

Termination in Session-Based Concurrency via Linear Logical Relations (Extended Version)

Jorge A. Pérez¹, Luís Caires¹, Frank Pfenning², and Bernardo Toninho^{1,2}

¹ CITI and Departamento de Informática, FCT, Universidade Nova de Lisboa

² Computer Science Department, Carnegie Mellon University

Abstract. In prior work we proposed an interpretation of intuitionistic linear logic propositions as session types for concurrent processes. The type system obtained from the interpretation ensures fundamental properties of session-based typed disciplines—most notably, type preservation, session fidelity, and global progress. In this paper, we complement and strengthen these results by developing a theory of logical relations. Our development is based on, and is remarkably similar to, that for functional languages, extended to an (intuitionistic) linear type structure. A main result is that well-typed processes always terminate (strong normalization). We also introduce a notion of observational equivalence for session typed processes. As applications, we prove that all proof conversions induced by the logic interpretation actually express observational equivalences, and explain how type isomorphisms resulting from linear logic equivalences are realized by coercions between interface types of session-based concurrent systems.

1 Introduction

Modern computing systems rely heavily on the concurrent communication of distributed software artifacts. Hence, to a large extent, guaranteeing their correctness amounts to ensuring consistent dialogues between these artifacts—an extremely challenging task given the complex interaction patterns they usually feature. *Session-based concurrency* has consolidated as a foundational approach to communication correctness: dialogues between participants are structured into *sessions*, the basic units of communication; descriptions of the interaction patterns are then abstracted as *session types* [11], which are statically checked against specifications. These specifications are usually given in the π -calculus, so we obtain *processes* communicating through so-called session channels connecting exactly two subsystems. The discipline of session types ensures session protocols in which actions always occur in dual pairs: when one partner sends, the other receives; when one partner offers a selection, the other chooses; when a session terminates, no further interaction may occur. New sessions may be dynamically created by invocation of shared servers. While *concurrency* arises in the simultaneous execution of sessions, *mobility* is present in the exchange of session and server names.

In session-based concurrency, typing disciplines usually guarantee communication correctness via (forms of) *subject reduction* and *progress* properties. The former states that well-typed processes always evolve to well-typed processes (a *safety* property); the latter says that well-typed processes will never run into a stuck state (a *liveness* property). In addition to ensure that sets of interactions adhere to their prescribed behavior, it

is sensible to require such interactions to be *finite*: while from a global perspective systems are meant to run forever, at a local level we would like participants which always respond within a finite amount of time, and never engage into infinite internal computations. *Termination* (more commonly known as *strong normalization* in the functional setting) is indeed a most desirable liveness property; in session-based concurrency, it may substantially improve the correctness guarantees provided by subject reduction and progress. Ensuring termination in concurrent calculi, however, is known to be hard: in (variants of) the π -calculus, proofs require heavy constraints on the language and/or its types, often relying on ad-hoc machineries (see [8] for a survey).

In this paper, we study termination in session-based concurrency. The starting point is our interpretation of (intuitionistic) linear logic propositions as session types [4], which has provided the first purely logical account of session types. We briefly outline the main ideas underlying the interpretation. While in the linear λ -calculus types are assigned to terms (denoting functions and values), in our interpretation types are assigned to names (denoting communication channels) and describe their session protocol. More precisely, an object of type $A \multimap B$ denotes a session that first inputs a session channel of type A , and then behaves as B —another interactive behavior. An object of type $A \otimes B$ denotes a session that first sends a session channel of type A and afterwards behaves as B . The $!A$ type is interpreted as a type of a shared server for sessions of type A ; the additive product and sum are interpreted as branch and choice session type operators, respectively. The type system distinguishes two kinds of type environments: a *linear* part Δ and an *unrestricted* part Γ , where weakening and contraction principles hold for Γ but not for Δ . A type judgment is then of the form $\Gamma; \Delta \vdash P :: z:C$, with Γ, Δ , and $z:C$ having pairwise disjoint domains. We refer to $\Gamma; \Delta$ and $z:C$ as the left- and right-hand side typings, respectively. Such a judgment intuitively asserts: process P implements session C along channel z , provided it is placed in a process environment that offers the sessions specified in Γ and Δ . The classic duality of (binary) session types is retained via the multiplicative/additive nature of linear logic propositions. This way, e.g., \otimes and \multimap are dual in that *using* a session of one type (in the left-hand side typing) is equivalent to *implementing* a type of the other (in the right-hand side typing).

The interpretation establishes a tight correspondence between session types for the π -calculus and intuitionistic linear logic: typing rules correspond to dual intuitionistic linear sequent calculus proof rules and, moreover, process reduction may be simulated in a type preserving way by proof conversions and reductions, and vice versa. As a result, we obtain subject reduction from which session fidelity follows. The typing discipline ensures global progress, beyond the restricted progress on a single session property obtained in pure session type systems. This is significant, as progress in traditional session type systems is only ensured via complex technical machineries.

Our main contribution is a simple theory of logical relations for session types. The method of logical relations has proved to be extremely productive in the functional setting; in fact, termination, various forms of equivalence, parametricity can be established via logical relations. In this presentation, we use logical relations to prove termination for (binary) session types. Although our interpretation assigns types to names (and not to terms, as in the typed λ -calculus), quite remarkably, we are able to define (linear) logical relations which are truly defined on the structure of types—as in logical relations

for the typed λ -calculus [21, 22]. A salient aspect of our proof is that it closely follows from the principles of the (linear) type system. As hinted at above, this is in sharp contrast with known proofs of termination in the π -calculus. To our knowledge, ours is the first proof of termination of its kind in the context of session-based concurrency.

Certifying termination of session-typed interacting programs is very important in practice. In server-client interactions, for instance, it is critical for clients to be sure that running some piece of code provided by a server (say, code embedded in web pages of a cloud application) will not cause it to get stuck indefinitely (as in a denial-of-service attack, or just due to some bug). Furthermore, strengthening session-based type disciplines with termination guarantees should be highly beneficial for the increasingly growing number of implementations (libraries, programming language extensions) based on session types foundations—see, e.g., [12, 16, 18].

Additional contributions are two applications of the termination result, which bear witness to its complementarity with the other properties derived from the interpretation. First, we study the *proof conversions* induced by the logic interpretation. In [4] a set of such conversions was shown to correspond to either structural congruence or reduction in the π -calculus. The proof conversions we study here (not considered in [4]) cannot be explained similarly: in the process side, they induce a form of “prefix commutation” which appears rather counterintuitive. We first define a notion of *typed observational equivalence*, following the intuitive meaning of type judgements. Then, we prove *soundness* of the proof conversions with respect to the observational equivalence, i.e., processes induced by proof conversions are shown to be observationally equivalent. This result thus elegantly explains the very subtle forms of causality that arise in the (interleaved) execution of concurrent sessions. As our second application, we explain how *type isomorphisms* resulting from linear logic equivalences are realized by coercions between interface types of session-based concurrent systems. We provide a simple behavioral characterization of these isomorphisms, by relying on typed observational equivalence. Type isomorphisms can be seen as a validation of our interpretation with respect to basic linear logic principles. These applications thus clarify further the relationship between linear logic propositions and structured communications. In both cases, termination is central in proofs, intuitively because the bisimulation game can match strong transitions with weak transitions which are always finite.

Section 2 presents the process model for our framework, a synchronous π -calculus with guarded choice. Section 3 recalls the type system derived from the logical interpretation and main results from [4]. Section 4 presents linear logical relations and the termination result. Section 5 introduces a typed observational equivalence for processes. Section 6 discusses the two applications. Section 7 collects final remarks and discusses related work. Most proofs have been moved to the Appendix.

2 Process Model: Syntax and Semantics

We introduce the syntax and operational semantics of the synchronous π -calculus [20] extended with (binary) guarded choice.

Definition 2.1 (Processes). Given an infinite set Λ of names (x, y, z, u, v) , the set of processes (P, Q, R) is defined by

$$P ::= \mathbf{0} \mid P \mid Q \mid (\nu y)P \mid x\langle y \rangle.P \mid x(y).P \mid !x(y).P \\ \mid [x \leftrightarrow y] \mid x.\text{inl}; P \mid x.\text{inr}; P \mid x.\text{case}(P, Q)$$

The operators $\mathbf{0}$ (inaction), $P \mid Q$ (parallel composition), and $(\nu y)P$ (name restriction) comprise the static fragment of any π -calculus. We then have $x\langle y \rangle.P$ (send name y on x and proceeds as P), $x(y).P$ (receive a name z on x and proceed as P with the input parameter y replaced by z), and $!x(y).P$ which denotes replicated (or persistent) input. The renaming construct $[x \leftrightarrow y]$ links names x and y ; it allows a simple identity axiom in the type system [23]. The remaining three operators define a minimal labeled choice mechanism, comparable to the n -ary branching constructs found in standard session π -calculi (see, e.g., [11]). Without loss of generality we restrict our model to binary choice. In restriction $(\nu y)P$ and input $x(y).P$ the distinguished occurrence of name y is binding, with scope P . For any process P , we denote the set of *free names* of P by $fn(P)$. A process is *closed* if it does not contain free occurrences of names. We identify process up to consistent renaming of bound names, writing \equiv_α for this congruence. We write $P\{x/y\}$ for the capture-avoiding substitution of name x for name y in P . Structural congruence expresses basic identities on the structure of processes, while reduction expresses the behavior of processes.

Definition 2.2. Structural congruence $(P \equiv Q)$ is the least congruence relation on processes such that

$$\begin{array}{ll} P \mid \mathbf{0} \equiv P & P \equiv_\alpha Q \Rightarrow P \equiv Q \\ P \mid Q \equiv Q \mid P & P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\ (\nu x)\mathbf{0} \equiv \mathbf{0} & x \notin fn(P) \Rightarrow P \mid (\nu x)Q \equiv (\nu x)(P \mid Q) \\ (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P & [x \leftrightarrow y] \equiv [y \leftrightarrow x] \end{array}$$

Definition 2.3. Reduction $(P \rightarrow Q)$ is the binary relation on processes defined by:

$$\begin{array}{ll} x\langle y \rangle.Q \mid x(z).P \rightarrow Q \mid P\{y/z\} & x\langle y \rangle.Q \mid !x(z).P \rightarrow Q \mid P\{y/z\} \mid !x(z).P \\ (\nu x)([x \leftrightarrow y] \mid P) \rightarrow P\{y/x\} \quad (x \neq y) & Q \rightarrow Q' \Rightarrow P \mid Q \rightarrow P \mid Q' \\ P \rightarrow Q \Rightarrow (\nu y)P \rightarrow (\nu y)Q & P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \rightarrow Q \\ x.\text{inr}; P \mid x.\text{case}(Q, R) \rightarrow P \mid R & x.\text{inl}; P \mid x.\text{case}(Q, R) \rightarrow P \mid Q \end{array}$$

Reduction specifies the computations a process performs on its own. By definition, it is closed under structural congruence. To characterize the interactions of a process with its environment, we extend the standard early transition system for the π -calculus [20] with labels and transition rules for the choice and renaming constructs. A transition $P \xrightarrow{\alpha} Q$ denotes that process P may evolve to process Q by performing the action represented by label α . Transition labels are given by

$$\alpha ::= \overline{x\langle y \rangle} \mid x(y) \mid \overline{(\nu y)x\langle y \rangle} \mid x.\text{inl} \mid x.\text{inr} \mid \overline{x.\text{inl}} \mid \overline{x.\text{inr}} \mid \tau$$

Actions are input $x(y)$, the left/right offers $x.\text{inl}$ and $x.\text{inr}$, and their matching co-actions, respectively the output $\overline{x\langle y \rangle}$ and bound output $\overline{(\nu y)x\langle y \rangle}$ actions, and the left/

$$\begin{array}{c}
x\langle y \rangle . P \xrightarrow{\overline{x(y)}} P \text{ (out)} \quad x(y) . P \xrightarrow{x(z)} P\{z/y\} \text{ (in)} \quad (\nu x)([x \leftrightarrow y] \mid P) \xrightarrow{\tau} P\{y/x\} \text{ (id)} \\
\\
\frac{P \xrightarrow{\alpha} Q}{P \mid R \xrightarrow{\alpha} Q \mid R} \text{ (par)} \quad \frac{P \xrightarrow{\overline{\alpha}} P' \quad Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \text{ (com)} \quad \frac{P \xrightarrow{\alpha} Q}{(\nu y)P \xrightarrow{\alpha} (\nu y)Q} \text{ (res)} \\
\\
\frac{P \xrightarrow{\overline{x(y)}} Q}{(\nu y)P \xrightarrow{(\nu y)x(y)} Q} \text{ (open)} \quad \frac{P \xrightarrow{(\nu y)x(y)} P' \quad Q \xrightarrow{x(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')} \text{ (close)} \\
\\
!x(y) . P \xrightarrow{x(z)} P\{z/y\} \mid !x(y) . P \text{ (rep)} \quad x.\text{inl}; P \xrightarrow{x.\text{inl}} P \text{ (lout)} \\
x.\text{inr}; P \xrightarrow{x.\text{inr}} P \text{ (rout)} \quad x.\text{case}(P, Q) \xrightarrow{x.\text{inl}} P \text{ (lin)} \quad x.\text{case}(P, Q) \xrightarrow{x.\text{inr}} Q \text{ (rin)}
\end{array}$$

Fig. 1. π -calculus Labeled Transition System.

right selections $\overline{x.\text{inl}}$ and $\overline{x.\text{inr}}$. The bound output $(\nu y)x\langle y \rangle$ denotes extrusion of a fresh name y along (channel) x . Internal action is denoted by τ ; in general, an action α ($\overline{\alpha}$) requires a matching $\overline{\alpha}$ (α) in the environment to enable progress, as specified by the transition rules. For a label α , we define the sets $fn(\alpha)$ and $bn(\alpha)$ of free and bound names, respectively, as usual. We denote by $s(\alpha)$ the subject of α (e.g., x in $x\langle y \rangle$).

Definition 2.4 (Labeled Transition System). *The relation labeled transition ($P \xrightarrow{\alpha} Q$) is defined by the rules in Fig. 1, subject to the side conditions: in rule (res), we require $y \notin fn(\alpha)$; in rule (par), we require $bn(\alpha) \cap fn(R) = \emptyset$; in rule (close), we require $y \notin fn(Q)$. We omit the symmetric versions of rules (par), (com), and (close).*

We write $\rho_1\rho_2$ for the composition of relations ρ_1, ρ_2 . Weak transitions are defined as usual: we write \Longrightarrow for the reflexive, transitive closure of $\xrightarrow{\tau}$. Given an action $\alpha \neq \tau$, notation $\xrightarrow{\alpha}$ stands for $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$; $\xrightarrow{\tau}$ stands for \Longrightarrow . We recall some basic facts about reduction, structural congruence, and labeled transition: closure of labeled transitions under structural congruence, and coincidence of τ -labeled transition and reduction [20]: (1) if $P \equiv^{\alpha} Q$, then $P \xrightarrow{\alpha} \equiv Q$, and (2) $P \rightarrow Q$ if and only if $P \xrightarrow{\tau} \equiv Q$.

3 Session Types as Dual Intuitionistic Linear Logic Propositions

As anticipated in the introduction, the type structure coincides with intuitionistic linear logic [10, 2], omitting atomic formulas and the additive constants \top and $\mathbf{0}$.

Definition 3.1 (Types). *Types (A, B, C) are given by*

$$A, B ::= \mathbf{1} \mid !A \mid A \otimes B \mid A \multimap B \mid A \oplus B \mid A \& B$$

Types are assigned to (channel) names, and are interpreted as a form of session types; an assignment $x:A$ enforces that the process will use x according to the discipline A . $A \otimes B$ types a session channel that first performs an output (sending a session channel of type A) to its partner before proceeding as specified by B . Similarly, $A \multimap B$ types a session channel that first performs an input (receiving a session channel of type A) from its partner, before proceeding as specified by B . Type $\mathbf{1}$ means that the session

$$\begin{array}{c}
\frac{}{\Gamma; x:A \vdash [x \leftrightarrow z] :: z:A} \text{(Tid)} \quad \frac{\Gamma; \Delta \vdash P :: T}{\Gamma; \Delta, x:1 \vdash P :: T} \text{(T1L)} \quad \frac{}{\Gamma; \cdot \vdash 0 :: x:1} \text{(T1R)} \\
\frac{\Gamma; \Delta, y:A, x:B \vdash P :: T}{\Gamma; \Delta, x:A \otimes B \vdash x(y).P :: T} \text{(T}\otimes\text{L)} \quad \frac{\Gamma; \Delta \vdash P :: y:A \quad \Gamma; \Delta' \vdash Q :: x:B}{\Gamma; \Delta, \Delta' \vdash (\nu y)x(y).(P \mid Q) :: x:A \otimes B} \text{(T}\otimes\text{R)} \\
\frac{\Gamma; \Delta \vdash P :: y:A \quad \Gamma; \Delta', x:B \vdash Q :: T}{\Gamma; \Delta, \Delta', x:A \multimap B \vdash (\nu y)x(y).(P \mid Q) :: T} \text{(T}\multimap\text{L)} \quad \frac{\Gamma; \Delta, y:A \vdash P :: x:B}{\Gamma; \Delta \vdash x(y).P :: x:A \multimap B} \text{(T}\multimap\text{R)} \\
\frac{\Gamma; \Delta \vdash P :: x:A \quad \Gamma; \Delta', x:A \vdash Q :: T}{\Gamma; \Delta, \Delta' \vdash (\nu x)(P \mid Q) :: T} \text{(Tcut)} \quad \frac{\Gamma; \cdot \vdash P :: y:A \quad \Gamma, u:A; \Delta \vdash Q :: T}{\Gamma; \Delta \vdash (\nu u)(!u(y).P \mid Q) :: T} \text{(Tcut')} \\
\frac{\Gamma, u:A; \Delta, y:A \vdash P :: T}{\Gamma, u:A; \Delta \vdash (\nu y)u(y).P :: T} \text{(Tcopy)} \\
\frac{\Gamma, u:A; \Delta \vdash P\{u/x\} :: T}{\Gamma; \Delta, x:!A \vdash P :: T} \text{(T!L)} \quad \frac{\Gamma; \cdot \vdash Q :: y:A}{\Gamma; \cdot \vdash !x(y).Q :: x:!A} \text{(T!R)} \\
\frac{\Gamma; \Delta, x:A \vdash P :: T \quad \Gamma; \Delta, x:B \vdash Q :: T}{\Gamma; \Delta, x:A \oplus B \vdash x.\text{case}(P, Q) :: T} \text{(T}\oplus\text{L)} \quad \frac{\Gamma; \Delta, x:B \vdash P :: T}{\Gamma; \Delta, x:A \& B \vdash x.\text{inr}; P :: T} \text{(T}\&\text{L}_2) \\
\frac{\Gamma; \Delta \vdash P :: x:A \quad \Gamma; \Delta \vdash Q :: x:B}{\Gamma; \Delta \vdash x.\text{case}(P, Q) :: x:A \& B} \text{(T}\&\text{R)} \quad \frac{\Gamma; \Delta, x:A \vdash P :: T}{\Gamma; \Delta, x:A \& B \vdash x.\text{inl}; P :: T} \text{(T}\&\text{L}_1) \\
\frac{\Gamma; \Delta \vdash P :: x:A}{\Gamma; \Delta \vdash x.\text{inl}; P :: x:A \oplus B} \text{(T}\oplus\text{R}_1) \quad \frac{\Gamma; \Delta \vdash P :: x:B}{\Gamma; \Delta \vdash x.\text{inr}; P :: x:A \oplus B} \text{(T}\oplus\text{R}_2)
\end{array}$$

Fig. 2. The Type System πDILL .

terminated, no further interaction will take place on it; names of type **1** may still be passed around in sessions, as opaque values. $A\&B$ types a session channel that offers its partner a choice between an A behavior (“left” choice) and a B behavior (“right” choice). Dually, $A \oplus B$ types a session that either selects “left” and then proceeds as specified by A , or else selects “right”, and then proceeds as specified by B . Type $!A$ types a shared (non-linearized) channel, to be used by a server for spawning an arbitrary number of new sessions (possibly none), each one conforming to type A .

A type environment is a collection of type assignments of the form $x:A$, where x is a name and A a type, the names being pairwise disjoint. Two kinds of type environments are subject to different structural properties: a *linear* part Δ and an *unrestricted* part Γ , where weakening and contraction principles hold for Γ but not for Δ . A judgment is then of the form $\Gamma; \Delta \vdash P :: z:C$ where name declarations in Γ are always propagated unchanged to all premises in the typing rules, while name declarations in Δ are handled multiplicatively or additively, depending on the nature of the type being defined. The domains of Γ , Δ and $z:C$ are required to be pairwise disjoint. Intuitively, such a judgment asserts: P is ensured to safely provide a usage of name z according to the behavior (session) specified by type C , whenever composed with any process environment providing usages of names according to the behaviors specified by names in $\Gamma; \Delta$. As shown in [4], in our case safety ensures that the behavior is free of communication errors and deadlock. A pure client Q that just relies on external services, and does not provide any, will be typed as $\Gamma; \Delta \vdash Q :: -:1$. In general, a process

P such that $\Gamma; \Delta \vdash P :: z:C$ represents a system providing behavior C at channel z , building on “services” declared in $\Gamma; \Delta$. Of particular interest is a system typed as $\Gamma; \Delta \vdash R :: z:!A$, representing a shared server. Quite interestingly, the asymmetry induced by the intuitionistic interpretation of $!A$ enforces locality of shared names but not of linear (session names), which exactly corresponds to the intended model of sessions.

The rules of our type system πDILL are given in Fig. 2. We use T, S for right-hand side singleton environments (e.g., $z:C$). Since in $(\text{T}\otimes\text{R})$ the sent name is always fresh, our typed calculus conforms to an internal mobility discipline [3], without loss of expressiveness. Rule (Tid) defines identity in terms of the renaming construct. The composition rules (Tcut/Tcut[!]) follow the “composition plus hiding” principle [1], extended to a name passing setting. Other forms of linear typing rules for parallel composition (as in, e.g., [13]) are derivable—see [4]. As we consider π -calculus terms up to structural congruence, typability is closed under \equiv by definition. πDILL enjoys the usual properties of equivariance, weakening, and contraction in Γ . The coverage property also holds: if $\Gamma; \Delta \vdash P :: z:A$ then $\text{fn}(P) \subseteq \Gamma \cup \Delta \cup \{z\}$. In the presence of type-annotated restrictions $(\nu x:A)P$, as usual in typed π -calculi [20], type-checking is decidable.

Session type constructors thus correspond directly to intuitionistic linear logic connectives. Typing judgments correspond to sequents in dual intuitionistic linear logic, by erasing processes [2, 6]. For space reasons, below we only provide some intuitions of the correspondence between πDILL and the DILL sequent calculus; see [4] for details.

DILL is equipped with a faithful proof term assignment, so sequents have the form $\Gamma; \Delta \vdash D : C$ where Γ is the unrestricted context, Δ the linear context, C a formula (= type), and D the proof term that faithfully represents the derivation of $\Gamma; \Delta \vdash C$. Given the parallel structure of the two systems, if $\Gamma; \Delta \vdash D : A$ is derivable in DILL then there is a process P and a name z such that $\Gamma; \Delta \vdash P :: z:A$ is derivable in πDILL , and the converse also holds: if $\Gamma; \Delta \vdash P :: z:A$ is derivable in πDILL there is a derivation D that proves $\Gamma; \Delta \vdash D : A$. This correspondence is made explicit by a translation from faithful proof terms to processes: for $\Gamma; \Delta \vdash D : C$ we write \hat{D}^z for the translation of D such that $\Gamma; \Delta \vdash \hat{D}^z :: z:C$. More precisely, we have *typed extraction*: we write $\Gamma; \Delta \vdash D \rightsquigarrow P :: z:A$, meaning “proof D extracts to P ”, whenever $\Gamma; \Delta \vdash D : A$ and $\Gamma; \Delta \vdash P :: z:A$ and $P \equiv \hat{D}^z$. Typed extraction is unique up to structural congruence. As processes are related by structural and computational rules, namely those involved in the definition of \equiv and \rightarrow , derivations in DILL are related by structural and computational rules, that express certain sound proof transformations that arise in cut-elimination. Reductions generally take place when a right rule meets a left rule for the same connective, and correspond to reduction steps in the process term assignment. Similarly, structural conversions in DILL correspond to structural equivalences in the π -calculus, since they just change the order of cuts.

