Comparing Deadlock-Free Session Typed Processes

Ornela Dardha
University of Glasgow, United Kingdom

Jorge A. Pérez
University of Groningen, The Netherlands

Besides respecting prescribed protocols, communication-centric systems should never "get stuck". This important requirement has been expressed by liveness properties such as progress or (dead)lock-freedom. Several typing disciplines that ensure these properties for mobile processes have been proposed. Unfortunately, very little is known about the precise relationship between these disciplines– and the classes of typed processes they induce.

In this paper, we compare $\mathscr{L}$ and $\mathscr{K}$, two classes of deadlock-free, session typed concurrent processes. The class $\mathscr{L}$ stands out for its canonicity: it results naturally from interpretations of linear logic propositions as session types. The class $\mathscr{K}$, obtained by encoding session types into Kobayashi's usage types, includes processes not typable in other typing systems.

We show that $\mathscr{L}$ is strictly included in $\mathscr{K}$. We also identify the precise condition under which $\mathscr{L}$ and $\mathscr{K}$ coincide. One key observation is that the *degree of sharing* between parallel processes determines a new expressiveness hierarchy for typed processes. We also provide a type-preserving rewriting procedure of processes in $\mathscr{K}$ into processes in $\mathscr{L}$. This procedure suggests that, while effective, the degree of sharing is a rather subtle criteria for distinguishing typed processes.

1 Introduction

The goal of this work is to formally relate different type systems for the π-calculus. Our interest is in *session-based concurrency*, a type-based approach to communication correctness: dialogues between participants are structured into *sessions*, basic communication units; descriptions of interaction sequences are then abstracted as *session types* [11] which are checked against process specifications. We offer the first formal comparison between different type systems that enforce *(dead)lock-freedom*, the liveness property that ensures session communications never "get stuck". Our approach relates the classes of typed processes that such systems induce. To this end, we identify a property on the structure of typed parallel processes, the *degree of sharing*, which is key in distinguishing two salient classes of deadlock-free session processes, and in shedding light on their formal underpinnings.

In session-based concurrency, types enforce correct communications through different safety and liveness properties. Basic correctness properties are *communication safety* and *session fidelity*: while the former ensures absence of errors (e.g., communication mismatches), the latter ensures that well-typed processes respect the protocols prescribed by session types. Moreover, a central (liveness) property for safe processes is that they should never "get stuck". This is the well-known *progress* property, which asserts that a well-typed term either is a final value or can further reduce [17]. In calculi for concurrency, this property has been formalized as *deadlock-freedom* ("a process is deadlock-free if it can always reduce until it eventually terminates, unless the whole process diverges" [13]) or as *lock-freedom* ("a process is lock free if it can always reduce until it eventually terminates, even if the whole process diverges" [12]). Notice that in the absence of divergent behaviors, deadlock- and lock-freedom coincide.

(Dead)lock-freedom guarantees that all communications will eventually succeed, an appealing requirement for communicating processes. Several advanced type disciplines that ensure deadlock-free processes have been proposed (see, e.g., [2,4,8,12,13,15,20]). Unfortunately, these disciplines consider

Submitted to:
EXPRESS/SOS 2015

different process languages or are based in rather different principles. As a result, very little is known about how they relate to each other. This begs several research questions: What is the formal relationship between these type disciplines? What classes of deadlock-free processes do they induce?

In this paper, we tackle these open questions by comparing $\mathscr{L}$ and $\mathscr{K}$, two salient classes of deadlock-free, session typed processes (Definition 4.2):

- $\mathscr{L}$ contains all session processes that are well-typed according to the Curry-Howard correspondence of linear logic propositions as session types [2, 21]. This suffices, because the type system derived from such a correspondence ensures communication safety, session fidelity, and deadlock-freedom.
- $\mathscr{K}$ contains all session processes that enjoy safety and fidelity (as ensured by the system of Vasconcelos [19]) and are (dead)lock-free by combining Kobayashi's type system based on *usages* [12, 13] with Dardha et al.'s encodability result [7].

There are good reasons for considering $\mathscr{L}$ and $\mathscr{K}$. On the one hand, due to its deep logical foundations, $\mathscr{L}$ appears to us as the *canonic* class of deadlock-free session processes, upon which all other classes should be compared. Indeed, this class arguably offers the most principled yardstick for comparisons. On the other hand, $\mathscr{K}$ integrates session type checking with the sophisticated usage discipline developed by Kobayashi for π-calculus processes. This (indirect) approach to deadlock-freedom (first proposed in [14], later developed in [7]) is fairly general, as it may capture sessions with subtyping, polymorphism, and higher-order communication. Also, as informally shown in [3], $\mathscr{K}$ strictly includes classes of typed processes induced by other type systems for deadlock-freedom in sessions [4, 8, 15].

One key observation in our development is that $\mathscr{K}$ corresponds to a *family* of classes of deadlock-free processes, denoted $\mathscr{K}_0, \mathscr{K}_1, \cdots, \mathscr{K}_n$, which is defined by the *degree of sharing* between their parallel components. Intuitively, $\mathscr{K}_0$ is the sub-class of $\mathscr{K}$ with *independent parallel composition*: for all processes $P \mid Q \in \mathscr{K}_0$, subprocesses P and Q do not share any sessions. Then, $\mathscr{K}_1$ is the subclass of $\mathscr{K}$ which contains $\mathscr{K}_0$ but admits also processes with parallel components that share at most one session. Then, $\mathscr{K}_n$ contains deadlock-free session processes whose parallel components share at most n sessions.

Contributions. In this paper, we present three main contributions:

1. We show that the inclusion between the constituent classes of $\mathscr{K}$ is *strict* (Theorem 4.4). That is, we have:
$$\mathscr{K}_0 \subset \mathscr{K}_1 \subset \mathscr{K}_2 \subset \cdots \subset \mathscr{K}_n \subset \mathscr{K}_{n+1} \tag{1}$$
Although not extremely surprising, the significance of this result lies in the fact that it talks about concurrency (via the degree of sharing) but implicitly also about the potential sequentiality of parallel processes. As such, processes in $\mathscr{K}_k$ are necessarily "more parallel" than those in $\mathscr{K}_{k+1}$. Interestingly, the degree of sharing in $\mathscr{K}_0, \ldots, \mathscr{K}_n$ can be characterized in a very simple way, via a natural condition in the rule for parallel composition in Kobayashi's type system for deadlock-freedom.
2. We show that $\mathscr{L}$ and $\mathscr{K}_1$ coincide (Theorem 4.6). That is, there are deadlock-free session processes that cannot be typed by systems derived from the Curry-Howard interpretation of session types [2,21], but that can be admitted by the (indirect) approach of [7]. This result is significant: it establishes the precise status of systems based on [2, 21] with respect to previous (non Curry-Howard) disciplines. Indeed, it formally confirms that linear logic interpretations of session types naturally induce the most basic form of concurrent cooperation (sharing of exactly one session), embodied as the principle of "composition plus hiding", a distinguishing feature of such interpretations.
3. We define a rewriting procedure of processes in $\mathscr{K}$ into $\mathscr{L}$ (Defintion 5.7). Intuitively, due to our previous observation and characterization of the degree of sharing in session typed processes, it is

quite natural to convert a process in $\mathscr{K}$ into another, more parallel process in $\mathscr{L}$. In essence, the procedure replaces sequential prefixes with representative parallel components. The rewriting procedure satisfies type-preservation, and enjoys the compositionality and operational correspondence criteria as stated in [10] (cf. Theorems 5.8 and 5.10). These properties not only witness the significance of the rewriting procedure; they also confirm that the degree of sharing is a rather subtle criteria for formally distinguishing deadlock-free, session typed processes.

To the best of our knowledge, these contributions define the first formal comparison between fundamentally distinct type systems for deadlock-freedom in session communications. Previous comparisons, such as the ones in [3] and [2, §6], are informal: they are based on representative "corner cases", i.e., examples of deadlock-free session processes typable in one system but not in some other.

The paper is structured as follows. § 2 summarizes the framework of [19]: a session π-calculus and its associated type system. In § 3 we present the two typed approaches to deadlock-freedom for sessions. § 4 defines the classes $\mathscr{L}$ and $\mathscr{K}$, formalizes the hierarchy (1), and shows that $\mathscr{L}$ and $\mathscr{K}_1$ coincide. In § 5 we give the rewriting procedure of $\mathscr{K}_n$ into $\mathscr{L}$ and establish its properties. § 6 collects some concluding remarks. Omitted definitions and proofs are included in Appendix A.

2 Session π-calculus

In this section we introduce the session π-calculus and its associated type system which ensures communication safety and session fidelity. Our presentation follows that of Vasconcelos [19]. The syntax is given in Figure 1 (upper part). Let P, Q range over processes x, y over channels and v over values; for simplicity, the set of values coincides with that of channels. In examples, we often use $\mathbf{n}$ to denote a terminated channel that cannot be further used.

Process $\overline{x}\langle v\rangle.P$ denotes the output of value v along channel x, with continuation P. Dually, process $x(y).P$ denotes an input along channel x with continuation P, with y denoting a placeholder. Process $x \triangleleft l_j.P$ uses channel x to select l_j from a labelled choice process, being $x \triangleright \{l_i : P_i\}_{i\in I}$, so as to trigger P_j; labels indexed by the finite set I are pairwise distinct. We also consider the terminated process (denoted $\mathbf{0}$), the parallel composition of P and Q (denoted $P \mid Q$), and the (double) restriction operator, noted $(\nu xy)P$: the intention is that channels x and y denote *dual session endpoints* in P. We shall omit $\mathbf{0}$ whenever possible and write, e.g., $\overline{x}\langle\mathbf{n}\rangle$ instead of $\overline{x}\langle\mathbf{n}\rangle.\mathbf{0}$. Notions of bound/free variables in processes are standard; we write $\mathsf{fn}(P)$ to denote the set of free names of P. Also, we write $P[^{v}/_{z}]$ to denote the (capture-avoiding) substitution of free occurrences of z in P with v.

