D}_2 :: \Gamma_1, A, \Delta_2, E}{\Gamma_1, A, [\Delta_1]\Delta_2, D \otimes E}\ \otimes$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \Delta_2, E$ by i.h. 1 on $\mathcal{D}_2$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, [\Delta_1]\Delta_2, D \otimes E,$ by $\otimes$ on $\mathcal{D}_1$ and $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, A, [D]E}{\Gamma_1, A, D \parr E}\ \parr$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [D]E$ by i.h. 1 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, D \parr E$ by $\parr$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, A, \mathcal{S}_l[\![\oplus\Delta]\!]D \quad \mathcal{D}_2 :: \Gamma_1, A, \mathcal{S}_r[\![\oplus\Delta]\!]E}{\Gamma_1, A, \oplus\Delta, D \,\&\, E}\ \&$$

$\mathcal{G}_1 :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\oplus\Delta]\!]D$ by i.h. 1 on $\mathcal{D}_1$ and $\mathcal{E}$
$\mathcal{G}_2 :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\oplus\Delta]\!]E$ by i.h. 1 on $\mathcal{D}_2$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, \oplus\Delta, D \,\&\, E$ by $\&$ on $\mathcal{G}_1$ and $\mathcal{G}_2$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, A, \mathcal{S}_l[\![\Delta]\!]C, D}{\Gamma_1, A, \mathcal{S}_l[\![\Delta, D \oplus E]\!]C}\ \oplus_1$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta]\!]C, D$ by i.h. 1 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta, D \oplus E]\!]C$ by $\oplus_1$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, A, \mathcal{S}_r[\![\Delta]\!]C, E}{\Gamma_1, A, \mathcal{S}_r[\![\Delta, D \oplus E]\!]C}\ \oplus_2$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta]\!]C, E$ by i.h. 1 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta, D \oplus E]\!]C$ by $\oplus_2$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, A, \mathcal{Q}[\![\Delta]\!]C, D}{\Gamma_1, A, \mathcal{Q}[\![\Delta, ?D]\!]C}\ ?$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta]\!]C, D$ by i.h. 1 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta, ?D]\!]C$ by ? on $\mathcal{G}$

2. By induction on $\mathcal{F}$.
Impossible Cases: Ax, **1**, !.
Key Case:

$$\mathcal{F} = \frac{\mathcal{F}_1 :: A^\perp, C \quad \mathcal{F}_2 :: \Gamma_3, B^\perp, D}{\Gamma_3, C \otimes D, [A^\perp]B^\perp}\ \otimes$$

$\mathcal{G}_1 :: \Delta_1, C$ by i.h. 1 on $\mathcal{D}$ and $\mathcal{F}_1$
$\mathcal{G}_2 :: \Delta_2, \Gamma_2, \Gamma_3, D$ by i.h. 1 on $\mathcal{E}$ and $\mathcal{F}_2$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, C \otimes D$ by $\otimes$ on $\mathcal{G}_1$ and $\mathcal{G}_2$

Commutative Case:

$$\mathcal{F} = \frac{\mathcal{F}_1 :: \Delta_1', C \quad \mathcal{F}_2 :: \Gamma_3, [A^\perp]B^\perp, \Delta_2', D}{\Gamma_3, [\Delta_1']\Delta_2', C \otimes D, [A^\perp]B^\perp}\ \otimes$$

$\mathcal{G} :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \Delta_2', D$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_2$
$[\Delta_1']\Delta_2', C \otimes D, [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3$ by $\otimes$ on $\mathcal{F}_1$ and $\mathcal{G}$

Commutative Case:

$$\mathcal{F} = \frac{\mathcal{F}_1 :: \Gamma_3, [A^\perp]B^\perp, [*]}{\Gamma_3, [A^\perp]B^\perp, \perp}\ \perp$$

$\mathcal{G} :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, [*]$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_1$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \perp$ by $\perp$ on $\mathcal{G}$

Commutative Case:

$$\mathcal{F} = \frac{\mathcal{F}_1 :: \Gamma_3, [A^\perp]B^\perp, [C]D}{\Gamma_3, [A^\perp]B^\perp, C \parr D}\ \parr$$