We now recall some results from [4] that will be useful for our developments.

Theorem 3.2 (Subject Reduction). *If $\Gamma; \Delta \vdash P :: z:A$ and $P \rightarrow Q$ then $\Gamma; \Delta \vdash Q :: z:A$.*

For any P , define *live*(P) iff $P \equiv (\nu \bar{n})(\pi.Q|R)$ for some $\pi.Q, R, \bar{n}$ where $\pi.Q$ is a *non-replicated* guarded process.

Theorem 3.3 (Progress). *If $\cdot; \cdot \vdash P :: z:1$ and *live*(P) then exists a Q such that $P \rightarrow Q$.*

4 Termination of Well-Typed Processes

A process P *terminates* (written $P \Downarrow$) if there is no infinite reduction path from P . We show that well-typed processes always terminate. The proof uses the method of logical relations, and can be summarized into two steps: (1) Definition of a logical predicate on processes, by induction on the structure of types. Processes in the predicate are terminating by definition. (2) Proof that every well-typed process is in the logical predicate. We begin stating properties of structural congruence with respect to termination.

Definition 4.1. We write $\equiv_!$ for the least congruence relation on processes which results from extending structural congruence \equiv (Def. 2.2) with the following axioms.

1. $(\nu u)(!u(z).P \mid (\nu y)(Q \mid R)) \equiv_! (\nu y)((\nu u)(!u(z).P \mid Q) \mid (\nu u)(!u(z).P \mid R))$
2. $(\nu u)(!u(y).P \mid (\nu v)(!v(z).Q \mid R))$
 $\equiv_! (\nu v)((!v(z).(\nu u)(!u(y).P \mid Q)) \mid (\nu u)(!u(y).P \mid R))$
3. $(\nu u)(!u(y).Q \mid P) \equiv_! P$ if $u \notin \text{fn}(P)$

Intuitively, (1) and (2) represent principles for the distribution of replicated servers among processes, while (3) formalizes the garbage-collection of replicated servers which cannot be invoked by any process. Notice that $\equiv_!$ was already defined in [4] (Def 4.3), and noted \simeq_s . These axioms are called the *sharpened replication axioms* and are known to express sound behavioral equivalences up to strong bisimilarity in our typed setting.

Proposition 4.2. Let P and Q be well-typed processes.

1. If $P \rightarrow P'$ and $P \equiv_! Q$ then there is Q' such that $Q \rightarrow Q'$ and $P' \equiv_! Q'$.
2. If $P \xrightarrow{\alpha} P'$ and $P \equiv_! Q$ then there is Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \equiv_! Q'$.

Proposition 4.3. If $P \Downarrow$ and $P \equiv_! Q$ then $Q \Downarrow$.

First Step: The Logical Predicate and its Closure Properties. We define a logical predicate on well-typed processes and establish a few associated closure properties. More precisely, we define a sequent-indexed family of sets of processes (process predicates) so that a set of processes $\mathcal{L}[\Gamma; \Delta \vdash T]$ enjoying certain closure properties is assigned to any sequent $\Gamma; \Delta \vdash T$. The logical predicate is defined by induction on the structure of sequents. The base case, given below, considers sequents with empty left-hand side typing, where we abbreviate $\mathcal{L}[\Gamma; \Delta \vdash T]$ by $\mathcal{L}[T]$. We write $P \not\rightarrow$ to mean that P cannot reduce; it can perform visible actions, though.

Definition 4.4 (Logical Predicate - Base case). For any type $T = z : A$ we inductively define $\mathcal{L}[T]$ as the set of all processes P such that $P \Downarrow$ and $\cdot \vdash P :: T$ and

$$\begin{aligned}
P \in \mathcal{L}[z:\mathbf{1}] & \text{ if } \forall P'. (P \Longrightarrow P' \wedge P' \not\rightarrow) \Rightarrow P' \equiv_! \mathbf{0} \\
P \in \mathcal{L}[z:A \multimap B] & \text{ if } \forall P' y. (P \xrightarrow{z(y)} P') \Rightarrow \forall Q \in \mathcal{L}[y:A]. (\nu y)(P' \mid Q) \in \mathcal{L}[z:B] \\
P \in \mathcal{L}[z:A \otimes B] & \text{ if } \forall P' y. (P \xrightarrow{(\nu y)z(y)} P') \Rightarrow \\
& \exists P_1, P_2. (P' \equiv_! P_1 \mid P_2 \wedge P_1 \in \mathcal{L}[y:A] \wedge P_2 \in \mathcal{L}[z:B]) \\
P \in \mathcal{L}[z:!A] & \text{ if } \forall P'. (P \Longrightarrow P') \Rightarrow \exists P_1. (P' \equiv_! !z(y).P_1 \wedge P_1 \in \mathcal{L}[y:A]) \\
P \in \mathcal{L}[z:A \& B] & \text{ if } (\forall P'. (P \xrightarrow{z.\text{inl}} P') \Rightarrow P' \in \mathcal{L}[z:A]) \\
& \wedge (\forall P'. (P \xrightarrow{z.\text{inr}} P') \Rightarrow P' \in \mathcal{L}[z:B]) \\
P \in \mathcal{L}[z:A \oplus B] & \text{ if } (\forall P'. (P \xrightarrow{z.\text{inl}} P') \Rightarrow P' \in \mathcal{L}[z:A]) \\
& \wedge (\forall P'. (P \xrightarrow{z.\text{inr}} P') \Rightarrow P' \in \mathcal{L}[z:B])
\end{aligned}$$

Some comments are in order. First, observe that processes in the logical predicate are terminating by definition. Also, notice that the use of $\equiv_!$ in $\mathcal{L}[z:\mathbf{1}]$ is justified by the fact that a terminated process may be well the composition of a number of replicated servers with no potential clients. Using suitable processes that “close” the derivative of the transition, in $\mathcal{L}[z:A \multimap B]$ and $\mathcal{L}[z:A \otimes B]$ we adhere to the linear logic interpretations for input and output types, respectively. In particular, in $\mathcal{L}[z:A \otimes B]$ it is worth observing how $\equiv_!$ is used to “split” the derivative of the transition, thus preserving consistency with the separate, non-interfering nature of the multiplicative conjunction. The definition of $\mathcal{L}[z:!A]$ is also rather structural, relying again on the distribution principles embodied in $\equiv_!$. The definition of $\mathcal{L}[z:A \& B]$ and $\mathcal{L}[z:A \oplus B]$ are self-explanatory.

Below, we extend the logical predicate to arbitrary typing environments. Observe how we stick to the principles of rules (Tcut) and (Tcut!) for this purpose.

Definition 4.5 (Logical Predicate - Inductive case). For any sequent $\Gamma; \Delta \vdash T$ with a non-empty left hand side environment, we define $\mathcal{L}[\Gamma; \Delta \vdash T]$ to be the set of processes inductively defined as follows:

$$\begin{aligned}
P \in \mathcal{L}[\Gamma; y : A, \Delta \vdash T] & \text{ if } \forall R \in \mathcal{L}[y : A]. (\nu y)(R \mid P) \in \mathcal{L}[\Gamma; \Delta \vdash T] \\
P \in \mathcal{L}[u : A, \Gamma; \Delta \vdash T] & \text{ if } \forall R \in \mathcal{L}[y : A]. (\nu u)(!u(y).R \mid P) \in \mathcal{L}[\Gamma; \Delta \vdash T]
\end{aligned}$$

We often rely on the following alternative characterization of the sets $\mathcal{L}[\Gamma; \Delta \vdash T]$.

Definition 4.6. Let $\Gamma = u_1:G_1, \dots, u_n:G_n$, and $\Delta = x_1:A_1, \dots, x_m:A_m$ be a non-linear and a linear typing environment, respectively. Letting $I = \{1, \dots, n\}$ and $J = \{1, \dots, m\}$, we define the sets of processes \mathcal{C}_Γ and \mathcal{C}_Δ as:

$$\mathcal{C}_\Gamma \stackrel{\text{def}}{=} \left\{ \prod_{i \in I} !u_i(y_i).R_i \mid R_i \in \mathcal{L}[y_i:G_i] \right\} \quad \mathcal{C}_\Delta \stackrel{\text{def}}{=} \left\{ \prod_{j \in J} Q_j \mid Q_j \in \mathcal{L}[x_j:A_j] \right\}$$

Because of the rôle of left-hand side typing contexts, processes in $\mathcal{C}_\Gamma, \mathcal{C}_\Delta$ are then logical representatives of the behavior specified by Γ and Δ , respectively.

Proposition 4.7. *Let Γ, Δ be a non-linear and a linear typing environment, resp. Then, for all $Q \in \mathcal{C}_\Gamma$ and for all $R \in \mathcal{C}_\Delta$, we have $Q \Downarrow$ and $R \Downarrow$. Moreover, $Q \not\rightarrow$.*

The proof of the following lemma is immediate from Definitions 4.5 and 4.6.

Lemma 4.8. *Let $\Gamma; \Delta \vdash P :: T$, with $\Gamma = u_1:G_1, \dots, u_n:G_n$ and $\Delta = x_1:A_1, \dots, x_m:A_m$. We have: $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$ iff $\forall Q \in \mathcal{C}_\Gamma, \forall R \in \mathcal{C}_\Delta, (\nu \tilde{u}, \tilde{x})(P \mid Q \mid R) \in \mathcal{L}[T]$.*

The following closure properties will be of the essence in the second step of the proof, when we will show that well-typed processes are in the logical predicate. We begin by stating closure of $\mathcal{L}[T]$ wrt substitution and structural congruence:

Proposition 4.9. *Let A be a type. If $P \in \mathcal{L}[z:A]$ then $P\{x/z\} \in \mathcal{L}[x:A]$.*

Proposition 4.10. *Let P, Q be well-typed. If $P \in \mathcal{L}[T]$ and $P \equiv Q$ then $Q \in \mathcal{L}[T]$.*

The following proposition provides a basic liveness guarantee for certain processes, based on their type.

Proposition 4.11. *Let $P \in \mathcal{L}[z:T]$ with $T \in \{A \otimes B, A \multimap B, A \oplus B, A \& B\}$. Then, there exist α, P' such that (i) $P \xrightarrow{\alpha} P'$, and (ii) if $T = A \otimes B$ then $\alpha = \overline{(\nu y)z\langle y \rangle}$; if $T = A \multimap B$ then $\alpha = z\langle y \rangle$; if $T = A \oplus B$ then $\alpha = z.\text{inr}$ or $\alpha = z.\text{inl}$; if $T = A \& B$ then $\alpha = z.\text{inr}$ or $\alpha = z.\text{inl}$.*

We now extend Proposition 4.10 so as to state closure of $\mathcal{L}[T]$ under $\equiv_!$.

Proposition 4.12. *Let P, Q be well-typed. If $P \in \mathcal{L}[T]$ and $P \equiv_! Q$ then $Q \in \mathcal{L}[T]$.*

Proof. By induction the definition of $P \equiv_! Q$. Appendix A.3, Page 22. \square

We now state *forward* and *backward* closure of the logical predicate with respect to reduction; these are typical ingredients in the method of logical relations.

Proposition 4.13 (Forward Closure). *If $P \in \mathcal{L}[T]$ and $P \rightarrow P'$ then $P' \in \mathcal{L}[T]$.*

Proposition 4.14 (Backward Closure). *If for all P_i such that $P \rightarrow P_i$ we have $P_i \in \mathcal{L}[T]$ then $P \in \mathcal{L}[T]$.*

The final closure property concerns parallel composition of processes:

Proposition 4.15 (Weakening). *Let P, Q be processes such that $P \in \mathcal{L}[T]$ and $Q \in \mathcal{L}[-:1]$. Then, $P \mid Q \in \mathcal{L}[T]$.*

Proof. By induction on the structure of type T . See Appendix A.4, Page 24. \square

Second Step: Well-typed Processes are in the Logical Predicate. We now prove that well-typed processes are in the logical predicate. Because of the definition of the logical predicate, termination of well-typed processes then follows as an immediate consequence of this fact (Theorem 4.17).

Lemma 4.16. *Let P be a process. If $\Gamma; \Delta \vdash P :: T$ then $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$.*

Proof. [Sketch] By induction on the derivation of $\Gamma; \Delta \vdash P :: T$, with a case analysis on the last typing rule used. Thus, we have 18 cases to check. In all cases, we appeal to Lemma 4.8 and show that every $M = (\nu \tilde{u}, \tilde{x})(P \mid G \mid D)$ with $G \in \mathcal{C}_\Gamma$ and $D \in \mathcal{C}_\Delta$, is in $\mathcal{L}[T]$. In case (Tid), the proof uses Proposition 4.9 (closure wrt substitution) and Proposition 4.14 (backward closure). In cases (T \otimes L), (T \multimap L), (Tcopy), (T \oplus L), (T $\&$ L₁), and (T $\&$ L₂), the proof proceeds in two steps: first, relying on Proposition 4.13 (forward closure) we show that every M'' such that $M \Longrightarrow M''$ is in $\mathcal{L}[T]$; then, we use this result in combination with Proposition 4.14 (backward closure) to conclude that $M \in \mathcal{L}[T]$. In cases (T1R), (T \otimes R), (T \multimap R), (T!R), (T \oplus R₁), and (T \oplus R₂), the proof consists in showing that M conforms to some specific case of Definition 4.4. Case (T1L) uses Proposition 4.15 (weakening). Cases (T \otimes L), (T \multimap L), (T \oplus L), and (T $\&$ L₁), use the liveness guarantee given by Proposition 4.11. Cases (Tcopy), (T!L) and (Tcut[!]) use Proposition 4.10 (closure under \equiv). Cases (Tcut), (T \multimap R), and (T!R) use Proposition 4.12 (closure under $\equiv_!$). See Appendix A.5, Page 21. \square

We now state the main result of this section: well-typed processes terminate.

Theorem 4.17 (Termination). *If $\Gamma; \Delta \vdash P :: T$ then $P \Downarrow$.*

Proof. Follows from previously proven facts. By assumption, we have $\Gamma; \Delta \vdash P :: T$. Using this and Lemma 4.16 we obtain $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$. Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta$: combining $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$ and Lemma 4.8 gives us $(\nu \tilde{u}, \tilde{x})(P \mid G \mid D) \in \mathcal{L}[T]$. By using this, together with Definition 4.4, we infer $(\nu \tilde{u}, \tilde{x})(P \mid G \mid D) \Downarrow$. Since Proposition 4.7 ensures that $G \Downarrow$ and $D \Downarrow$, this latter result allows us to conclude $P \Downarrow$. \square

5 An Observational Equivalence for Typed Processes

Here we introduce *typed context bisimilarity*, an observational equivalence over typed processes. It is defined contextually, as a binary relation indexed over sequents. Roughly, typed context bisimilarity equates two processes if, once coupled with all of their requirements (as described by the left-hand side typing), they perform the same actions (as described by the right-hand side typing). To formalize this intuition, we rely on a combination of inductive and coinductive arguments. The base case of the definition covers the cases in which the left-hand side typing environment is empty (i.e., the process requires nothing from its context to execute): the bisimulation game is then defined by induction on the structure of the (right-hand side) type, following the expected behavior in each case. The inductive case covers the cases in which the left-hand side typing environment is not empty: the tested processes are put in parallel with processes implementing the behaviors described in the left-hand side typing.

Below, we use \mathcal{S} to range over sequents of the form $\Gamma; \Delta \vdash T$. In the following, we write $\vdash T$ to stand for $\cdot; \cdot \vdash T$. The definition of typed context bisimilarity relies on *type-respecting* relations, which are indexed by sequents \mathcal{S} .

Definition 5.1 (Type-respecting relations). *A type-respecting binary relation over processes, written $\{\mathcal{R}_\mathcal{S}\}_\mathcal{S}$, is defined as a family of relations over processes indexed by \mathcal{S} . We often write \mathcal{R} to refer to the whole family. We write $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$ to mean that (i) $\Gamma; \Delta \vdash P :: T$ and $\Gamma; \Delta \vdash Q :: T$, and (ii) $(P, Q) \in \mathcal{R}_{\Gamma; \Delta \vdash T}$.*

Definition 5.2 (Typed Context Bisimilarity). A symmetric type-respecting binary relation over processes \mathcal{R} is a typed context bisimulation if

Base Cases

Tau $\vdash P \mathcal{R} Q :: T$ implies that for all P' such that $P \xrightarrow{\tau} P'$, there exists a Q' such that $Q \Longrightarrow Q'$ and $\vdash P' \mathcal{R} Q' :: T$

Input $\vdash P \mathcal{R} Q :: x:A \multimap B$ implies that for all P' such that $P \xrightarrow{x(y)} P'$, there exists a Q' such that $Q \xrightarrow{x(y)} Q'$ and for all R such that $\vdash R :: y:A$,
 $\vdash (\nu y)(P' \mid R) \mathcal{R} (\nu y)(Q' \mid R) :: x:B$.

Output $\vdash P \mathcal{R} Q :: x:A \otimes B$ implies that for all P' such that $P \xrightarrow{(\nu y)x(y)} P'$, there exists a Q' such that $Q \xrightarrow{(\nu y)x(y)} Q'$ and for all R such that $\cdot; y:A \vdash R :: \neg \mathbf{1}$,
 $\vdash (\nu y)(P' \mid R) \mathcal{R} (\nu y)(Q' \mid R) :: x:B$.

Rep Input $\vdash P \mathcal{R} Q :: x:!A$ implies that for all P' such that $P \xrightarrow{x(z)} P'$, there exists a Q' such that $Q \xrightarrow{x(z)} Q'$ and, for all R such that $\cdot; y:A \vdash R :: \neg \mathbf{1}$,
 $\vdash (\nu z)(P' \mid R) \mathcal{R} (\nu z)(Q' \mid R) :: x:!A$.

Choice $\vdash P \mathcal{R} Q :: x:A \& B$ implies both:

- If $P \xrightarrow{x.inl} P'$ then $\vdash P' \mathcal{R} Q' :: x:A$, for some Q' such that $Q \xrightarrow{x.inl} Q'$; and
- If $P \xrightarrow{x.inr} P'$ then $\vdash P' \mathcal{R} Q' :: x:B$, for some Q' such that $Q \xrightarrow{x.inr} Q'$.

Selection $\vdash P \mathcal{R} Q :: x:A \oplus B$ implies both:

- If $P \xrightarrow{x.inl} P'$ then $\vdash P' \mathcal{R} Q' :: x:A$ for some Q' such that $Q \xrightarrow{x.inl} Q'$; and
- If $P \xrightarrow{x.inr} P'$ then $\vdash P' \mathcal{R} Q' :: x:B$ for some Q' such that $Q \xrightarrow{x.inr} Q'$.

Inductive Cases

Linear Names $\Gamma; \Delta, y:A \vdash P \mathcal{R} Q :: T$ implies that

for all R such that $\vdash R :: y:A$, then $\Gamma; \Delta \vdash (\nu y)(P \mid R) \mathcal{R} (\nu y)(Q \mid R) :: T$.

Shared Names $\Gamma, u:A; \Delta \vdash P \mathcal{R} Q :: T$ implies that for all R such that $\vdash R :: z:A$, then $\Gamma; \Delta \vdash (\nu u)(!u(z).R \mid P) \mathcal{R} (\nu u)(!u(z).R \mid Q) :: T$.

We write \approx for the union of all typed context bisimulations, and call it typed context bisimilarity.

Observe how in all cases, a strong action is matched with a weak transition. In proofs, we shall exploit the fact that by virtue of Theorem 4.17 such a weak transition is always finite. In the base case, the clauses for input, output, and replication decree the closure of the tested processes with a process R that “complements” the continuation of the tested behavior; observe the very similar treatment for output and replication (where R depends on some behavior), and contrast it with that for input (where R provides the behavior). Also, notice how all clauses but that for replication are defined coinductively for the tested processes (in the sense that closed evolutions should be in the relation), but inductively on the type indexing the relation—the clause for replication may be thus considered as the only fully coinductive one. Also worth noticing is how the closures defined in such clauses (and those defined by the clauses in the inductive case) follow closely the spirit of Tcut/Tcut¹ rules in the type system.

Definition 5.2 immediately suggests a proof technique for showing that two processes are typed context bisimilar. First, close the processes with representatives of their

context, applying repeatedly the inductive case until the left-hand side typing is empty. Then, follow the usual co-inductive proof technique, and show a typed-respecting relation containing the processes obtained in the first step. Here we develop some auxiliary results to realize these intuitions.

We use K, K' to range over *contexts*, i.e., processes with a hole $[\cdot]$. In particular, we use parallel contexts: contexts in which the hole can only occur in parallel.

Definition 5.3. Let Γ and Δ be non-empty typing environments. The set of parallel contexts $\mathcal{K}_{\Gamma;\Delta}$ is defined by induction on the typing environments as follows:

$$\begin{aligned} K \in \mathcal{K}_{\emptyset;\emptyset} & \text{ if } K = [\cdot] \\ K \in \mathcal{K}_{\Gamma;u:G;\Delta} & \text{ if } K \equiv (\nu u)(K' \mid !u(y).R) \quad \text{for some } K' \in \mathcal{K}_{\Gamma;\Delta} \text{ and } \vdash R :: y:G \\ K \in \mathcal{K}_{\Gamma;\Delta,x:C} & \text{ if } K \equiv (\nu x)(K' \mid S) \quad \text{for some } K' \in \mathcal{K}_{\Gamma;\Delta} \text{ and } \vdash S :: x:C \end{aligned}$$

Proposition 5.4. Let $\Gamma = u_1:G_1, \dots, u_n:G_n$ and $\Delta = x_1:A_1, \dots, x_m:A_m$ be typing environments. Letting $I = \{1, \dots, n\}$ and $J = \{1, \dots, m\}$, we say that $K \in \mathcal{K}_{\Gamma;\Delta}$ if

$$K \equiv (\nu \tilde{u}, \tilde{x})([\cdot] \mid \prod_{i \in I} !u_i(y_i).R_i \mid \prod_{j \in J} S_j) \quad \text{with } \vdash R_i :: y_i:G_i \text{ and } \vdash S_j :: x_j:C_j$$

The following proposition allows us to move from an (inductive) proof under non-empty typing environments Γ, Δ to a (coinductive) proof under empty environments, with pairs of processes within parallel contexts $\mathcal{K}_{\Gamma;\Delta}$.

Proposition 5.5. $\Gamma; \Delta \vdash P \approx Q :: T$ implies $\vdash K[P] \approx K[Q] :: T$, for every parallel context $K \in \mathcal{K}_{\Gamma;\Delta}$.

Proof. By induction on the size of Γ and Δ ; see Appendix B.1 in Page 34. \square

Definition 5.6. A type-respecting relation \mathcal{R} is an equivalence if it has the following three properties: (i) Reflexivity: $\Gamma; \Delta \vdash P :: T$ implies $\Gamma; \Delta \vdash P \mathcal{R} P :: T$; (ii) Symmetry: $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta \vdash Q \mathcal{R} P :: T$; (iii) Transitivity: $\Gamma; \Delta \vdash P \mathcal{R} P' :: T$ and $\Gamma; \Delta \vdash P' \mathcal{R} Q :: T$ imply $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$.

Proposition 5.7. \approx is an equivalence relation.

In our setting, a notion of congruence for type-respecting relations turns out to be quite type-directed: both right- and left-hand side typings are quite explicit on the compositionality properties of processes. Defining such a notion is relatively straightforward: unsurprisingly, it mirrors the structure of the typing rules. For space reasons, we elide the details; see Appendix B.2 (Page 34) for the definition and proof that \approx is indeed a congruence.

