Taking Linear Logic Apart

Wen Kokke
University of Edinburgh
Edinburgh, Scotland
wen.kokke@ed.ac.uk

Fabrizio Montesi
University of Southern Denmark
Odense, Denmark
fmontesi@imada.sdu.dk

Marco Peressotti
University of Southern Denmark
Odense, Denmark
peressotti@imada.sdu.dk

Process calculi based on logic, such as πDILL and CP, provide a foundation for deadlock-free concurrent programming. However, in previous work, there is a mismatch between the rules for constructing proofs and the term constructors of the π-calculus. We introduce Hypersequent Classical Processes (HCP), which addresses this mismatch using hypersequents (collections of sequents) to register parallelism in the typing judgements. We prove that HCP enjoys deadlock-freedom and a series of properties that relate it back to CP.

1 Introduction

Classical Processes (CP) (Wadler, 2012) is a process calculus inspired by the correspondence between the session-typed π-calculus and linear logic (Caires and Pfenning, 2010), where processes correspond to proofs, session types (communication protocols) to propositions, and communication to cut elimination. This correspondence allows for exchanging methods between the two fields. For example, the proof theory of linear logic can be used to guarantee progress for processes (Caires and Pfenning, 2010; Wadler, 2012).

The main attraction of CP is that its semantics are prescribed by the cut elimination procedure of Classical Linear Logic (CLL). This permits us to reuse the metatheory of linear logic "as is" to reason about the behaviour of processes. However, there is a mismatch between the structure of the proof terms of CLL and the term constructs of the standard π-calculus (Milner et al., 1992a,b). For instance, the term for output of a linear name is $x[y].(P \mid Q)$, which is read "send y over x and proceed as P in parallel to Q". Note that this is a single term constructor, which takes all four arguments at the same time. This is caused by directly adopting the ($\otimes$)-rule from CLL as the process calculus construct for sending: the ($\otimes$)-rule has two premises (corresponding to P and Q in the output term), and checks that they share no resources (in the output term, x can be used only by P, and y can be used only by Q).

There is no independent parallel term $(P \mid Q)$ in the grammar of CP terms. Instead, parallel composition shows up in any term which corresponds to a typing rule which splits the context. Even if we were to add an independent parallel composition via the Mix-rule, as suggested in the original presentation of CP (Wadler, 2012), there would be no way to allow the composed process P and Q to communicate as in the standard π-calculus, as there is no independent name restriction either! Instead, synchronisation is governed by the "cut" operator $(\nu x)(P \mid Q)$, which composes P and Q, enabling them to communicate along x. Worse, if we naively add an independent parallel composition as well as a name restriction, using the rules shown below, we lose cut elimination, and therefore deadlock-freedom!

$$\frac{P \vdash \Gamma \qquad Q \vdash \Delta}{P \mid Q \vdash \Gamma, \Delta}\text{Mix} \qquad \frac{P \vdash \Gamma, x : A, y : A^{\perp}}{(\nu xy)P \vdash \Gamma}\text{“Cut”}$$

Submitted to:
Linearity and TLLA 2018

This syntactic mismatch has an effect on the semantics as well. For instance, the β-reduction for output and input in CP is $(\nu x)(x[y].(P \mid Q) \mid x(y).R) \Longrightarrow (\nu y)(P \mid (\nu x)(Q \mid R))$. Here, the parallel composition $(P \mid Q)$ is of no relevance to this communication, yet the rule needs to inspect it to be able to nest the name restrictions appropriately in the resulting term.

In this paper, we introduce Hypersequent Classical Processes (HCP), which addresses this mismatch. The key insight is to register parallelism in the typing judgements using hypersequents (Avron, 1991), a technique from logic which generalises judgements from one sequent to many. This allows us to take apart the term constructs used in Classical Processes (CP) to more closely match those of the standard π-calculus. We proceed as follows. We start by introducing CP (Section 2). Then, we introduce HCP and prove it enjoys subject reduction and progress (Section 3). We prove that every CP process is an HCP process, and that HCP supports the same communication protocols as CP, and no more (Section 4). Finally, we discuss related work (Section 5).

2 Classical Processes

In this section, we introduce CP. In order to keep the discussion of HCP in Section 3 simple, we restrict ourselves to the multiplicative-additive subset of CP. We foresee no problems in extending the proofs in Section 3 to cover the remaining features of CP (polymophism and the exponentials).

2.1 Terms

The term language of CP is a variant of the π-calculus. The variables x, y, and z range over channel names. The construct $x{\leftrightarrow}y$ links two channels (Boreale, 1998; Sangiorgi, 1996), forwarding messages received on x to y and vice versa. The construct $(\nu x)(P \mid Q)$ creates a new channel x, and composes two processes, P and Q, which communicate on x, in parallel. Therefore, in $(\nu x)(P \mid Q)$ the name x is bound in both P and Q. In $x(y).P$ and $x[y].(P \mid Q)$, round brackets denote input, square brackets denote output. CP uses bound output (Sangiorgi, 1996), meaning that both input and output bind a new name. In $x(y).P$ the new name y is bound in P. In $x[y].(P \mid Q)$, the new name y is only bound in P, while x is only bound in Q.

Definition 2.1 (Terms). Process terms are given by the following grammar:

$P,Q,R ::=$	$x{\leftrightarrow}y$	link	$\mid$	$(\nu x)(P \mid Q)$	parallel composition, "cut"
$\mid$	$x[y].(P \mid Q)$	output	$\mid$	$x(y).P$	input
$\mid$	$x[].0$	halt	$\mid$	$x().P$	wait
$\mid$	$x \triangleleft \mathtt{inl}.P$	select left choice	$\mid$	$x \triangleleft \mathtt{inr}.P$	select right choice
$\mid$	$x \triangleright \{\mathtt{inl}: P; \mathtt{inr}: Q\}$	offer binary choice	$\mid$	$x \triangleright \{\}$	offer nullary choice

Terms in CP are identified up to structural congruence, which states that links are symmetric, and parallel compositions $(\nu x)(P \mid Q)$ are associative and commutative.