The operational semantics is given in terms of a reduction relation, noted $P \to Q$. This relation is defined by the rules in Figure 1 (lower part), and relies on a standard notion of structural congruence, noted $\equiv$ (see [19] for details). We write $\to^*$ to denote the reflexive, transitive closure of $\to$. Observe that interaction involves prefixes with different channels (endpoints), and always occurs in the context of an outermost restriction, which declares dual endpoints. Key rules are (R-Com) and (R-Case), denoting the interaction of output/input prefixes and selection/branching constructs, respectively. Rules (R-Par), (R-Res), and (R-Str) are standard.

The syntax of session types, ranged over $T, S, \ldots$, is given by the following grammar.[1]

$$T, S ::= \mathbf{end} \;\mid\; ?T.S \;\mid\; !T.S \;\mid\; \&\{l_i : S_i\}_{i\in I} \;\mid\; \oplus\{l_i : S_i\}_{i\in I}$$

Above, **end** is the type of an endpoint with a terminated protocol. The type $?T.S$ is assigned to an endpoint that first receives a value of type T and then continues according to the protocol described

[1] Notice that our session types correspond to the linear pretypes in [19].

$$
\begin{array}{rlll}
P,Q ::= & \overline{x}\langle v\rangle.P & \text{(output)} & \mathbf{0} \quad \text{(inaction)} \\
& x(y).P & \text{(input)} & P \mid Q \quad \text{(composition)} \\
& x \triangleleft l_j.P & \text{(selection)} & (\nu xy)P \quad \text{(session restriction)} \\
& x \triangleright \{l_i : P_i\}_{i\in I} & \text{(branching)} & \\
v ::= & x & \text{(channel)} &
\end{array}
$$

$$
\begin{array}{ll}
\text{(R-Com)} & (\nu xy)(\overline{x}\langle v\rangle.P \mid y(z).Q) \to (\nu xy)(P \mid Q[v/z]) \\
\text{(R-Case)} & (\nu xy)(x \triangleleft l_j.P \mid y \triangleright \{l_i : P_i\}_{i\in I}) \to (\nu xy)(P \mid P_j) \quad j \in I \\
\text{(R-Str)} & P \equiv P',\ P \to Q,\ Q' \equiv Q \Longrightarrow P' \to Q'
\end{array}
\qquad
\begin{array}{ll}
\text{(R-Par)} & P \to Q \Longrightarrow P \mid R \to Q \mid R \\
\text{(R-Res)} & P \to Q \Longrightarrow (\nu xy)P \to (\nu xy)Q
\end{array}
$$

Figure 1: Session π-calculus: syntax and semantics.

$$
\text{(T-Nil)}\ \frac{}{x : \mathbf{end} \vdash_{\mathrm{ST}} \mathbf{0}}
\qquad
\text{(T-Par)}\ \frac{\Gamma_1 \vdash_{\mathrm{ST}} P \quad \Gamma_2 \vdash_{\mathrm{ST}} Q}{\Gamma_1 \circ \Gamma_2 \vdash_{\mathrm{ST}} P \mid Q}
\qquad
\text{(T-Res)}\ \frac{\Gamma, x : T, y : \overline{T} \vdash_{\mathrm{ST}} P}{\Gamma \vdash_{\mathrm{ST}} (\nu xy)P}
\qquad
\text{(T-In)}\ \frac{\Gamma, x : S, y : T \vdash_{\mathrm{ST}} P}{\Gamma, x : ?T.S \vdash_{\mathrm{ST}} x(y).P}
$$

$$
\text{(T-Out)}\ \frac{\Gamma, x : S \vdash_{\mathrm{ST}} P}{\Gamma, x : !T.S, y : T \vdash_{\mathrm{ST}} \overline{x}\langle y\rangle.P}
\qquad
\text{(T-Brch)}\ \frac{\Gamma, x : S_i \vdash_{\mathrm{ST}} P_i \quad \forall i \in I}{\Gamma, x : \&\{l_i : S_i\}_{i\in I} \vdash_{\mathrm{ST}} x \triangleright \{l_i : P_i\}_{i\in I}}
\qquad
\text{(T-Sel)}\ \frac{\Gamma, x : S_j \vdash_{\mathrm{ST}} P \quad \exists j \in I}{\Gamma, x : \oplus\{l_i : S_i\}_{i\in I} \vdash_{\mathrm{ST}} x \triangleleft l_j.P}
$$

Figure 2: Typing rules for the π-calculus with sessions.

by S. Dually, type $!T.S$ is assigned to an endpoint that first outputs a value of type T and then continues according to the protocol described by S. Type $\oplus\{l_i : S_i\}_{i\in I}$, an *internal choice*, is a generalization of output; type $\&\{l_i : S_i\}_{i\in I}$, an *external choice*, is a generalization of input type. Notice that session types describe *sequences* of structured behaviors; they do not admit parallel composition operators.

A central notion in session-based concurrency is *duality*, which relates session types offering opposite (i.e., complementary) behaviors. Duality stands at the basis of communication safety and session fidelity. Given a session type T, its dual type $\overline{T}$ is defined as follows:

$$
\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
\overline{!T.S} & \triangleq & ?T.\overline{S} & \overline{?T.S} & \triangleq & !T.\overline{S} & \overline{\mathbf{end}} & \triangleq & \mathbf{end} \\
\overline{\oplus\{l_i : S_i\}_{i\in I}} & \triangleq & \&\{l_i : \overline{S}_i\}_{i\in I} & \overline{\&\{l_i : S_i\}_{i\in I}} & \triangleq & \oplus\{l_i : \overline{S}_i\}_{i\in I} & & &
\end{array}
$$

Typing contexts, ranged over by Γ, Γ', are sets of typing assignments $x : T$. Given a context Γ and a process P, a session typing judgement is of the form $\Gamma \vdash_{\mathrm{ST}} P$. Typing rules are given in Figure 2. Rule (T-Par) depends on a splitting operator, noted $\circ$, which performs a combination of session types [19]. Rule (T-Res) depends on duality of session types as defined above. Rule (T-Nil) states that the terminated process is well-typed under a terminated channel. Rule (T-Par) types the parallel composition of two processes under the composition of the corresponding typing contexts. Rule (T-Res) types the restriction process under the assumption that the endpoints being restricted have dual types. Rules (T-In) and (T-Out) type the receiving and sending of a value over a channel x, respectively. Finally, rules (T-Brch) and (T-Sel) are generalizations of input and output over a labelled set of processes.

The main guarantees of the typing system are *communication safety* and *session fidelity*, i.e., typed processes respect their ascribed protocols, as represented by session types.

Theorem 2.1 (Type Preservation for Session Types)**.** *If* $\Gamma \vdash_{\mathtt{ST}} P$ *and* $P \to Q$*, then* $\Gamma \vdash_{\mathtt{ST}} Q$*.*

The following notion of well-formed processes is key to single out meaningful typed processes.

Definition 2.2 (Well-Formedness for Sessions)**.** *A process is* well-formed *if for any of its structural congruent processes of the form* $(\nu\widetilde{xy})(P \mid Q)$ *the following hold.*

- *If* P *and* Q *are prefixed at the same variable, then the variable performs the same action (input or output, branching or selection).*
- *If* P *is prefixed in* x_i *and* Q *is prefixed in* y_i *where* $x_i y_i \in \widetilde{xy}$*, then* $P \mid Q \to$*.*

It is important to notice that well-typedness of a process does not imply the process is well-formed. We have the following theorem:

Theorem 2.3 (Type Safety for Sessions [19])**.** *If* $\vdash_{\mathtt{ST}} P$ *then* P *is well-formed.*

We present the main result of the session type system. The following theorem states that a well-typed closed process does not reduce to an ill-formed one. It follows immediately from Theorems 2.1 and 2.3.

Theorem 2.4 ([19])**.** *If* $\vdash_{\mathtt{ST}} P$ *and* $P \to^* Q$*, then* Q *is well-formed.*

An important observation is that the session type system given above does not automatically exclude processes which reach a "stuck state." This is because the interleaving of communication prefixes in typed processes may create extra causal dependencies not described by session types. (This can be made precise with the definition of deadlock-free session process that we give next.) A particularly insidious class of deadlocks is due to cyclic interleaving of channels in processes. For example, consider a process such as $P \triangleq (\nu xy)(\nu wz)(\overline{x}\langle\mathbf{n}\rangle.\overline{w}\langle\mathbf{n}\rangle \mid z(t).y(s))$: it represents the implementation of two (simple) independent sessions, which get intertwined (i.e., blocked) due to the nesting induced by input and output prefixes. We have that $\mathbf{n} : \mathbf{end} \vdash_{\mathtt{ST}} P$ even if P is unable to reduce. (A deadlock-free variant of P would be, e.g., process $P' \triangleq (\nu xy)(\nu wz)(\overline{x}\langle\mathbf{n}\rangle.\overline{w}\langle\mathbf{n}\rangle \mid y(s).z(t))$, typable in $\vdash_{\mathtt{ST}}$.)

We are now ready to give the formal definition of deadlock freedom for sessions. We will say that a process is *deadlock-free* if any communication action that becomes active during execution is eventually consumed. Below, we assume that reduction sequences are *fair*, as formalized in [12].