$\mathcal{G} :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, [C]D$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_1$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, C \parr D$ by $\parr$ on $\mathcal{G}$

Commutative Case:

$$\mathcal{F} = \dfrac{\mathcal{F}_1 :: \Gamma_3, [A^\perp]B^\perp, \mathcal{S}_l[\![\oplus\Delta]\!]C \quad \mathcal{F}_2 :: \Gamma_3, [A^\perp]B^\perp, \mathcal{S}_r[\![\oplus\Delta]\!]D}{\Gamma_3, [A^\perp]B^\perp, \oplus\Delta, C \,\&\, D}\ \&$$

$\mathcal{G}_1 :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{S}_l[\![\oplus\Delta]\!]C$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_1$
$\mathcal{G}_2 :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{S}_r[\![\oplus\Delta]\!]D$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_2$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \oplus\Delta, C \,\&\, D$ by & on $\mathcal{G}_1$ and $\mathcal{G}_2$

Commutative Case:

$$\mathcal{F} = \dfrac{\mathcal{F}_1 :: \Gamma_3, [A^\perp]B^\perp, \mathcal{S}_l[\![\Delta]\!]C, D}{\Gamma_3, [A^\perp]B^\perp, \mathcal{S}_l[\![\Delta, D \oplus E]\!]C}\ \oplus_1$$

$\mathcal{G} :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{S}_l[\![\Delta]\!]C, D$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_1$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{S}_l[\![\Delta, D \oplus E]\!]C$ by $\oplus_1$ on $\mathcal{G}$

Commutative Case:

$$\mathcal{F} = \dfrac{\mathcal{F}_1 :: \Gamma_3, [A^\perp]B^\perp, \mathcal{S}_l[\![\Delta]\!]C, E}{\Gamma_3, [A^\perp]B^\perp, \mathcal{S}_l[\![\Delta, D \oplus E]\!]C}\ \oplus_2$$

$\mathcal{G} :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{S}_l[\![\Delta]\!]C, E$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_1$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{S}_l[\![\Delta, D \oplus E]\!]C$ by $\oplus_2$ on $\mathcal{G}$

Commutative Case:

$$\mathcal{F} = \dfrac{\mathcal{F}_1 :: \Gamma_3, [A^\perp]B^\perp, \mathcal{Q}[\![\Delta]\!]C, D}{\Gamma_3, [A^\perp]B^\perp, \mathcal{Q}[\![\Delta, ?D]\!]C}\ ?$$

$\mathcal{G} :: [\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{Q}[\![\Delta]\!]C, D$ by i.h. 2 on $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{F}_1$
$[\Delta_1]\Delta_2, \Gamma_2, \Gamma_3, \mathcal{Q}[\![\Delta, ?D]\!]C$ by ? on $\mathcal{G}$

3. By induction on $\mathcal{D}$
Impossible Cases: Ax, **1**, !
Key Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1]\!]C, A}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C}\ \oplus_1$$

$\Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}_1$ and $\mathcal{E}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, [*]}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \perp}\ \perp$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [*], \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, \perp, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\perp$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Delta_3, D \quad \mathcal{D}_2 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \Delta_4, E}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, [\Delta_3]\Delta_4, D \otimes E}\ \otimes$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \Delta_4, E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_2$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, [\Delta_3]\Delta_4, D \otimes E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\otimes$ on $\mathcal{D}_1$ and $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, [D]E}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, D \parr E}\ \parr$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [D]E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, D \parr E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\parr$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{S}_l[\![\oplus\Delta]\!]D \quad \mathcal{D}_2 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{S}_r[\![\oplus\Delta]\!]E}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \oplus\Delta, D \,\&\, E}\ \&$$

$\mathcal{G}_1 :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\oplus\Delta]\!]D, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$, $\mathcal{E}$, and $\mathcal{F}$
$\mathcal{G}_2 :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\oplus\Delta]\!]E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_2$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, D \,\&\, E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\&$ on $\mathcal{G}_1$ and $\mathcal{G}_2$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{S}_r[\![\Delta_3]\!]F, D}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_1$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3]\!]F, D, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_1$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{S}_r[\![\Delta_3]\!]F, E}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_2$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3]\!]F, E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_2$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{Q}[\![\Delta]\!]E, D}{\Gamma_1, \mathcal{S}_l[\![\Delta_1, A \oplus B]\!]C, \mathcal{Q}[\![\Delta, ?D]\!]E}\ ?$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta]\!]E, D, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 3 on $\mathcal{D}_1$, $\mathcal{E}$, and $\mathcal{F}$
$\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta, ?D]\!]E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by ? on $\mathcal{G}$

4. By induction on $\mathcal{E}$

Impossible Cases: Ax, **1**, !

