Linear Logical Relations and Observational Equivalences for Session-Based Concurrency

Jorge A. Pérez[a], Luís Caires[a], Frank Pfenning[b], Bernardo Toninho[a,b]

[a] *CITI and Departamento de Informática, FCT Universidade Nova de Lisboa*
[b] *Computer Science Department, Carnegie Mellon University*

Abstract

We investigate *strong normalization*, *confluence*, and *behavioral equality* in the realm of session-based concurrency. These interrelated issues underpin advanced correctness analysis in models of structured communications. The starting point for our study is an interpretation of linear logic propositions as session types for communicating processes, proposed in prior work. Strong normalization and confluence are established by developing a theory of *logical relations*. Defined upon a linear type structure, our logical relations remain remarkably similar to those for functional languages. We also introduce a natural notion of *observational equivalence* for session-typed processes. Strong normalization and confluence come in handy in the associated coinductive reasoning: 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.

Keywords: Session Types, Linear Logic, Strong Normalization, Confluence, Logical Relations, Observational Equivalences

1. Introduction

Modern computing systems rely heavily on the *communication* between distributed software artifacts. Hence, to a large extent, guaranteeing system correctness amounts to ensuring consistent dialogues between such artifacts. This is a challenging task, given the complex interaction patterns that communicating systems usually feature. *Session-based concurrency* provides a foundational approach to communication correctness: concurrent dialogues are structured into basic units called *sessions*; descriptions of the interaction patterns are then abstracted as *session types* [21, 22, 16], which are statically checked against specifications. These specifications are usually given in the π-calculus [30, 41], so we obtain *processes* interacting on so-called session channels which connect exactly two subsystems. The discipline of session types ensures 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 invoking shared servers. While *concurrency* arises in the simultaneous execution of sessions, *mobility* results from the exchange of session and server names.

The goal of this paper is to investigate *strong normalization*, *confluence*, and *typed behavioral equivalences* for session-typed, communicating processes. These interrelated issues underpin advanced correctness analysis in models of structured communications. Our study builds upon the interpretation of linear logic propositions as session types put forward by Caires and Pfenning in [10]. In a concurrent setting, such an interpretation defines a tight propositions-as-types/proofs-as-programs correspondence, in the style of the Curry-Howard isomorphism for the simply-typed λ-calculus [23]. In the interpretation, types (linear logic propositions) are assigned to names (communication channels) and describe their session protocol; typing rules correspond to linear sequent calculus proof rules and processes correspond to proof objects in logic derivations. Moreover, process reduction may be simulated by proof conversions and reductions, and vice versa. As a result, typed processes enjoy strong forms of *subject reduction* (type preservation) and *global progress* (deadlock-freedom). While 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 get into a stuck state (a *liveness* property). These strong correctness properties make the framework in [10] a convenient basis for our study of strong normalization, confluence, and behavioral equivalences. Well-studied in sequential settings, these three interrelated issues constitute substantial challenges for theories of communicating processes, as we motivate next.

In typed functional calculi, *strong normalization* ensures that well-typed terms do not have infinite reduction sequences. Types rule out divergent computations; termination of reduction entails consistency of the corresponding logical systems. In the realm of communicating processes, reduction captures atomic synchronization; associated behavioral types exclude unintended structured interactions. As a result, strong normalization acquires an enhanced significance in a concurrent setting. In fact, even if subject reduction and progress are typically regarded as the key correctness guarantees for processes, requiring strongly normalizing behaviors is also most sensible: while from a global perspective systems are meant to run forever, at a local level we wish responsive participants which always react within a finite amount of time, and never engage into infinite internal behavior. For instance, in server-client applications it is critical for clients to be sure that running code provided by the server will not cause them to get stuck indefinitely (as in a denial-of-service attack, or just due to some bug).

Closely related to strong normalization, *confluence* is another appealing property. In a communication-based setting, confluence would strengthen correctness guarantees by ensuring *predictability* of structured interactions. This benefit may be more concretely appreciated by considering the principle of *typed process composition* derived from the logic interpretation. In [10], typing judgments specify both the session behavior that a process *offers* (or *implements*) and the set of (unrestricted and linear) behaviors that it *requires* to do so. Given this typed interface, each such behavioral dependencies is satisfied by first (i) composing the given process with another one which realizes the required behavior, and then (ii) restricting the name in which the behavior is required/offered. As a result, the interactions between the given process and the processes implementing its dependencies are unobservable. In this context, the interplay of confluence with strong normalization would be significant, as it could crucially ensure

that session behavior as declared by judgments will be always offered, independently from any arbitrary interleaving of internal reductions from different sources.

Now, in sharp contrast to the normalizing, confluent nature of computation in many typed functional calculi, process calculi are inherently non-terminating, non-confluent models of concurrent computation. Hence, unsurprisingly, ensuring strong normalization and confluence in calculi for concurrency is a hard problem: in (variants of) the π-calculus, proofs require heavy constraints on the language and/or its types, often relying on ad-hoc machineries (see [14] for a survey on termination in process calculi). As a first challenge, we wonder: building upon our linear type structure, directly obtained from the Curry-Howard correspondence in [10], can we establish useful forms of strong normalization and confluence for session-typed, communicating processes?

While from an operational standpoint strong normalization and confluence are relevant, at a more foundational level they are also related to notions of *typed equality*. For instance, in the simply-typed λ-calculus, strong normalization and confluence ensure that normal forms exist and are unique, and entail decidability of denotational equality. In our concurrent setting, strong normalization is also related to *behavioral equivalence* —arguably the most basic notion in a theory of processes. Behavioral equivalences enable us to formally assert when two processes denote the "same behavior". A first, basic connection between strong normalization, confluence, and behavioral equivalence is obtained by means of subject reduction/type preservation: process behavior (as declared by typing judgements) is preserved along arbitrary reduction steps. Building upon this connection, any notion of behavioral equality over session-typed processes should be necessarily informed by the correspondence between session types and linear logic propositions. As detailed in [10], such a correspondence is realized by relating *proof conversions* in linear logic with appropriate notions in the process setting. Interestingly, by virtue of such proof conversions the correspondence already induces a notion of typed process equality. As illustration, consider the following process equalities, two instances of proof conversions:

$$(\nu x)(P \mid \overline{z}\langle y\rangle.(Q \mid R)) \;\simeq_c\; \overline{z}\langle y\rangle.((\nu x)(P \mid Q) \mid R) \tag{1}$$

$$x(y).z(w).P \;\simeq_c\; z(w).x(y).P \quad (\text{with } x \neq z) \tag{2}$$

In our framework, while equality (1) results from the interplay of typed constructs for output and process composition, (2) arises from the typing of two independent sessions (on names x and z). Crucially, in both cases, the equated processes are syntactically very different and yet they are associated to the *same* typing judgment —that is, their logic-based session interface decrees the same behavior. As a second challenge, we ask: can we define a notion of typed process equality that is both natural and intuitive, that enjoys good properties (e.g. congruence), and that captures the notion of equality that is already induced by the logic interpretation via proof conversions?

A clear understanding of the status of strong normalization, confluence, and process equalities would provide a fundamental stepping stone towards a deeper understanding on how logic-based session types delineate communications. That is, basic behavioral equivalences over *equally typed* processes (in which strong normalization and confluence are expected to play substantial rôles) may also provide a basis for reasoning about the behavior of processes with *different types*. In fact, given that session

types represent service interfaces of distributed software artifacts, it is legitimate to ask whether the logical interpretation enables reasoning techniques at the level of session types. Such techniques appear very useful from a pragmatic perspective —for instance, they could enable natural notions of interface compatibility. Reasoning techniques at the level of types would also be useful from the more foundational standpoint of typed equality. To illustrate this, let us consider the session-typed interpretation of $\otimes$ given in [10], whereby an object of type $A \otimes B$ denotes a session that first *outputs* a channel of type A and then behaves as B. This intuitive description may suggest an asymmetric interpretation, as opposed to the well-known symmetric nature of $\otimes$. This apparent asymmetry is already clarified in [10]: using a suitable typed process, it is shown how a session of type $A \otimes B$ may be coerced into one of type $B \otimes A$ (and viceversa). This justification, however, leaves open the general issue of equality over session types. In fact, we wish to understand the formal meaning in our setting of a notion of typed equality, in such a way that expected logic principles such as

$$A \otimes B \simeq B \otimes A \tag{3}$$

are properly justified. A final challenge would be then: building upon typed process equivalences, can we derive a simple notion of equality over session types that justifies/validates principles such as (3) above but also arbitrary interface transformations?

With the aim of addressing the challenges described above, the present paper offers the following technical contributions:

(1) We present a simple theory of *logical relations* for session-typed processes, and use it to show that well-typed processes are both *strongly normalizing* and *confluent*.

The method of *logical relations* [42, 43] has proved to be extremely productive in the functional setting; properties such as strong normalization and parametricity can be established via logical relations. Although the logic interpretation in [10] assigns types to names (and not to terms, as in the typed λ-calculus), quite remarkably, our *linear* logical relations are truly defined on the structure of types—as in logical relations for the typed λ-calculus [42, 43]. This allows for simple proofs of strong normalization and confluence, which follow closely the principles of the (linear) type system. To our knowledge, ours are the first proofs of their kind in the context of session-based concurrency.

(2) We investigate a *behavioral theory* for session-typed processes, defined as a typed contextual equivalence which follows the principles of the logical interpretation.

Well-studied in the untyped case, behavioral equivalences have been only little studied for session-typed processes (in fact, the only previous work we are aware of is [26]). We introduce *typed context bisimilarity*, a natural notion of observational equivalence for typed processes. We show how, thanks to the combination of type preservation, progress, strong normalization, and confluence, typed context bisimilarity satisfies τ*-inertness*, as studied by Groote and Sellink [19]. Intuitively, τ-inertness says that reduction (internal behavior) does not change process behavior. This is most relevant for verification, as it means that our well-typed processes

can perform arbitrarily many reductions while remaining in the same equivalence class. In our setting, this guarantee is neatly complemented by strong normalization, which ensures finitely many reductions.

(3) By relying on the above results, we then develop two *applications*, which clarify further the nature of the logical interpretation of session types put forward in [10]:

- We prove that proof conversions are *sound* with respect to observational equivalence. This way, processes equalities induced by proof conversions (such as (1) and (2) above) correspond to typed context bisimilarities. This soundness result elegantly explains subtle forms of causality that arise in the execution of concurrent sessions.
- Building upon typed bisimilarity, we offer a characterization of *type isomorphisms* (see, e.g., [17]). Intuitively, such isomorphisms result from linear logic equivalences which are realized by process coercions. Our characterization allows us to show that principles such as (3) above are indeed isomorphisms.

Our applications thus shed further light on the relationship between linear logic and structured communications. Strong normalization and confluence properties are central in the associated coinductive reasoning, intuitively because in the bisimulation game strong transitions are always matched by weak transitions with finite and confluent internal behavior.

Organization. Next, in Section 2, we present our process model, a synchronous π-calculus with guarded choice. Section 3 recalls the type system derived from the logical interpretation and main results from [10]. Section 4 presents proof conversions, describing inference permutability issues derived from the logical interpretation. Section 5 presents linear logical relations for typed processes, as well as the proof of strong normalization and confluence. Section 6 introduces typed context bisimilarity and studies its main properties. Section 7 presents our two applications. Finally, Section 8 discusses related work, and Section 9 collects some final remarks. A number of proofs and technical details have been moved to the Appendix.

This paper is an extended version of the conference paper [33]. In this presentation, we provide full technical details and include some new material: Section 4, on proof conversions; Section 5.3, on a proof confluence via linear logical relations; and the proof of τ-inertness given in Section 6.

2. Process Model: Syntax and Semantics

We introduce the syntax and operational semantics of the synchronous π-calculus [41] extended with (binary) guarded choice.

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

$$\begin{array}{rcl} P & ::= & \mathbf{0} \mid P \mid Q \mid (\boldsymbol{\nu} y)P \mid x\langle y\rangle.P \mid x(y).P \mid !x(y).P \\ & & \mid [x \leftrightarrow y] \mid x.\mathtt{inl}; P \mid x.\mathtt{inr}; P \mid x.\mathtt{case}(P, Q) \end{array}$$

The operators $\mathbf{0}$ (inaction), $P \mid Q$ (parallel composition), and $(\boldsymbol{\nu} y)P$ (name restriction) comprise the static fragment of the π-calculus. We then have prefixed processes $x\langle y\rangle.P$ (send name y on x and proceed as P), $x(y).P$ (receive a name z on x and proceed as P with parameter y replaced by z), and $!x(y).P$ which denotes replicated (persistent) input. Following [39], we write $\overline{x}\langle y\rangle$ as an abbreviation for $(\nu y)x\langle y\rangle$. The *forwarding construct* $[x \leftrightarrow y]$ equates names x and y; it is a primitive representation of a copycat process, akin to the link processes used in internal mobility encodings of name-passing [5]. As described in Section 3, this construct allows for a simple identity axiom in the type system [44]. 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., [22]). Without loss of generality we restrict our model to binary choice. In restriction $(\boldsymbol{\nu} y)P$ and input $x(y).P$ the distinguished occurrence of name y is binding, with scope P.

The set of *free names* of a process P is denoted $\mathit{fn}(P)$. A process is *closed* if it does not contain free occurrences of names. We identify process up to consistent renaming of bound names, writing $\equiv_\alpha$ for this congruence. We write $P\{^x/_y\}$ for the capture-avoiding substitution of x for y in P. While *structural congruence* expresses basic identities on the structure of processes, *reduction* expresses the behavior of processes.

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

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

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

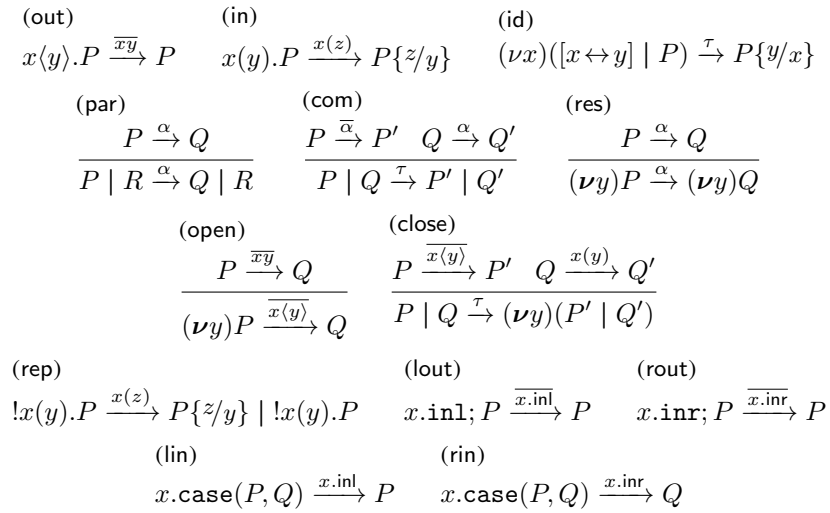
$$
\begin{array}{ll}
x\langle y\rangle.Q \mid x(z).P \to Q \mid P\{^y/_z\} & x\langle y\rangle.Q \mid !x(z).P \to Q \mid P\{^y/_z\} \mid !x(z).P \\
(\nu x)([x \leftrightarrow y] \mid P) \to P\{^y/_x\} \ \ (x \neq y) & Q \to Q' \Rightarrow P \mid Q \to P \mid Q' \\
P \to Q \Rightarrow (\boldsymbol{\nu}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

By definition, reduction is closed under $\equiv$. It specifies the computations a process performs on its own. To define the interactions of a process with its environment, we extend the early transition system for the π-calculus [41] with labels and transition rules for the choice and forwarding constructs. A transition $P \xrightarrow{\alpha} Q$ denotes that P may evolve to Q by performing the action represented by label α. Labels are given by:

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

Actions are input $x(y)$, the left/right offers $x.\mathsf{inl}$ and $x.\mathsf{inr}$, and their matching co-actions, respectively the free output $\overline{xy}$ and bound output $\overline{x\langle y\rangle}$ actions, and the left/right selections $\overline{x.\mathsf{inl}}$ and $\overline{x.\mathsf{inr}}$. The bound output $\overline{x\langle y\rangle}$ denotes extrusion of a fresh name y along 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 $\mathit{fn}(\alpha)$ and $\mathit{bn}(\alpha)$ of free and bound names, respectively, as usual. We denote by $s(\alpha)$ the subject of α (e.g., x in $x\langle y\rangle$).

Figure 1 π-calculus Labeled Transition System.



Definition 2.4 (Labeled Transition System)**.** *The relation* labeled transition *($P \xrightarrow{\alpha} Q$) is defined by the rules in Figure 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 $\alpha \neq \tau$, notation $\stackrel{\alpha}{\Longrightarrow}$ stands for $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$ and $\stackrel{\tau}{\Longrightarrow}$ 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 [41]: (1) if $P \equiv \xrightarrow{\alpha} Q$ then $P \xrightarrow{\alpha} \equiv Q$, and (2) $P \to Q$ if and only if $P \xrightarrow{\tau} \equiv Q$.

3. Session Types as Intutionistic Linear Logic Propositions

As anticipated in the introduction, the type structure coincides with intuitionistic linear logic [18, 2], omitting atomic formulas and the additive constants $\top$ and $\mathbf{0}$.

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

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

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

Figure 2 The Type System πDILL.

$$\text{(Tid)}\ \frac{}{\Gamma; x{:}A \vdash [x \leftrightarrow z] :: z{:}A} \qquad \text{(T1L)}\ \frac{\Gamma; \Delta \vdash P :: T}{\Gamma; \Delta, x{:}\mathbf{1} \vdash P :: T} \qquad \text{(T1R)}\ \frac{}{\Gamma; \cdot \vdash \mathbf{0} :: x{:}\mathbf{1}}$$

$$\text{(T}\otimes\text{L)}\ \frac{\Gamma; \Delta, y{:}A, x{:}B \vdash P :: T}{\Gamma; \Delta, x{:}A \otimes B \vdash x(y).P :: T} \qquad \text{(T}\otimes\text{R)}\ \frac{\Gamma; \Delta \vdash P :: y{:}A \quad \Gamma; \Delta' \vdash Q :: x{:}B}{\Gamma; \Delta, \Delta' \vdash \overline{x}\langle y\rangle.(P \mid Q) :: x{:}A \otimes B}$$

$$\text{(T}\multimap\text{L)}\ \frac{\Gamma; \Delta \vdash P :: y{:}A \quad \Gamma; \Delta', x{:}B \vdash Q :: T}{\Gamma; \Delta, \Delta', x{:}A \multimap B \vdash \overline{x}\langle y\rangle.(P \mid Q) :: T} \qquad \text{(T}\multimap\text{R)}\ \frac{\Gamma; \Delta, y{:}A \vdash P :: x{:}B}{\Gamma; \Delta \vdash x(y).P :: x{:}A \multimap B}$$

$$\text{(Tcut)}\ \frac{\Gamma; \Delta \vdash P :: x{:}A \quad \Gamma; \Delta', x{:}A \vdash Q :: T}{\Gamma; \Delta, \Delta' \vdash (\nu x)(P \mid Q) :: T} \qquad \text{(Tcut}^{!}\text{)}\ \frac{\Gamma; \cdot \vdash P :: y{:}A \quad \Gamma, u{:}A; \Delta \vdash Q :: T}{\Gamma; \Delta \vdash (\nu u)(!u(y).P \mid Q) :: T}$$

$$\text{(T!L)}\ \frac{\Gamma, u{:}A; \Delta \vdash P\{^{u}/_{x}\} :: T}{\Gamma; \Delta, x{:}!A \vdash P :: T} \qquad \text{(Tcopy)}\ \frac{\Gamma, u{:}A; \Delta, y{:}A \vdash P :: T}{\Gamma, u{:}A; \Delta \vdash \overline{u}\langle y\rangle.P :: T} \qquad \text{(T!R)}\ \frac{\Gamma; \cdot \vdash Q :: y{:}A}{\Gamma; \cdot \vdash !x(y).Q :: x{:}!A}$$

$$\text{(T}\oplus\text{L)}\ \frac{\Gamma; \Delta, x{:}A \vdash P :: T \quad \Gamma; \Delta, x{:}B \vdash Q :: T}{\Gamma; \Delta, x{:}A \oplus B \vdash x.\mathtt{case}(P, Q) :: T} \qquad \text{(T}\&\text{R)}\ \frac{\Gamma; \Delta \vdash P :: x{:}A \quad \Gamma; \Delta \vdash Q :: x{:}B}{\Gamma; \Delta \vdash x.\mathtt{case}(P, Q) :: x{:}A \,\&\, B}$$

$$\text{(T}\&\text{L}_1\text{)}\ \frac{\Gamma; \Delta, x{:}A \vdash P :: T}{\Gamma; \Delta, x{:}A \,\&\, B \vdash x.\mathtt{inl}; P :: T} \qquad \text{(T}\oplus\text{R}_1\text{)}\ \frac{\Gamma; \Delta \vdash P :: x{:}A}{\Gamma; \Delta \vdash x.\mathtt{inl}; P :: x{:}A \oplus B}$$

$$\text{(T}\&\text{L}_2\text{)}\ \frac{\Gamma; \Delta, x{:}B \vdash P :: T}{\Gamma; \Delta, x{:}A \,\&\, B \vdash x.\mathtt{inr}; P :: T} \qquad \text{(T}\oplus\text{R}_2\text{)}\ \frac{\Gamma; \Delta \vdash P :: x{:}B}{\Gamma; \Delta \vdash x.\mathtt{inr}; P :: x{:}A \oplus B}$$

no further interaction will take place on it; names of type $\mathbf{1}$ may still be passed around in sessions, as opaque values. $A\&B$ types a 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-linear) 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. When empty, Γ and Δ are denoted by '$\cdot$'. A type judgment is 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. Such a judgment asserts: P is ensured to safely provide a usage of name z according to the behavior specified by type C, whenever composed with any process environment providing usages of names according to the behaviors specified by names in $\Gamma; \Delta$.

Our type judgment defines an intuitive reading of processes. Given $\Gamma; \Delta \vdash P :: z{:}C$, process P represents a system providing behavior C at channel z, building on "services" declared in Γ and Δ. This way, for instance, a client Q that relies on external services and does not provide any would be typed as $\Gamma; \Delta \vdash Q :: -{:}\mathbf{1}$, while a system typed as $\Gamma; \Delta \vdash R :: z{:}!A$ represents a shared server. 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 $\pi\texttt{DILL}$ are given in Figure 2. We use T, S for right-hand side singleton environments (e.g., $z{:}C$). Rule (Tid) defines identity in terms of the forwarding construct. Since in rule (T$\otimes$R) the sent name is always fresh, our typed calculus conforms to an internal mobility discipline [5, 39], without loss of expressiveness. The composition rules (Tcut/Tcut$^!$) follow the "composition plus hiding" principle [1], extended to a name-passing setting. Other linear typing rules for parallel composition (as in, e.g., [25]) are derivable—see [10]. As we consider π-calculus terms up to structural congruence, typability is closed under $\equiv$ by definition. $\pi\texttt{DILL}$ enjoys the usual properties of equivariance, weakening, and contraction in Γ. The coverage property also holds: if $\Gamma; \Delta \vdash P :: z{:}A$ then $\mathit{fn}(P) \subseteq \Gamma \cup \Delta \cup \{z\}$. In the presence of type-annotated restrictions $(\nu x{:}A)P$, as usual in typed π-calculi [41], type-checking is decidable.

Session type constructors thus correspond directly to intuitionistic linear logic connectives. By erasing processes, typing judgments in $\pi\texttt{DILL}$ correspond to DILL, a sequent formulation of Barber's dual intuitionistic linear logic [2, 12]. Below we informally recall this correspondence; see [10, 11] for details.

DILL is equipped with a faithful proof term assignment: sequents have the form $\Gamma; \Delta \vdash D : C$, where Γ is the unrestricted context, Δ the linear context, C a formula (i.e., a 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 $\pi\texttt{DILL}$. The converse also holds: if $\Gamma; \Delta \vdash P :: z{:}A$ is derivable in $\pi\texttt{DILL}$ 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: given $\Gamma; \Delta \vdash D : C$, 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. Reduc-

tions 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 [10, 11], on *subject reduction (type preservation)* and *progress (deadlock-freedom)* for well-typed processes. For any P, define $live(P)$ iff $P \equiv (\nu\widetilde{n})(\pi.Q \mid R)$, for some sequence of names $\widetilde{n}$, a process R, and a *non-replicated* guarded process $\pi.Q$.

Theorem 3.1 (Subject Reduction)**.** *If* $\Gamma;\Delta \vdash P :: z{:}A$ *and* $P \to Q$ *then* $\Gamma;\Delta \vdash Q :: z{:}A$.

Lemma 3.1. *Let* $\Gamma;\Delta \vdash D \rightsquigarrow P :: z{:}C$. *If* $live(P)$ *then there is a* Q *such that either (1)* $P \to Q$, *or (2)* $P \xrightarrow{\alpha} Q$ *for* α *where* $s(\alpha) \in (z, \Gamma, \Delta)$. *Moreover, if* $C = !A$ *for some* A, *then* $s(\alpha) \neq z$.

Theorem 3.2 (Progress)**.** *If* $\cdot;\cdot \vdash P{::}z{:}\mathbf{1}$ *and* $live(P)$ *then exists a* Q *such that* $P \to Q$.

We close this section recalling some other auxiliar results from [10, 11].

