Forwarders as Process Compatibility, Logically

Marco Carbone[1][0000−0001−9479−2632], Sonia Marin[2][1111−2222−3333−4444], and Carsten Schürmann[1][0000−0003−4793−0099]

IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen, Denmark
{maca,carsten}@itu.dk
University College London, Gower St, London WC1E 6BT, UK
s.marin@ucl.ac.uk

Abstract. Session types define protocols that processes must follow when communicating. The special case of binary session types, i.e. type annotations of protocols between two parties, is known to be in a propositions-as-types correspondence with linear logic. In previous work, we have shown that the generalization to multiparty session types can be expressed either by coherence proofs or by arbiters, processes that act as middleware by forwarding messages according to the given protocol. In this paper, following the propositions-as-types fashion, we generalize arbiters to a logic, which we call forwarder logic, a fragment of classical linear logic still satisfying cut-elimination. Our main result is summarized as follows: forwarders generalize coherence and give an elegant proof-theoretic characterization of multiparty compatibility, a property of concurrent systems guaranteeing that all sent messages are eventually received and no deadlock ever occurs.

Keywords: Session types · Propositions as sessions · Cut elimination

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. [14], are type annotations that ascribe protocols to processes in a concurrent system and 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, e.g., an integer, I will never receive, e.g., 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, session types make sure that the processes are *compatible* and that they exchange messages in the prescribed way for the concurrent system to work correctly. They do that 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*, type annotations of 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 by Wadler [23,24]. 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, for the lack of a better word, the "fabric" through which two processes communicate while abstracting away from practical details, e.g., message delay, message order, or message buffering.

The situation is not as direct for *multiparty session types* [15,16], type annotations for protocols with more than two 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, through which multiple processes communicate, 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 compatible judgements, 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 be modelled formally 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 *asynchronous forwarders* or *forwarders* in short, that provides one possible such characterisation and also guarantees multiparty session compatibility.

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. Eventually, they re-transmit messages only after receiving them, without interpreting, modifying, or computing with them.

In this work, asynchronous forwarders support buffers of unlimited size. This preserves the order of messages from the same sender, i.e., after receiving a message from one party, the forwarder enqueues it until the message is delivered to its destination. Forwarders can be used to explain communication patterns as they occur in practice, such as message routing, proxy services, and runtime monitors for message flows [17].

The meta-theoretic study of forwarders allows us to conclude that there is a correspondence between forwarders and multiparty compatibility. Forwarders are stronger than coherence and can be used to guide the communication of multiple processes similarly to the multi-cut elimination in [5] and so they are a way to justify multiparty compatibility. The reverse direction also holds, i.e., if a multiparty session of processes is compatible, they can be linked by a forwarder; in such a way, forwarders as a logic provide a proof theory for multiparty compatibility. In this paper, we also show that forwarders can safely be composed through cut elimination, which allows us to combine the fabric between two concurrent systems (figuring arbitrary many processes).

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

- a logical characterisation of *forwarders* (§ 4);
- a reductive operational semantics based on cut-elimination (§ 5);
- a correspondence between multiparty compatibility and forwarders, generalising coherence: every forwarder guarantees correct multiparty communication (§ 6) and any compatible multiparty session can be emulated by a forwarder (§ 7).

Additionally, § 2 gently introduces the main concepts on an example and § 3 recaps the definitions of types, processes, and CP-typing, while § 8 discusses related and future work and concluding remarks are in § 9.

2 Preview

We now proceed with a gentle introduction to asynchronous forwarders by informally describing the classic *2-buyer protocol* [15,16], 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 which gives a precise description of how each endpoint has to act.

$$\begin{gathered} b_1 : \mathbf{name} \otimes \mathbf{cost}^{\perp} \parr \mathbf{cost} \otimes \mathbf{1} \quad b_2 : \mathbf{cost}^{\perp} \parr \mathbf{cost}^{\perp} \parr ((\mathbf{addr} \otimes \mathbf{1}) \oplus \mathbf{1}) \\ s : \mathbf{name}^{\perp} \parr \mathbf{cost} \otimes \mathbf{cost} \otimes ((\mathbf{addr}^{\perp} \parr \perp) \mathbin{\&} \perp) \end{gathered} \tag{1}$$

For example, $b_1 : \mathbf{name} \otimes \mathbf{cost}^{\perp} \parr \mathbf{cost} \otimes \mathbf{1}$ 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.

Any three processes who respectively use endpoints b_1, b_2, and s according to the type specifications above are going to execute this protocol correctly because these type specifications are *compatible*. In a binary setting (only two endpoints communicating to each other) compatibility is usually expressed by type duality: the dual of a type is the type obtained by inverting every output

with an input, and vice-versa. In a multiparty setting, such compatibility can be expressed as *coherence* [5] but, we will argue, also with asynchronous forwarders. The core idea of multiparty compatibility is to establish for each output which endpoint should receive it. We propose to base compatibility on whether there exists a process (*forwarder*) able to glue the (duals of the) types of the different endpoints, e.g., in the 2-buyers case, we can establish compatibility if there is a process typable in the context formed by the duals of the types in (1). Indeed, such a forwarder process exists and could have the following simple behaviour: the request containing the book name is received over some endpoint connected to b_1 and then forwarded over an endpoint connected to s, then the same is done over b_1 and b_2 for the amount, and so on. We observe that such a process, provided that it indeed respects the dual of the types in (1), could still have many different variations. For example, the first send can happen at a later point rather than immediately after the request has been received. Yet, a forwarder cannot be any process: it must be such that i) anything that has been received is eventually sent, ii) anything that is sent must have been previously received, and iii) the order of messages between any two points must be preserved. Our theory of forwarders captures precisely such requirements.

Although the notion of coherence also satisfies these properties, by capturing all and only requirements i), ii), and iii), we can model the composition of processes that cannot be captured by coherence. Consider for example two endpoints x and y willing to communicate with the following protocol – called a *criss-cross*: they both send a message to each other, and then the messages are received, according to the following types

$$x : \mathbf{name} \otimes \mathbf{cost} \parr \mathbf{1} \qquad\qquad y : \mathbf{cost}^{\perp} \otimes \mathbf{name}^{\perp} \parr \perp$$

Such protocol leads to no error (assuming processes implement an asynchronous semantics), still the two types above are not coherent [5]. On the other hand, we can easily write a forwarder typable in the context $x : \mathbf{name}^{\perp} \parr \mathbf{cost}^{\perp} \otimes \perp, y : \mathbf{cost} \parr \mathbf{name} \otimes \mathbf{1}$ formed by their duals, i.e., a process that first receives on both x and y and then forwards the received messages over to y and x, respectively.

3 Preliminaries: CP and Classical Linear Logic

In order to make the presentation of asynchronous forwarders easier to comprehend, we give an introduction to the proposition-as-sessions approach [24]. This comprises the syntax of types and processes and the interpretation of processes as sequent proofs in classical linear logic (CLL). In the interest of space, we restrict this presentation to the multiplicative fragment. The treatment of the additive and exponential fragments can be found in the appendix.

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:

$$A ::= \quad a \mid a^{\perp} \mid 1 \mid \perp \mid (A \otimes A) \mid (A \parr A) \tag{2}$$

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

$$\frac{P \vdash \Delta_1, y : A_1 \quad Q \vdash \Delta_2, x : A_2}{x[y \triangleright P].Q \vdash \Delta_1, \Delta_2, x : A_1 \otimes A_2}\ \otimes \qquad \frac{P \vdash \Delta, y : A_1, x : A_2}{x(y).P \vdash \Delta, x : A_1 \parr A_2}\ \parr$$

Fig. 1. Sequent Calculus (Multiplicative Fragment) for CP and Classical Linear Logic

Atoms a and negated atoms $a^{\perp}$ are basic dual types. Types $\mathbf{1}$ and $\perp$ denote an endpoint that must close with 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 continues as B.

Duality. Operators can be grouped in pairs of duals that reflect the input-output duality. Consequently, standard duality $(\cdot)^{\perp}$ on types is inductively defined as:

$$(a^{\perp})^{\perp} = a \qquad \mathbf{1}^{\perp} = \perp \qquad (A \otimes B)^{\perp} = A^{\perp} \parr B^{\perp}$$

Processes. We use a standard language of *processes* to represent communicating entities (including forwarders) which is a variant of the π-calculus [20] with specific communication primitives as usually done for session calculi. Moreover, given that the theory of this paper is based on the proposition-as-sessions correspondence with CLL, we adopt a syntax akin to that of Wadler [24]:

$P, Q ::=$	$x \leftrightarrow y$	(link)	$(\boldsymbol{\nu} xy)\,(P \mid Q)$	(parallel)
	$x().P$	(wait)	$x[]$	(close)
	$x(y).P$	(input)	$x[y \triangleright P].Q$	(output)

A link $x \leftrightarrow y$ is a binary forwarder, i.e., a process that forwards any communication between endpoints x and 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. Note that we use endpoints instead of channels [22]. 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 [24] 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 [21], hence the output term $x[y \triangleright P].Q$ is a compact version of the π-calculus term $(\nu y)\,\overline{x}y.(P \mid Q)$.

CP-typing. As shown by Wadler [24], among all of the many process expressions one can write, classical linear logic (CLL) characterises a subset that is well-behaved, i.e. they satisfy deadlock freedom and session fidelity.

Judgements are defined as $P \vdash \Delta$ with Δ a set of named types, i.e., $\Delta ::= \varnothing \mid x : A, \Delta$. The system called CP, in Figure 1, uses CLL to type processes.

CP can be extended with a structural rule for defining composition of processes which corresponds to the Cut rule from classical linear logic:

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

In linear logic this rule is admissible, i.e., the CLL derivations of the two premises can be combined into a derivation of the conclusion with no occurrence of the Cut rule. Moreover, this is a constructive procedure, called *cut-elimination*, meaning that the proof with cut is inductively transformed into a proof without cut. The strength of the proposition-as-type correspondence stems from the fact that it carries on to the proof level, as it was shown that the cut-elimination steps correspond to reductions in the π-calculus [3,23].

4 Asynchronous Forwarders

Following a proposition-as-types approach, we aim at a restriction of CP such that derivable judgements are inhabited by forwarder processes only. For the sake of clarity, we start our development focusing on the multiplicative fragment of linear logic (rules Ax, $\mathbf{1}$, $\perp$, $\otimes$, and $\parr$). We give a full extension to additives and exponentials in the appendix.

Forwarders form a subclass of processes. Our development focuses exclusively on those processes that are also typable in classical linear logic. I.e., our goal is to identify all those CP processes that are also forwarders. In order to do so, we must add further information in the standard CP contexts.

Extended types. To type forwarders, we extend the syntax of types (formulas) with annotations that make explicit where messages should be forwarded from and to. This is similar to local types $!\mathsf{p}.T$ and $?\mathsf{p}.T$ [8] expressing an output and an input to and from role p respectively. Intuitively, the reason for that is that in order to achieve cut elimination for forwarders, we need to store more information on the typing contexts (and endpoint types). We will see this in the section. The meaning of each operator remains the same as in CP as shown in the previous section. The definition of duality is also the same.

$$B ::= \quad a \quad \mid \quad a^{\perp} \mid \quad \mathbf{1}^{u_1,\ldots,u_n} \mid \quad \perp^{u} \mid \quad (A \otimes^{u} B) \mid \quad (A \parr^{u} B)$$

The left hand side A of $\otimes$ and $\parr$ are not annotated, as in (4), and become dynamically labelled when needed by the typing routine. Note that in this section we only consider the subset of A without &, $\oplus$, !, and ?. We will see that units demonstrate some *gathering* behaviour which explains the need to annotate $\mathbf{1}$ with a non-empty list of an arbitrary number of distinct names. We may write $\tilde{u}$ for $u_1, \ldots, u_n$ when the size of the list is irrelevant.

We define a bidirectional map between annotated and non-annotated types. For a proposition A defined as in (4), we write $A(x)$ for the formula obtained by annotating every operator in A with the name x. Conversely, we write $\llcorner B \lrcorner$ for the formula obtained from B by removing all the annotations.

Contexts. What we need is to be able to enforce the main features that characterise a forwarder, namely i) any received message must be forwarded, ii) any message that is going to be sent must be something that has been previously received, and iii) the order of messages between any two points must be preserved. In order to enforce these requirements, we add more information to the standard CP judgement. For example, let us consider the input process $x(y).P$. In CP, the typing environment for such process must be such that endpoint x has type $A \parr B$ such that P has type $y : A$ and $x : B$. However, the context is not telling us at all that y is actually a message that has been received and, as such, it should not be used by P for further communications but just forwarded over some other channel. In order to remember this fact when we type the subprocess P, we actually insert $y : A$ into a queue that belongs to endpoint x where we put all the types of messages received over it. I.e., when typing P, the context will contain $[\![\Psi]\!][^u y : A]x : B$. That still means that x must have type B and y must have type A in P, but also that $y : A$ has been received over x (it is in x's queue) and we are intending to forward it to endpoint u. Moreover, Ψ contains the types of messages that have been previously received over x. The forwarders behave asynchronously. They can input arbitrarily many messages, which are enqueued at the arrival point, without blocking the possibility of producing an output from the same endpoint. This behaviour is captured by the notion of queues of *boxed* messages, i.e. messages that are in-transit.

$$[\![\Psi]\!] ::= \varnothing \mid \quad [^u *][\![\Psi]\!] \mid \quad [^u y : A][\![\Psi]\!]$$

A queue element $[^u x : A]$ expresses that some name x of type A has been received and will need to later be forwarded to endpoint u. Similarly, $[^u *]$ indicates that a request for closing a session has been received and must be forwarded to u.

The notation $[\![\Psi]\!]$ is convenient as when needed we can refer to Ψ, the set of all the elements that appear in $[\![\Psi]\!]$ removing the annotated brackets. For example, for $[\![\Psi]\!] = [^v x : A_1][^u y : A_2][^w *]$, then $\Psi = \{y : A_2, x : A_1, *\}$.

The order of messages needing to be forwarded to *independent* endpoints is irrelevant. Hence, we consider queue $[\![\Psi_1]\!][^x \ldots][^y \ldots][\![\Psi_2]\!]$ equivalent to queue $[\![\Psi_1]\!][^y \ldots][^x \ldots][\![\Psi_2]\!]$ whenever $x \neq y$. For a given endpoint x however the order of two messages $[^x \ldots][^x \ldots]$ is crucial and must be maintained throughout the forwarding. By attaching a queue to each endpoint we get a typing context.

$$\Gamma ::= \quad \varnothing \quad \mid \quad \Gamma, [\![\Psi]\!]x : B \quad \mid \quad \Gamma, [\![\Psi]\!]x : \cdot$$

The element $[\![\Psi]\!]x : B$ of a context Γ indicates that the messages in $[\![\Psi]\!]$ have been received at endpoint x. The special case $[\![\Psi]\!]x : \cdot$ is denoting the situation when endpoint x no longer needs to be used for communication, but still has a non-empty queue of messages to forward.