Key Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, B, A^\perp}{\Gamma_2, \mathcal{S}_l[\![B \oplus D]\!]A^\perp}\ \oplus_1$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta_1]\!]C, B$ by i.h. 1 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{S}_l[\![\Delta_1, B \oplus D]\!]C$ by $\oplus_1$ on $\mathcal{G}$.

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\perp, [*]}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\perp, \bot}\ \bot$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [*], \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \bot, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\bot$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Delta_3, D \quad \mathcal{E}_2 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \Delta_4, E}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, [\Delta_3]\Delta_4, D \otimes E}\ \otimes$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \Delta_4, E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma_1, \Gamma_2, [\Delta_3]\Delta_4, D \otimes E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\otimes$ on $\mathcal{E}_1$ and $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, [D]E}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, D \parr E}\ \parr$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [D]E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, D \parr E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\parr$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{S}_l[\![\oplus\Delta]\!]D \quad \mathcal{E}_2 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{S}_r[\![\oplus\Delta]\!]E}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \oplus\Delta, D \,\&\, E}\ \&$$

$\mathcal{G}_1 :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\oplus\Delta]\!]D, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_1$
$\mathcal{G}_2 :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\oplus\Delta]\!]E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma_1, \Gamma_2, D \,\&\, E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\&$ on $\mathcal{G}_1$ and $\mathcal{G}_2$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{S}_r[\![\Delta_3]\!]F, D}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_1$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3]\!]F, D, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_1$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{S}_r[\![\Delta_3]\!]F, E}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_2$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3]\!]F, E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_2$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{Q}[\![\Delta]\!]E, D}{\Gamma_2, \mathcal{S}_l[\![\Delta_2]\!]A^\bot, \mathcal{Q}[\![\Delta, ?D]\!]E}\ ?$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta]\!]E, D, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by i.h. 4 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta, ?D]\!]E, \mathcal{S}_l[\![\Delta_1, \Delta_2]\!]C$ by ? on $\mathcal{G}$

5. The proof is analogous to (3), simply by replacing $\mathcal{S}_l$ by $\mathcal{S}_r$.
6. The proof is analogous to (4), simply by replacing $\mathcal{S}_l$ by $\mathcal{S}_r$.

7. By induction on $\mathcal{D}$
Impossible Cases: Ax, **1**, !
Key Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1]\!]C, A}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C}\ ?$$

$\Gamma_1, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}_1$ and $\mathcal{E}$
Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, [*]}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \bot}\ \bot$$

$\mathcal{G} :: \Gamma_1, [*], \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \bot, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\bot$ on $\mathcal{G}$
Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Delta_3, D \quad \mathcal{D}_2 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \Delta_4, E}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, [\Delta_3]\Delta_4, D \otimes E}\ \otimes$$

$\mathcal{G} :: \Gamma_1, \Delta_4, E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_2$ and $\mathcal{E}$
$\Gamma_1, [\Delta_3]\Delta_4, D \otimes E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\otimes$ on $\mathcal{D}_1$ and $\mathcal{G}$
Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, [D]E}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, D \parr E}\ \parr$$

$\mathcal{G} :: \Gamma_1, [D]E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, D \parr E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\parr$ on $\mathcal{G}$
Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{S}_l[\![\oplus\Delta]\!]D \quad \mathcal{D}_2 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{S}_r[\![\oplus\Delta]\!]E}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \oplus\Delta, D \,\&\, E}\ \&$$