Lemma 3.2 (Action 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* $s(\alpha) = x$ *and* $C = A \otimes B$ *then* $\alpha = \overline{x\langle y\rangle}$.
3. *If* $P \xrightarrow{\alpha} Q$ *and* $s(\alpha) = x$ *and* $C = A{\multimap}B$ *then* $\alpha = x(y)$.
4. *If* $P \xrightarrow{\alpha} Q$ *and* $s(\alpha) = x$ *and* $C = A \,\&\, B$ *then* $\alpha = x.\mathsf{inl}$ *or* $\alpha = x.\mathsf{inr}$.
5. *If* $P \xrightarrow{\alpha} Q$ *and* $s(\alpha) = x$ *and* $C = A \oplus B$ *then* $\alpha = \overline{x.\mathsf{inl}}$ *or* $\alpha = \overline{x.\mathsf{inr}}$.
6. *If* $P \xrightarrow{\alpha} Q$ *and* $s(\alpha) = x$ *and* $C = !A$ *then* $\alpha = x(y)$.

Lemma 3.3 (Preservation Lemma, Output Case)**.** *Assume*

- $\Gamma;\Delta_1 \vdash D \rightsquigarrow P :: x{:}A_1 \otimes A_2$ *with* $P \xrightarrow{\overline{x\langle y\rangle}} P'$*; and*
- $\Gamma;\Delta_2, x{:}A_1 \otimes A_2 \vdash E \rightsquigarrow Q :: z{:}C$ *with* $Q \xrightarrow{x(y)} Q'$.

Then: $\Gamma;\Delta_1,\Delta_2 \vdash F \rightsquigarrow R :: z : C$ *for* $R \equiv (\nu y)(\nu x)(P' \mid Q')$.

Lemma 3.4. *Assume* $\Gamma;\Delta \vdash D \rightsquigarrow P :: z{:}C$ *and not* $live(P)$. *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* $P \equiv (\nu\widetilde{x})(!z(y).R \mid R')$

4. Inference Permutability and 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 detailed in [10]), 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 [10] 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 $\simeq_c$.

We illustrate proof conversions and their associated π-calculus processes; Figure 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 Figure 3—related to two applications of rule (T$\otimes$L) —could be stated as

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

where Γ and Δ are environments, A, B, C, D are types, and T is a right-hand side typing. For the sake of illustration, however, in Figure 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 two 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 Figure 3 (Page 12) are examples (the first five involve (Tcut), the following 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 associated to independent (non-interfering) sessions—the last two rows in Figure 3 are examples.

We formally introduce the set of process equalities induced by proof conversions.

Definition 4.1 (Proof Conversions)**.** *We define $\simeq_c$ as the least congruence on processes induced by the process equalities in Figures 4, 5, 6, and 7 (Pages 14–17).*

This way, process equalities of the first kind are given in Figures 4 and 5, while Figures 6 and 7 collect equalities of the second kind. As mentioned above, each of the equalities described by $\simeq_c$ is related to approrpriate left- and right-hand side typings. We illustrate the rôle of these typings with examples. Consider equality I-6 in Figure 4, which states the permutability of (T$\multimap$L) and (Tcut):

$$\dfrac{\Gamma; \Delta_1 \vdash D :: x{:}C \quad \dfrac{\Gamma; \Delta_2, x{:}C \vdash \hat{E} :: z{:}A \quad \Gamma; \Delta_3, y{:}B \vdash \hat{F} :: T}{\Gamma; \Delta_2, \Delta_3, x{:}C, y{:}A \multimap B \vdash \overline{y}\langle z\rangle.(\hat{E} \mid \hat{F}) :: T}\ (\mathsf{T}\multimap\mathsf{L})}{\Gamma; \Delta, y{:}A \multimap B \vdash (\nu x)(\hat{D} \mid \overline{y}\langle z\rangle.(\hat{E} \mid \hat{F})) :: T}\ (\mathsf{Tcut})$$

where $\Delta = \Delta_1, \Delta_2, \Delta_3$. Permutability is justified by the following inference:

$$\dfrac{\dfrac{\Gamma; \Delta_1 \vdash \hat{D} :: x{:}C \quad \Gamma; \Delta_2, x{:}C \vdash \hat{E} :: z{:}A}{\Gamma; \Delta_1, \Delta_2 \vdash (\nu x)(\hat{D} \mid \hat{E}) :: z{:}A}\ (\mathsf{Tcut}) \qquad \Gamma; \Delta_3, y{:}B \vdash \hat{F} :: T}{\Gamma; \Delta, y{:}A \multimap B \vdash \overline{y}\langle z\rangle.((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F}) :: T}\ (\mathsf{T}\multimap\mathsf{L})$$

Figure 3 A sample of process equalities induced by proof conversions (cf. Def. 4.1)

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

Process equalities of the first kind can be classified into three subgroups. The first and second subgroups are associated to permutability with (Tcut)—see Figure 4; the third subgroup is associated to permutability with (Tcut$^!$)—see Figure 5. More precisely, the first subgroup, given by process equalities I-1/I-15, includes the interaction of a process $\hat{D}$ offering a service C on x, with some process requiring such service; this process varies according to the particular rule considered. While this first subgroup covers the interaction of (Tcut) with both left and right rules, the second subgroup, given by process equalities I-16/I-24, covers the interaction of (Tcut) with left rules only. This distinction is due to the shape of rule (Tcut); in order to see this, compare the inferences justifying equality I-6 (given above) with those for equality I-17:

$$
\dfrac{\dfrac{\Gamma; \Delta_1 \vdash \hat{D} :: z{:}A \quad \Gamma; \Delta_2, y{:}B \vdash \hat{E} :: x{:}C}{\Gamma; \Delta_1, \Delta_2, y{:}A \multimap B \vdash \overline{y}\langle z\rangle.(\hat{D} \mid \hat{E}) :: x{:}C}\ (\mathsf{T}\multimap\mathsf{L}) \qquad \Gamma; \Delta_3, x{:}C \vdash \hat{F} :: T}{\Gamma; \Delta, y{:}A \multimap B \vdash\ (\nu x)(\overline{y}\langle z\rangle.(\hat{D} \mid \hat{E}) \mid \hat{F}) :: T}\ (\mathsf{Tcut})
$$

where $\Delta = \Delta_1, \Delta_2, \Delta_3$. Permutability is then justified by the following inference:

$$
\dfrac{\Gamma; \Delta_1 \vdash \hat{D} :: z{:}A \qquad \dfrac{\Gamma; \Delta_2, y{:}B \vdash \hat{E} :: x{:}C \quad \Gamma; \Delta_3, x{:}C \vdash \hat{F} :: T}{\Gamma; \Delta_2, \Delta_3, y{:}B \vdash (\nu x)(\hat{E} \mid \hat{F}) :: T}\ (\mathsf{Tcut})}{\Gamma; \Delta, y{:}A \multimap B \vdash \overline{y}\langle z\rangle.(\hat{D} \mid (\nu x)(\hat{E} \mid \hat{F})) :: T}\ (\mathsf{T}\multimap\mathsf{L})
$$

Observe how both equalities lead to structurally congruent processes. Finally, the third subgroup (equalities I-25/I-39) involves the permutability of left and right rules with (Tcut$^!$).

The permutability for the process equalities of the second kind, described in Figures 6 and 7, follows the same principles. Consider, for instance, equality II-20, which states the permutability of (T&L$_1$) and (T⊗R):

$$\dfrac{\dfrac{\Gamma;\Delta_1, z{:}C \vdash P :: y{:}A \quad \Gamma;\Delta_2 \vdash Q :: x{:}B}{\Gamma;\Delta, z{:}C \vdash \overline{x}\langle y\rangle.(P \mid Q) :: x{:}A \otimes B}\ (\mathsf{T}\otimes\mathsf{R})}{\Gamma;\Delta, z{:}C \,\&\, D \vdash z.\mathtt{inl}; \overline{x}\langle y\rangle.(P \mid Q) :: x{:}A \otimes B}\ (\mathsf{T}\&\mathsf{L}_1)$$

where $\Delta = \Delta_1, \Delta_2$. Permutability is justified by the following inference:

$$\dfrac{\dfrac{\Gamma;\Delta_1, z{:}C \vdash P :: z{:}A}{\Gamma;\Delta, z{:}C \,\&\, D \vdash z.\mathtt{inl}; P :: y{:}A}\ (\mathsf{T}\&\mathsf{L}_1) \qquad \Gamma;\Delta_2 \vdash Q :: x{:}B}{\Gamma;\Delta, z{:}C \,\&\, D \vdash \overline{x}\langle y\rangle.(z.\mathtt{inl}; P \mid Q) :: x{:}A \otimes B}\ (\mathsf{T}\otimes\mathsf{R})$$

Not all permutations are sound nor are possible. In particular, for permutability of two inference rules to be sound, one of them has to be a left rule; the permutation of two right rules leads to unsound transformations. Process equalities of the second kind can also be divided into two subgroups: those involving two left rules (see Figure 6) and those involving one left and one right rule (see Figure 7). For the sake of space, we consider only combinations with rule (T&L$_1$); permutations involving (T&L$_2$) are easily derivable. While there is no rule that can permute with (T1R), rule (T1L) can permute with all rules without changing the process structure. The situation is similar for (T!R) and (T!L): the former is incompatible for permutation with all rules, while the latter can permute with all rules, excepting (T!R). The effect of (T!L) in processes is a substitution; equated processes only differ in the scope of such a substitution.

5. Linear Logical Relations for Session-Typed Processes

Here we introduce a theory of *linear* logical relations for session types, and use it to prove that well-typed processes are strong normalizing and confluent. As customary, the proof can be summarized into two steps:

(1) Definition of a logical predicate on processes, by induction on the structure of (session) types. By definition, processes in the predicate are strongly normalizing (resp. confluent).

(2) Proof that every well-typed process is in the logical predicate.

In some previous works in previous calculi [40, 14], strong normalization is simply referred to as *termination*. In what follows, we often use the two terms interchangeably.

5.1. Preliminaries

Definition 5.1 (Termination)**.** *A process P* terminates*, noted $P\Downarrow$, if there is no infinite reduction path from P.*

We begin by stating an extension to $\equiv$, which will be useful in our developments.

Definition 5.2. *We write $\equiv_!$ for the least congruence relation on processes which results from extending structural congruence $\equiv$ (Def. 2.2) with axioms (1)–(3) below:*

Figure 4 Process equalities induced by proof conversions, first kind (first and second subgroups)

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F})) \simeq_c \overline{z}\langle y\rangle.((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F}) :: z{:}A \otimes B \quad \text{(I-1)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F})) \simeq_c \overline{z}\langle y\rangle.(\hat{E} \mid (\nu x)(\hat{D} \mid \hat{F})) :: z{:}A \otimes B \quad \text{(I-2)}$$

$$\Gamma;\Delta, y{:}A \otimes B \vdash (\nu x)(\hat{D} \mid y(z).\hat{E}) \simeq_c y(z).(\nu x)(\hat{D} \mid \hat{E}) :: T \quad \text{(I-3)}$$

$$\Gamma;\Delta, y{:}A \otimes B \vdash (\nu x)(y(z).\hat{D} \mid \hat{E}) \simeq_c y(z).(\nu x)(\hat{D} \mid \hat{E}) :: T \quad \text{(I-4)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid z(y).\hat{E}) \simeq_c z(y).(\nu x)(\hat{D} \mid \hat{E}) :: z{:}A \multimap B \quad \text{(I-5)}$$

$$\Gamma;\Delta, y{:}A \multimap B \vdash (\nu x)(\hat{D} \mid \overline{y}\langle z\rangle.(\hat{E} \mid \hat{F})) \simeq_c \overline{y}\langle z\rangle.((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F}) :: T \quad \text{(I-6)}$$

$$\Gamma;\Delta, y{:}A \multimap B \vdash (\nu x)(\hat{D} \mid \overline{y}\langle z\rangle.(\hat{E} \mid \hat{F})) \simeq_c \overline{y}\langle z\rangle.(\hat{E} \mid (\nu x)(\hat{D} \mid \hat{F})) :: T \quad \text{(I-7)}$$

$$\Gamma;\Delta, y{:}A \multimap B \vdash (\nu x)(\overline{y}\langle z\rangle.(\hat{E} \mid \hat{D}) \mid \hat{F}) \simeq_c \overline{y}\langle z\rangle.(\hat{E} \mid (\nu x)(\hat{D} \mid \hat{F})) :: T \quad \text{(I-8)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid z.\mathtt{case}(\hat{E}, \hat{F})) \simeq_c z.\mathtt{case}((\nu x)(\hat{D} \mid \hat{E}), (\nu x)(\hat{D} \mid \hat{F})) :: z{:}A \,\&\, B \quad \text{(I-9)}$$

$$\Gamma;\Delta, y{:}A \,\&\, B \vdash (\nu x)(\hat{D} \mid y.\mathtt{inl}; \hat{E}) \simeq_c y.\mathtt{inl}; (\nu x)(\hat{D} \mid \hat{E}) :: T \quad \text{(I-10)}$$

$$\Gamma;\Delta, y{:}A \,\&\, B \vdash (\nu x)(\hat{D} \mid y.\mathtt{inr}; \hat{F}) \simeq_c y.\mathtt{inr}; (\nu x)(\hat{D} \mid \hat{F}) :: T \quad \text{(I-11)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid z.\mathtt{inl}; \hat{E}) \simeq_c z.\mathtt{inl}; (\nu x)(\hat{D} \mid \hat{E}) :: z{:}A \oplus B \quad \text{(I-12)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid z.\mathtt{inr}; \hat{F}) \simeq_c z.\mathtt{inr}; (\nu x)(\hat{D} \mid \hat{F}) :: z{:}A \oplus B \quad \text{(I-13)}$$

$$\Gamma;\Delta, y{:}A \oplus B \vdash (\nu x)(\hat{D} \mid y.\mathtt{case}(\hat{E}, \hat{F})) \simeq_c y.\mathtt{case}((\nu x)(\hat{D} \mid \hat{E}), (\nu x)(\hat{D} \mid \hat{F})) :: T \quad \text{(I-14)}$$

$$\Gamma, u{:}A;\Delta \vdash (\nu x)(\hat{D} \mid \overline{u}\langle y\rangle.\hat{E}) \simeq_c \overline{u}\langle y\rangle.(\nu x)(\hat{D} \mid \hat{E}) :: T \quad \text{(I-15)}$$

$$\Gamma;\Delta, y{:}A \multimap B \vdash (\nu x)(y(z).\hat{D} \mid \hat{F}) \simeq_c y(z).(\nu x)(\hat{D} \mid \hat{F}) :: T \quad \text{(I-16)}$$

$$\Gamma;\Delta, y{:}A \multimap B \vdash (\nu x)(\overline{y}\langle z\rangle.(\hat{D} \mid \hat{E}) \mid \hat{F}) \simeq_c \overline{y}\langle z\rangle.(\hat{D} \mid (\nu x)(\hat{E} \mid \hat{F})) :: T \quad \text{(I-17)}$$

$$\Gamma;\Delta, y{:}A \,\&\, B \vdash (\nu x)(y.\mathtt{inl}; \hat{D} \mid \hat{F}) \simeq_c y.\mathtt{inl}; (\nu x)(\hat{D} \mid \hat{F}) :: T \quad \text{(I-18)}$$

$$\Gamma;\Delta, y{:}A \,\&\, B \vdash (\nu x)(y.\mathtt{inr}; \hat{D} \mid \hat{F}) \simeq_c y.\mathtt{inr}; (\nu x)(\hat{D} \mid \hat{F}) :: T \quad \text{(I-19)}$$

$$\Gamma;\Delta, y{:}A \oplus B \vdash (\nu x)(y.\mathtt{case}(\hat{D}, \hat{E}) \mid \hat{F}) \simeq_c y.\mathtt{case}((\nu x)(\hat{D} \mid \hat{F}), (\nu x)(\hat{E} \mid \hat{F})) :: T \quad \text{(I-20)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D}\{y/u\} \mid \hat{E}) \simeq_c (\nu x)(\hat{D} \mid \hat{E})\{y/u\} :: T \quad \text{(I-21)}$$

$$\Gamma;\Delta \vdash (\nu x)(\hat{D} \mid \hat{E}\{y/u\}) \simeq_c (\nu x)(\hat{D} \mid \hat{E})\{y/u\} :: T \quad \text{(I-22)}$$

$$\Gamma, u{:}A;\Delta \vdash (\nu x)(\overline{u}\langle y\rangle.\hat{D} \mid \hat{F}) \simeq_c \overline{u}\langle y\rangle.(\nu x)(\hat{D} \mid \hat{F}) :: T \quad \text{(I-23)}$$

$$\Gamma, u{:}A;\Delta \vdash (\nu x)(\hat{D} \mid \overline{u}\langle y\rangle.\hat{F}) \simeq_c \overline{u}\langle y\rangle.(\nu x)(\hat{D} \mid \hat{F}) :: T \quad \text{(I-24)}$$

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. $(\nu u)(!u(y).Q \mid P) \equiv_! P \quad$ *if* $u \notin \mathit{fn}(P)$

These axioms are called the *sharpened replication axioms* [41] and are known to

Figure 5 Process equalities induced by proof conversions, first kind (third subgroup)

$$\Gamma;\cdot \vdash (\nu u)((!u(y).\hat{D}) \mid \mathbf{0}) \simeq_c \mathbf{0} :: -:\mathbf{1} \tag{I-25}$$

$$\Gamma;\Delta \vdash (\nu u)((!u(y).\hat{D}) \mid \hat{E}) \simeq_c (\nu u)((!u(y).\hat{D}) \mid \hat{E}) :: T \tag{I-26}$$

$$\Gamma;\Delta \vdash (\nu u)((!u(y).\hat{D}) \mid \overline{x}\langle z\rangle.(\hat{E} \mid \hat{F})) \simeq_c \overline{x}\langle z\rangle.((\nu u)((!u(y).\hat{D}) \mid \hat{E}) \mid (\nu u)((!u(y).\hat{D}) \mid \hat{F})) :: x{:}A \otimes B \tag{I-27}$$

$$\Gamma;\Delta, y{:}A \otimes B \vdash (\nu u)((!u(y).\hat{D}) \mid y(z).\hat{E}) \simeq_c y(z).(\nu u)((!u(y).\hat{D}) \mid \hat{E}) :: T \tag{I-28}$$

$$\Gamma;\Delta \vdash (\nu u)(!u(y).\hat{D} \mid z(y).\hat{E}) \simeq_c z(y).(\nu u)(!u(y).\hat{D} \mid \hat{E}) :: z{:}A \multimap B \tag{I-29}$$

$$\Gamma;\Delta, y{:}A \multimap B \vdash (\nu u)((!u(y).\hat{D}) \mid \overline{y}\langle z\rangle.(\hat{E} \mid \hat{F})) \simeq_c \overline{y}\langle z\rangle.(((\nu u)(!u(y).\hat{D} \mid \hat{E}) \mid (\nu u)((!u(y).\hat{D}) \mid \hat{F}))) :: T \tag{I-30}$$

$$\Gamma;\Delta \vdash (\nu u)((!u(y).\hat{D}) \mid z.\mathtt{case}(\hat{E},\hat{F})) \simeq_c z.\mathtt{case}((\nu u)((!u(y).\hat{D}) \mid \hat{E}),(\nu u)((!u(y).\hat{D}) \mid \hat{F}))::z{:}A \,\&\, B \tag{I-31}$$

$$\Gamma;\Delta, y{:}A \,\&\, B \vdash (\nu u)(!u(z).\hat{D} \mid y.\mathtt{inl};\hat{E}) \simeq_c y.\mathtt{inl};(\nu u)(!u(z).\hat{D} \mid \hat{E}) :: T \tag{I-32}$$

$$\Gamma;\Delta, y{:}A \,\&\, B \vdash (\nu u)(!u(z).\hat{D} \mid y.\mathtt{inr};\hat{F}) \simeq_c y.\mathtt{inr};(\nu u)(!u(z).\hat{D} \mid \hat{F}) :: T \tag{I-33}$$

$$\Gamma;\Delta \vdash (\nu u)(!u(y).\hat{D} \mid z.\mathtt{inl};\hat{E}) \simeq_c z.\mathtt{inl};(\nu u)(!u(y).\hat{D} \mid \hat{E}) :: z{:}A \oplus B \tag{I-34}$$

$$\Gamma;\Delta \vdash (\nu u)(!u(y).\hat{D} \mid z.\mathtt{inr};\hat{F}) \simeq_c z.\mathtt{inr};(\nu u)(!u(y).\hat{D} \mid \hat{F}) :: z{:}A \oplus B \tag{I-35}$$

$$\Gamma;\Delta, y{:}A \oplus B \vdash (\nu u)(!u(z).\hat{D} \mid y.\mathtt{case}(\hat{E},\hat{F})) \simeq_c y.\mathtt{case}((\nu u)(!u(z).\hat{D} \mid \hat{E}),(\nu u)(!u(z).\hat{D} \mid \hat{F})) :: T \tag{I-36}$$

$$\Gamma;\Delta \vdash (\nu u)(!u(y).\hat{D} \mid !x(z).\hat{E}) \simeq_c !x(z).(\nu u)(!u(y).\hat{D} \mid \hat{E}) :: x{:}!A \tag{I-37}$$

$$\Gamma;\Delta \vdash (\nu u)(!u(y).\hat{D} \mid \hat{E}\{y/v\}) \simeq_c (\nu u)(!u(y).\hat{D} \mid \hat{E})\{y/v\}::x{:}T \tag{I-38}$$

$$\Gamma;\Delta \vdash (\nu u)(!u(y).\hat{D} \mid \overline{v}\langle y\rangle.\hat{E}) \simeq_c \overline{v}\langle y\rangle.(\nu u)(!u(y).\hat{D} \mid \hat{E})) :: T \tag{I-39}$$

express sound behavioral equivalences up to strong bisimilarity in our typed setting. Intuitively, (1) and (2) represent principles for the distribution of shared servers among processes, while (3) formalizes the garbage collection of shared servers which cannot be invoked by any process. Notice that $\equiv_!$ was defined in [10] (Def 4.3), and noted $\simeq_s$.

Proposition 5.1. *Let P and Q be well-typed processes.*

1. *If $P \longrightarrow P'$ and $P \equiv_! Q$ then there is Q' such that $Q \longrightarrow Q'$ 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 $P' \equiv_! Q'$.*

Proof. By induction on the derivation of $P \equiv_! Q$, then by case analysis on $\longrightarrow$ and $\xrightarrow{\alpha}$, respectively. □

Proposition 5.2. *If $P\Downarrow$ and $P \equiv_! Q$ then $Q\Downarrow$.*

Figure 6 Process equalities induced by proof conversions, second kind (first subgroup).

$$\Gamma; \Delta, x{:}A \otimes B, z{:}C \otimes D \vdash x(y).z(w).P \simeq_c z(w).x(y).P :: T \quad \text{(II-1)}$$

$$\Gamma; \Delta, z{:}D \multimap C, x{:}A \multimap B \vdash \overline{z}\langle w\rangle.(R \mid \overline{x}\langle y\rangle.(P \mid Q)) \simeq_c \overline{x}\langle y\rangle.(P \mid \overline{z}\langle w\rangle.(R \mid Q)) :: T \quad \text{(II-2)}$$

$$\Gamma; \Delta, z{:}D \multimap C, x{:}A \multimap B \vdash \overline{z}\langle w\rangle.(R \mid \overline{x}\langle y\rangle.(P \mid Q)) \simeq_c \overline{x}\langle y\rangle.(\overline{z}\langle w\rangle.(R \mid P) \mid Q) :: T \quad \text{(II-3)}$$

$$\Gamma; \Delta, w{:}C \multimap D, x{:}A \otimes B \vdash \overline{w}\langle z\rangle.(Q \mid x(y).P) \simeq_c x(y).\overline{w}\langle z\rangle.(Q \mid P) :: T \quad \text{(II-4)}$$

$$\Gamma; \Delta, w{:}C \multimap D, x{:}A \otimes B \vdash \overline{w}\langle z\rangle.(x(y).P \mid Q) \simeq_c x(y).\overline{w}\langle z\rangle.(P \mid Q) :: T \quad \text{(II-5)}$$

$$\Gamma, u{:}A, v{:}C; \Delta \vdash \overline{u}\langle y\rangle.\overline{v}\langle x\rangle.P \simeq_c \overline{v}\langle x\rangle.\overline{u}\langle y\rangle.P :: T \quad \text{(II-6)}$$

$$\Gamma, u{:}C; \Delta, x{:}A \multimap B \vdash \overline{u}\langle z\rangle.\overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(\overline{u}\langle z\rangle.P \mid Q) :: T \quad \text{(II-7)}$$

$$\Gamma, u{:}C; \Delta, x{:}A \multimap B \vdash \overline{u}\langle z\rangle.\overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(P \mid \overline{u}\langle z\rangle.Q) :: T \quad \text{(II-8)}$$

$$\Gamma, u{:}A; \Delta, z{:}C \otimes D \vdash \overline{u}\langle y\rangle.z(w).P \simeq_c z(w).\overline{u}\langle y\rangle.P :: T \quad \text{(II-9)}$$

$$\Gamma; \Delta, x{:}A \oplus B, y{:}C \oplus D \vdash y.\mathtt{case}(x.\mathtt{case}(P_1, Q_1), x.\mathtt{case}(P_2, Q_2)) \simeq_c x.\mathtt{case}(y.\mathtt{case}(P_1, P_2), y.\mathtt{case}(Q_1, Q_2)) :: T \quad \text{(II-10)}$$

$$\Gamma, u{:}C; \Delta, x{:}A \oplus B \vdash \overline{u}\langle z\rangle.x.\mathtt{case}(P, Q) \simeq_c x.\mathtt{case}(\overline{u}\langle z\rangle.P\,,\ \overline{u}\langle z\rangle.Q) :: T \quad \text{(II-11)}$$

$$\Gamma; \Delta, w{:}A \multimap E, z{:}C \,\&\, D \vdash z.\mathtt{case}(\overline{w}\langle y\rangle.(P \mid R_1)\,,\ \overline{w}\langle y\rangle.(P \mid R_2)) \simeq_c \overline{w}\langle y\rangle.(P \mid z.\mathtt{case}(R_1 \mid R_2)) :: T \quad \text{(II-12)}$$

$$\Gamma; \Delta, z{:}C \oplus D, x{:}A \otimes B \vdash z.\mathtt{case}(x(y).P, x(y).Q) \simeq_c x(y).z.\mathtt{case}(P, Q) :: T \quad \text{(II-13)}$$

$$\Gamma; \Delta, x{:}A \,\&\, B, z{:}C \,\&\, D \vdash x.\mathtt{inl}; y.\mathtt{inl}; P \simeq_c y.\mathtt{inl}; x.\mathtt{inl}; P :: T \quad \text{(II-14)}$$

$$\Gamma; \Delta, x{:}A \oplus B, y{:}C \,\&\, D \vdash x.\mathtt{case}(y.\mathtt{inl}; P, y.\mathtt{inl}; Q) \simeq_c y.\mathtt{inl}; x.\mathtt{case}(R, Q) :: T \quad \text{(II-15)}$$

$$\Gamma, u{:}C; \Delta, x{:}A \,\&\, B \vdash z.\mathtt{inl}; \overline{u}\langle y\rangle.P \simeq_c \overline{u}\langle y\rangle.z.\mathtt{inl}; P :: T \quad \text{(II-16)}$$

$$\Gamma; \Delta, z{:}C \,\&\, D, x{:}A \multimap B \vdash z.\mathtt{inl}; \overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(z.\mathtt{inl}; P \mid Q) :: T \quad \text{(II-17)}$$

$$\Gamma; \Delta, z{:}C \,\&\, D, x{:}A \multimap B \vdash z.\mathtt{inl}; \overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(P \mid z.\mathtt{inl}; Q) :: T \quad \text{(II-18)}$$

$$\Gamma; \Delta, z{:}C \,\&\, D, x{:}A \otimes B \vdash z.\mathtt{inl}; x(y).P \simeq_c x(y).z.\mathtt{inl}; P :: T \quad \text{(II-19)}$$

Proof. Follows by Prop 5.1, by noticing that: (i) axioms (1) and (2) of $\equiv_!$ do not add new input-guarded replicated processes and (ii) axiom (3) may add a new input-guarded replicated process (if read from right to left) which cannot be invoked. □

5.2. Logical Relations for Strong Normalization of Well-typed Processes

We now introduce a theory of linear logical relations for session-typed processes, and use it to prove strong normalization.