Definition 2.2 (Structural congruence). The structural congruence $\equiv$ is the congruence closure over terms which satisfies the following additional axioms:

$$\begin{array}{llll}
(\leftrightarrow\text{-sym}) & x{\leftrightarrow}y & \equiv & y{\leftrightarrow}x \\
(\nu\text{-comm}) & (\nu x)(P \mid Q) & \equiv & (\nu x)(Q \mid P) \\
(\nu\text{-assoc}) & (\nu x)(P \mid (\nu y)(Q \mid R)) & \equiv & (\nu y)((\nu x)(P \mid Q) \mid R) \ \text{ if } x \notin R \text{ and } y \notin P
\end{array}$$

Reductions relate processes with their reduced forms e.g., a reduction $P \Longrightarrow Q$ denotes that the process P can reduce to the process Q in a single step.

Definition 2.3 (Reduction). Reductions are described by the smallest relation $\Longrightarrow$ on process terms closed under rules below.

$$\begin{array}{lllll}
(\leftrightarrow) & (\nu x)(w{\leftrightarrow}x \mid P) & \Longrightarrow & P\{w/x\} & \\
(\beta{\otimes}{\parr}) & (\nu x)(x[y].(P \mid Q) \mid x(y).R) & \Longrightarrow & (\nu y)(P \mid (\nu x)(Q \mid R)) & \\
(\beta\mathbf{1}\bot) & (\nu x)(x[].0 \mid x().P) & \Longrightarrow & P & \\
(\beta{\oplus}{\&}_1) & (\nu x)(x \triangleleft \mathtt{inl}.P \mid x \triangleright \{\mathtt{inl} : Q; \mathtt{inr} : R\}) & \Longrightarrow & (\nu x)(P \mid Q) & \\
(\beta{\oplus}{\&}_2) & (\nu x)(x \triangleleft \mathtt{inr}.P \mid x \triangleright \{\mathtt{inl} : Q; \mathtt{inr} : R\}) & \Longrightarrow & (\nu x)(P \mid R) & \\
(\kappa{\otimes}_1) & (\nu x)(y[z].(P \mid Q) \mid R) & \Longrightarrow & y[z].((\nu x)(P \mid R) \mid Q) & \text{if } x \notin Q \\
(\kappa{\otimes}_2) & (\nu x)(y[z].(P \mid Q) \mid R) & \Longrightarrow & y[z].(P \mid (\nu x)(Q \mid R)) & \text{if } x \notin P \\
(\kappa{\parr}) & (\nu x)(y(z).P \mid R) & \Longrightarrow & y(z).(\nu x)(P \mid R) & \\
(\kappa\bot) & (\nu x)(y().P \mid R) & \Longrightarrow & y().(\nu x)(P \mid R) & \\
(\kappa{\oplus}_1) & (\nu x)(y \triangleleft \mathtt{inl}.P \mid R) & \Longrightarrow & y \triangleleft \mathtt{inl}.(\nu x)(P \mid R) & \\
(\kappa{\oplus}_2) & (\nu x)(y \triangleleft \mathtt{inr}.P \mid R) & \Longrightarrow & y \triangleleft \mathtt{inr}.(\nu x)(P \mid R) & \\
(\kappa{\&}) & (\nu x)(y \triangleright \{\mathtt{inl} : P; \mathtt{inr} : Q\} \mid R) & \Longrightarrow & y \triangleright \{\mathtt{inl} : (\nu x)(P \mid R); \mathtt{inr} : (\nu x)(Q \mid R)\} & \\
(\kappa\top) & (\nu x)(y \triangleright \{\} \mid R) & \Longrightarrow & y \triangleright \{\} &
\end{array}$$

$$\frac{P \Longrightarrow P'}{(\nu x)(P \mid Q) \Longrightarrow (\nu x)(P' \mid Q)}\ (\gamma\nu) \qquad \frac{P \equiv Q \qquad Q \Longrightarrow Q' \qquad Q' \equiv P'}{P \Longrightarrow P'}\ (\gamma{\equiv})$$

Relations $\Longrightarrow^{+}$ and $\Longrightarrow^{\star}$ are the transitive, and the reflexive, transitive closures of $\Longrightarrow$, respectively.

2.2 Types

Channels in CP are typed using a session type system which corresponds to classical linear logic.

Definition 2.4 (Types).

$$\begin{array}{rllcll}
A, B, C ::= & A \otimes B & \text{pair of independent processes} & \mid & \mathbf{1} & \text{unit for } \otimes \\
\mid & A \parr B & \text{pair of interdependent processes} & \mid & \bot & \text{unit for } \parr \\
\mid & A \oplus B & \text{internal choice} & \mid & \mathbf{0} & \text{unit for } \oplus \\
\mid & A \,\&\, B & \text{external choice} & \mid & \top & \text{unit for } \&
\end{array}$$

A channel of type $A \otimes B$ represents a pair of channels, which communicate with two independent processes—that is to say, two processes who share no channels. A process acting on a channel of type $A \otimes B$ will send one endpoint of a fresh channel, and then split into a pair of independent processes. One of these processes will be responsible for an interaction of type A over the fresh channel, while the other process continues to interact as B.