Definition 2.5 (Deadlock-Freedom for Session π-Calculus)**.** *A process* P_0 *is* deadlock-free *if for any fair reduction sequence* $P_0 \to P_1 \to P_2 \to \ldots$*, we have that*

1. $P_i \equiv (\nu\widetilde{xy})(\overline{x}\langle v\rangle.Q \mid R)$*, for* $i \geq 0$*, implies that there exists* $n \geq i$ *such that* $P_n \equiv (\nu\widetilde{x'y'})(\overline{x}\langle v\rangle.Q \mid y(z).R_1 \mid R_2)$ *and* $P_{n+1} \equiv (\nu\widetilde{x'y'})(Q \mid R_1[{}^{v}/_{z}] \mid R_2)$*;*
2. $P_i \equiv (\nu\widetilde{xy})(x \triangleleft l_j.Q \mid R)$*, for some* $i \geq 0$*, implies that there exists* $n \geq i$ *such that* $P_n \equiv (\nu\widetilde{x'y'})(x \triangleleft l_j.Q \mid y \triangleright \{l_k : R_k\}_{k \in I \cup \{j\}} \mid S)$ *and* $P_{n+1} \equiv (\nu\widetilde{x'y'})(Q \mid R_j \mid S)$*.*

For simplicity, above we have omitted the cases for input and branching, which have expected definitions.

3 Two Approaches to Deadlock Freedom

We introduce two approaches to deadlock-free, session typed processes. The first one, given in § 3.1, comes from interpretations of linear logic propositions as session types [1, 2, 21]; the second approach, summarized in § 3.2, combines usage types for the standard π-calculus with encodings of session processes and types [7]. Based on these two approaches, in § 4 we will define the classes $\mathscr{L}$ and $\mathscr{K}$.

3.1 Linear Logic Foundations for Session Types

The linear logic interpretation of session types was introduced by Caires and Pfenning [2], and developed by Wadler [21] and others. Initially proposed for intutitionistic linear logic, here we consider an interpretation based on classical linear logic with mix principles, following a recent presentation by Caires [1].

The syntax and semantics of processes are as in § 2 except for the following differences. First, we have the standard restriction operator $(\nu x)P$, which replaces the double restriction. Second, we have a so-called *forwarding process*, denoted $[x \leftrightarrow y]$, which intuitively "fuses" names x and y. Besides these differences in syntax, we have also some minor modifications in reduction rules. Differences with respect to the language considered in § 2 are summarized in the following:

$$P,Q ::= \quad (\nu x)P \quad \text{(channel restriction)} \quad \mid \quad [x \leftrightarrow y] \quad \text{(forwarding)}$$

$$\begin{array}{llll} \text{(R-ChCom)} & \overline{x}\langle v\rangle.P \mid x(z).Q \to P \mid Q[v/z] & \text{(R-Fwd)} & (\nu x)([x \leftrightarrow y] \mid P) \to P[y/x] \\ \text{(R-ChCase)} & x \triangleleft l_j.P \mid x \triangleright \{l_i : P_i\}_{i\in I} \to P \mid P_j \quad j \in I & \text{(R-ChRes)} & P \to Q \Longrightarrow (\nu x)P \to (\nu x)Q \end{array}$$

Observe how interaction of input/output prefixes and selection/branching is no longer covered by an outermost restriction. As for the typing system, we consider the so-called C-types which correspond to linear logic propositions. They are given by the following grammar:

$$A,B ::= \bot \;\mid\; \mathbf{1} \;\mid\; A \otimes B \;\mid\; A \parr B \;\mid\; \oplus\{l_i : A_i\}_{i\in I} \;\mid\; \&\{l_i : A_i\}_{i\in I}$$

Intuitively, $\bot$ and $\mathbf{1}$ are used to type a terminated endpoint. Type $A \otimes B$ is associated to an endpoint that first outputs an object of type A and then behaves according to B. Dually, type $A \parr B$ is the type of an endpoint that first inputs an object of type A and then continues as B. The interpretation of $\oplus\{l_i : A_i\}_{i\in I}$ and $\&\{l_i : A_i\}_{i\in I}$ as select and branch behaviors follows as expected.

We define a full duality on C-types, which exactly corresponds to the negation operator of CLL $(\cdot)^{\perp}$. The *dual* of type A, denoted $\overline{A}$, is inductively defined as follows:

$$\begin{array}{rclcrclcrcl} \overline{\mathbf{1}} & = & \bot & \qquad & \overline{\bot} & = & \mathbf{1} & \qquad & \overline{\oplus\{l_i : A_i\}_{i\in I}} & = & \&\{l_i : \overline{A}_i\}_{i\in I} \\ \overline{A \otimes B} & = & \overline{A} \parr \overline{B} & & \overline{A \parr B} & = & \overline{A} \otimes \overline{B} & & \overline{\&\{l_i : A_i\}_{i\in I}} & = & \oplus\{l_i : \overline{A}_i\}_{i\in I} \end{array}$$

Recall that $A \multimap B \triangleq \overline{A} \parr B$. As explained in [1], considering mix principles means admitting $\bot \multimap \mathbf{1}$ and $\mathbf{1} \multimap \bot$, and therefore $\bot = \mathbf{1}$. We write $\bullet$ to denote either $\bot$ or $\mathbf{1}$, and decree that $\overline{\bullet} = \bullet$.

Typing contexts, sets of typing assignments $x : A$, are ranged over $\Delta, \Delta', \ldots$. The empty context is denoted '$\cdot$'. Typing judgments are then of the form $P \vdash_{\mathtt{CH}} \Delta$. Figure 3 gives the typing rules associated to the linear logic interpretation. Salient points include the use of bound output $(\nu y)\overline{x}\langle y\rangle.P$, which is abbreviated as $\overline{x}(y)P$. Another highlight is the "composition plus hiding" principle implemented by rule (T-cut), which integrates parallel composition and restriction in a single rule. Indeed, there is no dedicated rule for restriction. Also, rule (T-mix) enables the typing of *independent parallel compositions*, i.e., the composition of two processes that do not share sessions.

We now collect main results for this typing system; see [1, 2] for details. For any P, define $live(P)$ if and only if $P \equiv (\nu \widetilde{n})(\pi.Q \mid R)$, where π is an input, output, selection, or branching prefix.

Theorem 3.1 (Type Preservation for C-Types)**.** *If $P \vdash_{\mathtt{CH}} \Delta$ and $P \longrightarrow Q$ then $Q \vdash_{\mathtt{CH}} \Delta$.*

Theorem 3.2 (Progress)**.** *If $P \vdash_{\mathtt{CH}} \cdot$ and $live(P)$ then $P \longrightarrow Q$, for some Q.*

$$\text{(T-}\mathbf{1}\text{)}\ \frac{}{\mathbf{0} \vdash_{\mathtt{CH}} x{:}\bullet} \qquad \text{(T-}\bot\text{)}\ \frac{P \vdash_{\mathtt{CH}} \Delta}{P \vdash_{\mathtt{CH}} x{:}\bullet, \Delta} \qquad \text{(T-id)}\ \frac{}{[x \leftrightarrow y] \vdash_{\mathtt{CH}} x{:}A, y{:}\overline{A}}$$

$$\text{(T-}\parr\text{)}\ \frac{P \vdash_{\mathtt{CH}} \Delta, y{:}A, x{:}B}{x(y).P \vdash_{\mathtt{CH}} \Delta, x{:}A \parr B} \qquad \text{(T-}\otimes\text{)}\ \frac{P \vdash_{\mathtt{CH}} \Delta, y{:}A \qquad Q \vdash_{\mathtt{CH}} \Delta', x{:}B}{\overline{x}(y).(P \mid Q) \vdash_{\mathtt{CH}} \Delta, \Delta', x{:}A \otimes B} \qquad \text{(T-cut)}\ \frac{P \vdash_{\mathtt{CH}} \Delta, x{:}\overline{A} \qquad Q \vdash_{\mathtt{CH}} \Delta', x{:}A}{(\nu x)(P \mid Q) \vdash_{\mathtt{CH}} \Delta, \Delta'}$$

$$\text{(T-}\oplus\text{)}\ \frac{P \vdash_{\mathtt{CH}} \Delta, x{:}A_j \qquad j \in I}{x \triangleleft l_j.P \vdash_{\mathtt{CH}} \Delta, x{:} \oplus \{l_i : A_i\}_{i \in I}} \qquad \text{(T-\&)}\ \frac{P_i \vdash_{\mathtt{CH}} \Delta, x{:}A_i \qquad \forall i \in I}{x \triangleright \{l_i : P_i\}_{i \in I} \vdash_{\mathtt{CH}} \Delta, x{:}\&\{l_i : A_i\}_{i \in I}} \qquad \text{(T-mix)}\ \frac{P \vdash_{\mathtt{CH}} \Delta \qquad Q \vdash_{\mathtt{CH}} \Delta'}{P \mid Q \vdash_{\mathtt{CH}} \Delta, \Delta'}$$

Figure 3: Typing rules for the π-calculus with C-types.

3.2 Deadlock Freedom by Encodability

As mentioned above, the second approach to deadlock-free session processes is *indirect*, in the sense that establishing deadlock freedom for session processes appeals to usage types for the π-calculus [12,13], for which type systems enforcing deadlock freedom are well-established. Formally, this reduction exploits encodings of processes and types: a session process $\Gamma \vdash_{\mathtt{ST}} P$ is encoded into a (standard) π-calculus process $[\![\Gamma]\!]_f \vdash^n_{\mathtt{KB}} [\![P]\!]_f$. Next we introduce the syntax of standard π-calculus processes with variant values (§ 3.2.1), the discipline of usage types (§ 3.2.2), and the encodings of session processes and types into standard π-calculus processes and usage types, respectively (§ 3.2.3).