Judgements and rules. A judgement denoted by $P \Vdash \Gamma$ types the forwarder processes P that connect the endpoints in Γ. The rules enforce the asynchronous forwarding behaviour by adding elements to queues using rules for $\bot$ and $\parr$, which forces them to be later removed from queues by the corresponding rules for $\mathbf{1}$ and $\otimes$. The rules are reported in Fig. 2.

$$\frac{}{x \leftrightarrow y \Vdash x : a^{\perp}, y : a}\ \text{Ax} \qquad \frac{}{x[] \Vdash \{[^{x}*]u_i : \cdot\}_{1 \le i \le n}, x : \mathbf{1}^{\tilde{u}}}\ \mathbf{1}$$

$$\frac{P \Vdash \Gamma, [\![\Psi]\!][^{u}y : A]x : B}{x(y).P \Vdash \Gamma, [\![\Psi]\!]x : A \parr^{u} B}\ \parr \qquad \frac{P \Vdash \Gamma, [\![\Psi]\!][^{u}*]x : \cdot}{x().P \Vdash \Gamma, [\![\Psi]\!]x : \perp^{u}}\ \perp$$

$$\frac{P \Vdash z : A(y)^{\perp}, y : A(z) \quad Q \Vdash \Gamma, [\![\Psi_u]\!]u : C, [\![\Psi_x]\!]x : B}{x[y \triangleright P].Q \Vdash \Gamma, [^{x}z : A^{\perp}][\![\Psi_u]\!]u : C, [\![\Psi_x]\!]x : A \otimes^{u} B}\ \otimes$$

Fig. 2. Forwarder multiplicative rules

Rule Ax is identical to the one of CP. Rules **1** and $\perp$ forward a request to close a session. Rule $\perp$ receives the request on endpoint x and enqueues it as $[^{u}*]$ if it needs to forward it to u. Note that in the premiss of $\perp$ the endpoint is terminated pending the remaining messages in the corresponding queue being dispatched. Eventually all endpoints but one will be terminated in the same manner. Rule **1** will then be applicable. Note that the behaviour of $x().P$ and $x[]$ work as gathering, several terminated endpoints connect to the last active endpoint typed with a **1**. Rules $\otimes$ and $\parr$ forward a message. Rule $\parr$ receives the message $y : A$ and enqueues it as $[^{u}y : A]$ to be forwarded to endpoint u. Dually, rule $\otimes$ applied to a $\otimes^{u}$ sends the message at the top of the queue of endpoint u if it has the dual type. Note that this idea can be generalised to a gathering behaviour where several messages are sent at the same time. Messages will be picked from queues belonging to distinct endpoints, which would require us to annotate the tensor with a list of endpoints. As a consequence, the left premiss of $\otimes$ rule would be a new forwarder consisting of the gathered messages. For the sake of simplicity we discuss this generalisation only in the appendix.

Note how annotations put constraints on how the proof is constructed, e.g., annotating an $x : A \parr B$ with u ensures us that the proof will contain a $\otimes$-rule application on endpoint u at a later point.

Example 1. $P := x(u).y(v).y[u' \triangleright u \leftrightarrow u'].x[v' \triangleright v' \leftrightarrow v].x().y[]$ is one of the forwarders that can prove the compatibility of the types involved in the criss-cross protocol (in § 2), as illustrated by the derivation below in forwarder logic.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{F_1 := u \leftrightarrow u'}{\Vdash u : \mathbf{name}^{\perp}, u' : \mathbf{name}}\ \text{Ax} \quad \dfrac{\dfrac{F_2 := v' \leftrightarrow v}{\Vdash v' : \mathbf{cost}^{\perp}, v : \mathbf{cost}}\ \text{Ax} \quad \dfrac{\dfrac{F_3 := x().y[]}{y[] \Vdash [^{y}*]x : \cdot, y : \mathbf{1}}\ \mathbf{1}}{F_3 \Vdash x : \perp, y : \mathbf{1}}\ \perp}{x[v' \triangleright F_2].F_3 \Vdash x : \mathbf{cost}^{\perp} \otimes \perp, [^{x}v : \mathbf{cost}]y : \mathbf{1}}\ \otimes}{y[u' \triangleright F_1].x[v' \triangleright F_2].F_3 \Vdash [^{y}u : \mathbf{name}^{\perp}]x : \mathbf{cost}^{\perp} \otimes \perp, [^{x}v : \mathbf{cost}]y : \mathbf{name} \otimes \mathbf{1}}\ \parr}{y(v).y[u' \triangleright F_1].x[v' \triangleright F_2].F_3 \Vdash [^{y}u : \mathbf{name}^{\perp}]x : \mathbf{cost}^{\perp} \otimes \perp, y : \mathbf{cost} \parr \mathbf{name} \otimes \mathbf{1}}\ \parr}{P \Vdash x : \mathbf{name}^{\perp} \parr \mathbf{cost}^{\perp} \otimes \perp, y : \mathbf{cost} \parr \mathbf{name} \otimes \mathbf{1}}$$

We conclude this section by stating that every forwarder is also a CP process, the embedding $\llcorner\cdot\lrcorner$ being extended to contexts as:

$$\llcorner[\![\Psi]\!]x : B, \Gamma\lrcorner = \llcorner[\![\Psi]\!]\lrcorner, x : \llcorner B\lrcorner, \llcorner\Gamma\lrcorner \qquad \llcorner[\![\Psi]\!]x : \cdot, \Gamma\lrcorner = \llcorner[\![\Psi]\!]\lrcorner, \llcorner\Gamma\lrcorner$$

$$\llcorner[^u y : A][\![\Psi]\!]\lrcorner = y : A, \llcorner[\![\Psi]\!]\lrcorner \qquad \llcorner[^u \mathcal{X}][\![\Psi]\!]\lrcorner = \llcorner[^u *][\![\Psi]\!]\lrcorner = \llcorner[\![\Psi]\!]\lrcorner$$

Proposition 2. *Any forwarder is typable in CP, i.e., if* $P \Vdash \Gamma$*, then* $P \vdash \llcorner\Gamma\lrcorner$.

5 Semantics of Asynchronous Forwarders

Our next task is to lay the groundwork that will eventually allow us to establish a semantics for this system via a cut elimination procedure. Adding label to queues and connectives brings along some notational burden, but it is instrumental to complete the cut-elimination proof. However, this notational overhead comes at a price, since the labels need to be meticulously maintained: the cut-elimination procedure does not only simplify the formulas and derivations involved in the cut but also rewrites the labels used within a sequent.

Elements of a context are formulas which (*i*) feature annotated connectives and (*ii*) are equipped with queues of messages to be forwarded; these two new features need to appear in the standard cut rule of linear logic, and, as a consequence, pose new challenges to the cut-elimination procedure. A cut rule must cut two elements of a context of the form $[\![\Psi_1]\!]x : A$ and $[\![\Psi_2]\!]y : A^\perp$. As shown by Proposition 2, in a standard linear logic cut rule, the elements of Ψ_1 and Ψ_2 would freely occur in the conclusion context as independent formulas. However, in forwarder logic, they are attached to x and y, respectively, and they are the type of messages that must be forwarded. Hence, after cutting $x : A$ and $y : A^\perp$ we must attach Ψ_1 and Ψ_2 to other endpoints in the context of the conclusion. This operation is called *distribution*.

Moreover, since queue elements in the rest of the context are also annotated with endpoints, we must handle those referring to x and y, which are doomed to disappear after the cut. The *substitution* operation does this guided by the annotations on the cut formulas.

Both operations *substitution* and *distribution* work on a two dimensional depiction of the cut judgements $\Gamma_1, [\![\Psi_1]\!]x : A$ and $\Gamma_2, [\![\Psi_2]\!]y : A^\perp$ as follows:

$$\left\{ \begin{array}{lcl} \Gamma_1 & \triangleright & [\![\Psi_1]\!]x : A \\ \Gamma_2 & \triangleright & [\![\Psi_2]\!]y : A^\perp \end{array} \right\} \tag{3}$$

We use the $\triangleright$ to single out the cut-formulas and make the rules that we introduce below more readable. $\triangleright$ should not be confused with derivability in intuitionistic logic. Moreover, when we need to single out an operator, for instance to substitute a different name for its annotation, we use the notation $B\{\otimes^u\}$, this indicates the first (leftmost) occurrence of the symbol $\otimes^u$ within proposition B.

Example 3. The proposition $B := (a \otimes b) \otimes^u (c \parr d) \otimes^u e$ is a well-formed annotated type. We would write $B\{\otimes^u\}$ to point to the leftmost occurrence of $\otimes^u$. Consequently $B\{\otimes^x\}$ would denote the formula $(a \,\&\, b) \otimes^x (c \parr d) \otimes^u e$. On the other hand, $B[x/u]$ would indicate a global substitution of x for u, namely the formula $(a \otimes b) \otimes^x (c \parr d) \otimes^x e$.

Distribution. Given a pair of contexts depicted as in (3), we proceed by distributing each element of the queues $[\![\Psi_1]\!]$ and $[\![\Psi_2]\!]$ to queues in Γ_2 and Γ_1 respectively. Indeed, elements in $[\![\Psi_1]\!]$ ($[\![\Psi_2]\!]$) have been received at endpoint x (y), but when executing the cut x and y disappear and only the other endpoints in Γ_1 and Γ_2 remain. Messages in Ψ_1 (Ψ_2) need now to come from an available endpoint from Γ_2 (Γ_1). We always distribute by picking the top element (in our notation below, a box labelled with d) of one of the queues and since the cut rule is symmetric, we pick from Ψ_1.

We start with equipping each queue on the left with a $\bullet$ marker that ensures that the messages from successive applications of the distr rule preserve their respective orders. Then, assuming that part of the queues have already been distributed, the following is the distr rewriting that allows us to distribute the top element of the queue in front of endpoint x to endpoint c in Γ_2

$$\left\{\begin{array}{lcl} \Gamma_1, [\![\Psi_{2d}]\!] \bullet [\![\Psi_d]\!] d : D\{\otimes^x\} & \triangleright & [^d b : B][\![\Psi_1]\!] x : A \\ \Gamma_2, [\![\Psi_{1c}]\!] \bullet [\![\Psi_c]\!] c : C & \triangleright & [\![\Psi_2]\!] y : A^\perp \end{array}\right\} \rightarrow_{\mathsf{distr}}$$
$$\left\{\begin{array}{lcl} \Gamma_1, [\![\Psi_{2d}]\!] \bullet [\![\Psi_d]\!] d : D\{\otimes^c\} & \triangleright & [\![\Psi_1]\!] x : A \\ \Gamma_2, [\![\Psi_{1c}]\!] [^d b : B] \bullet [\![\Psi_c]\!] c : C & \triangleright & [\![\Psi_2]\!] y : A^\perp \end{array}\right\}$$

An application of this rule yields again two sequents. We move $[^d b : B]$ from the queue in front of $x : A$ to endpoint c and the left hand side of $\triangleright$ in the top sequent is updated to reflect that d has one fewer communication link with x and communicates with channel c instead, i.e., $d : D\{\otimes^x\}$ becomes $d : D\{\otimes^c\}$. Eventually, when both queues $[\![\Psi_1]\!]$ and $[\![\Psi_2]\!]$ have been distributed, the obsolete $\bullet$ markers are removed and the process enters the next phase, substitution.

Note that the distr rule is not deterministic, in that the rule does not uniquely determine, to which endpoint the element $[^d b : B]$ is distributed to. Hence, each application of distr will determine a particular cut-rule.

Substitution. After distribution, the cut-formulas are free from their respective queues which have been placed over endpoints in the rest of the contexts. The substitution operation is now going to take care of all those occurrences of x and y in the contexts that still must disappear by exploiting the annotations on the cut-formula. We start from a configuration of the following form:

$$\left\{\begin{array}{lcl} \Gamma_1 & \triangleright & x : A \\ \Gamma_2 & \triangleright & y : A^\perp \end{array}\right\}$$

The substitution operation is defined inductively following the respective structures of $x : A$ and $y : A^\perp$. We distinguish three cases, the $\otimes/\parr$ case, the atomic case, and the unit case.

Substitution for $\otimes$ and $\parr$. Assume that the cut formulas are of the form $x : A \otimes^b D$ and $y : A^\perp \parr^c D^\perp$. This enforces that, in the bottom left context (Γ_2), the endpoint c must always be typed by a proposition that contains a $\otimes^y$. The y will need to be substituted by $\otimes^b$. In the top left context (Γ_1) however, there can be two scenarios: either the endpoint b has an element such as $[^x a : A^\perp]$ in its queue, i.e. typed with the dual of A and that needs to be forwarded to x, or it has a $\parr^x$ within its type. In both cases, x needs to be substituted by c.

In the first case, it gives the following rewriting

$$\left\{\begin{array}{lcl} \Gamma_1, [^x a : A^\perp][\![\Psi_b]\!]b : B & \triangleright & x : A \otimes^b D \\ \Gamma_2, [\![\Psi_c]\!]c : C\{\otimes^y\} & \triangleright & y : A^\perp \parr^c D^\perp \end{array}\right\} \rightarrow_{\mathsf{subst}}$$
$$\left\{\begin{array}{lcl} \Gamma_1, [^c a : A^\perp][\![\Psi_b]\!]b : B & \triangleright & x : D \\ \Gamma_2, [\![\Psi_c]\!]c : C\{\otimes^b\} & \triangleright & y : D^\perp \end{array}\right\}$$

In the second case, it gives a similar rewriting

$$\left\{\begin{array}{lcl} \Gamma_1, [\![\Psi_b]\!]b : B\{\parr^x\} & \triangleright & x : A \otimes^b D \\ \Gamma_2, [\![\Psi_c]\!]c : C\{\otimes^y\} & \triangleright & y : A^\perp \parr^c D^\perp \end{array}\right\} \rightarrow_{\mathsf{subst}}$$
$$\left\{\begin{array}{lcl} \Gamma_1, [\![\Psi_b]\!]b : B\{\parr^c\} & \triangleright & x : D \\ \Gamma_2, [\![\Psi_c]\!]c : C\{\otimes^b\} & \triangleright & y : D^\perp \end{array}\right\}$$

Substitution for atoms. Substituting atoms is straightforward and completes the substitutions sequence, by producing the standard judgement.

$$\left\{\begin{array}{lcl} \Gamma_1 & \triangleright & x : a \\ \Gamma_2 & \triangleright & y : a^\perp \end{array}\right\} \rightarrow_{\mathsf{subst}} \Gamma_1, \Gamma_2$$

Substitution for units. Substitution for units resemble the multiplicative connective but with the added feature of gathering (for a treatment of the multiplicative connective that includes gathering, see the appendix), but it terminates the rewriting sequence similarly to the atoms. Assume the cut formulas are $x : \mathbf{1}^{\tilde{a}\tilde{b}}$ and $y : \perp^c$. This means that the top context (Γ_1) can potentially contain some terminated endpoints (here, the a_i's), equipped with a queue of the form $[\![\Psi_{a_i}]\!][^x *]$ and some other channels (here, the b_j's) whose type includes a $\perp^x$. These x will need to be substituted by c. It also means that the endpoint c's type, in the bottom context (Γ_2), embeds a unit $\mathbf{1}$ labelled with a set of endpoints which includes y. This y will need to be replaced by the list on endpoints $\tilde{a}\tilde{b}$. This gives us the following rewriting

$$\left\{\begin{array}{lcl} \Gamma_1, \{[\![\Psi_{a_i}]\!][^x *]a_i : \cdot\}_i, \{[\![\Psi_{b_j}]\!]b_j : B_j\{\perp^x\}\}_j & \triangleright & x : \mathbf{1}^{\tilde{a}\tilde{b}} \\ \Gamma_2, [\![\Psi_c]\!]c : C\{\mathbf{1}^{y\tilde{u}}\} & \triangleright & y : \perp^c \end{array}\right\} \rightarrow_{\mathsf{subst}}$$
$$\Gamma_1, \{[\![\Psi_{a_i}]\!][^c *]a_i : \cdot\}_i, \{[\![\Psi_{b_j}]\!]b_j : B_j\{\perp^c\}\}_j, \Gamma_2, [\![\Psi_c]\!]c : C\{\mathbf{1}^{\tilde{a}\tilde{b}\tilde{u}}\}$$

In summary, the definition of the cut-rule is a bit more complicated than in classical linear logic: What used to be a simple identification of the cut-formula in the left sequent and its dual on the right, has become a sequence of somewhat cumbersome distribution and substitution steps. Hence, we will show how to adapt the proof accordingly in the rest of this section.