$\mathcal{G}_1 :: \Gamma_1, \mathcal{S}_l[\![\oplus\Delta]\!]D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}$
$\mathcal{G}_2 :: \Gamma_1, \mathcal{S}_r[\![\oplus\Delta]\!]E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_2$ and $\mathcal{E}$
$\Gamma_1, D \,\&\, E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by & on $\mathcal{G}_1$ and $\mathcal{G}_2$
Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{S}_r[\![\Delta_3]\!]F, D}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_1$$

$\mathcal{G} :: \Gamma_1, \mathcal{S}_r[\![\Delta_3]\!]F, D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_1$ on $\mathcal{G}$
Left-Commutative Case:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{S}_r[\![\Delta_3]\!]F, E}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_2$$

$\mathcal{G} :: \Gamma_1, \mathcal{S}_r[\![\Delta_3]\!]F, E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_2$ on $\mathcal{G}$

Left-Commutative Case:

$$\mathcal{D} = \dfrac{\mathcal{D}_1 :: \Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{Q}[\![\Delta]\!]E, D}{\Gamma_1, \mathcal{Q}[\![\Delta_1, ?A]\!]C, \mathcal{Q}[\![\Delta, ?D]\!]E}\ ?$$

$\mathcal{G} :: \Gamma_1, \mathcal{Q}[\![\Delta]\!]E, D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 7 on $\mathcal{D}_1$ and $\mathcal{E}$
$\Gamma_1, \mathcal{Q}[\![\Delta, ?D]\!]E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by ? on $\mathcal{G}$

8. By induction on $\mathcal{E}$.
Impossible Cases: Ax, **1**, !
Key Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, B, A^\perp}{\Gamma_2, \mathcal{Q}[\![?B]\!]A^\perp}\ ?$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta_1]\!]C, B$ by i.h. 1 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta_1, ?B]\!]C$ by ? on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, [*]}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \perp}\ \perp$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [*], \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \perp, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\perp$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Delta_3, D \quad \mathcal{E}_2 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \Delta_4, E}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, [\Delta_3]\Delta_4, D \otimes E}\ \otimes$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \Delta_4, E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma_1, \Gamma_2, [\Delta_3]\Delta_4, D \otimes E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\otimes$ on $\mathcal{E}_1$ and $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, [D]E}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, D \parr E}\ \parr$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, [D]E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, D \parr E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\parr$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{S}_l[\![\oplus\Delta]\!]D \quad \mathcal{E}_2 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{S}_r[\![\oplus\Delta]\!]E}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \oplus\Delta, D \,\&\, E}\ \&$$

$\mathcal{G}_1 :: \Gamma_1, \Gamma_2, \mathcal{S}_l[\![\oplus\Delta]\!]D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\mathcal{G}_2 :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\oplus\Delta]\!]E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma_1, \Gamma_2, D \,\&\, E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\&$ on $\mathcal{G}_1$ and $\mathcal{G}_2$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{S}_r[\![\Delta_3]\!]F, D}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_1$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3]\!]F, D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_1$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{S}_r[\![\Delta_3]\!]F, E}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F}\ \oplus_2$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3]\!]F, E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{S}_r[\![\Delta_3, D \oplus E]\!]F, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by $\oplus_2$ on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{Q}[\![\Delta]\!]E, D}{\Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, \mathcal{Q}[\![\Delta, ?D]\!]E}\ ?$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta]\!]E, D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta, ?D]\!]E, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by ? on $\mathcal{G}$

Right-Commutative Case:

$$\mathcal{E} = \dfrac{\mathcal{E}_1 :: \Gamma_2, \mathcal{Q}[\![\Delta_2]\!]A^\perp, D}{\Gamma_2, \mathcal{Q}[\![\Delta_2, ?D]\!]A^\perp}\ ?$$

$\mathcal{G} :: \Gamma_1, \Gamma_2, D, \mathcal{Q}[\![\Delta_1, \Delta_2]\!]C$ by i.h. 8 on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma_1, \Gamma_2, \mathcal{Q}[\![\Delta_1, \Delta_2, ?D]\!]C$ by ? on $\mathcal{G}$

C Proof of Lemma 12

Lemma 12. Let $P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C$. Then, there exists P' such that

$$\frac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C} \oplus_l$$

(similar result for the right case, with $\oplus_r$).