A channel of type $A \parr B$ represents a pair of interdependent channels, which are used within a single process. A process acting on a channel of type $A \parr B$ will receive a channel to act on, and communicate on its channels in whatever order it pleases. This means that the usage of one channel can depend on that of another—e.g., the interaction of type B could depend on the result of the interaction of type A, or vise versa, and if A and B are complex types, their interactions could likewise interweave in complex ways.

A process acting on a channel of type $A \oplus B$ either sends the value inl to select an interaction of type A or the value inr to select one of type B. A process acting on a channel of type $A \,\&\, B$ receives such a value, and then offers an interaction of either type A or B, correspondingly.

Duality plays a crucial role in both linear logic and session types. In CP, the two endpoints of a channel are assigned dual types. This ensures that, for instance, whenever a process sends across a channel, the process on the other end of that channel is waiting to receive. Each type A has a dual, written $A^\perp$. Duality is an involutive function i.e., $(A^\perp)^\perp = A$.

Definition 2.5 (Duality).

$$\begin{array}{lllllllllllll}
(A \otimes B)^\perp & = & A^\perp \parr B^\perp & \mathbf{1}^\perp & = & \perp & (A \parr B)^\perp & = & A^\perp \otimes B^\perp & \perp^\perp & = & \mathbf{1} \\
(A \oplus B)^\perp & = & A^\perp \mathbin{\&} B^\perp & \mathbf{0}^\perp & = & \top & (A \mathbin{\&} B)^\perp & = & A^\perp \oplus B^\perp & \top^\perp & = & \mathbf{0}
\end{array}$$

An environment associates channels with types. Names in environments must be unique, and two environments Γ and Δ can only be combined as Γ,Δ if $\mathrm{fv}(\Gamma) \cap \mathrm{fv}(\Delta) = \varnothing$.

Definition 2.6 (Environments). $\Gamma, \Delta, \Theta ::= \cdot \mid \Gamma, x : A$

A typing judgement associates a process with collections of typed channels.

Definition 2.7 (Typing judgements). A typing judgement $P \vdash x_1 : A_1, \ldots, x_n : A_n$ denotes that the process P communicates along channels x_1, ..., x_n following protocols A_1, ..., A_n. Typing judgements are derived using rules below.

Structural rules

$$\frac{}{x \leftrightarrow y \vdash x : A, y : A^\perp}\,\text{Ax} \qquad \frac{P \vdash \Gamma, x : A \qquad Q \vdash \Delta, x : A^\perp}{(\nu x)(P \mid Q) \vdash \Gamma, \Delta}\,\text{Cut}$$

Logical rules

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

$$\frac{P \vdash \Gamma}{x().P \vdash \Gamma, x : \perp}\,(\perp) \qquad \frac{}{x[].0 \vdash x : \mathbf{1}}\,(\mathbf{1})$$

$$\frac{P \vdash \Gamma, x : A}{x \triangleleft \mathtt{inl}.P \vdash \Gamma, x : A \oplus B}\,(\oplus_1) \qquad \frac{P \vdash \Gamma, x : B}{x \triangleleft \mathtt{inr}.P \vdash \Gamma, x : A \oplus B}\,(\oplus_2)$$

$$\frac{P \vdash \Gamma, x : A \qquad Q \vdash \Gamma, x : B}{x \triangleright \{\mathtt{inl} : P; \mathtt{inr} : Q\} \vdash \Gamma, x : A \mathbin{\&} B}\,(\&) \qquad (norulefor\mathbf{0}) \qquad \frac{}{x \triangleright \{\} \vdash \Gamma, x : \top}\,(\top)$$

2.3 Metatheory

CP enjoys subject reduction, termination, and progress (Wadler, 2012).

Lemma 2.8 (Preservation for $\equiv$). If $P \equiv Q$, then $P \vdash \Gamma$ iff $Q \vdash \Gamma$.

Proof. By induction on the derivation of $P \equiv Q$. □

Theorem 2.9 (Preservation). If $P \vdash \Gamma$ and $P \Longrightarrow Q$, then $Q \vdash \Gamma$.

Proof. By induction on the derivation of $P \Longrightarrow Q$. □

Theorem 2.10 (Termination). If $P \vdash \Gamma$, then there are no infinite $\Longrightarrow$-reduction sequences.

Proof. Every reduction reduces a single cut to zero, one or two cuts. However, each of these cuts is smaller, measured in the size of the cut term. Furthermore, each instance of the structural congruence preserves the size of the cut. Therefore, there cannot be an infinite $\Longrightarrow$-reduction sequence. □

Theorem 2.11 (Progress). If $P \vdash \Gamma$, then there exists a Q such that $P \Longrightarrow^{\star} Q$ and Q is not a cut.

Proof. By induction on the derivation of $P \vdash \Gamma$. If the last rule is Cut, there are four cases: a) if either side of the cut is an axiom, we apply ($\leftrightarrow$); b) if either side of the cut is itself a cut, we recursively eliminate the cut; c) if both sides are logical rules acting on the cut formula, we apply one of the β-rules; d) otherwise, at least one side is a logical rule acting on a formula other than the cut formula, in which case we apply one of the κ-rules. $\square$

3 Hypersequent Classical Processes

In this section, we introduce Hypersequent Classical Processes (HCP), a variant of CP which registers parallelism in the typing judgements using hypersequents, allowing us to take apart the monolithic term constructors of CP (e.g., $x[y].(P \mid Q)$) into the corresponding π-calculus term constructs.

3.1 Terms

The term language of HCP is a variant of CP where the term constructs have been taken apart into primitives which more closely resemble the π-calculus primitives.

Definition 3.1 (Terms).