3.2.1 Processes

The syntax and semantics of the π-calculus with usage types build upon those in § 2. We require some modifications. First, the encoding of terms presented in § 3.2.3, requires polyadic communication. Rather than branching and selection constructs, the π-calculus that we consider here includes a *case* construct $\mathbf{case}\, v\, \mathbf{of}\, \{l_i\_x_i \triangleright P_i\}_{i \in I}$ that uses *variant value* l_j_v. Moreover, we consider the standard channel restriction, rather than double restriction. These modifications are summarized below:

$$\begin{array}{rrll} P, Q ::= & (\nu x)P & \text{(channel restriction)} & \mid \quad \mathbf{case}\, v\, \mathbf{of}\, \{l_i\_x_i \triangleright P_i\}_{i \in I} \quad \text{(case)} \\ v ::= & l_j\_v & \text{(variant value)} & \end{array}$$

$$\text{(R}\pi\text{-Com)}\quad \overline{x}\langle\widetilde{v}\rangle.P \mid x(\widetilde{z}).Q \to P \mid Q[\widetilde{v}/\widetilde{z}] \qquad\qquad \text{(R}\pi\text{-Res)}\quad P \to Q \Longrightarrow (\nu x)P \to (\nu x)Q$$

$$\text{(R}\pi\text{-Case)}\quad \mathbf{case}\, l_j\_v\, \mathbf{of}\, \{l_i\_x_i \triangleright P_i\}_{i \in I} \to P_j[v/x_i] \quad j \in I$$

To conclude, we give the formal definition of deadlock freedom.

Definition 3.3 (Deadlock-Freedom for Standard π-Calculus)**.** *A process P_0 is* lock-free *under fair scheduling, if for any fair reduction sequence $P_0 \to P_1 \to P_2 \to \cdots$ the following hold*

1. *if $P_i \equiv (\nu\tilde{x})(\overline{x}\langle\widetilde{v}\rangle.Q \mid R)$ (for $i \geq 0$), implies that there exists $n \geq i$ such that $P_n \equiv (\nu\tilde{x})(\overline{x}\langle\widetilde{v}\rangle.Q \mid x(\widetilde{z}).R_1 \mid R_2)$ and $P_{n+1} \equiv (\nu\tilde{x})(Q \mid R_1[\widetilde{v}/\widetilde{z}] \mid R_2)$;*
2. *if $P_i \equiv (\nu\tilde{x})(x(\widetilde{z}).Q \mid R)$ for some $i \geq 0$, there exists $n \geq i$ such that $P_n \equiv (\nu\tilde{x})(x(\widetilde{z}).Q \mid \overline{x}\langle\widetilde{v}\rangle.R_1 \mid R_2)$ and $P_{n+1} \equiv (\nu\tilde{x})(Q[\widetilde{v}/\widetilde{z}] \mid R_1 \mid R_2)$.*

$$\begin{array}{rlll} U ::= & ?^{\mathrm{o}}_{\kappa}.U & \text{(used in input)} & \emptyset \quad \text{(not usable)} \\ & !^{\mathrm{o}}_{\kappa}.U & \text{(used in output)} & (U_1 \mid U_2) \quad \text{(used in parallel)} \\ T ::= & U[\widetilde{T}] & \text{(channel types)} & \langle l : T \rangle_{i \in I} \quad \text{(variant type)} \end{array}$$

Figure 4: Syntax of usage types for the π-calculus

3.2.2 Usage Types

The syntax of usage types is defined in Figure 4. For simplicity, we let α range over actions input ? or output !. The usage $\emptyset$ describes a channel that cannot be used at all. We will often omit $\emptyset$, and so we will write U instead of $U.\emptyset$. Usages $?^{\mathrm{o}}_{\kappa}.U$ and $!^{\mathrm{o}}_{\kappa}.U$ describe channels that can be used once for input and output, respectively and then used according to the continuation usage U. The *obligation* o and *capability* κ range over the set of natural numbers. The usage $U_1 \mid U_2$ describes a channel that is used according to U_1 by one process and U_2 by another processes in parallel.

Obligation and capability describe inter-channel dependencies. Following [13], their relation may be described as:

- An obligation of level n must be fulfilled by using only capabilities of level *less than* n. Said differently, an action of obligation n must be prefixed by actions of capabilities less than n.
- For an action with capability of level n, there must exist a co-action with obligation of level *less than or equal to* n.

Typing contexts are sets of typing assignments and are ranged over Γ, Γ'. A typing judgement is of the form $\Gamma \vdash^{n}_{\mathtt{KB}} P$: the annotation n explicitly denotes the greatest *degree of sharing* admitted in parallel processes. Before commenting on the typing rules (given in Figure 5), we discuss some important auxiliary notions. First, the composition operation on types (denoted $\mid$, and used in rules Tπ-(PAR)$_n$ and Tπ-(OUT)) is based on the composition of usages and is defined as follows:

$$\langle l_i : T_i \rangle_{i \in I} \mid \langle l_i : T_i \rangle_{i \in I} = \langle l_i : T_i \rangle_{i \in I} \qquad U_1[\widetilde{T}] \mid U_2[\widetilde{T}] = (U_1 \mid U_2)[\widetilde{T}]$$

The generalization of $\mid$ to typing contexts, denoted $(\Gamma_1 \mid \Gamma_2)(x)$, is defined as expected. The unary operation $\uparrow^t$ applied to usage U lifts its obligation level *up to* t; it is defined inductively as:

$$\uparrow^t \emptyset = \emptyset \qquad \uparrow^t \alpha^{\mathrm{o}}_{\kappa}.U = \alpha^{max(\mathrm{o},t)}_{\kappa}.U \qquad \uparrow^t (U_1 \mid U_2) = (\uparrow^t U_1 \mid \uparrow^t U_2)$$

The $\uparrow^t$ is extended to types and typing contexts as expected. The notion of *duality* on usage types (used in the proofs of our main results in Appendix A) is defined as follows. It simply exchanges ? and !:

$$\overline{\emptyset[]} = \emptyset[] \qquad \overline{?^{\mathrm{o}}_{\kappa}.U[\widetilde{T}]} = !^{\mathrm{o}}_{\kappa}.\overline{U}[\widetilde{T}] \qquad \overline{!^{\mathrm{o}}_{\kappa}.U[\widetilde{T}]} = ?^{\mathrm{o}}_{\kappa}.\overline{U}[\widetilde{T}]$$

Operator " ; " in $\Delta = x : [T]\alpha^{\mathrm{o}}_{\kappa}$; Γ, used in rules (Tπ-IN) and (Tπ-OUT), is such that the following hold:

$$\mathsf{dom}(\Delta) = \{x\} \cup \mathsf{dom}(\Gamma) \qquad \Delta(x) = \begin{cases} \alpha^{\mathrm{o}}_{\kappa}.U[\widetilde{T}] & \text{if } \Gamma(x) = U[\widetilde{T}] \\ \alpha^{\mathrm{o}}_{\kappa}[\widetilde{T}] & \text{if } x \notin dom(\Gamma) \end{cases} \qquad \Delta(y) = \uparrow^{\kappa+1} \Gamma(y) \quad \text{if } y \neq x$$

The final required notion is that of a *reliable usage*. It builds upon the following definition:

Definition 3.4. *Let U be a usage. The input and output* obligation levels *(resp.* capability levels*) of U, written $\mathsf{ob}_?(U)$ and $\mathsf{ob}_!(U)$ (resp. $\mathsf{cap}_?(U)$ and $\mathsf{cap}_!(U)$), are defined as:*

$$\begin{array}{rclcrcl} \mathsf{ob}_{\alpha}(\alpha^{\mathrm{o}}_{\kappa}.U) & = & \mathrm{o} & & \mathsf{cap}_{\alpha}(\alpha^{\mathrm{o}}_{\kappa}.U) & = & \kappa \\ \mathsf{ob}_{\alpha}(U_1 \mid U_2) & = & min(\mathsf{ob}_{\alpha}(U_1), \mathsf{ob}_{\alpha}(U_2)) & & \mathsf{cap}_{\alpha}(U_1 \mid U_2) & = & min(\mathsf{cap}_{\alpha}(U_1), \mathsf{cap}_{\alpha}(U_2)) \end{array}$$

$$\frac{}{x : \emptyset[] \vdash^n_{\mathtt{KB}} \mathbf{0}}\ (\text{T}\pi\text{-NIL}) \qquad \frac{\Gamma, x : U[\widetilde{T}] \vdash^n_{\mathtt{KB}} P \quad \mathsf{rel}(U)}{\Gamma \vdash^n_{\mathtt{KB}} (\nu x)P}\ (\text{T}\pi\text{-RES}) \qquad \frac{\Gamma_1 \vdash^n_{\mathtt{KB}} P \qquad \Gamma_2 \vdash^n_{\mathtt{KB}} Q \qquad |\Gamma_1 \cap \Gamma_1| \leq n}{\Gamma_1 \mid \Gamma_2 \vdash^n_{\mathtt{KB}} P \mid Q}\ (\text{T}\pi\text{-PAR}_n) \qquad \frac{\Gamma, \widetilde{y} : \widetilde{T} \vdash^n_{\mathtt{KB}} P}{x : ?^0_\kappa[\widetilde{T}]\ ; \Gamma \vdash^n_{\mathtt{KB}} x(\tilde{y}).P}\ (\text{T}\pi\text{-IN})$$

$$\frac{\Gamma_1 \vdash^n_{\mathtt{KB}} \tilde{v} : \widetilde{T} \qquad \Gamma_2 \vdash^n_{\mathtt{KB}} P}{x : !^0_\kappa[\widetilde{T}]\ ; (\Gamma_1 \mid \Gamma_2) \vdash^n_{\mathtt{KB}} \overline{x}\langle\tilde{v}\rangle.P}\ (\text{T}\pi\text{-OUT}) \qquad \frac{\Gamma \vdash^n_{\mathtt{KB}} v : T_j \quad \exists j \in I}{\Gamma \vdash^n_{\mathtt{KB}} l_j \_ v : \langle l_i : T_i\rangle_{i\in I}}\ (\text{T}\pi\text{-LVAL}) \qquad \frac{\Gamma_1 \vdash^n_{\mathtt{KB}} v : \langle l_i : T_i\rangle_{i\in I} \qquad \Gamma_2, x_i : T_i \vdash^n_{\mathtt{KB}} P_i \quad \forall i \in I}{\Gamma_1, \Gamma_2 \vdash^n_{\mathtt{KB}} \mathbf{case}\, v\, \mathbf{of}\, \{l_i \_ x_i \triangleright P_i\}_{i\in I}}\ (\text{T}\pi\text{-CASE})$$

Figure 5: Typing rules for the π-calculus with usage types with degree of sharing n.