Reductive semantics. The Cut is defined in terms of distribution and substitution (applying these rewritings until they come to quiescence) and represents the interaction between two active processes P and Q. Recall that due to the non-determinism of the distr-rule, each distribution sequence will determine a different conclusion Γ.

$$\frac{P \Vdash \Gamma_1, [\![\Psi_1]\!]x : A \quad Q \Vdash \Gamma_2, [\![\Psi_2]\!]y : A^\perp \quad \{\Gamma_1 \triangleright [\![\Psi_1]\!]x : A, \Gamma_2 \triangleright [\![\Psi_2]\!]y : A^\perp\} \rightarrow^*_{\mathsf{distr}} \rightarrow^*_{\mathsf{subst}} \Gamma}{(\boldsymbol{\nu} xy)\,(P \mid Q) \Vdash^{\textsc{Cut}} \Gamma}\ \textsc{Cut}$$

$$
\begin{array}{lll}
(B_1) & (\boldsymbol{\nu} xy)\,(z \leftrightarrow x \mid Q) & \longrightarrow_\beta \; Q[z/y] \\
(B_2) & (\boldsymbol{\nu} xy)\,(x[] \mid y().Q) & \longrightarrow_\beta \; Q^{y \rightsquigarrow \tilde{u}} \\
(C_1) & (\boldsymbol{\nu} xy)\,(P \mid z().Q) & \longrightarrow_\beta \; z().(\boldsymbol{\nu} xy)\,(P \mid Q) \\
(C_2) & (\boldsymbol{\nu} xy)\,(P \mid u(v).Q) & \longrightarrow_\beta \; u(v).(\boldsymbol{\nu} xy)\,(P \mid Q) \\
(C_3) & (\boldsymbol{\nu} xy)\,(P \mid (z[v \triangleright Q].R)) & \longrightarrow_\beta \; z[v \triangleright Q].(\boldsymbol{\nu} xy)\,(P \mid R) \\
(K) & (\boldsymbol{\nu} xy)\,(x[a \triangleright P].Q \mid y(c).R) & \longrightarrow_\beta \; (\boldsymbol{\nu} xy)\,(Q \mid (\boldsymbol{\nu} a \triangleright c)\,(P \mid R))
\end{array}
$$

Fig. 3. Cut-reductions for the multiplicative fragment

We denote by $P \Vdash^{\textsc{Cut}} \Gamma$ the fact that a derivation can be constructed using the rules previously introduced for forwarders as well as the additional Cut rule. The rank of a Cut is defined as the size (number of connectives and units) of A. By extension the cut-rank of process P such that $P \Vdash^{\textsc{Cut}} \Gamma$, denoted as $\mathtt{rank}(P)$, is the maximum of the ranks of Cut rules occurring in the derivation of this judgement.

As one could hope in the proposition-as-type methodology, semantics of forwarders is obtained through cut-reductions.

Theorem 4 (Admissibility of Cut). *Let $P \Vdash \Gamma_1, [\![\Psi_1]\!]x : A$ and $Q \Vdash \Gamma_2, [\![\Psi_2]\!]y : A^\perp$. Then, for any Γ such that $\{\Gamma_1 \rhd [\![\Psi_1]\!]x : A,\ \Gamma_2 \rhd [\![\Psi_2]\!]y : A^\perp\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma$ there exists $R \Vdash \Gamma$, more precisely, we can define a sequence of reductions $(\boldsymbol{\nu} xy)\,(P \mid Q) \longrightarrow^*_\beta R$ that preserves typing.*

We will describe the method of the proof, for the details see the appendix. It proceeds by lexicographic induction on the structures of A, P and Q. That is, the induction hypothesis may be applied whenever (i) the rank of the cut gets smaller, or (ii) the rank stays the same and the cut is applied to at least one smaller process while the other stays the same.

As usual, we can distinguish (B)ase cases, (K)ey cases and (C)ommutative cases, see Figure 3. In the base cases (B_1) and (B_2), we reach the end of the reduction sequence and the cut disappears all together. In (B_1) we replace the cut with a simple uniform substitution; in (B_2) on the other hand, we require a non-uniform substitution reproducing the distr rewriting happening in the original cut. In the commutative cases (C_1), (C_2) and (C_3), the cut is reduced to a cut on P (that remains the same) and on a subprocess of Q, to which the induction hypothesis can easily be applied. Identical reductions are of course available symmetrically inverting the roles of P and Q. In the key case, the cut is rewritten into a cut of smaller rank on process Q and $S = (\boldsymbol{\nu} a \triangleright c)\,(P \mid R)$. This process S is obtained by Lemma 12 (see appendix).

The last piece of the puzzle that we need to complete the cut-elimination proof is a property of the distribution/substitution rewriting. To ensure that the type of $(\boldsymbol{\nu} xy)\,(Q \mid (\boldsymbol{\nu} a \triangleright c)\,(P \mid R))$ is indeed the same as the one of the original process $(\boldsymbol{\nu} xy)\,(x[a \triangleright P].Q \mid y(c).R)$, we rely on the fact that, if

$$
\left\{
\begin{array}{lcl}
\Gamma_1, [^x d : A^\perp][\![\Psi_u]\!]u : C & \rhd & [\![\Psi_1]\!]x : A \otimes^u B \\
\Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^y\} & \rhd & [\![\Psi_2]\!]y : A^\perp \parr^v B^\perp
\end{array}
\right\} \rightarrow^*_{\mathsf{distr}} \rightarrow^*_{\mathsf{subst}} \Gamma
$$

then there exists a distribution-substitution rewriting sequence such that

$$\left\{\begin{array}{lcl} \Gamma_1, [\![\Psi_u]\!] u : C & \triangleright & [\![\Psi_1]\!] x : B \\ \Gamma_2, [\![\Psi_v]\!] v : D\{\otimes^y\} & \triangleright & [\![\Psi_2]\!][^v d : A^\perp] y : B^\perp \end{array}\right\} \rightarrow^*_{\mathsf{distr}} \rightarrow^*_{\mathsf{subst}} \Gamma$$

This can be established by induction over the length of the rewriting sequence.

6 Asynchronous Forwarders Generalise Coherence

So far, we have focused on asynchronous forwarders syntax and semantics. As already explained intuitively in Section 2, one of the main contributions of this work is to use forwarders as a medium among communicating processes (more precisely those typable in the system CP presented in Section 3).

Multiparty Process Composition. We start by focusing on the *structural* rule that can be added to CP, namely the Cut, as seen in Section 3. After that, we will introduce an alternative (more general) rule that makes use of asynchronous forwarders. Rule Cut corresponds to parallel composition of processes. The implicit side condition that this rule uses is *duality*, i.e., we can compose two processes if endpoints x and y have a dual type. Carbone et al. [5] generalise the concept of duality to that of *coherence*. Coherence, denoted by $\vDash$, generalises duality to many endpoints, allowing for a cut rule that composes many processes in parallel

$$\frac{\{R_i \vdash \Sigma_i, x_i : A_i\}_{i\leq n} \quad G \vDash \{x_i : A_i\}_{i\leq n}}{(\boldsymbol{\nu} \tilde{x} : G)\,(R_1 \mid \ldots \mid R_n) \vdash \{\Sigma_i\}_{i\leq n}}\ \text{MCut}$$

The judgement $G \vDash \{x_i : A_i\}_{i\leq n}$ intuitively says that the $x_i : A_i$'s are compatible and the execution of the R_i will proceed without any error (no deadlock, no type mismatch in messages). Such a result is formalised by an MCut elimination theorem analogous to the one of CP. We leave G abstract here: it is a proof term and it corresponds to a global type (see [5]).

Our goal here is to replace the notion of coherence with an asynchronous forwarders Q, hoping for a rule resembling the following

$$\frac{\{R_i \vdash \Sigma_i, x_i : A_i\}_{i\leq n} \quad Q \Vdash \{x_i : A_i^\perp\}_{i\leq n}}{(\boldsymbol{\nu} \tilde{x} : Q)\,(R_1 \mid \ldots \mid R_n) \vdash \{\Sigma_i\}_{i\leq n}}\ \text{MCutF}$$

Asynchronous forwarders are more general than coherence: every coherence proof can be transformed into an *arbiter* process [5], which is indeed a forwarder, while there are judgements that are not coherent but are provable in our forwarder logic (see Example 1). In the rule MCutF, the role of a forwarder (replacing coherence) is to be a middleware that decides whom to forward messages to. This means that when a process R_i sends a message to the middleware, the message must be stored by the forwarder, who will later forward it to the right receiver.

Since our goal is to show that MCutF is admissible (and hence we can eliminate it from any correct proof), we extend such rule to account for messages

in transit that are temporarily held by the forwarder. In order to do so, we use the forwarders queues and some extra premises and define MCutQ as:

$$\frac{\{P_j \vdash \Delta_j, y_j : A_j\}_{j\leq m} \quad \{R_i \vdash \Sigma_i, x_i : B_i\}_{i\leq n} \quad Q \Vdash \left\{[\![\Psi_i]\!]x_i : B_i^{\perp}\right\}_{i\leq n}, \{[\![\Psi_i]\!]x_i : \cdot\}_{n<i\leq p}}{(\boldsymbol{\nu}\tilde{x} : Q[\tilde{y} \triangleleft P_1, \ldots, P_m])\,(R_1 \mid \ldots \mid R_n) \vdash \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}$$

We have three types of process terms: P_j's, R_i's and Q. Processes R_i's are the processes that we are composing, implementing a multiparty session. Q is the forwarder whose role is to certify compatibility and to determine, at run time, who talks to whom. Finally, processes P_i's must be linked to messages in the forwarder queue. Such processes are there because of the way $\otimes$ and $\parr$ work in linear logic. This will become clearer when we look at the reduction steps that lead to cut admissibility. This imposes a side condition on the rule, namely that

$$\bigcup_{i\leq p} \Psi_i \setminus \{*\} = \left\{y_j : A_j^{\perp}\right\}_{j\leq m}$$

Note that we need to introduce a new syntax for this new structural rule: in $(\boldsymbol{\nu}\tilde{x} : Q[\tilde{y} \triangleleft P_1, \ldots, P_m])\,(R_1 \mid \ldots \mid R_n)$, the list $P_1, \ldots, P_m$ denotes those messages (processes) in transit that are going to form a new session after the communication has taken place. In the remainder we (slightly abusively) abbreviate both $\{P_1, \ldots, P_m\}$ and $(R_1 \mid \ldots \mid R_n)$ as $\tilde{P}$ and $\tilde{R}$ respectively.

Semantics and MCutF-admissibility. In the previous paragraph, we have informally argued that forwarders generalise the notion of coherence as a notion of compatibility for composing processes typable in classical linear logic. In order to do that formally, we show that MCutF is admissible, yielding a semantics for our extended CP (with MCutF) in a proposition-as-types fashion.

We proceed by looking at all cases that involve the multiplicative fragment (see appendix for the full set of rules). In the sequel, we use the following abbreviations, $\Gamma = \left\{[\![\Psi_i]\!]x_i : B_i^{\perp}\right\}_{i\leq n}, \{[\![\Psi_i]\!]x_i : \cdot\}_{n<i\leq p}$ and $\Gamma - k = \Gamma \setminus \left\{[\![\Psi_k]\!]x_k : B_k^{\perp}\right\}$. We also omit (indicated as "...") the premises of the MCutQ that do not play a role in the reduction at hand, and assume that they are always the same as above, that is, $\{P_j \vdash \Delta_j, y_j : A_j\}_{j\leq m}$ and $\{R_i \vdash \Sigma_i, x_i : B_i\}_{i\leq n}$.

Send Message ($\otimes$). This is the case when a process intends to send a message, which corresponds to a $\otimes$ rule. As a consequence, the forwarder has to be ready to receive the message (to then forward it later):

$$\frac{\dfrac{P \vdash \Delta, y : A \quad R \vdash \Sigma, x : B}{x[y \triangleright P].R \vdash \Delta, \Sigma, x : A \otimes B}\otimes \qquad \ldots \qquad \dfrac{Q \Vdash [\![\Psi]\!][^{x_k}y : A^{\perp}]x : B^{\perp}, \Gamma}{x(y).Q \Vdash [\![\Psi]\!]x : A^{\perp} \parr^{x_k} B^{\perp}, \Gamma}\parr}{(\boldsymbol{\nu}x\tilde{x} : x(y).Q[\tilde{y} \triangleleft \tilde{P}])\,(x[y \triangleright P].R \mid \tilde{R}) \vdash \Delta, \Sigma, \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}\ \text{MCutQ}$$

The process on the left is ready to send the message to the forwarder. By inspecting the forwarder, it is clear that the message will have to be forwarded to endpoint x_k, at a later stage. Observe that the nature of $\otimes$ forces us to deal with the process P: the idea is that when the forwarder will finalise the communication (by sending to a process R' owning endpoint x_k) process P will be composed with R'. For now, we obtain the reductum:

$$\frac{P \vdash \Delta, y : A \quad R \vdash \Sigma, x : B \qquad \ldots \qquad Q \Vdash [\![\Psi]\!][^{x_k}y : A^{\perp}]x : B^{\perp}, \Gamma}{(\boldsymbol{\nu}x\tilde{x} : Q[y, \tilde{y} \triangleleft P, \tilde{P}])\,(R \mid \tilde{R}) \vdash \Delta, \Sigma, \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}\ \text{MCutQ}$$

Receive Message ($\parr$). At a later point, the forwarder will be able to complete the forwarding operation by connecting with a process ready to receive ($\parr$ rule):

$$\dfrac{\dfrac{R \vdash \Sigma, y : A, x : B}{x(y).R \vdash \Sigma, x : A \parr B}\parr \quad \dfrac{\begin{array}{c}P \vdash \Delta, z : A^{\perp} \qquad \dots \\ S \Vdash z : A, y : A^{\perp} \quad Q \Vdash [\![\Psi_x]\!] x : B^{\perp}, \Gamma\end{array}}{x[y \triangleright S].Q \Vdash [\![\Psi_x]\!] x : A^{\perp} \otimes^{x_k} B^{\perp}, [^{x} z : A][\![\Psi_k]\!] x_k : B_k^{\perp}, \Gamma - k}\otimes}{(\boldsymbol{\nu} x\tilde{x} : x[y \triangleright S].Q[z, \tilde{y} \lhd P, \tilde{P}])\,(x(y).R \mid \tilde{R}) \vdash \Delta, \Sigma, \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}$$

Key ingredients are process P with endpoint z of type $A^{\perp}$, endpoint x_k in the forwarder with a boxed endpoint z with type A, and process $x(y).R$ ready to receive.