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

Figure 7 Process equalities induced by proof conversions, second kind (second subgroup).

$$\Gamma; \Delta, z{:}C \,\&\, D \vdash z.\mathtt{inl}; \overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(P \mid z.\mathtt{inl}; Q) :: x{:}A \otimes B \quad \text{(II-20)}$$

$$\Gamma; \Delta, z{:}C \,\&\, D \vdash z.\mathtt{inl}; \overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(z.\mathtt{inl}; P \mid Q) :: x{:}A \otimes B \quad \text{(II-21)}$$

$$\Gamma; \Delta, z{:}D \oplus E \vdash z.\mathtt{case}(\overline{x}\langle y\rangle.(P_1 \mid Q)\,,\, \overline{x}\langle y\rangle.(P_2 \mid Q)) \simeq_c \overline{x}\langle y\rangle.(Q \mid z.\mathtt{case}(P_1\,,\, P_2)) :: x{:}A \otimes B \quad \text{(II-22)}$$

$$\Gamma; \Delta, z{:}D \oplus E \vdash z.\mathtt{case}(\overline{x}\langle y\rangle.(Q \mid P_1)\,,\, \overline{x}\langle y\rangle.(Q \mid P_2)) \simeq_c \overline{x}\langle y\rangle.(z.\mathtt{case}(P_1\,,\, P_2) \mid Q) :: x{:}A \otimes B \quad \text{(II-23)}$$

$$\Gamma, u{:}C; \Delta \vdash \overline{u}\langle w\rangle.\overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(\overline{u}\langle w\rangle.P \mid Q) :: x{:}A \otimes B \quad \text{(II-24)}$$

$$\Gamma, u{:}C; \Delta \vdash \overline{u}\langle w\rangle.\overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(P \mid \overline{u}\langle w\rangle.Q) :: x{:}A \otimes B \quad \text{(II-25)}$$

$$\Gamma; \Delta, w{:}C \multimap D \vdash \overline{w}\langle z\rangle.(R \mid \overline{x}\langle y\rangle.(P \mid Q)) \simeq_c \overline{x}\langle y\rangle.(P \mid \overline{w}\langle z\rangle.(R \mid Q)) :: x{:}A \otimes B \quad \text{(II-26)}$$

$$\Gamma; \Delta, x{:}C \multimap D \vdash \overline{z}\langle y\rangle.(\overline{x}\langle w\rangle.(P \mid Q) \mid R) \simeq_c \overline{x}\langle w\rangle.(P \mid \overline{z}\langle y\rangle.(R \mid Q)) :: z{:}A \otimes B \quad \text{(II-27)}$$

$$\Gamma; \Delta, z{:}C \otimes D \vdash z(w).\overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(z(w).P \mid Q) :: x{:}A \otimes B \quad \text{(II-28)}$$

$$\Gamma; \Delta, z{:}C \otimes D \vdash z(w).\overline{x}\langle y\rangle.(P \mid Q) \simeq_c \overline{x}\langle y\rangle.(P \mid z(w).Q) :: x{:}A \otimes B \quad \text{(II-29)}$$

$$\Gamma; \Delta, z{:}C \,\&\, D \vdash z.\mathtt{inl}; x(y).P \simeq_c x(y).z.\mathtt{inl}; P :: x{:}A \multimap B \quad \text{(II-30)}$$

$$\Gamma; \Delta, z{:}C \oplus D \vdash x(y).z.\mathtt{case}(P, Q) \simeq_c z.\mathtt{case}(x(y).P\,,\, x(y).Q) :: x{:}A \multimap B \quad \text{(II-31)}$$

$$\Gamma, u{:}C; \Delta \vdash \overline{u}\langle w\rangle.x(y).P \simeq_c x(y).\overline{u}\langle w\rangle.P :: x{:}A \multimap B \quad \text{(II-32)}$$

$$\Gamma; \Delta, w{:}C \multimap D \vdash \overline{w}\langle z\rangle.(R \mid x(y).P)) \simeq_c x(y).\overline{w}\langle z\rangle.(R \mid P)) :: x{:}A \multimap B \quad \text{(II-33)}$$

$$\Gamma; \Delta, z{:}C \otimes D \vdash x(y).z(w).P \simeq_c z(w).x(y).P :: x{:}A \multimap B \quad \text{(II-34)}$$

$$\Gamma; \Delta, y{:}C \,\&\, D \vdash y.\mathtt{inl}; x.\mathtt{case}(P, Q) \simeq_c x.\mathtt{case}(y.\mathtt{inl}; P\,,\, y.\mathtt{inl}; Q) :: x{:}A \,\&\, B \quad \text{(II-35)}$$

$$\Gamma; \Delta, y{:}C \oplus D \vdash x.\mathtt{case}(y.\mathtt{case}(P_1, Q_1), y.\mathtt{case}(P_2, Q_2)) \simeq_c y.\mathtt{case}(x.\mathtt{case}(P_1, P_2), x.\mathtt{case}(Q_1, Q_2)) :: x{:}A \,\&\, B \quad \text{(II-36)}$$

$$\Gamma; u{:}A; \Delta \vdash x.\mathtt{case}(\overline{u}\langle y\rangle.P\,,\, \overline{u}\langle y\rangle.Q) \simeq_c \overline{u}\langle y\rangle.x.\mathtt{case}(P, Q) :: x{:}A \,\&\, B \quad \text{(II-37)}$$

$$\Gamma; \Delta, z{:}C \multimap D \vdash \overline{z}\langle y\rangle.(R \mid x.\mathtt{case}(P, Q)) \simeq_c x.\mathtt{case}(\overline{z}\langle y\rangle.(R \mid P)\,,\, \overline{z}\langle y\rangle.(R \mid Q)) :: x{:}A \,\&\, B \quad \text{(II-38)}$$

$$\Gamma; \Delta, x{:}A \otimes B \vdash z.\mathtt{case}(x(y).P\,,\, x(y).Q) \simeq_c x(y).z.\mathtt{case}(P, Q) :: z{:}C \,\&\, D \quad \text{(II-39)}$$

$$\Gamma; \Delta, y{:}C \,\&\, D \vdash y.\mathtt{inl}; x.\mathtt{inl}; P \simeq_c x.\mathtt{inl}; y.\mathtt{inl}; P :: x{:}A \oplus B \quad \text{(II-40)}$$

$$\Gamma; \Delta, y{:}A \oplus B \vdash x.\mathtt{inl}; y.\mathtt{case}(P, Q) \simeq_c y.\mathtt{case}(x.\mathtt{inl}; P, x.\mathtt{inl}; Q) :: x{:}A \oplus B \quad \text{(II-41)}$$

$$\Gamma; u{:}A; \Delta \vdash x.\mathtt{inl}; \overline{u}\langle y\rangle.P \simeq_c \overline{u}\langle y\rangle.x.\mathtt{inl}; P :: x{:}A \oplus B \quad \text{(II-42)}$$

$$\Gamma; \Delta, z{:}D \multimap C \vdash \overline{z}\langle y\rangle.(Q \mid x.\mathtt{inl}; P) \simeq_c x.\mathtt{inl}; \overline{z}\langle y\rangle.(Q \mid P) :: x{:}A \oplus B \quad \text{(II-43)}$$

$$\Gamma; \Delta, x{:}A \otimes B \vdash x(y).z.\mathtt{inl}; P \simeq_c z.\mathtt{inl}; x(y).P :: z{:}C \oplus D \quad \text{(II-44)}$$

$$\Gamma; \Delta, x{:}!C \vdash \overline{z}\langle y\rangle.(P\{x/u\} \mid Q) \simeq_c (\overline{z}\langle y\rangle.(P \mid Q))\{x/u\} :: z{:}A \otimes B \quad \text{(II-45)}$$

$$\Gamma; \Delta, x{:}!C \vdash \overline{z}\langle y\rangle.(P \mid Q\{x/u\}) \simeq_c (\overline{z}\langle y\rangle.(P \mid Q))\{x/u\} :: z{:}A \otimes B \quad \text{(II-46)}$$

typing, where we abbreviate $\mathcal{L}[\Gamma; \Delta \vdash T]$ by $\mathcal{L}[T]$. We write $P \not\longrightarrow$ to mean that P cannot reduce; it may perform visible actions, though.

Definition 5.3 (Logical Predicate for Termination, Base case)**.** *For any type $T = z{:}A$ we inductively define $\mathcal{L}[T]$ as the set of all P such that $P\Downarrow$ and $\cdot;\cdot \vdash P :: T$ and*

$$
\begin{array}{rcl}
P \in \mathcal{L}[z{:}\mathbf{1}] & \textit{iff} & \forall P'.(P \Longrightarrow P' \wedge P' \not\longrightarrow) \Rightarrow P' \equiv_! \mathbf{0} \\
P \in \mathcal{L}[z{:}A \multimap B] & \textit{iff} & \forall P', y.(P \stackrel{z(y)}{\Longrightarrow} P') \Rightarrow \forall Q \in \mathcal{L}[y{:}A].(\nu y)(P' \mid Q) \in \mathcal{L}[z{:}B] \\
P \in \mathcal{L}[z{:}A \otimes B] & \textit{iff} & \forall P', y.(P \stackrel{\overline{z\langle y\rangle}}{\Longrightarrow} P') \Rightarrow \\
 & & \exists P_1, P_2.(P' \equiv_! P_1 \mid P_2 \wedge P_1 \in \mathcal{L}[y{:}A] \wedge P_2 \in \mathcal{L}[z{:}B]) \\
P \in \mathcal{L}[z{:}!A] & \textit{iff} & \forall P'.(P \Longrightarrow P') \Rightarrow \exists P_1.(P' \equiv_! !z(y).P_1 \wedge P_1 \in \mathcal{L}[y{:}A]) \\
P \in \mathcal{L}[z{:}A \,\&\, B] & \textit{iff} & (\forall P'.(P \stackrel{z.\mathsf{inl}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}[z{:}A]) \wedge \\
 & & (\forall P'.(P \stackrel{z.\mathsf{inr}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}[z{:}B]) \\
P \in \mathcal{L}[z{:}A \oplus B] & \textit{iff} & (\forall P'.(P \stackrel{\overline{z.\mathsf{inl}}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}[z{:}A]) \wedge \\
 & & (\forall P'.(P \stackrel{\overline{z.\mathsf{inr}}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}[z{:}B])
\end{array}
$$

Some comments are in order. First, observe how the definition of $\mathcal{L}[T]$ relies on both reductions and weak transitions, and the fact 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 shared 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 definitions 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 adhere to the principles of rules (Tcut) and (Tcut!) for this purpose.

Definition 5.4 (Logical Predicate for Termination, Inductive case)**.** *For any sequent $\Gamma;\Delta \vdash T$ with a non-empty left-hand side environment, we define $\mathcal{L}[\Gamma;\Delta \vdash T]$ as the set of processes inductively defined as follows:*

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

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

Definition 5.5. *Let $\Gamma = u_1{:}B_1, \ldots, u_k{:}B_k$, and $\Delta = x_1{:}A_1, \ldots, x_n{:}A_n$ be a non-linear and a linear typing environment, resp. Letting $I = \{1, \ldots, k\}$, $J = \{1, \ldots, n\}$, we define the sets of processes $\mathcal{C}_\Gamma$ and $\mathcal{C}_\Delta$ as:*

$$
\mathcal{C}_\Gamma \stackrel{def}{=} \Big\{ \prod_{i \in I} !u_i(y_i).R_i \;\mid R_i \in \mathcal{L}[y_i{:}B_i] \Big\} \qquad \mathcal{C}_\Delta \stackrel{def}{=} \Big\{ \prod_{j \in J} Q_j \;\mid Q_j \in \mathcal{L}[x_j{:}A_j] \Big\}
$$

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

Proposition 5.3. *Let Γ and 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, $Q \not\longrightarrow$.*

Proof. By Definition 5.5, every process in $\mathcal{C}_\Delta$ corresponds to the composition of non-interfering, terminating processes. Hence, $R\Downarrow$. The same applies for processes in $\mathcal{C}_\Gamma$, which, by construction, correspond to the composition of input-guarded replicated processes. Hence, $Q\Downarrow, Q \not\longrightarrow$. □

The proof of the following lemma is immediate from Definitions 5.4 and 5.5.

Lemma 5.1. *Let P be a process such that $\Gamma; \Delta \vdash P :: T$, with $\Gamma = u_1{:}B_1, \ldots, u_k{:}B_k$ and $\Delta = x_1{:}A_1, \ldots, x_n{:}A_n$. We then have:*

$$P \in \mathcal{L}[\Gamma; \Delta \vdash T] \text{ iff } \forall Q \in \mathcal{C}_\Gamma, \forall R \in \mathcal{C}_\Delta, (\nu \widetilde{u}, \widetilde{x})(P \mid Q \mid R) \in \mathcal{L}[T].$$

The following closure properties will be fundamental in the second step of the proof, when we will show that well-typed processes are in the logical predicate. We first state closure of $\mathcal{L}[T]$ with respect to substitution and structural congruence:

Proposition 5.4. *Let A be a type. If $P \in \mathcal{L}[z{:}A]$ then $P\{^x/_z\} \in \mathcal{L}[x{:}A]$.*

Proof. Immediate from Definition 5.3. □

Proposition 5.5. *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 $P \equiv Q$, using Propositions 5.1 and 5.2, and the fact that well-typed processes are closed under $\equiv$ by definition. □

The next proposition provides a basic liveness guarantee for typed processes.

Proposition 5.6. *Let $\cdot; \cdot \vdash P :: z{:}T$ and $P\Downarrow$, with $T \in \{A \otimes B, A \multimap B, A \oplus B, A \,\&\, B\}$. Then, there exist α, P' such that*

(i) $P \stackrel{\alpha}{\Longrightarrow} P'$, *and*

(ii) if $T{=}A \otimes B$ then $\alpha = \overline{z\langle y \rangle}$; if $T{=}A \multimap B$ then $\alpha = z(y)$; if $T{=}A \oplus B$ then $\alpha = \overline{z.\mathsf{inr}}$ or $\alpha = \overline{z.\mathsf{inl}}$; if $T{=}A \,\&\, B$ then $\alpha = z.\mathsf{inr}$ or $\alpha = z.\mathsf{inl}$.

Proof. Since $T \notin \{\mathbf{1}, !T'\}$ then, using Lemma 3.4, we know that $live(P)$ holds. Hence, Lemma 3.1 can be used to infer that either $P \longrightarrow P'$ or $P \stackrel{\alpha}{\rightarrow} P'$, 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 3.2. □

We now extend Proposition 5.5 so as to state closure of $\mathcal{L}[T]$ under $\equiv_!$.

Proposition 5.7. *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 $P \equiv_! Q$. See Appendix A.1, Page 42. □

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

Proof. By induction on the structure of type T. In all cases, we must show that: (i) $P'\Downarrow$; (ii) P' is well-typed; and (iii) $P' \in \mathcal{L}[T]$. First, by assumption and Def. 5.3, we have that $P\Downarrow$; then, since $P \longrightarrow P'$, we have $P'\Downarrow$ as well. Well-typedness follows from Theorem 3.1. Finally, $P' \in \mathcal{L}[T]$ follows by definition of weak transition. □

Proposition 5.9 (Backward Closure)**.** *If for all P_i such that $P \longrightarrow P_i$ we have $P_i \in \mathcal{L}[T]$ then $P \in \mathcal{L}[T]$.*

Proof. By induction on the structure of type T. In all cases, we must show that: (i) $P\Downarrow$; (ii) P is well-typed; and (iii) $P \in \mathcal{L}[T]$. Items (i) and (iii) are immediate from Definition 5.1 and subject reduction (Theorem 3.1), respectively. Item (iii) follows by definition of weak transition, noticing that if $P' \stackrel{\alpha}{\Longrightarrow} P''$ and $P \longrightarrow P'$ then also $P \stackrel{\alpha}{\Longrightarrow} P''$ holds. □

The final closure property concerns parallel composition of processes:

Proposition 5.10 (Weakening)**.** *Let P, Q be processes such that $P \in \mathcal{L}[T]$ and $Q \in \mathcal{L}[-:\mathbf{1}]$. Then, $P \mid Q \in \mathcal{L}[T]$.*

Proof. By induction on the structure of type T. See Appendix A.2, Page 44. □

Second Step: Well-typed Processes are in the Logical Predicate. We now prove that well-typed processes are in the logical predicate. Because of Definition 5.3, termination of well-typed processes will follow as a consequence.

Lemma 5.2. *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. We have 18 cases to check; in all of them, we use Lemma 5.1 to show that every $M = (\nu \widetilde{u}, \widetilde{x})(P \mid G \mid D)$ with $G \in \mathcal{C}_\Gamma$ and $D \in \mathcal{C}_\Delta$, is in $\mathcal{L}[T]$. In case (Tid), we use Proposition 5.4 (closure wrt substitution) and Proposition 5.9 (backward closure). In cases (T$\otimes$L), (T$\multimap$L), (Tcopy), (T$\oplus$L), (T$\&$L$_1$), and (T$\&$L$_2$), we proceed in two steps: first, using Proposition 5.8 (forward closure) we show that every M'' such that $M \Longrightarrow M''$ is in $\mathcal{L}[T]$; then, we combine this result with Proposition 5.9 (backward closure) to conclude that $M \in \mathcal{L}[T]$. In cases (T$\mathbf{1}$R), (T$\otimes$R), (T$\multimap$R), (T!R), (T$\oplus$R$_1$), and (T$\oplus$R$_2$), we show that M conforms to a specific case of Definition 5.3. Case (T$\mathbf{1}$L) uses Proposition 5.10 (weakening). Cases (T$\otimes$L), (T$\multimap$L), (T$\oplus$L), and (T$\&$L$_1$) use the liveness guarantee given by Proposition 5.6. Cases (Tcopy), (T!L), and (Tcut$^!$) use Proposition 5.5 (closure under $\equiv$). Cases (Tcut), (T$\multimap$R), and (T!R) use Proposition 5.7 (closure under $\equiv_!$). See Appendix A.3, Page 42 for details. □

We now state our first main result: well-typed processes terminate.

Theorem 5.1 (Well-typed Processes are Terminating)**.** *If $\Gamma; \Delta \vdash P :: T$ then $P\Downarrow$.*

Proof. Follows from previously proven facts:

$\Gamma; \Delta \vdash P :: T$	[Assumption]	(a)
$P \in \mathcal{L}[\Gamma; \Delta \vdash T]$	[By Lemma 5.2 and (a)]	(b)
Pick any $G \in \mathcal{C}_\Gamma$, $D \in \mathcal{C}_\Delta$:		
$G\Downarrow, D\Downarrow$	[By Prop 5.3]	(c)
$(\nu \widetilde{u}, \widetilde{x})(P \mid G \mid D) \in \mathcal{L}[T]$	[By Lemma 5.1 on (b)]	(d)
$(\nu \widetilde{u}, \widetilde{x})(P \mid G \mid D)\Downarrow$	[From (d) and Def 5.3]	(e)
$P\Downarrow$	[Consequence of (c) and (e)]	

□

5.3. Well-typed Processes are Confluent

We now adapt the logical relations and the proof technique of Section 5.2 to the case of *confluence*. The required adjustments concern mainly closure properties.

Definition 5.6 (Confluence)**.** *A process P is* confluent*, written $P\diamondsuit$, if for any P_1, P_2 such that $P \Longrightarrow P_1$ and $P \Longrightarrow P_2$, there exists a P' such that $P_1 \Longrightarrow P'$ and $P_2 \Longrightarrow P'$.*

Proposition 5.11 (Properties of Confluent Processes)**.** *Assume well-typed processes $P, P', P_1, \ldots, P_k, Q$.*

1. *Forward closure: If $P\diamondsuit$ and $P \longrightarrow P'$ then $P'\diamondsuit$.*
2. *Backward closure: If for all P_i such that $P \longrightarrow P_i$ we have that $P_i\diamondsuit$, then $P\diamondsuit$.*
3. *Closure wrt composition: Let P, Q be such that (i) $\cdot; \cdot \vdash P :: x{:}A$, (ii) $\cdot; x{:}A \vdash Q :: T$, (iii) $P\diamondsuit$, and (iv) $Q\diamondsuit$. Then $(\nu x)(P \mid Q)\diamondsuit$.*

Proof. See Appendix A.4 (Page 54) for details. □

Proposition 5.12. *If $P\diamondsuit$ and $P \equiv_! Q$ then $Q\diamondsuit$.*

Proof. Follows immediately from Proposition 5.1. □

First Step: The Logical Predicate and its Closure Properties. The following logical predicate for confluence is essentially the same as for termination (cf. Definition 5.3). Hence, subsequent auxiliary definitions and closure properties mirror those in Section 5.2.

Definition 5.7 (Logical Predicate for Confluence, Base case)**.** *For any type $T = z{:}A$ we inductively define $\mathcal{L}^{\Diamond}[T]$ as the set of all P such that $P\Diamond$ and $\cdot;\cdot \vdash P :: T$ and*

$$
\begin{array}{rcl}
P \in \mathcal{L}^{\Diamond}[z{:}\mathbf{1}] & \textit{iff} & \forall P'.(P \Longrightarrow P' \wedge P' \not\longrightarrow) \Rightarrow P' \equiv_! \mathbf{0} \\
P \in \mathcal{L}^{\Diamond}[z{:}A \multimap B] & \textit{iff} & \forall P' y.(P \overset{z(y)}{\Longrightarrow} P') \Rightarrow \forall Q \in \mathcal{L}^{\Diamond}[y{:}A].(\nu y)(P' \mid Q) \in \mathcal{L}^{\Diamond}[z{:}B] \\
P \in \mathcal{L}^{\Diamond}[z{:}A \otimes B] & \textit{iff} & \forall P' y.(P \overset{\overline{z\langle y\rangle}}{\Longrightarrow} P') \Rightarrow \\
 & & \exists P_1, P_2.(P' \equiv_! P_1 \mid P_2 \wedge P_1 \in \mathcal{L}^{\Diamond}[y{:}A] \wedge P_2 \in \mathcal{L}^{\Diamond}[z{:}B]) \\
P \in \mathcal{L}^{\Diamond}[z{:}!A] & \textit{iff} & \forall P'.(P \Longrightarrow P') \Rightarrow \exists P_1.(P' \equiv_! !z(y).P_1 \wedge P_1 \in \mathcal{L}^{\Diamond}[y{:}A]) \\
P \in \mathcal{L}^{\Diamond}[z{:}A \,\&\, B] & \textit{iff} & (\forall P'.(P \overset{z.\mathsf{inl}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}^{\Diamond}[z{:}A]) \wedge \\
 & & (\forall P'.(P \overset{z.\mathsf{inr}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}^{\Diamond}[z{:}B]) \\
P \in \mathcal{L}^{\Diamond}[z{:}A \oplus B] & \textit{iff} & (\forall P'.(P \overset{\overline{z.\mathsf{inl}}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}^{\Diamond}[z{:}A]) \wedge \\
 & & (\forall P'.(P \overset{\overline{z.\mathsf{inr}}}{\Longrightarrow} P') \Rightarrow P' \in \mathcal{L}^{\Diamond}[z{:}B])
\end{array}
$$

Below, we extend $\mathcal{L}^{\Diamond}[T]$ to arbitrary typing environments.