The definition of reliable usages depends on a reduction relation on usages, noted $U \to U'$. Intuitively, $U \to U'$ means that if a channel of usage U is used for communication, then after the communication occurs, the channel should be used according to usage U'. Thus, e.g., $?^{\mathrm{o}}_\kappa.U_1 \mid ?^{\mathrm{o}'}_{\kappa'}.U_2$ reduces to $U_1 \mid U_2$.

Definition 3.5 (Reliability)**.** *We write* $\mathsf{con}_\alpha(U)$ *when* $\mathsf{ob}_{\overline{\alpha}}(U) \leq \mathsf{cap}_\alpha(U)$. *We write* $\mathsf{con}(U)$ *when* $\mathsf{con}_?(U)$ *and* $\mathsf{con}_!(U)$ *hold. Usage U is* reliable*, noted* $\mathsf{rel}(U)$*, if* $\mathsf{con}(U')$ *holds* $\forall U'$ *such that* $U \to^* U'$.

Typing Rules. The typing rules for the standard π-calculus with usage types are in Figure 5. The only difference with respect to the rules in Kobayashi's systems [12, 13] is that we annotate typing judgements with the degree of sharing, explicitly declared in rule (Tπ-PAR$_n$)–see below. Rule (Tπ-NIL) states that the terminated process is typed under a terminated channel. Rule (Tπ-RES) states that process $(\nu x)P$ is well-typed if the usage for x is reliable (cf. Definition 3.5). Rules (Tπ-IN) and (Tπ-OUT) type input and output processes in a typing context where the " ; " operator is used in order to increase the obligation level of the channels in continuation P. Rules (Tπ-LVAL) and (Tπ-CASE) type a choice: the first types a variant value with a variant type; the second types a case process using a variant value as its guard.

Given a degree of sharing n, rule (Tπ-PAR$_n$) states that the parallel composition of processes P and Q (typable under contexts Γ_1 and Γ_2, respectively) is well-typed under the typing context $\Gamma_1 \mid \Gamma_2$ only if $|\Gamma_1 \cap \Gamma_2| \leq n$. This allows to simply characterize the "concurrent cooperation" between P and Q. As a consequence, if $P \vdash^n_{\mathtt{KB}}$ then $P \vdash^k_{\mathtt{KB}}$, for any $k \leq n$. Observe that the typing rule for parallel composition in [12, 13] is the same as (Tπ-PAR$_n$), except for condition $|\Gamma_1 \cap \Gamma_2| \leq n$, which is not specified.

The next theorems imply that well-typed processes by the type system in Figure 5 are deadlock-free.

Theorem 3.6 (Type Preservation for Usage Types)**.** *If* $\Gamma \vdash^n_{\mathtt{KB}} P$ *and* $P \to Q$*, then* $\Gamma' \vdash^n_{\mathtt{KB}} Q$ *for some* Γ' *such that* $\Gamma \to \Gamma'$.

Theorem 3.7 (Deadlock-Freedom)**.** *If* $\emptyset \vdash^n_{\mathtt{KB}} P$*, then* $P \to Q$ *for some* Q.

Corollary 3.8. *If* $\emptyset \vdash^n_{\mathtt{KB}} P$*, then* P *is deadlock-free.*

3.2.3 Encodings of Processes and Types

Encoding of Terms. In order to relate classes of processes obtained by the different type systems presented so far, we need to rewrite a session typed or C-typed process into a usage typed process by adopting the continuation-passing style. This is because the only way we have to mimic the structure of a session type or a C-type is by sending its continuation as a payload over a channel. This idea is developed in [7] and before that in [14] and we recall it in Figure 6.

$$
\begin{array}{lcllcl}
[\![\overline{x}\langle v\rangle.P]\!]_f & \triangleq & (\nu c)\overline{f_x}\langle v,c\rangle.[\![P]\!]_{f,\{x\mapsto c\}} & [\![x\triangleright\{l_i:P_i\}_{i\in I}]\!]_f & \triangleq & f_x(y).\ \mathbf{case}\, y\, \mathbf{of}\, \{l_i\_c\triangleright [\![P_i]\!]_{f,\{x\mapsto c\}}\}_{i\in I} \\
[\![x(y).P]\!]_f & \triangleq & f_x(y,c).[\![P]\!]_{f,\{x\mapsto c\}} & [\![(\nu xy)P]\!]_f & \triangleq & (\nu c)[\![P]\!]_{f,\{x,y\mapsto c\}} \\
[\![x\triangleleft l_j.P]\!]_f & \triangleq & (\nu c)\overline{f_x}\langle l_j\_c\rangle.[\![P]\!]_{f,\{x\mapsto c\}} & [\![P\mid Q]\!]_f & \triangleq & [\![P]\!]_f\mid [\![Q]\!]_f
\end{array}
$$

Figure 6: Encoding of session processes into π-calculus processes.

$$
\begin{array}{rcl}
[\![\mathbf{end}]\!]_{\mathsf{su}} & = & \emptyset[] \\
[\![?T.S]\!]_{\mathsf{su}} & = & ?^{\mathrm{o}}_{\kappa}[[\![T]\!]_{\mathsf{su}},[\![S]\!]_{\mathsf{su}}] \\
[\![!T.S]\!]_{\mathsf{su}} & = & !^{\mathrm{o}}_{\kappa}[[\![T]\!]_{\mathsf{su}},[\![\overline{S}]\!]_{\mathsf{su}}] \\
[\![\&\{l_i:S_i\}_{i\in I}]\!]_{\mathsf{su}} & = & ?^{\mathrm{o}}_{\kappa}[\langle l_i:[\![S_i]\!]_{\mathsf{su}}\rangle_{i\in I}] \\
[\![\oplus\{l_i:S_i\}_{i\in I}]\!]_{\mathsf{su}} & = & !^{\mathrm{o}}_{\kappa}[\langle l_i:[\![\overline{S_i}]\!]_{\mathsf{su}}\rangle_{i\in I}]
\end{array}
\qquad
\begin{array}{rcl}
[\![\mathbf{end}]\!]_{\mathsf{c}} & = & \bullet \\
[\![?T.S]\!]_{\mathsf{c}} & = & [\![T]\!]_{\mathsf{c}}\parr[\![S]\!]_{\mathsf{c}} \\
[\![!T.S]\!]_{\mathsf{c}} & = & [\![\overline{T}]\!]_{\mathsf{c}}\otimes[\![S]\!]_{\mathsf{c}} \\
[\![\&\{l_i:S_i\}_{i\in I}]\!]_{\mathsf{c}} & = & \&\{l_i:[\![S_i]\!]_{\mathsf{c}}\}_{i\in I} \\
[\![\oplus\{l_i:S_i\}_{i\in I}]\!]_{\mathsf{c}} & = & \oplus\{l_i:[\![S_i]\!]_{\mathsf{c}}\}_{i\in I}
\end{array}
$$

Figure 7: Encodings of session types into usage types (Left) and C-types (Right).

Encoding of Types. We formally relate session types and logic propositions yo usage types by means of the encodings given in Figure 7. The former one, denoted as denoted $[\![\cdot]\!]_{\mathsf{su}}$, is taken from [7].

Definition 3.9. *Let Γ be a session typing context. The encoding $[\![\cdot]\!]_f$ into usage typing context and $[\![\cdot]\!]_{\mathsf{c}}$ into C-typing context is inductively defined as follows:*

$$[\![\emptyset]\!]_f = [\![\emptyset]\!]_{\mathsf{c}} \triangleq \emptyset \qquad [\![\Gamma,x:T]\!]_f \triangleq [\![\Gamma]\!]_f, f_x:[\![T]\!]_{\mathsf{su}} \qquad [\![\Gamma,x:T]\!]_{\mathsf{c}} \triangleq [\![\Gamma]\!]_{\mathsf{c}}, x:[\![T]\!]_{\mathsf{c}}$$

Lemma 3.10 (Duality and encoding of session types)**.** *Let T,S be finite session types. Then: (i) $\overline{T}=S$ if and only if $\overline{[\![T]\!]_{\mathsf{c}}}=[\![S]\!]_{\mathsf{c}}$; (ii) $\overline{T}=S$ if and only if $\overline{[\![T]\!]_{\mathsf{su}}}=[\![S]\!]_{\mathsf{su}}$.*

Proof. The proof is by induction on the duality relation of session types. □

On deadlock freedom by encoding. The next results relate the notions of deadlock freedom and typing and encoding.

Proposition 3.11. *Let P be a deadlock-free session process, then $[\![P]\!]_f$ is a deadlock-free π-process.*

Proof. The above corollary follows immediately by the encoding of terms given in Figure 6, Definition 2.5 and Definition 3.3. □

Next we recall an important result relating deadlock freedom and typing, by following [3].

Corollary 3.12. *Let P be a session process such that $\vdash_{\mathtt{ST}} P$. If $\vdash^{n}_{\mathtt{KB}} [\![P]\!]_f$ is deadlock-free then P is deadlock-free.*

4 A Hierarchy of Deadlock-Free Session Typed Processes

Preliminaries. To formally define the classes $\mathscr{L}$ and $\mathscr{K}$, we require some auxiliary definitions. We let $\overline{x}(y).P$ denote bound output. The following translation addresses minor syntactic differences between session typed processes (cf. § 2) and the processes typable in the linear logic interpretation of session types (cf. § 3.1). Such differences concern output actions and the restriction operator:

Definition 4.1. *Let P be a session process. The translation $\{\!\{\cdot\}\!\}$ is defined as*

$$\{\!\{\overline{x}\langle y\rangle.P\}\!\} \;=\; \overline{x}(z).([z\leftrightarrow y] \mid \{\!\{P\}\!\}) \qquad\qquad \{\!\{(\nu xy)P\}\!\} \;=\; (\nu w)\{\!\{P\}\!\}[w/x][w/y] \quad w \notin \mathsf{fn}(P)$$

and as an homomorphism for the other process constructs.