6 Applications

We use typed context bisimilarity—together with termination, subject reduction, and progress results—to clarify two issues derived from the logical interpretation: soundness of proof conversions and observational characterizations of type isomorphisms.

$$\begin{aligned}
& (\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F})) \simeq_c (\nu y)z\langle y \rangle.((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F}) \\
& \quad (\nu x)(\hat{D} \mid y(z).\hat{E}) \simeq_c y(z).(\nu x)(\hat{D} \mid \hat{E}) \\
& \quad (\nu x)(\hat{D} \mid y.\mathbf{inl}; \hat{E}) \simeq_c y.\mathbf{inl}; (\nu x)(\hat{D} \mid \hat{E}) \\
& \quad (\nu x)(\hat{D} \mid (\nu y)u\langle y \rangle.\hat{E}) \simeq_c (\nu y)u\langle y \rangle.(\nu x)(\hat{D} \mid \hat{E}) \\
& \quad (\nu x)(\hat{D} \mid y.\mathbf{case}(\hat{E}, \hat{F})) \simeq_c y.\mathbf{case}((\nu x)(\hat{D} \mid \hat{E}), (\nu x)(\hat{D} \mid \hat{F})) \\
& \quad (\nu u)((!u(y).\hat{D}) \mid \mathbf{0}) \simeq_c \mathbf{0} \\
& (\nu u)((!u(y).\hat{D}) \mid (\nu z)(x\langle z \rangle.(\hat{E} \mid \hat{F}))) \simeq_c (\nu z)(x\langle z \rangle.((\nu u)((!u(y).\hat{D}) \mid \hat{E}) \mid (\nu u)((!u(y).\hat{D}) \mid \hat{F}))) \\
& \quad (\nu u)((!u(y).\hat{D}) \mid y(z).\hat{E}) \simeq_c y(z).(\nu u)((!u(y).\hat{D}) \mid \hat{E}) \\
& \quad (\nu u)((!u(z).\hat{D}) \mid y.\mathbf{inl}; \hat{E}) \simeq_c y.\mathbf{inl}; (\nu u)((!u(z).\hat{D}) \mid \hat{E}) \\
& \quad (\nu u)((!u(z).\hat{D}) \mid y.\mathbf{case}(\hat{E}, \hat{F})) \simeq_c y.\mathbf{case}((\nu u)((!u(z).\hat{D}) \mid \hat{E}), (\nu u)((!u(z).\hat{D}) \mid \hat{F})) \\
& \quad (\nu u)((!u(y).\hat{D}) \mid !x(z).\hat{E}) \simeq_c !x(z).(\nu u)((!u(y).\hat{D}) \mid \hat{E}) \\
& \quad (\nu u)((!u(y).\hat{D}) \mid (\nu y)v\langle y \rangle.\hat{E}) \simeq_c (\nu y)v\langle y \rangle.(\nu u)((!u(y).\hat{D}) \mid \hat{E}) \\
& (\nu w)z\langle w \rangle.(R \mid (\nu y)x\langle y \rangle.(P \mid Q)) \simeq_c (\nu y)x\langle y \rangle.(P \mid (\nu w)z\langle w \rangle.(R \mid Q)) \\
& \quad x(y).z(w).P \simeq_c z(w).x(y).P
\end{aligned}$$

Fig. 3. A sample of process equalities induced by proof conversions

Soundness of Proof Conversions. Derivations in DILL are related by structural and computational rules that express sound proof transformations that arise in cut-elimination. As mentioned in Section 3 (and fully detailed in [4]), in our interpretation reductions and structural conversions in DILL correspond to reductions and structural congruence in the π -calculus. There is, however, a group of conversions in DILL not considered in [4] and which do not correspond to neither reduction or structural congruence in the process side. We call them *proof conversions*: they induce a congruence on typed processes, denoted \simeq_c . In this section, we show *soundness* of \simeq_c with respect to \approx , that is, processes extracted from proof conversions are typed contextually bisimilar.

We illustrate the proof conversions and their associated π -calculus processes; Fig. 3 presents a sample of process equalities extracted from them. Each equality $M \simeq_c N$ is associated to appropriate right- and left-hand side typings; this way, e.g., the last equality in Fig. 3—associated to two applications of rule (T \otimes L)—could be stated as

$$\cdot; x:A \otimes B, z:C \otimes D \vdash x(y).z(w).P \simeq_c z(w).x(y).P :: T$$

where A, B, C, D are types and T is a right-hand side typing. For the sake of illustration, however, in Fig. 3 these typings are elided, as we would like to stress on the consequences of conversions on the process side. Proof conversions describe the interplay of two rules in a type-preserving way: regardless of the order in which the rules are applied, they lead to typing derivations with the same right- and left-hand side typings, but with syntactically different processes. We consider two kinds of proof conversions. The first kind captures the interplay of left/right rules with Tcut/Tcut[!] rules; the first twelve rows in Fig. 3 are examples (the first five involve (Tcut), the other seven involve (Tcut[!])). The second kind captures the interplay of left and right rules with each other; typically they describe type preserving transformations which commute actions from non-interfering sessions inside a process (the last two rows in Fig. 3 are examples).

Let us comment on the fifth process equality in Fig. 3. It corresponds to the interplay of rules (Tcut) and (T \oplus L), under typing assumptions $\Gamma; \Delta_1 \vdash \hat{D} :: x:C,$

$$\Gamma; \Delta_2, y:A, x:C \vdash \hat{E}::T, \text{ and } \Gamma; \Delta_2, y:A, x:C \vdash \hat{F}::T. \text{ Letting } \Delta = \Delta_1, \Delta_2, \text{ we have:}$$

$$\Gamma, \Delta, y:A \oplus B \vdash \underbrace{(\nu x)(\hat{D} \mid y.\text{case}(\hat{E}, \hat{F}))}_{(1)} \simeq_c \underbrace{y.\text{case}((\nu x)(\hat{D} \mid \hat{E}), (\nu x)(\hat{D} \mid \hat{F}))}_{(2)} :: T$$

with types T, A, B , and C , linear environments Δ_1, Δ_2 , and non-linear environment Γ .

Read from (1) to (2), this conversion can be interpreted as the “promotion” of the choice at y , which causes \hat{D} to get “delayed” as a result. However, such a delay is seen to be only apparent once we examine the individual typing of \hat{D} and the whole typing derivation. The first typing assumption says that \hat{D} is able to offer behavior C at x (a free name in \hat{D}), as long as it is placed in a context in which the behaviors described by names in Γ, Δ_1 are available. The left-hand side typing for both processes says that they can offer some behavior T , as long as behaviors described by names in Γ, Δ and behavior $A \oplus B$ at y are provided. Crucially, since x is private to (1), type T cannot correspond to $x:C$. That is, even if \hat{D} is at the top-level in (1) its behavior is not immediately available. Also because of the left-hand side typing, we know that (1) and (2) are only able to interact with some selection at y ; only then, \hat{D} will be able to interact with either \hat{E} or \hat{F} , whose behavior depends on the presence of behavior C at x . A conversion of (1) into (2) could be seen as a “behavioral optimization” if one considers that (2) has only one available prefix, while (1) has two parallel components.

For all proof conversions, the apparent phenomenon of “prefix promotion” induced by proof conversions can be explained along the above lines. In our soundness result (Theorem 6.2 below), the crucial point is capturing the fact that some top-level processes may not be able to *immediately* exercise their behavior (cf. \hat{D} in (1) above). The proof uses some notations on type-respecting relations. $\mathcal{I}_{\Gamma; \Delta \vdash T}$ stands for the relation $\{(P, Q) : \Gamma; \Delta \vdash P :: T, \Gamma; \Delta \vdash Q :: T\}$ which collects pairs of processes with identical left- and right-hand side typings. Based on the logical interpretation of types, we introduce a notion of “continuation relation” for pairs of typed processes:

Definition 6.1. Using \boxtimes to range over \otimes, \multimap and \boxplus to range over $\oplus, \&$, we define the type-respecting relation $\mathcal{W}_{\vdash x:A}$ by induction on the right-hand side typing, as follows:

$$\begin{aligned} \mathcal{W}_{\vdash x:1} &= \mathcal{I}_{\vdash x:1} & \mathcal{W}_{\vdash x:A \boxtimes B} &= \mathcal{I}_{\vdash x:B} \cup \mathcal{W}_{\vdash x:B} \\ \mathcal{W}_{\vdash x:!A} &= \mathcal{I}_{\vdash x:!A} & \mathcal{W}_{\vdash x:A \boxplus B} &= \mathcal{I}_{\vdash x:A} \cup \mathcal{W}_{\vdash x:A} \cup \mathcal{I}_{\vdash x:B} \cup \mathcal{W}_{\vdash x:B} \end{aligned}$$

This way, e.g., the continuation relation for $x:A \otimes B$ is $\mathcal{I}_{\vdash x:B} \cup \mathcal{W}_{\vdash x:B}$: it contains all pairs typed by $\vdash x:B$ (as processes of type $x:A \otimes B$ are to be typed by $x:B$ after the output action) as well as those pairs in the continuation relation for $x:B$.

Theorem 6.2 (Soundness of Proof Conversions). Let P, Q be processes such that (i) $\Gamma; \Delta \vdash D \rightsquigarrow P :: T$; (ii) $\Gamma; \Delta \vdash E \rightsquigarrow Q :: T$; (iii) $P \simeq_c Q$. Then, $\Gamma; \Delta \vdash P \approx Q :: T$

Proof. By coinduction, exhibiting appropriate typed context bisimulations for each proof conversion. In the bisimulation game, we exploit termination of well-typed processes (Theorem 4.17) to ensure that actions can be matched with finite weak transitions, and subject reduction (Theorem 3.2) to ensure type preservation under reductions.

We detail the case for the first proof conversion in Fig. 3—see Appendix C.2 (Page 37) for other cases. It corresponds to the interplay of rules (T \otimes R) and (Tcut). We have to show that $\Gamma; \Delta \vdash M \approx N :: z:A \otimes B$ where

$$\begin{aligned} \Gamma; \Delta_1 \vdash \hat{D} :: x:C \quad \Gamma; \Delta_2, x:C \vdash \hat{E} :: y:A \quad \Gamma; \Delta_3 \vdash \hat{F} :: z:B \quad \Delta = \Delta_1, \Delta_2, \Delta_3 \quad (1) \\ M = (\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F})) \quad N = (\nu y)z\langle y \rangle.((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F}) \end{aligned}$$

By virtue of Prop. 5.5, we have to show that for every $K \in \mathcal{K}_{\Gamma; \Delta}$, we have $\vdash K[M] \approx K[N] :: z:A \otimes B$. In turn, this implies exhibiting a typed context bisimilarity \mathcal{R} containing the pair $(K[M], K[N])$. We thus define $\mathcal{R} = \mathcal{W}_{\vdash z:A \otimes B} \cup \mathcal{S} \cup \mathcal{S}^{-1}$, with

$$\mathcal{S} = \{(K_1[M'], K_2[N]) : M \Longrightarrow M', K_1, K_2 \in \mathcal{K}_{\Gamma; \Delta}\}$$

and $\mathcal{W}_{\vdash z:A \otimes B}$ is as in Definition 6.1. Notice that \mathcal{S} is a type-respecting relation indexed by $\vdash z:A \otimes B$. In fact, using the typings in (2)—with $\Gamma = \Delta = \emptyset$ —and exploiting subject reduction (Theorem 3.2), it can be checked that for all $(P, Q) \in \mathcal{S}$ both $\vdash P :: z:A \otimes B$ and $\vdash Q :: z:A \otimes B$ can be derived.

We now show that \mathcal{R} is a typed context bisimilarity. Pick any $K \in \mathcal{K}_{\Gamma; \Delta}$. Using Proposition 5.4, we can assume $K = (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid [\cdot])$ where

- $K_\Gamma \equiv \prod_{i \in I} !u_i(y_i).R_i$, with $\vdash R_i :: y_i:D_i$, for every $u_i:D_i \in \Gamma$;
- $K_\Delta \equiv \prod_{j \in J} S_j$, with $\vdash S_j :: x_j:C_j$, for every $x_j:C_j \in \Delta$.

Clearly, $(K[M], K[N]) \in \mathcal{S}$, and so it is in \mathcal{R} . Now, suppose $K[M]$ moves first: $K[M] \xrightarrow{\alpha} M_1^*$. We have to find a matching action α from $K[N]$, i.e., $K[N] \xrightarrow{\alpha} N_1^*$. Since $\vdash K[M] :: z:A \otimes B$, we have two possible cases for α :

1. Case $\alpha = \tau$. We consider the possibilities for the origin of the reduction:
 - (a) $K_\Gamma \xrightarrow{\tau} K'_\Gamma$ and $K[M] \xrightarrow{\tau} K'[M]$. However, this cannot be the case, as by construction K_Γ corresponds to the parallel composition of input-guarded replicated processes which cannot evolve on their own.
 - (b) $K_\Delta \xrightarrow{\tau} K'_\Delta$ and $K[M] \xrightarrow{\tau} K'[M]$. Then, for some $l \in J$, $S_l \xrightarrow{\tau} S'_l$:

$$K[M] \xrightarrow{\tau} (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K'_\Delta \mid M) = K'[M] = M_1^*$$

Now, context K is the same in $K[N]$. Then K_Δ occurs identically in $K[N]$, and this reduction can be matched by a finite weak transition:

$$K[N] \Longrightarrow (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K''_\Delta \mid N) = K''[N] = N_1^*$$

By subject reduction (Theorem 3.2), $\vdash S'_l :: x_l:C_l$; hence, K', K'' are in $\mathcal{K}_{\Gamma; \Delta}$. Hence, the pair $(K'[M], K''[N])$ is in \mathcal{S} (as $M \Longrightarrow M$) and so it is in \mathcal{R} .

- (c) $M \xrightarrow{\tau} M'$ and $K[M] \xrightarrow{\tau} K[M']$. Since $M = (\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F}))$, the only possibility is that there is a \hat{D}_1 such that $\hat{D} \xrightarrow{\tau} \hat{D}_1$ and $M' = (\nu x)(\hat{D}_1 \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F}))$. This way,

$$K[M] \xrightarrow{\tau} (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid M') = K[M'] = M_1^*$$

We observe that $K[N]$ cannot match this action, but $K[N] \Longrightarrow K[N]$ is a valid weak transition. Hence, $N_1^* = K[N]$. By subject reduction (Theorem 3.2), we infer that $\vdash K[M'] :: z:A \otimes B$. We use this fact to observe that the pair $(K[M'], K[N])$ is included in \mathcal{S} . Hence, it is in \mathcal{R} .

- (d) There is an interaction between M and K_Γ or between M and K_Δ : this is only possible by the interaction of \hat{D} with K_Γ or K_Δ on names in \tilde{u}, \tilde{x} . Again, the only possible weak transition from $K[N]$ matching this reduction is $K[N] \Longrightarrow K[N]$, and the analysis proceeds as in the previous case.
2. Case $\alpha \neq \tau$. Then the only possibility, starting from $K[M]$, is an output action of the form $\alpha = (\nu y)z\langle y \rangle$. This action can only originate in M :

$$K[M] \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{x}, \tilde{u})(K_\Gamma \mid K_\Delta \mid (\nu x)(\hat{D} \mid (\nu y)(\hat{E} \mid \hat{F}))) = M_1^*$$

Process $K[N]$ can match this action via the following finite (weak) transition:

$$K[N] \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{x}, \tilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D}' \mid \hat{E}') \mid \hat{F}')) = N_1^*$$

Observe how N_1^* reflects the changes in $K[N]$ due to the possible reductions before and after the output action. By definition of \approx (output case), we consider the composition of M_1^* and N_1^* with any V such that $y:A \vdash V :: -:1$. Using the typings in (2) and subject reduction (Theorem 3.2), we infer both

$$\begin{aligned} \vdash M_2^* &= (\nu \tilde{x}, \tilde{u})(K_\Gamma \mid K_\Delta \mid (\nu x)(\hat{D} \mid (\nu y)(\hat{E} \mid V \mid \hat{F}))) :: z:B \\ \vdash N_2^* &= (\nu \tilde{x}, \tilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D}' \mid \hat{E}' \mid V) \mid \hat{F}')) :: z:B \end{aligned}$$

Hence, the pair (M_2^*, N_2^*) is in $\mathcal{W}_{\vdash z:A \otimes B}$ and so it is in \mathcal{R} .

Now suppose that $K[N]$ moves first: $K[N] \xrightarrow{\alpha} N_1^*$. We have to find a matching action α from $K[M]$: $K[M] \xrightarrow{\alpha} M_1^*$. Similarly as before, there are two cases: either $\alpha = \tau$ or $\alpha = (\nu y)z\langle y \rangle$. The former is as detailed before; the only difference is that reductions from $K[N]$ can only be originated in K_Δ ; these are matched by $K[M]$ with finite weak transitions originating in both K and in M . We thus obtain pairs of processes in \mathcal{S}^{-1} . The analysis for the case for output mirrors the given above and is omitted. \square

Type Isomorphisms. In type theory, types A and B are called *isomorphic* if there are morphisms (proofs in our case) π_A of $B \vdash A$ and π_B of $A \vdash B$ which compose to the identity in both ways—see, e.g., [9]. We adapt this notion to our setting, using typed context bisimilarity to account for *isomorphisms* in linear logic. (Below, we write $P^{\langle \tilde{x} \rangle}$ for a process parametric on a sequence of names x_1, \dots, x_n .)

Definition 6.3 (Isomorphism). *Two types A and B are called isomorphic, noted $A \simeq B$, if, for any names x, y, z , there exist processes $P^{\langle x, y \rangle}$ and $Q^{\langle y, x \rangle}$ such that:*

- (i) $\cdot; x:A \vdash P^{\langle x, y \rangle} :: y:B$; (ii) $\cdot; y:B \vdash Q^{\langle y, x \rangle} :: x:A$;
- (iii) $\cdot; x:A \vdash (\nu y)(P^{\langle x, y \rangle} \mid Q^{\langle y, z \rangle}) \approx [x \leftrightarrow z] :: z:A$; and
- (iv) $\cdot; y:B \vdash (\nu x)(Q^{\langle y, x \rangle} \mid P^{\langle x, z \rangle}) \approx [y \leftrightarrow z] :: z:B$.

Thus, intuitively, if A, B are service specifications then by establishing $A \simeq B$ one can claim that having A is as good as having B , because we can build one from the other using an isomorphism. Isomorphisms in linear logic can then be used to simplify/transform service interfaces in the π -calculus. They can also help validating our interpretation with respect to basic linear logic principles. Let us consider the simple example of multiplicative conjunction \otimes . A basic linear logic principle is $A \otimes B \vdash B \otimes A$.

Our interpretation of $A \otimes B$ may appear asymmetric as, in general, a channel of type $A \otimes B$ is not typable by $B \otimes A$. Theorem 6.4 below states the symmetric nature of \otimes as a type isomorphism: symmetry is realized by a process which *coerces* any session of type $A \otimes B$ to a session of type $B \otimes A$. Other sensible isomorphisms, such as, e.g., $(A \oplus B) \multimap C \simeq (A \multimap C) \& (B \multimap C)$, can be handled similarly.

Theorem 6.4. *Let A, B be any type, as in Def 3.1. Then $A \otimes B \simeq B \otimes A$.*

Proof. We verify conditions (i)-(iv) hold for processes $P^{(x,y)}, Q^{(y,x)}$ defined as

$$\begin{aligned} P^{(x,y)} &= x(u).(\nu n)y\langle n \rangle.([x \leftrightarrow n] \mid [u \leftrightarrow y]) \\ Q^{(y,x)} &= y(w).(\nu m)x\langle m \rangle.([y \leftrightarrow m] \mid [w \leftrightarrow x]) \end{aligned}$$

Checking (i)-(ii), i.e., $\cdot; x:A \otimes B \vdash P^{(x,y)} :: y:B \otimes A$ and $\cdot; y:B \otimes A \vdash Q^{(y,x)} :: x:A \otimes B$ is easy; the use of rule (Tid) ensures that both typings hold for any A, B .

We then show (iii) and (iv). We sketch only the proof of (iii); the proof of (iv) is analogous. Let $M = (\nu y)(P^{(x,y)} \mid Q^{(y,z)}), N = [x \leftrightarrow z]$; we need to show $\cdot; x:A \otimes B \vdash M \approx N :: z:A \otimes B$. By Prop. 5.5, we have to show that for every $K \in \mathcal{K}_{\cdot; x:A \otimes B}$, we have $\vdash K[M] \approx K[N] :: z:A \otimes B$. In turn, this implies exhibiting a typed context bisimilarity \mathcal{R} containing $(K[M], K[N])$. Letting $\mathcal{S} = \{(R_1, R_2) : K[M] \Longrightarrow R_1, K[N] \Longrightarrow R_2\}$, we set $\mathcal{R} = \mathcal{W}_{\vdash z:A \otimes B} \cup \mathcal{S} \cup \mathcal{S}^{-1}$. Following expected lines, \mathcal{R} can be shown to be a typed context bisimilarity; see Appendix C.3 in Page 46 for details. \square

7 Related Work and Concluding Remarks

Related Work. In [24], termination for an asynchronous π -calculus is established by exploiting action types enhanced with causality and linearity information. As in our work, the proof of termination in [24] uses logical relations; their semantic interpretation of types relies heavily on conditions for ensuring the duality/complementarity of processes, which are provided by the action type system. As applications, [24] establishes a finite axiomatization of weak bisimilarity and liveness of linear processes. Logical relations are also used in [19] to prove termination for a functional, localized variant of the π -calculus; this fragment is characterized via constraints on the syntax and the types of processes, which are arguably more standard than those in [24]. The result is then extended to a larger language by simulation-based proofs.

In comparison to the type systems studied in [24, 19], the type system considered here—binary session types—is more standard and well-established. In fact, while we started from a well-known type discipline and established termination, both [24, 19] seem to have followed a somewhat opposite path, namely finding suitable type disciplines that guarantee termination. As a result, and because of the interpretation of intuitionistic linear logic as session types, our logical relations are more intuitive than in [24, 19]: they are truly defined on the structure of types, and do not require any (syntactic, semantic) conditions. Moreover, our proof is more direct than the one in [19], as we do not require to appeal to syntactic constraints on the processes.

A very recent work has proposed another interpretation of session types as linear logic propositions [7]. It is based on *soft* linear logic [15], and so the exponential “!”

is treated following a non canonical discipline that uses two different typing environments. Hence, typing rules and judgements in [7] are rather different from ours. A bound on the length of reductions starting from well-typed-processes is obtained; the proof appeals to techniques from Implicit Computational Complexity. While we do not provide a similar bound, we find it remarkable that our proof of termination (a very strong property in itself) follows *only* the principles and properties of [4]; in contrast to the developments in [7], our proof does not appeal to extraneous technical devices, and preserves a standard, intuitive treatment of “!”. This is particularly desirable for extensions/generalizations of our framework, such as the proposed in [23, 17].

Loosely related to typed context bisimilarity is [25], where a form of *linear bisimilarity* is proposed; it treats some visible actions as internal actions, thus leading to a larger equivalence which is shown to be a congruence. The only work on behavioral equivalences for session-based concurrency we are aware of is [14]. It introduces a behavioral theory for a variant of the π -calculus with asynchronous session communication and an event inspection primitive for buffered messages. The aim is to capture the distinction between order-preserving communications (inside already established connections) and non-order-preserving communications (outside established connections). In particular, the behavioral theory of [14] accounts for some principles for prefix commutation that appear similar to those induced by our proof conversions. However, the origin and the nature of these commutations are quite different. In fact, while in [14] prefix commutation arises from the distinction mentioned above, commutations in our (synchronous) framework are due to causality relations captured by types.