$P, Q, R ::=$	$x{\leftrightarrow}y$	link	$\mid$	0	terminated process
$\mid$	$(\nu x)P$	name restriction, "cut"	$\mid$	$(P \mid Q)$	parallel composition, "mix"
$\mid$	$x[y].P$	output	$\mid$	$x(y).P$	input
$\mid$	$x[].P$	halt	$\mid$	$x().P$	wait
$\mid$	$x \triangleleft \mathtt{inl}.P$	select left choice	$\mid$	$x \triangleleft \mathtt{inr}.P$	select right choice
$\mid$	$x \triangleright \{\mathtt{inl} : P; \mathtt{inr} : Q\}$	offer binary choice	$\mid$	$x \triangleright \{\}$	offer nullary choice

A pleasant effect of our updated syntax is that it makes our structural congruence much more standard: it has associativity, commutativity, and a unit for parallel composition, commutativity of name restrictions, and scope extrusion.

Definition 3.2 (Structural congruence). The structural congruence $\equiv$ is the congruence closure over terms which satisfies the following additional axioms:

$$\begin{array}{llllll}
(\leftrightarrow\text{-sym}) & x{\leftrightarrow}y \equiv y{\leftrightarrow}x & (\text{halt}) & P \mid 0 \equiv P & & \\
(\mid\text{-comm}) & P \mid Q \equiv Q \mid P & (\mid\text{-assoc}) & P \mid (Q \mid R) \equiv (P \mid Q) \mid R & & \\
(\nu\text{-comm}) & (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P & (\text{scope-ext}) & (\nu x)(P \mid Q) \equiv P \mid (\nu x)Q & \text{if } x \notin P &
\end{array}$$

There are two changes to the reduction system. First, since $x[y].P$ and $x[].P$ are now terms in their own right, the ($\beta\otimes⅋$) and ($\beta\mathbf{1}\bot$) rules are simpler. Second, since we decomposed $(\nu x)(P \mid Q)$ into an independent name restriction and parallel composition, the κ-rules and the relevant γ-rule all decompose as well.

Definition 3.3 (Reduction). Reductions are described by the smallest relation $\Longrightarrow$ on process terms closed under the rules in Figure 1. Relations $\Longrightarrow^{+}$ and $\Longrightarrow^{\star}$ are the transitive, and the reflexive, transitive closures of $\Longrightarrow$, respectively.

$$\begin{array}{lll}
(\leftrightarrow) & (\nu x)(w{\leftrightarrow}x \mid P) & \Longrightarrow\ P\{w/x\} \\
(\beta{\otimes}{\parr}) & (\nu x)(x[y].P \mid x(y).R) & \Longrightarrow\ (\nu x)(\nu y)(P \mid R) \\
(\beta\mathbf{1}\bot) & (\nu x)(x[].P \mid x().Q) & \Longrightarrow\ P \mid Q \\
(\beta{\oplus}\&_1) & (\nu x)(x \triangleleft \mathtt{inl}.P \mid x \triangleright \{\mathtt{inl}: Q; \mathtt{inr}: R\}) & \Longrightarrow\ (\nu x)(P \mid Q) \\
(\beta{\oplus}\&_2) & (\nu x)(x \triangleleft \mathtt{inr}.P \mid x \triangleright \{\mathtt{inl}: Q; \mathtt{inr}: R\}) & \Longrightarrow\ (\nu x)(P \mid R) \\
(\kappa\nu\otimes) & (\nu x)y[z].P & \Longrightarrow\ y[z].(\nu x)P \\
(\kappa\nu\parr) & (\nu x)y(z).P & \Longrightarrow\ y(z).(\nu x)P \\
(\kappa\nu\mathbf{1}) & (\nu x)y[].P & \Longrightarrow\ y[].(\nu x)P \\
(\kappa\nu\bot) & (\nu x)y().P & \Longrightarrow\ y().(\nu x)P \\
(\kappa\nu\oplus_1) & (\nu x)y \triangleleft \mathtt{inl}.P & \Longrightarrow\ y \triangleleft \mathtt{inl}.(\nu x)P \\
(\kappa\nu\oplus_2) & (\nu x)y \triangleleft \mathtt{inr}.P & \Longrightarrow\ y \triangleleft \mathtt{inr}.(\nu x)P \\
(\kappa\nu\&) & (\nu x)y \triangleright \{\mathtt{inl}: P; \mathtt{inr}: Q\} & \Longrightarrow\ y \triangleright \{\mathtt{inl}: (\nu x)P; \mathtt{inr}: (\nu x)Q\} \\
(\kappa\nu\top) & (\nu x)y \triangleright \{\} & \Longrightarrow\ y \triangleright \{\} \\
(\kappa|\otimes) & (y[z].P \mid R) & \Longrightarrow\ y[z].(P \mid R) \\
(\kappa|\parr) & (y(z).P \mid R) & \Longrightarrow\ y(z).(P \mid R) \\
(\kappa|\mathbf{1}) & (y[].P \mid R) & \Longrightarrow\ y[].(P \mid R) \\
(\kappa|\bot) & (y().P \mid R) & \Longrightarrow\ y().(P \mid R) \\
(\kappa|\oplus_1) & (y \triangleleft \mathtt{inl}.P \mid R) & \Longrightarrow\ y \triangleleft \mathtt{inl}.(P \mid R) \\
(\kappa|\oplus_2) & (y \triangleleft \mathtt{inr}.P \mid R) & \Longrightarrow\ y \triangleleft \mathtt{inr}.(P \mid R) \\
(\kappa|\&) & (y \triangleright \{\mathtt{inl}: P; \mathtt{inr}: Q\} \mid R) & \Longrightarrow\ y \triangleright \{\mathtt{inl}: (P \mid R); \mathtt{inr}: (Q \mid R)\} \\
(\kappa|\top) & (y \triangleright \{\} \mid R) & \Longrightarrow\ y \triangleright \{\}
\end{array}$$

$$\frac{P \Longrightarrow P'}{(\nu x)P \Longrightarrow (\nu x)P'}\,\gamma\nu \qquad \frac{P \Longrightarrow P'}{P \mid Q \Longrightarrow P' \mid Q}\,\gamma| \qquad \frac{P \equiv Q \qquad Q \Longrightarrow Q' \qquad Q' \equiv P'}{P \Longrightarrow P'}\,\gamma\equiv$$

Figure 1: Hypersequent Classical Processes, reduction relation.