Definition 5.8 (Logical Predicate - Inductive case)**.** *For any sequent $\Gamma;\Delta \vdash T$ with a non-empty left hand side environment, we define $\mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T]$ to be the set of processes inductively defined as follows:*

$$
\begin{array}{l}
P \in \mathcal{L}^{\Diamond}[\Gamma; y{:}A, \Delta \vdash T] \textit{ if } \forall R \in \mathcal{L}^{\Diamond}[y{:}A].(\nu y)(R \mid P) \in \mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T] \\
P \in \mathcal{L}^{\Diamond}[u{:}A, \Gamma;\Delta \vdash T] \textit{ if } \forall R \in \mathcal{L}^{\Diamond}[y{:}A].(\nu u)(!u(y).R \mid P) \in \mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T]
\end{array}
$$

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

Definition 5.9. *Let $\Gamma = u_1{:}B_1, \ldots, u_k{:}B_k$, and $\Delta = x_1{:}A_1, \ldots, x_n{:}A_n$ be a non-linear and a linear typing environment, resp. Letting $I = \{1,\ldots,k\}$ and $J = \{1,\ldots,n\}$, we define the sets of processes $\mathcal{C}^{\Diamond}_{\Gamma}$ and $\mathcal{C}^{\Diamond}_{\Delta}$ as:*

$$
\mathcal{C}^{\Diamond}_{\Gamma} \overset{\textit{def}}{=} \Big\{ \prod_{i\in I} !u_i(y_i).R_i \;\mid R_i \in \mathcal{L}^{\Diamond}[y_i{:}B_i] \Big\} \qquad \mathcal{C}^{\Diamond}_{\Delta} \overset{\textit{def}}{=} \Big\{ \prod_{j\in J} Q_j \;\mid Q_j \in \mathcal{L}^{\Diamond}[x_j{:}A_j] \Big\}
$$

We define sets of processes $\mathcal{C}^{\Diamond}_{\Gamma}$ and $\mathcal{C}^{\Diamond}_{\Delta}$ as logical representatives of the behavior specified by Γ and Δ, respectively.

Proposition 5.13. *Let Γ and Δ be a non-linear and a linear typing environment, respectively. Then, for all $Q \in \mathcal{C}^{\Diamond}_{\Gamma}$ and for all $R \in \mathcal{C}^{\Diamond}_{\Delta}$, we have $Q\Diamond$ and $R\Diamond$. Furthermore, $Q \not\longrightarrow$ and $R\Downarrow$.*

Proof. By Definition 5.9, every $R \in \mathcal{C}^{\Diamond}_{\Delta}$ corresponds to the composition of independent, confluent processes. Hence, using Proposition 5.11 (3), we have $R\Diamond$. Also, R is the composition of well-typed processes, which by Theorem 5.1 are all terminating. Hence, $R\Downarrow$. As for $Q \in \mathcal{C}^{\Diamond}_{\Gamma}$, by construction it corresponds to the composition of input-guarded replicated processes. Hence, $Q \not\longrightarrow$. □

Lemma 5.3. *Let* $\Gamma;\Delta \vdash P::T$*, with* $\Gamma{=}u_1{:}B_1,\ldots,u_k{:}B_k$ *and* $\Delta{=}x_1{:}A_1,\ldots,x_n{:}A_n$*. We have:* $P \in \mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T]$ *iff* $\forall Q \in \mathcal{C}^{\Diamond}_{\Gamma}, \forall R \in \mathcal{C}^{\Diamond}_{\Delta}, (\nu \widetilde{u},\widetilde{x})(P \mid Q \mid R) \in \mathcal{L}^{\Diamond}[T]$.

Proof. The proof follows from Definitions 5.8 and 5.9, Proposition 5.13, and closure of confluent processes under composition (Proposition 5.11 (3)). □

We now state the closure properties required to show that well-typed processes are in the logical predicate for confluence.

Proposition 5.14. *Let* A *be a type. If* $P \in \mathcal{L}^{\Diamond}[z{:}A]$ *then* $P\{x/z\} \in \mathcal{L}^{\Diamond}[x{:}A]$.

Proof. Immediate from Definition 5.7. □

Proposition 5.15. *Let* P, Q *be well-typed. If* $P \in \mathcal{L}^{\Diamond}[T]$ *and* $P \equiv Q$ *then* $Q \in \mathcal{L}^{\Diamond}[T]$.

Proof. By induction the definition of $P \equiv Q$, using Propositions 5.1 and 5.12, and the fact that well-typed processes are closed under $\equiv$ by definition. □

We now extend Proposition 5.15 so as to state closure of $\mathcal{L}^{\Diamond}[T]$ under $\equiv_!$.

Proposition 5.16. *Let* P, Q *be well-typed. If* $P \in \mathcal{L}^{\Diamond}[T]$ *and* $P \equiv_! Q$ *then* $Q \in \mathcal{L}^{\Diamond}[T]$.

Proof. By induction the definition of $P \equiv_! Q$, following the lines of the proof of Proposition 5.7. □

We now state *forward* and *backward* closure of $\mathcal{L}^{\Diamond}[T]$ with respect to reduction.

Proposition 5.17 (Forward Closure)**.** *If* $P \in \mathcal{L}^{\Diamond}[T]$ *and* $P \longrightarrow P'$ *then* $P' \in \mathcal{L}^{\Diamond}[T]$.

Proof. By induction on the structure of type T. In all cases, we must show that: (i) $P'\Diamond$; (ii) P' is well-typed; and (iii) $P' \in \mathcal{L}^{\Diamond}[T]$, as in Def. 5.7. First, by Proposition 5.11 (1) we have that since $P\Diamond$ and $P \longrightarrow P'$, then $P'\Diamond$ as well. Well-typedness follows from subject reduction (Theorem 3.1). Finally, $P' \in \mathcal{L}^{\Diamond}[T]$ follows by definition of weak transition. □

Proposition 5.18 (Backward Closure)**.** *If for all* P_i *such that* $P \longrightarrow P_i$ *we have* $P_i \in \mathcal{L}^{\Diamond}[T]$ *then* $P \in \mathcal{L}^{\Diamond}[T]$.

Proof. By induction on the structure of type T. In all cases, we must show that: (i) $P\Diamond$ (ii) P is well-typed; and (iii) $P \in \mathcal{L}^{\Diamond}[T]$, as in Def. 5.7. Items (i) and (ii) follow by Proposition 5.11 (2) and subject reduction (Theorem 3.1), respectively. Item (iii) follows by definition of weak transition, noticing that if $P' \stackrel{\alpha}{\Longrightarrow} P''$ and $P \longrightarrow P'$ then clearly $P \stackrel{\alpha}{\Longrightarrow} P''$ holds. □

Second Step: Well-typed Processes are in the Logical Predicate. We now prove that well-typed processes are in the logical predicate.

Lemma 5.4. *Let P be a process. If $\Gamma;\Delta \vdash P :: T$ then $P \in \mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T]$.*

Proof. By induction on the derivation of $\Gamma;\Delta \vdash P :: T$, with a case analysis on the last typing rule used. See Appendix A.5 (Page 55) for details. □

We now state the desired result: well-typed processes are confluent.

Theorem 5.2 (Well-typed Processes are Confluent)**.** *If $\Gamma;\Delta \vdash P :: T$ then $P\Diamond$.*

Proof. Follows from previously proven facts. By assumption, we have $\Gamma;\Delta \vdash P :: T$. Using this and Lemma 5.4 we get $P \in \mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T]$. Pick any $G \in \mathcal{C}^{\Diamond}_{\Gamma}, D \in \mathcal{C}^{\Diamond}_{\Delta}$: combining $P \in \mathcal{L}^{\Diamond}[\Gamma;\Delta \vdash T]$ and Lemma 5.3 gives us $(\nu\widetilde{u},\widetilde{x})(P \mid G \mid D) \in \mathcal{L}^{\Diamond}[T]$. By using this, together with Definition 5.7, we infer $(\nu\widetilde{u},\widetilde{x})(P \mid G \mid D)\Diamond$. Since Proposition 5.13 ensures $G\Diamond$ and $D\Diamond$, this last result implies $P\Diamond$. □

6. Observational Equivalences for Session-Typed Processes

In this section, we investigate the behavioral theory for session-typed processes. We introduce *typed context bisimilarity* (noted $\approx$), a labelled bisimulation which closely follows the nature of typing judgments.

6.1. Auxiliary Definitions

We sometimes write $\vdash P :: T$ to stand for $\cdot;\cdot \vdash P :: T$ and $\Gamma;\Delta \vdash P, Q :: T$ to mean that both $\Gamma;\Delta \vdash P :: T$ and $\Gamma;\Delta \vdash Q :: T$ hold. Below, we use $\mathcal{S}$ to range over sequents of the form $\Gamma;\Delta \vdash T$. We will rely on *type-respecting* relations, which are indexed by sequents $\mathcal{S}$. We will use binary relations, so the adjective "binary" will be always omitted.

Definition 6.1 (Type-respecting relations)**.** *A* type-respecting 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, and use notation $\Gamma;\Delta \vdash P\,\mathcal{R}\,Q :: T$ to mean both (i) $\Gamma;\Delta \vdash P, Q :: T$ and (ii) $(P, Q) \in \mathcal{R}_{\Gamma;\Delta\vdash T}$.*

We use $\mathcal{R}, \mathcal{R}', \ldots$ to range over type-respecting relations. In the following, we will often omit the adjective "type-respecting".

Definition 6.2. *A relation $\mathcal{R}$ is said to be*

- Reflexive*, if $\Gamma;\Delta \vdash P :: T$ implies $\Gamma;\Delta \vdash P\,\mathcal{R}\,P :: T$;*
- Symmetric*, if $\Gamma;\Delta \vdash P\,\mathcal{R}\,Q :: T$ implies $\Gamma;\Delta \vdash Q\,\mathcal{R}\,P :: T$;*
- Transitive*, $\Gamma;\Delta \vdash P\,\mathcal{R}\,P' :: T$ and $\Gamma;\Delta \vdash P'\,\mathcal{R}\,Q :: T$ imply $\Gamma;\Delta \vdash P\,\mathcal{R}\,Q :: T$.*

Moreover, $\mathcal{R}$ is said to be an equivalence *if it is reflexive, symmetric, and transitive.*

In order to define *contextual* relations, we introduce a natural notion of *(typed) process contexts*. Intuitively, a context is a process that contains one hole, noted $\bullet$. Holes are *typed*: a hole, denoted $\bullet_{\Gamma;\Delta\vdash T}$, can only be filled in with a process matching its type. We shall use $K, K', \ldots$ for ranging over properly defined contexts, in the sense given next. We rely on left- and right-hand side typings for defining contexts and its properties precisely. We consider contexts with exactly one hole, but our definitions are easy to generalize.

We rely on a minimal extension of the syntax of processes (Definition 2.1) with $\bullet$. We then extend sequents, in the following way:

$$\mathcal{H}; \Gamma; \Delta \vdash K :: S$$

Intuitively, $\mathcal{H}$ contains a description of a hole occurring in (context) K: we have that $\bullet_{\Gamma;\Delta\vdash T}; \Gamma; \Delta' \vdash K :: S$ is the type of a context K whose hole is to be substituted by some process P such that $\Gamma; \Delta \vdash P :: T$. As a result of the substitution, we obtain a process $\Gamma; \Delta' \vdash K[P] :: S$. Since we consider at most one hole, $\mathcal{H}$ is either empty or has exactly one element. If $\mathcal{H}$ is empty then K is a process and we obtain the usual typing rules; we write $\Gamma; \Delta \vdash R :: T$ rather than $\cdot; \Gamma; \Delta \vdash R :: T$. The definition of typed contexts is completed by extending the type system with the following two rules:

$$\text{(Thole)}\ \frac{}{\bullet_{\Gamma;\Delta\vdash T}; \Gamma; \Delta \vdash \bullet :: T} \qquad \text{(Tfill)}\ \frac{\Gamma; \Delta \vdash R :: T \quad \bullet_{\Gamma;\Delta\vdash T}; \Gamma; \Delta' \vdash K :: S}{\Gamma; \Delta' \vdash K[R] :: S}$$

Axiom (Thole) allows to introduce holes into typed contexts. In rule (Tfill), R is a process (it does not have any holes), and K is a context with a hole of type $\Gamma; \Delta \vdash T$. The substitution of occurrences of $\bullet$ in K with R, noted $K[R]$ is sound as long as the typings of R coincide with those declared in $\mathcal{H}$ for K.

As an example, consider a simple parallel context, $(\nu x)(\bullet \mid P)$ which is filled in with an appropriately typed process R:

$$\dfrac{\Gamma; x{:}C, \Delta_2 \vdash R :: T \qquad \dfrac{\dfrac{\vdots}{\Gamma; \Delta_1 \vdash P :: x{:}C} \quad \dfrac{}{\bullet_{\Gamma;x:C,\Delta_2\vdash T}; \Gamma; x{:}C, \Delta_2 \vdash \bullet :: T}\ \text{(Thole)}}{\bullet_{\Gamma;x:C,\Delta_2\vdash T}; \Gamma; \Delta_1, \Delta_2 \vdash (\nu x)(P \mid \bullet) :: T}\ \text{(Tcut)}}{\Gamma; \Delta_1, \Delta_2 \vdash (\nu x)(P \mid R) :: T}\ \text{(Tfill)}$$

As we have seen, contexts in our setting are hardly arbitrary: only type-compatible processes are inserted into holes. Based on this observation, and following the typing rules, we define a notion of contextual relation in our typed setting:

Definition 6.3 (Contextual Relation)**.** *A relation $\mathcal{R}$ is* contextual *if it satisfies the conditions in Figure 8 (Page 69).*

Some comments to the conditions associated to Definition 6.3 are in order. In all cases, observe how the typing rules guide the shape of allowed contexts. For instance, item (1) is easily seen to correspond to rule (T$\multimap$R) and associated to the input context

$$\bullet_{\Gamma;\Delta,y:A\vdash x:B}; \Gamma; \Delta \vdash x(y).\bullet :: x{:}A \multimap B$$

to be filled in by any P such that $\Gamma; \Delta, y{:}A \vdash P :: x{:}B$. In fact, premises of each rule suggest where to place holes; rules with two premises lead to two different contexts. Observe how item (0) involves the forwarding construct; this could be seen as a form of closure under substitution, which renames the right-hand side typing of a process. Items (8)-(13) correspond to closure with respect to parallel contexts, which in our typed setting also involves closure with respect to restriction, following rules (Tcut) and (Tcut$^!$). Notice that while closure under arbitrary process composition is not allowed, closure under *independent* parallel composition (i.e., the parallel composition of any typed process P with any process Q offering $-{:}\mathbf{1}$) is permitted (cf. Items (12) and (13)). This is justified by the derived typing rule (comp) (cf. [10]).

Remark 6.1. *Notice that not all the contextuality conditions in Figure 8 apply in the case $\mathcal{R}$ relates processes related under empty left-hand side typing environments. Indeed, only items (0), (2)-(8), (10)-(13), and (15) apply in that case.*

6.2. Typed Context Bisimilarity

We define *typed context bisimilarity*, a labeled bisimilarity for 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) typing, 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 required behaviors (as described in the left-hand side typing).

Definition 6.4 (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* $Q \overset{x(y)}{\Longrightarrow} Q'$ *and for all* R *such that* $\vdash R :: y{:}A$*,*
$\vdash (\nu y)(R \mid P')\,\mathcal{R}\,(\nu y)(R \mid Q') :: x{:}B$*.*

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

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

Choice $\vdash P \mathcal{R} Q :: x{:}A \,\&\, B$ *implies* <u>*both*</u>*:*

- *If* $P \xrightarrow{x.\mathsf{inl}} P'$ *then* $\vdash P' \mathcal{R} Q' :: x{:}A$*, for some* Q' *such that* $Q \stackrel{x.\mathsf{inl}}{\Longrightarrow} Q'$*;* <u>*and*</u>
- *If* $P \xrightarrow{x.\mathsf{inr}} P'$ *then* $\vdash P' \mathcal{R} Q' :: x{:}B$*, for some* Q' *such that* $Q \stackrel{x.\mathsf{inr}}{\Longrightarrow} Q'$*.*

Selection $\vdash P \mathcal{R} Q :: x{:}A \oplus B$ *implies* <u>*both*</u>*:*

- *If* $P \xrightarrow{\overline{x.\mathsf{inl}}} P'$ *then* $\vdash P' \mathcal{R} Q' :: x{:}A$ *for some* Q' *such that* $Q \stackrel{\overline{x.\mathsf{inl}}}{\Longrightarrow} Q'$*;* <u>*and*</u>
- *If* $P \xrightarrow{\overline{x.\mathsf{inr}}} P'$ *then* $\vdash P' \mathcal{R} Q' :: x{:}B$ *for some* Q' *such that* $Q \stackrel{\overline{x.\mathsf{inr}}}{\Longrightarrow} Q'$*.*

<u>*Inductive Cases*</u>

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

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.

In all cases, a strong action is matched with a weak transition. In proofs, we shall exploit the fact that Theorem 5.1 ensures that weak transitions are 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/Tcut$^!$) rules in the type system.

6.3. Properties of Typed Context Bisimilarity

We establish some properties of typed context bisimilarity: equivalence (Proposition 6.1); closure under independent parallel composition (Proposition 6.2); a simplification for the bisimulation proof technique (Proposition 6.3); contextuality/congruence (Lemma 6.1); and τ-intertness (Lemma 6.2).

Proposition 6.1. $\approx$ *is an equivalence, in the sense of Definition 6.2.*

Proof. Reflexivity and symmetry are immediate from the definition of type-respecting relations. Transitivity is easy by showing a suitable typed context bisimulation. □

Intuitively, independent parallel composition refers to the composition of some given process with another processes typed with $-{:}\mathbf{1}$.

Proposition 6.2 (Closure under independent composition)**.** *Let P, Q, S be processes such that $\Gamma; \Delta \vdash P \approx Q :: T$ and $\Gamma; \Delta' \vdash S :: -:\mathbf{1}$ hold. Then we have: $\Gamma; \Delta, \Delta' \vdash P \mid S \approx Q \mid S :: T$*

Proof. Straightforward by showing the appropriate bisimulation, using the fact that composition with arbitrary processes offering type $\mathbf{1}$ is type preserving, and by noticing that S cannot interact with P, Q. □

Definition 6.4 immediately suggests a proof technique for showing that two processes are typed context bisimilar. First, close the processes with *parallel representatives* of their context, applying repeatedly the inductive cases 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. More precisely, given a left-hand side typing $\Gamma; \Delta$, below we define the set $\mathcal{K}_{\Gamma;\Delta\vdash T}$ of parallel representatives of Γ, Δ. This is a set of parallel process contexts which represent the closures generated by the inductive case of typed context bisimilarity. These parallel representatives will be useful to simplify proofs for $\approx$.

Definition 6.5 (Parallel Representatives)**.** *Let Γ and Δ be typing environments defined as $\Gamma = u_1{:}B_1, \ldots, u_n{:}B_n$ and $\Delta = x_1{:}A_1, \ldots, x_m{:}A_m$, respectively, with $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, m\}$. We say that K is a* parallel representative *in $\mathcal{K}_{\Gamma;\Delta\vdash T}$ if*

$$K \equiv (\nu \widetilde{u}, \widetilde{x})(\bullet \mid \prod_{i \in I} !u_i(y_i).R_i \mid \prod_{j \in J} S_j)$$

with $\vdash R_i :: y_i{:}B_i$ and $\vdash S_j :: x_j{:}A_j$, for every $i \in I$ and $j \in J$.

Clearly, for every left-hand side typing there may be many parallel representatives, corresponding to different implementations of the required behaviors. It is easy to see that parallel representatives are well-typed: if $K \in \mathcal{K}_{\Gamma;\Delta\vdash T}$ then $\bullet_{\Gamma;\Delta\vdash T}; \cdot; \cdot \vdash K :: T$. In fact, filling in a context $K \in \mathcal{K}_{\Gamma;\Delta\vdash T}$ with a process $\Gamma, \Delta \vdash P :: T$ will lead to process $\vdash K[P] :: T$, which requires nothing from its environment. This is the essence of the desired simplification, formalized by the following proposition. It allows us to convert an (inductive) proof under non-empty typing environments Γ, Δ into a (coinductive) proof under empty environments, with processes enclosed within parallel contexts.

Proposition 6.3. *$\Gamma; \Delta \vdash P \approx Q :: T$ implies $\vdash K[P] \approx K[Q] :: T$, where K is any parallel representative in $\mathcal{K}_{\Gamma;\Delta\vdash T}$, as in Definition 6.5.*

Proof. See Appendix B.1 (Page 57) for details. □

Based on the logical interpretation, we introduce a notion of "continuation relation" for pairs of typed processes. This will be useful to define and reason about type-respecting relations. Below, $\mathcal{I}_{\Gamma;\Delta\vdash T}$ stands for the relation

$$\{(P, Q) \ : \ \Gamma; \Delta \vdash P, Q :: T \}$$

which collects pairs of processes with identical left- and right-hand side typings.

Definition 6.6. *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:*

$$\mathcal{W}_{\vdash x:\mathbf{1}} = \mathcal{I}_{\vdash x:\mathbf{1}} \qquad \mathcal{W}_{\vdash x:A\boxtimes B} = \mathcal{I}_{\vdash x:B} \cup \mathcal{W}_{\vdash x:B}$$
$$\mathcal{W}_{\vdash x:!A} = \mathcal{I}_{\vdash x:!A} \qquad \mathcal{W}_{\vdash x:A\boxplus B} = \mathcal{I}_{\vdash x:A} \cup \mathcal{W}_{\vdash x:A} \cup \mathcal{I}_{\vdash x:B} \cup \mathcal{W}_{\vdash x:B}$$

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 $x{:}B$.

We now prove that $\approx$ is a contextual relation. That is, $\approx$ is a congruence with respect to the typed contexts associated to Definition 6.3.

Lemma 6.1 (Contextuality of $\approx$)**.** *Typed context bisimilarity is a contextual relation, in the sense of Definition 6.3.*

Proof. The proof proceeds by coinduction, showing a typed context bisimulation for each of the conditions associated to Def. 6.3. We shall exploit the proof technique given by Prop. 6.3, which allows to consider $\approx$ under empty left-hand side contexts, for pairs of processes enclosed within appropriate parallel representatives. As a result, it suffices to consider only some of the conditions in Table 8; see Remark 6.1. Most cases are easy; below we detail one of them: closure with respect to output, Item (2). (See Appendix B.2, Page 57 for other cases).

We have to show that $\Gamma; \Delta \vdash P \approx Q :: y{:}A$ implies

$$\Gamma; \Delta, \Delta' \vdash \overline{x}\langle y\rangle.(P \mid S) \approx \overline{x}\langle y\rangle.(Q \mid S) :: x{:}A \otimes B$$

for any S, x, B, Δ' such that $\Gamma; \Delta' \vdash S :: x{:}B$. Using Proposition 6.3, this can be simplified, and it suffices to show that $\vdash K_1[P] \approx K_1[Q] :: y{:}A$ implies

$$\vdash K_2[\overline{x}\langle y\rangle.(K_1[P] \mid S)] \approx K_2[\overline{x}\langle y\rangle.(K_1[Q] \mid S)] :: x{:}A \otimes B$$

where $K_1 \in \mathcal{K}_{\Gamma;\Delta\vdash y:A}$ and $K_2 \in \mathcal{K}_{\cdot;\Delta'\vdash x:A\otimes B}$.

Let $M = K_2[\overline{x}\langle y\rangle.(K_1[P] \mid S)]$ and $N = K_2[\overline{x}\langle y\rangle.(K_1[Q] \mid S)]$. Define

$$\begin{array}{rcl} \mathcal{R}_2 & = & \{(M,N) : \vdash K_1[P] \approx K_1[Q] :: y{:}A,\ K_1 \in \mathcal{K}_{\Gamma;\Delta\vdash y:A},\ K_2 \in \mathcal{K}_{\cdot;\Delta'\vdash x:A\otimes B}\} \\ & \cup & \mathcal{W}_{\vdash x:B} \end{array}$$

We show that $\mathcal{R}_2$ is a typed context bisimulation. Suppose M moves first: $M \xrightarrow{\alpha} M'$. We must find a matching action from N such that $N \stackrel{\alpha}{\Longrightarrow} N'$. There are two possibilities for α: either $\alpha = \tau$ or $\alpha = \overline{x\langle y\rangle}$. In the first case, we have $M \xrightarrow{\tau} K_3[\overline{x}\langle y\rangle.(K_1[P] \mid S)] = M'$, where $K_2 \xrightarrow{\tau} K_3$. Since K_2 occurs identically in N by construction, this action can be matched and we have $N \Longrightarrow K_4[\overline{x}\langle y\rangle.(K_1[Q] \mid S)] = N'$, where $K_2 \Longrightarrow K_4$. Subject reduction (Theorem 3.1) ensures both $K_3 \in \mathcal{K}_{\Gamma;\Delta\vdash y:A}$ and $K_4 \in \mathcal{K}_{\cdot;\Delta'\vdash x:A\otimes B}$, and so $(M', N') \in \mathcal{R}_2$.