After reduction, we obtain the following:

$$\dfrac{(\boldsymbol{\nu} yz : S)\,(R \mid P) \vdash \Sigma, \Delta, x : B \qquad \dots \qquad Q \Vdash [\![\Psi_x]\!] x : B^{\perp}, \Gamma}{(\boldsymbol{\nu} x\tilde{x} : Q[\tilde{y} \lhd \tilde{P}])\,((\boldsymbol{\nu} yz : S)\,(R \mid P) \mid \tilde{R}) \vdash \Delta, \Sigma, \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}\ \textsc{MCutQ}$$

Where the left premiss is obtained as follows:

$$\dfrac{R \vdash \Sigma, y : A, x : B \quad P \vdash \Delta, z : A^{\perp} \quad S \Vdash z : A, y : A^{\perp}}{(\boldsymbol{\nu} yz : S)\,(R \mid P) \vdash \Sigma, \Delta, x : B}\ \textsc{MCutQ}$$

meaning that now the message (namely process P) has finally been delivered and it can be directly linked to R with a new (but smaller) MCutQ.

Units ($\perp$ and $\mathbf{1}$). These cases are a simplified version of $\parr$ and $\otimes$ respectively:

$$\dfrac{\dfrac{P \vdash \Delta}{x().P \vdash \Delta, x : \perp}\perp \quad \dfrac{}{x[] \Vdash x : \mathbf{1}^{\tilde{x}}, \{[^{x}*]x_i : \cdot\}_i}\mathbf{1}}{(\boldsymbol{\nu} x : x[])\,(x().P) \vdash \Delta}\ \textsc{MCutQ} \qquad \Longrightarrow \quad P \vdash \Delta$$

$$\dfrac{\dfrac{}{x[] \vdash x : \mathbf{1}}\mathbf{1} \quad \dots \quad \dfrac{Q \Vdash [\![\Psi_x]\!][^{x_k}*]x : \cdot, \Gamma}{x().Q \Vdash [\![\Psi_x]\!] x : \perp^{x_k}, \Gamma}\perp}{(\boldsymbol{\nu} x\tilde{x} : x().Q[\tilde{y} \lhd \tilde{P}])\,(x[] \mid \tilde{R}) \vdash \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}\ \textsc{MCutQ}$$

$$\Longrightarrow \quad \dfrac{\dots \qquad Q \Vdash [\![\Psi_x]\!][^{x_k}*]x : \cdot, \Gamma}{(\boldsymbol{\nu} x\tilde{x} : Q[\tilde{y} \lhd \tilde{P}])\,(\tilde{R}) \vdash \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}}\ \textsc{MCutQ}$$

Axiom. Finally, the axiom (only defined on atoms, see [5] for a discussion on eta-expansion) can completely erase a cut as follows:

$$\dfrac{\dfrac{}{x \leftrightarrow z1 \vdash x : a, z : a^{\perp}}\ \textsc{Ax} \quad \dfrac{}{y \leftrightarrow w \vdash y : a^{\perp}, w : a}\ \textsc{Ax} \quad \dfrac{}{x \leftrightarrow y \Vdash x : a^{\perp}, y : a}\ \textsc{Ax}}{(\boldsymbol{\nu} xy : x \leftrightarrow y)\,(x \leftrightarrow z \mid y \leftrightarrow w) \vdash z : a^{\perp}, w : a}\ \textsc{MCutQ}$$

$$\Longrightarrow \quad \dfrac{}{z \leftrightarrow w \vdash z : a^{\perp}, w : a}\ \textsc{Ax}$$

These reductions allow us to prove the key lemma of this section.

Lemma 5 (Admissibility of MCutQ). *If $\{P_j \vdash \Delta_j, y_j : A_j\}_{j\leq m}$ and $\{R_i \vdash \Sigma_i, x_i : B_i\}_{i\leq n}$ and $Q \Vdash \{[\![\Psi_i]\!] x_i : B_i^{\perp}\}_{i\leq n}, \{[\![\Psi_i]\!] x_i : \cdot\}_{n<i\leq p}$ then there exists a process S such that $(\boldsymbol{\nu} \tilde{x} : Q[\tilde{y} \lhd \tilde{P}])\,\tilde{R} \Rightarrow^{*} S$ and $S \vdash \{\Delta_j\}_{j\leq m}, \{\Sigma_i\}_{i\leq n}$.*

Proof (Sketch). By lexicographic induction on (i) the sum of sizes of the B_i's and (ii) the sum of sizes of the R_i's. The base cases and key cases have been detailed above. The commutative cases are straightforward and only need to consider the possible last rule applied to a premiss of the form $R_i \vdash \Sigma_i, x_i : B_i$. □

We can finally conclude with the following theorem as a special case.

Theorem 6 (Admissibility of MCutF). *If* $\{R_i \vdash \Sigma_i, x_i : A_i\}_{i \leq n}$ *and* $Q \Vdash \{x_i : A_i^{\perp}\}_{i \leq n}$ *then there exists a process* S *such that* $(\boldsymbol{\nu} \tilde{x} : Q)\,(R_1 \mid \ldots \mid R_n) \Rightarrow^* S$ *and* $S \vdash \{\Delta_j\}_{j \leq m}, \{\Sigma_i\}_{i \leq n}$.

7 Asynchronous Forwarders as Multiparty Compatibility

In the previous section, we have shown that our forwarders can be used to govern the composition of processes. In this section, we show that forwarders precisely capture the intuitive notion of compatibility, i.e., *given a set of processes, is there a communication pattern they can follow so that they progress without reaching an error?* The concept of compatibility has been studied in the context of multiparty session types [18,10]. Below, we adapt multiparty compatibility to our logical setting, focusing on CP processes. We start by introducing the concept of semantics for type contexts:

Definition 7 (Type-Context Semantics). *Let* Γ *be a forwarder type context. Then, we define* $\xrightarrow{\alpha}$ *as the minimum relation satisfying the following rules:*

$$
\begin{array}{lcl}
\Gamma, [\![\Psi]\!] x : A \parr^{u} B & \xrightarrow{x \otimes u} & \Gamma, [\![\Psi]\!][^{u} y : A] x : B \\
\Gamma, [^{x} z : A^{\perp}][\![\Psi_u]\!] u : C, [\![\Psi_x]\!] x : A \otimes^{u} B & \xrightarrow{x \parr u} & \Gamma, [\![\Psi_u]\!] u : C, [\![\Psi_x]\!] x : B \\
\Gamma, [\![\Psi]\!] x : \perp^{u} & \xrightarrow{x \mathbf{1} u} & \Gamma, [\![\Psi]\!][^{u} *] x : \cdot \\
\{[^{x} *] u_i : \cdot\}_{1 \leq i \leq n}, x : \mathbf{1}^{\tilde{u}} & \xrightarrow{x \perp \tilde{u}} & \checkmark \\
x : a^{\perp}, y : a & \xrightarrow{x \leftrightarrow y} & \checkmark
\end{array}
$$

The rules above capture an asynchronous semantics for typing contexts (for the multiplicative fragment of CP). We observe that our definition, despite looking different, is equivalent to that given by Ghilezal et al. [10]. In fact, our context uses a dualised version: in order to obtain their setting, we should just consider a double context containing the dual of the endpoint types and the dual of the queues, respectively. Also note that since we have no infinite computations, we do not need to consider fairness. Using the relation on contexts above, we can define when a set of endpoints successfully progresses without reaching an error. This can be formalised by the concept of live path. In the sequel, let α range over all the possible labels of the relation above. Moreover, let $\alpha_1, \ldots, \alpha_n$ be a *path* for a context Γ whenever there exist $\Gamma_1, \ldots, \Gamma_n$ such that $\Gamma \xrightarrow{\alpha_1} \Gamma_1 \ldots \xrightarrow{\alpha_n} \Gamma_n$.

Definition 8 (Live Path). *Let* Γ *be a context and let* $\alpha_1, \ldots, \alpha_n$ *be a path for* Γ. *We say that the path* $\alpha_1, \ldots, \alpha_n$ *is* live *if* $\Gamma \xrightarrow{\tilde{\alpha}} \checkmark$ *and, if for some* $i < n$,

1. $\Gamma_i = \Gamma_i', [^{x} z : A^{\perp}][\![\Psi_u]\!] u : C$, *then there exists* $k > i$ *such that* $\alpha_k = x \parr u$;

2. $\Gamma_i = \Gamma_i', [^x*]u : \cdot$, *then* $\alpha_n = x \perp \tilde{u}$ *with* $u \in \tilde{u}$;
3. $\Gamma_i = \Gamma_i', [\![\Psi_x]\!]x : A \otimes^u B$, *then there exists* $k > i$ *such that* $\alpha_k = x \parr u$;
4. $\Gamma_i = \Gamma_i', [\![\Psi_x]\!]x : \mathbf{1}^{\tilde{u}}$, *then* $\alpha_n = x \perp \tilde{u}$.

Conditions (1) and (2) state that every message that has been enqueued in the forwarder is eventually forwarded. On the other hand, conditions (3) and (4) state that every forwarding instruction is eventually executed. These conditions are basically what our forwarders do, as stated by the following:

Theorem 9. *Γ has a live path iff there exists a forwarder F such that $F \Vdash \Gamma$.*

Proof. We need to prove each direction separately. For the if direction, we proceed by induction on the proof of F (logically, on its η-long normal form). For the only-if direction, we proceed by induction on the length of the path. □

8 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 us to introduce forwarders. As a result, forwarders provide a more general notion of compatibility. An earlier unpublished version of this work [6], proposes synchronous forwarders, i.e., the restriction of forwarders with only buffers of size one. In that case, we show that we can always construct a coherence proof from a synchronous forwarder. However, synchronous forwarders fail to capture all the possible interleaving of an arbiter (encoding of coherence to processes).

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 previous work [5], they do generate arbiters as linear logic proofs, which are special instances of forwarders. In this work, we generalise this approach to characterise exactly which processes can justify the compatibility of several processes.

Sangiorgi [21], probably the first to treat forwarders for the π-calculus, uses binary forwarders, i.e., processes that only forward between two channels, which are 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 [13] 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. [9] 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 for forwarding). Such a restriction is similar to the intuition behind our forwarders, with the key difference that their methodology would not apply to 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 modelled by a "connection-cut".

Recent work [17,11] 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 forwarders can be seen as a generalisation to multiparty monitors. Multiparty monitors are also addressed by Hamers and Jongmans [12], but not in a linear logic context.

Our forwarder mechanism may be confused with that of locality [19], 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 (which then must occur 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.

9 Conclusions

In this paper, we have presented a logical characterisation of forwarders and their semantics based on cut elimination. Additionally, we have shown that forwarders can replace the notion of coherence when composing multiple processes; indeed, forwarders can implement all and only compatible multiparty communications within linear logically typable sessions. We discuss some aspects of forwarders developed in this paper and identify possible future extensions.

Process Language. Our process language is based on [24] with some omissions. For the sake of presentation, we have left out the additive and exponential fragment (which can be found in the appendix). Moreover, we have not included polymorphic communications. We show in the appendix that the logic presented in this paper extends directly to additives and exponentials, and we conjecture that our forwarder logic extends to polymorphic types $\exists X.A$ and $\forall X.A$ as well. A further extension to support recursion is left to 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. An earlier version of the forwarder logic was based on intuitionistic linear logic, but moving to CLL required fewer rules and greatly improved the presentation. Nevertheless, our results should be easily reproducible in intuitionistic linear logic.

Variants of Coherence. Our results show that forwarders are a generalisation of coherence proofs. Indeed, coherence would correspond to the notion of *synchronous forwarders* [6], the restriction of forwarders with only buffers of size one. As a follow-up, we would like to investigate, whether other syntactic restrictions of forwarders also induce interesting generalised notions of coherence, and, as a consequence, generalisations of global types.