Concluding Remarks. Certifying termination of interacting concurrent systems is of paramount importance, from both foundational and practical standpoints. By relying on the principles established by an interpretation of linear logic as session types [4], we have introduced a theory of logical relations for session-typed disciplines; its main application is a proof that well-typed processes always terminate. This way, in addition to *safety* properties (nothing bad happens, cf. subject reduction), we have shown that session-typed processes also enjoy an important *liveness* property such as termination. Also, we have introduced a natural observational equivalence for typed processes. We developed two applications of these results. First, we have shown soundness of *proof conversions* with respect to observational equivalence—an issue left open in [4]. Second, we studied *type isomorphisms* resulting from linear logic equivalences in our setting. The basic properties of the interpretation—especially, the combination of subject reduction and termination—were of the essence in both applications. Ongoing work concerns sound *and* complete axiomatizations of typed context bisimilarity via proof conversions. Having introduced the method of logical relations for session types, in the short term we plan to explore further the method for obtaining other results, such as parametricity. Also, we would like to deepen on the consequences, in the process side, of recent extensions to the logic interpretation [23, 17].

References

1. S. Abramsky. Computational interpretations of linear logic. *Theor. Comput. Sci.*, 111:3–57, April 1993.

2. A. Barber. Dual intuitionistic linear logic. Technical report, LFCS-96-347, Univ. of Edinburgh, 1996.
3. M. Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theor. Comput. Sci.*, 195:205–226, March 1998.
4. L. Caires and F. Pfenning. Session types as intuitionistic linear propositions. In *CONCUR'2010*, volume 6269 of *LNCS*, pages 222–236. Springer, 2010.
5. L. Caires and F. Pfenning. Session types as intuitionistic linear propositions – extended version, 2011. In preparation.
6. B.-Y. E. Chang, K. Chaudhuri, and F. Pfenning. A judgmental analysis of linear logic. Technical report, CMU-CS-03-131R, Carnegie Mellon University, 2003.
7. U. Dal Lago and P. Di Giambardino. Soft session types. In *Proc. of 18th Workshop on Expressiveness in Concurrency – EXPRESS'11*, volume 64 of *EPTCS*, pages 59–73, 2011.
8. R. Demangeon, D. Hirschhoff, and D. Sangiorgi. Mobile processes and termination. In *Semantics and Algebraic Specification*, volume 5700 of *LNCS*. Springer, 2009.
9. R. Di Cosmo. A short survey of isomorphisms of types. *Mathematical Structures in Computer Science*, 15(5):825–838, 2005.
10. J.-Y. Girard and Y. Lafont. Linear logic and lazy computation. In *TAPSOFT'87, Vol.2*, volume 250 of *LNCS*, pages 52–66. Springer, 1987.
11. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 122–138. Springer, 1998.
12. R. Hu, N. Yoshida, and K. Honda. Session-based distributed programming in java. In *Proc. of ECOOP*, volume 5142 of *LNCS*, pages 516–541. Springer, 2008.
13. N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the pi-calculus. In *POPL*, pages 358–371, 1996.
14. D. Kouzapas, N. Yoshida, R. Hu, and K. Honda. On asynchronous session semantics. In *Proc. of FMOODS-FORTE'2011*, volume 6722 of *LNCS*, pages 228–243. Springer, 2011.
15. Y. Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.
16. N. Ng, N. Yoshida, O. Pernet, R. Hu, and Y. Kryptis. Safe parallel programming with session java. In *Proc. of COORDINATION*, volume 6721 of *LNCS*, pages 110–126. Springer, 2011.
17. F. Pfenning, L. Caires, and B. Toninho. Proof-carrying code in a session-typed process calculus. In *Proc. of CPP '11*. Springer, 2011. To appear.
18. R. Pucella and J. A. Tov. Haskell session types with (almost) no class. In *Proc. of ACM SIGPLAN Symposium on Haskell*, pages 25–36. ACM, 2008.
19. D. Sangiorgi. Termination of processes. *Mathematical Structures in Computer Science*, 16(1):1–39, 2006.
20. D. Sangiorgi and D. Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001.
21. R. Statman. Logical relations and the typed lambda-calculus. *Information and Control*, 65(2/3):85–97, 1985.
22. W. W. Tait. Intensional Interpretations of Functionals of Finite Type I. *J. Symbolic Logic*, 32:198–212, 1967.
23. B. Toninho, L. Caires, and F. Pfenning. Dependent session types via intuitionistic linear type theory. In *Proc. of PPDP '11*, pages 161–172, New York, NY, USA, 2011. ACM.
24. N. Yoshida, M. Berger, and K. Honda. Strong normalisation in the pi-calculus. *Inf. Comput.*, 191(2):145–202, 2004.
25. N. Yoshida, K. Honda, and M. Berger. Linearity and bisimulation. *J. Log. Algebr. Program.*, 72(2):207–238, 2007.

A Proofs of Section 4 (Termination)

Below, we write $P \not\rightarrow$ to mean that P cannot reduce; it can perform visible actions, though. Also, we write $P \rightarrow^k P'$ to denote a reduction sequence of length k from P to P' . Given a process $P \Downarrow$, we write $\text{mlen}(P)$ to stand for the length of the longest reduction sequence originating from P . Given terminating processes P_1, \dots, P_n , notation $\text{mlen}(P_1, \dots, P_n)$ stands for $\text{mlen}(P_1) + \dots + \text{mlen}(P_n)$.

A.1 Additional Requirements

These are results from [4] and the associated technical report [5].

Lemma A.1 (Action Shape Characterization Lemmas – Excerpt). *Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: x:C$. Then we have: (1) If $P \xrightarrow{\alpha} Q$ and $C = \mathbf{1}$ then $s(\alpha) \neq x$. (2) If $P \xrightarrow{\alpha} Q$ and $y:\mathbf{1} \in \Delta$ then $s(\alpha) \neq y$. (3) If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = x$ and $C = A \otimes B$ then $\alpha = \overline{(\nu y)x\langle y \rangle}$. (4) If $P \xrightarrow{\alpha} Q$ and $s(\alpha) = y$ and $y:A \otimes B \in \Delta$ then $\alpha = y(z)$.*

Lemma A.2 (Preservation Lemma). *Assume (a) $\Gamma; \Delta_1 \vdash D \rightsquigarrow P :: x:A_1 \otimes A_2$ with $P \xrightarrow{(\nu y)x\langle y \rangle} P'$; and (b) $\Gamma; \Delta_2, x:A_1 \otimes A_2 \vdash E \rightsquigarrow Q :: z:C$ with $Q \xrightarrow{x\langle y \rangle} Q'$. Then (1) $\text{cut } D(x.E) \equiv \equiv F$ for some F ; (2) $\Gamma; \Delta_1, \Delta_2 \vdash F \rightsquigarrow R :: z:C$ for $R \equiv (\nu y)(\nu x)(P' \mid Q')$.*

Lemma A.3. *Assume $\Gamma; \Delta \vdash D \rightsquigarrow P :: z:C$ and not $\text{live}(P)$. Then*

1. $C = \mathbf{1}$ or $C = !C'$, for some C' ;
2. $(x_i : A_i) \in \Delta$ implies $A_i = \mathbf{1}$ or there is B_j with $A_i = !B_j$;
3. $C = !C'$ implies $P \equiv (\nu \bar{x})(!z(y).R \mid R')$

A.2 Proof of Proposition 4.11

We repeat the statement in Page 10:

Proposition A.4. *Let $P \in \mathcal{L}[z:T]$ with $T \in \{A \otimes B, A \multimap B, A \oplus B, A \& B\}$. Then, there exist α, P' such that (i) $P \xrightarrow{\alpha} P'$, and (ii) if $T = A \otimes B$ then $\alpha = \overline{(\nu y)z\langle y \rangle}$; if $T = A \multimap B$ then $\alpha = z(y)$; if $T = A \oplus B$ then $\alpha = z.\text{inr}$ or $\alpha = z.\text{inl}$; if $T = A \& B$ then $\alpha = z.\text{inr}$ or $\alpha = z.\text{inl}$.*

Proof. By Def 4.4, we have that $\cdot; \cdot \vdash P :: z:T$ and $P \Downarrow$. Since $T \notin \{\mathbf{1}, !T'\}$ then, using Lemma A.3, we know that $\text{live}(P)$ holds. Hence, the Progress Lemma can be used to infer that either $P \rightarrow P'$ or $P \xrightarrow{\alpha} P'$, with $s(\alpha) = z$. Termination ensures that such reductions, before or after α , are finite. This gives us Part (i). Part (ii) on the actual shape of α can be inferred using Lemma A.1. \square

A.3 Proof of Proposition 4.12

We repeat the statement in Page 10:

Proposition A.5. *Let P, Q be well-typed processes. If $P \in \mathcal{L}[T]$ and $P \equiv_! Q$ then $Q \in \mathcal{L}[T]$.*

Proof. By induction the definition of $P \equiv_! Q$ (Def. 4.1). Given Prop 4.10, it suffices to consider only the sharpened replication axioms. In each case, we use an auxiliary induction on the structure of T , which relies on Prop 4.3 and on the operational correspondence between P and Q given by Prop 4.2.

- Axiom (1): Then we have two sub-cases. In the first one, we have

$$\begin{aligned} P &= (\nu u)(!u(z).P_1 \mid (\nu y)(P_2 \mid P_3)) \\ Q &= (\nu y)((\nu u)(!u(z).P_1 \mid P_2) \mid (\nu u)(!u(z).P_1 \mid P_3)) \end{aligned}$$

Hence, sub-process $!u(z).P_1$ has been distributed to the unguarded processes P_2 and P_3 . We proceed by induction on the structure of the type T . Each case proceeds by showing that Q satisfies the termination, well-typedness, and operational correspondence requirements stated in Def 4.4. For the latter, we use Prop 4.2(1) and 4.2(2). We have six cases to check; we detail only some of them as the rest is similar.

Case $P \in \mathcal{L}[z:1]$. Then $P \Downarrow$ and $\cdot; \cdot \vdash P :: z:1$ and for all P' such that $P \Longrightarrow P'$ and $P' \not\rightarrow$ it implies that $P' \equiv_! \mathbf{0}$. First, by Prop 4.3, we have that $Q \Downarrow$. Now, since $\cdot; \cdot \vdash P :: z:1$ it is easy to show that there exists a typing derivation for $\cdot; \cdot \vdash Q :: z:1$. Finally, by Prop 4.2(1), we know that Q can match any reduction from P . Therefore, there exists a Q' such that $Q \Longrightarrow Q'$ and $Q' \not\rightarrow$ and $P' \equiv_! Q'$. By transitivity of $\equiv_!$, we have that $Q' \equiv_! \mathbf{0}$ and so $Q \in \mathcal{L}[z:1]$, as desired.

Case $P \in \mathcal{L}[z:A \multimap B]$. Then $P \Downarrow$ and $\cdot; \cdot \vdash P :: z:A \multimap B$. Hence, by Prop 4.11, P has an input action on z . Moreover, by Def 4.4, for all P', y such that $P \xrightarrow{z(y)} P'$ it implies that $\forall R \in \mathcal{L}[y:A]. (\nu y)(P' \mid R) \in \mathcal{L}[z:B]$. First, by Prop 4.3, we have that $Q \Downarrow$. Now, since $\cdot; \cdot \vdash P :: z:A \multimap B$, it is easy to show that there exists a typing derivation for $\cdot; \cdot \vdash Q :: z:A \multimap B$. Finally, by Prop 4.2(1) and 4.2(2), we know that Q can match any reduction/transition from P . Therefore, there exists a Q' such that $Q \xrightarrow{z(y)} Q'$ and $P' \equiv_! Q'$. Now, by induction hypothesis we have that $\forall R \in \mathcal{L}[y:A]. (\nu y)(Q' \mid R) \in \mathcal{L}[z:B]$, and so $Q \in \mathcal{L}[z:A \multimap B]$, as desired.

Case $P \in \mathcal{L}[z:A \otimes B]$. Then $P \Downarrow$ and $\cdot; \cdot \vdash P :: z:A \otimes B$. Hence, by Prop 4.11, P has an output action on z . Moreover, by Def 4.4, for all P', y such that $P \xrightarrow{(\nu y)z(y)} P'$ it implies that there exist P_1, P_2 such that $P' \equiv_! P_1 \mid P_2$ and $P_1 \in \mathcal{L}[y:A]$ and $P_2 \in \mathcal{L}[z:B]$. First, by Prop 4.3, we have that $Q \Downarrow$. Now, since $\cdot; \cdot \vdash P :: z:A \otimes B$, it is easy to show that there exists a typing derivation for $\cdot; \cdot \vdash Q :: z:A \otimes B$. Now, by Prop 4.2(1) and 4.2(2), we know that Q can match any reduction/transition from P . Therefore, there exists a Q' such that

$Q \xrightarrow{(\nu y)z(y)} Q'$ and $P' \equiv_! Q'$. Now, by transitivity we have that $Q' \equiv_! P_1 \mid P_2$, and so $Q \in \mathcal{L}[z:A \otimes B]$, as desired.

Case $P \in \mathcal{L}[z:!A]$. Similar to the case $P \in \mathcal{L}[z:1]$.

The second sub-case is symmetric to the first one, with P defined as Q and Q defined as P . As such, sub-process $!u(z).P_1$ has been “factorized” from the process expression. The analysis follows the lines of the first case and is omitted.

- Axiom (2): Then we have two sub-cases. In the first one, we have:

$$\begin{aligned} P &= (\nu u)(!u(y).P_1 \mid (\nu v)(!v(z).P_2 \mid P_3)) \\ Q &= (\nu v)((!v(z).(\nu u)(!u(y).P_1 \mid P_2)) \mid (\nu u)(!u(y).P_1 \mid P_3)) \end{aligned}$$

Similarly as before, sub-process $!u(y).P_1$ has been distributed to the unguarded process P_3 and to the input-guarded replicated process $!v(z).P_2$. We proceed by induction on the structure of the type T . Each case proceeds by showing that Q satisfies the requirements stated in Def 4.4. The analysis mirrors the one given above for Axiom (1), using Prop 4.3, observing that typability of P under some type T implies typability of Q under T , and exploiting the operational correspondence between P and Q given by Prop 4.2(1) and 4.2(2).

In the second sub-case, P defined as Q and Q defined as P . As such, sub-process $!u(y).P_1$ has been factorized from the process expression. The analysis follows the lines of the first sub-case and is omitted.

- Axiom (3): Then we have two sub-cases. In the first one, we have

$$P = (\nu u)(!u(y).P_1 \mid P_2) \text{ with } u \notin \text{fn}(P_2) \quad Q = P_2$$

Hence, sub-process $!u(y).P_1$ is discarded, as it cannot be invoked by P_2 . We proceed by induction on the structure of the type T . Each case proceeds by showing that Q satisfies the requirements stated in Def 4.4. The crucial point is to observe that since $u \notin \text{fn}(P_2)$ then every reduction/transition from P originates in P_2 , and so they can be trivially matched by Q . As a consequence, P belongs to $\mathcal{L}[z:T]$, for some $z \neq u$. We have six cases to check; we detail two of them, the others are similar:

Case $P \in \mathcal{L}[z:1]$. Then $P \Downarrow$ and $;\cdot \vdash P :: z:1$ and for all P' such that $P \Longrightarrow P'$ and $P' \not\rightarrow$ it implies that $P' \equiv_! \mathbf{0}$. First, by Prop 4.3, we have that $Q \Downarrow$. Now, since $;\cdot \vdash P :: z:1$ it is possible to show that $;\cdot \vdash Q :: z:1$. Notice also that since $u \notin \text{fn}(P_2)$, none of the reductions from P to P' is a synchronization on u . Hence, every reduction of P originates in P_2 , and since $Q = P_2$, the thesis trivially holds.

Case $P \in \mathcal{L}[z:A \multimap B]$. Then $P \Downarrow$ and $;\cdot \vdash P :: z:A \multimap B$. Hence, by Prop 4.11, P has an input action on z . Moreover, by Def 4.4, for all P', y such that $P \xrightarrow{z(y)} P'$ it implies that $\forall R \in \mathcal{L}[y:A].(\nu y)(P' \mid R) \in \mathcal{L}[z:B]$. First, by Prop 4.3, we have that $Q \Downarrow$. Now, since $;\cdot \vdash P :: z:A \multimap B$ then it can be shown that $;\cdot \vdash Q :: z:A \multimap B$. Now, since $u \notin \text{fn}(P_2)$, none of the reductions/transition from P to P' is a synchronization on u . Hence, every reduction and transition of P originates in P_2 , and since $Q = P_2$, we immediately infer that $Q \in \mathcal{L}[z:A \multimap B]$, as desired.

The second sub-case is the symmetric of the first one, with P defined as Q and Q defined as P . That is, process Q is the extension of $P = P_2$ with a process $!u(y).P_1$ that it cannot invoke. Notice that we assume well-typed processes, and so the extended process Q is well-typed as well. The analysis follows the lines of the first case and is omitted. \square

A.4 Proof of Proposition 4.15

We repeat the statement in Page 10:

Proposition A.6 (Weakening). *Let P, Q be processes such that $P \in \mathcal{L}[T]$ and $Q \in \mathcal{L}[-:1]$. Then, $P \mid Q \in \mathcal{L}[T]$.*

Proof. By induction on the structure of the type T . First, it is worth observing that $P \in \mathcal{L}[T]$ and $Q \in \mathcal{L}[-:1]$ imply $;\cdot \vdash P :: T$ and $;\cdot \vdash Q :: -:1$, respectively. Hence, we can derive the typing $;\cdot \vdash P \mid Q :: T$ (cf. the derived rule (comp)). In fact, the type of Q indicates it cannot offer any visible action to its environment, and so it is “independent” from it.

If $T = -:1$ then $P \mid Q$ represents the parallel composition of two terminating processes that cannot interact with each other. Hence, for all R such that $P \mid Q \Longrightarrow R$ and $R \not\rightarrow$ we have that $R \equiv_! \mathbf{0}$, and so $P \mid Q \in \mathcal{L}[-:1]$. The cases in which $T \neq -:1$ rely on the fact that if $P \xrightarrow{\alpha} P'$ then there exists a process R such that $P \mid Q \xrightarrow{\alpha} R$. The proof is by induction on $k = \text{mlen}(Q)$. If $k = 0$ then $Q \not\rightarrow$ and for every weak transition $P \xrightarrow{\alpha} P'$, we have $P \mid Q \xrightarrow{\alpha} P' \mid Q = R$. In the inductive case, we assume $k > 0$, and so reductions (or the action α) from P may go interleaved with reductions from Q . Given $P \xrightarrow{\alpha} P'$ then by induction hypothesis there is an R' such that $P \mid Q \xrightarrow{\alpha} P' \mid Q' = R'$, with Q reducing to Q' in $k - 1$ steps. Then, if $Q' \rightarrow Q''$ we would have $P \mid Q \xrightarrow{\alpha} P' \mid Q' \rightarrow P' \mid Q''$ which is equivalent to write $P \mid Q \xrightarrow{\alpha} R$, with $R = P' \mid Q''$, and we are done. Finally, we observe that, given $P \xrightarrow{\alpha} P'$, process Q (and its derivatives) pose no difficulties when decomposing P' into smaller processes (in the case $T = z:A \otimes B$, for instance). Hence, we can conclude that if $P \in \mathcal{L}[T]$ then $P \mid Q \in \mathcal{L}[T]$, as desired. \square

A.5 Proof of Lemma 4.16

We repeat the statement in Page 11 below. In the proof, we use G, G', \dots and D, D', \dots to range over processes in \mathcal{C}_Γ and \mathcal{C}_Δ , respectively. Also, by a slight abuse of notation we write $\mathcal{L}[x:A]$ and $!z(y).\mathcal{L}[y:A]$ to denote a process included in $\mathcal{L}[x:A]$ and $\mathcal{L}![z:A]$, respectively.

Lemma A.7. *Let P be a process. If $\Gamma; \Delta \vdash P :: T$ then $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$.*

Proof. By induction on the derivation of $\Gamma; \Delta \vdash P :: T$, with a case analysis on the last typing rule used.

Thus, we have 18 cases to check. In all cases, we appeal to Lemma 4.8 and show that every $M = (\nu \tilde{u}, \tilde{x})(P \mid G \mid D)$ with $G \in \mathcal{C}_\Gamma$ and $D \in \mathcal{C}_\Delta$, is in $\mathcal{L}[T]$. In case (Tid),

the proof uses Prop 4.9 (closure wrt substitution) and Prop 4.14 (backward closure). In cases (T \otimes L), (T \multimap L), (Tcopy), (T \oplus L), (T&L₁), and (T&L₂), the proof proceeds in two steps: first, relying on Prop 4.13 (forward closure) we show that every M'' such that $M \Longrightarrow M''$ is in $\mathcal{L}[T]$; then, we use this result in combination with Prop 4.14 (backward closure) to conclude that $M \in \mathcal{L}[T]$. In cases (T1R), (T \otimes R), (T \multimap R), (T!R), (T \oplus R₁), and (T \oplus R₂), the proof consists in showing that M conforms to some specific case of Def 4.4. Case (T1L) uses Prop 4.15 (weakening). Cases (T \otimes L), (T \multimap L), (T \oplus L), and (T&L₁), use the liveness guarantee given by Prop 4.11. Cases (Tcopy), (T!L) and (Tcut[!]) use Prop 4.10 (closure under \equiv). Cases (Tcut), (T \multimap R), and (T!R) use Prop 4.12 (closure under \equiv_1).