In the second case we $M \xrightarrow{\overline{x\langle y\rangle}} K_2[K_1[P] \mid S] = M'$. Process N can match this action, followed by zero or more reductions: $N \stackrel{\overline{x\langle y\rangle}}{\Longrightarrow} K_4[K_3[Q'] \mid S'] = N'$,

where $K_2 \Longrightarrow K_4$, $K_1 \Longrightarrow K_3$, $Q \Longrightarrow Q'$, and $S \Longrightarrow S'$. (Recall that K_1 and K_2 are parallel contexts, and so they are able to interact.) Theorem 5.1 ensures that these reductions are finite. Since $\vdash K_1[P] \approx K_1[Q] :: y{:}A$, and because of τ-closedness, we have $\vdash K_1[P] \approx K_3[Q'] :: y{:}A$. Subject reduction (Theorem 3.1) ensures $\vdash S, S' :: x{:}B$. Following the output clause of $\approx$, we consider the closure of M' and N' with a process L such that $y{:}A \vdash L :: -{:}\mathbf{1}$. Such closures correspond to $K_2[(\nu y)(K_1[P] \mid L) \mid S]$ and $K_4[(\nu y)(K_3[Q'] \mid L) \mid S']$, respectively. We verify that the type of these closures is indeed $x{:}B$, as required by the output clause. Since $\vdash K_1[P], K_3[Q'] :: y{:}A$, these processes can be composed with L, and we obtain

$$\vdash (\nu y)(K_1[P] \mid L), (\nu y)(K_3[Q'] \mid L) :: -{:}\mathbf{1}$$

The desired pair of processes can be obtained via an independent parallel composition with S, K_2, S', and K_4, respectively:

$$\vdash K_2[(\nu y)(K_1[P] \mid L) \mid S], K_4[(\nu y)(K_3[Q'] \mid L) \mid S'] :: x{:}B$$

Hence, $\big(K_2[(\nu y)(K_1[P] \mid L) \mid S],\ K_4[(\nu y)(K_3[Q'] \mid L) \mid S')]\big) \in \mathcal{R}_2$ and we are done. The reasoning when N moves first is completely symmetric. □

We now state *τ-inertness*, a property of transition systems which follows as a direct consequence of the results of our framework, in particular, confluence (Theorem 5.2) and the definition of typed context bisimilarity. Following Groote and Sellink [19], this property may be stated in a general way:

Definition 6.7 (τ-inertness)**.** *Let $(S, \longrightarrow)$ be a transition system, where S is a set of states and $\longrightarrow \subseteq S \times S$. Also, let $\sim$ stand for an equivalence relation on the elements of S. We say that $(S, \longrightarrow)$ is* τ-inert with respect to $\sim$ *if $P \longrightarrow P'$ implies $P \sim P'$.*

τ-inertness is typically defined for labeled transition systems with a designated internal action τ, hence its name. In our case, since the LTS and the reduction relation coincide, we can safely work with reductions, and show that the class of well-typed processes is τ-inert with respect to $\approx$. Intuitively, τ-inertness says that reduction does not change the behavior of a process. It is therefore a property relevant for verification, as it ensures that well-typed processes can perform arbitrarily many reductions remaining in the same equivalence class; this is strengthened by the fact that termination (Theorem 5.1) ensures that these reductions are only finitely many. Adapting Definition 6.7 to our setting, we have:

Lemma 6.2 (τ-inertness wrt $\approx$)**.** *Let P be a process such that $\Gamma; \Delta \vdash P :: T$. Suppose $P \longrightarrow P'$. Then $\Gamma; \Delta \vdash P \approx P' :: T$.*

Proof. By coinduction, exhibiting an appropriate typed context bisimulation. Using Prop. 6.3, we work under an empty left-hand side typing. We thus define a type-respecting relation containing $(K[P], K[P'])$, for any $K \in \mathcal{K}_{\Gamma;\Delta \vdash T}$ (letting Id to stand for the identity relation):

$$\mathcal{R} = \{(K[P], K[P'])\ :\ P \longrightarrow P'\,,\ K \in \mathcal{K}_{\Gamma;\Delta \vdash T}\} \cup Id \cup \mathcal{W}_{\vdash T}$$

Notice that by assumption, $\vdash K[P] :: T$; by subject reduction (Theorem 3.1) $\vdash K[P'] :: T$. We show that $\mathcal{R}$ is a typed context bisimilarity. Suppose $K[P]$ moves first, i.e., $K[P] \xrightarrow{\alpha} M$, for some α, M. We must show a matching action $K[P'] \stackrel{\alpha}{\Longrightarrow} N$. We distinguish two cases, when $\alpha \neq \tau$ and when $\alpha = \tau$:

- If $\alpha \neq \tau$ then, necessarily, the action is related to the type T. Appropriate inversion lemmas (Lemma 3.2) can be used to determine the actual label of α. Now, we know that $\vdash K[P], K[P'] :: T$ and that the only difference between $K[P]$ and $K[P']$ is an internal action; since $\alpha \neq \tau$, these conditions ensure that $K[P']$ can match the action α and that there exists an N such that $K[P'] \stackrel{\alpha}{\Longrightarrow} K'[P'']$, where $K \Longrightarrow K'$. The analysis concludes by a case analysis on the shape of T; depending of T, the definition of $\approx$ determines the actual shape of the derivatives that should be found in $\mathcal{R}$. All cases are easy (output, input, and replicated input require suitable process closures) and covered by the definition of $\mathcal{W}_{\vdash T}$, which ensures that $(M, N) \in \mathcal{W}_{\vdash T}$.
- If $\alpha = \tau$ then there are two subcases: $M \equiv K[P']$ (i.e., α is the same τ action that leads from $K[P]$ to $K[P']$) and $M \not\equiv K[P']$ (i.e., α corresponds to a different τ action from $K[P]$). In the first subcase, $K[P']$ can trivially match this reduction with zero reductions, i.e., $K[P'] \Longrightarrow K[P'] = N$. Since the pair $(K[P'], K[P'])$ is in $\mathcal{R}$ we are done. In the second subcase, $K[P']$ is able to match this τ action because of confluence (Theorem 5.2). Call τ_1 the τ action from $K[P]$ to $K[P']$, and let α be τ_2. That is, $K[P]$ can exercise both τ_1 and τ_2. Confluence ensures that if $K[P]$ performs τ_1 first, then its derivative $K[P']$ can still exercise τ_2—this internal action is not discarded. Therefore, if $K[P]$ challenges $K[P']$ with τ_2, confluence ensures that $K[P']$ can perform τ_2, possibly preceded and followed by other internal actions. A matching action $K[P'] \Longrightarrow N$, in which the weak transition contains τ_2, thus exists, and it is easy to see that $(M, N) \in \mathcal{R}$, and we are done.

Now suppose that $K[P']$ moves first, i.e., that $K[P'] \xrightarrow{\alpha} N$. We must show a matching action $K[P] \stackrel{\alpha}{\Longrightarrow} M$. Since $K[P']$ is a τ-derivative of $K[P]$, it is easy to show that $K[P]$ can always match any action from $K[P']$: $K[P] \longrightarrow K[P'] \xrightarrow{\alpha} N$, for any α, N. This can be rewritten as $K[P] \stackrel{\alpha}{\Longrightarrow} N$ and we are done. □

7. Applications

In this section, we first establish the soundness of proof conversions with respect to typed context bisimilarity, and then introduce a behavioral characterization of type isomorphisms. Besides clarifying further the intrinsic properties of the logical interpretation of session types, these applications illustrate the interplay of typed context bisimilarity and the properties of the type system (subject reduction, progress, termination, confluence).

7.1. Soundness of Proof Conversions

Recall that, by Definition 4.1, $\simeq_c$ stands for the congruence on typed processes induced by *proof conversions*. We now show *soundness* of $\simeq_c$ with respect to $\approx$, that

is, we show that processes extracted from proof conversions are typed contextually bisimilar.

Before formally stating and proving this claim, we provide some intuitions on it. Consider the fifth process equality in Figure 3 (Page 12). It corresponds to the interplay of rules (Tcut) and (T$\oplus$L), under typing assumptions $\Gamma;\Delta_1 \vdash \hat{D} :: x{:}C$, $\Gamma;\Delta_2, y{:}A, x{:}C \vdash \hat{E}{::}T$, and $\Gamma;\Delta_2, y{:}A, x{:}C \vdash \hat{F}{::}T$. Letting $\Delta = \Delta_1, \Delta_2$, we have:

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

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

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 the behaviors declared 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 7.1 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). Recall that $\mathcal{I}_{\Gamma;\Delta\vdash T}$ stands for the relation which collects pairs of processes with identical left- and right-hand side typings. Also, we use the continuation relations $\mathcal{W}_{\vdash x:A}$ (cf. Definition 6.6).

Theorem 7.1 (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 5.1) to ensure that actions can be matched with finite weak transitions, and subject reduction (Theorem 3.1) to ensure type preservation under reductions.

We detail the case for the first proof conversion in Figure 4 —see Appendix C.1 (Page 61) for other cases. This proof conversion 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

$$\Delta = \Delta_1, \Delta_2, \Delta_3 \quad \Gamma;\Delta_1 \vdash \hat{D} :: x{:}C \quad \Gamma;\Delta_2, x{:}C \vdash \hat{E} :: y{:}A \quad \Gamma;\Delta_3 \vdash \hat{F} :: z{:}B \qquad (4)$$
$$M = (\nu x)(\hat{D} \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F})) \quad N = \overline{z}\langle y\rangle.((\nu x)(\hat{D} \mid \hat{E}) \mid \hat{F})$$

Using Proposition 6.3, 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 bisimulation $\mathcal{R}$ containing the pair $(K[M], K[N])$. We define $\mathcal{R} = \mathcal{W}_{\vdash z:A\otimes B} \cup \mathcal{S} \cup \mathcal{S}^{-1}$, with

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

and $\mathcal{W}_{\vdash z:A\otimes B}$ is as in Definition 6.6. Notice that $\mathcal{S}$ is a type-respecting relation indexed by $\vdash z{:}A \otimes B$. In fact, using the typings in (4)—with $\Gamma = \Delta = \emptyset$—and exploiting subject reduction (Theorem 3.1), 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 bisimulation. Pick any $K \in \mathcal{K}_{\Gamma;\Delta}$. Using Definition 6.5, we can assume $K = (\nu \widetilde{u}, \widetilde{x})(K_\Gamma \mid K_\Delta \mid [\cdot])$ 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: $K[M] \xrightarrow{\alpha} M_1^\star$. We have to find a matching action α from $K[N]$, i.e., $K[N] \stackrel{\alpha}{\Longrightarrow} N_1^\star$. 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 $l \in J$, $S_l \xrightarrow{\tau} S'_l$:

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

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 (Theorem 5.1):

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

By subject reduction (Theorem 3.1), $\vdash S'_l :: x_l{:}C_l$; 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 $M = (\nu x)(\hat{D} \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F}))$, the only possibility is that there is a $\hat{D}_1$ such that $\hat{D} \xrightarrow{\tau} \hat{D}_1$ and $M' = (\nu x)(\hat{D}_1 \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F}))$. This way,

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

We observe that $K[N]$ cannot match this action, but $K[N] \Longrightarrow K[N]$ is a valid weak transition. Hence, $N_1^\star = K[N]$. By subject reduction (Theorem 3.1), 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 $\widetilde{u}, \widetilde{x}$. 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 $\alpha = \overline{z\langle y\rangle}$. This action can only originate in M:

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

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

$$K[N] \stackrel{\overline{z\langle y\rangle}}{\Longrightarrow} (\nu\widetilde{x},\widetilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D}' \mid \hat{E}') \mid \hat{F}')) = N_1^\star$$

Observe how $N_1^\star$ 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 $M_1^\star$ and $N_1^\star$ with any V such that $y{:}A \vdash V :: -{:}\mathbf{1}$. Using the typings in (4) and subject reduction (Theorem 3.1), we infer both

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

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

Now suppose that $K[N]$ moves first: $K[N] \xrightarrow{\alpha} N_1^\star$. We have to find a matching action α from $K[M]$: $K[M] \stackrel{\alpha}{\Longrightarrow} M_1^\star$. Similarly as before, there are two cases: either $\alpha = \tau$ or $\alpha = \overline{z\langle y\rangle}$. 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. □

7.2. A Behavioral Characterization of Type Isomorphisms

In type theory, types A and B are called *isomorphic* if there are morphisms π_A of $B \vdash A$ and π_B of $A \vdash B$ which compose to the identity in both ways—see, e.g., [17]. We adapt this notion to our setting, by using proofs as morphisms, and by using typed context bisimilarity to account for *isomorphisms* in linear logic. (Below, we write $P^{\langle\widetilde{x}\rangle}$ for a process parametric on a sequence of names $x_1, \ldots, x_n$.)

Definition 7.1 (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. As an example, let us consider 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 7.2 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 $B \otimes A$.

Theorem 7.2. *Let A, B, and C be any type, as in Def 3.1. Then the following hold:*

(i)	$A \otimes B$	$\simeq$	$B \otimes A$
(ii)	$(A \oplus B) \multimap C$	$\simeq$	$(A \multimap C) \,\&\, (B \multimap C)$
(iii)	$!(A \,\&\, B)$	$\simeq$	$!A \otimes !B$

Proof. We give details for the proof of *(i)* above; see Appendix C.2, Page 67, for further details.
We check conditions (i)-(iv) of Def. 7.1 for processes $P^{\langle x,y\rangle}, Q^{\langle y,x\rangle}$ defined as

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

Checking (i)-(ii), i.e., $\cdot\,; x{:}A \otimes B \vdash P^{\langle x,y\rangle} :: y{:}B \otimes A$ and $\cdot\,; y{:}B \otimes A \vdash Q^{\langle y,x\rangle} :: x{:}A \otimes B$ is easy; rule (Tid) ensures that both typings hold for any A, B. We sketch only the proof of (iii); the proof of (iv) is analogous. Let $M = (\nu y)(P^{\langle x,y\rangle} \mid Q^{\langle y,z\rangle})$ and $N = [x \leftrightarrow z]$; we need to show $\cdot\,; x{:}A \otimes B \vdash M \approx N :: z{:}A \otimes B$. By Proposition 6.3, 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 bisimulation $\mathcal{R}$ 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 bisimulation. □

8. Related Work

Logical Relations in Concurrency. In a concurrent/process calculi setting, logical relations (or closely related techniques) have been investigated by Berger, Honda, and Yoshida [45, 3, 4], Sangiorgi [40], Caires [8], and Boudol [7]. None of these works considers session types, and so the logical relations proposed in such works are very different from ours. Boudol [7] relies on the classical realizability technique (together with a type and effect system) to establish termination in a higher-order imperative language. Caires [8] proposes a semantic approach to proving soundness for type systems for concurrency, by relying on a spatial logic interpretation of types. More related to our developments are works by Yoshida, Berger, Honda [45] and by Sangiorgi [40], which aim at identifying terminating fragments of the π-calculus by using types, relying on arguments based on logical relations. The logical relations framework developed in [45] is extended in [3, 4] to the case of a second-order, polymorphic π-calculus. A

main result in [3, 4] is a proof of termination using the method of reducibility candidates; while [3] reports a relational parametricity result, [4] puts forward a behavioral theory based on generic transitions and a fully abstract embedding of System F. All of these works consider typing disciplines different from session types; consequently, associated semantic interpretations of types are very different from ours, and rely on constraints on the syntax and the types of processes. In sharp contrast to [45, 40], which aim at type disciplines that guarantee termination, here we started from a well-established type discipline for the π-calculus and have used linear logical relations to show termination and confluence of well-typed processes. We have shown how the interpretation of intuitionistic linear logic as session types in [10] leads to intuitive logical relations, naturally defined on the structure of types. In this sense, our approach is more principled than in [45, 40], as it is not an adaptation of the method, but rather an instantiation of the method on our canonical linear type structure.

Logical Interpretations of Session Types. Dal Lago and Giamberardino [13] introduce an interpretation of session types as *soft* linear logic propositions [27]. As a result, the exponential "!" is treated following a non canonical discipline that uses two different typing environments. Hence, typing rules and judgements in [13] are rather different from ours. A bound on the length of reductions starting from well-typed-processes is obtained; the proof uses techniques from Implicit Computational Complexity. Neither confluence, observational equivalences, nor issues of inference permutability and type isomorphisms are addressed in [13]. Although here we do not provide a similar bound, it is remarkable that our proof of termination follows *only* the principles and properties of [10]; in contrast to [13], our proof appeals to well-known technical devices, and allows us to retain a standard, intuitive treatment of "!". This is particularly desirable for extensions/generalizations of our logical interpretation of session types, such as the proposed in [44, 34].

Loosely related is Mazurak and Zdancewic's Lolliproc [28], a functional language with support for concurrency based on control operators. Lolliproc's operational semantics is based on a runtime process calculus; thread communication is defined in terms of protocol types which are given a *classic* linear logic interpretation. As in our case, type soundness, strong normalization, and confluence results hold for Lolliproc; however, the details of the associated proof techniques are rather different from ours.

Determinacy and Confluence in Process Calculi. In term rewriting systems such as the λ-calculus, determinacy and confluence are well-understood issues, and typically rely on (unlabeled) reduction semantics. For process calculi, a semantics given in terms of labelled transition systems, is often useful for it describes the interaction of processes with their environment. As a result, notions of determinacy and confluence for process calculi typically account for those labels, thus setting a major difference with respect to traditional notions. It is worth noticing that our notion of confluence (Definition 5.6) considers only weak transitions based on internal behavior, and so it is closer to classical definitions of confluence rather than to the definitions used in process calculi. Early studies of determinacy and confluence for process calculi are due to Milner, in the setting of CCS [29]; his interest was on proper definitions of such notions, focusing on syntactic conditions on process constructs so as to build determinate, confluent sys-

tems by construction. There is a close relationship between determinacy, confluence, τ-inertness and the given notion of equivalence; Groote and Sellink [19] provide a general study on such a relationship, focusing on the impact of such notions on process verification. Milner's approach to confluence was extended to the π-calculus by Walker and Philippou [35], and by Nestmann [31] who characterizes (forms of) confluence in terms of so-called *port uniqueness* for polarized name-passing, which is ensured by static typing. Most related to our work is [26], which adapts Walker and Philippou's techniques to establish session determinacy and confluence for a session-typed *asynchronous* π-calculus. The above mentioned differences in the definition of determinacy and confluence prevent detailed comparisons with our confluence result, which relies on reductions and is shown using logical relations.

Typed Behavioral Equivalences. Previous works on behavioral equivalences for *typed* process calculi have considered a number of different typing disciplines. For instance, behavioral theories for calculi with linear types (e.g., [25]), input/output types (e.g., [6, 36, 15]), subtyping with name matching (e.g., [20]), and polymorphic types (e.g., [37]) have been put forward. Still, the only work on behavioral equivalences for session-typed processes we are aware of is [26]. It studies the behavioral theory of a π-calculus with *asynchronous, event-based* binary session communication. The aim is to capture the distinction between order-preserving communications (those inside already established connections) and non-order-preserving communications (those outside such connections). The behavioral theory in [26] accounts for 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, in [26] prefix commutation arises from the above-mentioned distinction, whereas commutations in our (synchronous) framework are due to causality relations captured by types. Loosely related to typed context bisimilarity is [46], where a form of *linear bisimilarity* is proposed; following a linear type structure, it treats some visible actions as internal actions, thus leading to an equivalence larger than standard bisimilarity which is a congruence.

9. Concluding Remarks

In this paper, we have introduced a theory of linear logical relations and a notion of typed behavioral equivalences for session-typed, concurrent processes. These developments extend the interpretation of linear logic propositions as session types developed by Caires and Pfenning in [10].

Our theory of linear logical relations is remarkably similar to that for functional languages; although in our setting session types are assigned to names (and not to terms), our linear logical relations are defined on the structure of types, relying both on process reductions and labeled transitions. A main application of this theory is a proof that well-typed processes are both strongly normalizing (Theorem 5.1) and confluent (Theorem 5.2). In practice, certifying termination and confluence of session-typed programs is important. We believe the extended correctness guarantees given by our results could be highly beneficial for the increasingly growing number of practical implementations (libraries, programming language extensions) based on session types foundations—see, e.g., [24, 32, 38].

We have also presented a behavioral theory for session-typed processes. We introduced *typed context bisimilarity*, a novel labeled bisimilarity over typed processes, and studied its properties. Our definition follows from the intuitive meaning of type judgments, and is stated in the style of conventional definitions for untyped processes. In addition to studying its main properties, we have illustrated this typed observational equivalence in two applications, which strengthen the properties of the logic interpretation established in [10]. On the one hand, we have shown soundness of *proof conversions* with respect to observational equivalence—an issue left open in [10] (Theorem 7.1). On the other hand, we studied *type isomorphisms* resulting from linear logic equivalences in our setting (Theorem 7.2). The basic properties of the interpretation—especially, the combination of subject reduction and termination—were of the essence in the proofs of both applications.

There are a number of intuitive similarities in the definitions used in formalizing our theory of linear logical relations and those required for developing our behavioral theory. In recent work, we have discovered a formal connection between the two topics: in [9] we have generalized the linear logic relations here developed to the case of *parametric polymorphism*. In this extended setting, existential and universal quantification over types are interpreted as a form of session type-passing; using logical relations we have characterized barbed congruence in a sound and complete way. In future work, we plan to adapt the results here presented to the case of the interpretation of session types into *classical* linear logic, as defined in [11].

Acknowledgments. This research was supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the Carnegie Mellon Portugal Program, under grants INTERFACES NGN-44 / 2009 and SFRH / BD / 33763 / 2009, and CITI.

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. Berger, K. Honda, and N. Yoshida. Genericity and the pi-calculus. In *FoSSaCS*, volume 2620 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2003.

[4] M. Berger, K. Honda, and N. Yoshida. Genericity and the pi-calculus. *Acta Inf.*, 42(2-3):83–141, 2005.

[5] M. Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theor. Comput. Sci.*, 195:205–226, March 1998.

[6] M. Boreale and D. Sangiorgi. Bisimulation in name-passing calculi without matching. In *LICS*, pages 165–175, 1998.

[7] G. Boudol. Typing termination in a higher-order concurrent imperative language. *Inf. Comput.*, 208(6):716–736, 2010.

[8] L. Caires. Logical semantics of types for concurrency. In *CALCO 2007*, volume 4624 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2007.

[9] L. Caires, J. A. Pérez, F. Pfenning, and B. Toninho. Relational parametricity for polymorphic session types. Technical report, CMU-CS-12-108, Carnegie Mellon Univ., Apr 2012.

[10] L. Caires and F. Pfenning. Session types as intuitionistic linear propositions. In *CONCUR'2010*, volume 6269 of *LNCS*, pages 222–236. Springer, 2010.

[11] L. Caires, F. Pfenning, and B. Toninho. Linear logic propositions as session types, 2012. Under Revision - `http://www.cs.cmu.edu/~fp/papers/sessions12.pdf`.

[12] 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.

[13] U. Dal Lago and P. Di Giamberardino. Soft session types. In *Proc. of 18th Workshop on Expressiveness in Concurrency – EXPRESS'11*, volume 64 of *EPTCS*, pages 59–73, 2011.

[14] R. Demangeon, D. Hirschkoff, and D. Sangiorgi. Mobile processes and termination. In *Semantics and Algebraic Specification*, volume 5700 of *LNCS*, pages 250–273. Springer, 2009.

[15] Y. Deng and D. Sangiorgi. Towards an algebraic theory of typed mobile processes. *Theor. Comput. Sci.*, 350(2-3):188–212, 2006.

[16] M. Dezani-Ciancaglini and U. de'Liguoro. Sessions and session types: An overview. In *WS-FM 2009*, volume 6194 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 2010.

[17] R. Di Cosmo. A short survey of isomorphisms of types. *Mathematical Structures in Computer Science*, 15(5):825–838, 2005.

[18] 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.

[19] J. F. Groote and M. P. A. Sellink. Confluence for process verification. *Theor. Comput. Sci.*, 170(1-2):47–81, 1996.

[20] M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Mathematical Structures in Computer Science*, 14(5):651–684, 2004.

[21] K. Honda. Types for dynamic interaction. In *CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 509–523. Springer, 1993.

[22] 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.

[23] W. A. Howard. The formulae-as-types notion of construction. Unpublished note. An annotated version appeared in: *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, 479–490, Academic Press (1980), 1969.

[24] 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.

[25] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the pi-calculus. In *POPL*, pages 358–371, 1996.

[26] 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.

[27] Y. Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.

[28] K. Mazurak and S. Zdancewic. Lolliproc: to concurrency from classical linear logic via curry-howard and control. In *ICFP'10*, pages 39–50. ACM, 2010.

[29] R. Milner. *Communication and concurrency*. Prentice Hall, 1995.

[30] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, part I/II. *Inf. Comput.*, 100(1):1–77, 1992.

[31] U. Nestmann. *On Determinacy and Nondeterminacy in Concurrent Programming*. PhD thesis, Technische Fakultat, Universitat Erlangen, 1996.