3.2 Types

We use the same definitions for types and environments for HCP as we used for CP. However, we introduce a new layer on top of sequents: hypersequents. As CP is a one-sided logic, and it uses the left-hand side of the turnstile to write the process, the traditional hypersequent notation can look confusing: "$P \vdash \Gamma_1 \mid \ldots \mid \ \vdash \Gamma_n$" seems to claim that P acts according to protocol Γ_1. What are all the other Γs doing there? Are they typing empty processes? Therefore, we opt to leave out the repeated turnstile, and instead work with the notion of "hyper-environments". However, we will still refer to our system as a hypersequent system. A hyper-environment is either empty, or consist of a series of typing environments, separated by vertical bars. A hyper-environment $\Gamma_1 \mid \ldots \mid \Gamma_n$ types a series of n entangled, but independent processes.

Definition 3.4 (Hyper-environments). $\mathscr{G}, \mathscr{H} ::= \varnothing \mid \mathscr{G} \mid \Gamma$

A hyper-environment is a multiset of environments. While names within environments must be unique, names may be shared between multiple environments in a hyper-environment. We write $\mathscr{G} \mid \mathscr{H}$ to combine two hyper-environments.

Typing judgements in HCP associate processes with hyper-environments. H-Mix composes two processes in parallel, but remembers that they are independent in the sequent. H-Cut and $(\otimes)$ take as their premise a process which consists of at least two independent processes, and connects them, eliminating the vertical bar. Each logical rule has the side condition that $x \notin \mathcal{G}$, which can be read as "you cannot act on one end-point of x if you are also holding its other end-point". This prevents self-locking processes, e.g., $x[].x().0$.

Definition 3.5 (Typing judgements). A typing judgement $P \vdash \Gamma_1 \mid \ldots \mid \Gamma_n$ denotes that the process P consists of n independent, but potentially entangled processes, each of which communicates according to its own protocol Γ_i. Typing judgements can be constructed using the inference rules below.

Structural rules

$$\frac{}{x{\leftrightarrow}y \vdash x : A, y : A^\perp}\text{Ax} \qquad \frac{P \vdash \mathcal{G} \mid \Gamma, x : A \mid \Delta, x : A^\perp}{(\nu x)P \vdash \mathcal{G} \mid \Gamma, \Delta}\text{H-Cut}$$

$$\frac{P \vdash \mathcal{G} \qquad Q \vdash \mathcal{H}}{P \mid Q \vdash \mathcal{G} \mid \mathcal{H}}\text{H-Mix} \qquad \frac{}{0 \vdash \varnothing}\text{H-Halt}$$

Logical rules

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

$$\frac{P \vdash \varnothing}{x[].P \vdash x : \mathbf{1}}\mathbf{1} \qquad \frac{P \vdash \mathcal{G} \mid \Gamma}{x().P \vdash \mathcal{G} \mid \Gamma, x : \bot}(\bot)$$

$$\frac{P \vdash \mathcal{G} \mid \Gamma, x : A}{x \triangleleft \texttt{inl}.P \vdash \mathcal{G} \mid \Gamma, x : A \oplus B}(\oplus_1) \qquad \frac{P \vdash \mathcal{G} \mid \Gamma, x : B}{x \triangleleft \texttt{inr}.P \vdash \mathcal{G} \mid \Gamma, x : A \oplus B}(\oplus_2)$$

$$\frac{P \vdash \mathcal{G} \mid \Gamma, x : A \qquad Q \vdash \mathcal{G} \mid \Gamma, x : B}{x \triangleright \{\texttt{inl} : P; \texttt{inr} : Q\} \vdash \mathcal{G} \mid \Gamma, x : A \,\&\, B}(\&)$$

$$(norulefor\mathbf{0}) \qquad \frac{}{x \triangleright \{\} \vdash \mathcal{G} \mid \Gamma, x : \top}(\top)$$

Furthermore, each logical rule has the side condition that $x \notin \mathcal{G}$.

3.3 Metatheory

HCP enjoys subject reduction, termination, and progress.

Lemma 3.6 (Preservation for $\equiv$). If $P \equiv Q$, then $P \vdash \mathcal{G}$ iff $Q \vdash \mathcal{G}$.

Proof. By induction on the derivation of $P \equiv Q$. □

Theorem 3.7 (Preservation). If $P \vdash \mathcal{G}$ and $P \Longrightarrow Q$, then $Q \vdash \mathcal{G}$.

Proof. By induction on the derivation of $P \Longrightarrow Q$. □

Theorem 3.8 (Termination). If $P \vdash \mathcal{G}$, then there are no infinite $\Longrightarrow$-reduction sequences.

Proof. As Theorem 2.10. □

Theorem 3.9 (Progress). If $P \vdash \mathcal{G}$, then there exists a Q such that $P \Longrightarrow^\star Q$ and Q is not a cut or a mix.