0. Case (Tid): $F; x:A \vdash [x \leftrightarrow z] :: z:A$.

Pick any $G \in \mathcal{C}_F$:

(a) $G \Downarrow, G \not\rightarrow$ [By Prop 4.7]

(b) $D \in \mathcal{L}[x:A]$

(c) $M = (\nu \tilde{u}, x)([x \leftrightarrow z] \mid G \mid D) \in \mathcal{L}[z:A]$

The proof of (c) is immediate:

(d) $M \longrightarrow (\nu \tilde{u})(G\{z/x\} \mid D\{z/x\})$
 $\equiv_1 D\{z/x\} = M'$ [Since $x \notin \text{fn}(G)$]

(e) $M' \in \mathcal{L}[z:A]$ [By (b) and Prop 4.9]

(f) $M \in \mathcal{L}[z:A]$ [By (d), (e), and Prop 4.14]

$[x \leftrightarrow z] \in \mathcal{L}[F; x:A \vdash z:A]$ [By (c) and Lemma 4.8]

1. Case (T1R): $F; \cdot \vdash \mathbf{0} :: z:\mathbf{1}$.

Pick any $G \in \mathcal{C}_F$:

(a) $G \Downarrow, G \not\rightarrow$ [By Prop 4.7]

(b) $M = (\nu \tilde{u})(\mathbf{0} \mid G) \in \mathcal{L}[z:\mathbf{1}]$

The proof of (b) is immediate:

(c) $M \not\rightarrow \wedge M \equiv_1 \mathbf{0}$ [Using (a)]

(d) $M \in \mathcal{L}[z:\mathbf{1}]$ [By (c) and Def 4.4]

$\mathbf{0} \in \mathcal{L}[F; \cdot \vdash z:\mathbf{1}]$ [By (b) and Lemma 4.8]

2. Case (T1L): $\Gamma; \Delta, z:1 \vdash P :: T$.

(a) $\Gamma; \Delta \vdash P :: T$	[Premise of rule (T1L)]
(b) $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$	[By i.h. on (a)]
Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta$:	
(c) $M_1 = (\nu \tilde{u}, \tilde{x})(P \mid G \mid D) \in \mathcal{L}[T]$	[By Lemma 4.8 on (b)]
Pick any $R \in \mathcal{L}[z:1]$ and fix $M_2 = M_1 \mid R$	
(d) $M_2 \in \mathcal{L}[T]$	[By (c) and Prop 4.15]
(e) $(\nu \tilde{u}, \tilde{x}, z)(P \mid G \mid D \mid R) \in \mathcal{L}[T]$	[Expanding (d)]
$P \in \mathcal{L}[\Gamma; \Delta, z:1 \vdash T]$	[By (e) and Lemma 4.8]

3. Case (T⊗L): $\Gamma; \Delta, z:A \otimes B \vdash z(y).P :: T$

(a) $\Gamma; \Delta, y:A, z:B \vdash P :: T$	[Premise of rule (T⊗L)]
(b) $P \in \mathcal{L}[\Gamma; \Delta, y:A, z:B \vdash T]$	[By i.h. on (a)]
Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta$:	
(c) $G \Downarrow, G \not\rightarrow, D \Downarrow$	[By Prop 4.7]
(d) $(\nu \tilde{u}, \tilde{x}, y, z)(P \mid G \mid D \mid \mathcal{L}[y:A] \mid \mathcal{L}[z:B]) \in \mathcal{L}[T]$	[By Lemma 4.8 on (b)]
Pick $R \in \mathcal{L}[z:A \otimes B]$:	
(e) $R \Downarrow$	[By Def 4.4]
(f) $R \xrightarrow{(\nu y)z\langle y \rangle} R'$	[By Prop 4.11]
(g) $R' \equiv_1 R'_1 \mid R'_2 \wedge R'_1 \in \mathcal{L}[y:A] \wedge R'_2 \in \mathcal{L}[z:B]$	[By Def 4.4]
Fix $M = (\nu \tilde{u}, \tilde{x}, z)(z(y).P \mid G \mid D \mid R)$	
(h) $\forall M''. M \Longrightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$	
We prove (h) by induction on $k = \text{mlen}(D, R)$:	
Base case $k = 0$. Hence, $D \not\rightarrow$, and $R \not\rightarrow$:	
$M \longrightarrow (\nu \tilde{u}, \tilde{x}, z, y)(P \mid G \mid D \mid R'_1 \mid R'_2) = M''$	[Because of (f)]
$M'' \in \mathcal{L}[T]$	[Using (d) and (g)]
Inductive case $k > 0$:	
Fix the set $W = \{M' \mid M \longrightarrow^{k-1} M'\}$	
$\forall M' \in W. M' \in \mathcal{L}[T]$	[By i.h.]
Fix the set $W' = \{M'' \mid M' \longrightarrow M'' \wedge M' \in W\}$	
$\forall M'' \in W'. M'' \in \mathcal{L}[T]$	[By Prop 4.13]
(i) $M \in \mathcal{L}[T]$	[By (h) and Prop 4.14]
$z(y).P \in \mathcal{L}[\Gamma; \Delta, z:A \otimes B \vdash T]$	[By (i) and Lemma 4.8]

4. Case (T⊗R): $\Gamma; \Delta, \Delta' \vdash (\nu y)z\langle y \rangle.(P \mid Q) :: z:A \otimes B$

- (a) $\Gamma; \Delta \vdash P :: y:A$ [Premise of rule (T⊗R)]
 - (b) $\Gamma; \Delta' \vdash Q :: z:B$ [Premise of rule (T⊗R)]
 - (c) $P \in \mathcal{L}[\Gamma; \Delta \vdash y:A]$ [By i.h on (a)]
 - (d) $Q \in \mathcal{L}[\Gamma; \Delta' \vdash z:B]$ [By i.h on (b)]
- Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta, D' \in \mathcal{C}_{\Delta'}$:
- (e) $G \Downarrow, G \not\rightarrow, D \Downarrow, D' \Downarrow$ [By Prop 4.7]
 - (f) $(\nu \tilde{u}, \tilde{x}_1)(P \mid G \mid D) \in \mathcal{L}[y:A]$ [By Lemma 4.8 on (c)]
 - (g) $(\nu \tilde{u}, \tilde{x}_2)(Q \mid G \mid D') \in \mathcal{L}[z:B]$ [By Lemma 4.8 on (d)]
- Fix $\tilde{x} = \tilde{x}_1 \cup \tilde{x}_2$:

$$(h) M = (\nu \tilde{u}, \tilde{x})((\nu y)z\langle y \rangle.(P \mid Q) \mid G \mid D \mid D') \in \mathcal{L}[z:A \otimes B]$$

We prove (h) by induction on $k = \text{mlen}(D, D')$: [Possible by (e)]

Base case $k = 0$. Hence, $D \not\rightarrow$, and $D' \not\rightarrow$:

$$(i) M \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{u}, \tilde{x})(P \mid Q \mid G \mid D \mid D') = M'$$

$$M' \equiv \underbrace{(\nu \tilde{u}, \tilde{x}_1)(P \mid G \mid D)}_{M'_1} \mid \underbrace{(\nu \tilde{u}, \tilde{x}_2)(Q \mid G \mid D')}_{M'_2}$$

- (j) $M'_1 \in \mathcal{L}[y:A]$ [By (f)]
 - (k) $M'_2 \in \mathcal{L}[z:B]$ [By (g)]
- $M \in \mathcal{L}[z:A \otimes B]$ [By Def 4.4, using (i), (j), and (k)]

Inductive case $k > 0$:

$$\text{Fix the set } W = \{M' \mid M \rightarrow^{k-1} M'\}$$

$$\forall M' \in W. M' \in \mathcal{L}[z:A \otimes B] \quad \text{[By i.h.]}$$

$$\text{Fix the set } W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$$

$$\forall M'' \in W'. M'' \in \mathcal{L}[z:A \otimes B] \quad \text{[By Prop 4.13]}$$

$$(\nu y)z\langle y \rangle.(P \mid Q) \in \mathcal{L}[\Gamma; \Delta, \Delta' \vdash z:A \otimes B] \quad \text{[By (h) and Lemma 4.8]}$$

5. Case (T→L): $\Gamma; \Delta, \Delta', z:A \multimap B \vdash (\nu y)z\langle y \rangle.(P \mid Q) :: T$

- (a) $\Gamma; \Delta \vdash P :: y:A$ [Premise of rule (T→L)]
 - (b) $\Gamma; \Delta', z:B \vdash Q :: T$ [Premise of rule (T→L)]
 - (c) $P \in \mathcal{L}[\Gamma; \Delta \vdash y:A]$ [By i.h on (a)]
 - (d) $Q \in \mathcal{L}[\Gamma; \Delta', z:B \vdash T]$ [By i.h on (b)]
- Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta, D' \in \mathcal{C}_{\Delta'}$:
- (e) $G \Downarrow, G \not\rightarrow, D \Downarrow, D' \Downarrow$ [By Prop 4.7]
 - (f) $(\nu \tilde{u}, \tilde{x}_1)(P \mid G \mid D) \in \mathcal{L}[y:A]$ [By Lemma 4.8 on (c)]
 - (g) $(\nu \tilde{u}, \tilde{x}_2, z)(Q \mid G \mid D' \mid \mathcal{L}[z:B]) \in \mathcal{L}[T]$ [By Lemma 4.8 on (d)]

Fix $\tilde{x} = \tilde{x}_1 \cup \tilde{x}_2$ and pick $R \in \mathcal{L}[z:A \multimap B]$:

(h) $R \Downarrow$ [By Def 4.4]

(i) $R \xrightarrow{z\langle y \rangle} R'$ [By Prop 4.11]

(j) $\forall Q \in \mathcal{L}[y:A]. (\nu y)(R' \mid Q) \in \mathcal{L}[z:B]$ [By Def 4.4]

Fix $M = (\nu \tilde{u}, \tilde{x}, z)((\nu y)z\langle y \rangle.(P \mid Q) \mid G \mid D \mid D' \mid R)$

(k) $\forall M''. M \Longrightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$

We prove (k) by induction on $k = \text{mlen}(D, D', R)$: [Possible by (e) and (h)]

Base case $k = 0$. Hence, $D \not\rightarrow, D' \not\rightarrow, R \not\rightarrow$:

$M \rightarrow (\nu \tilde{u}, \tilde{x}, z, y)(P \mid Q \mid G \mid D \mid D' \mid R') = M''$ [Because of (i)]

Fix $M^* = (\nu \tilde{u}, \tilde{x}_1)(P \mid G \mid D)$:

(l) $M^* \in \mathcal{L}[y:A]$ [Using (f)]

$M'' \equiv_! (\nu \tilde{u}, \tilde{x}_2, z)(Q \mid G \mid D' \mid (\nu y)(R' \mid M^*)) = M_1$

(m) $(\nu y)(R' \mid M^*) \in \mathcal{L}[z:B]$ [Using (j) and (l)]

(n) $M_1 \in \mathcal{L}[T]$ [Using (g) and (m)]

$M'' \in \mathcal{L}[T]$ [By Prop 4.12 and (n)]

Inductive case $k > 0$:

Fix the set $W = \{M' \mid M \rightarrow^{k-1} M'\}$

$\forall M' \in W. M' \in \mathcal{L}[T]$ [By i.h.]

Fix the set $W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$

$\forall M'' \in W'. M'' \in \mathcal{L}[T]$ [By Prop 4.13]

(o) $M \in \mathcal{L}[T]$ [By (k) and Prop 4.14]

$(\nu y)z\langle y \rangle.(P \mid Q) \in \mathcal{L}[\Gamma; \Delta, \Delta', z:A \multimap B \vdash T]$ [By (o) and Lemma 4.8]

6. Case (T \multimap R): $\Gamma; \Delta \vdash z(y).P :: z:A \multimap B$

(a) $\Gamma; \Delta, y:A \vdash P :: z:B$ [Premise of rule (T \multimap R)]

(b) $P \in \mathcal{L}[\Gamma; \Delta, y:A \vdash z:B]$ [By i.h on (a)]

Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta$:

(c) $G \Downarrow, G \not\rightarrow, D \Downarrow$ [By Prop 4.7]

(d) $(\nu \tilde{u}, \tilde{x}, y)(P \mid G \mid D \mid \mathcal{L}[y:A]) \in \mathcal{L}[z:B]$ [By Lemma 4.8 on (b)]

(e) $M = (\nu \tilde{u}, \tilde{x})(z(y).P \mid G \mid D) \in \mathcal{L}[z:A \multimap B]$

We prove (e) by induction on $k = \text{mlen}(D)$: [Possible by (c)]

Base case $k = 0$. Hence, $D \not\rightarrow$:

$$(f) M \xrightarrow{z(y)} (\nu \tilde{u}, \tilde{x})(P \mid G \mid D) = M_1$$

Pick any $R \in \mathcal{L}[y:A]$:

$$(g) (\nu \tilde{u}, \tilde{x}, y)(P \mid G \mid D \mid R) \in \mathcal{L}[z:B] \quad [\text{Using (d)}]$$

$$M \in \mathcal{L}[z:A \multimap B] \quad [\text{By Def 4.4, using (f),(g)}]$$

Inductive case $k > 0$:

$$\text{Fix the set } W = \{M' \mid M \rightarrow^{k-1} M'\}$$

$$\forall M' \in W. M' \in \mathcal{L}[z:A \multimap B] \quad [\text{By i.h.}]$$

$$\text{Fix the set } W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$$

$$\forall M'' \in W'. M'' \in \mathcal{L}[z:A \multimap B] \quad [\text{Prop 4.13}]$$

$$z(y).P \in \mathcal{L}[\Gamma; \Delta \vdash z:A \multimap B] \quad [\text{By Lemma 4.8 on (e)}]$$

7. Case (Tcut): $\Gamma; \Delta, \Delta' \vdash (\nu z)(P \mid Q) :: T$

$$(a) \Gamma; \Delta \vdash P :: z:A \quad [\text{Premise of rule (Tcut)}]$$

$$(b) \Gamma; \Delta', z:A \vdash Q :: T \quad [\text{Premise of rule (Tcut)}]$$

$$(c) P \in \mathcal{L}[\Gamma; \Delta \vdash z:A] \quad [\text{By i.h. on (a)}]$$

$$(d) Q \in \mathcal{L}[\Gamma; \Delta', z:A \vdash T] \quad [\text{By i.h. on (b)}]$$

Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta, D' \in \mathcal{C}_{\Delta'}$:

$$(e) (\nu \tilde{u}, \tilde{x}_1)(P \mid G \mid D) \in \mathcal{L}[z:A] \quad [\text{By Lemma 4.8 on (c)}]$$

$$(f) (\nu \tilde{u}, \tilde{x}_2, z)(Q \mid G \mid D' \mid \mathcal{L}[z:A]) \in \mathcal{L}[T] \quad [\text{By Lemma 4.8 on (d)}]$$

$$\text{Fix } M = (\nu \tilde{u}, \tilde{x})((\nu z)(P \mid Q) \mid G \mid D \mid D')$$

$$M \equiv_! (\nu z)((\nu \tilde{u}, \tilde{x}_2)(Q \mid G \mid D') \mid \underbrace{(\nu \tilde{u}, \tilde{x}_1)(P \mid G \mid D)}_{M_1}) = M'$$

$$(g) M_1 \in \mathcal{L}[z:A] \quad [\text{Using (e)}]$$

$$(h) M' \in \mathcal{L}[T] \quad [\text{Combining (g) and (f)}]$$

$$(i) M \in \mathcal{L}[T] \quad [\text{Using (h) and Prop 4.12}]$$

$$(j) (\nu \tilde{u}, \tilde{x})((\nu z)(P \mid Q) \mid G \mid D \mid D') \in \mathcal{L}[T] \quad [\text{Expanding (i)}]$$

$$(\nu z)(P \mid Q) \in \mathcal{L}[\Gamma; \Delta, \Delta' \vdash T] \quad [\text{By Lemma 4.8 on (j)}]$$

8. Case (Tcut[!]): $\Gamma; \Delta \vdash (\nu z)(!z(y).P \mid Q) :: T$

$$(a) \Gamma; \cdot \vdash P :: y:A \quad [\text{Premise of rule (Tcut[!])}]$$

$$(b) \Gamma, z:A; \Delta \vdash Q :: T \quad [\text{Premise of rule (Tcut[!])}]$$

$$(c) P \in \mathcal{L}[\Gamma; \cdot \vdash y:A] \quad [\text{By i.h. on (a)}]$$

$$(d) Q \in \mathcal{L}[\Gamma, z:A; \Delta \vdash T] \quad [\text{By i.h. on (b)}]$$

Pick any $G \in \mathcal{C}_\Gamma$, $D \in \mathcal{C}_\Delta$:

(e) $(\nu\tilde{u})(P \mid G) \in \mathcal{L}[y:A]$ [By Lemma 4.8 on (c)]

(f) $(\nu\tilde{u}, \tilde{x}, z)(Q \mid G \mid !z(y).\mathcal{L}[y:A] \mid D) \in \mathcal{L}[T]$ [By Lemma 4.8 on (d)]

Fix $M = (\nu\tilde{u}, z, \tilde{x})(!z(y).P \mid Q \mid G \mid D)$

(g) $M \in \mathcal{L}[T]$ [Combining (e) and (f)]

$M \equiv (\nu\tilde{u}, \tilde{x})((\nu z)(!z(y).P \mid Q) \mid G \mid D) = M'$

(h) $M' \in \mathcal{L}[T]$ [From (g), using Prop 4.10]

$(\nu z)(!z(y).P \mid Q) \in \mathcal{L}[\Gamma; \Delta \vdash T]$ [By Lemma 4.8 on (h)]

9. Case (Tcopy): $\Gamma, z:A; \Delta \vdash (\nu y)z\langle y \rangle.P :: T$

(a) $\Gamma, z:A; \Delta, y:A \vdash P :: T$ [Premise of rule (Tcopy)]

(b) $P \in \mathcal{L}[\Gamma, z:A; \Delta, y:A \vdash P :: T]$ [By i.h on (a)]

Pick any $G \in \mathcal{C}_\Gamma$, $D \in \mathcal{C}_\Delta$:

(c) $G \Downarrow, G \not\rightarrow, D \Downarrow$ [By Prop 4.7]

(d) $(\nu\tilde{u}, z, \tilde{x}, y)(P \mid G \mid !z(y).\mathcal{L}[y:A] \mid D \mid \mathcal{L}[y:A]) \in \mathcal{L}[T]$ [By Lemma 4.8 on (b)]

Pick $R \in \mathcal{L}[y:A]$:

Fix $M = (\nu\tilde{u}, z, \tilde{x})((\nu y)z\langle y \rangle.P \mid G \mid !z(y).R \mid D)$

(e) $\forall M''. M \Longrightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$

We prove (e) by induction on $k = \text{mlen}(D)$: [Possible by (c)]

Base case $k = 0$. Hence, $D \not\rightarrow$:

$M \rightarrow \equiv (\nu\tilde{u}, z, \tilde{x}, y)(P \mid G \mid !z(y).R \mid D \mid R) = M''$

$M'' \in \mathcal{L}[T]$ [Using (d) and Prop 4.10]

Inductive case $k > 0$:

Fix the set $W = \{M' \mid M \rightarrow^{k-1} M'\}$

$\forall M' \in W. M' \in \mathcal{L}[T]$ [By i.h.]

Fix the set $W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$

$\forall M'' \in W'. M'' \in \mathcal{L}[T]$ [By Prop 4.13]

(f) $M \in \mathcal{L}[T]$ [By (e) and Prop 4.14]

$(\nu y)z\langle y \rangle.P \in \mathcal{L}[\Gamma, z:A; \Delta \vdash T]$ [By (f) and Lemma 4.8]

10. Case (T!L): $\Gamma; \Delta, y:!A \vdash P :: T$

(a) $\Gamma, z:A; \Delta \vdash P\{z/y\} :: T$ [Premise of rule (T!L)]

(b) $P\{z/y\} \in \mathcal{L}[\Gamma, z:A; \Delta \vdash T]$ [By i.h on (a)]

Pick any $G \in \mathcal{C}_\Gamma$, $D \in \mathcal{C}_\Delta$:

(c) $(\nu\tilde{u}, z, \tilde{x})(P\{z/y\} \mid G \mid !z(w).\mathcal{L}[w:A] \mid D) \in \mathcal{L}[T]$ [By Lemma 4.8 on (b)]

(d) $(\nu\tilde{u}, y, \tilde{x})(P \mid G \mid \underbrace{!y(w).\mathcal{L}[w:A]}_R \mid D) \in \mathcal{L}[T]$ [By \equiv (α -conv) on (c)]

- (e) $R \in \mathcal{L}[y:!A]$ [By Def 4.4]
 $P \in \mathcal{L}[\Gamma; \Delta, y:!A \vdash T]$ [By (d), (e), Prop 4.10, and Lemma 4.8]

11. Case (T!R): $\Gamma; \cdot \vdash !z(y).P :: z:!A$

- (a) $\Gamma; \cdot \vdash P :: y:A$ [Premise of rule (T!R)]
 (b) $P \in \mathcal{L}[\Gamma; \cdot \vdash y:A]$ [By i.h on (a)]
 Pick any $G \in \mathcal{C}_\Gamma$:
 (c) $G \Downarrow, G \not\rightarrow$ [By Prop 4.7]
 (d) $(\nu \tilde{u})(P \mid G) \in \mathcal{L}[y:A]$ [By Lemma 4.8 on (b)]
 Fix $M = (\nu \tilde{u})(!z(y).P \mid G)$
 $M \equiv !z(y).(\nu \tilde{u})(P \mid G) = M'$ [By Def 4.1, Axiom (2)]
 (e) $M' \in \mathcal{L}[z:!A]$ [By Def 4.4, using (d)]
 (f) $M \in \mathcal{L}[z:!A]$ [By (e) and Prop 4.12]
 $!z(y).P \in \mathcal{L}[\Gamma; \cdot \vdash z:!A]$ [By (f) and Lemma 4.8]

12. Case (T \oplus L): $\Gamma; \Delta, z:A \oplus B \vdash z.\text{case}(P, Q) :: T$

- (a) $\Gamma; \Delta, z:A \vdash P :: T$ [Premise of rule (T \oplus L)]
 (b) $\Gamma; \Delta, z:B \vdash Q :: T$ [Premise of rule (T \oplus L)]
 (c) $P \in \mathcal{L}[\Gamma; \Delta, z:A \vdash T]$ [By i.h on (a)]
 (d) $Q \in \mathcal{L}[\Gamma; \Delta, z:B \vdash T]$ [By i.h on (b)]
 Pick any $G \in \mathcal{C}_\Gamma, D \in \mathcal{C}_\Delta$:
 (e) $G \Downarrow, G \not\rightarrow, D \Downarrow$ [By Prop 4.7]
 (f) $(\nu \tilde{u}, \tilde{x}, z)(P \mid G \mid D \mid \mathcal{L}[z:A]) \in \mathcal{L}[T]$ [By Lemma 4.8 on (c)]
 (g) $(\nu \tilde{u}, \tilde{x}, z)(Q \mid G \mid D \mid \mathcal{L}[z:B]) \in \mathcal{L}[T]$ [By Lemma 4.8 on (d)]
 Pick $R \in \mathcal{L}[z:A \oplus B]$:
 (h) $R \Downarrow$ [By Def 4.4]
 (i) $R \xrightarrow{\overline{z.\text{inl}}} R' \vee R \xrightarrow{\overline{z.\text{inr}}} R'$ [By Prop 4.11]
 (j) $R \xrightarrow{\overline{z.\text{inl}}} R' \Rightarrow R' \in \mathcal{L}[z:A] \wedge R \xrightarrow{\overline{z.\text{inr}}} R' \Rightarrow R' \in \mathcal{L}[z:B]$ [By Def 4.4]
 Fix $M = (\nu \tilde{u}, \tilde{x}, z)(z.\text{case}(P, Q) \mid G \mid D \mid R)$:
 (k) $\forall M''. M \Rightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$

We prove (k) by induction on $k = \text{mlen}(D, R)$: [Possible by (e) and (h)]

Base case $k = 0$. Hence, $D \not\rightarrow, R \not\rightarrow$:

$M \rightarrow M''_1 \vee M'' \rightarrow M''_2$, where :

$M''_1 = (\nu \tilde{u}, \tilde{x}, z)(P \mid G \mid D \mid R')$ [Because of (i)]

$M''_2 = (\nu \tilde{u}, \tilde{x}, z)(Q \mid G \mid D \mid R')$ [Because of (i)]

$M''_1 \in \mathcal{L}[T]$ [Using (f)]

$M''_2 \in \mathcal{L}[T]$ [Using (g)]

Inductive case $k > 0$:

Fix the set $W = \{M' \mid M \rightarrow^{k-1} M'\}$

$\forall M' \in W. M' \in \mathcal{L}[T]$ [By i.h.]

Fix the set $W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$

$\forall M'' \in W'. M'' \in \mathcal{L}[T]$ [By Prop 4.13]

(l) $M \in \mathcal{L}[T]$ [By (k) and Prop 4.14]

$z.\text{case}(P, Q) \in \mathcal{L}[\Gamma; \Delta, z:A \oplus B \vdash T]$ [By (l) and Lemma 4.8]

13. Case (T&L₁): $\Gamma; \Delta, z:A \& B \vdash z.\text{inl}; P :: T$

(a) $\Gamma; \Delta, z:A \vdash P :: T$ [Premise of rule (T&L₁)]

(b) $P \in \mathcal{L}[\Gamma; \Delta, z:A \vdash T]$ [By i.h on (a)]

Pick any $G \in \mathcal{C}_T, D \in \mathcal{C}_\Delta$:

(c) $G \Downarrow, G \not\rightarrow, D \Downarrow$ [By Prop 4.7]

(d) $(\nu \tilde{u}, \tilde{x}, z)(P \mid G \mid D \mid \mathcal{L}[z:A]) \in \mathcal{L}[T]$ [By Lemma 4.8 on (b)]

Pick $R \in \mathcal{L}[z:A \& B]$:

(e) $R \Downarrow$ [By Def 4.4]

(f) $R \xrightarrow{z.\text{inl}} R_1 \vee R \xrightarrow{z.\text{inr}} R_2$ [By Prop 4.11]

(g) $R \xrightarrow{z.\text{inl}} R_1 \Rightarrow R_1 \in \mathcal{L}[z:A] \wedge R \xrightarrow{z.\text{inr}} R_2 \Rightarrow R_2 \in \mathcal{L}[z:B]$ [By Def 4.4]

Fix $M = (\nu \tilde{u}, \tilde{x}, z)(z.\text{inl}; P \mid G \mid D \mid R)$:

(h) $\forall M''. M \Rightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$

We prove (h) by induction on $k = \text{mlen}(D, R)$: [Possible by (c) and (e)]

Base case $k = 0$. Hence, $D \not\rightarrow, R \not\rightarrow$:

$M \rightarrow (\nu \tilde{u}, \tilde{x}, z)(P \mid G \mid D \mid R_1) = M''$

$M'' \in \mathcal{L}[T]$ [Using (d) and (g)]

Inductive case $k > 0$:

Fix the set $W = \{M' \mid M \rightarrow^{k-1} M'\}$

$\forall M' \in W. M' \in \mathcal{L}[T]$ [By i.h.]

