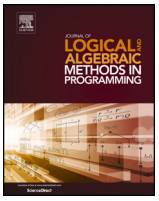
J. Log. Algebraic Methods Program. 141 (2024) 100986

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

journal homepage: www.elsevier.com/locate/jlamp



A logical account of subtyping for session types

Ross Horne [a,*], Luca Padovani [b]

[a] Computer and Information Sciences, University of Strathclyde, Glasgow, United Kingdom
[b] University of Camerino, Scuola di Scienze e Tecnologie Polo di Informatica, Camerino, Italy

ARTICLE INFO

Keywords:
Session types
Subtyping
Linear logic
Fixed points
Termination

ABSTRACT

We study iso-recursive and equi-recursive subtyping for session types in a logical setting, where session types are propositions of multiplicative/additive linear logic extended with least and greatest fixed points. Both subtyping relations admit a simple characterization that can be roughly spelled out as the following lapalissade: every session type is larger than the smallest session type and smaller than the largest session type. We observe that, because of the logical setting in which they arise, these subtyping relations preserve termination in addition to the usual safety properties of sessions.

1. Introduction

Session types [1–3] are descriptions of communication protocols supported by an elegant correspondence with linear logic [4–7] that provides session type systems with solid logical foundations. As an example, below is the definition of a session type describing the protocol implemented by a mathematical server (in the examples of this section, $\&$ and $\oplus$ are n-ary operators denoting external and internal labelled choices, respectively):

$$B = \&\{\mathsf{end} : \bot, \mathsf{add} : \mathsf{Num}^{\bot} \parr \mathsf{Num}^{\bot} \parr \mathsf{Num} \otimes B\}$$

According to the session type B, the server first waits for a label – either end or add – that identifies the operation requested by the client. If the label is end, the client has no more requests and the server terminates. If the label is add, the server waits for two numbers, sends their sum back to the client and then makes itself available again offering the same protocol B. In this example, we write $\mathsf{Num}^{\bot}$ for the type of numbers being consumed and Num for the type of numbers being produced. A client of this server could implement a communication protocol described by the following session type:

$$A = \oplus\{\mathsf{add} : \mathsf{Num} \otimes \mathsf{Num} \otimes \mathsf{Num}^{\bot} \parr \oplus\{\mathsf{end} : \mathbf{1}\}\}$$

This client sends the label add followed by two numbers, it receives the result and then terminates the interaction with the server by sending the label end. When we connect two processes through a session, we expect their interaction to be flawless. In many session type systems, this is guaranteed by making sure that the session type describing the behaviour of one process is the *dual* of the session type describing the behaviour of its peer. *Duality*, often denoted by $\cdot^{\bot}$, is the operator on session types that inverts the *direction* of messages. In the above example it is clear that A is *not* the dual of B nor is B the dual of A. Nonetheless, we would like such client and such server to be declared compatible, since the client is exercising only a subset of the capabilities of the server. To

* Corresponding author.
E-mail address: ross.horne@strath.ac.uk (R. Horne).

https://doi.org/10.1016/j.jlamp.2024.100986
Received 18 November 2023; Received in revised form 23 May 2024; Accepted 23 May 2024

Available online 28 May 2024

express this compatibility we have to resort to a more complex relation between A and B, either by observing that B (the behaviour of the server) is a *more accommodating* version of $A^\perp$ or by observing that A (the behaviour of the client) is a *less demanding* version of $B^\perp$. We make these relations by means of a *subtyping relation* $\leqslant$ for session types. Subtyping enhances the applicability of type systems by means of the well-known substitution principle: an entity of type C can be used where an entity of type D is expected if C is a subtype of D. After the initial work of Gay and Hole [8] many subtyping relations for session types have been studied [9–13]. Such subtyping relations differ widely in the way they are defined and/or in the properties they preserve, but they all share the fact that subtyping is essentially defined by the branching structure of session types given by labels. To illustrate this aspect, let us consider again the session types A and B defined above. We have

$$B \leqslant \&\{\mathsf{add} : \mathsf{Num}^\perp \parr \mathsf{Num}^\perp \parr \mathsf{Num} \otimes \&\{\mathsf{end} : \bot\}\} = A^\perp \tag{1}$$

meaning that a server behaving as B can be safely used where a server behaving as $A^\perp$ is expected. Dually, we also have

$$A \leqslant \oplus\{\mathsf{end} : \mathbf{1}, \mathsf{add} : \mathsf{Num} \otimes \mathsf{Num} \otimes \mathsf{Num}^\perp \parr B^\perp\} = B^\perp \tag{2}$$

meaning that a client behaving as A can be safely used where a client behaving as $B^\perp$ is expected. Note how subtyping is crucially determined by the sets of labels that can be received/sent when comparing two related types. In (1), the server of type B is willing to accept any label from the set $\{\mathsf{end}, \mathsf{add}\}$, which is a *superset* of $\{\mathsf{add}\}$ that we have in $A^\perp$. In (2), the client is (initially) sending a label from the set $\{\mathsf{add}\}$, which is a subset of $\{\mathsf{end}, \mathsf{add}\}$ that we have in $B^\perp$. This co/contra variance of labels in session types is a key distinguishing feature of all known notions of subtyping for session types.[1]

In this work we study the notion of subtyping for session types in a setting where session types are propositions of $\mu\mathsf{MALL}^\infty$ [15, 16], the infinitary proof theory of multiplicative additive linear logic extended with least and greatest fixed points. Our investigation has two objectives. First, to understand whether and how it is possible to capture the well-known co/contra variance of behaviours when the connectives used to describe branching session types ($\&$ and $\oplus$ of linear logic) have fixed (binary) rather than variable arity. Second, to understand whether there are critical aspects of subtyping that become relevant when typing derivations are meant to be logically sound.

At the core of our proposal is the observation that, when session types (hence process behaviours) are represented by linear logic propositions [4–6], there is no process that behaves according to the type $\mathbf{0}$ and virtually every process can be declared to behave according to the type $\top$. If we think of a session type as the set of processes that behave according to that type, this means that the additive constants $\mathbf{0}$ and $\top$ may serve well as the least and greatest elements of a session subtyping relation. Somewhat surprisingly, the subtyping relation arising by these properties of $\mathbf{0}$ and $\top$ allows us to express essentially the same subtyping relations arising from the usual co/contra variance of labels. For example, following our proposal the session type of the client, previously denoted A, would instead be written as

$$C = \oplus\{\mathsf{end} : \mathbf{0}, \mathsf{add} : \mathsf{Num} \otimes \mathsf{Num} \otimes \mathsf{Num}^\perp \parr \oplus\{\mathsf{end} : \mathbf{1}, \mathsf{add} : \mathbf{0}\}\}$$

using which we can derive both

$$B \leqslant \&\{\mathsf{end} : \top, \mathsf{add} : \mathsf{Num}^\perp \parr \mathsf{Num}^\perp \parr \mathsf{Num} \otimes \&\{\mathsf{end} : \bot, \mathsf{add} : \top\}\} = C^\perp$$

as well as

$$C \leqslant B^\perp$$

without comparing sets of labels, but just using the fact that $\mathbf{0}$ is the least session type and $\top$ the greatest one. Basically, instead of *omitting those labels* that correspond to impossible continuations (*cf.* the missing end and add in A), we use the uninhabited session type $\mathbf{0}$ or its dual $\top$ *as impossible continuations* (*cf.* C). It could be argued that the difference between the two approaches is mostly cosmetic. Indeed, it is easy to devise (de)sugaring functions to rewrite session types from one syntax to the other. However, the novel approach we propose allows us to recast the well-known subtyping relation for session types in a logical setting. A first consequence of this achievement is that the soundness of the type system *with subtyping* does not require an *ad hoc* proof, but follows from the soundness of the type system *without subtyping*. In addition, we find out that the subtyping relations we propose preserve not only the usual *safety properties* – communication safety, protocol fidelity and deadlock freedom – but also *termination*, which is a *liveness property*.

Structure of the paper. In Section 2 we introduce $\mu\mathsf{CP}^\infty$, our reference calculus of sessions closely related to $\mu\mathsf{CP}$ [6] and CP [4]. In Section 3 we present the type language and the typing rules for $\mu\mathsf{CP}^\infty$ along with the termination property that $\mu\mathsf{CP}^\infty$ enjoys as a consequence of its relationship with $\mu\mathsf{MALL}^\infty$ [15,16]. Sections 5 and 6 are devoted to the study and comparison of two logical subtyping relations that differ in the treatment of fixed point operators: in Section 5 we study *iso-recursive subtyping*, which is quite restrictive insofar fixed point operators are concerned but enjoys the expected safe substitution principle; in Section 6 we study *equi-recursive subtyping*, which is coarser than iso-recursive subtyping but whose soundness proof requires the introduction of explicit

[1] Gay and Hole [8] and other authors [9,10,12] define subtyping for session types in such a way that the *opposite* relations of eqs. (1) and (2) hold. Both viewpoints are viable depending on whether session types are considered to be types of *channels* or types of *processes*. Here we take the latter stance, referring to Gay [14] for a comparison of the two approaches.

Table 1
Syntax of $\mu\mathsf{CP}^{\infty}$.

$P, Q ::=$		**Process**	$\mid (x)(P \mid Q)$	composition
	$\mathsf{A}\langle\overline{x}\rangle$	invocation	$\mid \mathsf{fail}\ x$	failure
$\mid$	$x().P$	signal input	$\mid x[]$	signal output
$\mid$	$x(y).P$	channel input	$\mid x[y](P \mid Q)$	channel output
$\mid$	$\mathsf{case}\ x\{P, Q\}$	choice input	$\mid x[\mathsf{in}_i].P$	choice output ($i \in \{0, 1\}$)
$\mid$	$\mathsf{corec}\ x.P$	corecursion	$\mid \mathsf{rec}\ x.P$	recursion

Table 2
Structural pre-congruence and reduction semantics of $\mu\mathsf{CP}^{\infty}$.

$$[\text{S-PAR-COMM}] \quad (x)(P \mid Q) \preccurlyeq (x)(Q \mid P)$$

$$[\text{S-PAR-ASSOC}] \quad (x)(P \mid (y)(Q \mid R)) \preccurlyeq (y)((x)(P \mid Q) \mid R) \quad x \notin \mathsf{fn}(R), y \notin \mathsf{fn}(P)$$

$$[\text{S-CALL}] \quad \mathsf{A}\langle\overline{x}\rangle \preccurlyeq P \quad \mathsf{A}(\overline{x}) \triangleq P$$

$$[\text{R-COMM}] \quad \frac{}{(x)(x[y](P \mid Q) \mid x(y).R) \to (y)(P \mid (x)(Q \mid R))}\ x \notin \mathsf{fn}(P)$$

$$[\text{R-CASE}] \quad \frac{}{(x)(x[\mathsf{in}_i].P \mid \mathsf{case}\ x\{Q_0, Q_1\}) \to (x)(P \mid Q_i)}\ i \in \{0, 1\}$$

$$[\text{R-CLOSE}] \quad \frac{}{(x)(x[] \mid x().P) \to P}\ x \notin \mathsf{fn}(P)$$

$$[\text{R-REC}] \quad \frac{}{(x)(\mathsf{rec}\ x.P \mid \mathsf{corec}\ x.Q) \to (x)(P \mid Q)}$$

$$[\text{R-PAR}] \quad \frac{P \to Q}{(x)(P \mid R) \to (x)(Q \mid R)}$$

$$[\text{R-STRUCT}] \quad \frac{P \preccurlyeq P' \qquad P' \to Q' \qquad Q' \preccurlyeq Q}{P \to Q}$$

coercions. We wrap up in Section 7 with a more detailed discussion of related and future work. The appendix contains some lengthy proofs that do not fit well into the main body of the paper.

Origin of the material. This is a restructured and extended version of a paper that appears in the proceedings of the PLACES'23 workshop [17]. The present version includes new examples, the treatment of iso-recursive subtyping (Section 5) as well as detailed proofs, notably in Section 4, which are not present in the workshop proceedings.

2. Syntax and semantics of $\mu\mathsf{CP}^{\infty}$

The syntax of $\mu\mathsf{CP}^{\infty}$ is shown in Table 1 and makes use of a set of *process names* $\mathsf{A}, \mathsf{B}, \ldots$ and of an infinite set of *channels* x, y, z and so on. The calculus includes standard forms representing communication actions: $\mathsf{fail}\ x$ models a process failing on x; $x().P$ and $x[]$ model the input/output of a termination signal on x; $\mathsf{case}\ x\{P, Q\}$ and $x[\mathsf{in}_i].P$ model the input/output of a label in_i on x; $x(y).P$ and $x[y](P \mid Q)$ model the input/output of a channel y on x. Process $x[y](P \mid Q)$ outputs a *new* channel y which binds occurrence of y in P, but, importantly, y cannot appear in Q, which has the effect of restricting the communication topology.

In addition, $\mu\mathsf{CP}^{\infty}$ has prefixes $\mathsf{corec}\ x$ and $\mathsf{rec}\ x$ for modelling (co)recursive processes whose behaviour is described by a greatest/least fixed point respectively. These forms should not be confused with recursive binders; they are better seen as checkpoints in the proof that, respectively, produce and consume energy. The energy must be in balance to sustain computations along infinite branches of the proof. The syntax originates from prior work on Curry-Howard correspondences for finitary proof systems for linear logic with fixed points [6].[2]

A process of the form $(x)(P \mid Q)$ models a session x connecting two parallel processes P and Q and the form $\mathsf{A}\langle\overline{x}\rangle$ models the invocation of the process named A with parameters $\overline{x}$. For each process name A we assume that there is a unique global definition of the form $\mathsf{A}(\overline{x}) \triangleq P$ that gives its meaning. Hereafter $\overline{x}$ denotes a possibly empty sequence of channels. The notions of free and bound channels are defined in the expected way. We identify processes up to renaming of bound channels and we write $\mathsf{fn}(P)$ for the set of free channels of P.

The operational semantics of $\mu\mathsf{CP}^{\infty}$ is shown in Table 2 and consists of a structural precongruence relation $\preccurlyeq$ and a reduction relation $\to$, both of which are fairly standard. We write $P \to$ if $P \to Q$ for some Q and we say that P is *stuck*, notation $P \not\to$, if not $P \to$.

[2] The difference with the work of Lindley and Morris [6] is that we have only one name in $\mathsf{corec}\ x$ while the construct in prior work features two names. This is due to differences between the structure of proofs in finitary and infinitary proof systems. In finitary proof systems, the rule for co-recursion features two premises, while only one premise is required in the rule for co-recursion in an infinitary proof system.

Example 1. We can model client and server described in Section 1 as the processes below.

$$\mathsf{Client}(x) \triangleq \mathsf{rec}\ x.x[\mathsf{in}_1].\mathsf{rec}\ x.x[\mathsf{in}_0].x[]$$
$$\mathsf{Server}(x,z) \triangleq \mathsf{corec}\ x.\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x,z\rangle\}$$

For simplicity, we only focus on the overall structure of the processes rather than on the actual mathematical operations they perform, so we omit any exchange of concrete data from this model. ⌟

We provide here a definition of the *termination* property ensured by our type system (under specific conditions that we will clarify in subsequent sections).