Let $P \vdash \Gamma, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C$. Then, there exists P' such that

$$\frac{P' \vdash \Gamma, \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P' \vdash \Gamma, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C} \ ?$$

Proof. We look at the case of $\oplus_l$ in detail. The other cases are very similar. We show that, besides the rule introducing $A \oplus B$, the proof does not change its structure at all. This is done by induction on the size of the proof by showing that $\oplus_l$ can permute down with any of the other rules.

- $\mathbf{1}/Ax$. Not applicable.
- $\bot$. If the last applied rule is $\bot$, then it must be such that:

$$\frac{P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, [*]}{w().P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, w : \bot} \bot$$

  By induction hypothesis, there exists P' such that

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, [*]}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, [*]} \oplus_l}{w().x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, w : \bot} \bot$$

  Clearly, we can make the two rules commute, obtaining:

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, [*]}{w().P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, w : \bot} \bot}{x[\mathsf{inl}].w().P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, w : \bot} \oplus_l$$

- $\parr$. If the last applied rule is $\parr$, then it must be such that:

$$\frac{P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, [y : D]w : E}{w(y).P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, w : D \parr E} \parr$$

  By induction hypothesis, there exists P' such that

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, [y : D]w : E}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, [y : D]w : E} \oplus_l}{w(y).x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, w : D \parr E} \parr$$

  Clearly, we can make the two rules commute, obtaining:

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, [y : D]w : E}{w(y).P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!]z : C, x : A, w : D \parr E} \parr}{x[\mathsf{inl}].w(y).P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!]z : C, w : D \parr E} \oplus_l$$

– $\otimes$. If the last applied rule is $\otimes$, then it must be such that:

$$\frac{P \vdash \Delta_1, y : D \quad Q \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \Delta_2, w : E}{w[y \triangleright P].Q \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, [\Delta_1]\Delta_2, w : D \otimes E} \otimes$$

By induction hypothesis, there exists Q' such that

$$\frac{P \vdash \Delta_1, y : D \quad \dfrac{Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \Delta_2, w : E}{x[\mathsf{inl}].Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \Delta_2, w : E} \oplus_l}{w[y \triangleright P].x[\mathsf{inl}].Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, [\Delta_1]\Delta_2, w : D \otimes E} \otimes$$

Clearly, we can make the two rules commute, obtaining:

$$\frac{\dfrac{P \vdash \Delta_1, y : D \quad Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \Delta_2, w : E}{w[y \triangleright P].Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, [\Delta_1]\Delta_2, w : D \otimes E} \otimes}{x[\mathsf{inl}].w[y \triangleright P].Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, [\Delta_1]\Delta_2, w : D \otimes E} \oplus_l$$

– $\oplus_l/\oplus_r$. Here we can have three subcases. If the last applied rule is $\oplus_l$ and it is indeed working on endpoint x then we are done (this is the base case). Otherwise, rule $\oplus_l$ may be working either on a formulas inside the box $\mathcal{S}_l[\![\Delta]\!] z : C$ (in Δ) or in some other box. In both cases, we proceed as usual. Below, we look at the case where the formula is in another box.

$$\frac{P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\Delta']\!] y : F, w : D}{w[\mathsf{inl}].P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!] y : F} \oplus_l$$

By induction hypothesis, there exists P' such that

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_l[\![\Delta']\!] y : F, w : D}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\Delta']\!] y : F, w : D} \oplus_l}{w[\mathsf{inl}].x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!] y : F} \oplus_l$$

Clearly, we can make the two rules commute, obtaining:

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_l[\![\Delta']\!] y : F, w : D}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!] y : F} \oplus_l}{w[\mathsf{inl}].x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\Delta', w : D \oplus E]\!] y : F} \oplus_l$$

– $\&$. In this case, we need to apply the induction hypothesis to both branches of the rule $\&$. If that is the last applied rule, then it must have the following format:

$$\frac{P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\oplus\Delta]\!] w : C \quad Q \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_r[\![\oplus\Delta]\!] w : D}{w.\mathsf{case}(P, Q) \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \oplus\Delta, w : C \,\&\, D} \&$$