[32] N. Ng, N. Yoshida, O. Pernet, R. Hu, and Y. Kryftis. Safe parallel programming with session java. In *Proc. of COORDINATION*, volume 6721 of *LNCS*, pages 110–126. Springer, 2011.

[33] J. A. Pérez, L. Caires, F. Pfenning, and B. Toninho. Linear logical relations for session-based concurrency. In *ESOP*, volume 7211 of *Lecture Notes in Computer Science*, pages 539–558. Springer, 2012.

[34] F. Pfenning, L. Caires, and B. Toninho. Proof-carrying code in a session-typed process calculus. In *Proc. of CPP '11*, volume 7086 of *LNCS*, pages 21–36. Springer, 2011.

[35] A. Philippou and D. Walker. On confluence in the pi-calculus. In *ICALP*, volume 1256 of *Lecture Notes in Computer Science*, pages 314–324. Springer, 1997.

[36] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.

[37] B. C. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000.

[38] 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.

[39] D. Sangiorgi. pi-calculus, internal mobility, and agent-passing calculi. *Theor. Comput. Sci.*, 167(1&2):235–274, 1996.

[40] D. Sangiorgi. Termination of processes. *Mathematical Structures in Computer Science*, 16(1):1–39, 2006.

[41] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001.

[42] R. Statman. Logical relations and the typed lambda-calculus. *Information and Control*, 65(2/3):85–97, 1985.

[43] W. W. Tait. Intensional Interpretations of Functionals of Finite Type I. *J. Symbolic Logic*, 32:198–212, 1967.

[44] 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.

[45] N. Yoshida, M. Berger, and K. Honda. Strong normalisation in the pi -calculus. *Inf. Comput.*, 191(2):145–202, 2004.

[46] N. Yoshida, K. Honda, and M. Berger. Linearity and bisimulation. *J. Log. Algebr. Program.*, 72(2):207–238, 2007.

Appendix A. Proofs of Section 5 (Logical Relations)

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

Appendix A.1. Proof of Proposition 5.7

We repeat the statement in Page 19:

Proposition Appendix A.1 (5.7)**.** *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 on the definition of $P \equiv_! Q$ (Def. 5.2). Given Prop 5.5, 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 5.2 and on the operational correspondence between P and Q given by Prop 5.1.

- 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 termination, well-typedness, and operational correspondence requirements stated in Def 5.3. For the latter, we use Prop 5.1(1) and 5.1(2). We have six cases to check; we detail only some of them as the rest is similar.

  Case $P \in \mathcal{L}[z{:}\mathbf{1}]$**.** Then $P\Downarrow$ and $\cdot;\cdot \vdash P :: z{:}\mathbf{1}$ and for all P' such that $P \Longrightarrow P'$ and $P' \not\longrightarrow$ it implies that $P' \equiv_! \mathbf{0}$. First, by Prop 5.2, we have that $Q\Downarrow$. Now, since $\cdot;\cdot \vdash P :: z{:}\mathbf{1}$ it is easy to show that there exists a typing derivation for $\cdot;\cdot \vdash Q :: z{:}\mathbf{1}$. Finally, by Prop 5.1(1), we know that Q can match any reduction from P. Therefore, there exists a Q' such that $Q \Longrightarrow Q'$ and $Q' \not\longrightarrow$ and $P' \equiv_! Q'$. By transitivity of $\equiv_!$, we have that $Q' \equiv_! \mathbf{0}$ and so $Q \in \mathcal{L}[z{:}\mathbf{1}]$, 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 5.6, P has an input action on z. Moreover, by Def 5.3, for all P', y such that $P \overset{z(y)}{\Longrightarrow} P'$ it implies that $\forall R \in \mathcal{L}[y{:}A].(\nu y)(P' \mid R) \in \mathcal{L}[z{:}B]$. First, by Prop 5.2, 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 5.1(1) and 5.1(2), we know that Q can match any reduction/transition from P. Therefore, there exists a Q' such that $Q \overset{z(y)}{\Longrightarrow} Q'$ 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 5.6, P has an output action on z. Moreover, by Def 5.3, for all P', y such that $P \stackrel{(\nu y)z\langle y\rangle}{\Longrightarrow} P'$ 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 5.2, 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 5.1(1) and 5.1(2), we know that Q can match any reduction/transition from P. Therefore, there exists a Q' such that $Q \stackrel{(\nu y)z\langle y\rangle}{\Longrightarrow} Q'$ 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 $P \in \mathcal{L}[z{:}\mathbf{1}]$.

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 5.3. The analysis mirrors the one given above for Axiom (1), using Prop 5.2, 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 5.1(1) and 5.1(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

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

  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 5.3. The crucial point is to observe that since $u \notin \mathit{fn}(P_2)$ 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 $P \in \mathcal{L}[z{:}\mathbf{1}]$**.** Then $P\Downarrow$ and $\cdot;\cdot \vdash P :: z{:}\mathbf{1}$ and for all P' such that $P \Longrightarrow P'$ and $P' \not\longrightarrow$ it implies that $P' \equiv_! \mathbf{0}$. First, by Prop 5.2, we have that $Q\Downarrow$. Now, since $\cdot;\cdot \vdash P :: z{:}\mathbf{1}$ it is possible to show that $\cdot;\cdot \vdash Q :: z{:}\mathbf{1}$. Notice also that since $u \notin \mathit{fn}(P_2)$, 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 $\cdot;\cdot \vdash P :: z{:}A{\multimap}B$. Hence, by Prop 5.6, P has an input action on z. Moreover, by Def 5.3, for all P', y such that $P \stackrel{z(y)}{\Longrightarrow} P'$ it implies that $\forall R \in \mathcal{L}[y{:}A].(\nu y)(P' \mid R) \in \mathcal{L}[z{:}B]$. First, by Prop 5.2, we have that $Q\Downarrow$. Now, since $\cdot;\cdot \vdash P :: z{:}A{\multimap}B$ then it can be shown that $\cdot;\cdot \vdash Q :: z{:}A{\multimap}B$. Now, since $u \notin \mathit{fn}(P_2)$, 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.

□

Appendix A.2. Proof of Proposition 5.10

We repeat the statement in Page 20:

Proposition Appendix A.2 (5.10 – Weakening)**.** *Let P, Q be processes such that $P \in \mathcal{L}[T]$ and $Q \in \mathcal{L}[-{:}\mathbf{1}]$. 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 $Q \in \mathcal{L}[-{:}\mathbf{1}]$ imply $\cdot;\cdot \vdash P :: T$ and $\cdot;\cdot \vdash Q :: -{:}\mathbf{1}$, respectively. Hence, we can derive the typing $\cdot;\cdot \vdash P \mid Q :: T$ (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 $T = -{:}\mathbf{1}$ 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 $P \mid Q \in \mathcal{L}[-{:}\mathbf{1}]$. The cases in which $T \neq -{:}\mathbf{1}$ rely on the fact that if $P \stackrel{\alpha}{\Longrightarrow} P'$ then there exists a process R such that $P \mid Q \stackrel{\alpha}{\Longrightarrow} R$. The proof is by induction on $k = \mathsf{mlen}(Q)$. If $k = 0$ then $Q \not\longrightarrow$ and for every weak transition $P \stackrel{\alpha}{\Longrightarrow} P'$, we have $P \mid Q \stackrel{\alpha}{\Longrightarrow} P' \mid Q = R$. 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 $P \stackrel{\alpha}{\Longrightarrow} P'$ then by induction hypothesis there is an R' such that $P \mid Q \stackrel{\alpha}{\Longrightarrow} P' \mid Q' = R'$, with Q reducing to Q' in $k-1$ steps. Then, if $Q' \longrightarrow Q''$ we would have $P \mid Q \stackrel{\alpha}{\Longrightarrow} P' \mid Q' \longrightarrow P' \mid Q''$ which is equivalent to write $P \mid Q \stackrel{\alpha}{\Longrightarrow} R$, with $R = P' \mid Q''$, and we are done. Finally, we observe that, given $P \stackrel{\alpha}{\Longrightarrow} P'$, 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. □

Appendix A.3. Proof of Lemma 5.2

We repeat the statement in Page 20 below. In the proof, we use $G, G', \ldots$ and $D, D', \ldots$ to range over processes in $\mathcal{C}_\Gamma$ and $\mathcal{C}_\Delta$, 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 Appendix A.1 (5.2)**.** *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 of them, we use Lemma 5.1 and show that every $M = (\nu\widetilde{u},\widetilde{x})(P \mid G \mid D)$ with $G \in \mathcal{C}_\Gamma$ and $D \in \mathcal{C}_\Delta$, is in $\mathcal{L}[T]$. In case (Tid), the proof uses Prop 5.4 (closure wrt substitution) and Prop 5.9 (backward closure). In cases (T$\otimes$L), (T$\multimap$L), (Tcopy), (T$\oplus$L), (T$\&$L$_1$), and (T$\&$L$_2$), the proof proceeds in two steps: first, relying on Prop 5.8 (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 5.9 (backward closure) to conclude that $M \in \mathcal{L}[T]$. In cases (T**1**R), (T$\otimes$R), (T$\multimap$R), (T!R), (T$\oplus$R$_1$), and (T$\oplus$R$_2$), the proof consists in showing that M conforms to some specific case of Def 5.3. Case (T**1**L) uses Prop 5.10 (weakening). Cases (T$\otimes$L), (T$\multimap$L), (T$\oplus$L), and (T$\&$L$_1$), use the liveness guarantee given by Prop 5.6. Cases (Tcopy), (T!L)and (Tcut$^!$) use Prop 5.5 (closure under $\equiv$). Cases (Tcut), (T$\multimap$R), and (T!R) use Prop 5.7 (closure under $\equiv_!$).

0. Case (Tid**):** $\Gamma; x{:}A \vdash [x \leftrightarrow z] :: z{:}A$.

Pick any $G \in \mathcal{C}_\Gamma$:
(a) $G\Downarrow, G \not\longrightarrow$ [By Prop 5.3]
(b) $D \in \mathcal{L}[x{:}A]$
(c) $M = (\nu\widetilde{u}, x)([x \leftrightarrow z] \mid G \mid D) \in \mathcal{L}[z{:}A]$
The proof of (c) is immediate:
(d) $M \longrightarrow (\nu\widetilde{u})(G\{^z/_x\} \mid D\{^z/_x\})$
$\equiv_!\ D\{^z/_x\} = M'$ [Since $x \notin \mathit{fn}(G)$]
(e) $M' \in \mathcal{L}[z{:}A]$ [By (b) and Prop 5.4]
(f) $M \in \mathcal{L}[z{:}A]$ [By (d), (e), and Prop 5.9]
$[x \leftrightarrow z] \in \mathcal{L}[\Gamma; x{:}A \vdash z{:}A]$ [By (c) and Lemma 5.1]

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

Pick any $G \in \mathcal{C}_\Gamma$:
(a) $G\Downarrow, G \not\longrightarrow$ [By Prop 5.3]
(b) $M = (\nu\widetilde{u})(\mathbf{0} \mid G) \in \mathcal{L}[z{:}\mathbf{1}]$
The proof of (b) is immediate:
(c) $M \not\longrightarrow \wedge M \equiv_!\ \mathbf{0}$ [Using (a)]
(d) $M \in \mathcal{L}[z{:}\mathbf{1}]$ [By (c) and Def 5.3]
$\mathbf{0} \in \mathcal{L}[\Gamma;\cdot \vdash z{:}\mathbf{1}]$ [By (b) and Lemma 5.1]

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

(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) $M_1 = (\nu \widetilde{u}, \widetilde{x})(P \mid G \mid D) \in \mathcal{L}[T]$ [By Lemma 5.1 on (b)]
Pick any $R \in \mathcal{L}[z{:}\mathbf{1}]$ and fix $M_2 = M_1 \mid R$
(d) $M_2 \in \mathcal{L}[T]$ [By (c) and Prop 5.10]
(e) $(\nu \widetilde{u}, \widetilde{x}, z)(P \mid G \mid D \mid R) \in \mathcal{L}[T]$ [Expanding (d)]
$P \in \mathcal{L}[\Gamma; \Delta, z{:}\mathbf{1} \vdash T]$ [By (e) and Lemma 5.1]

3. Case (T$\otimes$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$\otimes$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) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]
(d) $(\nu \widetilde{u}, \widetilde{x}, y, z)(P \mid G \mid D \mid \mathcal{L}[y{:}A] \mid \mathcal{L}[z{:}B]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (b)]
Pick $R \in \mathcal{L}[z{:}A \otimes B]$:
(e) $\cdot; \cdot \vdash R :: z{:}A \otimes B,\ R\Downarrow$ [By Def 5.3]
(f) $R \stackrel{\overline{z\langle y \rangle}}{\Longrightarrow} R'$ [By (e) and Prop 5.6]
(g) $R' \equiv_! R'_1 \mid R'_2 \wedge R'_1 \in \mathcal{L}[y{:}A] \wedge R'_2 \in \mathcal{L}[z{:}B]$ [By Def 5.3]
Fix $M = (\nu \widetilde{u}, \widetilde{x}, z)(z(y).P \mid G \mid D \mid R)$
(h) $\forall M''.M \Longrightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$
We prove (h) by induction on $k = \mathsf{mlen}(D, R)$: [Possible by (c) and (e)]
Base case $k = 0$. Hence, $D \not\longrightarrow$, and $R \not\longrightarrow$:
$M \longrightarrow (\nu \widetilde{u}, \widetilde{x}, z, y)(P \mid G \mid D \mid R'_1 \mid R'_2) = M''$ [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 5.8]
(i) $M \in \mathcal{L}[T]$ [By (h) and Prop 5.9]
$z(y).P \in \mathcal{L}[\Gamma; \Delta, z{:}A \otimes B \vdash T]$ [By (i) and Lemma 5.1]

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

(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) $G{\Downarrow}, G \not\longrightarrow, D{\Downarrow}, D'{\Downarrow}$ [By Prop 5.3]
(f) $(\nu\widetilde{u}, \widetilde{x_1})(P \mid G \mid D) \in \mathcal{L}[y{:}A]$ [By Lemma 5.1 on (c)]
(g) $(\nu\widetilde{u}, \widetilde{x_2})(Q \mid G \mid D') \in \mathcal{L}[z{:}B]$ [By Lemma 5.1 on (d)]
Fix $\widetilde{x} = \widetilde{x_1} \cup \widetilde{x_2}$:
(h) $M = (\nu\widetilde{u}, \widetilde{x})((\nu y)z\langle y\rangle.(P \mid Q) \mid G \mid D \mid D') \in \mathcal{L}[z{:}A \otimes B]$
We prove (h) by induction on $k = \mathsf{mlen}(D, D')$: [Possible by (e)]
Base case $k = 0$. Hence, $D \not\longrightarrow$, and $D' \not\longrightarrow$:
(i) $M \xrightarrow{\overline{z\langle y\rangle}} (\nu\widetilde{u}, \widetilde{x})(P \mid Q \mid G \mid D \mid D') = M'$
$M' \equiv_! \underbrace{(\nu\widetilde{u}, \widetilde{x_1})(P \mid G \mid D)}_{M_1'} \mid \underbrace{(\nu\widetilde{u}, \widetilde{x_2})(Q \mid G \mid D')}_{M_2'}$
(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 5.3, using (i), (j), and (k)]
Inductive case $k > 0$:
Fix the set $W = \{M' \mid M \longrightarrow^{k-1} M'\}$
$\forall M' \in W.M' \in \mathcal{L}[z{:}A \otimes B]$ [By i.h.]
Fix the set $W' = \{M'' \mid M' \longrightarrow M'' \wedge M' \in W\}$
$\forall M'' \in W'.M'' \in \mathcal{L}[z{:}A \otimes B]$ [By Prop 5.8]
$\overline{z}\langle y\rangle.(P \mid Q) \in \mathcal{L}[\Gamma; \Delta, \Delta' \vdash z{:}A \otimes B]$ [By (h) and Lemma 5.1]

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

(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) $G{\Downarrow}, G \not\longrightarrow, D{\Downarrow}, D'{\Downarrow}$ [By Prop 5.3]
(f) $(\nu\widetilde{u}, \widetilde{x_1})(P \mid G \mid D) \in \mathcal{L}[y{:}A]$ [By Lemma 5.1 on (c)]
(g) $(\nu\widetilde{u}, \widetilde{x_2}, z)(Q \mid G \mid D' \mid \mathcal{L}[z{:}B]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (d)]

Fix $\widetilde{x} = \widetilde{x_1} \cup \widetilde{x_2}$ and pick $R \in \mathcal{L}[z{:}A \multimap B]$:

(h) $\cdot;\cdot \vdash R :: z{:}A \multimap B,\ R\Downarrow$ [By Def 5.3]

(i) $R \stackrel{z(y)}{\Longrightarrow} R'$ [By (h) and Prop 5.6]

(j) $\forall Q \in \mathcal{L}[y{:}A].(\nu y)(R' \mid Q) \in \mathcal{L}[z{:}B]$ [By Def 5.3]

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

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

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

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

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

Fix $M^* = (\nu \widetilde{u}, \widetilde{x_1})(P \mid G \mid D)$:

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

$M'' \equiv_! (\nu \widetilde{u}, \widetilde{x_2}, z)(Q \mid G \mid D' \mid\ (\nu y)(R' \mid M^*)) = M_1$

(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 5.7 and (n)]

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 5.8]

(o) $M \in \mathcal{L}[T]$ [By (k) and Prop 5.9]

$\overline{z}\langle y\rangle.(P \mid Q) \in \mathcal{L}[\Gamma; \Delta, \Delta', z{:}A \multimap B \vdash T]$ [By (o) and Lemma 5.1]

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) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]

(d) $(\nu \widetilde{u}, \widetilde{x}, y)(P \mid G \mid D \mid \mathcal{L}[y{:}A]) \in \mathcal{L}[z{:}B]$ [By Lemma 5.1 on (b)]

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

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

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

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

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

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

$M \in \mathcal{L}[z{:}A \multimap B]$ [By Def 5.3, using (f),(g)]

Inductive case $k > 0$:

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

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

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

$\forall M'' \in W'. M'' \in \mathcal{L}[z{:}A \multimap B]$ [Prop 5.8]

$z(y).P \in \mathcal{L}[\Gamma; \Delta \vdash z{:}A \multimap B]$ [By Lemma 5.1 on (e)]

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

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

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

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

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

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

(e) $(\nu \widetilde{u}, \widetilde{x_1})(P \mid G \mid D) \in \mathcal{L}[z{:}A]$ [By Lemma 5.1 on (c)]

(f) $(\nu \widetilde{u}, \widetilde{x_2}, z)(Q \mid G \mid D' \mid \mathcal{L}[z{:}A]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (d)]

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

$M \equiv_! (\nu z)((\nu \widetilde{u}, \widetilde{x_2})(Q \mid G \mid D') \mid \underbrace{(\nu \widetilde{u}, \widetilde{x_1})(P \mid G \mid D)}_{M_1}) = M'$

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

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

(i) $M \in \mathcal{L}[T]$ [Using (h) and Prop 5.7]

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

$(\nu z)(P \mid Q) \in \mathcal{L}[\Gamma; \Delta, \Delta' \vdash T]$ [By Lemma 5.1 on (j)]

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

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

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

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

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

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

(e) $(\nu \widetilde{u})(P \mid G) \in \mathcal{L}[y{:}A]$ [By Lemma 5.1 on (c)]

(f) $(\nu \widetilde{u}, \widetilde{x}, z)(Q \mid G \mid !z(y).\mathcal{L}[y{:}A] \mid D) \in \mathcal{L}[T]$ [By Lemma 5.1 on (d)]

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

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

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

(h) $M' \in \mathcal{L}[T]$ [From (g), using Prop 5.5]

$(\nu z)(!z(y).P \mid Q) \in \mathcal{L}[\Gamma; \Delta \vdash T]$ [By Lemma 5.1 on (h)]

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

(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) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]
(d) $(\nu\widetilde{u}, z, \widetilde{x}, y)(P \mid G \mid !z(y).\mathcal{L}[y{:}A] \mid D \mid \mathcal{L}[y{:}A]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (b)]
Pick $R \in \mathcal{L}[y{:}A]$:
Fix $M = (\nu\widetilde{u}, z, \widetilde{x})(\overline{z}\langle y\rangle.P \mid G \mid !z(y).R \mid D)$
(e) $\forall M''.M \Longrightarrow M'' \Rightarrow M'' \in \mathcal{L}[T]$
We prove (e) by induction on $k = \mathsf{mlen}(D)$: [Possible by (c)]
Base case $k = 0$. Hence, $D \not\longrightarrow$:
$M \longrightarrow\!\!\equiv (\nu\widetilde{u}, z, \widetilde{x}, y)(P \mid G \mid !z(y).R \mid D \mid R) = M''$
$M'' \in \mathcal{L}[T]$ [Using (d) and Prop 5.5]
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 5.8]
(f) $M \in \mathcal{L}[T]$ [By (e) and Prop 5.9]
$\overline{z}\langle y\rangle.P \in \mathcal{L}[\Gamma, z{:}A; \Delta \vdash T]$ [By (f) and Lemma 5.1]

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) $(\nu\widetilde{u}, z, \widetilde{x})(P\{z/y\} \mid G \mid !z(w).\mathcal{L}[w{:}A] \mid D) \in \mathcal{L}[T]$ [By Lemma 5.1 on (b)]
(d) $(\nu\widetilde{u}, y, \widetilde{x})(P \mid G \mid \underbrace{!y(w).\mathcal{L}[w{:}A]}_{R} \mid D) \in \mathcal{L}[T]$ [By $\equiv$ (α-conv) on (c)]
(e) $R \in \mathcal{L}[y{:}!A]$ [By Def 5.3]
$P \in \mathcal{L}[\Gamma; \Delta, y{:}!A \vdash T]$ [By (d), (e), Prop 5.5, and Lemma 5.1]

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) $G\Downarrow, G \not\longrightarrow$ [By Prop 5.3]

(d) $(\nu\widetilde{u})(P \mid G) \in \mathcal{L}[y{:}A]$ [By Lemma 5.1 on (b)]

Fix $M = (\nu\widetilde{u})(!z(y).P \mid G)$

$M \equiv_! !z(y).(\nu\widetilde{u})(P \mid G) = M'$ [By Def 5.2, Axiom (2)]

(e) $M' \in \mathcal{L}[z{:}!A]$ [By Def 5.3, using (d)]

(f) $M \in \mathcal{L}[z{:}!A]$ [By (e) and Prop 5.7]

$!z(y).P \in \mathcal{L}[\Gamma; \cdot \vdash z{:}!A]$ [By (f) and Lemma 5.1]

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

(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) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]

(f) $(\nu\widetilde{u}, \widetilde{x}, z)(P \mid G \mid D \mid \mathcal{L}[z{:}A]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (c)]

(g) $(\nu\widetilde{u}, \widetilde{x}, z)(Q \mid G \mid D \mid \mathcal{L}[z{:}B]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (d)]

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

(h) $\cdot; \cdot \vdash R :: z{:}A \oplus B,\ R\Downarrow$ [By Def 5.3]

(i) $R \stackrel{\overline{z.\mathsf{inl}}}{\Longrightarrow} R' \vee R \stackrel{\overline{z.\mathsf{inr}}}{\Longrightarrow} R'$ [By (h) and Prop 5.6]

(j) $R \stackrel{\overline{z.\mathsf{inl}}}{\Longrightarrow} R' \Rightarrow R' \in \mathcal{L}[z{:}A] \wedge R \stackrel{\overline{z.\mathsf{inr}}}{\Longrightarrow} R' \Rightarrow R' \in \mathcal{L}[z{:}B]$ [By Def 5.3]

Fix $M = (\nu\widetilde{u}, \widetilde{x}, z)(z.\mathtt{case}(P, Q) \mid G \mid D \mid R)$:

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

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

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

$M \longrightarrow M_1'' \vee M'' \longrightarrow M_2''$, where :

$M_1'' = (\nu\widetilde{u}, \widetilde{x}, z)(P \mid G \mid D \mid R')$ [Because of (i)]

$M_2'' = (\nu\widetilde{u}, \widetilde{x}, z)(Q \mid G \mid D \mid R')$ [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 $W = \{M' \mid M \longrightarrow^{k-1} M'\}$

$\forall M' \in W.M' \in \mathcal{L}[T]$ [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 5.8]

(l) $M \in \mathcal{L}[T]$ [By (k) and Prop 5.9]

$z.\mathtt{case}(P, Q) \in \mathcal{L}[\Gamma; \Delta, z{:}A \oplus B \vdash T]$ [By (l) and Lemma 5.1]

13. Case (T&L$_1$): $\Gamma; \Delta, z{:}A \,\&\, B \vdash z.\mathtt{inl}; P :: T$

(a) $\Gamma; \Delta, z{:}A \vdash P :: T$ [Premise of rule (T&L$_1$)]

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

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

(c) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]

(d) $(\nu\widetilde{u}, \widetilde{x}, z)(P \mid G \mid D \mid \mathcal{L}[z{:}A]) \in \mathcal{L}[T]$ [By Lemma 5.1 on (b)]

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

(e) $\cdot; \cdot \vdash R :: z{:}A \,\&\, B,\ R\Downarrow$ [By Def 5.3]

(f) $R \overset{z.\mathsf{inl}}{\Longrightarrow} R_1 \vee R \overset{z.\mathsf{inr}}{\Longrightarrow} R_2$ [By (e) and Prop 5.6]

(g) $R \overset{z.\mathsf{inl}}{\Longrightarrow} R_1 \Rightarrow R_1 \in \mathcal{L}[z{:}A] \wedge R \overset{z.\mathsf{inr}}{\Longrightarrow} R_2 \Rightarrow R_2 \in \mathcal{L}[z{:}B]$ [By Def 5.3]

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

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

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

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

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

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

Inductive case $k > 0$:

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

$\forall M' \in W.M' \in \mathcal{L}[T]$ [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 5.8]

(j) $M \in \mathcal{L}[T]$ [By (h) and Prop 5.9]

$z.\mathtt{inl}; P \in \mathcal{L}[\Gamma; \Delta, z{:}A \,\&\, B \vdash T]$ [By (j) and Lemma 5.1]

14. Case (T&L$_2$): Analogous to case (T&L$_1$).

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

(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) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]

(f) $(\nu\widetilde{u},\widetilde{x})(P \mid G \mid D) \in \mathcal{L}[z{:}A]$ [By Lemma 5.1 on (c)]

(g) $(\nu\widetilde{u},\widetilde{x})(Q \mid G \mid D) \in \mathcal{L}[z{:}B]$ [By Lemma 5.1 on (d)]

(h) $M = (\nu\widetilde{u},\widetilde{x})(z.\mathtt{case}(P,Q) \mid G \mid D) \in \mathcal{L}[z{:}A \,\&\, B]$

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

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

(i) $M \xrightarrow{z.\mathsf{inl}} M_1 \wedge M \xrightarrow{z.\mathsf{inr}} M_2$, where :

$M_1 = (\nu\widetilde{u},\widetilde{x})(P \mid G \mid D)$

$M_2 = (\nu\widetilde{u},\widetilde{x})(Q \mid G \mid D)$

(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 5.3, using (i), (j), (k)]

Inductive case $k > 0$:

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

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

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

$\forall M'' \in W'.M'' \in \mathcal{L}[z{:}A \,\&\, B]$ [By Prop 5.8]

$z.\mathtt{case}(P,Q) \in \mathcal{L}[\Gamma;\Delta \vdash z{:}A \,\&\, B]$ [By (h) and Lemma 5.1]

16. Case (T$\oplus$R$_1$): $\Gamma;\Delta \vdash z.\mathtt{inl};P :: z{:}A \oplus B$

(a) $\Gamma;\Delta \vdash z.\mathtt{inl};P :: z{:}A \oplus B$ [Premise of rule (T$\oplus$R$_1$)]

(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) $G\Downarrow, G \not\longrightarrow, D\Downarrow$ [By Prop 5.3]

(d) $(\nu\widetilde{u},\widetilde{x})(P \mid G \mid D) \in \mathcal{L}[z{:}A]$ [By Lemma 5.1 on (b)]

(e) $M = (\nu\widetilde{u},\widetilde{x})(z.\mathtt{inl};P \mid G \mid D) \in \mathcal{L}[z{:}A \oplus B]$

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

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

(i) $M \xrightarrow{\overline{z.\mathtt{inl}}} (\nu\widetilde{u},\widetilde{x})(P \mid G \mid D) = M_1$

(j) $M_1 \in \mathcal{L}[z{:}A]$ [Using (d)]

$M \in \mathcal{L}[z{:}A \oplus B]$ [Using (i), (j), and Def 5.3]

Inductive case $k > 0$:

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

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

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

$\forall M'' \in W'.M'' \in \mathcal{L}[z{:}A \oplus B]$ [By Prop 5.8]

$z.\mathtt{inl}; P \in \mathcal{L}[\Gamma; \Delta \vdash z{:}A \oplus B]$ [By (e) and Lemma 5.1]

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

□

Appendix A.4. Proof of Proposition 5.11

We repeat the statement in Page 21 and present its proof.