References

1. Barbanera, F., Dezani-Ciancaglini, M.: Open multiparty sessions. In: Bartoletti, M., Henrio, L., Mavridou, A., Scalas, A. (eds.) Proceedings 12th Interaction and Concurrency Experience, ICE 2019, Copenhagen, Denmark, 20-21 June 2019. EPTCS, vol. 304, pp. 77–96 (2019). https://doi.org/10.4204/EPTCS.304.6, https://doi.org/10.4204/EPTCS.304.6
2. Caires, L., Pérez, J.A.: Multiparty session types within a canonical binary theory, and beyond. In: Albert, E., Lanese, I. (eds.) 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. Lecture Notes in Computer Science, vol. 9688, pp. 74–95. Springer (2016). https://doi.org/10.1007/978-3-319-39570-8_6, https://doi.org/10.1007/978-3-319-39570-8_6
3. Caires, L., Pfenning, F.: Session types as intuitionistic linear propositions. In: CONCUR. pp. 222–236 (2010)
4. Caires, L., Pfenning, F., Toninho, B.: Linear logic propositions as session types. Math. Struct. Comput. Sci. **26**(3), 367–423 (2016). https://doi.org/10.1017/S0960129514000218, https://doi.org/10.1017/S0960129514000218
5. Carbone, M., Lindley, S., Montesi, F., Schürmann, C., Wadler, P.: Coherence generalises duality: A logical explanation of multiparty session types. In: Desharnais, J., Jagadeesan, R. (eds.) 27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada. LIPIcs, vol. 59, pp. 33:1–33:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany (2016). https://doi.org/10.4230/LIPIcs.CONCUR.2016.33, https://doi.org/10.4230/LIPIcs.CONCUR.2016.33
6. Carbone, M., Marin, S., Schürmann, C.: Synchronous forwarders. CoRR **abs/2102.04731** (2021), https://arxiv.org/abs/2102.04731
7. Carbone, M., Montesi, F., Schürmann, C., Yoshida, N.: Multiparty session types as coherence proofs. In: CONCUR. pp. 412–426 (2015)
8. Coppo, M., Dezani-Ciancaglini, M., Yoshida, N., Padovani, L.: Global progress for dynamically interleaved multiparty sessions. MSCS **760**, 1–65 (2015)
9. Gardner, P., Laneve, C., Wischik, L.: Linear forwarders. Inf. Comput. **205**(10), 1526–1550 (2007). https://doi.org/10.1016/j.ic.2007.01.006, https://doi.org/10.1016/j.ic.2007.01.006
10. Ghilezan, S., Pantovic, J., Prokic, I., Scalas, A., Yoshida, N.: Precise subtyping for asynchronous multiparty sessions. Proc. ACM Program. Lang. **5**(POPL), 1–28 (2021). https://doi.org/10.1145/3434297, https://doi.org/10.1145/3434297
11. Gommerstadt, H., Jia, L., Pfenning, F.: Session-typed concurrent contracts. In: Ahmed, A. (ed.) 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. Lecture Notes in Computer Science, vol. 10801, pp. 771–798. Springer (2018). https://doi.org/10.1007/978-3-319-89884-1_27, https://doi.org/10.1007/978-3-319-89884-1_27
12. Hamers, R., Jongmans, S.: Discourje: Runtime verification of communication protocols in clojure. In: Biere, A., Parker, D. (eds.) Tools and Algorithms

for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12078, pp. 266–284. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_15, https://doi.org/10.1007/978-3-030-45190-5_15

13. van den Heuvel, B., Pérez, J.A.: Session type systems based on linear logic: Classical versus intuitionistic. In: Balzer, S., Padovani, L. (eds.) Proceedings of the 12th International Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software, PLACES@ETAPS 2020, Dublin, Ireland, 26th April 2020. EPTCS, vol. 314, pp. 1–11 (2020)
14. Honda, K., Vasconcelos, V., Kubo, M.: Language primitives and type disciplines for structured communication-based programming. In: ESOP. pp. 22–138 (1998)
15. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: Necula, G.C., Wadler, P. (eds.) Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008. pp. 273–284. ACM (2008)
16. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. JACM **63**(1), 9 (2016), also: POPL, 2008, pages 273–284
17. Jia, L., Gommerstadt, H., Pfenning, F.: Monitors and blame assignment for higher-order session types. In: Bodík, R., Majumdar, R. (eds.) Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016. pp. 582–594. ACM (2016). https://doi.org/10.1145/2837614.2837662, https://doi.org/10.1145/2837614.2837662
18. Lange, J., Yoshida, N.: Verifying asynchronous interactions via communicating session automata. In: Dillig, I., Tasiran, S. (eds.) Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11561, pp. 97–117. Springer (2019). https://doi.org/10.1007/978-3-030-25540-4_6, https://doi.org/10.1007/978-3-030-25540-4_6
19. Merro, M., Sangiorgi, D.: On asynchrony in name-passing calculi. Mathematical Structures in Computer Science **14**(5), 715–767 (2004). https://doi.org/10.1017/S0960129504004323
20. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I and II. Information and Computation **100**(1), 1–40,41–77 (Sep 1992)
21. Sangiorgi, D.: pi-calculus, internal mobility, and agent-passing calculi. Theor. Comput. Sci. **167**(1&2), 235–274 (1996), https://doi.org/10.1016/0304-3975(96)00075-8
22. Vasconcelos, V.T.: Fundamentals of session types. Inf. Comput. **217**, 52–70 (2012)
23. Wadler, P.: Propositions as sessions. In: ICFP. pp. 273–286 (2012)
24. Wadler, P.: Propositions as sessions. Journal of Functional Programming **24**(2–3), 384–418 (2014), also: ICFP, pages 273–286, 2012

A Full CP and Classical Linear Logic

Types.

$$A ::= \quad a \mid a^{\perp} \mid 1 \mid \perp \mid (A \otimes A) \mid (A \parr A) \mid (A \oplus A) \mid (A \,\&\, A) \mid !A \mid ?A \qquad (4)$$

Duality.

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

Processes.

$P, Q ::=$	$x \leftrightarrow y$	(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)

CP-typing.

$$\frac{}{x \leftrightarrow y \vdash x : a^{\perp}, y : a}\ \textsc{Ax} \qquad \frac{P \vdash \Delta}{x().P \vdash \Delta, x : \perp}\ \perp \qquad \frac{}{x[] \vdash x : \mathbf{1}}\ \mathbf{1}$$

$$\frac{P \vdash \Delta_1, y : A_1 \quad Q \vdash \Delta_2, x : A_2}{x[y \triangleright P].Q \vdash \Delta_1, \Delta_2, x : A_1 \otimes A_2}\ \otimes \qquad \frac{P \vdash \Delta, y : A_1, x : A_2}{x(y).P \vdash \Delta, x : A_1 \parr A_2}\ \parr$$

$$\frac{P \vdash \Delta, x : A_1}{x[\mathsf{inl}].P \vdash \Delta, x : A_1 \oplus A_2}\ \oplus_1 \qquad \frac{P \vdash \Delta, x : A_2}{x[\mathsf{inr}].P \vdash \Delta, x : A_1 \oplus A_2}\ \oplus_2 \qquad \frac{P \vdash \Delta, x : A_1 \quad Q \vdash \Delta, x : A_2}{x.\mathsf{case}(P, Q) \vdash \Delta, x : A_1 \,\&\, A_2}\ \&$$

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

$$\frac{P \vdash \Delta}{P \vdash \Delta, x : ?A}\ \textsc{Weaken} \qquad \frac{P \vdash \Delta, y : ?A, z : ?A}{P\{x/y, x/z\} \vdash \Delta, x : ?A}\ \textsc{Contract}$$

B Rules – including gathering and broadcasting

We can complexify the types in order to allow broadcasting and gathering.

$$\begin{array}{rcllll} A, B & ::= & a & \mid & a^{\perp} & \\ & \mid & \mathbf{1}^{[u_1,\ldots,u_n]} & \mid & \perp^{u} & \\ & \mid & (A \otimes^{[u_1,\ldots,u_n]} B) & \mid & (A \parr^{u} B) & \\ & \mid & (A \oplus^{u} B) & \mid & (A \,\&^{[u_1,\ldots,u_n]} B) & \\ & \mid & ?^{u} A & \mid & !^{[u_1,\ldots,u_n]} A & \end{array}$$

Contexts need to be changed as well.

$$\begin{array}{rcllllll} [\![\Psi]\!] & ::= & \cdot & \mid & [^{u}*] & \mid & [^{u}y : B] & \mid & [\![\Psi]\!][\![\Psi]\!] \\ & & & \mid & [^{u}\mathcal{L}] & \mid & [^{u}\mathcal{R}] & \mid & [^{u}\mathcal{Q}] \end{array}$$

$[^{u}\mathcal{L}]$ (or $[^{u}\mathcal{R}]$) and $[^{u}\mathcal{Q}]$ indicate that a branching request and server invocation, respectively, has been received and must be forwarded

When it needs to be forwarded to a list of endpoints, we may use the abbreviation $[^{\tilde{u}}\mathcal{X}]$ for denoting the $[^{u_1}\mathcal{X}]\dots[^{u_n}\mathcal{X}]$ when $\tilde{u} = u_1, \dots, u_n$. In this case, we also assume the implicit rewriting $[^{\varnothing}\mathcal{X}][\![\Psi]\!] \equiv [\![\Psi]\!]$.

The rules then need to be adapted as follows.

$$\frac{}{x \leftrightarrow y \Vdash x : a^{\perp}, y : a}\ \text{Ax} \qquad \frac{P \Vdash \Gamma, [\![\Psi]\!][^{u}*]x : \cdot}{x().P \Vdash \Gamma, [\![\Psi]\!]x : \perp^{u}}\ \perp \qquad \frac{}{x[] \Vdash \{[^{x}*]u_i : \cdot\}_i, x : \mathbf{1}^{\tilde{u}}}\ \mathbf{1}$$

$$\frac{P \Vdash \Gamma, [\![\Psi]\!][^{u}y : A]x : B}{x(y).P \Vdash \Gamma, [\![\Psi]\!]x : A \parr^{u} B}\ \parr \qquad \frac{P \Vdash \{\Delta_i\}_i, y : A \quad Q \Vdash \Gamma, \{[\![\Psi_i]\!]u_i : C_i\}_i, [\![\Psi]\!]x : B}{x[y \triangleright P].Q \Vdash \Gamma, \{[^{x}\Delta_i][\![\Psi_i]\!]u_i : C_i\}_i, [\![\Psi]\!]x : A \otimes^{\tilde{u}} B}\ \otimes\ (\Delta_i \neq \varnothing)$$

$$\frac{P \Vdash \Gamma, [\![\Psi]\!][^{\tilde{u}}\mathcal{L}]x : A \quad Q \Vdash \Gamma, [\![\Psi]\!][^{\tilde{u}}\mathcal{R}]x : B}{x.\mathsf{case}(P, Q) \Vdash \Gamma, [\![\Psi]\!]x : A \,\&^{\tilde{u}}\, B}\ \&\ (\tilde{u} \neq \varnothing)$$

$$\frac{P \Vdash \Gamma, [\![\Psi_z]\!]z : C, [\![\Psi_x]\!]x : A}{x[\mathsf{inl}].P \Vdash \Gamma, [^{x}\mathcal{L}][\![\Psi_z]\!]z : C, [\![\Psi_x]\!]x : A \oplus^{z} B}\ \oplus_l \qquad \frac{P \Vdash \Gamma, [\![\Psi_z]\!]z : C, [\![\Psi_x]\!]x : B}{x[\mathsf{inr}].P \Vdash \Gamma, [^{x}\mathcal{R}][\![\Psi_z]\!]z : C, [\![\Psi_x]\!]x : A \oplus^{z} B}\ \oplus_r$$

$$\frac{P \Vdash \{u_i : ?B_i\}_i, [^{\tilde{u}}\mathcal{Q}]y : A}{!x(y).P \Vdash \{u_i : ?B_i\}_i, x : !^{\tilde{u}}A}\ !\ (\tilde{u} \neq \varnothing) \qquad \frac{P \Vdash \Gamma, [\![\Psi_z]\!]z : C, [\![\Psi_x]\!]y : A}{?x[y].P \Vdash \Gamma, [^{x}\mathcal{Q}][\![\Psi_z]\!]z : C, [\![\Psi_x]\!]x : ?^{z}A}\ ?$$

Proposition 2 is naturally extended to additives and exponentials.

Proposition 10. *Any forwarder is a CP-typable process, that is, if $P \Vdash \Gamma$, then $P \vdash \llcorner\Gamma\lrcorner$. The embedding $\llcorner\cdot\lrcorner$ being extended as: $\llcorner[^{u}\mathcal{X}][\![\Psi]\!]\lrcorner = \llcorner[\![\Psi]\!]\lrcorner$.*

C Cut admissibility

C.1 General distribution and substitution

Distribution

Multiplicatives with gathering

$$\left\{\begin{array}{l}\Gamma_1, [\![\Psi_{2z}]\!] \bullet [\![\Psi_z]\!]z : B\{\otimes^{\tilde{v}x}\} \triangleright [^{z}\{\Delta_i\}_i][\![\Psi_1]\!]x : A \\ \Gamma_2, \{[\![\Psi_{1c_i}]\!] \bullet [\![\Psi_{c_i}]\!]c_i : C_i\}_i \triangleright [\![\Psi_2]\!]y : A^{\perp}\end{array}\right\} \to_{\mathsf{distr}}$$

$$\left\{\begin{array}{l}\Gamma_1, [\![\Psi_{2z}]\!] \bullet [\![\Psi_z]\!]z : B\{\otimes^{\tilde{v}\tilde{c}}\} \triangleright [\![\Psi_1]\!]x : A \\ \Gamma_2, \{[\![\Psi_{1c_i}]\!][^{z}\Delta_i] \bullet [\![\Psi_{c_i}]\!]c_i : C_i\}_i \triangleright [\![\Psi_2]\!]y : A^{\perp}\end{array}\right\}$$

Additives

$$\left\{\begin{array}{l}\Gamma_1, [\![\Psi_{2d}]\!] \bullet [\![\Psi_d]\!]d : D\{\oplus^{\boldsymbol{x}}\} \triangleright [^{d}\mathcal{L}][\![\Psi_1]\!]x : A \\ \quad \Gamma_2, [\![\Psi_{1c}]\!] \bullet [\![\Psi_c]\!]c : D \triangleright [\![\Psi_2]\!]y : A^{\perp}\end{array}\right\} \to_{\mathsf{distr}}$$

$$\left\{\begin{array}{l}\Gamma_1, [\![\Psi_{2d}]\!] \bullet [\![\Psi_d]\!]d : D\{\oplus^{\mathbf{c}}\} \triangleright [\![\Psi_1]\!]x : A \\ \quad \Gamma_2, [\![\Psi_{1c}]\!][^{d}\mathcal{L}] \bullet [\![\Psi_c]\!]c : C \triangleright [\![\Psi_2]\!]y : A^{\perp}\end{array}\right\}$$

Exponentials

$$\left\{\begin{array}{r} \Gamma_1, [\![\Psi_{2d}]\!] \bullet [\![\Psi_d]\!] d : D\{?^{\boldsymbol{x}}\} \triangleright [^d\mathcal{Q}][\![\Psi_1]\!] x : A \\ \Gamma_2, [\![\Psi_{1c}]\!] \bullet [\![\Psi_c]\!] c : D \triangleright [\![\Psi_2]\!] y : A^\perp \end{array}\right\} \to_{\mathsf{distr}}$$

$$\left\{\begin{array}{l} \Gamma_1, [\![\Psi_{2d}]\!] \bullet [\![\Psi_d]\!] d : D\{?^{\boldsymbol{c}}\} \triangleright [\![\Psi_1]\!] x : A \\ \Gamma_2, [\![\Psi_{1c}]\!][^d\mathcal{Q}] \bullet [\![\Psi_c]\!] c : C \triangleright [\![\Psi_2]\!] y : A^\perp \end{array}\right\}$$

Substitution

Multiplicatives with gathering

$$\left\{\begin{array}{ll} \Gamma_1, [\![\Psi_z]\!] z : E\{\otimes^{\tilde{v}} x\} & \triangleright x : A \parr^z B \\ \Gamma_2, \{[^y\Delta_i][\![\Psi_i]\!] c_i : C_i\}_i, \{[\![\Psi_j]\!] d_j : D_j\{\parr^y\}\}_j & \triangleright y : A^\perp \otimes^{\tilde{c}\tilde{d}} B^\perp \end{array}\right\} \to_{\mathsf{subst}}$$

$$\left\{\begin{array}{ll} \Gamma_1, [\![\Psi_z]\!] z : E\{\otimes\}^{\tilde{v}\tilde{c}\tilde{d}} & \triangleright x : B \\ \Gamma_2, \{[^z\Delta_i][\![\Psi_i]\!] c_i : C_i\}_i, \{[\![\Psi_j]\!] d_j : D_j\{\parr^z\}\}_j & \triangleright y : B^\perp \end{array}\right\}$$