Definition 1 *(Terminating process).* A *reduction sequence* of a process P is a (finite or infinite) sequence $(P_0, P_1, \dots)$ of processes such that $P_0 = P$ and $P_i \to P_{i+1}$ whenever $i+1$ is a valid index of the sequence. We say that P is *terminating* if every reduction sequence of P is finite and the last element in it is (structurally precongurent to) a process of the form $x[]$ for some x.

Note that this notion of termination entails both *deadlock and lock freedom* [18,19]. More precisely, if we call "pending action" every prefix of a process that describes a communication (namely the prefixes $x()$, $x[y]$, $x(y)$, $x[\mathsf{in}_i]$, $\mathsf{case}\ x$, $\mathsf{rec}\ x$ and $\mathsf{corec}\ x$), then for a terminating process P we have:

- if $P \Rightarrow Q \not\to$, then Q must be of the form $x[]$, hence Q contains no pending actions (deadlock freedom);
- if $P \Rightarrow Q$, then $Q \Rightarrow x[]$ for some x, hence each pending action in Q can be (and is) eventually performed (lock freedom).

Remark 1. This calculus, in the tradition of Caires-Pfenning-Wadler, is not a normal session calculus outside their paradigm. For example, consider the process where $A\langle x\rangle \triangleq \mathsf{case}\ x\{x[\mathsf{in}_0].A\langle x\rangle, x[]\}$.

$$(z)(z[x](x[\mathsf{in}_0].A\langle x\rangle \mid A\langle x\rangle) \mid z(x).x[\mathsf{in}_1].\mathsf{case}\ x\{x().x().y[], \mathsf{fail}\ x\})$$

Experts familiar with session calculi, who match up communications in the obvious way, may reasonably assume that the above process has one infinite execution path, and infinitely many execution paths that terminate with process $y[]$. Yet the process, which is syntactically valid,[3] blocks after one transition step. This is because the rules of $\mu\mathsf{CP}^\infty$ do not allow the two processes created under a par to be rearranged such that they communicate with each other. In particular, the restrictions on the rule [S-PAR-ASSOC], combined with the way that reduction rules insist on the channel involved in a communication to tightly bound around the interacting processes, prevent all further communications. This has the consequences of eliminating races, such as the race in the process above.

Perhaps it would be better, from a broader process calculus perspective, to use a different symbol for parallel composition under an output prefix, in order to indicate the separation enforced by the rules. However, we do not change the established syntax in this paper, so that the syntax may be easily matched with papers in the literature [5,4]. ⌟

3. Type system

The type language for $\mu\mathsf{CP}^\infty$ consists of the propositions of $\mu\mathsf{MALL}^\infty$ [15,16,20], namely the multiplicative/additive fragment of linear logic extended with least and greatest fixed points. We start from the definition of *pre-types*, which are linear logic propositions built using type variables taken from an infinite set and ranged over by X and Y.

$$A, B ::= X \mid \bot \mid \mathbf{1} \mid \top \mid \mathbf{0} \mid A \parr B \mid A \otimes B \mid A \,\&\, B \mid A \oplus B \mid \nu X.A \mid \mu X.A$$

The usual notions of free and bound type variables apply. A *type* is a closed pre-type. We write $A^\perp$ for the *dual* of A, which is defined in the expected way with the proviso that $X^\perp = X$. This way of dualizing type variables is not problematic since we will always apply $\cdot^\perp$ to types, which contain no free type variables. As usual, we write $A\{B/X\}$ for the (pre-)type obtained by replacing every X occurring free in the pre-type A with the type B. Hereafter we let κ range over the constants $\mathbf{0}$, $\mathbf{1}$, $\bot$ and $\top$, we let $\star$ range over the connectives $\&$, $\oplus$, $\parr$ and $\otimes$ and σ range over the binders μ and ν. Also, we say that any type of the form $\sigma X.A$ is a σ-type and we adopt the standard convention by which the scope of σX extends as much as possible to the right.

We write $\preceq$ for the *sub-formula* relation on types. To be precise, $\preceq$ is the least preorder on types such that $A \preceq \sigma X.A$ and $A_i \preceq A_1 \star A_2$. For example, consider $A \stackrel{\mathsf{def}}{=} \mu X.\nu Y.(1 \oplus X)$ and its unfolding $A' \stackrel{\mathsf{def}}{=} \nu Y.(1 \oplus A)$. We have $A \preceq 1 \oplus A \preceq A'$, hence A is a sub-formula of A'. Given a set $\mathcal{T}$ of types we write $\min \mathcal{T}$ for the $\preceq$-minimum type in $\mathcal{T}$ when it is defined.

Typing judgements have the form $P \vdash \Gamma$ where P is a process and Γ is a typing context, namely a finite map from channels to types. We can read this judgement roughly as the fact that P behaves as described by the types in the range of Γ with respect to the channels in the domain of Γ. We write $\mathsf{dom}(\Gamma)$ for the domain of Γ, we write $x : A$ for the typing context with domain $\{x\}$ that maps x to A, we write Γ, Δ for the union of Γ and Δ when $\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$.

[3] It fails only the condition that channels delegated in the send operation should not appear in the right parallel component. That condition is particular to this family of typed session calculi and is not standard for the π-calculus.

Table 3
Typing rules for $\mu\mathsf{CP}^{\infty}$.

$$[\textsc{call}]\ \frac{P \vdash \overline{x : A}}{\mathsf{A}\langle\overline{x}\rangle \vdash \overline{x : A}}\ \mathsf{A}(\overline{x}) \triangleq P \qquad [\textsc{cut}]\ \frac{P \vdash \Gamma, x : A \qquad Q \vdash \Delta, x : A^{\perp}}{(x)(P \mid Q) \vdash \Gamma, \Delta}$$

$$[\top]\ \frac{}{P \vdash \Gamma, x : \top}\ \mathsf{fn}(P) \subseteq \mathsf{dom}(\Gamma) \cup \{x\} \qquad \textbf{(no rule for 0)} \qquad [\bot]\ \frac{P \vdash \Gamma}{x().P \vdash \Gamma, x : \bot}$$

$$[\mathbf{1}]\ \frac{}{x[] \vdash x : \mathbf{1}} \qquad [\parr]\ \frac{P \vdash \Gamma, y : A, x : B}{x(y).P \vdash \Gamma, x : A \parr B} \qquad [\otimes]\ \frac{P \vdash \Gamma, y : A \qquad Q \vdash \Delta, x : B}{x[y](P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B}$$

$$[\&]\ \frac{P \vdash \Gamma, x : A \qquad Q \vdash \Gamma, x : B}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : A \,\&\, B} \qquad [\oplus]\ \frac{P \vdash \Gamma, x : A_i}{x[\mathsf{in}_i].P \vdash \Gamma, x : A_0 \oplus A_1}\ i \in \{0,1\}$$

$$[\nu]\ \frac{P \vdash \Gamma, x : A\{\nu X.A/X\}}{\mathsf{corec}\ x.P \vdash \Gamma, x : \nu X.A} \qquad [\mu]\ \frac{P \vdash \Gamma, x : A\{\mu X.A/X\}}{\mathsf{rec}\ x.P \vdash \Gamma, x : \mu X.A}$$

Table 4
Proof rules of $\mu\mathsf{MALL}^{\infty}$.

$$[\textsc{cut}]\ \frac{\vdash \Gamma, F \qquad \vdash \Delta, F^{\perp}}{\vdash \Gamma, \Delta} \qquad [\top]\ \frac{}{\vdash \Gamma, \top} \qquad [\bot]\ \frac{\vdash \Gamma}{\vdash \Gamma, \bot} \qquad [\mathbf{1}]\ \frac{}{\vdash \mathbf{1}} \qquad [\parr]\ \frac{\vdash \Gamma, F, G}{\vdash \Gamma, F \parr G}$$

$$[\otimes]\ \frac{\vdash \Gamma, F \qquad \vdash \Delta, G}{\vdash \Gamma, \Delta, F \otimes G} \qquad [\&]\ \frac{\vdash \Gamma, F \qquad \vdash \Gamma, G}{\vdash \Gamma, F \,\&\, G} \qquad [\oplus]\ \frac{\vdash \Gamma, A_i}{\vdash \Gamma, A_0 \oplus A_1}\ i \in \{0,1\}$$

$$[\nu]\ \frac{\vdash \Gamma, F\{\nu X.F/X\}}{\vdash \Gamma, \nu X.F} \qquad [\mu]\ \frac{\vdash \Gamma, F\{\mu X.F/X\}}{\vdash \Gamma, \mu X.F}$$

The typing rules of $\mu\mathsf{CP}^{\infty}$ are shown in Table 3 and, with the exception of [CALL], they correspond to the proof rules of $\mu\mathsf{MALL}^{\infty}$ (Table 4) in which the context is the sequent being proved and the process is a syntactic representation of the proof. The rules for the multiplicative constants and for the connectives are standard. The rule [$\top$] allows every process (whose free names are in the domain of the typing context) to be declared well typed if the context contains an association $x : \top$. As a special case, the failed process $\mathsf{fail}\ x$ can be typed only by such rule. While $\mathsf{fail}\ x$ is somewhat redundant since any process can be used in its place, it is a convenient normal form for dead code. The side condition is not strictly necessary insofar the soundness of the type system is concerned, but it enforces the usual property that the channels occurring free in the process must be associated with a type in the context.

There is no rule for $\mathbf{0}$, as usual. The rules [σ] where $\sigma \in \{\mu, \nu\}$ simply unfold fixed points regardless of their nature. The rule [CALL] unfolds a process invocation into its definition, checking that the invocation and the definition are well typed in the same context. Finally, [CUT] checks that the composition $(x)(P \mid Q)$ is well typed provided that the behaviour of P with respect to x is complementary to the behaviour of Q with respect to x.

For readers familiar with the literature notice that axioms with conclusion $A, A^{\perp}$ are omitted; as is the corresponding "link" construct in the calculus. This is for brevity, since the sequent $A, A^{\perp}$ is provable for any type A. This axiom is perpendicular to this study and its omission happens to make connections with $\mu\mathsf{MALL}^{\infty}$ more immediate.

Like in $\mu\mathsf{MALL}^{\infty}$, the rules are meant to be interpreted coinductively so that a judgement $P \vdash \Gamma$ is deemed derivable if there is an arbitrary (finite or infinite) typing derivation whose conclusion is $P \vdash \Gamma$.

Example 2. Let us show the typing derivations for the processes discussed in Example 1. To this aim, let $A \stackrel{\mathsf{def}}{=} \mu X.\mathbf{0} \oplus (\mu X.\mathbf{1} \oplus \mathbf{0})$ and $B \stackrel{\mathsf{def}}{=} \nu X.(\bot \,\&\, X)$. Now we derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x[] \vdash x : \mathbf{1}}\,[\mathbf{1}]}{x[\mathsf{in}_0].x[] \vdash x : \mathbf{1} \oplus \mathbf{0}}\,[\oplus]}{\mathsf{rec}\ x.x[\mathsf{in}_0].x[] \vdash x : \mu X.\mathbf{1} \oplus \mathbf{0}}\,[\mu]}{x[\mathsf{in}_1].\mathsf{rec}\ x.x[\mathsf{in}_0].x[] \vdash x : \mathbf{0} \oplus (\mu X.\mathbf{1} \oplus \mathbf{0})}\,[\oplus]}{\mathsf{rec}\ x.x[\mathsf{in}_1].\mathsf{rec}\ x.x[\mathsf{in}_0].x[] \vdash x : A}\,[\mu]}{\mathsf{Client}\langle x\rangle \vdash x : A}\,[\textsc{call}]$$

as well as

$$\dfrac{\dfrac{\dfrac{}{z[] \vdash z : \mathbf{1}}\,[\mathbf{1}]}{x().z[] \vdash x : \bot, z : \mathbf{1}}\,[\bot] \qquad \dfrac{\vdots}{\mathsf{Server}\langle x, z\rangle \vdash x : B, z : \mathbf{1}}}{\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z\rangle\} \vdash x : \bot \,\&\, B, z : \mathbf{1}}\,[\&]}{\mathsf{corec}\ x.\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z\rangle\} \vdash x : B, z : \mathbf{1}}\,[\nu]}{\mathsf{Server}\langle x, z\rangle \vdash x : B, z : \mathbf{1}}\,[\textsc{call}]$$

Intuitively, these derivations assign Client and Server their *most precise* types. Yet, we cannot use *these* typing derivations to connect Client and Server via a session x by means of the rule [CUT] since $A \neq B^{\perp}$. The idea is that it should be possible to exploit the relation $A \leqslant B^{\perp}$ (or its dual $B \leqslant A^{\perp}$) to obtain a well-typed composition. We will explore this possibility in the next sections. ⌟

It is a known fact that not every $\mu\mathsf{MALL}^{\infty}$ derivation is a valid one [15,16,20]. In order to characterize the valid derivations we need some auxiliary notions which we recall below.

Definition 2 *(Thread).* Let $\gamma = (P_i \vdash \Gamma_i)_{i\in\mathbb{N}}$ be an infinite branch in a typing derivation and recall that $P_{i+1} \vdash \Gamma_{i+1}$ is a premise of $P_i \vdash \Gamma_i$. A *thread* of γ is a sequence $(x_i)_{i\geq k}$ of channels such that $x_i \in \mathsf{dom}(\Gamma_i)$ and either $x_i = x_{i+1}$ or $P_i = x_i[x_{i+1}](P_{i+1} \mid Q)$ or $P_i = x_i(x_{i+1}).P_{i+1}$ for every $i \geq k$.

Intuitively, a thread is an infinite sequence of channel names $(x_i)_{i\geq k}$ that are found starting from some position k in an infinite branch $(P_i \vdash \Gamma_i)_{i\in\mathbb{N}}$ and that pertain to the same session. For example, consider the derivation for Server in Example 2 and observe that there is only one infinite branch, the rightmost one. The sequence $(x, x, x, \dots)$ is a thread that starts at the root of the derivation. In general such threads need not start at the root of the derivation, hence the condition $i \geq k$ above. This is because, the channels involved in a thread may begin with a channel created by a cut rule.

Definition 3 *(ν-thread).* Given a branch $\gamma = (P_i \vdash \Gamma_i)_{i\in\mathbb{N}}$ and a thread $t = (x_i)_{i\geq k}$ of γ, we write $\mathsf{inf}(\gamma, t) \stackrel{\mathsf{def}}{=} \{A \mid \exists^{\infty} i \geq k : \Gamma_i(x_i) = A\}$. We say that t is a *ν-thread* of γ if $\min \mathsf{inf}(\gamma, t)$ is a ν-type. Hereafter $\exists^{\infty} i$ means the existence of infinitely many i's with the stated property.