Fix the set $W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$

$\forall M'' \in W'. M'' \in \mathcal{L}[T]$ [By Prop 4.13]

(j) $M \in \mathcal{L}[T]$ [By (h) and Prop 4.14]

$z.\text{inl}; P \in \mathcal{L}[\Gamma; \Delta, z:A \& B \vdash T]$ [By (j) and Lemma 4.8]

14. Case (T&L₂): Analogous to case (T&L₁).

15. Case (T&R): $\Gamma; \Delta \vdash z.\text{case}(P, Q) :: z:A \& B$

(a) $\Gamma; \Delta \vdash P :: z:A$ [Premise of rule (T&R)]

(b) $\Gamma; \Delta \vdash Q :: z:B$ [Premise of rule (T&R)]

(c) $P \in \mathcal{L}[\Gamma; \Delta \vdash z:A]$ [By i.h on (a)]

(d) $Q \in \mathcal{L}[\Gamma; \Delta \vdash z:B]$ [By i.h on (b)]

Pick any $G \in \mathcal{C}_\Gamma$, $D \in \mathcal{C}_\Delta$:

- (e) $G \Downarrow, G \not\rightarrow, D \Downarrow$ [By Prop 4.7]
- (f) $(\nu \tilde{u}, \tilde{x})(P \mid G \mid D) \in \mathcal{L}[z:A]$ [By Lemma 4.8 on (c)]
- (g) $(\nu \tilde{u}, \tilde{x})(Q \mid G \mid D) \in \mathcal{L}[z:B]$ [By Lemma 4.8 on (d)]
- (h) $M = (\nu \tilde{u}, \tilde{x})(z.\text{case}(P, Q) \mid G \mid D) \in \mathcal{L}[z:A \& B]$

We prove (h) by induction on $k = \text{mlen}(D)$: [Possible by (e)]

Base case $k = 0$. Hence, $D \not\rightarrow$:

- (i) $M \xrightarrow{z.\text{inl}} M_1 \wedge M \xrightarrow{z.\text{inr}} M_2$, where :
 $M_1 = (\nu \tilde{u}, \tilde{x})(P \mid G \mid D)$
 $M_2 = (\nu \tilde{u}, \tilde{x})(Q \mid G \mid D)$
- (j) $M_1 \in \mathcal{L}[z:A]$ [Using (f)]
- (k) $M_2 \in \mathcal{L}[z:B]$ [Using (g)]
- $M \in \mathcal{L}[z:A \& B]$ [By Def 4.4, using (i), (j), (k)]

Inductive case $k > 0$:

- Fix the set $W = \{M' \mid M \rightarrow^{k-1} M'\}$
- $\forall M' \in W. M' \in \mathcal{L}[z:A \& B]$ [By i.h.]
- Fix the set $W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$
- $\forall M'' \in W'. M'' \in \mathcal{L}[z:A \& B]$ [By Prop 4.13]
- $z.\text{case}(P, Q) \in \mathcal{L}[\Gamma; \Delta \vdash z:A \& B]$ [By (h) and Lemma 4.8]

16. Case (T \oplus R₁): $\Gamma; \Delta \vdash z.\text{inl}; P :: z:A \oplus B$

- (a) $\Gamma; \Delta \vdash z.\text{inl}; P :: z:A \oplus B$ [Premise of rule (T \oplus R₁)]
- (b) $P \in \mathcal{L}[\Gamma; \Delta \vdash z:A]$ [By i.h on (a)]

Pick any $G \in \mathcal{C}_\Gamma$, $D \in \mathcal{C}_\Delta$:

- (c) $G \Downarrow, G \not\rightarrow, D \Downarrow$ [By Prop 4.7]
- (d) $(\nu \tilde{u}, \tilde{x})(P \mid G \mid D) \in \mathcal{L}[z:A]$ [By Lemma 4.8 on (b)]
- (e) $M = (\nu \tilde{u}, \tilde{x})(z.\text{inl}; P \mid G \mid D) \in \mathcal{L}[z:A \oplus B]$

We prove (e) by induction on $k = \text{mlen}(D)$: [Possible by (c)]

Base case $k = 0$. Hence, $D \not\rightarrow$:

- (i) $M \xrightarrow{z.\text{inl}} (\nu \tilde{u}, \tilde{x})(P \mid G \mid D) = M_1$
- (j) $M_1 \in \mathcal{L}[z:A]$ [Using (d)]
- $M \in \mathcal{L}[z:A \oplus B]$ [Using (i), (j), and Def 4.4]

Inductive case $k > 0$:

Fix the set $W = \{M' \mid M \rightarrow^{k-1} M'\}$

$\forall M' \in W. M' \in \mathcal{L}[z:A \oplus B]$ [By i.h.]

Fix the set $W' = \{M'' \mid M' \rightarrow M'' \wedge M' \in W\}$

$\forall M'' \in W'. M'' \in \mathcal{L}[z:A \oplus B]$ [By Prop 4.13]

$z.\text{inl}; P \in \mathcal{L}[\Gamma; \Delta \vdash z:A \oplus B]$ [By (e) and Lemma 4.8]

17. Case $(\mathbf{T} \oplus \mathbf{R}_2)$: Analogous to case $(\mathbf{T} \oplus \mathbf{R}_1)$.

B Proofs from Section 5 (Typed Context Bisimilarity)

B.1 Proof of Proposition 5.5

We repeat the statement in Page 13. Below, $\#(\Gamma)$, $\#(\Delta)$ denote the cardinality of Γ , Δ , respectively.

Proposition B.1. $\Gamma; \Delta \vdash P \approx Q :: T$ implies $;\cdot \vdash K[P] \approx K[Q] :: T$, for every $K \in \mathcal{K}_{\Gamma, \Delta}$ as in Def. 5.3.

Proof. By induction on $n = \#(\Gamma) + \#(\Delta)$, using the inductive cases in the definition of \approx (Def. 5.2). The base case is when $n = 0$: then both typing environments are empty and, since $K \in \mathcal{K}_{\emptyset, \emptyset} = [\cdot]$, $K[P] = P$ and $K[Q] = Q$. Hence, the thesis trivially holds.

Let us consider the inductive case, when $n > 0$. There are two sub-cases. In the first one, we have $\Gamma, u_i:G_i; \Delta \vdash P \approx Q :: T$. By induction hypothesis, we have that $\Gamma; \Delta \vdash P \approx Q :: T$ implies $;\cdot \vdash K[P] \approx K[Q] :: T$, for every $K \in \mathcal{K}_{\Gamma, \Delta}$. Now, by definition of \approx for non-linear names, $\Gamma, u_i:G_i; \Delta \vdash P \approx Q :: T$ implies $\Gamma; \Delta \vdash (\nu u_i)(!u_i(y_i).S \mid P) \approx (\nu u_i)(!u_i(y_i).S \mid Q) :: T$, for every S such that $\vdash S :: y_i:G_i$. Now, using the induction hypothesis, the latter allows us to infer $;\cdot \vdash K_1[(\nu u_i)(!u_i(y_i).S \mid P)] \approx K_1[(\nu u_i)(!u_i(y_i).S \mid Q)] :: T$, for every $K_1 \in \mathcal{K}_{\Gamma, \Delta}$. We observe that, for any P , $K_1[(\nu u_i)(!u_i(y_i).S \mid P)]$ is equivalent to $K_0[P]$, with $K_0 = (\nu u_i)(!u_i(y_i).S \mid K_1[\cdot])$. Using Prop 5.4, we infer that $K_0 \in \mathcal{K}_{\Gamma, u_i:G_i; \Delta}$. Therefore, $\Gamma, u_i:G_i; \Delta \vdash P \approx Q :: T$ implies $;\cdot \vdash K[P] \approx K[Q] :: T$, for any $K \in \mathcal{K}_{\Gamma, u_i:G_i; \Delta}$, as desired.

In the second sub-case, we have $\Gamma; \Delta, x_j:A_j \vdash P \approx Q :: T$. The analysis follows the same lines as before—exploiting the definition of \approx for linear names and Prop 5.4—and is omitted. \square

B.2 Proof of Proposition B.3

Here we define a typed version of congruence in our setting. Then, we prove that typed context bisimilarity adheres to such a notion.

Definition B.2 (Congruence). A type-respecting relation \mathcal{R} is a typed congruence if and only if

InputR $\Gamma; \Delta, y:A \vdash P \mathcal{R} Q :: x:B$ implies $\Gamma; \Delta \vdash x(y).P \mathcal{R} x(y).Q :: x:A \multimap B$

OutputR $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: y:A$ and $\Gamma; \Delta' \vdash P_2 \mathcal{R} Q_2 :: x:B$ imply
 $\Gamma; \Delta, \Delta' \vdash (\nu y)x\langle y \rangle.(P_1 \parallel P_2) \mathcal{R} (\nu y)x\langle y \rangle.(Q_1 \parallel Q_2) :: x:A \otimes B$
RepInputR $\Gamma; \cdot \vdash P \mathcal{R} Q :: y:A$ implies $\Gamma; \cdot \vdash !x\langle y \rangle.P \mathcal{R} !x\langle y \rangle.Q :: x:A$
CaseR $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: x:A$ and $\Gamma; \Delta \vdash P_2 \mathcal{R} Q_2 :: x:B$ imply
 $\Gamma; \Delta \vdash x.\text{case}(P_1, P_2) \mathcal{R} x.\text{case}(Q_1, Q_2) :: x:A \& B$
ChoiceR1 $\Gamma; \Delta \vdash P \mathcal{R} Q :: x:A$ implies $\Gamma; \Delta \vdash x.\text{inl}; P \mathcal{R} x.\text{inl}; Q :: x:A \oplus B$
ChoiceR2 $\Gamma; \Delta \vdash P \mathcal{R} Q :: x:B$ implies $\Gamma; \Delta \vdash x.\text{inr}; P \mathcal{R} x.\text{inr}; Q :: x:A \oplus B$
Cut $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: x:A$ and $\Gamma; \Delta', x:A \vdash P_2 \mathcal{R} Q_2 :: T$ imply
 $\Gamma; \Delta, \Delta' \vdash (\nu x)(P_1 \parallel P_2) \mathcal{R} (\nu x)(Q_1 \parallel Q_2) :: T$
CutBang $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: y:A$ and $\Gamma, u:A; \Delta \vdash P_2 \mathcal{R} Q_2 :: T$ imply
 $\Gamma; \Delta \vdash (\nu u)(!u\langle y \rangle.P_1 \parallel P_2) \mathcal{R} (\nu u)(!u\langle y \rangle.Q_1 \parallel Q_2) :: T$
InputL $\Gamma; \Delta, y:A, x:B \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x:A \otimes B \vdash x\langle y \rangle.P \mathcal{R} x\langle y \rangle.Q :: T$
OutputL $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: y:A$ and $\Gamma; \Delta', x:B \vdash P_2 \mathcal{R} Q_2 :: T$ imply
 $\Gamma; \Delta, \Delta', x:A \multimap B \vdash (\nu y)x\langle y \rangle.(P_1 \parallel P_2) \mathcal{R} (\nu y)x\langle y \rangle.(Q_1 \parallel Q_2) :: T$
Copy $\Gamma, u:A; \Delta, y:A \vdash P \mathcal{R} Q :: T$ implies $\Gamma, u:A; \Delta \vdash (\nu y)u\langle y \rangle.P \mathcal{R} (\nu y)u\langle y \rangle.Q :: T$
RepInputL $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: y:A$ and $\Gamma, u:A; \Delta' \vdash P_2 \mathcal{R} Q_2 :: T$ imply
 $\Gamma; \Delta, \Delta' \vdash (\nu u)(!u\langle y \rangle.P_1 \parallel P_2) \mathcal{R} (\nu u)(!u\langle y \rangle.Q_1 \parallel Q_2) :: T$
CaseL $\Gamma; \Delta, x:A \vdash P_1 \mathcal{R} Q_1 :: T$ and $\Gamma; \Delta, x:B \vdash P_2 \mathcal{R} Q_2 :: T$ imply
 $\Gamma; \Delta, x:A \oplus B \vdash x.\text{case}(P_1, P_2) \mathcal{R} x.\text{case}(Q_1, Q_2) :: T$
ChoiceL1 $\Gamma; \Delta, x:A \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x:A \& B \vdash x.\text{inl}; P \mathcal{R} x.\text{inl}; Q :: T$
ChoiceL2 $\Gamma; \Delta, x:B \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x:A \& B \vdash x.\text{inr}; P \mathcal{R} x.\text{inr}; Q :: T$

Proposition B.3. \approx is a congruence, in the sense of Def. B.2.

Proof. All the cases proceed by exhibiting the appropriate typed context bisimulations. We consider only the case of **Cut**, and so we show that $\Gamma; \Delta \vdash P_1 \mathcal{R} Q_1 :: x:A$ and $\Gamma; \Delta', x:A \vdash P_2 \mathcal{R} Q_2 :: T$ imply $\Gamma; \Delta, \Delta' \vdash (\nu x)(P_1 \parallel P_2) \mathcal{R} (\nu x)(Q_1 \parallel Q_2) :: T$.

Case Cut: By virtue of Prop 5.5, we can consider directly the following statement:

$$\begin{aligned} & \cdot \vdash K_1[P_1] \mathcal{R} K_1[Q_1] :: x:A \wedge \cdot \vdash K_2[P_2] \mathcal{R} K_2[Q_2] :: T \Rightarrow \\ & \quad \cdot \vdash K[(\nu x)(P_1 \parallel P_2)] \mathcal{R} K[(\nu x)(Q_1 \parallel Q_2)] :: T \end{aligned}$$

for any $K_1 \in \mathcal{K}_{\Gamma; \Delta}$, $K_2 \in \mathcal{K}_{\Gamma; \Delta', x:A}$, and $K \in \mathcal{K}_{\Gamma; \Delta, \Delta'}$. Therefore, we should show that

$$\begin{aligned} \mathcal{R}_{\vdash T} = & \{ (K[(\nu x)(P_1 \parallel P_2)], K[(\nu x)(Q_1 \parallel Q_2)]) : \\ & \cdot \vdash K_1[P_1] \approx K_1[Q_1] :: x:A, \\ & \cdot \vdash K_2[P_2] \approx K_2[Q_2] :: T, \\ & K_1 \in \mathcal{K}_{\Gamma; \Delta}, K_2 \in \mathcal{K}_{\Gamma; \Delta', x:A}, K \in \mathcal{K}_{\Gamma; \Delta, \Delta'} \} \cup \mathcal{W}_{\vdash T} \end{aligned}$$

is a typed context bisimilarity. Let us abbreviate $K[(\nu x)(P_1 \parallel P_2)]$ and $K[(\nu x)(Q_1 \parallel Q_2)]$ as M and N , respectively. By virtue of Prop 5.4, we have both

$$M \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(P_1 \parallel P_2)) \quad N \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(Q_1 \parallel Q_2))$$

where, using I to index over names in Γ and J to index over names in Δ, Δ' , we have $V = \prod_{i \in I} !u_i(y_i).R_i \mid \prod_{j \in J} S_j$, with $\vdash R_i :: y_i:G_i$ and $\vdash S_j :: x_j:C_j$.

Suppose first that M moves first: $M \xrightarrow{\alpha} M'$; we need to find a matching action $N \xrightarrow{\alpha} N'$. Using the typing, we observe that there are two possibilities for α :

1. $\alpha = \tau$, and by subject reduction (Theorem 3.2) $\cdot; \cdot \vdash M' :: T$
2. $\alpha \neq \tau$, and both α and the type of M' depend on the actual shape of type T

We consider case (1) first, and so we assume that $M \xrightarrow{\tau} M'$. We consider the different possibilities for the origin of the reduction:

1. The reduction originates from V . More precisely, by Prop 5.4 the reduction originates in the part of V implementing names in Δ, Δ' , as the part of V implementing names in Γ cannot evolve on its own. Therefore, for some V' , we have both $V \xrightarrow{\tau} V'$ and $M' \equiv (\nu \tilde{u}, \tilde{x})(V' \mid (\nu x)(P_1 \mid P_2))$. By subject reduction (Theorem 3.2), the type of V' is the same than that of V , which in turn means that $\vdash M' :: T$. Since V occurs identically in M and N , this reduction can be trivially matched by N , and so we have that $N \Longrightarrow N'$, with $\vdash N' :: T$, also by subject reduction. Notice that by virtue of Theorem 4.17 this weak transition is finite. Therefore, the pair (M', N') is in $\mathcal{R}_{\vdash T}$, and we are done.
2. The reduction originates from P_1 . Then, for some P'_1 , we have $P_1 \xrightarrow{\tau} P'_1$ and $M' \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(P'_1 \mid P_2))$. By subject reduction (Theorem 3.2), the type remains unchanged, which in turn means that $\vdash M' :: T$. We then exploit the fact that $\cdot; \cdot \vdash K_1[P_1] \approx K_1[Q_1] :: x:A$ to infer that N can match this reduction through an independent reduction of Q_1 . That is, by virtue of Theorem 4.17 there is a finite weak transition $Q_1 \Longrightarrow Q'_1$ such that $N \Longrightarrow N' \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(Q'_1 \mid Q_2))$. By subject reduction we infer $\vdash N' :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_{\vdash T}$, and we are done.
3. The reduction originates from P_2 . We proceed analogously as in the previous case, relying on the assumption $\cdot; \cdot \vdash K_2[P_2] \approx K_2[Q_2] :: T$.
4. The reduction originates from the interaction of P_1 and K . Therefore, for some V', P'_1 , we have $M' \equiv (\nu \tilde{u}, \tilde{x})(V' \mid (\nu x)(P'_1 \mid P_2))$. By subject reduction (Theorem 3.2), we can infer that $\vdash M' :: T$. Since V occurs identically in M and N , and $\cdot; \cdot \vdash K_1[P_1] \approx K_1[Q_1] :: x:A$, we infer that this interaction can be matched by N . Hence, there is a weak transition $N \Longrightarrow N' \equiv (\nu \tilde{u}, \tilde{x})(V' \mid (\nu x)(Q'_1 \mid Q_2))$. By Theorem 4.17 such a transition is finite; subject reduction (Theorem 3.2) ensures that $\vdash N' :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_{\vdash T}$, and we are done.
5. The reduction originates from the interaction of P_2 and K . We proceed analogously as in the previous case, relying on $\cdot; \cdot \vdash K_2[P_2] \approx K_2[Q_2] :: T$.
6. The reduction originates from the interaction of P_1 and P_2 . Therefore, $M' \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(P'_1 \mid P'_2))$. Using the typings of each process, we infer that this interaction is only possible via a synchronization on x , which offers (case of P_1) and requires (case of P_2) a behavior described by A . We then proceed by structural induction on the type A . All the cases are covered by the preservation lemmas, which formalize the interaction of complementary actions. Take, for instance, the case $A = T_1 \otimes T_2$. Using Lemma A.1 we infer $P_1 \xrightarrow{(\nu y)x(y)} P'_1$ and $P_2 \xrightarrow{x(y)} P'_2$. Using Lemma A.2 we infer that P' is well-typed: $\vdash (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(P'_1 \mid P'_2)) :: T$. Since $K_1[P_1] \approx K_1[Q_1]$ and $K_2[P_2] \approx K_2[Q_2]$, we know that these actions

can be matched by Q_1 and Q_2 : $Q_1 \xrightarrow{(\nu y)x(y)} Q'_1$ and $Q_2 \xrightarrow{x(y)} Q'_2$. By virtue of Theorem 4.17 these are finite weak transitions. That is, there is an N' such that $N' \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(Q'_1 \mid Q'_2))$. Using again Lemma A.2 and subject reduction (Theorem 3.2), one can show that N' is well-typed: $\vdash (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(Q'_1 \mid Q'_2)) :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_{\vdash T}$ and we are done.

Now we consider case (2), and so we assume $M \xrightarrow{\alpha} M'$, for some $\alpha \neq \tau$. The shape of α depends on the structure of T ; the typing information ensures that T can only be provided by P_2 . Therefore, we proceed by induction on the structure of the type T . We consider only the case $T = x:A_1 \otimes A_2$; the other cases are similar or simpler. Then, by Lemma A.1, $\alpha = (\nu y)x(y)$ and $M' \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(P_1 \mid P'_2))$. Since $K_2[P_2] \approx K_2[Q_2]$, we know that this action can be matched by Q_2 : indeed we have $Q_2 \xrightarrow{(\nu y)x(y)} Q'_2$. By virtue of Theorem 4.17 this is a finite weak transition. There is then an N' such that $N' \equiv (\nu \tilde{u}, \tilde{x})(V \mid (\nu x)(Q_1 \mid Q'_2))$. Now we follow the definition of \approx for output actions. Then, for any R such that $\cdot; y:A_1 \vdash R :: -:1$, we verify that both $\vdash (\nu y)(M' \mid R) :: x:A_2$ and $\vdash (\nu y)(N' \mid R) :: x:A_2$ hold. Consequently, the pair $((\nu y)(M' \mid R), (\nu y)(N' \mid R)) \in \mathcal{W}_{\vdash x:A_1 \otimes A_2}$, and we are done.

The case in which $N \xrightarrow{\alpha} N'$ moves first is completely symmetric. \square

C Proofs from Section 6 (Applications)

C.1 Full list of proof conversions

Figures 4–7 show proof conversions that involve the interaction of left and right rules with cut and cut¹ rules. For the other proof conversions, see Figure 8, Page 48.

Definition C.1. We define \simeq_c as the least congruence on derivations generated by the commuting conversions in the right hand side of Figures 4, 5, 6, and 7. We extend \simeq_c on processes as the congruence generated by the process equations on the left hand side of the such figures, plus the equations in Figure 8.

Proposition C.2. Let $\Gamma; \Delta \vdash D \rightsquigarrow P :: T$ with $D \simeq_c E$. Then there is Q such that $\Gamma; \Delta \vdash E \rightsquigarrow Q :: T$ and $P \simeq_c Q$.

Proof. Immediate from Figures 4, 5, 6, 7, and 8.