Additives

$$\left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : C_i\{\oplus^x\}\}_i & \triangleright x : A \,\&^{\tilde{u}}\, B \\ \Gamma_2, [^y\mathcal{L}][\![\Psi_v]\!] v : D & \triangleright y : A^\perp \oplus^v B^\perp \end{array}\right\} \longrightarrow_{\mathsf{subst}} \left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : C_i\{\oplus^v\}\}_i & \triangleright x : B \\ \Gamma_2, [^{\tilde{u}}\mathcal{L}][\![\Psi_v]\!] v : D & \triangleright y : B^\perp \end{array}\right\}$$

$$\left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : C_i\{\oplus^x\}\}_i & \triangleright x : A \,\&^{\tilde{u}}\, B \\ \Gamma_2, [\![\Psi_v]\!] v : D\{\&^{y\tilde{z}}\} & \triangleright y : A^\perp \oplus^v B^\perp \end{array}\right\} \longrightarrow_{\mathsf{subst}} \left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : C_i\{\oplus^v\}\}_i & \triangleright x : B \\ \Gamma_2, [\![\Psi_v]\!] v : D\{\&^{\tilde{u}\tilde{z}}\} & \triangleright y : B^\perp \end{array}\right\}$$

Exponentials

$$\left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : B_i\{?^x\}\}_i & \triangleright x : !^{\tilde{u}} A \\ \Gamma_2, [^y\mathcal{Q}][\![\Psi_z]\!] z : C & \triangleright y : ?^z A^\perp \end{array}\right\} \longrightarrow_{\mathsf{subst}} \left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : B_i\{?^z\}\}_i & \triangleright x : A \\ \Gamma_2, [^{\tilde{u}}\mathcal{Q}][\![\Psi_z]\!] z : C & \triangleright y : A^\perp \end{array}\right\}$$

$$\left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : B_i\{?^x\}\}_i & \triangleright x : !^{\tilde{u}} A \\ \Gamma_2, [\![\Psi_z]\!] z : C\{!^y\} & \triangleright y : ?^z A^\perp \end{array}\right\} \longrightarrow_{\mathsf{subst}} \left\{\begin{array}{ll} \Gamma_1, \{[\![\Psi_{u_i}]\!] u_i : B_i\{?^z\}\}_i & \triangleright x : A \\ \Gamma_2, [\![\Psi_z]\!] z : C\left\{!^{\tilde{u}}\right\} & \triangleright y : A^\perp \end{array}\right\}$$

C.2 Cut-in-box lemmas

Lemma 11. *Let $P \Vdash \Delta, x : A$ and $Q \Vdash \Gamma, [\![\Psi_1]\!][^z w : A^\perp][\![\Psi_2]\!] y : B$ be derivable. Then, there exists R such that $R \Vdash^{\text{Cut}} \Gamma, [\![\Psi_1]\!][^z\Delta][\![\Psi_2]\!] y : B$ is derivable with* Cut *and* $\mathtt{rank}(R) = \mathtt{size}(A)$*. We denote this process as $R = (\boldsymbol{\nu} x \triangleright w)\,(P \mid Q)$*

Proof. By induction on the structure of Q. Because of the shape of its typing environment, Q cannot be of the shape $a \leftrightarrow b$ nor of the shape $a[]$.

Case $Q = z[v \triangleright S].T$ and $w \in \mathtt{fn}(S)$. Assume we have the following

$$\frac{S \Vdash w : A^\perp, \{\Delta_i\}_i, v : D \quad T \Vdash \Gamma', [\![\Psi_y]\!] y : B, \{[\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : E}{z[v \triangleright S].T \Vdash \Gamma', [^z w : A^\perp][\![\Psi_y]\!] y : B, \{[^z\Delta_i][\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : D \otimes^{y\tilde{u}} E}\ \otimes$$

We take $R = z[v \triangleright (\boldsymbol{\nu} xw)\,(P \mid S)].T$ and indeed $\mathtt{rank}(R) = \mathtt{size}(A)$ as follows:

$$\dfrac{\dfrac{P \Vdash \Delta, x : A \quad S \Vdash w : A^{\perp}, \{\Delta_i\}_i, v : D}{(\boldsymbol{\nu} xw)\,(P \mid S) \Vdash \Delta, \{\Delta_i\}_i, v : D}\ \textsc{Cut} \quad T \Vdash \Gamma', [\![\Psi_y]\!] y : B, \{[\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : E}{z[v \triangleright (\boldsymbol{\nu} xw)\,(P \mid S)].T \Vdash \Gamma', [^{z}\Delta][\![\Psi_y]\!] y : B, \{[^{z}\Delta_i][\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : D \otimes^{y\tilde{u}} E}\ \otimes$$

This works because the labels in the left premiss are irrelevant to the conclusion, since the Δ's and the D in the conclusion are not annotated (inside boxes and antecedent of $\otimes$ resp.).

Case $Q = z[v \triangleright S].T$ and $w \in \mathtt{fn}(T)$. Assume we are in the following context

$$\dfrac{S \Vdash \Delta', \{\Delta_i\}_i, v : C \quad T \Vdash \Gamma', [\![\Psi_1]\!][^{z}w : A^{\perp}][\![\Psi_2]\!] y : B, \{[\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : D}{z[v \triangleright S].T \Vdash \Gamma', [^{z}\Delta'][\![\Psi_1]\!][^{z}w : A^{\perp}][\![\Psi_2]\!] y : B, \{[^{z}\Delta_i][\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : C \otimes^{y\tilde{u}} D}\ \otimes$$

By applying the induction hypothesis to P and T (which indeed is a subprocess of Q), we obtain R_0 such that

$$R_0 \Vdash^{\textsc{Cut}} \Gamma', [\![\Psi_1]\!][^{z}\Delta][\![\Psi_2]\!] y : B, \{[\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : D$$

and $\mathtt{rank}(R_0) = \mathtt{size}(A)$. So we can take $R = z[v \triangleright S].R_0$ and indeed,

$$R \Vdash^{\textsc{Cut}} \Gamma', [^{z}\Delta'][\![\Psi_1]\!][^{z}\Delta][\![\Psi_2]\!] y : B, \{[^{z}\Delta_i][\![\Psi_{u_i}]\!] u_i : C_i\}_i, [\![\Psi_z]\!] z : C \otimes^{y\tilde{u}} D$$

When Q is of any other shape we can apply the induction hypothesis in a similar (or even simpler) way. □

In the binary case, it is possible to prove a stronger version of this lemma where Cut is not needed to construct R, hence its rank is trivially null.

Lemma 12. *Let $P \Vdash z : A^{\perp}, x : A$ and $Q \Vdash \Gamma, [\![\Psi_1]\!][^{u}w : A^{\perp}][\![\Psi_2]\!] y : B$. Then, there exists R such that $R \Vdash \Gamma, [\![\Psi_1]\!][^{u}z : A^{\perp}][\![\Psi_2]\!] y : B$.*

Proof. By induction on the structure of Q.

Because of the shape of its typing environment, Q cannot be of the shape $a \leftrightarrow b$ nor of the shape $a[]$.

When $Q = u[v \triangleright S].T$ and $w \in \mathtt{fn}(S)$, we can take $R = u[x \triangleright P].T$.

When $Q = u[v \triangleright S].T$ and $w \in \mathtt{fn}(T)$, by applying the induction hypothesis to P and T (subprocess of Q), we obtain R_0 and take $R = u[v \triangleright S].R_0$.

When Q is of any other shape we can apply the induction hypothesis in a similar (simpler) way.

C.3 Admissibility of Cut

Theorem 13 (Admissibility of Cut). *Let $P \Vdash \Gamma_1, [\![\Psi_1]\!] x : A$ and $Q \Vdash \Gamma_2, [\![\Psi_2]\!] y : A^{\perp}$. For any Γ such that*

$$\{\Gamma_1 \triangleright [\![\Psi_1]\!] x : A,\ \Gamma_2 \triangleright [\![\Psi_2]\!] y : A^{\perp}\} \longrightarrow^{*}_{\mathsf{distr}} \longrightarrow^{*}_{\mathsf{subst}} \Gamma$$

there exists $R \Vdash \Gamma$.

Proof. We proceed by lexicographic induction on the structures of A, P and Q. That is, the induction hypothesis may be applied whenever (i) the rank of the cut gets smaller, or (ii) the rank stays the same and the cut is applied to at least one smaller process while the other stays the same.

Axiom

$$\dfrac{\dfrac{}{z \leftrightarrow x \Vdash z : a^{\perp}, x : a}\ \textsc{Ax} \qquad Q \Vdash [\![\Psi_2]\!] y : a^{\perp}, \Gamma_2}{(\boldsymbol{\nu} xy)\,(z \leftrightarrow x \ \mid\ Q) \Vdash [\![\Psi_2]\!] z : a^{\perp}, \Gamma_2[z/y]}\ \textsc{Cut} \quad \longrightarrow_{\beta} \quad Q[z/y] \Vdash [\![\Psi_2]\!] z : a^{\perp}, \Gamma_2[z/y]$$

Units

$$\dfrac{\dfrac{}{x[] \Vdash \{[^{x}*]u_i : \cdot\}_i, x : \mathbf{1}^{\tilde{u}}}\ \mathbf{1} \qquad \dfrac{Q \Vdash [\![\Psi_2]\!][^{v}*] y : \cdot, \Gamma_2, [\![\Psi_c]\!] v : C\{\mathbf{1}^{y\tilde{z}}\}}{y().Q \Vdash [\![\Psi_2]\!] y : \perp^{v}, \Gamma_2, [\![\Psi_c]\!] v : C\{\mathbf{1}^{y\tilde{z}}\}}\ \perp}{(\boldsymbol{\nu} xy)\,(x[] \ \mid\ y().Q) \Vdash \Gamma}\ \textsc{Cut}$$

$$\longrightarrow_{\beta} \quad \dfrac{Q \Vdash [\![\Psi_2]\!][^{v}*] y : \cdot, \Gamma_2, [\![\Psi_c]\!] v : C\{\mathbf{1}^{y\tilde{z}}\}}{Q^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}} \Vdash \{[\![\Psi_{u_i}]\!][^{v}*] u_i : \cdot\}_i, \Gamma_2^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}}, [\![\Psi_c]\!] v : C^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}}\{\mathbf{1}^{\tilde{u}\tilde{z}}\}}\ y \rightsquigarrow \tilde{u}$$

This is a special rule: It is not a uniform substitution replacing y by $\tilde{u}$ but it reproduces the changes performed by the distribution of Ψ over the u_is. Because $[\![\Psi]\!]$ is distributed from a terminated endpoint y to another list of endpoints in $\tilde{u}$ that are also terminated, this rule is trivially admissible.

$$\left\{ \begin{array}{r} \{[^{x}*]u_i : \cdot\}_i \rhd x : \mathbf{1}^{\tilde{u}} \\ \Gamma_2, [\![\Psi_c]\!] v : C\{\mathbf{1}^{\tilde{z}_1 y \tilde{z}_2}\} \rhd [\![\Psi]\!] y : \perp^{v} \end{array} \right\} \rightarrow^{*}_{\mathsf{distr}} \left\{ \begin{array}{r} \{[\![\Psi_{u_i}]\!][^{x}*]u_i : \cdot\}_i \rhd x : \mathbf{1}^{\tilde{u}} \\ \Gamma_2^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}}, [\![\Psi_c]\!] v : C^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}}\{\mathbf{1}^{\tilde{z}_1 y \tilde{z}_2}\} \rhd y : \perp^{v} \end{array} \right\}$$

$$\rightarrow_{\mathsf{subst}} \{[\![\Psi_{u_i}]\!][^{v}*]u_i : \cdot\}_i, \Gamma_2^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}}, [\![\Psi_c]\!] v : C^{\mathsf{y}\rightsquigarrow\tilde{\mathsf{u}}}\{\mathbf{1}^{\tilde{z}_1 \tilde{u} \tilde{z}_2}\} = \Gamma$$

Multiplicatives

$$\dfrac{\dfrac{P \Vdash d : A^{\perp}, a : A \quad Q \Vdash \Gamma_1, [\![\Psi_u]\!] u : C, [\![\Psi_1]\!] x : B}{x[a \rhd P].Q \Vdash \Gamma_1, [^{x} d : A^{\perp}][\![\Psi_u]\!] u : C, [\![\Psi_1]\!] x : A \otimes^{u} B}\ \otimes \qquad \dfrac{R \Vdash [\![\Psi_2]\!][^{v} c : A^{\perp}] y : B^{\perp}, \Gamma_2, [\![\Psi_v]\!] v : D\{\otimes^{y}\}}{y(c).R \Vdash [\![\Psi_2]\!] y : A^{\perp} \parr^{v} B^{\perp}, \Gamma_2, [\![\Psi_v]\!] v : D\{\otimes^{y}\}}\ \parr}{(\boldsymbol{\nu} xy)\,(x[a \rhd P].Q \ \mid\ y(c).R) \Vdash \Gamma}\ \textsc{Cut}$$

$$\longrightarrow_{\beta} \quad \dfrac{Q \Vdash \Gamma_1, [\![\Psi_u]\!] u : C, [\![\Psi_1]\!] x : B \quad (\boldsymbol{\nu} a \rhd c)\,(P \ \mid\ R) \Vdash [\![\Psi_2]\!][^{v} d : A^{\perp}] y : B^{\perp}, \Gamma_2, [\![\Psi_v]\!] v : D\{\otimes^{y}\}}{(\boldsymbol{\nu} xy)\,(Q \ \mid\ (\boldsymbol{\nu} a \rhd c)\,(P \ \mid\ R)) \Vdash \Gamma}\ \textsc{Cut}$$

By Lemma 12, we know that $S \Vdash [\![\Psi_2]\!][^{v} d : A^{\perp}] y : B^{\perp}, \Gamma_2, [\![\Psi_v]\!] v : D\{\otimes^{y}\}$, which gives us the right premiss above. Then, the new Cut (on B) is admissible by induction hypothesis on the rank as $\mathtt{size}(B) < \mathtt{size}(A \otimes^{u} B)$.

$$\left\{ \begin{array}{r} \Gamma_1, [^{x} d : A^{\perp}][\![\Psi_u]\!] u : C \rhd [\![\Psi_1]\!] x : A \otimes^{u} B \\ \Gamma_2, [\![\Psi_v]\!] v : D\{\otimes^{y}\} \rhd [\![\Psi_2]\!] y : A^{\perp} \parr^{v} B^{\perp} \end{array} \right\} \rightarrow^{*}_{\mathsf{distr}} \left\{ \begin{array}{r} \Gamma_1'', [\![\Psi_{2u}]\!][^{x} d : A^{\perp}][\![\Psi_u]\!] u : C'' \rhd x : A \otimes^{u} B \\ \Gamma_2'', [\![\Psi_{1v}]\!][\![\Psi_v]\!] v : D''\{\otimes^{y}\} \rhd y : A^{\perp} \parr^{v} B^{\perp} \end{array} \right\}$$

$$\to_{\mathsf{subst}} \left\{ \begin{array}{c} \Gamma_1'', [\![\Psi_{2u}]\!][^v d : A^\perp][\![\Psi_u]\!]u : C'' \rhd x : B \\ \Gamma_2'', [\![\Psi_{1v}]\!][\![\Psi_v]\!]v : D''\{\otimes^u\} \rhd y : B^\perp \end{array} \right\} \to^*_{\mathsf{subst}} \Gamma$$