Let $[\![\cdot]\!]_{\mathsf{c}}$ denote the transition of session types into linear logic propositions in Figure 7 (right). Recall that $[\![\cdot]\!]_f$ stands for the translations for processes and $[\![\cdot]\!]_{\mathsf{su}}$ for the translation of types, both defined in [7], and given here in Figure 6 and Figure 7 (left), respectively. We may then formally define the languages under comparison:

Definition 4.2 (Typed Languages)**.** *The languages $\mathscr{L}$ and $\mathscr{K}_n$ ($n \geq 0$) are defined as follows:*

$$\begin{array}{rcl} \mathscr{L} & = & \{P \mid \exists \Gamma.\ (\Gamma \vdash_{\mathtt{ST}} P \wedge \{\!\{P\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}})\} \\ \mathscr{K}_n & = & \{P \mid \exists \Gamma, f.\ (\Gamma \vdash_{\mathtt{ST}} P \wedge [\![\Gamma]\!]_f \vdash^n_{\mathtt{KB}} [\![P]\!]_f)\} \end{array}$$

Main Results. Our first observation is that there are processes in $\mathscr{K}_2$ but not in $\mathscr{K}_1$:

Lemma 4.3. $\mathscr{K}_1 \subset \mathscr{K}_2$.

Proof. $\mathscr{K}_2$ contains (deadlock-free) session processes not captured in $\mathscr{K}_1$. A representative example is:

$$P_2 = (\nu a_1 b_1)(\nu a_2 b_2)(a_1(x).\,\overline{a_2}\langle x\rangle \mid \overline{b_1}\langle \mathbf{n}\rangle.\,b_2(z))$$

This process is not in $\mathscr{K}_1$ because it involves the composition of two parallel processes which share two sessions. As such, it is typable in $\vdash^n_{\mathtt{KB}}$ (with $n \geq 2$) but not in $\vdash^1_{\mathtt{KB}}$. □

The previous result generalizes easily, so as to define a hierarchy of deadlock-free, session processes:

Theorem 4.4. *For all $n \geq 1$, we have that $\mathscr{K}_n \subset \mathscr{K}_{n+1}$.*

Proof. Immediate by considering one of the following processes, which generalize process P_2 in Lemma 4.3:

$$\begin{array}{rcl} P_{n+1} & = & (\nu a_1 b_1)(\nu a_2 b_2)\cdots(\nu a_{n+1} b_{n+1})(a_1(x).\,\overline{a_2}\langle x\rangle.\cdots.\,\overline{a_{n+1}}\langle y\rangle \mid \overline{b_1}\langle \mathbf{n}\rangle.\,b_2(z).\,\cdots\,b_{n+1}(z)) \\ Q_{n+1} & = & (\nu a_1 b_1)(\nu a_2 b_2)\cdots(\nu a_{n+1} b_{n+1})(a_1(x).\,\overline{a_2}\langle x\rangle.\cdots.\,a_{n+1}(y) \mid \overline{b_1}\langle \mathbf{n}\rangle.\,b_2(z).\,\cdots\,\overline{b_{n+1}}\langle \mathbf{n}\rangle) \end{array}$$

To distinguish $\mathscr{K}_{n+1}$ from $\mathscr{K}_n$, we consider P_{n+1} if $n+1$ is even and Q_{n+1} otherwise. □

One main result of this paper is that $\mathscr{L}$ and $\mathscr{K}_1$ coincide. Before stating this result, a first observation we make is on the shape of session processes in $\mathscr{L}$. We have:

$$P ::= \mathbf{0} \;\mid\; [x\leftrightarrow y] \;\mid\; \overline{x}(y).P \;\mid\; x(y).P \;\mid\; x \triangleright \{l_i : P_i\}_{i\in I} \;\mid\; x \triangleleft l_j.P \;\mid\; P_1 \mid P_2 \;\mid\; (\nu xy)(P_1 \mid P_2)$$

Notice that typing for processes in $\mathscr{L}$ does not directly allow free output. Still, free output is representable (and typable) by linear logic types by means of the transformation in Definition 4.1. Thus, considered processes are not syntactically equal. In $\mathscr{L}$ there is cooperating composition (enabled by rule (T-cut) in Figure 3); independent composition can only be enabled by rule (T-mix). Arbitrary restriction is not allowed; the interpretation induces only restriction of parallel processes.

The following property is key in our developments: it connects our encodings of (dual) session types into usage types with reliability (Definition 3.5), a central notion to the type system for deadlock freedom. Recall that, unlike usage types, there is no parallel composition operator at the level of session types.

Proposition 4.5. *Let T be a session type. Then* $\mathsf{rel}(\llbracket T \rrbracket_{\mathsf{su}} \mid \llbracket \overline{T} \rrbracket_{\mathsf{su}})$ *holds.*

Proof (Sketch). By induction on the structure of session type T and the definitions of $\llbracket \cdot \rrbracket_{\mathsf{su}}$ and predicate $\mathsf{rel}(\cdot)$, using Lemma 3.10 (encodings of types preserve session type duality). See § A.1 for details. □

We then have the following main result, whose proof is detailed in § A.2:

Theorem 4.6. $\mathscr{L} = \mathscr{K}_1$.

Therefore, we have the following corollary, which attests that the class of deadlock-free session processes naturally induced by linear logic interpretations of session types are strictly included in the class induced by the indirect approach of Dardha et al. [7] (cf. § 3.2).

Corollary 4.7. $\mathscr{L} \subset \mathscr{K}_n$.

The fact that (deadlock-free) processes such as P_2 (cf. Lemma 4.3) are not in $\mathscr{L}$ is informally discussed in [2, §6]. However, [2] gives no formal comparisons with other classes of deadlock-free processes.

5 Rewriting $\mathscr{K}_n$ into $\mathscr{L}$

The hierarchy of deadlock-free session processes established by Theorem 4.4 is *subtle* in the following sense: if $P \in \mathscr{K}_{k+1}$ but $P \notin \mathscr{K}_k$ (with $k \geq 1$) then we know that there is a subprocess of P that needs to be "adjusted" in order to "fit in" $\mathscr{K}_k$. More precisely, we know that such a subprocess of P must become more parallel in order to be typable under the lesser degree of sharing k.

Here we propose a *rewriting procedure* that converts processes in $\mathscr{K}_n$ into processes in $\mathscr{K}_1$ (that is, $\mathscr{L}$, by Theorem 4.6). The rewriting procedure follows a simple idea: given a parallel process as input, return as output a process in which one of the components is kept unchanged, but the other is replaced by parallel representatives of the sessions implemented in it. Such parallel representatives are formally defined as characteristic processes and catalyzers, introduced next.

5.1 Preliminaries: Characteristic Processes and Catalyzers

Before presenting our rewriting procedure, let us first introduce some preliminary results.

Definition 5.1 (Characteristic Processes of a Session Type)**.** *Let T be a session type (cf. § 2). Given a name x, the set of* characteristic processes *of T, denoted $\{\!|T|\!\}^x$, is inductively defined as follows:*

$$\begin{array}{rcl}
\{\!|\mathbf{end}|\!\}^x & = & \{P \mid P \vdash_{\mathtt{CH}} x\!:\!\bullet\} \\
\{\!|?T.S|\!\}^x & = & \{x(y).P \mid P \vdash_{\mathtt{CH}} y\!:\!\llbracket T \rrbracket_{\mathsf{c}}, x\!:\!\llbracket S \rrbracket_{\mathsf{c}}\} \\
\{\!|!T.S|\!\}^x & = & \{\overline{x}(y).(P \mid Q) \mid P \in \{\!|\overline{T}|\!\}^y \wedge Q \in \{\!|S|\!\}^x\} \\
\{\!|\&\{l_i : S_i\}_{i\in I}|\!\}^x & = & \{x \triangleright \{l_i : P_i\}_{i\in I} \mid \forall i \in I.\ P_i \in \{\!|S_i|\!\}^x\} \\
\{\!|\oplus\{l_i : S_i\}_{i\in I}|\!\}^x & = & \bigcup_{i\in I} \{x \triangleleft l_i.P_i \mid P_i \in \{\!|S_i|\!\}^x\}
\end{array}$$

Definition 5.2 (Catalyser)**.** *Given a session type environment Γ, we define its associated* catalyser *as a process context $\mathscr{C}_\Gamma[\cdot]$, as follows:*

$$\mathscr{C}_\emptyset[\cdot] = [\cdot] \qquad \mathscr{C}_{\Gamma,x:T}[\cdot] = (\nu x)(\mathscr{C}_\Gamma[\cdot] \mid P) \quad \textit{with } P \in \{\!|\overline{T}|\!\}^x$$

We record the fact that characteristic processes are well-typed in the system of § 3.1:

Lemma 5.3. *Let T be a session type. For all $P \in \{\!|T|\!\}^x$, we have: $P \vdash_{\mathtt{CH}} x : \llbracket T \rrbracket_{\mathsf{c}}$*

Proof. See Appendix § A.3. □

We use $\{\!|T|\!\}^x \vdash_{\mathtt{CH}} x : [\![T]\!]_{\mathtt{c}}$ to denote the set of processes $P \in \{\!|T|\!\}^x$ such that $P \vdash_{\mathtt{CH}} x : [\![T]\!]_{\mathtt{c}}$.

Lemma 5.4 (Catalysers Preserve Typability)**.** *Let* $\Gamma \vdash_{\mathtt{ST}} P$ *and* $\Gamma' \subseteq \Gamma$. *Then* $\mathscr{C}_{\Gamma'}\big[P\big] \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}} \setminus [\![\Gamma']\!]_{\mathtt{c}}$.

Proof. Follows immediately by Definition 5.2. □

The following is a corollary of Lemma 5.4.