By induction hypothesis, there exist P' and Q' such that:

$$\frac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_l[\![\oplus\Delta]\!] w : C}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_l[\![\oplus\Delta]\!] w : C} \oplus_l \quad \dfrac{Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_r[\![\oplus\Delta]\!] w : D}{x[\mathsf{inl}].Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{S}_r[\![\oplus\Delta]\!] w : D} \oplus}{w.\mathsf{case}(x[\mathsf{inl}].P', x[\mathsf{inl}].Q') \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \oplus\Delta, w : C \,\&\, D} \&$$

As in the previous cases, we can now commute the two rules:

$$\dfrac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_l[\![\oplus\Delta]\!] w : C \quad Q' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{S}_r[\![\oplus\Delta]\!] w : D}{w.\mathsf{case}(P', Q') \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \oplus\Delta, w : C \,\&\, D}\,\&}{x[\mathsf{inl}].w.\mathsf{case}(P', Q') \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \oplus\Delta, w : C \,\&\, D}\,\oplus_l$$

– ?. In the case of ?, it must be the case that:

$$\dfrac{P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{Q}[\![\Delta]\!] z : C, x : A}{?x[y].P \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{Q}[\![\Delta, x : ?A]\!] z : C}\,?$$

By induction hypothesis, there exists P' such that

$$\dfrac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{Q}[\![\Delta]\!] t : C, y : A}{x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{Q}[\![\Delta]\!] t : C, y : A}\,\oplus_l}{?w[y].x[\mathsf{inl}].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{Q}[\![\Delta, w : ?A]\!] t : C}\,?$$

Finally, we can swap the two rules and obtain:

$$\dfrac{\dfrac{P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{Q}[\![\Delta]\!] t : C, y : A}{?w[y].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta]\!] z : C, x : A, \mathcal{Q}[\![\Delta, w : ?A]\!] t : C}\,?}{x[\mathsf{inl}].?w[y].P' \vdash \Gamma, \mathcal{S}_l[\![\Delta, x : A \oplus B]\!] z : C, \mathcal{Q}[\![\Delta, w : ?A]\!] t : C}\,\oplus_l$$

– !. Not applicable.

D Proof of Lemma 14 (part 1)

Lemma 14 (part 1). Let $\mathcal{D} :: P \vdash_{p_2} \Gamma, [y : A]x : B$ such that $\mathcal{D}$ is $\parr$-free. Then, there exists a $\parr$-free proof $\mathcal{E}$ and Q such that $\mathcal{E} :: Q \vdash_{p_2} \Gamma, x : A \parr B$.

Proof. We proceed by induction on the size of the proof, and look at the last applied rule. Below, we only report the key case. The full proof is in Appendix D

- $\mathbf{1}$/Ax/$\parr$ /!. Not applicable.
- $\bot$. In this case, we have:

$$\frac{P' \vdash \Gamma, [y : A]x : B, [*]}{z().P' \vdash \Gamma, [y : A]x : B, z : \bot}\ \bot$$

  By induction hypothesis, since $\mathcal{D}$ in $\mathcal{D} :: P'$ is $\parr$-free, by applying $\bot$ again, we obtain:

$$\frac{Q' \vdash \Gamma, x : A \parr B, [*]}{z().Q' \vdash \Gamma, x : A \parr B, z : \bot}\ \bot$$

- $\otimes\parr^{\bot}_{p}$. If the last applied rule is $\otimes\parr^{\bot}_{p}$, we have two cases:
  - $[y : A]x : B$ is not principle for $\otimes\parr^{\bot}_{p}$:

$$\frac{\cdot \vdash_{p_2} \Delta_1, \{x_i : A_i\}_i, y : C \quad \cdot \vdash_{p_2} \Gamma, [y : A]x : B, \Delta_2, \{x_i : B_i\}_i, y : D}{\cdot \vdash_{p_2} \Gamma, [y : A]x : B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr^{\bot}_{p}$$

    By induction hypothesis, since our proof is $\parr$-free, by applying $\otimes\parr^{\bot}_{p}$ again, we obtain:

$$\frac{\cdot \vdash_{p_2} \Delta_1, \{x_i : A_i\}_i, y : C \quad \cdot \vdash_{p_2} \Gamma, x : A \parr B, \Delta_2, \{x_i : B_i\}_i, y : D}{\cdot \vdash_{p_2} \Gamma, x : A \parr B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr^{\bot}_{p}$$