Given an infinite branch $\gamma = (P_i \vdash \Gamma_i)_{i\in\mathbb{N}}$ and a thread $t = (x_i)_{i\geq k}$ of γ, the thread identifies an infinite sequence $(\Gamma_i(x_i))_{i\geq k}$ of types. The set $\mathsf{inf}(\gamma, t)$ is the set of those types that occur infinitely often in this sequence and $\min \mathsf{inf}(\gamma, t)$ is the $\preceq$-minimum among these types (it can be shown that the minimum of any set $\mathsf{inf}(\gamma, t)$ is always defined [16]). We say that t is a ν-thread if such minimum type is a ν-type. In Example 2, the thread $t = (x, x, x, \dots)$ identifies the sequence $(B, B, \bot \,\&\, B, B, \dots)$ of types in which both B and $\bot \,\&\, B$ occur infinitely often. Since $B \preceq \bot \,\&\, B$ and B is a ν-type we conclude that t is a ν-thread.

Definition 4 *(Valid branch).* Let $\gamma = (P_i \vdash \Gamma_i)_{i\in\mathbb{N}}$ be an infinite branch of a typing derivation. We say that γ is *valid* if there is a ν-thread $(x_i)_{i\geq k}$ of γ such that $[\nu]$ is applied to infinitely many of the x_i.

Definition 4 establishes that a branch is valid if it contains a ν-thread in which the ν-type occurring infinitely often is also unfolded infinitely often. This happens in Example 2, in which the $[\nu]$ rule is applied infinitely often to unfold the type of x. The reader familiar with the $\mu\mathsf{MALL}^{\infty}$ literature may have spotted a subtle difference between our notion of valid branch and the standard one [15,16]. In $\mu\mathsf{MALL}^{\infty}$, a branch is valid only provided that the ν-thread in it is not "eventually constant", namely if the greatest fixed point that defines the ν-thread is unfolded infinitely many times. This condition is satisfied by our notion of valid branch because of the requirement that there must be infinitely many applications of $[\nu]$ concerning the names in the ν-thread. Now we can define the notion of valid typing derivation.

Definition 5. A typing derivation is *valid* if so are all its infinite branches.

Example 3. Consider a client $\mathsf{Chatter}(x) \triangleq \mathsf{rec}\ x.x[\mathsf{in}_1].\mathsf{Chatter}\langle x\rangle$ that engages into an infinite interaction with Server from Example 1 and let $C \stackrel{\text{def}}{=} \nu X.(\mathbf{1} \oplus X)$. Note that $C = B^{\perp}$ where B is the type used in Example 2. Now the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{Chatter}\langle x\rangle \vdash x : C}\,[\text{CALL}]}{x[\mathsf{in}_1].\mathsf{Chatter}\langle x\rangle \vdash x : \mathbf{1} \oplus C}\,[\oplus]}{\mathsf{rec}\ x.x[\mathsf{in}_1].\mathsf{Chatter}\langle x\rangle \vdash x : C}\,[\mu]}{\mathsf{Chatter}\langle x\rangle \vdash x : C}\,[\text{CALL}]$$

is invalid since the only infinite branch does not contain a ν-thread. If we allowed this derivation the composition $(x)(\mathsf{Chatter}\langle x\rangle \mid \mathsf{Server}\langle x, z\rangle)$ would be well typed and it would no longer be the case that well-typed processes terminate, as the interaction between Chatter and Server goes on forever. ⌟

Example 4. Similarly to the above example, the following pre-proof is not a valid proof, where $\mathsf{A} \triangleq (y)(y[] \mid y().\mathsf{A})$.

$$\dfrac{\dfrac{}{y[] \vdash x : \mathbf{1}}\,[\mathbf{1}] \qquad \dfrac{\dfrac{\vdots}{\mathsf{A} \vdash x : \mathbf{1}}}{y().\mathsf{A} \vdash y : \perp, x : \mathbf{1}}\,[\perp]}{\mathsf{A} \vdash x : \mathbf{1}}\,[\text{CUT}]$$

The process clearly reduces forever. The typing is however not valid because there is an infinite branch of the proof with a thread identifying only the type $\mathbf{1}$ infinitely often. Clearly $\mathbf{1}$ is not a ν-type, nor can it be unfolded at all. The fact that the process does reduce forever without demanding external interactions emphasises again the importance of validity. ⌟

Example 5. To understand better the implications of restricting ν-threads to only minimal types along a path, consider the following pre-proof. Let $\mathsf{A}(x) \triangleq \mathsf{rec}\ x.\mathsf{corec}\ x.\mathsf{A}\langle x\rangle$ and $\mathsf{B}(x) \triangleq \mathsf{corec}\ x.\mathsf{rec}\ x.\mathsf{B}\langle x\rangle$. We have:

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{A}\langle x\rangle \vdash x : \mu X.\nu Y.X}}{\mathsf{corec}\ x.\mathsf{A}\langle x\rangle \vdash x : \nu Y.\mu X.\nu Y.X}\,[\nu] \qquad \dfrac{\dfrac{\vdots}{\mathsf{B}\langle x\rangle \vdash x : \nu X.\mu Y.X, \Gamma}}{\mathsf{rec}\ x.\mathsf{B}\langle x\rangle \vdash x : \mu Y.\nu X.\mu Y.X, \Gamma}\,[\mu]}{\dfrac{\mathsf{A}\langle x\rangle \vdash x : \mu X.\nu Y.X \qquad \mathsf{B}\langle x\rangle \vdash x : \nu X.\mu Y.X, \Gamma}{(x)(\mathsf{A}\langle x\rangle \mid \mathsf{B}\langle x\rangle) \vdash \Gamma}\,[\text{CUT}]}$$

If the above pre-proof were a proof, then $\mu\mathsf{CP}^{\infty}$ would be inconsistent, since any context Γ, even the empty one, types the given process. The above pre-proof is however not valid since there is a thread along the left branch of the cut such that, although there are infinitely many unfoldings of a greatest fixed point, the type $\nu Y.\mu X.\nu Y.X$ that is unfolded is not $\preceq$-minimal along that branch, since $\mu X.\nu Y.X$ also occurs infinitely often and $\mu X.\nu Y.X \preceq \nu Y.\mu X.\nu Y.X$. ⌟

4. Behavioural properties of $\mu\mathbf{CP}^{\infty}$

We explain here some behavioural properties of $\mu\mathsf{CP}^{\infty}$ that are guaranteed by its relationship with $\mu\mathsf{MALL}^{\infty}$. We focus on termination, which can be guaranteed in certain typing contexts. In particular, processes that are well typed in a singleton context of the form $x : \mathbf{1}$ eventually reduce to $x[]$ in a finite number of steps. Formally, the above observation can be stated as follows.

Theorem 1. *If $P \vdash x : \mathbf{1}$ then P is terminating.*

The essence of the proof is easy to see once we note a few things. Firstly, there is a straightforward correspondence between $\mu\mathsf{CP}^{\infty}$ proofs and $\mu\mathsf{MALL}^{\infty}$ proofs, hence a valid typing derivation for $P \vdash x : \mathbf{1}$ corresponds to a valid $\mu\mathsf{MALL}^{\infty}$ proof with conclusion $\vdash \mathbf{1}$. Next, we appeal to a Curry-Howard correspondence to ensure that any maximal reduction sequence in $\mu\mathsf{CP}^{\infty}$ corresponds to a maximal reduction sequence in $\mu\mathsf{MALL}^{\infty}$ (of a particular form that we will clarify next). In $\mu\mathsf{MALL}^{\infty}$ cut elimination [15,16] means that any initial part of a cut-free proof is produced in finitely many steps by cut reductions (of another particular form). Since there is exactly one cut-free proof of $\vdash \mathbf{1}$ in $\mu\mathsf{MALL}^{\infty}$ where only the axiom $[\mathbf{1}]$ is applied, and since no cut may appear above an axiom, this procedure must erase all cuts.

The devil-in-the-details of the above proof sketch appears inside the brackets. Specifically, cut elimination in $\mu\mathsf{MALL}^{\infty}$ is reliant on reduction sequences of a particular form and it is not quite immediate from the definitions that the obvious Curry-Howard correspondence with reduction sequences in $\mu\mathsf{CP}^{\infty}$ yields an appropriate reduction sequence. Indeed, in many contexts an arbitrary reduction in $\mu\mathsf{CP}^{\infty}$ does not guarantee an appropriate reduction sequence, hence the restriction to the $x : \mathbf{1}$ typing context. For example, even a simple finite process such as $(x)(z().x().y[] \mid x[])$, which is well typed in the context $z : \perp, y : \mathbf{1}$, deadlocks with

a cut that cannot be reduced – a deadlock which is to be expected for the Caires-Pfenning-Wadler tradition (even when there are commuting conversations). For the infinite processes introduced by $\mu\mathsf{MALL}^\infty$, termination has additional nuances. We explain next the nuances of reduction sequences in $\mu\mathsf{MALL}^\infty$, so that we may expand the above proof sketch to a proof of termination.

4.1. Basic notions on labelled transition systems and their properties

In this section, we develop in greater detail the proof of Theorem 1. Although our proof will rely on an established cut elimination property of $\mu\mathsf{MALL}^\infty$, it takes some work to prove the existence of a correspondence between the calculi from which the desired termination property follows for $\mu\mathsf{CP}^\infty$. Proving cut elimination and proving the results required to establish a correspondence are separate endeavours. The established cut elimination result in $\mu\mathsf{MALL}^\infty$, which we rely on within the termination proof, and which our correspondence results then harness, is quite complex to prove and is detailed elsewhere [16]. Thus we do not reproduce the established proof of cut elimination for $\mu\mathsf{MALL}^\infty$ here. In what follows in this section, what we do provide is the details required to establish a correspondence that accounts for the differing reduction strategies of $\mu\mathsf{MALL}^\infty$ and $\mu\mathsf{CP}^\infty$. We then go on to put together the established cut elimination result and details of the correspondence to establish our termination result. The following generic definitions will be used.

A *labelled transition system* is a triple $(S, \mathcal{L}, \longrightarrow)$ consisting of a set of states S, a set of transition labels $\mathcal{L}$ and a transition relation $\longrightarrow \subseteq S \times \mathcal{L} \times S$. Hereafter, once we have fixed a labelled transition system $(S, \mathcal{L}, \longrightarrow)$, for every $S, T \in S$ and $\ell \in \mathcal{L}$ we will write $S \xrightarrow{\ell} T$ if $(S, \ell, T) \in \longrightarrow$, we will write $S \xrightarrow{\ell}$ if $S \xrightarrow{\ell} T$ for some T and we will write $S \xrightarrow{\ell}\!\!\!\!\!\!\not\;\;$ if not $S \xrightarrow{\ell}$. This definition is generic; labels will be instantiated shortly so that they identify instances of formulas involved in a cut.

A *reduction sequence* is a finite or infinite sequence $S_1 \ell_1 S_2 \ell_2 \cdots$ of alternated states and labels not ending with a label and such that $S_i \xrightarrow{\ell_i} S_{i+1}$ whenever $i+1$ is a valid index of the sequence. Given a reduction sequence $S_1 \ell_1 S_2 \ell_2 \cdots$ and a label ℓ, we say that ℓ is *enabled* (respectively *performed*) in the sequence if $S_i \xrightarrow{\ell}$ (respectively $\ell_i = \ell$) for some valid index i. Note that a performed transition is necessarily enabled, while an enabled transition is not necessarily performed. We say that a labelled transition system $(S, \mathcal{L}, \longrightarrow)$ is *event labelled* whenever, if a ℓ-labelled transition is enabled in two states then an ℓ-labelled transition is enabled in all states between yet is never performed except perhaps after the later of the two states. A consequence of this is that every reduction sequence does not contain duplicate labels, and hence in an event-labelled transition system each transition may be performed *at most once* in the evolution of a state.

We say that a reduction sequence $S_1 \ell_1 S_2 \ell_2 \cdots$ is *fair* if, for every valid index i with $S_i \xrightarrow{\ell}$, there exists $j > i$ such that $S_j \xrightarrow{\ell}\!\!\!\!\!\!\not\;\;$. That is, in a fair reduction sequence no transition can remain enabled forever. This also means that every finite fair transition sequence ends in a state that has no outgoing transitions. Note that this notion of "fairness" is formulated in a slightly non-standard way, but is aligned with the definition of fairness used in the proof of cut elimination for $\mu\mathsf{MALL}^\infty$ [16], which is why we choose it. Besides, for event-labelled transition systems, all reasonable notions of fairness collapse (cf. assumption (8) in [21]).

Given a labelled transition system $(S, \mathcal{L}, \longrightarrow)$, we say that $S \in S$ is *(fairly) terminating* if every (fair) reduction sequence starting with S is finite. Note that termination implies fair termination, but not vice versa.

We say that a labelled transition system $(S, \mathcal{L}, \longrightarrow)$ has the *Church-Rosser property* if, for every $S, S_1, S_2 \in S$ and $\ell_1, \ell_2 \in \mathcal{L}$ with $\ell_1 \neq \ell_2$, we have that $S \xrightarrow{\ell_i} S_i$ for every $i = 1, 2$ implies $S_i \xrightarrow{\ell_{3-i}} T$ for every $i = 1, 2$ and some $T \in S$. In a fair reduction sequence of a labelled transition system with the Church-Rosser property, every enabled transition is also performed.

Proposition 1. *Let $(S, \mathcal{L}, \mathcal{R})$ be an event-labelled transition system such that each state is Church-Rosser. If $S \in S$ is fairly terminating, then S is terminating.*

Proof. Suppose that S is not terminating. Then there exists an infinite reduction sequence $S \ell_1 S_1 \ell_2 S_2 \cdots$. Since the transition system is event labelled, the ℓ_i are all distinct and there are infinitely many of them. From the hypothesis that S is fairly terminating we deduce that every fair reduction sequence starting from S is finite. Moreover, since the transition system is Church-Rosser, every fair reduction sequence starting from S enables and performs the same finite set of transitions, in possibly different orders. Therefore, the set of transitions that are enabled and performed from S is finite. This contradicts the assumption that there exists an infinite reduction sequence starting from S. □

4.2. $\mu MALL^\infty$ as an event-labelled transition system

In this section we recall the main elements of the $\mu\mathsf{MALL}^\infty$ logical system, tailored so as to make it easier to relate them with $\mu\mathsf{CP}^\infty$.

Formulas. Formulas in $\mu\mathsf{MALL}^\infty$ are pairs A_α consisting of a type A (Section 3) and an *address* α that identifies the occurrence of the formula (and of its dual) within a proof derivation. We let F and G range over formulas. Addresses, ranged over by α and β, are strings from the alphabet $\{i, l, r\}$. We say that two addresses are *disjoint* if neither is a prefix of the other.