Proof. By induction on the derivation of $P \vdash \mathscr{G}$. If the last rule is a cut, there are three cases: a) if the rule under the cut is the corresponding mix, we eliminate both as in Theorem 2.11; b) if the rule under the cut is an unrelated mix, we rewrite by (scope-ext), and recursively eliminate the cut; c) if the rule under the cut is a logical rule, we apply one of the $\kappa\nu$-rules. If the last rule is mix, there are two cases: a) if the rule under the mix is a cut or another mix, we recursively eliminate it; b) if the rule under the mix is a logical rule, we apply one of the $\kappa|$-rules. □

4 Relation between CP and HCP

In this section, we discuss the relationship between CP and HCP. We will prove two important theorems: every CP process is an HCP process; and HCP supports the same protocols as CP. We define a translation from terms in CP to terms in HCP which breaks down the term constructs in CP into their more atomic constructs in HCP.

Definition 4.1.

$$
\begin{array}{lll@{\qquad}lll}
[\![x{\leftrightarrow}y]\!] & := & x{\leftrightarrow}y & [\![(\nu x)(P \mid Q)]\!] & := & (\nu x)([\![P]\!] \mid [\![Q]\!]) \\
[\![x[y].(P \mid Q)]\!] & := & x[y].([\![P]\!] \mid [\![Q]\!]) & [\![x(y).P]\!] & := & x(y).[\![P]\!] \\
[\![x[].0]\!] & := & x[].0 & [\![x().P]\!] & := & x().[\![P]\!] \\
[\![x \triangleleft \texttt{inl}.P]\!] & := & x \triangleleft \texttt{inl}.[\![P]\!] & [\![x \triangleleft \texttt{inr}.P]\!] & := & x \triangleleft \texttt{inr}.[\![P]\!] \\
[\![x \triangleright \{\texttt{inl}: P; \texttt{inr}: Q\}]\!] & := & x \triangleright \{\texttt{inl}: [\![P]\!]; \texttt{inr}: [\![Q]\!]\} & [\![x \triangleright \{\}]\!] & := & x \triangleright \{\}
\end{array}
$$

We will use this relation in the first proof, and its analogue for derivations in the second.

4.1 Every CP process is an HCP process

First, we prove that each CP process can be translated by this trivial translation to an HCP process, and that this translation respects structural congruence and reduction. Reductions from CP can be trivially translated to reductions in HCP. However, as we took apart each κ-rule into two separate rules, the reduction relation of HCP is a slight refinement over that of CP.

Theorem 4.2. If $P \vdash \Gamma$ in CP, then $[\![P]\!] \vdash \Gamma$ in HCP.

Proof. By induction on the derivation of $P \vdash \Gamma$. We show the interesting cases:

- Case Cut. We rewrite as follows:

$$
\dfrac{P \vdash \Gamma, x : A \qquad Q \vdash \Delta, x : A^{\perp}}{(\nu x)(P \mid Q) \vdash \Gamma, \Delta}\ \text{Cut}
\quad \Rightarrow \quad
\dfrac{\dfrac{[\![P]\!] \vdash \Gamma, x : A \qquad [\![Q]\!] \vdash \Delta, x : A^{\perp}}{[\![P]\!] \mid [\![Q]\!] \vdash \Gamma, x : A \mid \Delta, x : A^{\perp}}\ \text{H-Mix}}{(\nu x)([\![P]\!] \mid [\![Q]\!]) \vdash \Gamma, \Delta}\ \text{H-Cut}
$$

- Case $(\otimes)$. We rewrite as follows:

$$
\dfrac{P \vdash \Gamma, y : A \qquad Q \vdash \Delta, x : B}{x[y].(P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B}\ \otimes
\quad \Rightarrow \quad
\dfrac{\dfrac{[\![P]\!] \vdash \Gamma, y : A \qquad [\![Q]\!] \vdash \Delta, x : B}{[\![P]\!] \mid [\![Q]\!] \vdash \Gamma, y : A \mid \Delta, x : B}\ \text{H-Mix}}{x[y].([\![P]\!] \mid [\![Q]\!]) \vdash \Gamma, \Delta, x : A \otimes B}\ \otimes
$$

- Case $(\mathbf{1})$. We rewrite as follows:

$$
\dfrac{}{x[].0 \vdash x : \mathbf{1}}\ \mathbf{1}
\quad \Rightarrow \quad
\dfrac{\dfrac{}{0 \vdash \varnothing}\ \text{H-Halt}}{x[].0 \vdash x : \mathbf{1}}\ \mathbf{1}
$$

□

Theorem 4.3. $P \equiv Q$ in CP iff $[\![P]\!] \equiv [\![Q]\!]$ in HCP.

Proof. By induction on the derivation of $P \equiv Q$. □

Theorem 4.4. If $P \Longrightarrow Q$ in CP, then $[\![P]\!] \Longrightarrow^{+} [\![Q]\!]$ in HCP.

Proof. By induction on the the derivation of $P \Longrightarrow Q$. □

Theorem 4.5. If $[\![P]\!] \Longrightarrow R$ in HCP, then there is a Q such that $P \Longrightarrow Q$ in CP and $R \Longrightarrow^{+} [\![Q]\!]$ in HCP.