  - If $[y : A]x : B$ is indeed modified by by $\otimes\parr^{\bot}_{p}$, then we have a base case:

$$\frac{\cdot \vdash_{p_1} \Delta_1, A, \{x_i : A_i\}_i, y : C \quad \cdot \vdash_{p_1} \Gamma, B, \Delta_2, \{x_i : B_i\}_i, y : D}{\cdot \vdash_{p_1} \Gamma, [y : A]x : B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr^{\bot}_{p}$$

    Clearly, we can change the rule $\otimes\parr^{\bot}_{p}$ obtaining:

$$\frac{\cdot \vdash_{p_1} \Delta_1, A, \{x_i : A_i\}_i, y : C \quad \cdot \vdash_{p_1} \Gamma, B, \Delta_2, \{x_i : B_i\}_i, y : D}{\cdot \vdash_{p_1} \Gamma, x : A \parr B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr^{\bot}_{p}$$

- $\&$. In this case, we have:

$$\frac{P' \vdash \Gamma, [y : A]x : B, \mathcal{S}_l[\![\oplus\Delta]\!]x : A \quad Q' \vdash \Gamma, [y : A]x : B, \mathcal{S}_r[\![\oplus\Delta]\!]x : B}{x.\mathsf{case}(P', Q') \vdash \Gamma, [y : A]x : B, \oplus\Delta, x : A \,\&\, B}\ \&$$

  By induction hypothesis, since both proofs in the premise of $\&$ are $\parr$-free, by applying $\&$ again, we obtain:

$$\frac{P' \vdash \Gamma, x : A \parr B, \mathcal{S}_l[\![\oplus\Delta]\!]x : A \quad Q' \vdash \Gamma, x : A \parr B, \mathcal{S}_r[\![\oplus\Delta]\!]x : B}{x.\mathsf{case}(P', Q') \vdash \Gamma, x : A \parr B, \oplus\Delta, x : A \,\&\, B}\ \&$$

– $\oplus_l$ (similar for $\oplus_r$). In this case, we have:

$$\frac{P' \vdash \Gamma, [y : A]x : B, \mathcal{S}_l[\![\Delta]\!]z : C, w : A}{w[\mathsf{inl}].P' \vdash \Gamma, [y : A]x : B, \mathcal{S}_l[\![\Delta, w : A \oplus B]\!]z : C}\ \oplus_l$$

By induction hypothesis, since $\mathcal{D}$ in $\mathcal{D} :: P'$ is $\parr$-free, by applying $\oplus_l$ again, we obtain:

$$\frac{Q' \vdash \Gamma, x : A \parr B, \mathcal{S}_l[\![\Delta]\!]z : C, w : A}{w[\mathsf{inl}].Q' \vdash \Gamma, x : A \parr B, \mathcal{S}_l[\![\Delta, w : A \oplus B]\!]z : C}\ \oplus_l$$

– ?. In this case, we have:

$$\frac{P \vdash \Gamma, [y : A]x : B, \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P \vdash \Gamma, [y : A]x : B, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C}\ ?$$

By induction hypothesis, since $\mathcal{D}$ in $\mathcal{D} :: P'$ is $\parr$-free, by applying ? again, we obtain:

$$\frac{P \vdash \Gamma, x : A \parr B, \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P \vdash \Gamma, x : A \parr B, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C}\ ?$$

E Proof of Lemma 14 (part 2)

Lemma 14 (part2). Let $\mathcal{D} :: P \vdash_{p_2} \Gamma, [*]$ such that $\mathcal{D}$ is $\bot$-free. Then, there exists a $\bot$-free proof $\mathcal{E}$ and Q such that $\mathcal{E} :: Q \vdash_{p_2} \Gamma, x : \bot$.

Proof. Similar to that of previous Lemma.