The dual of a formula is obtained by dualizing the type, that is $(A_\alpha)^\perp \stackrel{\text{def}}{=} A^\perp_\alpha$. It is convenient to define operators on formulas which are named after the connectives and fixed points of the logic and that behave thus:

$$(A_{\alpha l} \star B_{\alpha r}) \stackrel{\text{def}}{=} (A \star B)_\alpha \qquad \sigma X.(A_{\alpha i}) \stackrel{\text{def}}{=} (\sigma X.A)_\alpha$$

Also, we postulate that $F\{A_\alpha/X\}$ substitutes the free occurrences of X within F with A, ignoring α.

Proofs. $\mu\mathsf{MALL}^\infty$ proofs are built using the rules in Table 4, where we use the metavariables Γ and Δ to denote $\mu\mathsf{MALL}^\infty$ sequents. The context will allow us to disambiguate the interpretation of these metavariables as typing contexts or as sequents. The correspondence between proof rules of $\mu\mathsf{MALL}^\infty$ and typing rules of $\mu\mathsf{CP}^\infty$ is obvious and for this reason we use the same labels to name them. The only differences between the two systems is the use of formulas instead of types, the absence of proof terms (processes), and the absence of the rule [CALL].

We draw from a stream of disjoint addresses in order to label the formulas in the conclusion of the proof and also to label each occurrence of a cut in that proof. Note that we can always obtain an infinite stream of disjoint addresses by considering families of addresses that begin with a prefix of the form $l^n r$ for arbitrary $n \in \mathbb{N}$. The labelling of all other formulas in a proof follows from the labelling of formulas in the conclusion and those introduced by cut rules in a deterministic fashion following from the structure of rules. This labelling convention ensures the following property.

Proposition 2. *If the addresses of formulas introduced by cuts and the addresses of formulas in the conclusion of a μMALL$^\infty$ proof are all disjoint, then every formula in each sequent of a proof has disjoint labels.*

The above follows immediately by inspection of the rules in Table 4.

Proof congruence. In general, the cut elimination procedure for $\mu\mathsf{MALL}^\infty$ requires some rearrangements in the structure of cuts. To enable such rearrangements, we introduce a structural pre-congruence relation $\preccurlyeq$ for proofs analogous to the one we have defined for $\mu\mathsf{CP}^\infty$ processes. Since $\mu\mathsf{MALL}^\infty$ proofs do not contain process invocations, the $\preccurlyeq$ relation for $\mu\mathsf{MALL}^\infty$ proofs only concerns the commutativity and associativity of cuts.

$$\frac{\vdash \Gamma, F \qquad \vdash \Delta, F^\perp}{\vdash \Gamma, \Delta} \preccurlyeq \frac{\vdash \Delta, F^\perp \qquad \vdash \Gamma, F}{\vdash \Gamma, \Delta} \qquad \frac{\vdash \Gamma, F \qquad \dfrac{\vdash \Delta, F^\perp, G \qquad \vdash \Theta, G^\perp}{\vdash \Delta, \Theta, F^\perp}}{\vdash \Gamma, \Delta, \Theta} \preccurlyeq \frac{\dfrac{\vdash \Gamma, F \qquad \vdash \Delta, F^\perp, G}{\vdash \Gamma, \Delta, G} \qquad \vdash \Theta, G^\perp}{\vdash \Gamma, \Delta, \Theta}$$

In the literature on $\mu\mathsf{MALL}^\infty$, the problem of rearranging cuts is addressed in a different way by considering a logical system called $\mu\mathsf{MALL}^\infty_{\mathsf{m}}$ that conservatively extends $\mu\mathsf{MALL}^\infty$ with a *multicut* rule.[4] Multicuts collapse a finite number of nested cuts into a single rule. It is easy to see that $\preccurlyeq$ relates $\mu\mathsf{MALL}^\infty$ proofs represented by the *same* multicut in $\mu\mathsf{MALL}^\infty_{\mathsf{m}}$. Using $\preccurlyeq$ instead of multicuts allows us to draw a closer relationship between $\mu\mathsf{CP}^\infty$ and $\mu\mathsf{MALL}^\infty$ both at the level of operational semantics and at the level of inference rules.

Internal reductions. Table 5 defines the *internal reductions* of $\mu\mathsf{MALL}^\infty$ (also called principal reductions in the proof-theory literature), which are those cut reduction steps in which a cut reduction eliminates dual constants, dual connectives or dual fixed points that are introduced right above the cut. Note that $\mu\mathsf{MALL}^\infty$ reductions are *labelled* with the address α that identifies the dual formulas being eliminated by the cut. The origin of this address from the proof is made explicit for example in the cut reduction of $\mathbf{1}$ and $\perp$ by indicating the address assigned with each unit. In the other rules, observe that the labels in the sequents appearing after the cut reflect the labels of the immediate subformulas of the principal connectives involved in the cut reduction step. The $\stackrel{\alpha}{\longrightarrow}$ relation is closed under $\preccurlyeq$, so that structurally pre-congruent proofs give raise to the same reductions (*cf.* [R-CONG]).

Also note that reductions preserve disjointness of labels. This follows by inspecting the form of addresses removed and created by internal cut reductions.

Proposition 3. *Both the disjointness of the every formula in each sequent of a proof and also the disjointness of each formula involved in a cut throughout a proof are preserved by internal reductions.*

Remark 2 *(external reductions).* In this paper we present only the internal reductions that correspond to reduction step in $\mu\mathsf{CP}^\infty$, and exclude the *external reductions* that feature in $\mu\mathsf{MALL}^\infty$. External reductions occur when the rule immediately above either branch of a cut does not interact at all with the formula being cut. In such cases, the rule passes through the cut and becomes a rule below the cut. The design decision to exclude such external reductions is so as to align strongly with $\mu\mathsf{CP}^\infty$. In turn, the design decision to exclude rules corresponding to external reductions in $\mu\mathsf{CP}^\infty$ originates in the work of Wadler [4], where he argues that external reductions should be excluded since they allow prefixes to move around in a way that does not mimic synchronous process calculi. This assumption has been challenged in work that allows external reductions, where the movement of outputs prefixes is used to model asynchrony to different degrees [23,24]. External reductions that propagate rules below the bottom-most cuts may alternatively be

[4] Not to be confused with the multicut rule used to cut multiple "coherent" parties [22].

Table 5
Internal reductions in $\mu\mathsf{MALL}^\infty$.

$$\frac{\dfrac{}{\vdash \mathbf{1}_\alpha}\,[\mathbf{1}] \qquad \dfrac{\vdash \Gamma}{\vdash \Gamma, \bot_\alpha}\,[\bot]}{\vdash \Gamma}\,[\textsc{cut}] \xrightarrow{\alpha} \vdash \Gamma$$

$$\frac{\dfrac{\vdash \Gamma, A_{al} \qquad \vdash \Delta, B_{ar}}{\vdash \Gamma, \Delta, (A \otimes B)_\alpha}\,[\otimes] \qquad \dfrac{\vdash \Theta, A^\perp_{al}, B^\perp_{ar}}{\vdash \Theta, (A^\perp \parr B^\perp)_\alpha}\,[\parr]}{\vdash \Gamma, \Delta, \Theta}\,[\textsc{cut}] \xrightarrow{\alpha}$$

$$\frac{\vdash \Gamma, A_{al} \qquad \dfrac{\vdash \Delta, B_{ar} \qquad \vdash \Theta, A^\perp_{al}, B^\perp_{ar}}{\vdash \Delta, \Theta, A^\perp_{al}}\,[\textsc{cut}]}{\vdash \Gamma, \Delta, \Theta}\,[\textsc{cut}]$$

$$\frac{\dfrac{\vdash \Gamma, A_{al}}{\vdash \Gamma, (A \oplus B)_\alpha}\,[\oplus] \qquad \dfrac{\vdash \Delta, A^\perp_{al} \qquad \vdash \Delta, B^\perp_{ar}}{\vdash \Delta, (A^\perp \,\&\, B^\perp)_\alpha}\,[\&]}{\vdash \Gamma, \Delta}\,[\textsc{cut}] \xrightarrow{\alpha}$$

$$\frac{\vdash \Gamma, A_{al} \qquad \vdash \Delta, A^\perp_{al}}{\vdash \Gamma, \Delta}\,[\textsc{cut}]$$

$$\frac{\dfrac{\vdash \Gamma, (A\{\mu X.A/X\})_{ai}}{\vdash \Gamma, (\mu X.A)_\alpha}\,[\mu] \qquad \dfrac{\vdash \Delta, (A^\perp\{\nu X.A^\perp/X\})_{ai}}{\vdash \Delta, (\nu X.A^\perp)_\alpha}\,[\nu]}{\vdash \Gamma, \Delta}\,[\textsc{cut}] \xrightarrow{\alpha}$$

$$\frac{\vdash \Gamma, (A\{\mu X.A/X\})_{ai} \qquad \vdash \Delta, (A^\perp\{\nu X.A^\perp/X\})_{ai}}{\vdash \Gamma, \Delta}\,[\textsc{cut}]$$

interpreted as interactions with an external environment, much like "catalysers" in dynamically interleaved sessions [25]. Indeed, there is a line of work that interprets the rules produced by external reductions, which opens up a whole world where we start to consider richer behavioural properties guaranteed by provable external actions of a session calculus [7]. Thus a crucial difference between the logical interpretation in the current paper and that prior work is whether we interpret only internal reductions or whether we interpret also the rules produced by external reductions in our behavioural properties, and since the latter allows us to interpret infinite reduction sequences, then behavioural properties beyond termination become significant. There remains much unexplored territory in that direction; hence our decision to consider a calculus where external reductions are excluded and rules on external channels are not interpreted is a political choice, fixing a minimal setting that is more widely adopted [26,6]. ⌟

4.3. Reduction strategies and termination

Cut elimination procedures in infinitary proof calculi depend on being able to reduce a proof such that any initial part of a cut-free proof may be produced. Cuts may be reduced by a number of strategies, and not all strategies yield a proof. Obviously, if there are infinitely many cuts that can be reduced, then if we allow cuts to be reduced anywhere in any order then there exists an infinite reduction sequence that never reduces the bottom-most cut and hence never produces a cut-free proof (cf. Example 6).

There are two reduction strategies that appear in our termination argument. Firstly, there is the head reduction strategy which, for internal reductions, corresponds tightly to reductions in $\mu\mathsf{CP}^\infty$.

Head reductions. We say that a cut in a proof derivation is *unguarded* if the path from the root of the derivation to the cut only goes through other cuts. A *head reduction strategy* arises when we restrict $\xrightarrow{\alpha}$ to *unguarded* cuts. This closely mimics the reduction strategy of $\mu\mathsf{CP}^\infty$, where reductions are not allowed to occur under prefixes.

When we consider internal head reductions only, that is, unguarded cut reduction steps following Table 5, we obtain a Church-Rosser property.

Proposition 4. *Assuming cuts are assigned disjoint labels, internal head reductions are Church-Rosser.*

Proof. Since all cuts have disjoint addresses and are replaced by longer labels, they are disabled once they are reduced. Because a head reduction strategy allows only reductions on unguarded cuts, no rule can disable another cut. □

We also need to reason about the following reduction strategy, since, in our correspondence results, there is a need to bridge the gap between the reduction strategy in $\mu\mathsf{MALL}^\infty$ and $\mu\mathsf{CP}^\infty$.

Bottom-most reductions. In this strategy a head reduction can be applied to any subtree of a proof where no cut rule appears along the path from the root to that sub-tree. This has the effect of reducing cuts deep inside the tree as long as they are in the bottom-most multicut along a branch of a tree. This bottom-most reduction strategy (which appears only implicitly in the text in established cut-elimination results for $\mu\mathsf{MALL}^\infty$ [15,16]), ensures cut elimination.

Example 6. Consider the following proof, which is valid since the left branch unfolds $\nu X.X$ repeatedly, while the right branch loops back to the beginning thereby applying cuts repeatedly (as suggested by the corresponding process term $\mathsf{A}(x,y) \triangleq (z)(\mathsf{B}\langle x,z\rangle \mid \mathsf{rec}\ z.\mathsf{corec}\ y.\mathsf{A}\langle z,y\rangle)$, where $\mathsf{B}(x,z) \triangleq \mathsf{corec}\ z.\mathsf{B}\langle x,z\rangle$).

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \mu X.X, \nu X.X}}{\vdash \mu X.X, \nu X.X}\,[\nu] \qquad \dfrac{\dfrac{\dfrac{\vdots}{\vdash \mu X.X, \nu X.X}}{\vdash \mu X.X, \nu X.X}\,[\nu]}{\vdash \mu X.X, \nu X.X}\,[\mu]}{\vdash \mu X.X, \nu X.X}\,[\text{CUT}] \qquad \dfrac{\dfrac{\dfrac{\vdots}{\vdash \mu X.X, \nu X.X}}{\vdash \mu X.X, \nu X.X}\,[\nu]}{\vdash \mu X.X, \nu X.X}\,[\mu]}{\vdash \mu X.X, \nu X.X}\,[\text{CUT}]$$

The above proof has infinitely many nested cuts that are ready to reduce. We use this example next to illustrate differences between a head reduction strategy and a bottom-most reduction strategy.

In a head reduction strategy only the bottom-most cut will reduce by one step and then the cut reduction sequence will block due to the absence of external rules. In contrast, in a bottom-most reduction strategy consisting of internal and external reductions, an external rule would permute the unfolding of the greatest fixed point on the right past the cut. The cut would then form a larger multicut at the bottom of the proof and the reduction procedure would continue to produce $[\nu]$ rules above the new rule that is produced. That bottom-most reduction strategy therefore can produce any initial part of the cut-free proof of the conclusion where $\nu X.X$ is unfolded repeatedly (that is $\mathsf{B}\langle x,z\rangle$ above). This productivity is despite the size of the bottom-most multicut growing indefinitely. ⌟

We are now ready to appeal to cut elimination as established already for $\mu\mathsf{MALL}^\infty$ [15,16] in order to argue about behavioural properties of cut reductions with respect to particular reduction strategies. Under a fair bottom-most reduction strategy in the environment $x : \mathbf{1}$, all unguarded cuts are eventually erased. Cut elimination guarantees that the bottom-most cut eventually produces a rule, which can only be achieved when an external cut reduction step is encountered or all bottom-most cuts are erased. We can argue that an external rule is never applied and hence all unguarded cuts must be erased.