Corollary 5.5. *Let* $\Gamma \vdash_{\mathtt{ST}} P$. *Then* $\mathscr{C}_{\Gamma}\big[P\big] \vdash_{\mathtt{CH}} \emptyset$.

5.2 Rewriting $\mathscr{K}_n$ in $\mathscr{L}$

We start this section with some notations. First, in order to represent pseudo-non deterministic binary choices between two equally typed processes, we introduce the following notation.

Convention 5.6. *Let* P_1, P_2 *be two processes such that* $k \notin \mathsf{fn}(P_1,P_2)$. *We write* $P_1 \parallel_k P_2$ *to stand for the process* $(\nu k)(k \triangleleft \mathtt{inx}.\mathbf{0} \mid k \triangleright \{\mathtt{inl} : P_1, \mathtt{inr} : P_2\})$, *where label* `inx` *stands for either* `inl` *or* `inr`.

Clearly, since session execution is purely deterministic, our use of notation $P_1 \parallel_k P_2$ is merely syntactic sugar to denote the fact that either P_1 or P_2 will be executed (and that the actual deterministic choice is not relevant). It is worth adding that Caires [1] has already developed the technical machinery required to include non deterministic behavior into the linear logic interpretation of session types.

For syntactic convenience, we annotate bound names in processes with session types, and write $(\nu xy : T)P$ and $x(y : T).P$, for some session type T. When the reduction relation involves a left or right choice in a binary labelled choice, as in (administrative) reductions due to pseudo-non deterministic choices (Convention 5.6), we sometimes annotate the reduction as $\rightarrow^{\mathtt{inl}}$ or $\rightarrow^{\mathtt{inr}}$. We let $\mathtt{C}$ denote a *process context*, i.e., a process with a hole. And finally, for a typing context Γ, we shall write $\{\!|\Gamma|\!\}$ to denote the process $\prod_{(w_i:T_i)\in\Gamma} \{\!|T_i|\!\}^{w_i}$. We are now ready to give the rewriting procedure from $\mathscr{K}_n$ to $\mathscr{L}$.

Definition 5.7 (Rewriting $\mathscr{K}_n$ into $\mathscr{L}$)**.** *Let* $P \in \mathscr{K}_n$ *such that* $\Gamma \vdash_{\mathtt{ST}} P$, *for some* Γ. *The encoding* $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!)$ *is a process of* $\mathscr{L}$ *inductively defined as follows:*

$$
\begin{array}{rll}
(\!|x : \mathbf{end} \vdash_{\mathtt{ST}} \mathbf{0}|\!) \triangleq \mathbf{0} & & \\
(\!|\Gamma \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P'|\!) \triangleq \overline{x}(z).\big([v \leftrightarrow z] \mid (\!|\Gamma', x : S \vdash_{\mathtt{ST}} P'|\!)\big) & & \Gamma = \Gamma', x : !T.S, v : T \\
(\!|\Gamma \vdash_{\mathtt{ST}} x(y : T).P'|\!) \triangleq x(y).(\!|\Gamma', x : S, y : T \vdash_{\mathtt{ST}} P'|\!) & & \Gamma = \Gamma', x : ?T.S \\
(\!|\Gamma \vdash_{\mathtt{ST}} x \triangleleft l_j.P'|\!) \triangleq x \triangleleft l_j.(\!|\Gamma', x : S_j \vdash_{\mathtt{ST}} P'|\!) & & \Gamma = \Gamma', x : \oplus\{l_i : S_i\}_{i\in I} \\
(\!|\Gamma \vdash_{\mathtt{ST}} x \triangleright \{l_i : P_i\}_{i\in I}|\!) \triangleq x \triangleright \{l_i : (\!|\Gamma', x : S_i \vdash_{\mathtt{ST}} P_i|\!)\}_{i\in I} & & \Gamma = \Gamma', x : \&\{l_i : S_i\}_{i\in I} \\
(\!|\Gamma \vdash_{\mathtt{ST}} (\nu \widetilde{xy} : \widetilde{S})(P \mid Q)|\!) \triangleq \{\!|\Gamma_2|\!\} \mid \mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1, \widetilde{x}{:}\widetilde{S} \vdash_{\mathtt{ST}} P|\!)\big[\widetilde{z}/\widetilde{x}\big]\big] & & \Gamma = \Gamma_1 \circ \Gamma_2 \ \wedge \Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} P \\
\qquad \parallel_k \{\!|\Gamma_1|\!\} \mid \mathscr{C}_{\widetilde{z}:\widetilde{V}}\big[(\!|\Gamma_2, \widetilde{y}{:}\widetilde{V} \vdash_{\mathtt{ST}} Q|\!)\big[\widetilde{z}/\widetilde{y}\big]\big] & & \Gamma_2, \widetilde{y} : \widetilde{V} \vdash_{\mathtt{ST}} Q \wedge V_i = \overline{S_i}
\end{array}
$$

We illustrate the procedure in § A.6. Notice that the rewriting procedure given in Definition 5.7 satisfies the compositionality criteria given in [10]. In particular, it is easy to see that the rewriting of a composition of terms is defined in terms of the rewriting of the constituent subterms. Indeed, e.g., $(\!|\Gamma_1 \circ \Gamma_2 \vdash_{\mathtt{ST}} (\nu xy : S)(P \mid Q)|\!)$ depends on a context including both $(\!|\Gamma_1, x : S \vdash_{\mathtt{ST}} P|\!)$ and $(\!|\Gamma_2, y : \overline{S} \vdash_{\mathtt{ST}} Q|\!)$.

In the following we present two important results about our rewriting procedure. First, we show is type preserving—see § A.4 for a proof.

Theorem 5.8 (Rewriting is Type Preserving)**.** *Let* $(\Gamma \vdash_{\mathtt{ST}} P) \in \mathscr{K}_n$*. Then,* $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$.

Notice that the inverse of the previous theorem is trivial by following the definition of typed encoding. We now state the operational correspondence theorem, whose proof is in § A.5. Let us write $\Gamma \vdash_{\mathtt{ST}} P_1, P_2$ whenever both $\Gamma \vdash_{\mathtt{ST}} P_1$ and $\Gamma \vdash_{\mathtt{ST}} P_2$ hold.

Definition 5.9. *Let* P, P' *be such that* $\Gamma \vdash_{\mathtt{ST}} P, P'$*. Then, we write* $P \doteqdot P'$ *if and only if* $P = \mathtt{C}[Q]$ *and* $P' = \mathtt{C}[Q']$*, for some context* $\mathtt{C}$*, and there is* Γ' *such that* $\Gamma' \vdash_{\mathtt{ST}} Q, Q'$.

Theorem 5.10 (Operational Correspondence)**.** *Let* $P \in \mathscr{K}_n$ *such that* $\Gamma \vdash_{\mathtt{ST}} P$ *for some* Γ*. Then we have:*

I) If $P \to P'$ *then there exist* Q, Q' *s.t. (i)* $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \to^{\mathtt{inx}} \to^* \equiv Q$*; (ii)* $Q \doteqdot Q'$*; (iii)* $(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) \to^{\mathtt{inx}} Q'$.

II) If $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \to^{\mathtt{inx}} \to^* \equiv Q$ *then there exists* P' *s.t.* $P \to P'$ *and* $Q \doteqdot (\!|\Gamma \vdash_{\mathtt{ST}} P'|\!)$.

6 Concluding Remarks

We have presented a formal comparison of fundamentally distinct type systems for deadlock-free, session typed processes. To the best of our knowledge, ours is the first work to establish precise relationships of this kind. Indeed, prior comparisons between type systems for deadlock freedom are informal, given in terms of representative examples typable in one type system but not in some other one.

An immediate difficulty in giving a unified account of different typed frameworks for deadlock freedom is the variety of process languages, type structures, and typing rules that define each framework. Indeed, our comparisons involve: the framework of session processes put forward by Vasconcelos [19]; the interpretation of linear logic propositions as session types by Caires [1]; the π-calculus with usage types defined by Kobayashi in [12]. Finding some common ground for comparing these three frameworks is not trivial—several translations/transformations were required in our developments to account for numerous syntactic differences. Overall, we believe that we managed to concentrate on essential semantic features of two salient classes of deadlock-free session processes, noted $\mathscr{L}$ and $\mathscr{K}$.

Our main contribution is identifying the *degree of sharing* as a subtle, important issue that underlies both session typing and deadlock freedom. We propose a simple characterization of the degree of sharing: in essence, it arises via an explicit premise for the typing rule for parallel composition in the system in [12]. The degree of sharing is shown to effectively induce a strict hierarchy of deadlock-free session processes in $\mathscr{K}$, as resulting from the approach of [7]. We showed that the most elementary (and non trivial) member of this hierarchy precisely corresponds to $\mathscr{L}$–arguably the most canonical class of session typed processes known to date. Furthermore, by exhibiting an intuitive rewriting procedure of processes in $\mathscr{K}$ into processes in $\mathscr{L}$, we demonstrated that the degree of sharing is a subtle criteria for distinguishing deadlock-free processes. As such, even if our technical developments are technically simple, in our view they substantially clarify our understanding of typed systems for liveness properties (such as deadlock freedom) in the context of π-calculus processes.

One direction of future work is to obtain *semantic characterizations* of the degree of sharing, in the form of a preorder on typed processes that may distinguish when one process "is more parallel" than another. Previous work in this line includes that of Corradini et al. [5] who consider basic (untyped) CCS processes. Another direction is to extend our formal relationships to cover typing disciplines with *infinite behavior*. In session types, infinite behavior is specified via recursive processes and types. We notice that the approach of [7] extends to account for recursion [6] and that infinite (yet non divergent) behavior has been already incorporated into logic-based session types [18]. In the light of recent (non behavioral) type systems that admit expressive recursive structures (cf. [9, 16]), one interesting issue is to assess whether the expressiveness of such recent systems entails also additional expressiveness for (recursive)

session processes, following the indirect approach of [6,7]. Finally, we would like to explore whether the rewriting procedure given in § 5 could be adapted into a procedure for *deadlock resolution* that replaces overly sequential portions of a deadlocked process with (more parallel) characteristic processes.