- Ax/ $\bot$ /!. Not applicable.
- $\mathbf{1}\bot^{\bot}$. This is the base case. If Γ contains $[*]$ then we do nothing. Otherwise,

$$\frac{}{\cdot \vdash_{p_1} \{x_i : \bot\}_i, y : \mathbf{1}, [*]}\ \mathbf{1}\bot^{\bot}$$

  which can be replaced by

$$\frac{}{\cdot \vdash_{p_1} \{x_i : \bot\}_i, y : \mathbf{1}, x : \bot}\ \mathbf{1}\bot^{\bot}$$

- $\otimes\parr_p^{\bot}$. In this case, we have:

$$\frac{\cdot \vdash_{p_1} \Delta_1, \{x_i : A_i\}_i, y : C \quad \cdot \vdash_{p_1} \Gamma, [*], \Delta_2, \{x_i : B_i\}_i, y : D}{\cdot \vdash_{p_1} \Gamma, [y : A]x : B, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr_p^{\bot}$$

  By induction hypothesis, since our proof is $\bot$-free, by applying $\otimes\parr_p^{\bot}$ again, we obtain:

$$\frac{\cdot \vdash_{p_1} \Delta_1, \{x_i : A_i\}_i, y : C \quad \cdot \vdash_{p_1} \Gamma, x : \bot, \Delta_2, \{x_i : B_i\}_i, y : D}{\cdot \vdash_{p_1} \Gamma, x : \bot, [\Delta_1]\Delta_2, \{x_i : A_i \parr B_i\}_i, y : C \otimes D}\ \otimes\parr_p^{\bot}$$

- $\parr$. In this case, we have:

$$\frac{P \vdash \Gamma, [*], [y : A]x : B}{x(y).P \vdash \Gamma, [*], x : A \parr B}\ \parr$$

  And, by induction hypothesis,

$$\frac{P \vdash \Gamma, x : \bot, [y : A]x : B}{x(y).P \vdash \Gamma, x : \bot, x : A \parr B}\ \parr$$

- $\&$. In this case, we have:

$$\frac{P' \vdash \Gamma, [*], \mathcal{S}_l[\![\oplus\Delta]\!]x : A \quad Q' \vdash \Gamma, [*], \mathcal{S}_r[\![\oplus\Delta]\!]x : B}{x.\mathsf{case}(P', Q') \vdash \Gamma, [y : A]x : B, \oplus\Delta, x : A \,\&\, B}\ \&$$

  By induction hypothesis, since both proofs in the premise of $\&$ are $\bot$-free, by applying $\&$ again, we obtain:

$$\frac{P' \vdash \Gamma, x : \bot, \mathcal{S}_l[\![\oplus\Delta]\!]x : A \quad Q' \vdash \Gamma, x : \bot, \mathcal{S}_r[\![\oplus\Delta]\!]x : B}{x.\mathsf{case}(P', Q') \vdash \Gamma, x : \bot, \oplus\Delta, x : A \,\&\, B}\ \&$$

– $\oplus_l$ (similar for $\oplus_r$). In this case, we have:

$$\frac{P' \vdash \Gamma, [*], \mathcal{S}_l[\![\Delta]\!]z : C, w : A}{w[\mathsf{inl}].P' \vdash \Gamma, [*], \mathcal{S}_l[\![\Delta, w : A \oplus B]\!]z : C} \oplus_l$$

By induction hypothesis, since $\mathcal{D}$ in $\mathcal{D} :: P'$ is $\bot$-free, by applying $\oplus_l$ again, we obtain:

$$\frac{Q' \vdash \Gamma, x : \bot, \mathcal{S}_l[\![\Delta]\!]z : C, w : A}{w[\mathsf{inl}].Q' \vdash \Gamma, x : \bot, \mathcal{S}_l[\![\Delta, w : A \oplus B]\!]z : C} \oplus_l$$

– ?. In this case, we have:

$$\frac{P \vdash \Gamma, [*], \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P \vdash \Gamma, [*], \mathcal{Q}[\![\Delta, x : ?A]\!]z : C} \ ?$$

By induction hypothesis, since $\mathcal{D}$ in $\mathcal{D} :: P'$ is $\bot$-free, by applying ? again, we obtain:

$$\frac{P \vdash \Gamma, x : \bot, \mathcal{Q}[\![\Delta]\!]z : C, y : A}{?x[y].P \vdash \Gamma, x : \bot, \mathcal{Q}[\![\Delta, x : ?A]\!]z : C} \ ?$$