Proposition 5. *If π is a proof of $\vdash \mathbf{1}$ in $\mu MALL^\infty$, then every maximal* fair *sequence of internal and external bottom-most reductions applied to π (that is, according to the established strategy for $\mu MALL^\infty$) is finite and yields the unique cut-free proof of $\vdash \mathbf{1}$ in $\mu MALL^\infty$. Furthermore, that fair reduction sequence consists only of internal head reductions.*

Proof. Due to cut elimination in $\mu\mathsf{MALL}^\infty$ [15,16], under a fair bottom-most cut reduction strategy, any cuts at the head of a proof will eventually produce a rule. There can be no rule at the bottom of the proof other than a cut or the rule [**1**], since no cut reduction can modify rules below the bottom-most cuts. Furthermore, since any external reductions applied at the bottom of a proof produces a rule other than [**1**] or cut, then no external rule may appear in the sequence of cut reductions. Since no rule can appear above [**1**] once it is produced, and internal rules may be applied only to the bottom-most rule, all internal reductions must be applied before the rule [**1**] is produced and hence must be applied to unguarded cuts. Furthermore, no rule other than [**1**] may appear in the resulting proof and hence the resulting sequence of internal head reductions must terminate. □

Since we know that the reduction sequence for the proofs of $\vdash \mathbf{1}$ must consist of only internal reductions applied at the head of a proof, we know we can restrict our analysis to unguarded cuts following the reduction system in Table 5. Since internal head reductions have the Church-Rosser property (Proposition 4), we can apply Proposition 1 to eliminate the fairness assumption from the above proof to obtain the following result.

Proposition 6. *If π is a proof of $\vdash \mathbf{1}$ in $\mu MALL^\infty$, then every sequence of internal head reductions applied to π eventually terminates yielding the unique cut-free proof of $\vdash \mathbf{1}$.*

Proof. Assume that π is a proof of $\vdash \mathbf{1}$ in $\mu\mathsf{MALL}^\infty$. Suppose that there is an infinite sequence of internal head reductions applied to π. By Proposition 5, every bottom-most sequence of *fair* internal and external reductions applied to π:

- eventually terminates yielding the unique cut-free proof of $\vdash \mathbf{1}$;

- consists only of internal head reductions.

The problem at this point is that we want to observe that since all these *fair* reduction sequence are terminating we can apply Proposition 1 to conclude that all reduction sequences are terminating independently of fairness. Yet this argument relies on **every** fair sequence of internal head reduction being covered by the above construction. The above does not (yet) exclude the possibility there is an infinite reduction sequence that is fair w.r.t. only the sub-system of $\mu\mathsf{MALL}^\infty$ that does not occur if we were to consider fair sequences for the whole of $\mu\mathsf{MALL}^\infty$, including all external bottom-most reductions of $\mu\mathsf{MALL}^\infty$.

Now suppose that, for π, there is a fair reduction sequence σ of head reduction using only the internal reductions in Table 5. We now aim to establish that, if such a σ were to exist, then σ would also be a fair bottom-most reduction sequence with respect to all internal and external reductions in $\mu\mathsf{MALL}^\infty$ [15,16]. Suppose, for contradiction, that σ were not fair in $\mu\mathsf{MALL}^\infty$ with respect to all bottom-most reductions. We break this failure of fairness with respect to all rules down into two possibilities:

1. there exists an external cut reductions step *er* that is perpetually enabled just above the bottom-most cuts (but never occurs nor is disabled);
2. there exists an internal cut reduction step *ir* that is perpetually enabled at the bottom of the proof (but never occurs nor is disabled).

In the first case above, since there is no external reduction in σ, clearly *er* never occurs in σ. Yet, since we assumed that it is perpetually enabled, this would mean that any prefix of σ can be extended to a fair bottom-most reduction sequence (not necessarily infinite) in $\mu\mathsf{MALL}^\infty$ in which *er* occurs. Thus by cut elimination for $\mu\mathsf{MALL}^\infty$ [15,16] that reduction sequence would produce the unique cut-free proof of $\vdash \mathbf{1}$. Now, observe that every external cut reduction step applied at the bottom of a proof, such as *er*, produces a new rule that appears below the bottom-most cuts in the proof. Yet $\vdash \mathbf{1}$ has only one cut-free proof consisting of a rule that cannot be produced by any external reduction of $\mu\mathsf{MALL}^\infty$, yielding a contraction. Hence there can be no external cut reduction step in σ that is perpetually enabled.

Now consider the second case above. We have assumed that the internal reduction step *ir* is enabled perpetually along σ. The enabled reduction step must therefore be the reduction of an unguarded cut at the bottom of the proof, and hence is a reduction following the rules in Table 5. Therefore, by fairness of σ with respect to internal head reductions, that reduction rule must eventually occur or be disabled, and hence cannot be perpetual.

Thereby we can conclude that σ must be fair in $\mu\mathsf{MALL}^\infty$ also with respect to all internal and external reductions (by construction, we already knew it consists only of internal reductions). Thus σ must therefore be finite, by cut elimination. Since σ is an arbitrary fair internal head sequence of π and it is finite, we can say that π is fairly terminating with respect to internal head reductions. Since internal head reductions form an event-labelled transition system the initial addresses labelling cuts are disjoint (Proposition 2), disjointness is preserved by internal reductions (Proposition 3), and proof for which labels are disjoint are Church-Rosser (Proposition 4), all proofs in σ are Church-Rosser. Therefore, we can apply Proposition 1, to conclude that π is terminating with respect to internal head reductions. □

4.4. Translation of $\mu\mathsf{CP}^\infty$ typing derivations into $\mu MALL^\infty$ proofs

The previous section concerns only internal head reduction sequences applied to $\mu\mathsf{MALL}^\infty$ proof. To link these results to $\mu\mathsf{CP}^\infty$ we rely on some properties that relate these calculi. The mapping here makes precise the Curry-Howard correspondence at play in this work.

The translation of $\mu\mathsf{CP}^\infty$ typing contexts into $\mu\mathsf{MALL}^\infty$ sequents is defined by a function $\lfloor\cdot\rfloor_\Sigma$ parametrised by a partial map Σ from channel names to addresses and defined as

$$\lfloor x_1 : A_1, \dots, x_n : A_n \rfloor_\Sigma = (A_1)_{\Sigma(x_1)}, \dots, (A_n)_{\Sigma(x_n)}$$

Assuming that the addresses in the range of Σ are disjoint, the formulas in $\lfloor\Gamma\rfloor_\Sigma$ have disjoint addresses as required by the notion of $\mu\mathsf{MALL}^\infty$ sequent.

The translation of $\mu\mathsf{CP}^\infty$ typing derivations into $\mu\mathsf{MALL}^\infty$ proof derivations is given by a function $\lfloor\cdot\rfloor^\sigma_\Sigma$ parametrized by an infinite stream σ of addresses and by a partial map Σ from channel names to addresses. The function is corecursively defined by the equations in Table 6, where we write $\pi :: P \vdash \Gamma$ to name π the derivation that concludes $P \vdash \Gamma$. This way, we can refer to such derivation on the right hand side of the translation wherever necessary. The translation is quite straightforward, the only necessary remarks concern the notation used to consume the stream σ and to extend the map Σ. We model a stream as a function from natural numbers to addresses and write $\alpha :: \sigma$ for the stream that begins with α and continues as σ. We write σ' and σ'' for the streams respectively defined by

$$(\sigma')(n) \stackrel{\text{def}}{=} \sigma(2n+1) \qquad (\sigma'')(n) \stackrel{\text{def}}{=} \sigma(2n)$$

so that σ' (respectively σ'') is the stream consisting of the addresses of σ in odd (respectively even) position. These operators allow us to split a stream into two disjoint streams, which we do whenever we translate a typing rule with two premises. We write $x \mapsto \alpha$ for the (partial) map from channel names to addresses that maps x to α and we write Σ, Σ' for the union of Σ and Σ' when they have disjoint domains. Note that in Table 6 we only show the case for in_0 in the translation of $\oplus$, the other case being analogous.

Table 6
Translation of $\mu\mathsf{CP}^{\infty}$ typing derivations into $\mu\mathsf{MALL}^{\infty}$ proofs.

$$\left\lfloor \frac{}{\mathsf{fail}\ x \vdash \Gamma, x : \top}\ [\top] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, \top_{\alpha}}\ [\top] \qquad \left\lfloor \frac{}{x[] \vdash x : \mathbf{1}}\ [\mathbf{1}] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{}{\vdash \mathbf{1}_{\alpha}}\ [\mathbf{1}]$$

$$\left\lfloor \frac{\pi :: P \vdash \Gamma}{x().P \vdash \Gamma, x : \bot}\ [\bot] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi \rfloor^{\sigma}_{\Sigma}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, \bot_{\alpha}}\ [\bot]$$

$$\left\lfloor \frac{\pi_1 :: P \vdash \Gamma, y : A \qquad \pi_2 :: Q \vdash \Delta, x : B}{x[y](P \mid Q) \vdash \Gamma, \Delta, x : A \otimes B}\ [\otimes] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi_1 \rfloor^{\sigma'}_{\Sigma, y \mapsto \alpha l} \qquad \lfloor \pi_2 \rfloor^{\sigma''}_{\Sigma, x \mapsto \alpha r}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, \lfloor \Delta \rfloor_{\Sigma}, (A \otimes B)_{\alpha}}\ [\otimes]$$

$$\left\lfloor \frac{\pi :: P \vdash \Gamma, y : A, x : B}{x(y).P \vdash \Gamma, x : A \parr B}\ [\parr] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi \rfloor^{\sigma}_{\Sigma, y \mapsto \alpha l, x \mapsto \alpha r}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, (A \parr B)_{\alpha}}\ [\parr]$$

$$\left\lfloor \frac{\pi :: P \vdash \Gamma, x : A}{x[\mathsf{in}_0].P \vdash \Gamma, x : A \oplus B}\ [\oplus] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi \rfloor^{\sigma}_{\Sigma, x \mapsto \alpha l}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, (A \oplus B)_{\alpha}}\ [\oplus]$$

$$\left\lfloor \frac{\pi_1 :: P \vdash \Gamma, x : A \qquad \pi_2 :: Q \vdash \Gamma, x : B}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : A \,\&\, B}\ [\&] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi_1 \rfloor^{\sigma'}_{\Sigma, x \mapsto \alpha l} \qquad \lfloor \pi_2 \rfloor^{\sigma''}_{\Sigma, x \mapsto \alpha r}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, (A \,\&\, B)_{\alpha}}\ [\&]$$

$$\left\lfloor \frac{\pi :: P \vdash \Gamma, x : A\{\nu X.A/X\}}{\mathsf{corec}\ x.P \vdash \Gamma, x : \nu X.A}\ [\nu] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi \rfloor^{\sigma}_{\Sigma, x \mapsto \alpha i}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, (\nu X.A)_{\alpha}}\ [\nu]$$

$$\left\lfloor \frac{\pi :: P \vdash \Gamma, x : A\{\mu X.A/X\}}{\mathsf{rec}\ x.P \vdash \Gamma, x : \mu X.A}\ [\mu] \right\rfloor^{\sigma}_{\Sigma, x \mapsto \alpha} = \frac{\lfloor \pi \rfloor^{\sigma}_{\Sigma, x \mapsto \alpha i}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, (\mu X.A)_{\alpha}}\ [\mu]$$

$$\left\lfloor \frac{\pi_1 :: P \vdash \Gamma, x : A \qquad \pi_2 :: Q \vdash \Delta, x : A^{\perp}}{(x)(P \mid Q) \vdash \Gamma, \Delta}\ [\mathrm{CUT}] \right\rfloor^{\alpha :: \sigma}_{\Sigma} = \frac{\lfloor \pi_1 \rfloor^{\sigma'}_{\Sigma, x \mapsto \alpha} \qquad \lfloor \pi_2 \rfloor^{\sigma''}_{\Sigma, x \mapsto \alpha}}{\vdash \lfloor \Gamma \rfloor_{\Sigma}, \lfloor \Delta \rfloor_{\Sigma}}\ [\mathrm{CUT}]$$

$$\left\lfloor \frac{\pi :: P \vdash \overline{x : A}}{A\langle \overline{x} \rangle \vdash \overline{x : A}}\ [\mathrm{CALL}] \right\rfloor^{\sigma}_{\Sigma} = \lfloor \pi \rfloor^{\sigma}_{\Sigma} \quad \text{if } A(\overline{x}) \triangleq P$$

Hereafter we always assume that streams consist of pairwise disjoint addresses. Similarly to Proposition 2, the choice of labelling preserves disjointness throughout a proof without the need to generate fresh addresses.

Now we have:

Proposition 7. *If $\alpha :: \sigma$ is a stream of disjoint addresses and $\pi :: P \vdash x : \mathbf{1}$ is a valid typing derivation, then $\lfloor \pi \rfloor^{\sigma}_{x \mapsto \alpha}$ is a valid $\mu MALL^{\infty}$ proof with conclusion $\vdash \mathbf{1}$.*

Proof. The validity condition for $\mu\mathsf{CP}^{\infty}$ typing derivations is modelled after that for $\mu\mathsf{MALL}^{\infty}$ proofs. □

Proposition 8. *Let $\pi_i :: P_i \vdash \overline{x : A}$ for $i = 1, 2$ and $\overline{\alpha} :: \sigma_1$ be a stream of disjoint addresses and $P_1 \to P_2$. Then $\lfloor \pi_1 \rfloor^{\sigma_1}_{x \mapsto \alpha} \xrightarrow{\beta} \lfloor \pi_2 \rfloor^{\sigma_2}_{x \mapsto \alpha}$ for some σ_2 such that $\overline{\alpha} :: \sigma_2$ is a stream of disjoint addresses and some $\beta \in \sigma_1 \setminus \sigma_2$. Furthermore the given reduction is an internal head reduction.*

Proposition 9. *Let $\pi_1 :: P_1 \vdash \overline{x : A}$ and $\overline{\alpha} :: \sigma_1$ be a stream of disjoint addresses and $\lfloor \pi_2 \rfloor^{\sigma_2}_{x \mapsto \alpha} \xrightarrow{\beta} \pi$. Then there exist P_2, π_2 and σ_2 such that $\pi_2 :: P_2 \vdash \overline{x : A}$ and $\pi = \lfloor \pi_2 \rfloor^{\sigma_2}_{x \mapsto \alpha}$ and $P_1 \to P_2$ and $\overline{\alpha} :: \sigma_2$ is a stream of disjoint addresses and $\beta \in \sigma_1 \setminus \sigma_2$.*

In Proposition 9, the address β uniquely identifies the two (dual) formulas in the cut being reduced. Depending on the shape of these formulas, in the proof tree after the reduction either β disappears completely (in case of a $[\mathbf{1}]/[\bot]$ reduction) or it is replaced by βi (in case of a $[\mu]/[\nu]$ reduction) or it is replaced by either βl or βr (in case of a $[\oplus]/[\&]$ reduction) or it is replaced by the two addresses βl and βr (in case of a $[\otimes]/[\parr]$ reduction). This is the reason why the translation of π_2 requires in general a stream of addresses σ_2 that is different from σ_1 used for the encoding of π_1.

4.5. Proof of Theorem 1

We are now ready to establish the main termination result of this work.