C.2 Additional cases for the proof of Proposition 6.2

We repeat the statement in Page 15:

Proposition C.3. Assume (i) $\Gamma; \Delta \vdash D \rightsquigarrow P :: T$, (ii) $\Gamma; \Delta \vdash E \rightsquigarrow Q :: T$, and (iii) $P \simeq_c Q$. Then, $\Gamma; \Delta \vdash P \approx Q :: T$

Proof. By coinduction, exhibiting appropriate typed context bisimulations for each commuting conversion. In the bisimulation game, we exploit termination of well-typed

$\text{cut } D(x. \mathbf{1L} y E_x)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid \hat{E})$
$\mathbf{1L} y(\text{cut } \tilde{D}(x. E_x))$	\rightsquigarrow	$(\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E})$
$\text{cut } D(x. \otimes R E_x F)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F}))$
$\otimes R(\text{cut } \tilde{D}(x. E_x)) F$	\rightsquigarrow	$(\nu y)z\langle y \rangle.((\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E}) \mid \hat{F})$
$\text{cut } D(x. \otimes R E F_x)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F}))$
$\otimes R E(\text{cut } \tilde{D}(x. F_x))$	\rightsquigarrow	$(\nu y)z\langle y \rangle.(\hat{E} \mid (\nu x)(\overset{\sim}{\hat{D}} \mid \hat{F}))$
$\text{cut } D(x. \otimes L y(z.y. E_{xzy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid y(z).\hat{E})$
$\otimes L y(z.y. \text{cut } \tilde{D}(x. E_{xzy}))$	\rightsquigarrow	$y(z).(\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E})$
$\text{cut } D(x. \multimap R(y. E_{xy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid z(y).\hat{E})$
$\multimap R(y. \text{cut } \tilde{D}(x. E_{xy}))$	\rightsquigarrow	$z(y).(\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E})$
$\text{cut } D(x. \multimap L y E_x(y. F_y))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid (\nu z)y\langle z \rangle.(\hat{E} \mid \hat{F}))$
$\multimap L y(\text{cut } \tilde{D}(x. E_x))(y. F_y)$	\rightsquigarrow	$(\nu z)y\langle z \rangle.((\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E}) \mid \hat{F})$
$\text{cut } D(x. \multimap L y E(y. F_{xy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid (\nu z)y\langle z \rangle.(\hat{E} \mid \hat{F}))$
$\multimap L y E(y. \text{cut } \tilde{D}(x. F_{xy}))$	\rightsquigarrow	$(\nu z)y\langle z \rangle.(\hat{E} \mid (\nu x)(\overset{\sim}{\hat{D}} \mid \hat{F}))$
$\text{cut } D(x. \& R E_x F_x)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid z.\text{case}(\hat{E}, \hat{F}))$
$\& R(\text{cut } \tilde{D}(x. E_x))(\text{cut } D(x. F_x))$	\rightsquigarrow	$z.\text{case}((\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E}), (\nu x)(\hat{D} \mid \hat{F}))$
$\text{cut } D(x. \& L_1 y(y. E_{xy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid y.\text{inl}; \hat{E})$
$\& L_1 y(y. \text{cut } \tilde{D}(x. E_{xy}))$	\rightsquigarrow	$y.\text{inl}; (\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E})$
$\text{cut } D(x. \& L_2 y(y. F_{xy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid y.\text{inr}; \hat{F})$
$\& L_2 y(y. \text{cut } \tilde{D}(x. F_{xy}))$	\rightsquigarrow	$y.\text{inr}; (\nu x)(\overset{\sim}{\hat{D}} \mid \hat{F})$

Fig. 4. Commuting Conversions (Part 1: 1–10)

processes (Theorem 4.17) to ensure that actions can be matched with finite weak transitions, and Theorem 3.2 to ensure preservation of type under reductions. We detail the cases of proof conversions No. 2 and 4 (cf. Figure 4), and No. 35 (cf. Figure 7).

Proof conversion No. 2. We then have that

$$\Gamma; \Delta \vdash \text{cut } D(x. \otimes R E_x F) \rightsquigarrow M = (\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F})) :: z:A \otimes B$$

$$\Gamma; \Delta \vdash \otimes R(\text{cut } \tilde{D}(x. E_x)) F \rightsquigarrow N = (\nu y)z\langle y \rangle.((\nu x)(\overset{\sim}{\hat{D}} \mid \hat{E}) \mid \hat{F}) :: z:A \otimes B$$

with

$$\Gamma; \Delta_1 \vdash \hat{D} :: x:C \quad \Gamma; \Delta_2, x:C \vdash \hat{E} :: y:A \quad \Gamma; \Delta_3 \vdash \hat{F} :: z:B \quad (2)$$

and $\Delta = \Delta_1, \Delta_2, \Delta_3$. We show that $M \simeq_c N$ implies $\Gamma; \Delta \vdash M \approx N :: z:A \otimes B$.

By virtue of Prop. 5.5, we have to show that for every $K \in \mathcal{K}_{\Gamma, \Delta}$, we have $\cdot; \cdot \vdash K[M] \approx K[N] :: z:A \otimes B$. In turn, this implies exhibiting a typed context bisimilarity

$\text{cut } D(x. \oplus R_1 E_x)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid z.\text{inl}; \hat{E})$
$\oplus R_1(\text{cut } \tilde{D}(x. E_x))$	\rightsquigarrow	$z.\text{inl}; (\nu \hat{x})(\hat{D} \mid \hat{E})$
$\text{cut } D(x. \oplus R_2 F_x)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid z.\text{inr}; \hat{F})$
$\oplus R_2(\text{cut } \tilde{D}(x. F_x))$	\rightsquigarrow	$z.\text{inr}; (\nu \hat{x})(\hat{D} \mid \hat{F})$
$\text{cut } D(x. \oplus L y(y. E_{xy})(y. F_{xy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid y.\text{case}(\hat{E}, \hat{F}))$
$\oplus L y(y. \text{cut } \tilde{D}(x. E_{xy}))(y. \text{cut } \tilde{D}(x. F_{xy}))$	\rightsquigarrow	$y.\text{case}((\nu x)(\hat{D} \mid \hat{E}), (\nu x)(\hat{D} \mid \hat{F}))$
$\text{cut } D(x. !L y(u. E_{xu}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid \hat{E}\{y/u\})$
$!L y(u. \text{cut } \tilde{D}(x. E_{xu}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid \hat{E})\{y/u\}$
$\text{cut } D(x. \text{copy } u.(y. E_{xy}))$	\rightsquigarrow	$(\nu x)(\hat{D} \mid (\nu y)u\langle y \rangle. \hat{E})$
$\text{copy } u(y. \text{cut } \tilde{D}(x. E_{xy}))$	\rightsquigarrow	$(\nu y)u\langle y \rangle. (\nu x)(\hat{D} \mid \hat{E})$
$\text{cut } (1L y D)(x. F_x)$	\rightsquigarrow	$(\nu x)(\hat{D} \mid \hat{F})$
$1L y(\text{cut } \tilde{D}(x. F_x))$	\rightsquigarrow	$(\nu x)\tilde{D} \mid \hat{F}$
$\text{cut } (\otimes L y(z.y. D_{yz}))(x. F_x)$	\rightsquigarrow	$(\nu x)(y(z).\hat{D} \mid \hat{F})$
$\otimes L y(z.y. \text{cut } \tilde{D}_{yz}(x. F_x))$	\rightsquigarrow	$y(z).(\nu \hat{x})(\hat{D} \mid \hat{F})$
$\text{cut } (\multimap L y D(y. E_y))(x. F_x)$	\rightsquigarrow	$(\nu x)((\nu z)y\langle z \rangle. (\hat{D} \mid \hat{E}) \mid \hat{F})$
$\multimap L y D(y. \text{cut } \tilde{E}_y(x. F_x))$	\rightsquigarrow	$(\nu z)y\langle z \rangle. (\hat{D} \mid (\nu x)(\hat{E} \mid \hat{F}))$
$\text{cut } (\& L_1 y(y. D_y))(x. F_x)$	\rightsquigarrow	$(\nu x)(y.\text{inl}; \hat{D} \mid \hat{F})$
$\& L_1 y(y. \text{cut } \tilde{D}_y(x. F_x))$	\rightsquigarrow	$y.\text{inl}; (\nu \hat{x})(\hat{D} \mid \hat{F})$
$\text{cut } (\& L_2 y(y. D_y))(x. F_x)$	\rightsquigarrow	$(\nu x)(y.\text{inr}; \hat{D} \mid \hat{F})$
$\& L_2 y(y. \text{cut } \tilde{D}_y(x. F_x))$	\rightsquigarrow	$y.\text{inr}; (\nu \hat{x})(\hat{D} \mid \hat{F})$

Fig. 5. Commuting Conversions (Part 2: 11–20)

\mathcal{R} containing the pair $(K[M], K[N])$. We thus define \mathcal{R} as :

$$\mathcal{R} = \mathcal{W}_{\vdash z:A \otimes B} \cup \mathcal{S} \cup \mathcal{S}^{-1} \text{ where:}$$

$$\mathcal{S} = \{(K_1[M'], K_2[N]) : M \Longrightarrow M', K_1, K_2 \in \mathcal{K}_{\Gamma; \Delta}\}$$

and $\mathcal{W}_{\vdash z:A \otimes B}$ is as in Def 6.1. Notice that \mathcal{S} is a type-respecting relation indexed by $\vdash z:A \otimes B$. In fact, using the typings in (2)—with $\Gamma = \Delta = \emptyset$ —and exploiting subject reduction (Theorem 3.2), it can be checked that for all $(P, Q) \in \mathcal{S}$ both $\vdash P :: z:A \otimes B$ and $\vdash Q :: z:A \otimes B$ can be derived.

We now show that \mathcal{R} is a typed context bisimilarity. Pick any $K \in \mathcal{K}_{\Gamma; \Delta}$. Using Prop 5.4, we can assume

$$K = (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid [\cdot])$$

$$\begin{array}{c}
\text{cut}(\oplus L y(y.D_y)(y.E_y))(x.F_x) \rightsquigarrow (\nu x)(y.\text{case}(\hat{D}, \hat{E}) | \hat{F}) \\
\oplus L y(y.\text{cut} D_y(x.\tilde{F}_x))(y.\text{cut} E_y(x.F_x)) \rightsquigarrow y.\text{case}((\nu x)(\hat{D} | \overset{\sim}{\hat{F}}), (\nu x)(\hat{E} | \hat{F})) \\
\text{cut}(!L y(u.D_u))(x.F_x) \rightsquigarrow (\nu x)(\hat{D}\{y/u\} | \hat{F}) \\
!L y(u.\text{cut} \tilde{D}_u(x.F_x)) \rightsquigarrow (\nu x)(\hat{D} | \overset{\sim}{\hat{F}_x}\{y/u\}) \\
\text{cut}(\text{copy } u.(y.D_y))(x.F_x) \rightsquigarrow (\nu x)((\nu y)u\langle y \rangle.\hat{D} | \hat{F}) \\
\text{copy } u(y.\text{cut} \tilde{D}_y(x.F_x)) \rightsquigarrow (\nu y)u\langle y \rangle.\overset{\sim}{(\nu x)(\hat{D} | \hat{F})} \\
\text{cut}^! D(u.\mathbf{1R}) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | \mathbf{0}) \\
\overset{\sim}{\mathbf{1R}} \rightsquigarrow \overset{\sim}{\mathbf{0}} \\
\text{cut}^! D(u.\mathbf{1L} y E_u) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | \hat{E}) \\
\mathbf{1L} y(\text{cut}^! \tilde{D}(u.E_u)) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | \hat{E}) \\
\text{cut}^! D(u.\otimes R E_u F_u) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | (\nu z)(x\langle z \rangle.(\hat{E} | \hat{F}))) \\
\otimes R(\text{cut}^! D(u.E_u))(\text{cut}^! D(u.F_u)) \rightsquigarrow (\nu z)(x\langle z \rangle.((\nu u)((!u(y).\hat{D}) | \hat{E}) | (\nu u)((!u(y).\hat{D}) | \hat{F}))) \\
\text{cut}^! D(u.\otimes L y(z.y.E_{uzy})) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | y\langle z \rangle.\hat{E}) \\
\otimes L y(z.y.\text{cut}^! \tilde{D}(u.E_{uzy})) \rightsquigarrow y\langle z \rangle.(\nu u)((!u(y).\hat{D}) | \hat{E}) \\
\text{cut}^! D(u.\text{--}\circ R(y.E_{uy})) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | z\langle y \rangle.\hat{E}) \\
\text{--}\circ R(y.\text{cut}^! \tilde{D}(u.E_{uy})) \rightsquigarrow z\langle y \rangle.(\nu u)((!u(y).\hat{D}) | \hat{E}) \\
\text{cut}^! D(u.\text{--}\circ L y E_u(y.F_{uy})) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | (\nu z)y\langle z \rangle.(\hat{E} | \hat{F})) \\
\text{--}\circ L(\text{cut}^! D(u.E_u))(y.\text{cut}^! D(u.F_{uy})) \rightsquigarrow (\nu z)y\langle z \rangle.(((\nu u)((!u(y).\hat{D}) | \hat{E}) | (\nu u)((!u(y).\hat{D}) | \hat{F}))) \\
\text{cut}^! D(u.\& R E_u F_u) \rightsquigarrow (\nu u)((!u(y).\hat{D}) | z.\text{case}(\hat{E}, \hat{F})) \\
\& R(\text{cut}^! D(u.E_u))(\text{cut}^! D(u.F_u)) \rightsquigarrow z.\text{case}((\nu u)((!u(y).\hat{D}) | \hat{E}), (\nu u)((!u(y).\hat{D}) | \hat{F}))
\end{array}$$

Fig. 6. Commuting Conversions (Part 3: 21–30)

where

- $K_\Gamma \equiv \prod_{i \in I} !u_i(y_i).R_i$, with $\vdash R_i :: y_i:D_i$, for every $u_i:D_i \in \Gamma$;
- $K_\Delta \equiv \prod_{j \in J} S_j$, with $\vdash S_j :: x_j:C_j$, for every $x_j:C_j \in \Delta$.

Clearly, $(K[M], K[N]) \in \mathcal{S}$, and so it is in \mathcal{R} . Now, suppose $K[M]$ moves first: $K[M] \xrightarrow{\alpha} M_1^*$. We have to find a matching action α from $K[N]$, i.e., $K[N] \xrightarrow{\alpha} N_1^*$. Since $\vdash K[M] :: z:A \otimes B$, we have two possible cases for α :

1. Case $\alpha = \tau$. We consider the possibilities for the origin of the reduction:
 - (a) $K_\Gamma \xrightarrow{\tau} K'_\Gamma$ and $K[M] \xrightarrow{\tau} K'[M]$. However, this cannot be the case, as by construction K_Γ corresponds to the parallel composition of input-guarded replicated processes which cannot evolve on their own.

$\text{cut}^! D(u. \&L_1 y(y. E_{uy}))$	\rightsquigarrow	$(\nu u)((!u(z).\hat{D}) \mid y.\text{inl}; \hat{E})$
$\&L_1 y(y. \text{cut}^! D(u. E_{uy}))$	\rightsquigarrow	$y.\text{inl}; (\nu u)((!u(z).\hat{D}) \mid \hat{E})$
$\text{cut}^! D(u. \&L_2 y(y. F_{uy}))$	\rightsquigarrow	$(\nu u)((!u(z).\hat{D}) \mid y.\text{inr}; \hat{F})$
$\&L_2 y(y. \text{cut}^! D(u. F_{uy}))$	\rightsquigarrow	$y.\text{inr}; (\nu u)((!u(z).\hat{D}) \mid \hat{F})$
$\text{cut}^! D(u. \oplus R_1 E_u)$	\rightsquigarrow	$(\nu u)((!u(y).\hat{D}) \mid z.\text{inl}; \hat{E})$
$\oplus R_1 (\text{cut}^! D(u. E_u))$	\rightsquigarrow	$z.\text{inl}; (\nu u)((!u(y).\hat{D}) \mid \hat{E})$
$\text{cut}^! D(u. \oplus R_2 F_u)$	\rightsquigarrow	$(\nu u)((!u(y).\hat{D}) \mid z.\text{inr}; \hat{F})$
$\oplus R_2 (\text{cut}^! D(u. F_u))$	\rightsquigarrow	$z.\text{inr}; (\nu u)((!u(y).\hat{D}) \mid \hat{F})$
$\text{cut}^! D(u. \oplus L y(y. E_{uy})(y. F_{uy}))$	\rightsquigarrow	$(\nu u)((!u(z).\hat{D}) \mid y.\text{case}(\hat{E}, \hat{F}))$
$\oplus L y(y. \text{cut}^! D(u. E_{uy}))(y. \text{cut}^! D(u. F_{uy}))$	\rightsquigarrow	$y.\text{case}((\nu u)((!u(z).\hat{D}) \mid \hat{E}), (\nu u)((!u(z).\hat{D}) \mid \hat{F}))$
$\text{cut}^! D(u. !R E_u)$	\rightsquigarrow	$(\nu u)((!u(y).\hat{D}) \mid !x(z).\hat{E})$
$!R (\text{cut}^! D(u. E_u))$	\rightsquigarrow	$!x(z).(\nu u)((!u(y).\hat{D}) \mid \hat{E})$
$\text{cut}^! D(u. !L y(v. E_{uv}))$	\rightsquigarrow	$(\nu u)((!u(y).\hat{D}) \mid \hat{E}\{y/v\})$
$!L y(v. \text{cut}^! D(u. E_{uv}))$	\rightsquigarrow	$(\nu u)((!u(y).\hat{D}) \mid \hat{E})\{y/v\}$
$\text{cut}^! D(u. \text{copy } v(y. E_{uy}))$	\rightsquigarrow	$(\nu u)((!u(y).\hat{D}) \mid (\nu y)v\langle y \rangle.\hat{E})$
$\text{copy } v(y. \text{cut}^! D(u. E_{uy}))$	\rightsquigarrow	$(\nu y)v\langle y \rangle.(\nu u)((!u(y).\hat{D}) \mid \hat{E})$

Fig. 7. Commuting Conversions (Part 4: 31–38)

(b) $K_\Delta \xrightarrow{\tau} K'_\Delta$ and $K[M] \xrightarrow{\tau} K'[M]$. Then, for some $l \in J$, $S_l \xrightarrow{\tau} S'_l$:

$$K[M] \xrightarrow{\tau} (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K'_\Delta \mid M) = K'[M] = M_1^*$$

Now, context K is the same in $K[N]$. Then K_Δ occurs identically in $K[N]$, and this reduction can be matched by a finite weak transition:

$$K[N] \Longrightarrow (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K''_\Delta \mid N) = K''[N] = N_1^*$$

By subject reduction (Theorem 3.2), $\vdash S'_l :: x_l:C_l$; hence, K', K'' are in $\mathcal{K}_{\Gamma, \Delta}$. Hence, the pair $(K'[M], K''[N])$ is in \mathcal{S} (as $M \Longrightarrow N$) and so it is in \mathcal{R} .

(c) $M \xrightarrow{\tau} M'$ and $K[M] \xrightarrow{\tau} K[M']$. Since $M = (\nu x)(\hat{D} \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F}))$, the only possibility is that there is a \hat{D}_1 such that $\hat{D} \xrightarrow{\tau} \hat{D}_1$ and $M' = (\nu x)(\hat{D}_1 \mid (\nu y)z\langle y \rangle.(\hat{E} \mid \hat{F}))$. This way,

$$K[M] \xrightarrow{\tau} (\nu \tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid M') = K[M'] = M_1^*$$

We observe that $K[N]$ cannot match this action, but $K[N] \Longrightarrow K[N]$ is a valid weak transition. Hence, $N_1^* = K[N]$. By subject reduction (Theorem 3.2), we infer that $\vdash K[M'] :: z:A \otimes B$. We use this fact to observe that the pair $(K[M'], K[N])$ is included in \mathcal{S} . Hence, it is in \mathcal{R} .

- (d) There is an interaction between M and K_Γ or between M and K_Δ : this is only possible by the interaction of \hat{D} with K_Γ or K_Δ on names in \tilde{u}, \tilde{x} . Again, the only possible weak transition from $K[N]$ matching this reduction is $K[N] \Longrightarrow K[N]$, and the analysis proceeds as in the previous case.
2. Case $\alpha \neq \tau$. Then the only possibility, starting from $K[M]$, is an output action of the form $\alpha = (\nu y)z\langle y \rangle$. This action can only originate in M :

$$K[M] \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{x}, \tilde{u})(K_\Gamma \mid K_\Delta \mid (\nu x)(\hat{D} \mid (\nu y)(\hat{E} \mid \hat{F}))) = M_1^*$$

Process $K[N]$ can match this action via the following finite (weak) transition:

$$K[N] \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{x}, \tilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D}' \mid \hat{E}') \mid \hat{F}')) = N_1^*$$

Observe how N_1^* reflects the changes in $K[N]$ due to the possible reductions before and after the output action. By definition of \approx (output case), we consider the composition of M_1^* and N_1^* with any V such that $y:A \vdash V :: -:1$. Using the typings in (2) and subject reduction (Theorem 3.2), we infer both

$$\begin{aligned} \vdash M_2^* &= (\nu \tilde{x}, \tilde{u})(K_\Gamma \mid K_\Delta \mid (\nu x)(\hat{D} \mid (\nu y)(\hat{E} \mid V \mid \hat{F}))) :: z:B \\ \vdash N_2^* &= (\nu \tilde{x}, \tilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D}' \mid \hat{E}' \mid V) \mid \hat{F}')) :: z:B \end{aligned}$$

Hence, the pair (M_2^*, N_2^*) is in $\mathcal{W}_{\vdash z:A \otimes B}$ and so it is in \mathcal{R} .

Now, let us suppose that $K[N]$ moves first: $K[N] \xrightarrow{\alpha} N_1^*$. We have to find a matching action α from $K[M]$: $K[M] \xrightarrow{\alpha} M_1^*$. Similarly as before, there are two cases: either $\alpha = \tau$ or $\alpha = (\nu y)z\langle y \rangle$. The former is as detailed before; the only difference is that reductions from $K[N]$ can only be originated in K_Δ ; these are matched by $K[M]$ with weak transitions originating in both K and in M . We therefore obtain pairs of processes in \mathcal{S}^{-1} .

We now detail the case in which $\alpha = (\nu y)z\langle y \rangle$. We have:

$$K[N] \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{x}, \tilde{u})(K_\Gamma \mid K_\Delta \mid (\nu y)((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F})) = N_1^*$$

and this action can be matched by $K[M]$ with a finite weak transition:

$$K[M] \xrightarrow{(\nu y)z\langle y \rangle} (\nu \tilde{x}, \tilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu x)(\hat{D}' \mid (\nu y)(\hat{E}' \mid \hat{F}')))) = M_1^*$$

where M_1^* takes into account the possible reductions before and after the output action. As before, we consider the composition of N_1^* and M_1^* with any V such that $y:A \vdash V :: -:1$. Using (2), we can infer both

$$\begin{aligned} \vdash N_2^* &= (\nu \tilde{x}, \tilde{u})(K_\Gamma \mid K_\Delta \mid (\nu y)((\nu x)(\hat{D} \mid \hat{E} \mid V) \mid \hat{F})) :: z:B \\ \vdash M_2^* &= (\nu \tilde{x}, \tilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu x)(\hat{D}' \mid (\nu y)(\hat{E}' \mid V \mid \hat{F}')))) :: z:B \end{aligned}$$

Hence, the pair (N_2^*, M_2^*) is in $\mathcal{W}_{\vdash z:A \otimes B}$ and so it is in \mathcal{R} . This concludes the proof for this case.

Proof conversion No. 4. We then have that

$$\begin{aligned} \Gamma; \Delta, y:A \otimes B \vdash \text{cut } D(x. \otimes \mathbb{L} y(z.y. E_{xzy})) &\rightsquigarrow M = (\nu x)(\hat{D} \mid y(z).\hat{E}) :: T \\ \Gamma; \Delta, y:A \otimes B \vdash \otimes \mathbb{L} y(z.y. \text{cut } D(x. E_{xzy})) &\rightsquigarrow N = y(z).(\nu x)(\hat{D} \mid \hat{E}) :: T \end{aligned}$$

with

$$\Gamma; \Delta_1 \vdash \hat{D} :: x:C \quad \Gamma; \Delta_2, x:C, z:A, y:B \vdash \hat{E} :: T \quad (3)$$

and $\Delta = \Delta_1, \Delta_2$. We show that $M \simeq_c N$ implies $\Gamma; \Delta, y:A \otimes B \vdash M \approx N :: T$.

By virtue of Prop. 5.5, we have to show that for every $K \in \mathcal{K}_{\Gamma; \Delta, y:A \otimes B}$, we have $\cdot; \cdot \vdash K[M] \approx K[N] :: T$. In turn, this implies exhibiting a typed context bisimilarity \mathcal{R} containing the pair $(K[M], K[N])$. We thus define \mathcal{R} as

$$\mathcal{R} = \mathcal{I}_{\vdash T} \cup \mathcal{W}_{\vdash T}$$

recalling that $\mathcal{I}_{\Gamma; \Delta \vdash T}$ stands for the relation $\{(P, Q) : \Gamma; \Delta \vdash P :: T, \Gamma; \Delta \vdash Q :: T\}$. We show that \mathcal{R} is a typed context bisimilarity. Pick any $K \in \mathcal{K}_{\Gamma; \Delta, y:A \otimes B}$. Using Prop 5.4, we can assume

$$K = (\nu \tilde{u}, \tilde{x}, y)(K_\Gamma \mid K_\Delta \mid V \mid [\cdot])$$

where

- $K_\Gamma \equiv \prod_{i \in I} !u_i(y_i).R_i$, with $\vdash R_i :: y_i:G_i$, for every $u_i:G_i \in \Gamma$;
- $K_\Delta \equiv \prod_{j \in J} S_j$, with $\vdash S_j :: x_j:C_j$, for every $x_j:C_j \in \Delta$;
- $\vdash V :: y:A \otimes B$.