On the other hand:

$$\left\{ \begin{array}{c} \Gamma_1, [\![\Psi_u]\!]u : C \rhd [\![\Psi_1]\!]x : B \\ \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^y\} \rhd [\![\Psi_2]\!][^v d : A^\perp]y : B^\perp \end{array} \right\} \to^*_{\mathsf{distr}} \left\{ \begin{array}{c} \Gamma_1'', [\![\Psi_{2u}]\!] \bullet [\![\Psi_u]\!]u : C'' \rhd x : B \\ \Gamma_2'', [\![\Psi_{1v}]\!] \bullet [\![\Psi_v]\!]v : D''\{\otimes^y\} \rhd [^v d : A^\perp]y : B^\perp \end{array} \right\}$$

$$\to_{\mathsf{distr}} \left\{ \begin{array}{c} \Gamma_1'', [\![\Psi_{2u}]\!][^v d : A^\perp][\![\Psi_u]\!]u : C'' \rhd x : B \\ \Gamma_2'', [\![\Psi_{1v}]\!][\![\Psi_v]\!]v : D''\{\otimes^u\} \rhd y : B^\perp \end{array} \right\} \to^*_{\mathsf{subst}} \Gamma$$

Multiplicatives with gathering

$$\dfrac{\dfrac{P \Vdash \{\Delta_i\}_i, a : A \quad Q \Vdash \Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_1]\!]x : B}{x[a \rhd P].Q \Vdash \Gamma_1, \{[^x \Delta_i][\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_1]\!]x : A \otimes^{\tilde{u}} B} \otimes \quad \dfrac{R \Vdash [\![\Psi_2]\!][^v c : A^\perp]y : B^\perp, \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\}}{y(c).R \Vdash [\![\Psi_2]\!]y : A^\perp \parr^v B^\perp, \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\}} \parr}{(\boldsymbol{\nu} xy)\,(x[a \rhd P].Q \mid y(c).R) \Vdash \Gamma} \textsc{Cut}$$

With Γ being defined as one of the possible rewriting:

$$\left\{ \begin{array}{l} \Gamma_1, \{[^x \Delta_i][\![\Psi_{u_i}]\!]u_i : C_i\}_i \rhd [\![\Psi_1]\!]x : A \otimes^{\tilde{u}} B \\ \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\} \rhd [\![\Psi_2]\!]y : A^\perp \parr^v B^\perp \end{array} \right\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

We want to reduce the cut as follows:

$$\dfrac{Q \Vdash \Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_1]\!]x : B \quad S \Vdash [\![\Psi_2]\!][^v \{\Delta_i\}_i]y : B^\perp, \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\}}{(\boldsymbol{\nu} xy)\,(Q \mid (\boldsymbol{\nu} a \rhd c)\,(P \mid R)) \Vdash \Gamma} \textsc{Cut}$$

By Lemma 11, we know that $S = (\boldsymbol{\nu} a \rhd c)\,(P \mid R)$ exists, that

$$S \Vdash^{\textsc{Cut}} [\![\Psi_2]\!][^v \{\Delta_i\}_i]y : B^\perp, \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\}$$

and that its rank is equal to the size of A. By induction hypothesis any cut in the derivation of S is admissible as $\mathtt{rank}(S) = \mathtt{size}(A) < \mathtt{size}(A \otimes^{\tilde{u}} B)$. That means in fact that

$$S \Vdash [\![\Psi_2]\!][^v \{\Delta_i\}_i]y : B^\perp, \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\}$$

without the need for cut, which gives us the right premiss.

Then, we need to check that we also get Γ as conclusion, namely that we get

$$\left\{ \begin{array}{l} \Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\}_i \rhd [\![\Psi_1]\!]x : B \\ \Gamma_2, [\![\Psi_v]\!]v : D\{\otimes^{y\tilde{z}}\} \rhd [\![\Psi_2]\!][^v \{\Delta_i\}_i]y : B^\perp \end{array} \right\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

Finally, the reduced Cut (on B) is of course admissible by induction hypothesis on the rank as $\mathtt{size}(B) < \mathtt{size}(A \otimes^{\tilde{u}} B)$.

Additives

$$\dfrac{\dfrac{P \Vdash \ldots, [\![\Psi_1]\!][^{\tilde{u}}\mathcal{L}]x : A \quad Q \Vdash \ldots, [\![\Psi_1]\!][^{\tilde{u}}\mathcal{R}]x : B}{x.\mathsf{case}(P,Q) \Vdash \Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\{\oplus^x\}\}_i, [\![\Psi_1]\!]x : A \,\&^{\tilde{u}}\, B}\,\& \quad \dfrac{R \Vdash [\![\Psi_2]\!]y : A^\perp, \Gamma_2, [\![\Psi_v]\!]v : D}{y[\mathsf{inl}].R \Vdash [\![\Psi_2]\!]y : A^\perp \oplus^v B^\perp, \Gamma_2, [^y\mathcal{L}][\![\Psi_v]\!]v : D}\,\oplus_l}{(\boldsymbol{\nu} xy)\,(x.\mathsf{case}(P,Q) \;\mid\; y[\mathsf{inl}].R) \Vdash \Gamma}\,\textsc{Cut}$$

$$\left\{\begin{array}{ll}\Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\{\oplus^x\}\}_i & \rhd [\![\Psi_1]\!]x : A \,\&^{\tilde{u}}\, B \\ \Gamma_2, [^y\mathcal{L}][\![\Psi_v]\!]v : D & \rhd [\![\Psi_2]\!]y : A^\perp \oplus^v B^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \left\{\begin{array}{ll}\Gamma_1', \{[\![\Psi_{2u_i}]\!][\![\Psi_{u_i}]\!]u_i : C_i'\{\oplus^x\}\}_i & \rhd x : A \,\&^{\tilde{u}}\, B \\ \Gamma_2', [\![\Psi_{1v}]\!][^y\mathcal{L}][\![\Psi_v]\!]v : D' & \rhd y : A^\perp \oplus^v B^\perp\end{array}\right\}$$

$$\longrightarrow_{\mathsf{subst}} \left\{\begin{array}{ll}\Gamma_1', \{[\![\Psi_{2u_i}]\!][\![\Psi_{u_i}]\!]u_i : C_i'\{\oplus^v\}\}_i & \rhd x : B \\ \Gamma_2', [\![\Psi_{1v}]\!][^{\tilde{u}}\mathcal{L}][\![\Psi_v]\!]v : D' & \rhd y : B^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

$$\dfrac{P \Vdash \Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\{\oplus^x\}\}_i, [\![\Psi_1]\!][^{\tilde{u}}\mathcal{L}]x : A \quad R \Vdash [\![\Psi_2]\!]y : A^\perp, \Gamma_2, [\![\Psi_v]\!]v : D}{(\boldsymbol{\nu} xy)\,(P \;\mid\; R) \Vdash \Gamma}\,\textsc{Cut}$$

$$\left\{\begin{array}{ll}\Gamma_1, \{[\![\Psi_{u_i}]\!]u_i : C_i\{\oplus^x\}\}_i & \rhd [\![\Psi_1]\!][^{\tilde{u}}\mathcal{L}]x : A \\ \Gamma_2, [\![\Psi_v]\!]v : D & \rhd [\![\Psi_2]\!]y : A^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \left\{\begin{array}{ll}\Gamma_1', \{[\![\Psi_{2u_i}]\!] \bullet [\![\Psi_{u_i}]\!]u_i : C_i'\{\oplus^x\}\}_i & \rhd [^{\tilde{u}}\mathcal{L}]x : A \\ \Gamma_2', [\![\Psi_{1v}]\!] \bullet [\![\Psi_v]\!]v : D' & \rhd y : A^\perp\end{array}\right\}$$

$$\longrightarrow_{\mathsf{distr}} \left\{\begin{array}{ll}\Gamma_1', \{[\![\Psi_{2u_i}]\!][\![\Psi_{u_i}]\!]u_i : C_i'\{\oplus^v\}\}_i & \rhd x : B \\ \Gamma_2', [\![\Psi_{1v}]\!][^{\tilde{u}}\mathcal{L}][\![\Psi_v]\!]v : D' & \rhd y : B^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

Exponentials

$$\dfrac{\dfrac{P \Vdash \{u_i : ?^a B_i\}_i, [^{\tilde{u}}\mathcal{Q}]a : A}{!x(a).P \Vdash \{u_i : ?^x B_i\}_i[x/a], x : !^{\tilde{u}} A}\,! \quad \dfrac{Q \Vdash \Gamma_2, [\![\Psi_z]\!]z : C, [\![\Psi_2]\!]b : A^\perp}{?y[b].Q \Vdash \{\Gamma_2, [^y\mathcal{Q}][\![\Psi_z]\!]z : C\}[y/b], [\![\Psi_2]\!]y : ?^z A^\perp}\,?}{(\boldsymbol{\nu} xy)\,(!x(a).P \;\mid\; ?y[b].Q) \Vdash \Gamma}\,\textsc{Cut}$$

$$\left\{\begin{array}{ll}\{u_i : ?^x B_i\}_i[x/a] & \rhd x : !^{\tilde{u}} A \\ \{\Gamma_2, [^y\mathcal{Q}][\![\Psi_z]\!]z : C\}[y/b] & \rhd [\![\Psi_2]\!]y : ?^z A^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \left\{\begin{array}{ll}\{[\![\Psi_{2u_i}]\!]u_i : ?^x B_i'\}_i[x/a] & \rhd x : !^{\tilde{u}} A \\ \{\Gamma_2', [^y\mathcal{Q}][\![\Psi_z]\!]z : C'\}[y/b] & \rhd y : ?^z A^\perp\end{array}\right\}$$

$$\longrightarrow_{\mathsf{subst}} \left\{\begin{array}{ll}\{[\![\Psi_{2u_i}]\!]u_i : ?^z B_i'\}_i[x/a] & \rhd x : A \\ \{\Gamma_2', [^{\tilde{u}}\mathcal{Q}][\![\Psi_z]\!]z : C'\}[y/b] & \rhd y : A^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

$$\longrightarrow_\beta \quad \dfrac{P \Vdash \{u_i : ?^a B_i\}_i, [^{\tilde{u}}\mathcal{Q}]a : A \quad Q \Vdash \Gamma_2, [\![\Psi_z]\!]z : C, [\![\Psi_2]\!]b : A^\perp}{\Vdash \Gamma}\,\textsc{Cut}$$

$$\left\{\begin{array}{ll}\{u_i : ?^a B_i\}_i & \rhd [^{\tilde{u}}\mathcal{Q}]a : A \\ \Gamma_2, [\![\Psi_z]\!]z : C & \rhd [\![\Psi_2]\!]b : A^\perp\end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \left\{\begin{array}{ll}\{[\![\Psi_{2u_i}]\!]u_i : ?^a B_i'\}_i & \rhd [^{\tilde{u}}\mathcal{Q}]a : A \\ \Gamma_2', [\![\Psi_z]\!]z : C' & \rhd b : A^\perp\end{array}\right\}$$

$$\longrightarrow_{\mathsf{distr}} \left\{ \begin{array}{l} \{[\![\Psi_{2u_i}]\!]u_i : ?^z B'_i\}_i \rhd a : A \\ \Gamma'_2, [^{\tilde{u}}\mathcal{Q}][\![\Psi_z]\!]z : C' \rhd b : A^\perp \end{array} \right\} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

Commutative $\otimes$ case

$$\dfrac{P \Vdash \Gamma_1, \{[\![\Psi_{w_j}]\!]w_j : E_j\}_j, [\![\Psi_1]\!]x : A \quad \dfrac{Q \Vdash \{\Delta_i\}_i, \Sigma, v : B \quad R \Vdash \Gamma_2, \{[\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_2]\!]y : A^\perp, [\![\Psi_z]\!]z : D}{z[v \rhd Q].R \Vdash \Gamma_2, \{[^z\Delta_i][\![\Psi_{u_i}]\!]u_i : C_i\}_i, [^z\Sigma][\![\Psi_2]\!]y : A^\perp, [\![\Psi_z]\!]z : B \otimes^{\tilde{u}y} D} \otimes}{(\boldsymbol{\nu} xy)\,(P \mid (z[v \rhd Q].R)) \Vdash \Gamma} \text{Cu}$$

$$\left\{ \begin{array}{ll} \Gamma_1, \{[\![\Psi_{w_j}]\!]w_j : E_j\}_j & \rhd [\![\Psi_1]\!]x : A \\ \Gamma_2, \{[^z\Delta_i][\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_z]\!]z : B \otimes^{\tilde{u}y} D & \rhd [^z\Sigma][\![\Psi_2]\!]y : A^\perp \end{array} \right\}$$

$$\longrightarrow_{\mathsf{distr}} \left\{ \begin{array}{ll} \Gamma_1, \{[^z\Sigma_j] \bullet [\![\Psi_{w_j}]\!]w_j : E_j\}_j & \rhd [\![\Psi_1]\!]x : A \\ \Gamma_2, \{\bullet[^z\Delta_i][\![\Psi_{u_i}]\!]u_i : C_i\}_i, \bullet[\![\Psi_z]\!]z : B \otimes^{\tilde{u}\tilde{w}} D & \rhd [\![\Psi_2]\!]y : A^\perp \end{array} \right\} \quad \text{with} \quad \Sigma = \{\Sigma_j\}_j$$

$$\longrightarrow^*_{\mathsf{distr}} \left\{ \begin{array}{ll} \Gamma'_1, \{[^z\Sigma_j][\![\Psi_{2w_j}]\!][\![\Psi_{w_j}]\!]w_j : E'_j\}_j & \rhd x : A \\ \Gamma'_2, \{[\![\Psi_{1u_i}]\!][^z\Delta_i][\![\Psi_{u_i}]\!]u_i : C'_i\}_i, [\![\Psi_{1z}]\!][\![\Psi_z]\!]z : B \otimes^{\tilde{u}\tilde{w}} D' & \rhd y : A^\perp \end{array} \right\} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

We want to cut P and R together, as we can ultimately apply the induction hypothesis to this Cut given that R is a subprocess of $z[v \rhd Q].R$:

$$\dfrac{P \Vdash \Gamma_1, \{[\![\Psi_{w_j}]\!]w_j : E_j\}_j, [\![\Psi_1]\!]x : A \quad R \Vdash \Gamma_2, \{[\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_2]\!]y : A^\perp, [\![\Psi_z]\!]z : D}{(\boldsymbol{\nu} xy)\,(P \mid R) \Vdash \Gamma_0} \text{Cut}$$