Theorem 1. *If $P \vdash x : \mathbf{1}$ then P is terminating.*

Table 7
Iso-recursive subtyping for session types.

$$\frac{}{\mathbf{0} \leqslant_{\mathsf{iso}} A}\ [\textsc{bot}] \qquad \frac{}{A \leqslant_{\mathsf{iso}} \top}\ [\textsc{top}] \qquad \frac{}{\kappa \leqslant_{\mathsf{iso}} \kappa}\ [\textsc{refl}] \qquad \frac{A \leqslant_{\mathsf{iso}} A' \qquad B \leqslant_{\mathsf{iso}} B'}{A \star B \leqslant_{\mathsf{iso}} A' \star B'}\ [\textsc{cong}]$$

$$\frac{A\{\sigma X.A/X\} \leqslant_{\mathsf{iso}} B\{\sigma X.B/X\}}{\sigma X.A \leqslant_{\mathsf{iso}} \sigma X.B}\ [\textsc{fix}]$$

Proof. We aim to establish that any process of $\mu\mathsf{CP}^{\infty}$, say P_0, is terminating whenever $\varpi_0 :: P_0 \vdash x : \mathbf{1}$ is a valid typing derivation. By Proposition 7, for some stream of disjoint addresses $\alpha_0 :: \sigma$, then $\pi_0 = \lfloor \varpi_0 \rfloor^{\sigma}_{x \mapsto \alpha_0}$ is a valid $\mu\mathsf{MALL}^{\infty}$ proof with conclusion $\vdash \mathbf{1}$.

Now consider any maximal reduction sequence of P_0, say θ, such that $P_0 \to P_1 \to \cdots$. By Proposition 8, there is a sequence of internal head reductions σ such that $\pi_0 \xrightarrow{\alpha_0} \pi_1 \xrightarrow{\alpha_1} \cdots$ where $\varpi_i :: P_i \vdash x : \mathbf{1}$ and $\lfloor \varpi_i \rfloor^{\sigma}_{x \mapsto \alpha_i} = \pi_i$. That is, we have a correspondence between reductions of $\mu\mathsf{CP}^{\infty}$ and internal head reductions of $\mu\mathsf{MALL}^{\infty}$. Suppose this sequence were not a maximal sequence of internal head reductions, then there exists n such that P_n is deadlocked but $\pi_n \xrightarrow{\alpha_n} \pi_{n+1}$. If that were so, then by Proposition 9 there exists a process Q such that $P_n \to Q$ contradicting the fact that P_n is deadlocked. Therefore, σ is maximal.

Now by Proposition 6, we have that σ is of finite length n and π_n is a cut-free proof of $\vdash \mathbf{1}$. Thus θ is finite and furthermore $P_n = x[]$. Since θ was arbitrary, we can conclude that P_0 is terminating, as required. □

5. Iso-recursive subtyping

We introduce here an iso-recursive subtype system [27]. It is iso-recursive in the sense that fixed points are handled as explicit operators that are part of the types, and the structure of the fixed points must align on each side of a subtype judgement. When treating types iso-recursively, a term with fixed points and its infinite unfolding are not considered to be equivalent, since the way that a type is unfolded is taken into account.

Table 7 shows the inference rules for iso-recursive subtyping judgements. The rules [BOT] and [TOP] establish that $\mathbf{0}$ and $\top$ are respectively the least and the greatest session type; the rules [REFL] and [CONG] establish reflexivity and pre-congruence of $\leqslant_{\mathsf{iso}}$ with respect to all the constants and connectives; finally, the rule [FIX] shows that we adopt an iso-recursive formulation of subtyping [27] whereby two (recursive) session types are related provided that fixed points of the same kind are found in the same locations.

Example 7. Consider the types $A \stackrel{\text{def}}{=} \mu X.\mathbf{0} \oplus (\mu X.\mathbf{1} \oplus \mathbf{0})$ and $B \stackrel{\text{def}}{=} \nu X.(\bot \,\&\, X)$ which, as we have seen in Example 2, describe the behaviour of Client and Server in Example 1. We can derive $A \leqslant_{\mathsf{iso}} B^{\bot}$ by means of the derivation

$$\dfrac{\dfrac{}{\mathbf{0} \leqslant_{\mathsf{iso}} \mathbf{1}}\ [\textsc{bot}] \qquad \dfrac{\dfrac{\dfrac{}{\mathbf{1} \leqslant_{\mathsf{iso}} \mathbf{1}}\ [\textsc{refl}] \qquad \dfrac{}{\mathbf{0} \leqslant_{\mathsf{iso}} B^{\bot}}\ [\textsc{bot}]}{\mathbf{1} \oplus \mathbf{0} \leqslant_{\mathsf{iso}} \mathbf{1} \oplus B^{\bot}}\ [\textsc{cong}]}{\mu X.\mathbf{1} \oplus \mathbf{0} \leqslant_{\mathsf{iso}} B^{\bot}}\ [\textsc{fix}]}{\mathbf{0} \oplus (\mu X.\mathbf{1} \oplus \mathbf{0}) \leqslant_{\mathsf{iso}} \mathbf{1} \oplus B^{\bot}}\ [\textsc{cong}]}{A \leqslant_{\mathsf{iso}} B^{\bot}}\ [\textsc{fix}]$$

as well as $B \leqslant_{\mathsf{iso}} A^{\bot}$ by means of the derivation

$$\dfrac{\dfrac{}{\bot \leqslant_{\mathsf{iso}} \top}\ [\textsc{top}] \qquad \dfrac{\dfrac{\dfrac{}{\bot \leqslant_{\mathsf{iso}} \bot}\ [\textsc{refl}] \qquad \dfrac{}{B \leqslant_{\mathsf{iso}} \top}\ [\textsc{top}]}{\bot \parr B \leqslant_{\mathsf{iso}} \bot \parr \top}\ [\textsc{cong}]}{B \leqslant_{\mathsf{iso}} \nu X.\bot \,\&\, \top}\ [\textsc{fix}]}{\bot \parr B \leqslant_{\mathsf{iso}} \top \,\&\, (\nu X.\bot \,\&\, \top)}\ [\textsc{cong}]}{B \leqslant_{\mathsf{iso}} A^{\bot}}\ [\textsc{fix}] \quad ⌟$$

It is easy to show that $\leqslant_{\mathsf{iso}}$ is a preorder and that $A \leqslant_{\mathsf{iso}} B$ implies $B^{\bot} \leqslant_{\mathsf{iso}} A^{\bot}$ in general. Indeed, as illustrated in Example 7, we obtain a derivation of $B^{\bot} \leqslant_{\mathsf{iso}} A^{\bot}$ from that of $A \leqslant_{\mathsf{iso}} B$ by dualizing every judgement and by turning every application of [BOT] (respectively [TOP]) into an application of [TOP] (respectively [BOT]).

We now show that our notion of subtyping is *sound* by proving the standard safe substitution principle: if "P has type A" and A is a subtype of B, then "P also has type B". We quote such informal statements since in general P may use more than one channel,

in which case the "type" of P is better described by the whole typing context in which P is well typed rather than by the type of a particular channel used by P.

Theorem 2. *If $P \vdash \Gamma, x : A$ and $A \leqslant_{\mathsf{iso}} B$, then $P \vdash \Gamma, x : B$.*

Proof. We corecursively define a function $\lceil \cdot \rceil_x^B$ that maps typing derivations for the judgement $P \vdash \Gamma, x : A$ into typing derivations for the judgement $P \vdash \Gamma, x : B$ under the hypothesis $A \leqslant_{\mathsf{iso}} B$. This function is defined by cases on the last rule applied to derive $P \vdash \Gamma, x : A$ as well as on the shape of B. We only illustrate a few representative cases, the others being analogous.

When we encounter the application of a typing rule that concerns the channel x and B happens to be $\top$, the derivation can be truncated by an application of $[\top]$:

$$\left\lceil \dfrac{\cdots}{P \vdash \Gamma, x : A}\ [*] \right\rceil_x^{\top} = \dfrac{}{P \vdash \Gamma, x : \top}\ [\top]$$

This truncation is safe because, by using $\top$ as a type for x, we know that there cannot be another process at the other end of the session (which is typed with $\mathbf{0}$). In other words, P is *dead code* that will never be executed.

When we encounter the application of a typing rule that concerns the channel x and B is not $\top$, we propagate the transformation upwards using the fact that $\leqslant_{\mathsf{iso}}$ is a congruence for all the connectives and fixed points. For example, in the particular case of $[\&]$ we have

$$\left\lceil \dfrac{\pi_1 :: P \vdash \Gamma, x : A_1 \qquad \pi_2 :: Q \vdash \Gamma, x : A_2}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : A_1 \,\&\, A_2}\ [\&] \right\rceil_x^{B_1 \& B_2} = \dfrac{\lceil \pi_1 \rceil_x^{B_1} \qquad \lceil \pi_2 \rceil_x^{B_2}}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : B_1 \,\&\, B_2}\ [\&]$$

where we use the notation $\pi :: P \vdash \Gamma$ to give name π to the derivation for the judgement $P \vdash \Gamma$. Note that from the hypothesis $A_1 \,\&\, A_2 \leqslant_{\mathsf{iso}} B_1 \,\&\, B_2$ we deduce $A_i \leqslant_{\mathsf{iso}} B_i$ for every $i = 1, 2$, thereby satisfying the requirements of the transformation in its corecursive applications.

When we encounter the application of a typing rule that concerns a channel y different from x, we simply propagate the transformation upwards while updating the type of x. Considering again the case of $[\&]$ we have:

$$\left\lceil \dfrac{\pi_1 :: P \vdash \Gamma, x : A, y : C_1 \qquad \pi_2 :: Q \vdash \Gamma, x : A, y : C_2}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : A, y : C_1 \,\&\, C_2}\ [\&] \right\rceil_x^{B} = \dfrac{\lceil \pi_1 \rceil_x^{B} \qquad \lceil \pi_2 \rceil_x^{B}}{\mathsf{case}\ x\{P, Q\} \vdash \Gamma, x : B, y : C_1 \,\&\, C_2}\ [\&]$$

The typing derivation for $P \vdash \Gamma, x : B$ resulting from the transformation is valid because every infinite branch in it corresponds to an infinite branch in the original typing derivation for $P \vdash \Gamma, x : A$ and contains the same rule applications. It may happen that the transformation *truncates* an infinite branch in the original derivation (see the first case discussed above), but then this branch becomes finite and requires no special attention. □

Example 8. From the derivations in Example 2 and Theorem 2 we deduce that the typing judgements $\mathsf{Client}\langle x \rangle \vdash x : B^{\perp}$ and $\mathsf{Server}\langle x, z \rangle \vdash x : A^{\perp}, z : \mathbf{1}$ are both derivable and that their derivations are valid. In particular, if we call π the typing derivation for $\mathsf{Server}\langle x, z \rangle \vdash x : B, z : \mathbf{1}$ and we let $C \stackrel{\mathsf{def}}{=} \nu X.\, \bot \,\&\, \top$, we have that $\lceil \pi \rceil_x^{A^{\perp}}$ is the following derivation

$$\dfrac{\dfrac{}{x().z[] \vdash x : \top, z : \mathbf{1}}\ [\top] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z[] \vdash z : \mathbf{1}}\ [\mathbf{1}]}{x().z[] \vdash x : \bot, z : \mathbf{1}}\ [\bot] \qquad \dfrac{\text{branch truncated here}}{\mathsf{Server}\langle x, z \rangle \vdash x : \top, z : \mathbf{1}}\ [\top]}{\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z \rangle\} \vdash x : \bot \,\&\, \top, z : \mathbf{1}}\ [\&]}{\mathsf{corec}\ x.\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z \rangle\} \vdash x : C, z : \mathbf{1}}\ [\nu]}{\mathsf{Server}\langle x, z \rangle \vdash x : C, z : \mathbf{1}}\ [\textsc{call}]}{\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z \rangle\} \vdash x : \top \,\&\, C, z : \mathbf{1}}\ [\&]}{\mathsf{corec}\ x.\mathsf{case}\ x\{x().z[], \mathsf{Server}\langle x, z \rangle\} \vdash x : A^{\perp}, z : \mathbf{1}}\ [\nu]}{\mathsf{Server}\langle x, z \rangle \vdash x : A^{\perp}, z : \mathbf{1}}\ [\textsc{call}]$$

which *truncates* the behaviour of Server so that it barely covers what is needed by Client. In the end we can build (at least) two different typing derivations showing that the composition $(x)(\mathsf{Client}\langle x \rangle \mid \mathsf{Server}\langle x, z \rangle)$ is well typed. ⌟

In our presentation the safe substitution principle is only a meta-theoretic property of the type system, but Theorem 2 can be used to justify the extension of the type system with an explicit subsumption rule like the following

$$[\textsc{iso-sub}]\quad \dfrac{\pi :: P \vdash \Gamma, x : A}{P \vdash \Gamma, x : B}\ A \leqslant_{\mathsf{iso}} B \quad \rightsquigarrow \quad \lceil \pi \rceil_x^B$$

Table 8
Equi-recursive subtyping for session types.

$$[\text{BOT}]\ \frac{}{\mathbf{0} \leqslant_{\text{equi}} A} \qquad [\text{TOP}]\ \frac{}{A \leqslant_{\text{equi}} \top} \qquad [\text{REFL}]\ \frac{}{\kappa \leqslant_{\text{equi}} \kappa} \qquad [\text{CONG}]\ \frac{A \leqslant_{\text{equi}} A' \qquad B \leqslant_{\text{equi}} B'}{A \star B \leqslant_{\text{equi}} A' \star B'}$$

$$[\text{LEFT-}\sigma]\ \frac{A\{\sigma X.A/X\} \leqslant_{\text{equi}} B}{\sigma X.A \leqslant_{\text{equi}} B} \qquad [\text{RIGHT-}\sigma]\ \frac{A \leqslant_{\text{equi}} B\{\sigma X.B/X\}}{A \leqslant_{\text{equi}} \sigma X.B}$$

which is interpreted as the derivation $\lceil \pi \rceil_x^B$.

6. Equi-recursive subtyping

In this section we develop an equi-recursive subtyping relation $\leqslant_{\text{equi}}$ which is more general than $\leqslant_{\text{iso}}$ and closer to the standard formulations of subtyping for recursive session types [8], but that does not satisfy the safe substitution principle (Theorem 2) in a strict sense: since processes include forms $\mathsf{rec}\ x$ and $\mathsf{corec}\ x$ that explicitly indicate the points in a typing derivation where a fixed point is unfolded, it is clear that a recursive session type and its unfolding cannot be treated as equivalent without breaking typeability. As we will see, we can still explain this coarser subtyping relation in our type system by resorting to a *coercion semantics*.

Equi-recursive subtyping is coinductively defined by the rules in Table 8. The rules [BOT], [TOP], [REFL] and [CONG] are the same as for $\leqslant_{\text{iso}}$. The difference is that in equi-recursive subtyping, fixed points can be unfolded independently by [LEFT-σ] and [RIGHT-σ]. These rules may look suspicious since they are applicable to either side of $\leqslant_{\text{equi}}$ regardless of the intuitive interpretation of μ and ν as least and greatest fixed points. In fact, if equi-recursive subtyping were solely defined by the derivability according to the rules in Table 8, the two fixed point operators would be equivalent. For example, both $\mu X.(\mathbf{1} \oplus X) \leqslant_{\text{equi}} \nu X.(\mathbf{1} \oplus X)$ and $\nu X.(\mathbf{1} \oplus X) \leqslant_{\text{equi}} \mu X.(\mathbf{1} \oplus X)$ are derivable even though only the first relation seems reasonable. We will see in Example 10 that allowing the second relation is actually *unsound*, in the sense that it compromises the termination property enjoyed by well-typed processes.