Proposition Appendix A.3 (Properties of Confluent Processes)**.** *Assume well-typed processes P, P', $P_1, \ldots, P_k$, Q. Then we have:*

1. *Forward closure: If $P\Diamond$ and $P \longrightarrow P'$ then $P'\Diamond$.*
2. *Backward closure: If for all P_i such that $P \longrightarrow P_i$ we have that $P_i\Diamond$, then $P\Diamond$.*
3. *Closure wrt composition: Let P, Q be such that (i) $\cdot; \cdot \vdash P :: x{:}A$, (ii) $\cdot; x{:}A \vdash Q :: T$, (iii) $P\Diamond$, and (iv) $Q\Diamond$. Then $(\nu x)(P \mid Q)\Diamond$.*

Proof. 1. By assumption, we have $P\Diamond$ and $P \longrightarrow P'$. We have to show that for any P_1, P_2 such that $P' \Longrightarrow P_1$ and $P' \Longrightarrow P_2$, there exists process P_3 such that $P_1 \Longrightarrow P_3$ and $P_2 \Longrightarrow P_3$. Since $P\Diamond$, for any R_1, R_2 such that $P \Longrightarrow R_1$ and $P \Longrightarrow R_2$, there exists an R' such that $R_1 \Longrightarrow R'$ and $R_2 \Longrightarrow R'$. This includes the particular case in which $P \longrightarrow P' \Longrightarrow P_1$ and $P \longrightarrow P' \Longrightarrow P_2$. The existence of a P_3 such that $P_1 \Longrightarrow P_3$ and $P_2 \Longrightarrow P_3$ follows from $P\Diamond$. Therefore, $P'\Diamond$.

2. Given that whenever $P \longrightarrow P_k$, we have $P_k\Diamond$, we need to "complete the diamond", starting from P. Precisely, we have to show that for any P_i, P_j such that $P \longrightarrow P_i$, $P \longrightarrow P_j$, $P_i\Diamond$, and $P_j\Diamond$, and for any R_1, R_2 such that $P \longrightarrow P_i \Longrightarrow R_1$ and $P \longrightarrow P_j \Longrightarrow R_2$, there exists an R' such that $R_1 \Longrightarrow R'$ and $R_2 \Longrightarrow R'$.

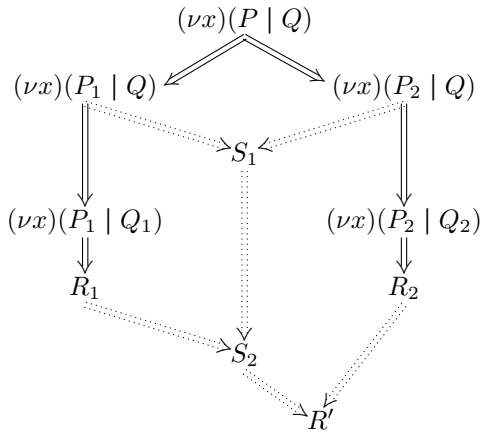
   To do so, we rely on the following fact. Let P be a process with two reductions $P \longrightarrow P_1$ and $P \longrightarrow P_2$ on private names x and y, respectively. If $P \longrightarrow P_1$ then there is always a P_2' such that $P_1 \longrightarrow P_2'$ via a synchronization on y. Conversely, if $P \longrightarrow P_2$ then there is always a P_1' such that $P_2 \longrightarrow P_1'$ via a synchronization on x. This can be shown by a case analysis on the different ways in which reductions can arise (communication, selection, shared server invokation), using linearity, subject reduction and progress. This fact implies that a reduction $P \longrightarrow P_i$ does not preclude reduction paths reachable if the first reduction is $P \longrightarrow P_j$. Since $P_i\Diamond$ and $P_j\Diamond$, this allows to conclude that $P\Diamond$, as desired.

3. There are two cases, depending on whether P and Q can interact. In turn, this depends on their typing. If $A{=}\mathbf{1}$ then P and Q do not interact and $P \mid Q$ represents their independent parallel composition. That is, their reductions proceed independently. The fact that $(P \mid Q)\diamondsuit$ holds then follows from $P\diamondsuit$ and $Q\diamondsuit$.

   If $A{\neq}\mathbf{1}$ then P and Q can interact. Since P, Q are well-typed, such an interaction occurs exclusively on name x. We consider the following two possibilities:

$$\begin{array}{rcl} R & \Longrightarrow & (\nu x)(P_1 \mid Q) \Longrightarrow (\nu x)(P_1 \mid Q_1) \Longrightarrow R_1 \\ R & \Longrightarrow & (\nu x)(P_2 \mid Q) \Longrightarrow (\nu x)(P_2 \mid Q_2) \Longrightarrow R_2 \end{array}$$

   where, intuitively, synchronizations on x are included in the last weak transition. We show that there exists an R' such that $R_1 \Longrightarrow R'$ and $R_2 \Longrightarrow R'$. Now, since $P\diamondsuit$ there exists a process $S_1 = (\nu x)(P_3 \mid Q)$ such that $(\nu x)(P_1 \mid Q) \Longrightarrow S_1$ and $(\nu x)(P_2 \mid Q) \Longrightarrow S_1$. By Property 1, we know that $(\nu x)(P_1 \mid Q)\diamondsuit$. Hence, there exists a process S_2 such that $R_1 \Longrightarrow S_2$ and $S_1 \Longrightarrow S_2$. Also by Property 1, we know that $(\nu x)(P_2 \mid Q)\diamondsuit$. This ensures the existence of the desired R': since $(\nu x)(P_2 \mid Q) \Longrightarrow S_2$ and $(\nu x)(P_2 \mid Q) \Longrightarrow R_2$, then there exists a process R' such that $S_2 \Longrightarrow R'$ and $R_2 \Longrightarrow R'$. The diagram below depicts the weak transitions just described.



□

Appendix A.5. Proof of Lemma 5.4

We repeat the statement in Page 24 and give details of the proof.

Lemma Appendix A.2. *Let P be a process. If $\Gamma; \Delta \vdash P :: T$ then $P \in \mathcal{L}^{\diamondsuit}[\Gamma; \Delta \vdash T]$.*

Proof. By induction on the derivation of $\Gamma; \Delta \vdash P :: T$, with a case analysis on the last typing rule used. We have 18 cases to check; in all cases, we use Lemma 5.3 to show that every $M = (\nu \widetilde{u}, \widetilde{x})(P \mid G \mid D)$ with $G \in \mathcal{C}^{\diamondsuit}_{\Gamma}$ and $D \in \mathcal{C}^{\diamondsuit}_{\Delta}$, is in $\mathcal{L}^{\diamondsuit}[T]$.

The proof follows closely the lines of the proof of Lemma 5.2 (Page 44); it exploits the fact that well-typed processes are always terminating (Theorem 5.1). In case (Tid),

we use Proposition 5.14 (closure wrt substitution) and Proposition 5.18 (backward closure). In cases (T$\otimes$L), (T$\multimap$L), (Tcopy), (T$\oplus$L), (T$\&$L$_1$), and (T$\&$L$_2$), we proceed in two steps: first, using Proposition 5.17 (forward closure) we show that every M'' such that $M \Longrightarrow M''$ is in $\mathcal{L}^{\Diamond}[T]$; then, we use this result in combination with Proposition 5.18 (backward closure) to conclude that $M \in \mathcal{L}^{\Diamond}[T]$. In cases (T$\mathbf{1}$R), (T$\otimes$R), (T$\multimap$R), (T!R), (T$\oplus$R$_1$), and (T$\oplus$R$_2$), we show that M conforms to a specific case of Definition 5.7. Case (T$\mathbf{1}$L) uses Proposition 5.11 (3). Cases (T$\otimes$L), (T$\multimap$L), (T$\oplus$L), and (T$\&$L$_1$) use the liveness guarantee given by Proposition 5.6. Cases (Tcopy), (T!L), and (Tcut$^!$) use Proposition 5.15 (closure under $\equiv$). Cases (Tcut), (T$\multimap$R), and (T!R) use Proposition 5.16 (closure under $\equiv_!$).

Below, we illustrate a few cases; the rest are essentially as in the proof of Lemma 5.2 (Page 44).

Case (Tid**):** $\Gamma; x{:}A \vdash [x \leftrightarrow z] :: z{:}A$.

Pick any $G \in \mathcal{C}_{\Gamma}^{\Diamond}$:
(a) $G\Diamond, G \not\longrightarrow$ [By Prop 5.13]
(b) $D \in \mathcal{L}^{\Diamond}[x{:}A]$
(c) $M = (\nu \widetilde{u}, x)([x \leftrightarrow z] \mid G \mid D) \in \mathcal{L}^{\Diamond}[z{:}A]$
The proof of (c) is immediate:
(d) $M \longrightarrow (\nu \widetilde{u})(G\{z/x\} \mid D\{z/x\})$
$\equiv_! D\{z/x\} = M'$ [Since $u_i \notin \mathit{fn}(D)$]
(e) $M' \in \mathcal{L}^{\Diamond}[z{:}A]$ [By (b) and Prop 5.14]
(f) $M \in \mathcal{L}^{\Diamond}[z{:}A]$ [By (d), (e), and Prop 5.18]
$[x \leftrightarrow z] \in \mathcal{L}^{\Diamond}[\Gamma; x{:}A \vdash z{:}A]$ [By (c) and Lemma 5.3]

Case (T$\mathbf{1}$R): $\Gamma; \cdot \vdash \mathbf{0} :: z{:}\mathbf{1}$.

Pick any $G \in \mathcal{C}_{\Gamma}^{\Diamond}$:
(a) $G\Diamond, G \not\longrightarrow$ [By Prop 5.13]
(b) $M = (\nu \widetilde{u})(\mathbf{0} \mid G) \in \mathcal{L}^{\Diamond}[z{:}\mathbf{1}]$
The proof of (b) is immediate:
(c) $M\Diamond \wedge M \not\longrightarrow \wedge M \equiv_! \mathbf{0}$ [Using (a)]
(d) $M \in \mathcal{L}^{\Diamond}[z{:}\mathbf{1}]$ [By (c) and Def 5.7]
$\mathbf{0} \in \mathcal{L}^{\Diamond}[\Gamma; \cdot \vdash z{:}\mathbf{1}]$ [By (b) and Lemma 5.3]

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

(a) $\Gamma; \Delta \vdash P :: T$ [Premise of rule (T1L)]

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

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

(c) $M_1 = (\nu \widetilde{u}, \widetilde{x})(P \mid G \mid D) \in \mathcal{L}^{\Diamond}[T]$ [By Lemma 5.3 on (b)]

Pick any $R \in \mathcal{L}^{\Diamond}[z{:}\mathbf{1}]$ and fix $M_2 = M_1 \mid R$

(d) $M_2 \in \mathcal{L}^{\Diamond}[T]$ [By (c) and Prop 5.11 (3)]

(e) $(\nu \widetilde{u}, \widetilde{x}, z)(P \mid G \mid D \mid R) \in \mathcal{L}^{\Diamond}[T]$ [Expanding (d)]

$P \in \mathcal{L}^{\Diamond}[\Gamma; \Delta, z{:}\mathbf{1} \vdash T]$ [By (e) and Lemma 5.3]

□

Appendix B. Proofs from Section 6 (Typed Context Bisimilarity)

Appendix B.1. Proof of Proposition 6.3

We repeat the statement in Page 28, and present its proof.

Proposition Appendix B.1 (6.3)**.** *$\Gamma; \Delta \vdash P \approx Q :: T$ implies $\vdash K[P] \approx K[Q] :: T$, where K is any parallel representative in $\mathcal{K}_{\Gamma;\Delta\vdash T}$, as in Definition 6.5.*

Proof. Let $\#(\Gamma)$, $\#(\Delta)$ denote the cardinality of Γ, Δ, respectively. The proof is by induction on $n = \#(\Gamma) + \#(\Delta)$. The base case is when $n = 0$: then both typing environments are empty and so $K = \bullet$. Hence, $K[P] = P$ and $K[Q] = Q$ and the thesis trivially holds. In the inductive case, $n > 0$, and there are two sub-cases. In the first one, we have $\Gamma, u_i{:}G_i; \Delta \vdash P \approx Q :: T$. By definition of $\approx$, it 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 $\vdash K_1[(\nu u_i)(!u_i(y_i).S \mid P)] \approx K_1[(\nu u_i)(!u_i(y_i).S \mid Q)] :: T$, for every $K_1 \in \mathcal{K}_{\Gamma,\Delta\vdash T}$. We observe that, for any R, $K_1[(\nu u_i)(!u_i(y_i).S \mid R)]$ is the same as having $K_0[R]$, with a context $K_0 = (\nu u_i)(!u_i(y_i).S \mid K_1[\cdot])$. By Definition 6.5, we infer that $K_0 \in \mathcal{K}_{\Gamma, u_i:G_i;\Delta\vdash T}$. Therefore, $\Gamma, u_i{:}G_i; \Delta \vdash P \approx Q :: T$ implies

$$\vdash K[P] \approx K[Q] :: T$$

for any $K \in \mathcal{K}_{\Gamma, u_i:G_i;\Delta}$, as desired. In the second sub-case, we have $\Gamma; \Delta, x_j{:}A_j \vdash P \approx Q :: T$, and the analysis follows the same lines as before. □

Appendix B.2. Additional Cases for Proof of Lemma 6.1

We repeat the statement in Page 29, and detail some additional cases, thus complementing the proof given in that page.

Lemma Appendix B.1 (Contextuality of $\approx$)**.** *Typed context bisimilarity is a contextual relation, in the sense of Definition 6.3.*

Proof. The proof proceeds by coinduction, exhibiting a typed context bisimulation for each of the conditions associated to Definition 6.3. We shall exploit the proof technique given by Proposition 6.3, which allows us to consider $\approx$ under empty left-hand side contexts, for pairs of processes enclosed within appropriate parallel representatives. As a result, it suffices to consider only some of the conditions in Table 8; see Remark 6.1. In Page 29 we have detailed the case of closure under output prefix; below we show the cases for closure under parallel composition and under replicated input (Items (8) and (15), respectively).

Item (8): We have to show that $\Gamma; \Delta_1 \vdash P \approx Q :: y{:}A$ implies

$$\Gamma; \Delta_1, \Delta_2 \vdash (\nu y)(P \mid S) \approx (\nu y)(Q \mid S) :: T$$

for any S, T, Δ_2 such that $\Gamma; \Delta_2, y{:}A \vdash S :: T$. Using Proposition 6.3, it suffices to show $\vdash K_1[P] \approx K_1[Q] :: y{:}A$ implies

$$\vdash K_2[(\nu y)(K_1[P] \mid S)] \approx K_2[(\nu y)(K_1[Q] \mid S)] :: T$$

where $K_1 \in \mathcal{K}_{\Gamma;\Delta_1 \vdash y:A}$ and $K_2 \in \mathcal{K}_{\cdot;\Delta_2 \vdash x:T}$.
Letting $M = K_2[(\nu y)(K_1[P] \mid S)]$, $N = K_2[(\nu y)(K_1[Q] \mid S)]$ we show that

$$\begin{array}{lcl} \mathcal{R}_8 & = & \{(M,N) : \vdash K_1[P] \approx K_1[Q] :: y{:}A,\ K_1 \in \mathcal{K}_{\Gamma;\Delta_1 \vdash y:A},\ K_2 \in \mathcal{K}_{\cdot;\Delta_2 \vdash x:T}\} \\ & \cup & \mathcal{W}_{\vdash T} \end{array}$$

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

1. $\alpha = \tau$ and $\vdash M' :: T$, using subject reduction (Theorem 3.1)
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 examine the different possibilities for the origin of the reduction:

1. The reduction originates from K_2. More precisely, by Definition 6.5 the reduction originates in the part of K implementing names in Δ_1, as the part of K_2 implementing names in Γ cannot evolve on its own (cf. Definition 6.5). Therefore, for some K_4, we have both $K_2 \xrightarrow{\tau} K_4$ and $M' = K_4[(\nu y)(K_1[P] \mid S)]$. By subject reduction (Theorem 3.1), the type of K_4 is the same than that of K, which in turn implies $\vdash M' :: T$. Since K_2 occurs identically in M and N, this reduction can be matched by N, possibly preceded/followed by zero or more reductions: and so we have that $N \Longrightarrow K_5[(\nu y)(K_3[Q'] \mid S')] = N'$, with $K_1 \Longrightarrow K_3$ $K_2 \Longrightarrow K_5$, $Q \Longrightarrow Q'$, and $S \Longrightarrow S'$. Theorem 5.1 ensures that these weak transitions are finite. Moreover, subject reduction (Theorem 3.1) ensures $\vdash N' :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_8$, and we are done.

2. The reduction originates from K_1. The argument proceeds analogously as in the previous case.
3. The reduction originates from P. Then, for some P', we have $P \xrightarrow{\tau} P'$ and $M' = K_2[(\nu y)(K_1[P'] \mid S)]$. By subject reduction (Theorem 3.1), the type remains unchanged, which in turn implies $\vdash M' :: T$. Since $\vdash K_1[P] \approx K_1[Q] :: y{:}A$, we infer that N can match this reduction: there is a Q' such that $Q \Longrightarrow Q'$. Again, reductions from Q may be preceded or followed by reductions from K_1, K_2, and S. More precisely, there is a weak transition $N \Longrightarrow K_5[(\nu y)(K_3[Q'] \mid S')] = N'$, with $K_1 \Longrightarrow K_3$ $K_2 \Longrightarrow K_5$, $Q \Longrightarrow Q'$, and $S \Longrightarrow S'$. Theorem 5.1 ensures these weak transitions are finite. Moreover, subject reduction (Theorem 3.1) ensures $\vdash N' :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_8$, and we are done.
4. The reduction originates from S. We proceed analogously as in the previous cases, relying on the fact that S is the same in M and N.
5. The reduction originates from the interaction of P and K_1. Therefore, for some K_3, P', we have $M' = K_2[(\nu y)(K_3[P'] \mid S)]$. By subject reduction (Theorem 3.1), we can infer that $\vdash M' :: T$. Since K_1 occurs identically in M and N, and $\vdash K_1[P] \approx K_1[Q] :: y{:}A$, we infer that this interaction can be matched by N. Hence, there is a weak transition $N \Longrightarrow K_5[(\nu y)(K_3[Q'] \mid S')] = N'$ with $K_1 \Longrightarrow K_3$, $K_2 \Longrightarrow K_5$ $Q \Longrightarrow Q'$, and $S \Longrightarrow S'$. Theorem 5.1 ensures these weak transitions are finite. Moreover, subject reduction (Theorem 3.1) ensures $\vdash N' :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_8$, and we are done.
6. The reduction originates from the interaction of S and K_2. The argument proceeds analogously as in the previous case.
7. The reduction originates from the interaction of P and S. Therefore, $M' = K_2[(\nu y)(K_1[P'] \mid S')]$. Using the typings of each process, we infer that this interaction is only possible via a synchronization on y, which offers (case of P) and requires (case of S) a behavior described by A. We then proceed by structural induction on type A. All the cases are covered by preservation lemmas which formalize the interaction of complementary actions. We detail only the case $A = A_1 \otimes A_2$; the other cases are similar. Using Lemma 3.2 we infer $P \xrightarrow{\overline{x\langle y\rangle}} P'$ and $S \xrightarrow{x(y)} S'$. Using Lemma 3.3 we infer that P' is well-typed, and we have $\vdash K_2[(\nu y)(K_1[P'] \mid S')] :: T$. Since $\vdash K_1[P] \approx K_1[Q] :: y{:}A$ and S is the same in N, we know that these actions can be matched by N, and that there exist Q', S' such that $Q \overset{\overline{(\nu y)x\langle y\rangle}}{\Longrightarrow} Q'$ and $S \overset{x(y)}{\Longrightarrow} S'$. Hence, there is an $N' = K_5[(\nu y)(K_3[Q'] \mid S')]$ with $K_1 \Longrightarrow K_3$ and $K_2 \Longrightarrow K_5$. By virtue of Theorem 5.1 these are all finite weak transitions. Using again Lemma 3.3 and subject reduction (Theorem 3.1), one can show that N' is well-typed: $\vdash K_5[(\nu y)(K_3[Q'] \mid S')] :: T$. Therefore, the pair (M', N') is in $\mathcal{R}_8$ 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 type T; the typing information ensures that T can only be provided by S. Therefore, we proceed by induction on the structure of the type T. We consider only the case $T = \underline{x{:}A_1} \otimes A_2$; the other cases are similar or simpler. Then, by Lemma 3.2, $\alpha = \overline{x\langle z\rangle}$ and $M' = K_2[(\nu y)(K_1[P] \mid S')]$. Since S is the same in N, we know that this action can be matched by N: indeed we have $S \stackrel{\overline{x\langle z\rangle}}{\Longrightarrow} S'$ and $N' = K_5[(\nu y)(K_3[Q'] \mid S')]$, with $K_1 \Longrightarrow K_3$, $K_2 \Longrightarrow K_5$, $Q \Longrightarrow Q'$, and $S \Longrightarrow S'$. Theorem 5.1 ensures these weak transitions are finite. Now we follow the definition of $\approx$ for output actions. Then, for any R such that $\cdot; z{:}A_1 \vdash R :: -{:}\mathbf{1}$, we verify that both $\vdash (\nu z)(M' \mid R) :: x{:}A_2$ and $\vdash (\nu z)(N' \mid R) :: x{:}A_2$ hold. Hence, the pair $((\nu z)(M' \mid R), (\nu z)(N' \mid R))$ is in $\mathcal{W}_{\vdash x:A_1 \otimes A_2}$ and we are done.