This gives us conclusion Γ_0 obtained by reproducing the $\longrightarrow^*_{\mathsf{distr}}$ rewriting sequence on $[\![\Psi_1]\!]$ and $[\![\Psi_2]\!]$ that we had above, followed by the deterministic $\longrightarrow^*_{\mathsf{subst}}$ sequence on $x : A$ and $y : A^\perp$.

$$\left\{ \begin{array}{ll} \Gamma_1, \{[\![\Psi_{w_j}]\!]w_j : E_j\}_j & \rhd [\![\Psi_1]\!]x : A \\ \Gamma_2, \{[\![\Psi_{u_i}]\!]u_i : C_i\}_i, [\![\Psi_z]\!]z : D & \rhd [\![\Psi_2]\!]y : A^\perp \end{array} \right\}$$

$$\longrightarrow^*_{\mathsf{distr}} \left\{ \begin{array}{ll} \Gamma'_1, \{[\![\Psi_{2w_j}]\!][\![\Psi_{w_j}]\!]w_j : E'_j\}_j & \rhd x : A \\ \Gamma'_2, \{[\![\Psi_{1u_i}]\!][\![\Psi_{u_i}]\!]u_i : C'_i\}_i, [\![\Psi_{1z}]\!][\![\Psi_z]\!]z : D' & \rhd y : A^\perp \end{array} \right\} \longrightarrow^*_{\mathsf{subst}} \Gamma_0$$

Finally we can reconstruct the original conclusion Γ by applying the $\otimes$ rule.

$$\dfrac{Q \Vdash \{\Delta_i\}_i, \Sigma, v : B \quad (\boldsymbol{\nu} xy)\,(P \mid R) \Vdash \Gamma_0}{z[v \rhd Q].(\boldsymbol{\nu} xy)\,(P \mid R) \Vdash \Gamma} \otimes$$

Commutative $\parr$ case

$$\dfrac{P \Vdash \Gamma_1, [\![\Psi_1]\!]x : A \quad \dfrac{Q \Vdash \Gamma_2, [\![\Psi_u]\!][^z v : B]u : C, [\![\Psi_2]\!]y : A^\perp}{u(v).Q \Vdash \Gamma_2, [\![\Psi_u]\!]u : B \parr^z C, [\![\Psi_2]\!]y : A^\perp} \parr}{(\boldsymbol{\nu} xy)\,(P \mid u(v).Q) \Vdash \Gamma} \text{Cut}$$

$$\left\{\begin{array}{ll} \Gamma_1 & \rhd [\![\Psi_1]\!]x : A \\ \Gamma_2, [\![\Psi_u]\!]u : B \parr^z C & \rhd [\![\Psi_2]\!]y : A^\perp \end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

$$\dfrac{P \Vdash \Gamma_1, [\![\Psi_1]\!]x : A \quad Q \Vdash \Gamma_2, [\![\Psi_u]\!][^z v : B]u : C, [\![\Psi_2]\!]y : A^\perp}{\dfrac{(\boldsymbol{\nu} xy)\,(P \mid Q) \Vdash \Gamma_0}{u(v).(\boldsymbol{\nu} xy)\,(P \mid Q) \Vdash \Gamma}\ \parr}\ \textsc{Cut}$$

$$\left\{\begin{array}{ll} \Gamma_1 & \rhd [\![\Psi_1]\!]x : A \\ \Gamma_2, [\![\Psi_u]\!][^z v : B]u : C & \rhd [\![\Psi_2]\!]y : A^\perp \end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma_0$$

Commutative $\perp$ case

$$\dfrac{P \Vdash \Gamma_1, [\![\Psi_1]\!]x : A \quad \dfrac{Q \Vdash [\![\Psi_2]\!]y : A^\perp, \Gamma_2, [\![\Psi_z]\!][^v *]z : \cdot, [\![\Psi_v]\!]v : C\{\mathbf{1}^{z\tilde{u}}\}}{z().Q \Vdash [\![\Psi_2]\!]y : A^\perp, \Gamma_2, [\![\Psi_z]\!]z : \perp^v, [\![\Psi_v]\!]v : C\{\mathbf{1}^{z\tilde{u}}\}}\ \perp}{(\boldsymbol{\nu} xy)\,(P \mid z().Q) \Vdash \Gamma}\ \textsc{Cut}$$

$$\left\{\begin{array}{lll} \Gamma_1 & \rhd & [\![\Psi_1]\!]x : A \\ \Gamma_2, [\![\Psi_z]\!]z : \perp^v, [\![\Psi_v]\!]v : C\{\mathbf{1}^{z\tilde{u}}\} & \rhd & [\![\Psi_2]\!]y : A^\perp \end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

$$\dfrac{P \Vdash \Gamma_1, [\![\Psi_1]\!]x : A \quad Q \Vdash [\![\Psi_2]\!]y : A^\perp, \Gamma_2, [\![\Psi_z]\!][^v *]z : \cdot, [\![\Psi_v]\!]v : C\{\mathbf{1}^{z\tilde{u}}\}}{(\boldsymbol{\nu} xy)\,(P \mid Q) \Vdash \Gamma_0}\ \textsc{Cut}$$

$$\left\{\begin{array}{lll} \Gamma_1 & \rhd & [\![\Psi_1]\!]x : A \\ \Gamma_2, [\![\Psi_z]\!][^v *]z : \cdot, [\![\Psi_v]\!]v : C\{\mathbf{1}^{z\tilde{u}}\} & \rhd & [\![\Psi_2]\!]y : A^\perp \end{array}\right\} \longrightarrow^*_{\mathsf{distr}} \longrightarrow^*_{\mathsf{subst}} \Gamma$$

$$\dfrac{(\boldsymbol{\nu} xy)\,(P \mid Q) \Vdash \Gamma_0}{z().(\boldsymbol{\nu} xy)\,(P \mid Q) \Vdash \Gamma}\ \perp$$

D MCutF admissibility

Here are all cases for MCutF elimination.

$$\dfrac{\{P_j \vdash \Delta_i, y_j : A_j\}_{j\le m} \quad \{R_i \vdash \Sigma_i, x_i : B_i\}_{i\le n} \quad Q \Vdash \{[\![\Psi_i]\!]x_i : B_i^\perp\}_{i\le n}, \{[\![\Psi_i]\!]x_i : \cdot\}_{n<i\le p}}{(\boldsymbol{\nu}\tilde{x} : (\boldsymbol{\nu}\tilde{y})\,Q[\tilde{P}])\,\tilde{R} \vdash \tilde{\Delta}, \tilde{\Sigma}}\ \textsc{MCutF}$$

The rule has a developed side condition: $\bigcup_{i\le p} \Psi_i \setminus \{\mathcal{L}, \mathcal{R}, \mathcal{Q}, *\} = \{y_j : A_j^\perp\}_{j\le m}$.

Send Message ($\otimes$).

$$\dfrac{\dfrac{P \vdash \Delta, y : A \quad R \vdash \Sigma, x : B}{x[y \triangleright P].R \vdash \Delta, \Sigma, x : A \otimes B}\otimes \quad \ldots \quad \dfrac{Q \Vdash [\![\Psi]\!][^{x_k} y : A^\perp]x : B^\perp, \Gamma}{x(y).Q \Vdash [\![\Psi]\!]x : A^\perp \parr^{x_k} B^\perp, \Gamma}\parr}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu}\tilde{y})\, x(y).Q[\tilde{P}])\,(x[y \triangleright P].R \mid \tilde{R}) \vdash \Delta, \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

$$\Longrightarrow$$

$$\dfrac{P \vdash \Delta, y : A \quad R \vdash \Sigma, x : B \quad \ldots \quad Q \Vdash [\![\Psi]\!][^{x_k} y : A^\perp]x : B^\perp, \Gamma}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu} y\tilde{y})\, Q[P, \tilde{P}])\,(R \mid \tilde{R}) \vdash \Delta, \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

Receive Message ($\parr$).

$$\dfrac{\dfrac{P \vdash \Delta, z : A^\perp \quad R \vdash \Sigma, y : A, x : B}{x(y).R \vdash \Sigma, x : A \parr B}\parr \quad \ldots \quad \dfrac{S \Vdash z : A, y : A^\perp \quad Q \Vdash [\![\Psi_x]\!]x : B^\perp, \Gamma}{x[y \triangleright S].Q \Vdash [\![\Psi_x]\!]x : A^\perp \otimes^{x_k} B^\perp, [^x z : A][\![\Psi_k]\!]x_k : B_k^\perp, \Gamma - k}\otimes}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu} z\tilde{y})\, x[y \triangleright S].Q[P, \tilde{P}])\,(x(y).R \mid \tilde{R}) \vdash \Delta, \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

$$\Longrightarrow$$

$$\dfrac{\dfrac{R \vdash \Sigma, y : A, x : B \quad P \vdash \Delta, z : A^\perp \quad S \Vdash z : A, y : A^\perp}{(\boldsymbol{\nu} yz : S)\,(R \mid P) \Vdash \Sigma, \Delta, x : B}\ \text{MCutF} \quad \ldots \quad Q \Vdash [\![\Psi_x]\!]x : B^\perp, \Gamma}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu}\tilde{y})\, Q[\tilde{P}])\,((\boldsymbol{\nu} yz : S)\,(R \mid P) \mid \tilde{R}) \vdash \Delta, \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

Note that now the message (namely process P) has finally been delivered and it can be directly linked with a new MCutF.

Internal choice ($\oplus$).

$$\dfrac{\dfrac{R \vdash \Sigma, x : A}{x[\mathsf{inl}].R \vdash \Sigma, x : A \oplus B}\oplus_l \quad \ldots \quad \dfrac{Q \Vdash [\![\Psi_x]\!][^{x_k}\mathcal{L}]x : A^\perp, \Gamma \quad S \Vdash [\![\Psi_x]\!][^{x_k}\mathcal{R}]x : B^\perp, \Gamma}{x.\mathsf{case}(Q, S) \Vdash [\![\Psi_x]\!]x : A^\perp \,\&^{x_k}\, B^\perp, \Gamma}\,\&}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu}\tilde{y})\, x.\mathsf{case}(Q, S)[\tilde{P}])\,(x[\mathsf{inl}].R \mid \tilde{R}) \vdash \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

$$\Longrightarrow$$

$$\dfrac{R \vdash \Sigma, x : A \quad \ldots \quad Q \Vdash [\![\Psi_x]\!][^{x_k}\mathcal{L}]x : A^\perp, \Gamma}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu}\tilde{y})\, Q[\tilde{P}])\,(R \mid \tilde{R}) \vdash \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

External choice ($\&$).

$$\dfrac{\dfrac{R \vdash \Sigma, y : A \quad S \vdash \Sigma, x : B}{x.\mathsf{case}(R, S) \vdash \Sigma, x : A \,\&\, B}\,\& \quad \ldots \quad \dfrac{Q \Vdash [\![\Psi_x]\!]x : A^\perp, \Gamma}{x[\mathsf{inl}].Q \Vdash [\![\Psi_x]\!]x : A^\perp \oplus^{x_k} B^\perp, [^x\mathcal{L}][\![\Psi_k]\!]x_k : B_k^\perp, \Gamma - k}\oplus_l}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu}\tilde{y})\, x[\mathsf{inl}].Q[\tilde{P}])\,(x.\mathsf{case}(R, S) \mid \tilde{R}) \vdash \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

$$\Longrightarrow$$

$$\dfrac{R \vdash \Sigma, y : A \quad \ldots \quad Q \Vdash [\![\Psi_x]\!]x : A^\perp, \Gamma}{(\boldsymbol{\nu} x\tilde{x} : (\boldsymbol{\nu}\tilde{y})\, Q[\tilde{P}])\,(R \mid \tilde{R}) \vdash \Sigma, \tilde{\Delta}, \tilde{\Sigma}}\ \text{MCutF}$$

Axiom.

$$\dfrac{\dfrac{}{x_1 \leftrightarrow z_1 \vdash x_1 : a, z_1 : a^\perp}\ \textsc{Ax} \quad \dfrac{}{x_2 \leftrightarrow z_2 \vdash x_2 : a^\perp, z_2 : a}\ \textsc{Ax} \quad \dfrac{}{x_1 \leftrightarrow x_2 \Vdash x_1 : a^\perp, x_2 : a}\ \textsc{Ax}}{(\boldsymbol{\nu} x_1 x_2 : x_1 \leftrightarrow x_2)\,(x_1 \leftrightarrow z_1 \ \mid\ x_2 \leftrightarrow z_2) \vdash z_1 : a^\perp, z_2 : a}\ \textsc{MCutF}$$

$$\Longrightarrow \qquad \dfrac{}{z_1 \leftrightarrow z_2 \vdash z_1 : a^\perp, z_2 : a}\ \textsc{Ax}$$

Close (1).

$$\dfrac{\dfrac{}{x[] \vdash x : \mathbf{1}}\ \mathbf{1} \quad \ldots \quad \dfrac{Q \Vdash [\![\Psi_x]\!][^{x_k}*]x : \cdot, \Gamma}{x().Q \Vdash [\![\Psi_x]\!]x : \perp^{x_k}, \Gamma}\ \perp}{(\boldsymbol{\nu} x \tilde{x} : (\boldsymbol{\nu} \tilde{y})\, x().Q[\tilde{P}])\,(x[] \ \mid\ \tilde{R}) \vdash \tilde{\Delta}, \tilde{\Sigma}}\ \textsc{MCutF}$$

$$\Longrightarrow \qquad \dfrac{\ldots \quad Q \Vdash [\![\Psi_x]\!][^{x_k}*]x : \cdot, \Gamma}{(\boldsymbol{\nu} x \tilde{x} : (\boldsymbol{\nu} \tilde{y})\, Q[\tilde{P}])\,(\tilde{R}) \vdash \tilde{\Delta}, \tilde{\Sigma}}\ \textsc{MCutF}$$

Wait ($\perp$).

$$\dfrac{\dfrac{P \vdash \Delta}{x().P \vdash \Delta, x : \perp}\ \perp \quad \dfrac{}{x[] \Vdash x : \mathbf{1}^{\tilde{x}}, \{[^x*]x_i : \cdot\}_i}\ \mathbf{1}\ (\vec{u} \neq \varnothing)}{(\boldsymbol{\nu} x : x[])\,(x().P) \vdash \Delta}\ \textsc{MCutF}$$

$$\Longrightarrow \quad P \vdash \Delta$$

Theorem 14 (Admissibility of multi-cut). *If* $\{P_j \vdash \Delta_i, y_j : A_j\}_{j \le m}$ *and* $\{R_i \vdash \Sigma_i, x_i : B_i\}_{i \le n}$ *and* $Q \Vdash \{[\![\Psi_i]\!]x_i : B_i^\perp\}_{i \le n}, \{[\![\Psi_i]\!]x_i : \cdot\}_{n < i \le p}$ *then* $(\boldsymbol{\nu} \tilde{x} : (\boldsymbol{\nu} \tilde{y})\, Q[\tilde{P}])\, \tilde{R} \vdash \tilde{\Delta}, \tilde{\Sigma}$