We obtain a sound equi-recursive subtyping relation by ruling out some infinite derivations as per the following (and final) definition.

Definition 6 *(Equi-recursive subtyping).* We say that A is an *equi-recursive subtype* of B if $A \leqslant_{\text{equi}} B$ is derivable and, for every infinite branch $(A_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ of the derivation, either (1) $\min\{C \mid \exists^\infty i : A_i = C\}$ is a μ-type or (2) $\min\{C \mid \exists^\infty i : B_i = C\}$ is a ν-type.

The clauses (1) and (2) of Definition 6 make sure that μ and ν are correctly interpreted as least and greatest fixed points. In particular, we expect the least fixed point to be subsumed by a greatest fixed point, but not vice versa in general. For example, consider once again the (straightforward) derivations for the aforementioned subtyping judgements $\mu X.(\mathbf{1} \oplus X) \leqslant_{\text{equi}} \nu X.(\mathbf{1} \oplus X)$ and $\nu X.(\mathbf{1} \oplus X) \leqslant_{\text{equi}} \mu X.(\mathbf{1} \oplus X)$. The first derivation satisfies both clauses (there is only one infinite branch, along which a μ-type is unfolded infinitely many times on the left hand side of $\leqslant_{\text{equi}}$ and a ν-type is unfolded infinitely many times on the right hand side of $\leqslant_{\text{equi}}$). The second derivation satisfies neither clause. Therefore, $\mu X.(\mathbf{1} \oplus X)$ is an equi-recursive subtype of $\nu X.(\mathbf{1} \oplus X)$ but not vice versa.

In both clauses of Definition 6 there is a requirement that the type of the fixed point on each side of the relation is determined by the $\preceq$-minimum of the types that appear infinitely often. This is needed to handle correctly alternating fixed points, by determining which one is actively contributing to the infinite path. To see what effect this has consider the types $A \stackrel{\text{def}}{=} \mu X.\nu Y.(\mathbf{1} \oplus X)$, $A' \stackrel{\text{def}}{=} \nu Y.(\mathbf{1} \oplus A)$, $B \stackrel{\text{def}}{=} \mu X.\mu Y.(\mathbf{1} \oplus X)$ and $B' \stackrel{\text{def}}{=} \mu Y.(\mathbf{1} \oplus B)$. Observe that A unfolds to A', A' unfolds to $\mathbf{1} \oplus A$, B unfolds to B' and B' unfolds to $\mathbf{1} \oplus B$. We have $A \leqslant_{\text{equi}} B$ despite Y is bound by a *greatest* fixed point on the left and by a *least* fixed point on the right. Indeed, both A and A' occur infinitely often in the (only) infinite branch of the derivation for $A \leqslant_{\text{equi}} B$, but $A \preceq A'$ according to the intuition that the $\preceq$-minimum type that occurs infinitely often is the one corresponding to the outermost fixed point. In this case, the outermost fixed point is μX which "overrides" the contribution of the inner fixed point νY. The interested reader may refer to the literature on $\mu\mathsf{MALL}^\infty$ [15,16] for details.

Hereafter, unless otherwise specified, we write $A \leqslant_{\text{equi}} B$ to imply that A is an equi-recursive subtype of B (according to Definition 6) and not simply that the judgement $A \leqslant_{\text{equi}} B$ is derivable.

Just like $\leqslant_{\text{iso}}$, also $\leqslant_{\text{equi}}$ is a preorder and $A \leqslant_{\text{equi}} B$ implies $B^\perp \leqslant_{\text{equi}} A^\perp$ in general. The proof of this latter property relies on the fact that the clauses in Definition 6 are dual to each other. The fact that $\leqslant_{\text{equi}}$ is a preorder is not as obvious as in the case of $\leqslant_{\text{iso}}$. In particular, if $A \leqslant_{\text{equi}} C$ and $C \leqslant_{\text{equi}} B$, it is relatively easy to build a derivation for $A \leqslant_{\text{equi}} B$, but then we also have to make sure that the two clauses of Definition 6 are satisfied. The following result (proved in Appendix A) shows that this is the case.

Theorem 3. $\leqslant_{\text{equi}}$ *is transitive.*

Table 9
Coercion semantics of equi-recursive subtyping (selected equations).

$$\left\lfloor \frac{}{\mathbf{0} \leqslant A} \right\rfloor_{x,y} \triangleq \mathsf{fail}\ x \qquad \left\lfloor \frac{}{A \leqslant \top} \right\rfloor_{x,y} \triangleq \mathsf{fail}\ y \qquad \left\lfloor \frac{}{\mathbf{1} \leqslant \mathbf{1}} \right\rfloor_{x,y} \triangleq x().y[]$$

$$\left\lfloor \frac{\pi_1 :: A \leqslant A' \qquad \pi_2 :: B \leqslant B'}{A \oplus B \leqslant A' \oplus B'} \right\rfloor_{x,y} \triangleq \mathsf{case}\ x\{y[\mathsf{in}_0].\lfloor \pi_1 \rfloor_{x,y}, y[\mathsf{in}_1].\lfloor \pi_2 \rfloor_{x,y}\}$$

$$\left\lfloor \frac{\pi_1 :: A \leqslant A' \qquad \pi_2 :: B \leqslant B'}{A \otimes B \leqslant A' \otimes B'} \right\rfloor_{x,y} \triangleq \underbrace{x(u).y[v](\lfloor \pi_1 \rfloor_{u,v} \mid \lfloor \pi_2 \rfloor_{x,y})}_{u \text{ and } v \text{ are fresh}}$$

$$\left\lfloor \frac{\pi :: A\{\mu X.A/X\} \leqslant B}{\mu X.A \leqslant B} \right\rfloor_{x,y} \triangleq \mathsf{corec}\ x.\lfloor \pi \rfloor_{x,y}$$

$$\left\lfloor \frac{\pi :: A \leqslant B\{\mu X.B/X\}}{A \leqslant \mu X.B} \right\rfloor_{x,y} \triangleq \mathsf{rec}\ y.\lfloor \pi \rfloor_{x,y}$$

There are two more properties of $\leqslant_{\mathsf{equi}}$ that is worth pointing out, namely that $\mu X.X \leqslant_{\mathsf{equi}} A$ and $A \leqslant_{\mathsf{equi}} \nu X.X$ hold for every A. In other words, $\mu X.X$ is equivalent to $\mathbf{0}$ and $\nu X.X$ is equivalent to $\top$. This is consistent with the intuition that the smallest least fixed point $\mu X.X$ is smaller than any type and the largest greatest fixed point $\nu X.X$ is larger than any type.

We now establish the soundness of $\leqslant_{\mathsf{equi}}$ by means of a *coercion semantics* [27], that is a translation function that takes a derivation π of a subtyping relation $A \leqslant_{\mathsf{equi}} B$ and generates a (well-typed) process $\lfloor \pi \rfloor_{x,y}$ that transforms (the protocol described by) A into (the protocol described by) B. The translation is parametrized by the two channels x and y on which the transformation takes place: the protocol A is "consumed" from x and "reissued" on y as a protocol B. In Table 9 we show a selection of representative cases, the remaining ones being obvious variations. Note that in general $\lfloor \pi \rfloor_{x,y}$ is (the invocation of) a recursive process.

We can now prove the following result.

Theorem 4. *If* $\pi :: A \leqslant_{\mathsf{equi}} B$, *then* $\lfloor \pi \rfloor_{x,y} \vdash x : A^{\perp}, y : B$.

Proof. The derivability of the judgement $\lfloor \pi \rfloor_{x,y} \vdash x : A^{\perp}, y : B$ follows immediately from the equations in Table 9. Concerning the validity of the resulting derivation, consider an infinite branch $\gamma \stackrel{\mathsf{def}}{=} (\lfloor \pi_i \rfloor_{x_i,y_i} \vdash x_i : A_i^{\perp}, y : B_i)_{i \in \mathbb{N}}$ in the typing derivation of the coercion where $A_0 = A$ and $B_0 = B$. This branch corresponds to an infinite branch $(A_i \leqslant_{\mathsf{equi}} B_i)_{i \in \mathbb{N}}$ in $\pi :: A \leqslant B$. According to Definition 6, either clause (1) or clause (2) holds for this branch. Suppose, without loss of generality, that clause (1) holds. Then $\min\{C \mid \exists^{\infty} i \in \mathbb{N} : A_i = C\}$ is a μ-type. According to Table 9 we have that $(x_i)_{i \in \mathbb{N}}$ is a ν-thread of γ, hence γ is a valid branch. □

Theorem 4 justifies the extension of the type system with a subsumption rule [EQUI-SUB] that is translated into an explicit application of the coercion corresponding to the subtyping relation being applied:

$$\frac{P \vdash \Gamma, x : A}{P \vdash \Gamma, x : B}\ [\text{EQUI-SUB}]\ \pi :: A \leqslant_{\mathsf{equi}} B \quad \rightsquigarrow \quad \frac{P\{y/x\} \vdash \Gamma, y : A \qquad \lfloor \pi \rfloor_{y,x} \vdash y : A^{\perp}, x : B}{(y)(P \mid \lfloor \pi \rfloor_{y,x}) \vdash \Gamma, x : B}\ [\text{CUT}]$$

This translation introduces a cut that combines P with type A and the coercion corresponding to the derivation $\pi :: A \leqslant_{\mathsf{equi}} B$ to yield a version of P with type B. The original P and the coercion are connected by a fresh channel y.

Example 9. Let $\mathsf{EquiClient}(x) \triangleq x[\mathsf{in}_1].x[\mathsf{in}_0].x[]$ be the same as $\mathsf{Client}\langle x \rangle$ from Example 1 except for the missing rec x actions. Also, let $A \stackrel{\mathsf{def}}{=} \mathbf{0} \oplus (\mathbf{1} \oplus \mathbf{0})$ and recall the type $B \stackrel{\mathsf{def}}{=} \nu X.(\perp \,\&\, X)$ from Example 2 which describes the behaviour of $\mathsf{Server}\langle x, z \rangle$ on x. It is easy to see that $\mathsf{EquiClient}\langle x \rangle \vdash x : A$, but we also have $A \not\leqslant_{\mathsf{iso}} B^{\perp}$ and indeed $\mathsf{EquiClient}\langle x \rangle$ would not be able to safely interact with $\mathsf{Server}\langle x \rangle$ because of the missing rec x actions. The following derivation

$$\dfrac{\dfrac{\dfrac{}{\mathbf{0} \leqslant_{\mathsf{equi}} \mathbf{1}}\,[\text{BOT}] \qquad \dfrac{\dfrac{\dfrac{}{\mathbf{1} \leqslant_{\mathsf{equi}} \mathbf{1}}\,[\text{REFL}] \qquad \dfrac{}{\mathbf{0} \leqslant B^{\perp}}\,[\text{BOT}]}{\mathbf{1} \oplus \mathbf{0} \leqslant_{\mathsf{equi}} \mathbf{1} \oplus B^{\perp}}\,[\text{CONG}]}{\mathbf{1} \oplus \mathbf{0} \leqslant_{\mathsf{equi}} B^{\perp}}\,[\mu]}{A \leqslant_{\mathsf{equi}} \mathbf{1} \oplus B^{\perp}}\,[\text{CONG}]}{A \leqslant_{\mathsf{equi}} B^{\perp}}\,[\mu]$$

is valid since it is finite, hence A is an equi-recursive subtype of $B^{\perp}$. This means $\mathsf{EquiClient}\langle x\rangle$ and $\mathsf{Server}\langle x, z\rangle$ can be connected by a cut through an application of [EQUI-SUB]. ⌟

Example 10. Let $\mathsf{CoChatter}(x) \triangleq \mathsf{corec}\ x.x[\mathsf{in}_1].\mathsf{CoChatter}\langle x\rangle$ be a variation of Chatter in Example 3 that unfolds the type of x by means of corec x instead of rec x. Now $\mathsf{CoChatter}\langle x\rangle \vdash x : \nu X.(\mathbf{1} \oplus X)$ is derivable and the derivation is a valid one, since the infinite branch in it contains a ν-thread. If we allowed the relation $\nu X.(\mathbf{1} \oplus X) \leqslant_{\mathsf{equi}} \mu X.(\mathbf{1} \oplus X)$ then it would be possible to obtain a well-typed composition of $\mathsf{CoChatter}\langle x\rangle$ and $\mathsf{Server}\langle x, z\rangle$, which is stuck and therefore not terminating in the sense of Definition 1. ⌟

7. Concluding remarks

We have studied iso-recursive and equi-recursive subtyping relations for session types in which $\mathbf{0}$ and $\top$ act as least and greatest elements. Despite the minimalistic look of the relations and the apparent rigidity in the syntax of types, in which the arity of internal and external choices is fixed, both relations capture the usual co/contra variance of labels thanks to the interpretation given to $\mathbf{0}$ and $\top$. Other refinement relations for session types with least and greatest elements have been studied in the past [28,12], although without an explicit correspondence with logic.

Unlike subtyping relations for session types [8,9,11,13] that only preserve *safety properties* of sessions (communication safety, protocol fidelity and deadlock freedom), our subtyping relations $\leqslant_{\mathsf{iso}}$ and $\leqslant_{\mathsf{equi}}$ also preserve termination, which is a *liveness property*. For this reason, $\leqslant_{\mathsf{iso}}$ and $\leqslant_{\mathsf{equi}}$ are somewhat related to *fair subtyping* [10,12], which preserves *fair termination* [29,30]. It appears that $\leqslant_{\mathsf{equi}}$ is coarser than fair subtyping, although the exact relationship between the two relations is difficult to characterize because of the fundamentally different ways in which recursive behaviours are represented in the syntax of types. The relation $\leqslant_{\mathsf{equi}}$ inherits least and greatest fixed points from $\mu\mathsf{MALL}^{\infty}$ [15,16], whereas fair subtyping has been studied on session type languages that either make use of general recursion [10] or that use regular trees directly [12]. A more conclusive comparison is left for future work.

Having conducted this investigation which follows closely the Caires-Pfenning-Wadler approach to session calculi and linear logic, it would be interesting to reconcile this work with established logical approaches to multi-party session types that also have a logical notion of subtyping and an infinitary proof calculus [7]. There are some obvious differences, notably the use of extensions of linear logic featuring a non-commutative connective which allows sequentiality to be treated in a manner that: (A) is more in line with traditional session calculi (see Remark 1); (B) makes use of all linear implications for subtyping; (C) restricts to a calculus where fixed-points can be equi-recursive and circularity is preserved by cut elimination. Notice that the difference (B) is as opposed to the restricted subset of implication in Sections 5 and 6, where the fragment of linear implication used for subtyping is chosen due to the fact that behavioural properties are not invariant with respect to all logical equivalences in interpretations following closely Caires-Pfenning-Wadler (e.g., neither isomorphism of types $A \otimes \mathbf{1} \multimap A$ and $A \multimap A \otimes \mathbf{1}$, nor $A \otimes B \multimap B \otimes A$ preserve behaviours, while all logical equivalences do in related work mainly thanks to using a non-commutative connective to model sequentiality). The difference (C) is an independent design choice that may be lifted or imposed in either line of work.