Clearly, $(K[M], K[N]) \in \mathcal{L}_T$, and so it is in \mathcal{R} . Now, suppose $K[M]$ moves first: $K[M] \xrightarrow{\alpha} M_1^*$. We have to find a matching action α from $K[N]$, i.e., $K[N] \xrightarrow{\alpha} N_1^*$. We consider two possible cases:

1. Case $\alpha = \tau$. We consider the possibilities for the origin of the reduction:
 - (a) $K_\Gamma \xrightarrow{\tau} K'_\Gamma$: This cannot be the case, as by construction this process correspond to the parallel composition of zero or more input guarded replications which cannot evolve on their own.
 - (b) $K_\Delta \xrightarrow{\tau} K'_\Delta$ and $K[M] \xrightarrow{\tau} (\nu \tilde{u}, \tilde{x}, y)(K_\Gamma \mid K'_\Delta \mid V \mid M) = M_1^*$. Since K_Δ occurs identically in both processes, this reduction can be matched by $K[N]$ with a finite weak transition:

$$K[N] \Longrightarrow (\nu \tilde{u}, \tilde{x}, y)(K_\Gamma \mid K''_\Delta \mid V' \mid M') = N_1^*$$

Using subject reduction (Theorem 3.2) it can be shown that $K', K'' \in \mathcal{K}_{\Gamma; \Delta, y:A \otimes B}$, and that V' and M' preserve the type of V and M , respectively. Hence, both $\vdash M_1^* :: T$ and $\vdash N_1^* :: T$ hold, and the pair (M_1^*, N_1^*) is in \mathcal{L}_T and so it is in \mathcal{R} .

- (c) $V \xrightarrow{\tau} V'$ and $K[M] \xrightarrow{\tau} (\nu \tilde{u}, \tilde{x}, y)(K_\Gamma \mid K_\Delta \mid V' \mid M) = M_1^*$. This case proceeds similarly as the previous one, as V occurs in both processes.

- (d) $M \xrightarrow{\tau} M'$ and $K[M] \xrightarrow{\tau} (\nu\tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid V \mid M') = M_1^*$. Since $M = (\nu x)(\hat{D} \mid y(z).\hat{E})$, the only possibility is that there is a \hat{D}_1 such that $\hat{D} \xrightarrow{\tau} \hat{D}_1$ and $M' = (\nu x)(\hat{D}_1 \mid y(z).\hat{E})$. This way,

$$K[M] \xrightarrow{\tau} (\nu\tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid V \mid M') = K[M'] = M_1^*$$

We observe that $K[N]$ cannot match this action, as \hat{D} is behind a prefix. Nevertheless, $K[N] \Longrightarrow K[N]$ is a valid weak transition, and so $N_1^* = K[N]$. By subject reduction (Theorem 3.2), we infer that $\vdash K[M'] :: T$. Hence, the pair (M_1^*, N_1^*) is included in \mathcal{L}_T , and so it is in \mathcal{R} .

- (e) The reduction arises from the interaction of V and M . This can only correspond to a synchronization on y . We have:

$$K[M] \xrightarrow{\tau} (\nu\tilde{u}, \tilde{x})(K_\Gamma \mid K_\Delta \mid (\nu y)(V' \mid (\nu x)(\hat{D} \mid \hat{E}\sigma))) = M_1^*$$

where σ stands for the substitution derived from the synchronization. This reduction can be matched by $K[N]$ via a finite weak transition:

$$K[N] \Longrightarrow (\nu\tilde{u}, \tilde{x})(K_\Gamma \mid K'_\Delta \mid (\nu y)(V' \mid (\nu x)(\hat{D}' \mid \hat{E}'\sigma))) = N_1^*$$

where N_1^* captures the fact that internal actions could have occurred before and after the synchronization on y . By subject reduction (Theorem 3.2), typing is preserved in both cases, and so $(M_1^*, N_1^*) \in \mathcal{R}$.

2. Case $\alpha \neq \tau$. Then α corresponds to the execution of some behavior described by T , in the right-hand side typing. However, this cannot be the case since, as specified by the typings in (3), the behavior described by T can only be provided by \hat{E} , which is behind an input prefix on y , both in $K[M]$ and $K[N]$. Therefore, behavior described by T cannot be exercised until such a prefix is consumed, and we have that, necessarily, $\alpha = \tau$. Observe that once such prefixes are consumed (via internal actions) the evolution corresponding to the behavior described by T is still in \mathcal{R} , as the continuation relation $\mathcal{W}_{\vdash T}$ is in \mathcal{R} .

The analysis when $K[N]$ moves first follows the same lines and is omitted.

Proof conversion No. 35. We then have that

$$\begin{aligned} \Gamma; \Delta, y:A \oplus B \vdash \text{cut}^! D(u. \oplus L y(y. E_{uy})(y. F_{uy})) &\rightsquigarrow \\ M = (\nu u)((!u(z).\hat{D}) \mid y.\text{case}(\hat{E}, \hat{F})) &:: T \\ \Gamma; \Delta, y:A \oplus B \vdash \oplus L y(y. \text{cut}^! D(u. E_{uy})) (y. \text{cut}^! D(u. F_{uy})) &\rightsquigarrow \\ N = y.\text{case}((\nu u)((!u(z).\hat{D}) \mid \hat{E}), (\nu u)((!u(z).\hat{D}) \mid \hat{F})) &:: T \end{aligned}$$

with

$$\Gamma; \cdot \vdash \hat{D} :: z:C \quad \Gamma, u:C; \Delta_1, y:A \vdash \hat{E} :: T \quad \Gamma, u:C; \Delta_2, y:B \vdash \hat{F} :: T \quad (4)$$

and $\Delta = \Delta_1, \Delta_2$. We show that $M \simeq_c N$ implies $\Gamma; \Delta \vdash M \approx N :: T$.

By virtue of Prop. 5.5, we have to show that for every $K \in \mathcal{K}_{\Gamma;\Delta,y:A\oplus B}$, we have $\cdot; \cdot \vdash K[M] \approx K[N] :: T$. In turn, this implies exhibiting a typed context bisimilarity \mathcal{R} containing the pair $(K[M], K[N])$. We thus define \mathcal{R} as

$$\mathcal{R} = \mathcal{I}_{\vdash T} \cup \mathcal{W}_{\vdash T}$$

We now show that \mathcal{R} is a typed context bisimilarity. Pick any $K \in \mathcal{K}_{\Gamma;\Delta,y:A\oplus B}$. Using Prop 5.4, we can assume

$$K = (\nu\tilde{u}, \tilde{x}, y)(K_{\Gamma} \mid K_{\Delta} \mid V \mid [\cdot])$$

where

- $K_{\Gamma} \equiv \prod_{i \in I} !u_i(y_i).R_i$, with $\vdash R_i :: y_i:D_i$, for every $u_i:D_i \in \Gamma$;
- $K_{\Delta} \equiv \prod_{j \in J} S_j$, with $\vdash S_j :: x_j:C_j$, for every $x_j:C_j \in \Delta$;
- $\vdash V :: y:A \oplus B$.

Clearly, $(K[M], K[N]) \in \mathcal{R}$. Now, suppose $K[M]$ moves first: $K[M] \xrightarrow{\alpha} M_1^*$. We have to find a matching action α from $K[N]$, i.e., $K[N] \xRightarrow{\alpha} N_1^*$. The analysis is similar to the one detailed for the commuting conversion No. 4. We consider two possible cases:

1. Case $\alpha = \tau$. We consider the possibilities for the origin of the reduction:
 - (a) $K_{\Gamma} \rightarrow K'_{\Gamma}$: This cannot be the case, as by construction this process correspond to the parallel composition of zero or more input guarded replications which cannot evolve on their own.
 - (b) $M \rightarrow M'$: This cannot be the case, as by inspecting the structure of M we observe that both the input guarded replication on u , and the selection on y cannot proceed on their own.
 - (c) $K_{\Delta} \rightarrow K'_{\Delta}$ and $K[M] \rightarrow (\nu\tilde{u}, \tilde{x})(K_{\Gamma} \mid K'_{\Delta} \mid V \mid M)$. This reduction can be matched by $K[N]$ with a finite weak transition, as K_{Δ} occurs identically in both processes. Using subject reduction (Theorem 3.2), it can be shown that the derivatives are still in \mathcal{R} .
 - (d) $V \rightarrow V'$ and $K[M] \rightarrow (\nu\tilde{u}, \tilde{x})(K_{\Gamma} \mid K_{\Delta} \mid V' \mid M) = M_1^*$. This case proceeds similarly, as V occurs identically in both $K[M]$ and $K[N]$.
 - (e) The reduction arises from a synchronization on y between V and M . Then we have two subcases. The first one is when $V \xrightarrow{y.\text{in}\bar{r}} V'$:

$$K[M] \rightarrow (\nu\tilde{u}, \tilde{x})(K_{\Gamma} \mid K_{\Delta} \mid (\nu y)(V' \mid (\nu u)((!u(z).\hat{D}) \mid \hat{E}))) = M_1^*$$

This reduction can be matched by $K[N]$ via a finite weak transition:

$$K[N] \xRightarrow{\alpha} (\nu\tilde{u}, \tilde{x})(K_{\Gamma} \mid K'_{\Delta} \mid (\nu y)(V'' \mid (\nu u)((!u(z).\hat{D}') \mid \hat{E}')) = N_1^*$$

where N_1^* reflects the fact that internal actions could have taken place after the synchronization on y . The typing of the process can be shown to be preserved by subject reduction (Theorem 3.2), and so $(M_1^*, N_1^*) \in \mathcal{R}$. The second subcase is when $S \xrightarrow{y.\text{in}\bar{l}} S'$; this case is similar to the first one.

2. Case $\alpha \neq \tau$: Then α corresponds to the execution of some behavior described by T , in the right-hand side typing. However, this cannot be the case since, as specified by the typings in (4), the behavior described by T can only be provided by \hat{E} or by \hat{F} , which are behind a selection prefix on y , both in $K[M]$ and $K[N]$. Therefore, the behavior described by T cannot be exercised until such a prefix is consumed, and we have that, necessarily, $\alpha = \tau$. Observe that once such prefixes are consumed (via internal actions) the evolution corresponding to the behavior described by T is still in \mathcal{R} , as the continuation relation $\mathcal{W}_{\vdash T}$ is in \mathcal{R} .

The analysis when $K[N] \xrightarrow{\alpha} N_1^*$ follows the same lines and is omitted.

C.3 Type Isomorphisms

We repeat the statement of Theorem 6.4 and present its full proof.

Theorem C.4. *Let A, B be any type, as in Def 3.1. Then $A \otimes B \simeq B \otimes A$.*

Proof. We verify conditions (i)-(iv) hold for processes $P^{(x,y)}$, $Q^{(y,x)}$ defined as

$$\begin{aligned} P^{(x,y)} &= x(u).(\nu n)y\langle n \rangle.([x \leftrightarrow n] \mid [u \leftrightarrow y]) \\ Q^{(y,x)} &= y(w).(\nu m)x\langle m \rangle.([y \leftrightarrow m] \mid [w \leftrightarrow x]) \end{aligned}$$

Checking (i)-(ii), i.e., $\cdot; x:A \otimes B \vdash P^{(x,y)} :: y:B \otimes A$ and $\cdot; y:B \otimes A \vdash Q^{(y,x)} :: x:A \otimes B$ is easy; for instance, the typing derivation for (i) is as follows:

$$\frac{\frac{x:B \vdash [x \leftrightarrow n] :: n:B \quad (\text{Tid}) \quad \frac{u:A \vdash [u \leftrightarrow y] :: y:A \quad (\text{Tid})}{u:A, x:B \vdash (\nu n)y\langle n \rangle.([x \leftrightarrow n] \mid [u \leftrightarrow y]) :: y:B \otimes A \quad (\text{T} \otimes \text{R})}}{x:A \otimes B \vdash x(u).(\nu n)y\langle n \rangle.([x \leftrightarrow n] \mid [u \leftrightarrow y]) :: y:B \otimes A \quad (\text{T} \otimes \text{L})}$$

Observe how the use of rule (Tid) ensures that typings hold for any A, B . We are then left to show (iii) and (iv). We sketch only the proof of (iii); the proof of (iv) is analogous. Let $M = (\nu y)(P^{(x,y)} \mid Q^{(y,z)})$, $N = [x \leftrightarrow z]$; we need to show $\cdot; x:A \otimes B \vdash M \approx N :: z:A \otimes B$. By Prop. 5.5, we have to show that for every $K \in \mathcal{K} \cdot; x:A \otimes B$, we have $\vdash K[M] \approx K[N] :: z:A \otimes B$. In turn, this implies exhibiting a typed context bisimilarity \mathcal{R} containing $(K[M], K[N])$. Letting $\mathcal{S} = \{(R_1, R_2) : K[M] \Longrightarrow R_1, K[N] \Longrightarrow R_2\}$, we set $\mathcal{R} = \mathcal{W}_{\vdash z:A \otimes B} \cup \mathcal{S} \cup \mathcal{S}^{-1}$. We show \mathcal{R} is a typed context bisimilarity; Pick any $K \in \mathcal{K} \cdot; x:A \otimes B$. Using Prop 5.4, we can assume $K = (\nu x)(T^{(x)} \mid [\cdot])$ where $\vdash T^{(x)} :: x:A \otimes B$. By Lemma A.1 and Theorem 4.17, there exist $l, T_1^{(x)}$ such that $T^{(x)} \xrightarrow{(\nu l)x\langle l \rangle} T_1^{(x)}$ in a finite transition. Clearly, $(K[M], K[N]) \in \mathcal{R}$. Now, suppose $K[N] \xrightarrow{\alpha} N_1^*$. We have to find a matching action α from $K[M]$, i.e., $K[N] \xrightarrow{\alpha} M_1^*$. $K[N]$ has only an internal action, which leads to the renaming of $T^{(x)}: K[N] \xrightarrow{\tau} T^{(z)} = N_1^*$. Using Theorem 4.17, $K[M]$ can match this action with a finite weak transition: $K[M] \Longrightarrow (\nu n)(T_1^{(n)} \mid (\nu m)z\langle m \rangle.([l \leftrightarrow m] \mid [n \leftrightarrow z])) = M_1^*$. Using Theorem 3.2, we know that $(N_1^*, M_1^*) \in \mathcal{S}^{-1}$. Now suppose $N_1^* \xrightarrow{(\nu l)z\langle l \rangle} T_1^{(z)}$; M_1^* can match this action with an output followed by a renaming: $M_1^* \xrightarrow{(\nu m)z\langle m \rangle} T_1^{(z)} \mid [l \leftrightarrow m]$. By

definition of \approx (output clause), we take a process $S^{(c)}$ such that $\cdot; c:A \vdash S^{(c)} \vdash \cdot; \mathbf{1}$, and compose it with N_1^* and M_1^* . We thus obtain $N_2^* = (\nu l)(T_1^{(z)} \mid S^{(l)})$ and $M_2^* = (\nu m)(T_1^{(z)} \mid [l \leftrightarrow m] \mid S^{(m)})$; it can be easily checked that $(N_2^*, M_2^*) \in \mathcal{W}_{\vdash z:A \otimes B}$. When $K[M]$ moves first, the analysis is similar and we omit it. \square

Another type isomorphism is the following.

Theorem C.5. *Let A, B , and C be any type, as in Def 3.1. Then we have:*

$$(A \oplus B) \multimap C \simeq (A \multimap C) \& (B \multimap C)$$

Proof. Easy, considering the following processes $P^{(x,y)}$ and $Q^{(y,x)}$ as:

$$\begin{aligned} P^{(x,y)} &= y.\text{case}(M, N) \text{ where: } M = y(m).(\nu n)x\langle n \rangle.(n.\text{inl}; [m \leftrightarrow n] \mid [x \leftrightarrow y]) \\ &\quad N = y(v).(\nu w)x\langle w \rangle.(w.\text{inr}; [v \leftrightarrow w] \mid [x \leftrightarrow y]) \\ Q^{(y,x)} &= x(m).m.\text{case}(R, S) \text{ where: } R = y.\text{inl}; (\nu n)y\langle n \rangle.([m \leftrightarrow n] \mid [y \leftrightarrow x]) \\ &\quad S = y.\text{inr}; (\nu w)y\langle w \rangle.([m \leftrightarrow w] \mid [y \leftrightarrow x]) \end{aligned}$$

\square

1. $\Gamma; \Delta, z:D \multimap C, x:A \multimap B \vdash (\nu w)z\langle w \rangle. (R \mid (\nu y)x\langle y \rangle. (P \mid Q))$
 $\simeq_c (\nu y)x\langle y \rangle. (P \mid (\nu w)z\langle w \rangle. (R \mid Q)) :: T$
2. $\Gamma; \Delta, x:A \otimes B, z:C \otimes D \vdash x(y).z(w).P \simeq_c z(w).x(y).P :: T$
3. $\Gamma; \Delta, x:A \& B, z:C \& D \vdash x.\text{inl}; y.\text{inl}; P \simeq_c y.\text{inl}; x.\text{inl}; P :: T$
4. $\Gamma, u:A, v:C; \Delta \vdash (\nu y)u\langle y \rangle. (\nu x)v\langle x \rangle. P \simeq_c (\nu x)v\langle x \rangle. (\nu y)u\langle y \rangle. P :: T$
5. $\Gamma; \Delta, z:C \oplus D, x:A \otimes B \vdash z.\text{case}(x(y).P, x(y).Q) \simeq_c x(y).z.\text{case}(P, Q) :: T$
6. $\Gamma; \Delta, z:C \otimes D \vdash x(y).z(w).P \simeq_c z(w).x(y).P :: x:A \multimap B$
7. $\Gamma, u:A; \Delta, z:C \otimes D \vdash (\nu y)u\langle y \rangle. z(w).P \simeq_c z(w).(\nu y)u\langle y \rangle. P :: T$
8. $\Gamma; \Delta, z:C \& D, x:A \otimes B \vdash z.\text{inl}; x(y).P \simeq_c x(y).z.\text{inl}; P :: T$
9. $\Gamma; \Delta, x:A \otimes B \vdash z.\text{case}(x(y).P, x(y).Q) \simeq_c x(y).z.\text{case}(P, Q) :: z:C \& D$
10. $\Gamma; \Delta, x:A \otimes B \vdash x(y).z.\text{inl}; P \simeq_c z.\text{inl}; x(y).P :: z:C \oplus D$
11. $\Gamma; \Delta, z:C \otimes D \vdash z(w).(\nu y)x\langle y \rangle. (P \mid Q) \simeq_c (\nu y)x\langle y \rangle. (z(w).P \mid Q) :: x:A \otimes B$
12. $\Gamma, u:C; \Delta \vdash (\nu w)u\langle w \rangle. (\nu y)x\langle y \rangle. (P \mid Q) \simeq_c (\nu y)x\langle y \rangle. ((\nu w)u\langle w \rangle. P \mid Q) :: x:A \otimes B$
13. $\Gamma; \Delta, z:C \& D \vdash z.\text{inl}; (\nu y)x\langle y \rangle. (P \mid Q) \simeq_c (\nu y)x\langle y \rangle. (z.\text{inl}; P \mid Q) :: x:A \otimes B$
14. $\Gamma; \Delta, w:C \multimap D, x:A \otimes B \vdash (\nu z)w\langle z \rangle. (Q \mid x(y).P) \simeq_c x(y).(\nu z)w\langle z \rangle. (Q \mid P) :: T$
15. $\Gamma, u:C; \Delta, x:A \multimap B \vdash (\nu z)u\langle z \rangle. (\nu y)x\langle y \rangle. (P \mid Q) \simeq_c (\nu y)x\langle y \rangle. ((\nu z)u\langle z \rangle. P \mid Q) :: T$
16. $\Gamma, u:C; \Delta \vdash (\nu w)u\langle w \rangle. x(y).P \simeq_c x(y).(\nu w)u\langle w \rangle. P :: x:A \multimap B$
17. $\Gamma; \Delta, z:C \oplus D \vdash x(y).z.\text{case}(P, Q) \simeq_c z.\text{case}(x(y).P, x(y).Q) :: x:A \multimap B$
18. $\Gamma; \Delta, z:C \& D, x:A \multimap B \vdash z.\text{inl}; (\nu y)x\langle y \rangle. (P \mid Q) \simeq_c (\nu y)x\langle y \rangle. (z.\text{inl}; P \mid Q) :: T$
19. $\Gamma; \Delta, z:C \& D \vdash z.\text{inl}; x(y).P \simeq_c x(y).z.\text{inl}; P :: x:A \multimap B$
20. $\Gamma; \Delta, z:C \multimap D \vdash (\nu y)z\langle y \rangle. (R \mid x.\text{case}(P, Q))$
 $\simeq_c x.\text{case}((\nu y)z\langle y \rangle. (R \mid P), (\nu y)z\langle y \rangle. (R \mid Q)) :: x:A \& B$
21. $\Gamma, u:C; \Delta, x:A \oplus B \vdash (\nu z)u\langle z \rangle. x.\text{case}(P, Q) \simeq_c x.\text{case}((\nu z)u\langle z \rangle. P, (\nu z)u\langle z \rangle. Q) :: T$
22. $\Gamma; \Delta, x:A \oplus B, y:A \oplus B \vdash y.\text{case}(x.\text{case}(P_1, Q_1), x.\text{case}(P_2, Q_2))$
 $\simeq_c x.\text{case}(y.\text{case}(P_1, P_2), y.\text{case}(Q_1, Q_2)) :: T$
23. $\Gamma, u:A; \Delta, x:A \& B \vdash z.\text{inl}; (\nu y)u\langle y \rangle. P \simeq_c (\nu y)u\langle y \rangle. z.\text{inl}; P :: T$
24. $\Gamma; \Delta, z:D \multimap C \vdash (\nu y)z\langle y \rangle. (Q \mid x.\text{inl}; P) \simeq_c x.\text{inl}; (\nu y)z\langle y \rangle. (Q \mid P) :: x:A \oplus B$
25. $\Gamma; u:A; \Delta \vdash x.\text{case}((\nu y)u\langle y \rangle. P, (\nu y)u\langle y \rangle. Q) \simeq_c (\nu y)u\langle y \rangle. x.\text{case}(P, Q) :: x:A \& B$
26. $\Gamma; u:A; \Delta \vdash x.\text{inl}; (\nu y)u\langle y \rangle. P \simeq_c (\nu y)u\langle y \rangle. x.\text{inl}; P :: x:A \oplus B$
27. $\Gamma; \Delta, y:A \oplus B \vdash x.\text{case}(y.\text{case}(P_1, Q_1), y.\text{case}(P_2, Q_2))$
 $\simeq_c y.\text{case}(x.\text{case}(P_1, P_2), x.\text{case}(Q_1, Q_2)) :: x:A \& B$
28. $\Gamma; \Delta, y:A \oplus B \vdash x.\text{inl}; y.\text{case}(P, Q) \simeq_c y.\text{case}(x.\text{inl}; P, x.\text{inl}; Q) :: x:A \oplus B$
29. $\Gamma; \Delta, y:C \& D \vdash y.\text{inl}; x.\text{case}(P, Q) \simeq_c x.\text{case}(y.\text{inl}; P, y.\text{inl}; Q) :: x:A \& B$
30. $\Gamma; \Delta, y:C \& D \vdash y.\text{inl}; x.\text{inl}; P \simeq_c x.\text{inl}; y.\text{inl}; P :: x:A \oplus B$

Fig. 8. Proof conversions of the second kind: prefix commutations