Proof. By induction on the derivation of $[\![P]\!] \Longrightarrow R$. The cases for $(\leftrightarrow)$ and the β-rules are trivial. For the $\kappa\nu$-rules, we rewrite by the appropriate $\kappa|$ rule, and vice versa. □

4.2 HCP supports the same communication protocols as CP

HCP exhibits some behaviours which are impossible to directly translate back into CP. For instance, in the process below, the choice sent on x will affect the choice between P and P', even though neither has access to the channel x.

$$\dfrac{\dfrac{P \vdash \Gamma \qquad Q \vdash \Delta, x : A}{P \mid Q \vdash \Gamma \mid \Delta, x : A}\text{H-Mix} \qquad \dfrac{P' \vdash \Gamma \qquad Q' \vdash \Delta, x : B}{P' \mid Q' \vdash \Gamma \mid \Delta, x : B}\text{H-Mix}}{x \triangleright \{\mathtt{inl} : P \mid Q; \mathtt{inr} : P' \mid Q'\} \vdash \Gamma \mid \Delta, x : A \mathbin{\&} B}\&$$

Instead, we will prove that HCP supports the same communication protocols as CP. This is the same as saying that it inhabits the same session types, or that the associated logical systems derive the same theorems. We show this by proving that we can internalise the hyper-environments as formulas in the logic. This is a standard method for proving the soundness of a hypersequent calculus.

We start off by defining a relation on derivations of HCP, which we call “disentanglement”. This relation allows us to move applications of H-Mix downwards in the proof tree. We can use this relation to rewrite any derivation to a form in which all mixes are either attached to their respective cuts or tensors, or at the top-level.

Definition 4.6. Disentanglement is described by the smallest relation $\rightsquigarrow$ on proof derivations closed under the rules in Figure 2, plus the associativity and commutativity of mixes. The relation $\rightsquigarrow^{\star}$ is the reflexive, transitive closure of $\rightsquigarrow$.

We named this relation “disentanglement” to reflect the intuition that proof in HCP represent multiple entangled CP proofs, which we can disentangle. However, this is not entirely accurate. The proof structure of HCP is slightly richer than that of CP. This can be seen in the example process above. In this case, when disentangling, we are forced to forget some of the proof structure. This can be seen in the last rule in Figure 2. Nonetheless, the relation is type preserving, and so it suffices for showing that HCP supports the same communication protocols as CP.

Disentanglement is terminating, and confluent up to the associativity and commutativity of mixes.

$$\dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2}\,\text{H-Mix} \quad \begin{matrix}\rho_3\\ \vdots\\ \vdash \mathscr{G}_3 \mid A^{\perp},\Delta\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \mathscr{G}_3 \mid \Gamma,\Delta}\,\text{H-Cut} \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_3\\ \vdots\\ \vdash \mathscr{G}_3 \mid A^{\perp},\Delta\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_3 \mid \Gamma,\Delta}\,\text{H-Cut} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \mathscr{G}_3 \mid \Gamma,\Delta}\,\text{H-Mix}$$

$$\dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A,\Gamma}\,\text{H-Mix} \quad \begin{matrix}\rho_3\\ \vdots\\ \vdash \mathscr{G}_3 \mid B,\Delta\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \mathscr{G}_3 \mid A \otimes B,\Gamma,\Delta}\,\otimes \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_3\\ \vdots\\ \vdash \mathscr{G}_3 \mid B,\Delta\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \otimes B,\Gamma,\Delta}\,\otimes \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \mathscr{G}_3 \mid A \otimes B,\Gamma,\Delta}\,\text{H-Mix}$$

$$\dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,B,\Gamma\end{matrix} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A,B,\Gamma}\,\text{H-Mix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \mathbin{⅋} B,\Gamma}\,⅋ \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,B,\Gamma\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A,B,\Gamma}\,⅋ \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \mathbin{⅋} B,\Gamma}\,\text{H-Mix}$$

$$\dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid \Gamma\end{matrix} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \Gamma}\,\text{H-Mix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \bot,\Gamma}\,\bot \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid \Gamma\end{matrix}}{\vdash \mathscr{G}_1 \mid \bot,\Gamma}\,\bot \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid \Gamma}\,\text{H-Mix}$$

$$\dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A,\Gamma}\,\text{H-Mix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \oplus B,\Gamma}\,\oplus_1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix}}{\vdash \mathscr{G}_1 \mid A \oplus B,\Gamma}\,\oplus_1 \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \oplus B,\Gamma}\,\text{H-Mix}$$

$$\dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A,\Gamma}\,\text{H-Mix} \quad \dfrac{\begin{matrix}\rho_3\\ \vdots\\ \vdash \mathscr{G}_1 \mid B,\Gamma\end{matrix} \quad \begin{matrix}\rho_4\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid B,\Gamma}\,\text{H-Mix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \mathbin{\&} B,\Gamma}\,\& \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{matrix}\rho_1\\ \vdots\\ \vdash \mathscr{G}_1 \mid A,\Gamma\end{matrix} \quad \begin{matrix}\rho_3\\ \vdots\\ \vdash \mathscr{G}_1 \mid B,\Gamma\end{matrix}}{\vdash \mathscr{G}_1 \mid A \mathbin{\&} B,\Gamma}\,\& \quad \begin{matrix}\rho_2\\ \vdots\\ \vdash \mathscr{G}_2\end{matrix}}{\vdash \mathscr{G}_1 \mid \mathscr{G}_2 \mid A \mathbin{\&} B,\Gamma}\,\text{H-Mix}$$

Figure 2: The disentanglement relation for HCP.

Lemma 4.7 (Disentangle). If there exists a derivation ρ of $\vdash \Gamma_1 \mid \ldots \mid \Gamma_n$ in HCP, then there exist derivations $\rho_1, \ldots, \rho_n$ of $\vdash \Gamma_1, \ldots, \vdash \Gamma_n$ in CP such that

$$\rho \rightsquigarrow^\star \frac{[\![\rho_1]\!] \qquad \ldots \qquad [\![\rho_n]\!]}{\vdash \Gamma_1 \mid \ldots \mid \Gamma_n}\text{ H-Mix}$$

Proof. We repeatedly apply the $\rightsquigarrow$-rules to the derivation ρ to move the mixes downwards. There are three cases: a) if a mix gets stuck under a cut, it forms a CP cut; b) if a mix gets stuck under a $(\otimes)$, it forms a CP $(\otimes)$; c) otherwise, it moves all the way to the top. All applications of $(\mathbf{1})$ are followed by an application of H-Halt, forming a CP $(\mathbf{1})$. □

An environment can be internalised as a type by collapsing it as a series of pars.