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

Item (15): We have to show that $\Gamma; \Delta \vdash P \approx Q :: y{:}A$ implies

$$\Gamma; \Delta \vdash !x(y).P \approx !x(y).Q :: x{:}!A$$

Using Proposition 6.3, it suffices to show that $\vdash K[P] \approx K[Q] :: y{:}A$ implies

$$\vdash !x(y).K[P] \approx !x(y).K[Q] :: x{:}!A$$

for any $K \in \mathcal{K}_{\Gamma;\Delta \vdash y:A}$. Let $M = !x(y).K[P]$ and $N = !x(y).K[Q]$. We show that

$$\mathcal{R}_{13} = \{(M, N) : \ \vdash K[P] \approx K[Q] :: y{:}A,\ K \in \mathcal{K}_{\Gamma;\Delta \vdash T}\} \cup \mathcal{W}_{\vdash x:!A}$$

is a typed context bisimulation. Suppose M moves first: $M \stackrel{\alpha}{\longrightarrow} M'$. We must find a matching action from N such that $N \stackrel{\alpha}{\Longrightarrow} N'$. The only possibility is an input on x and so we have $M \stackrel{x(z)}{\longrightarrow} !x(y).K[P] \mid K[P]\{^z/_y\} = M'$. Process N can match this action immediately: $N \stackrel{x(z)}{\longrightarrow} !x(y).K[Q] \mid K[Q]\{^z/_y\} = N'$. It is easy to show that typing is preserved by substitution, and so $\vdash K[P] \approx K[Q] :: y{:}A$ allows to infer $\vdash K[P]\{^z/_y\} \approx K[Q]\{^z/_y\} :: z{:}A$.

Following the clause for replicated input of $\approx$, we consider the closure of M' and N' with a process L such that $z{:}A \vdash L :: -{:}\mathbf{1}$. Such closures correspond, respectively, to

$$(\nu z)(K[P]\{^z/_y\} \mid L) \mid !x(y).K[P] \text{ and } (\nu z)(K[Q]\{^z/_y\} \mid L) \mid !x(y).K[Q]$$

We verify the type of these closures is indeed $x{:}!A$, as required by the replicated input clause. Since $\vdash K[P]\{^z/_y\}, K[Q]\{^z/_y\} :: z{:}A$, these processes can be composed with L, thus leading to processes of type $-{:}\mathbf{1}$. It is immediate to see that $\vdash !x(y).K[P], !x(y).K[Q] :: x{:}!A$; hence, via an independent parallel composition the two processes above are of type $x{:}!A$, and the pair

$$(\,(\nu z)(K[P]\{^z/_y\} \mid L) \mid !x(y).K[P]\,,\, (\nu z)(K[Q]\{^z/_y\} \mid L) \mid !x(y).K[Q]\,)$$

is in $\mathcal{R}_{13}$, as desired. The reasoning when N moves first is completely symmetric.

□

Appendix C. Proofs from Section 7.1 (Applications)

Appendix C.1. Additional Cases for the Proof of Theorem 7.1

We repeat the statement in Page 32, and detail some additional cases, thus complementing the proof given in that page.

Theorem Appendix C.1 (7.1)**.** *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 commuting conversion. In the bisimulation game, we exploit termination of well-typed processes (Theorem 5.1) to ensure that actions can be matched with finite weak transitions, and Theorem 3.1 to ensure preservation of type under reductions. We detail the cases of proof conversions I-2 and I-4 (cf. Figure 4), and I-36 (cf. Figure 5).

Proof conversion I-2 We then have that

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

with

$$\Gamma; \Delta_1 \vdash \hat{D} :: x{:}C \quad \Gamma; \Delta_2, x{:}C \vdash \hat{E} :: y{:}A \quad \Gamma; \Delta_3 \vdash \hat{F} :: z{:}B \tag{C.1}$$

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. 6.3, 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 $\mathcal{R}$ 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 6.6. Notice that $\mathcal{S}$ is a type-respecting relation indexed by $\vdash z{:}A \otimes B$. In fact, using the typings in (C.1)—with $\Gamma = \Delta = \emptyset$—and exploiting subject reduction (Theorem 3.1), 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 Def. 6.5, we can assume

$$K = (\nu \widetilde{u}, \widetilde{x})(\bullet \mid K_\Gamma \mid K_\Delta) \quad \text{where:}$$

- $K_\Gamma \equiv \prod_{i\in I} !u_i(y_i).R_i$, 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: $K[M] \xrightarrow{\alpha} M_1^\star$. We have to find a matching action α from $K[N]$, i.e., $K[N] \stackrel{\alpha}{\Longrightarrow} N_1^\star$. 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 $l \in J$, $S_l \xrightarrow{\tau} S'_l$:
   $$K[M] \xrightarrow{\tau} (\nu \widetilde{u}, \widetilde{x})(K_\Gamma \mid K'_\Delta \mid M) = K'[M] = M_1^\star$$
   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:
   $$K[N] \Longrightarrow (\nu \widetilde{u}, \widetilde{x})(K_\Gamma \mid K''_\Delta \mid N) = K''[N] = N_1^\star$$
   By subject reduction (Theorem 3.1), $\vdash S'_l :: x_l{:}C_l$; 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 $M = (\nu x)(\hat{D} \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F}))$, the only possibility is that there is a $\hat{D}_1$ such that $\hat{D} \xrightarrow{\tau} \hat{D}_1$ and $M' = (\nu x)(\hat{D}_1 \mid \overline{z}\langle y\rangle.(\hat{E} \mid \hat{F}))$. This way,
   $$K[M] \xrightarrow{\tau} (\nu \widetilde{u}, \widetilde{x})(K_\Gamma \mid K_\Delta \mid M') = K[M'] = M_1^\star$$
   We observe that $K[N]$ cannot match this action, but $K[N] \Longrightarrow K[N]$ is a valid weak transition. Hence, $N_1^\star = K[N]$. By subject reduction (Theorem 3.1), 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 $\widetilde{u}, \widetilde{x}$. 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 $\alpha = \overline{z\langle y\rangle}$. This action can only originate in M:
   $$K[M] \xrightarrow{\overline{z\langle y\rangle}} (\nu \widetilde{x}, \widetilde{u})(K_\Gamma \mid K_\Delta \mid (\nu x)(\hat{D} \mid (\nu y)(\hat{E} \mid \hat{F}))) = M_1^\star$$
   Process $K[N]$ can match this action via the following finite (weak) transition:
   $$K[N] \stackrel{\overline{z\langle y\rangle}}{\Longrightarrow} (\nu \widetilde{x}, \widetilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D}' \mid \hat{E}') \mid \hat{F}')) = N_1^\star$$

Observe how $N_1^\star$ reflects the changes in $K[N]$ due to the possible reductions before and after α. By definition of $\approx$ (output case), we consider the composition of $M_1^\star$ and $N_1^\star$ with any V such that $y{:}A \vdash V :: -{:}\mathbf{1}$. Using the typings in (C.1) and subject reduction (Theorem 3.1), we infer

$$\vdash M_2^\star = (\nu\widetilde{x},\widetilde{u})(K_\Gamma \mid K_\Delta \mid (\nu x)(\hat{D} \mid (\nu y)(\hat{E} \mid V \mid \hat{F}))) :: z{:}B$$
$$\vdash N_2^\star = (\nu\widetilde{x},\widetilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu y)((\nu x)(\hat{D'} \mid \hat{E'} \mid V) \mid \hat{F'})) :: z{:}B$$

Hence, the pair $(M_2^\star, N_2^\star)$ 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: $K[N] \xrightarrow{\alpha} N_1^\star$. We have to find a matching action α from $K[M]$: $K[M] \stackrel{\alpha}{\Longrightarrow} M_1^\star$. Similarly as before, there are two cases: either $\alpha = \tau$ or $\alpha = \overline{z\langle y\rangle}$. 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 $\alpha = \overline{z\langle y\rangle}$. We have:

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

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

$$K[M] \stackrel{\overline{z\langle y\rangle}}{\Longrightarrow} (\nu\widetilde{x},\widetilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu x)(\hat{D'} \mid (\nu y)(\hat{E'} \mid \hat{F'}))) = M_1^\star$$

where $M_1^\star$ takes into account the possible reductions before and after α. As before, we consider the composition of $N_1^\star$ and $M_1^\star$ with any V such that $y{:}A \vdash V :: -{:}\mathbf{1}$. Using (C.1), we can infer both

$$\vdash N_2^\star = (\nu\widetilde{x},\widetilde{u})(K_\Gamma \mid K_\Delta \mid (\nu y)((\nu x)(\hat{D} \mid \hat{E} \mid V) \mid \hat{F})) :: z{:}B$$
$$\vdash M_2^\star = (\nu\widetilde{x},\widetilde{u})(K'_\Gamma \mid K'_\Delta \mid (\nu x)(\hat{D'} \mid (\nu y)(\hat{E'} \mid V \mid \hat{F'}))) :: z{:}B$$

Hence, the pair $(N_2^\star, M_2^\star)$ 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 I-4 We then have that

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

with

$$\Gamma;\Delta_1 \vdash \hat{D} :: x{:}C \quad \Gamma;\Delta_2, x{:}C, z{:}A, y{:}B \vdash \hat{E} :: T \tag{C.2}$$

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. 6.3, 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 Def. 6.5, we can assume

$$K = (\nu \widetilde{u}, \widetilde{x}, y)(\bullet \mid K_\Gamma \mid K_\Delta \mid V)$$

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: $K[M] \xrightarrow{\alpha} M_1^\star$. We have to find a matching action α from $K[N]$, i.e., $K[N] \stackrel{\alpha}{\Longrightarrow} N_1^\star$. 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 corresponds to the composition of zero or more input-guarded replications which cannot evolve on their own.
   (b) $K_\Delta \xrightarrow{\tau} K'_\Delta$ and $K[M] \xrightarrow{\tau} (\nu \widetilde{u}, \widetilde{x}, y)(K_\Gamma \mid K'_\Delta \mid V \mid M) = M_1^\star$. Since K_Δ occurs identically in both processes, this reduction can be matched by $K[N]$ with a finite weak transition:

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

   Using subject reduction (Theorem 3.1) 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 $\vdash M_1^\star :: T$ and $\vdash N_1^\star :: T$ hold, and the pair $(M_1^\star, N_1^\star)$ is in $\mathcal{L}_T$ and so it is in $\mathcal{R}$.
   (c) $V \xrightarrow{\tau} V'$ and $K[M] \xrightarrow{\tau} (\nu \widetilde{u}, \widetilde{x}, y)(K_\Gamma \mid K_\Delta \mid V' \mid M) = M_1^\star$. This case proceeds similarly as the previous one, as V occurs in both processes.
   (d) $M \xrightarrow{\tau} M'$ and $K[M] \xrightarrow{\tau} (\nu \widetilde{u}, \widetilde{x})(K_\Gamma \mid K_\Delta \mid V \mid M') = M_1^\star$. 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,

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

   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 $N_1^\star = K[N]$. By subject reduction (Theorem 3.1), we infer that $\vdash K[M'] :: T$. Hence, the pair $(M_1^\star, N_1^\star)$ 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:

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

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

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

where $N_1^\star$ captures the fact that internal actions could have occurred before and after the synchronization on y. By subject reduction (Theorem 3.1), typing is preserved in both cases, and so $(M_1^\star, N_1^\star) \in \mathcal{R}$.

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 (C.2), 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 I-36 We then have that

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

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

with

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

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. 6.3, 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 Def. 6.5, we can assume

$$K = (\nu\widetilde{u},\widetilde{x},y)(\bullet \mid K_\Gamma \mid K_\Delta \mid V)$$

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: $K[M] \xrightarrow{\alpha} M_1^\star$. We have to find a matching action α from $K[N]$, i.e., $K[N] \stackrel{\alpha}{\Longrightarrow} N_1^\star$. The analysis is similar to the one detailed for the commuting conversion No. I-4. We consider two possible cases:

1. Case $\alpha = \tau$. We consider the possibilities for the origin of the reduction:
   (a) $K_\Gamma \longrightarrow K'_\Gamma$: This cannot be the case, as by construction this process corresponds to the composition of zero or more input guarded replications which cannot evolve on their own.
   (b) $M \longrightarrow M'$: 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) $K_\Delta \longrightarrow K'_\Delta$ and $K[M] \longrightarrow (\nu\widetilde{u},\widetilde{x})(K_\Gamma \mid K'_\Delta \mid V \mid M)$. 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 3.1), it can be shown that the derivatives are still in $\mathcal{R}$.
   (d) $V \longrightarrow V'$ and $K[M] \longrightarrow (\nu\widetilde{u},\widetilde{x})(K_\Gamma \mid K_\Delta \mid V' \mid M) = M_1^\star$. 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 $V \xrightarrow{\overline{y.\mathtt{inr}}} V'$:

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

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

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

   where $N_1^\star$ 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 3.1), and so $(M_1^\star, N_1^\star) \in \mathcal{R}$. The second subcase is when $S \xrightarrow{\overline{y.\mathtt{inl}}} S'$; 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 (C.3), 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 $K[N] \xrightarrow{\alpha} N_1^\star$ follows the same lines and is omitted.

□

Appendix C.2. Proof of Theorem 7.2

We repeat the statement of Theorem 7.2 and present its full proof.

Theorem Appendix C.2 (7.2)**.** *Let A, B be any type, as in Def 3.1. Then the following hold:*

(i)	$A \otimes B$	$\simeq$	$B \otimes A$
(ii)	$(A \oplus B) \multimap C$	$\simeq$	$(A \multimap C) \,\&\, (B \multimap C)$
(iii)	$!(A \,\&\, B)$	$\simeq$	$!A \otimes !B$

Proof. We detail the proof of (i). We verify conditions (i)-(iv) hold for processes $P^{\langle x,y\rangle}$, $Q^{\langle y,x\rangle}$ defined as

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

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

$$\dfrac{\dfrac{\dfrac{}{x{:}B \vdash [x \leftrightarrow n] :: n{:}B}\ (\mathsf{Tid}) \qquad \dfrac{}{u{:}A \vdash [u \leftrightarrow y] :: y{:}A}\ (\mathsf{Tid})}{u{:}A, x{:}B \vdash \overline{y}\langle n\rangle.([x \leftrightarrow n] \mid [u \leftrightarrow y]) :: y{:}B \otimes A}\ (\mathsf{T}\otimes\mathsf{R})}{x{:}A \otimes B \vdash x(u).\overline{y}\langle n\rangle.([x \leftrightarrow n] \mid [u \leftrightarrow y]) :: y{:}B \otimes A}\ (\mathsf{T}\otimes\mathsf{L})$$

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 $M = (\nu y)(P^{\langle x,y\rangle} \mid Q^{\langle y,z\rangle})$, $N = [x \leftrightarrow z]$; we need to show $\cdot\,; x{:}A \otimes B \vdash M \approx N :: z{:}A \otimes B$. By Prop. 6.3, 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 Def. 6.5, we can assume $K = (\nu x)(T^{\langle x\rangle} \mid [\cdot])$ where $\vdash T^{\langle x\rangle} :: x{:}A \otimes B$. By Lemma 3.2 and Theorem 5.1, there exist $l, T_1^{\langle x\rangle}$ such that $T^{\langle x\rangle} \overset{\overline{(\nu l)x\langle l\rangle}}{\Longrightarrow} T_1^{\langle x\rangle}$ in a finite transition.

Clearly, $(K[M], K[N]) \in \mathcal{R}$. Now, suppose $K[N] \xrightarrow{\alpha} N_1^\star$. We have to find a matching action α from $K[M]$, i.e., $K[N] \stackrel{\alpha}{\Longrightarrow} M_1^\star$. $K[N]$ has only an internal action, which leads to the renaming of $T^{\langle x\rangle}$: $K[N] \xrightarrow{\tau} T^{\langle z\rangle} = N_1^\star$. Using Theorem 5.1, $K[M]$ can match this action with a finite weak transition: $K[M] \Longrightarrow (\nu n)(T_1^{\langle n\rangle} \mid \overline{z}\langle m\rangle.([l \leftrightarrow m] \mid [n \leftrightarrow z])) = M_1^\star$. Using Theorem 3.1, we know that $(N_1^\star, M_1^\star) \in \mathcal{S}^{-1}$. Now suppose $N_1^\star \xrightarrow{\overline{z\langle l\rangle}} T_1^{\langle z\rangle}$; $M_1^\star$ can match this action with an output followed by a renaming: $M_1^\star \stackrel{\overline{z\langle m\rangle}}{\Longrightarrow} T_1^{\langle z\rangle} \mid [l \leftrightarrow m]$. By definition of $\approx$ (output clause), we take a process $S^{\langle c\rangle}$ such that $\cdot; c{:}A \vdash S^{\langle c\rangle} :: -{:}\mathbf{1}$, and compose it with $N_1^\star$ and $M_1^\star$. We thus obtain $N_2^\star = (\nu l)(T_1^{\langle z\rangle} \mid S^{\langle l\rangle})$ and $M_2^\star = (\nu m)(T_1^{\langle z\rangle} \mid [l \leftrightarrow m] \mid S^{\langle m\rangle})$; it can be easily checked that $(N_2^\star, M_2^\star) \in \mathcal{W}_{\vdash z:A\otimes B}$. When $K[M]$ moves first, the analysis is similar and we omit it.

The proof of *(ii)* uses processes

$$\begin{array}{rcll} \cdot; x{:}A \oplus B \multimap C & \vdash & P^{\langle x,y\rangle} :: (A \multimap C) \,\&\, (B \multimap C) & \\ \cdot; y{:}(A \multimap C) \,\&\, (B \multimap C) & \vdash & Q^{\langle y,x\rangle} :: x{:}A \oplus B \multimap C & \text{defined as:} \end{array}$$

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

The proof of *(iii)* uses processes

$$\begin{array}{rcll} \cdot; x{:}!(A \,\&\, B) & \vdash & P^{\langle x,y\rangle} :: y{:}!A \otimes !B & \\ \cdot; y{:}!A \otimes !B & \vdash & Q^{\langle y,x\rangle} :: x{:}!(A \,\&\, B) & \text{defined as:} \end{array}$$

$$\begin{array}{rll} P^{\langle x,y\rangle} = \overline{y}\langle n\rangle.(M \mid N) \text{ where} & M = \; !n(m).\overline{x}\langle l\rangle.l.\mathtt{inl}; [l \leftrightarrow m] \\ & N = \; !y(h).\overline{x}\langle k\rangle.k.\mathtt{inr}; [k \leftrightarrow h] \\ Q^{\langle y,x\rangle} = y(z).!x(n).n.\mathtt{case}(R, S) \text{ where} & R = \overline{z}\langle l\rangle.[l \leftrightarrow n] \\ & S = \overline{y}\langle k\rangle.[k \leftrightarrow n] \end{array}$$

□

Figure 8 Conditions for contextual type-respecting relations (cf. Def. 6.3)

A type-respecting relation $\mathcal{R}$ is contextual if

0. $\Gamma; \Delta \vdash P \mathcal{R} Q :: y{:}A$ implies $\Gamma; \Delta \vdash (\nu y)(P \mid [y \leftrightarrow z]) \mathcal{R} (\nu y)(Q \mid [y \leftrightarrow z]) :: z{:}A$, for any z such that $\Gamma; y{:}A \vdash [y \leftrightarrow z] :: z{:}A$
1. $\Gamma; \Delta, y{:}A \vdash P \mathcal{R} Q :: x{:}B$ implies $\Gamma; \Delta \vdash x(y).P \mathcal{R} x(y).Q :: x{:}A \multimap B$
2. $\Gamma; \Delta \vdash P \mathcal{R} Q :: y{:}A$ implies $\Gamma; \Delta, \Delta' \vdash \overline{x}\langle y\rangle.(P \mid S) \mathcal{R} a\overline{x}\langle y\rangle.(Q \mid S) :: x{:}A \otimes B$, for any x, S, B, Δ' such that $\Gamma; \Delta' \vdash S :: x{:}B$
3. $\Gamma; \Delta' \vdash P \mathcal{R} Q :: x{:}B$ implies $\Gamma; \Delta, \Delta' \vdash \overline{x}\langle y\rangle.(S \mid P) \mathcal{R} \overline{x}\langle y\rangle.(S \mid Q) :: x{:}A \otimes B$, for any y, S, A, Δ', such that $\Gamma; \Delta \vdash S :: y{:}A$
4. $\Gamma; \Delta \vdash P \mathcal{R} Q :: x{:}A$ implies $\Gamma; \Delta \vdash x.\mathtt{case}(P, S) \mathcal{R} x.\mathtt{case}(Q, S) :: x{:}A \,\&\, B$, for any S, B such that $\Gamma; \Delta \vdash S :: x{:}B$
5. $\Gamma; \Delta \vdash P \mathcal{R} Q :: x{:}B$ implies $\Gamma; \Delta \vdash x.\mathtt{case}(S, P) \mathcal{R} x.\mathtt{case}(S, Q) :: x{:}A \,\&\, B$, for any S, A such that $\Gamma; \Delta \vdash S :: x{:}A$
6. $\Gamma; \Delta \vdash P \mathcal{R} Q :: x{:}A$ implies $\Gamma; \Delta \vdash x.\mathsf{inl}; P \mathcal{R} x.\mathsf{inl}; Q :: x{:}A \oplus B$, for any B
7. $\Gamma; \Delta \vdash P \mathcal{R} Q :: x{:}B$ implies $\Gamma; \Delta \vdash x.\mathsf{inr}; P \mathcal{R} x.\mathsf{inr}; Q :: x{:}A \oplus B$, for any A
8. $\Gamma; \Delta \vdash P \mathcal{R} Q :: x{:}A$ implies $\Gamma; \Delta, \Delta' \vdash (\nu x)(P \mid S) \mathcal{R} (\nu x)(Q \mid S) :: T$, for any S, T, Δ' such that $\Gamma; \Delta', x{:}A \vdash S :: T$
9. $\Gamma; \Delta, x{:}A \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, \Delta' \vdash (\nu x)(S \mid P) \mathcal{R} (\nu x)(S \mid Q) :: T$, for any S, Δ' such that $\Gamma; \Delta' \vdash S :: x{:}A$
10. $\Gamma; \cdot \vdash P \mathcal{R} Q :: y{:}A$ implies $\Gamma; \Delta \vdash (\nu u)(!u(y).P \mid S) \mathcal{R} (\nu u)(!u(y).Q \mid S) :: T$, for any u, S, T, Δ such that $\Gamma, u{:}A; \Delta \vdash S :: T$
11. $\Gamma, u{:}A; \Delta \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta \vdash (\nu u)(!u(y).S \mid P) \mathcal{R} (\nu u)(!u(y).S \mid Q) :: T$, for any S, y such that $\Gamma; \cdot \vdash S :: y{:}A$
12. $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, \Delta' \vdash S \mid P \mathcal{R} S \mid Q :: T$, for any S, Δ' such that $\Gamma; \Delta' \vdash S :: -{:}\mathbf{1}$
13. $\Gamma; \Delta \vdash P \mathcal{R} Q :: -{:}\mathbf{1}$ implies $\Gamma; \Delta, \Delta' \vdash P \mid S \mathcal{R} Q \mid S :: T$, for any S, T, Δ' such that $\Gamma; \Delta' \vdash S :: T$
14. $\Gamma, u{:}A; \Delta \vdash P\{u/x\} \mathcal{R} Q\{u/x\} :: T$ implies $\Gamma; \Delta, x{:}!A \vdash P \mathcal{R} Q :: T$
15. $\Gamma; \cdot \vdash P \mathcal{R} Q :: y{:}A$ implies $\Gamma; \cdot \vdash !x(y).P \mathcal{R} !x(y).Q :: x{:}!A$, for any x
16. $\Gamma; \Delta, y{:}A, x{:}B \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x{:}A \otimes B \vdash x(y).P \mathcal{R} x(y).Q :: T$
17. $\Gamma; \Delta \vdash P \mathcal{R} Q :: y{:}A$ implies $\Gamma; \Delta, \Delta', x{:}A \multimap B \vdash \overline{x}\langle y\rangle.(P \mid S) \mathcal{R} \overline{x}\langle y\rangle.(Q \mid S) :: T$, for any x, B, S, T, Δ' such that $\Gamma; \Delta', x{:}B \vdash S :: T$
18. $\Gamma; \Delta, x{:}B \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, \Delta', x{:}A \multimap B \vdash \overline{x}\langle y\rangle.(P \mid S) \mathcal{R} \overline{x}\langle y\rangle.(Q \mid S) :: T$, for any y, A, S, Δ' such that $\Gamma; \Delta' \vdash S :: y{:}A$
19. $\Gamma, u{:}A; \Delta, y{:}A \vdash P \mathcal{R} Q :: T$ implies $\Gamma, u{:}A; \Delta \vdash \overline{u}\langle y\rangle.P \mathcal{R} \overline{u}\langle y\rangle.Q :: T$
20. $\Gamma; \Delta, x{:}A \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x{:}A \oplus B \vdash x.\mathtt{case}(P, S) \mathcal{R} x.\mathtt{case}(Q, S) :: T$, for any x, S, B such that $\Gamma; \Delta, x{:}B \vdash S :: T$
21. $\Gamma; \Delta, x{:}B \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x{:}A \oplus B \vdash x.\mathtt{case}(S, P) \mathcal{R} x.\mathtt{case}(S, Q) :: T$, for any A, S, T such that $\Gamma; \Delta, x{:}A \vdash S :: T$
22. $\Gamma; \Delta, x{:}A \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x{:}A \,\&\, B \vdash x.\mathsf{inl}; P \mathcal{R} x.\mathsf{inl}; Q :: T$
23. $\Gamma; \Delta, x{:}B \vdash P \mathcal{R} Q :: T$ implies $\Gamma; \Delta, x{:}A \,\&\, B \vdash x.\mathsf{inr}; P \mathcal{R} x.\mathsf{inr}; Q :: T$