On reflection, however, what was not obvious to the authors before conducting this detailed investigation, is that (A), (B) and (C) above are little more than stylistic differences leading to different calculi that may be appropriate for different applications. The real difference between these systems is not really Caires-Pfenning-Wadler v.s. non-commutative logic; more so, the difference is that the current paper interprets only processes typable in environment $x : \mathbf{1}$ and avoids entirely external cut reductions and cut reductions that are not applied at the head of a proof. In contrast, in the related logical approach [7] arbitrary type environments are taken fully into account (as long as they are provable) and both internal and external bottom-most cut reductions are taken into consideration. This means that behaviours in that related work are not necessarily terminating, and instead we are concerned with interpreting richer behavioural properties (see Remark 2). In this sense, these two perspective on infinitary proof theory for session calculi via extensions of linear logic are complementary parts of a bigger picture: where this work emphasises that internal channels should not block "external communications" (hence any sequence of internal reduction will be finite before some external interaction occurs); and the related work ensures those external communications form a "good" protocol in a sense aligning strongly with the literature on multi-party session types.

Insight gleaned concerning the design of infinitary proof calculi for session calculi should be exportable to other session calculi, perhaps even independently from any logical interpretation. In particular, it would be interesting to study subtyping for *asynchronous session types* [11,13]. This can be done by adopting a suitable coercion semantics to enable buffering of messages as in simple orchestrators [31,24].

CRediT authorship contribution statement

Ross Horne: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization.
Luca Padovani: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Transitivity of equi-recursive subtyping

Lemma 1. *If $A \leqslant_{\text{equi}} C$ and $C \leqslant_{\text{equi}} B$ then there exists D such that $A \leqslant_{\text{equi}} D$ and $D \leqslant_{\text{equi}} B$ and either the derivation of $A \leqslant_{\text{equi}} D$ does not end with an application of* [RIGHT-σ] *or the derivation of $D \leqslant_{\text{equi}} B$ does not end with an application of* [LEFT-σ].

Proof. Let m be the largest number of consecutive applications of [RIGHT-σ] from the root of the derivation for $A \leqslant_{\text{equi}} C$ and n be the largest number of consecutive applications of [LEFT-σ] from the root of the derivation for $C \leqslant_{\text{equi}} B$, where $m, n \in \mathbb{N} \cup \{\infty\}$. Suppose that $m = n = \infty$. Then we can define a family $\{C_i\}_{i \in \mathbb{N}}$ of types such that $C = C_0$ and $C_i = \sigma X.C_i'$ and $C_{i+1} = C_i'\{C_i/X\}$. Now $(A \leqslant_{\text{equi}} C_i)_{i \in \mathbb{N}}$ and $(C_i \leqslant_{\text{equi}} B)_{i \in \mathbb{N}}$ are infinite branches of the derivations for $A \leqslant_{\text{equi}} C$ and $C \leqslant_{\text{equi}} B$. From Definition 6 we deduce that $\min\{D \mid \exists^\infty i : C_i = D\}$ must simultaneously be a μ-type and a ν-type, which is impossible. Let $k \stackrel{\text{def}}{=} \min\{m, n\} \in \mathbb{N}$. Then we can find the desired D by unfolding C exactly k times. □

Theorem 3. *$\leqslant_{\text{equi}}$ is transitive.*

Proof. Let $\mathcal{S} \stackrel{\text{def}}{=} \{(A, B) \mid \exists C : A \leqslant_{\text{equi}} C \wedge C \leqslant_{\text{equi}} B\}$. We apply the coinduction principle to show that $\mathcal{S} \subseteq \leqslant_{\text{equi}}$. Using the coinduction principle, it suffices to show that if $(A, B) \in \mathcal{S}$, then $A \leqslant_{\text{equi}} B$ is the conclusion of a rule in Table 7 whose premises are all in $\mathcal{S}$. Consider an arbitrary pair $(A, B) \in \mathcal{S}$. By definition of $\mathcal{S}$ there exists C such that $A \leqslant_{\text{equi}} C$ and $C \leqslant_{\text{equi}} B$. From Lemma 1 we deduce that there exists D such that $A \leqslant_{\text{equi}} D$ and $D \leqslant_{\text{equi}} B$ and either the derivation of $A \leqslant_{\text{equi}} D$ does not end with an application of [RIGHT-σ] or the derivation of $D \leqslant_{\text{equi}} B$ does not end with an application of [LEFT-σ]. We reason by cases on the last rules applied in the derivations of $A \leqslant_{\text{equi}} D$ and $D \leqslant_{\text{equi}} B$, only considering the possible ones.

- ([BOT], any rule) Then $A = \mathbf{0}$. We conclude by observing that $\mathbf{0} \leqslant_{\text{equi}} B$ is the conclusion of [BOT] which has no premises.
- (any rule, [TOP]) Dual of the previous case.
- ([LEFT-σ], any rule) Then $A = \sigma X.A'$. From [LEFT-σ] we deduce $A'\{A/X\} \leqslant_{\text{equi}} D$. We conclude by observing that $A \leqslant_{\text{equi}} B$ is the conclusion of [LEFT-σ] and that $(A'\{A/X\}, B) \in \mathcal{S}$ by definition of $\mathcal{S}$.
- (any rule, [RIGHT-σ]) Dual of the previous case.
- ([REFL], [REFL]) Then $A = D = B = \kappa$ and we conclude by observing that $A \leqslant_{\text{equi}} B$ is the conclusion of [REFL], which has no premises.
- ([CONG], [CONG]) Then $A = A_1 \star A_2$ and $D = D_1 \star D_2$ and $B = B_1 \star B_2$. From [CONG] we deduce $A_i \leqslant_{\text{equi}} D_i$ and $D_i \leqslant_{\text{equi}} B_i$ for $i = 1, 2$. We conclude by observing that $A_1 \star A_2 \leqslant_{\text{equi}} B_1 \star B_2$ is the conclusion of [CONG] and that $(A_i, B_i) \in \mathcal{S}$ by definition of $\mathcal{S}$.

To conclude the proof we have to show that the subtyping derivation $\pi :: A \leqslant_{\text{equi}} B$ that we obtain with the above construction is a valid one. To this aim, consider two valid derivations $\pi_1 :: A \leqslant_{\text{equi}} C$ and $\pi_2 :: C \leqslant_{\text{equi}} B$ and an infinite branch $(A_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ of π. Then there exist two infinite branches $(A_i \leqslant_{\text{equi}} C_i)_{i \in \mathbb{N}}$ of π_1 and $(C_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ of π_2. Since π_1 and π_2 are valid derivations, they must satisfy at least one of clauses of Definition 6. We observe that if $(A_i \leqslant_{\text{equi}} C_i)_{i \in \mathbb{N}}$ satisfies clause (1) of Definition 6, then $(A_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ satisfies the same clause as well. Dually, if $(C_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ satisfies clause (2) of Definition 6, then $(A_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ satisfies the same clause as well. So, the only remaining case is when $(A_i \leqslant_{\text{equi}} C_i)_{i \in \mathbb{N}}$ satisfies clause (2) and $(C_i \leqslant_{\text{equi}} B_i)_{i \in \mathbb{N}}$ satisfies clause (1) of Definition 6. But then $\min\{D \mid \exists^\infty i : C_i = D\}$ must be a μ-type and a ν-type at the same time, hence this case is impossible. □

References

[1] K. Honda, Types for dyadic interaction, in: E. Best (Ed.), CONCUR '93, 4th International Conference on Concurrency Theory, Proceedings, Hildesheim, Germany, August 23–26, 1993, in: Lecture Notes in Computer Science, vol. 715, Springer, 1993, pp. 509–523.

[2] K. Honda, V.T. Vasconcelos, M. Kubo, Language primitives and type discipline for structured communication-based programming, in: C. Hankin (Ed.), Programming Languages and Systems - ESOP'98, 7th European Symposium on Programming, Lisbon, Portugal, March 28 – April 4, in: Lecture Notes in Computer Science, vol. 1381, Springer, 1998, pp. 122–138.

[3] H. Hüttel, I. Lanese, V.T. Vasconcelos, L. Caires, M. Carbone, P. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H.T. Vieira, G. Zavattaro, Foundations of session types and behavioural contracts, ACM Comput. Surv. 49 (2016) 3:1–3:36, https://doi.org/10.1145/2873052.

[4] P. Wadler, Propositions as sessions, J. Funct. Program. 24 (2014) 384–418, https://doi.org/10.1017/S095679681400001X.

[5] L. Caires, F. Pfenning, B. Toninho, Linear logic propositions as session types, Math. Struct. Comput. Sci. 26 (2016) 367–423, https://doi.org/10.1017/S0960129514000218.

[6] S. Lindley, J.G. Morris, Talking bananas: structural recursion for session types, in: J. Garrigue, G. Keller, E. Sumii (Eds.), Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18–22, 2016, ACM, 2016, pp. 434–447.

[7] R. Horne, Session subtyping and multiparty compatibility using circular sequents, in: I. Konnov, L. Kovács (Eds.), 31st International Conference on Concurrency Theory, CONCUR 2020, September 1–4, 2020, Vienna, Austria (Virtual Conference), in: LIPIcs, vol. 171, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 12.

[8] S.J. Gay, M. Hole, Subtyping for session types in the pi calculus, Acta Inform. 42 (2005) 191–225, https://doi.org/10.1007/s00236-005-0177-z.

[9] G. Castagna, M. Dezani-Ciancaglini, E. Giachino, L. Padovani, Foundations of session types, in: A. Porto, F.J. López-Fraguas (Eds.), Proceedings of the 11th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, Coimbra, Portugal, ACM, September 7–9, 2009, 2009, pp. 219–230.

[10] L. Padovani, Fair subtyping for open session types, in: F.V. Fomin, R. Freivalds, M.Z. Kwiatkowska, D. Peleg (Eds.), Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Proceedings, Part II, Riga, Latvia, July 8–12, 2013, in: Lecture Notes in Computer Science, vol. 7966, Springer, 2013, pp. 373–384.

[11] D. Mostrous, N. Yoshida, Session typing and asynchronous subtyping for the higher-order π-calculus, Inf. Comput. 241 (2015) 227–263, https://doi.org/10.1016/j.ic.2015.02.002.
[12] L. Padovani, Fair subtyping for multi-party session types, Math. Struct. Comput. Sci. 26 (2016) 424–464, https://doi.org/10.1017/S096012951400022X.
[13] S. Ghilezan, J. Pantović, I. Prokić, A. Scalas, N. Yoshida, Precise subtyping for asynchronous multiparty sessions, ACM Trans. Comput. Log. 24 (2023) 1–73, https://doi.org/10.1145/3568422.
[14] S.J. Gay, Subtyping supports safe session substitution, in: S. Lindley, C. McBride, P.W. Trinder, D. Sannella (Eds.), A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday, in: Lecture Notes in Computer Science, vol. 9600, Springer, 2016, pp. 95–108.
[15] D. Baelde, A. Doumane, A. Saurin, Infinitary proof theory: the multiplicative additive case, in: J. Talbot, L. Regnier (Eds.), 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 – September 1, 2016, in: LIPIcs, vol. 62, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Marseille, France, 2016, 42.
[16] A. Doumane, On the infinitary proof theory of logics with fixed points (Théorie de la démonstration infinitaire pour les logiques à points fixes), Ph.D. thesis, Paris Diderot University, France, 2017, https://tel.archives-ouvertes.fr/tel-01676953.
[17] R. Horne, L. Padovani, A logical account of subtyping for session types, in: I. Castellani, A. Scalas (Eds.), Proceedings 14th Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software, PLACES@ETAPS 2023, Paris, France, 22 April 2023, in: EPTCS, vol. 378, 2023, pp. 26–37.
[18] N. Kobayashi, A type system for lock-free processes, Inf. Comput. 177 (2002) 122–159, https://doi.org/10.1006/inco.2002.3171.
[19] L. Padovani, Deadlock and lock freedom in the linear π-calculus, in: T.A. Henzinger, D. Miller (Eds.), Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14–18, 2014, ACM, 2014, 72.
[20] D. Baelde, A. Doumane, D. Kuperberg, A. Saurin, Bouncing threads for circular and non-wellfounded proofs: towards compositionality with circular proofs, in: C. Baier, D. Fisman (Eds.), LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2–5, 2022, ACM, 2022, 63.
[21] R. van Glabbeek, P. Höfner, Progress, justness, and fairness, ACM Comput. Surv. 52 (2019) 69, https://doi.org/10.1145/3329125.
[22] M. Carbone, F. Montesi, C. Schürmann, N. Yoshida, Multiparty session types as coherence proofs, Acta Inform. 54 (2017) 243–269, https://doi.org/10.1007/s00236-016-0285-y.
[23] F. Aschieri, F.A. Genco, Par means parallel: multiplicative linear logic proofs as concurrent functional programs, Proc. ACM Program. Lang. 4 (2019), https://doi.org/10.1145/3371086.
[24] C.S. Marco Carbone, Sonia Marin, A Logical Interpretation of Asynchronous Multiparty Compatibility, 2023.
[25] M. Coppo, M. Dezani-Ciancaglini, N. Yoshida, L. Padovani, Global progress for dynamically interleaved multiparty sessions, Math. Struct. Comput. Sci. 26 (2016) 238–302, https://doi.org/10.1017/S0960129514000188.
[26] F. Derakhshan, F. Pfenning, Circular proofs as session-typed processes: a local validity condition, Log. Methods Comput. Sci. 18 (2) (2022), https://doi.org/10.46298/lmcs-18(2:8)2022.
[27] B.C. Pierce, Types and Programming Languages, MIT Press, 2002.
[28] L. Padovani, Session types = intersection types + union types, in: E. Pimentel, B. Venneri, J.B. Wells (Eds.), Proceedings Fifth Workshop on Intersection Types and Related Systems, ITRS 2010, Edinburgh, U.K., 9th July 2010, in: EPTCS, vol. 45, 2010, pp. 71–89.
[29] O. Grumberg, N. Francez, S. Katz, Fair termination of communicating processes, in: Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing, PODC '84, Association for Computing Machinery, New York, NY, USA, 1984, pp. 254–265.
[30] N. Francez, Fairness, Monographs in Comp. Sci., Springer, 1986.
[31] L. Padovani, Contract-based discovery of web services modulo simple orchestrators, Theor. Comput. Sci. 411 (2010) 3328–3347, https://doi.org/10.1016/j.tcs.2010.05.002.