Definition 4.8.

$$\begin{aligned} \invamp(\cdot) &= \bot \\ \invamp(x_1 : A_1, \ldots, x_n : A_n) &= A_1 \invamp \cdots \invamp A_n \quad \text{if } n \geq 1 \end{aligned}$$

Lemma 4.9. If $\vdash \Gamma$ in CP, then $\vdash \invamp\Gamma$ in CP.

Proof. By repeated application of $(\invamp)$. □

Furthermore, a hyper-environment can be internalised as a type by collapsing it as a series of tensors, where each constituent environment is internalised using $\invamp$. The empty hyper-environment $\varnothing$ is internalised as the unit of tensor.

Definition 4.10.

$$\begin{aligned} \otimes(\varnothing) &= \mathbf{1} \\ \otimes(\Gamma_1 \mid \ldots \mid \Gamma_n) &= \invamp\Gamma_1 \otimes \ldots \otimes \invamp\Gamma_n \quad \text{if } n \geq 1 \end{aligned}$$

Theorem 4.11. If $\vdash \mathcal{G}$ in HCP, then $\vdash \otimes\mathcal{G}$ in CP.

Proof. By case analysis on the structure of the hyper-environment $\mathcal{G}$. If $\mathcal{G} = \varnothing$, we apply $(\mathbf{1})$. If $\mathcal{G} = \Gamma_1 \mid \ldots \mid \Gamma_n$, we apply Lemma 4.7 to obtain proofs of $\vdash \Gamma_1, \ldots, \vdash \Gamma_n$ in CP, then we apply Lemma 4.9 to each of those proofs to obtain proofs of $\vdash \invamp\Gamma_1, \ldots, \vdash \invamp\Gamma_n$, and join them using $(\otimes)$ to obtain a single proof of $\vdash \otimes\mathcal{G}$ in CP. □

5 Related Work

Since its inception, linear logic has been described as the logic of concurrency (Girard, 1987). Correspondences between the proof theory of linear logic and variants of the π-calculus emerged soon afterwards (Abramsky, 1994; Bellin and Scott, 1994), by interpreting linear propositions as types for channels. Linearity inspired also the seminal theories of linear types for the π-calculus (Kobayashi et al., 1999) and session types (Honda et al., 1998). Even though the two theories do not have a direct correspondence with linear logic, the link is still strong enough that session types can be encoded into linear types (Dardha et al., 2017).

It took more than ten years for a formal correspondence between linear logic and (a variant of) session types to emerge, with the seminal paper by Caires and Pfenning (2010). This inspired the development of Classical Processes by Wadler (2012).

The idea of using hypersequents to capture parallelism in linear logic judgements is not novel: Carbone et al. (2018) extended the multiplicative-additive fragment of intuitionistic linear logic with hypersequents to type global descriptions of process communications known as choreographies. This work is distinct from our approach in that HCP is based on classical linear logic and manipulates hypersequents differently: in Carbone et al. (2018), hypersequents can be formed only when sequents share resources (cf., H-Mix), and resource sharing is then tracked using an additional connection modality (which is not present in HCP).

References

Abramsky, S. (1994). Proofs as processes. Theoretical Computer Science, 135(1):5–9.

Avron, A. (1991). Hypersequents, logical consequence and intermediate logics for concurrency. Annals of Mathematics and Artificial Intelligence, 4(3-4):225–248.

Bellin, G. and Scott, P. (1994). On the π-calculus and linear logic. Theoretical Computer Science, 135(1):11–65.

Boreale, M. (1998). On the expressiveness of internal mobility in name-passing calculi. Theoretical Computer Science, 195(2):205–226.

Caires, L. and Pfenning, F. (2010). Session Types as Intuitionistic Linear Propositions, pages 222–236. Springer Berlin Heidelberg, Berlin, Heidelberg.

Carbone, M., Montesi, F., and Schürmann, C. (2018). Choreographies, logically. Distributed Computing, 31(1):51–67.

Dardha, O., Giachino, E., and Sangiorgi, D. (2017). Session types revisited. Inf. Comput., 256:253–286.

Girard, J.-Y. (1987). Linear logic. Theoretical Computer Science, 50(1):1–101.

Honda, K., Vasconcelos, V. T., and Kubo, M. (1998). Language primitives and type discipline for structured communication-based programming. In Programming Languages and Systems - ESOP'98, 7th European Symposium on Programming, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings, pages 122–138.

Kobayashi, N., Pierce, B. C., and Turner, D. N. (1999). Linearity and the pi-calculus. ACM Transactions on Programming Languages and Systems, 21(5):914–947.

Milner, R., Parrow, J., and Walker, D. (1992a). A calculus of mobile processes, I. Information and Computation, 100(1):1–40.

Milner, R., Parrow, J., and Walker, D. (1992b). A calculus of mobile processes, II. Information and Computation, 100(1):41–77.

Sangiorgi, D. (1996). π-calculus, internal mobility, and agent-passing calculi. Theoretical Computer Science, 167(1-2):235–274.

Wadler, P. (2012). Propositions as sessions. In Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming, ICFP '12, pages 273–286, New York, NY, USA. ACM.