References

[1] L. Caires. Types and logic, concurrency and non-determinism. In *Essays for the Luca Cardelli Fest* - Microsoft Research Technical Report MSR-TR-2014-104, September 2014.

[2] L. Caires, F. Pfenning, and B. Toninho. Linear logic propositions as session types. *MSCS*, 2014. Extended abstract in Proc. of CONCUR'10.

[3] M. Carbone, O. Dardha, and F. Montesi. Progress as compositional lock-freedom. In *COORDINATION*, volume 8459 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2014.

[4] M. Carbone and S. Debois. A graphical approach to progress for structured communication in web services. In *Proc. of ICE'10*, pages 13–27, 2010.

[5] F. Corradini, R. Gorrieri, and D. Marchignoli. Towards parallelization of concurrent systems. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 32(4-6):99–125, 1998.

[6] O. Dardha. Recursive session types revisited. In M. Carbone, editor, *Proceedings Third Workshop on Behavioural Types, BEAT 2014, Rome, Italy, 1st September 2014.*, volume 162 of *EPTCS*, pages 27–34, 2014.

[7] O. Dardha, E. Giachino, and D. Sangiorgi. Session types revisited. In *PPDP*, pages 139–150, 2012.

[8] M. Dezani-Ciancaglini, U. de'Liguoro, and N. Yoshida. On progress for structured communications. In *TGC*, volume 4912 of *LNCS*, pages 257–275. Springer, 2007.

[9] E. Giachino, N. Kobayashi, and C. Laneve. Deadlock analysis of unbounded process networks. In *CONCUR*, pages 63–77, 2014.

[10] D. Gorla. Towards a unified approach to encodability and separation results for process calculi. *Inf. Comput.*, 208(9):1031–1053, 2010.

[11] K. Honda, V. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138. Springer, 1998.

[12] N. Kobayashi. A type system for lock-free processes. *Inf. Comput.*, 177(2):122–159, 2002.

[13] N. Kobayashi. A new type system for deadlock-free processes. In *CONCUR*, pages 233–247, 2006.

[14] N. Kobayashi. Type systems for concurrent programs. Extended version, Tohoku University, 2007.

[15] L. Padovani. From lock freedom to progress using session types. In *PLACES*, pages 3–19, 2013.

[16] L. Padovani. Deadlock and Lock Freedom in the Linear π-Calculus. In *CSL-LICS*, pages 72:1–72:10. ACM, 2014.

[17] B. C. Pierce. *Types and programming languages*. MIT Press, MA, USA, 2002.

[18] B. Toninho, L. Caires, and F. Pfenning. Corecursion and non-divergence in session-typed processes. In *Proc. of TGC 2014*, volume 8902, pages 159–175. Springer, 2014.

[19] V. T. Vasconcelos. Fundamentals of session types. *Inf. Comput.*, 217:52–70, 2012.

[20] H. T. Vieira and V. T. Vasconcelos. Typing progress in communication-centred systems. In *Proc. of COORDINATION 2013*, volume 7890 of *Lecture Notes in Computer Science*, pages 236–250. Springer, 2013.

[21] P. Wadler. Propositions as sessions. In *ICFP*, pages 273–286, 2012.

A Appendix

A.1 Proof of Proposition 4.5

We repeat the statement in Page 12:

Proposition A.1. *Let T be a session type. Then* $\mathsf{rel}(\llbracket T \rrbracket_{\mathsf{su}} \mid \llbracket \overline{T} \rrbracket_{\mathsf{su}})$ *holds.*

Proof. By induction on the structure of session type T and the definitions of $\llbracket \cdot \rrbracket_{\mathsf{su}}$ and predicate $\mathsf{rel}(\cdot)$, using Lemma 3.10 (encodings of types preserve session type duality).

1. $T = \mathbf{end}$. Then $\llbracket T \rrbracket_{\mathsf{su}} \mid \llbracket \overline{T} \rrbracket_{\mathsf{su}} = (\emptyset \mid \emptyset)[]$ and the thesis follows easily.
2. $T = !T_1.T_2$ for some T_1, T_2. By definition of inductive duality, $\overline{T} = ?T_1.\overline{T_2}$. Then $\llbracket T \rrbracket_{\mathsf{su}} \mid \llbracket \overline{T} \rrbracket_{\mathsf{su}} = (?^{\mathrm{o}}_{\kappa}.\emptyset \mid !^{\mathrm{o}'}_{\kappa'}.\emptyset)[\llbracket T_1 \rrbracket_{\mathsf{su}}, \llbracket T_2 \rrbracket_{\mathsf{su}}]$. By letting $\mathrm{o} = \mathrm{o}' = 0$ and $\kappa = \kappa' = 1$ the $\mathsf{rel}(\cdot)$ predicate holds easily. The case of $T = ?T_1.T_2$, for some T_1, T_2 is similar to the above.
3. $T = \&\{l_i : S_i\}_{i \in I}$, for some S_i with $i \in I$. By definition of inductive duality, $\overline{T} = \oplus\{l_i : \overline{S_i}\}_{i \in I}$. Since $\llbracket T \rrbracket_{\mathsf{su}} = ?^{\mathrm{o}}_{\kappa}[\langle l_i : \llbracket S_i \rrbracket_{\mathsf{su}} \rangle_{i \in I}]$ and $\llbracket \overline{T} \rrbracket_{\mathsf{su}} = !^{\mathrm{o}'}_{\kappa'}[\langle l_i : \llbracket S_i \rrbracket_{\mathsf{su}} \rangle_{i \in I}]$, we have that $\llbracket T \rrbracket_{\mathsf{su}} \mid \llbracket \overline{T} \rrbracket_{\mathsf{su}} = (?^{\mathrm{o}}_{\kappa}.\emptyset \mid !^{\mathrm{o}'}_{\kappa'}.\emptyset)[\langle l_i : \llbracket S_i \rrbracket_{\mathsf{su}} \rangle_{i \in I}]$. By letting $\mathrm{o} = \mathrm{o}' = 0$ and $\kappa = \kappa' = 1$ the $\mathsf{rel}(\cdot)$ predicate holds easily. The case of $T = \oplus\{l_i : S_i\}_{i \in I}$,for some S_i with $i \in I$ is similar to the above.

□

A.2 Proof of Theorem 4.6

We divide the proof into two auxiliary lemmas.

Lemma A.2. *If $P \in \mathscr{L}$ then, $P \in \mathscr{K}_1$.*

Proof. By structural induction on P.

1. $P = \mathbf{0}$: Then, by assumption we have: $x : \mathbf{end} \vdash_{\mathtt{ST}} \mathbf{0}$ and $\mathbf{0} \vdash_{\mathtt{CH}} x{:}\bullet$, for some x. The thesis follows immediately by rule (Tπ-NIL) (Figure 5) noticing that $\llbracket \mathbf{end} \rrbracket_{\mathsf{su}} = \emptyset[]$.
2. $P = [x \leftrightarrow y]$: Then, by assumption we have $x : T, y : \overline{T} \vdash_{\mathtt{ST}} [x \leftrightarrow y]$ and $[x \leftrightarrow y] \vdash_{\mathtt{CH}} x{:}\llbracket T \rrbracket_{\mathsf{c}}, y{:}\llbracket \overline{T} \rrbracket_{\mathsf{c}}$, for some session type T. The thesis follows by using rule (Tπ-FWD) (Figure 5) and exploiting Lemma 3.10 (encodings of types preserve session type duality).
3. $P = x(y).P'$: Then, by assumption and Definitions 4.1 and 4.2, we have

$$\begin{array}{lll} \Gamma, x : ?T.S & \vdash_{\mathtt{ST}} & x(y).P' \\ x(y).\{\!|P'|\!\} & \vdash_{\mathtt{CH}} & \llbracket \Gamma \rrbracket_{\mathsf{c}}, x : \llbracket T \rrbracket_{\mathsf{c}} \parr \llbracket S \rrbracket_{\mathsf{c}} \end{array} \tag{2}$$

for some context Γ and session types S, T. Then, by inversion on typing on (2) we have:

$$\frac{\Gamma, x : S, y : T \vdash_{\mathtt{ST}} P'}{\Gamma, x : ?T.S \vdash_{\mathtt{ST}} x(y).P'}$$

By Definitions 4.2 and Figure 6, we must show:

$$f_x : ?^0_{\kappa}[\llbracket T \rrbracket_{\mathsf{su}}, \llbracket S \rrbracket_{\mathsf{su}}]; \llbracket \Gamma \rrbracket_f \vdash^1_{\mathtt{KB}} f_x(y,c).\llbracket P' \rrbracket_{f,\{x \mapsto c\}}$$

By induction hypothesis we have:

$$[\![\Gamma]\!]_{f'}, f'_x : [\![S]\!]_{\mathsf{su}}, f'_y : [\![T]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P']\!]_{f'}$$

Let $f = f' \setminus \{x \mapsto c, y \mapsto y\}$. We can re-write the above judgement as follows:

$$[\![\Gamma]\!]_f, c : [\![S]\!]_{\mathsf{su}}, y : [\![T]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P']\!]_{f,\{x\mapsto c\}} \tag{3}$$

Then, the thesis follows from (3) by using rule (Tπ-IN) given in Figure 5.

4. $P = \overline{x}\langle y\rangle.P'$: Then, by assumption and Definitions 4.1 and 4.2, we have:

$$\begin{array}{rcl} \Gamma, x : !T.S, y : T & \vdash_{\mathtt{ST}} & \overline{x}\langle y\rangle.P' \\ \overline{x}(z).([z\leftrightarrow y] \mid \{\!\{P'\}\!\}) & \vdash_{\mathtt{CH}} & [\![\Gamma]\!]_{\mathsf{c}}, x : [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}, y : [\![T]\!]_{\mathsf{c}} \end{array} \tag{4}$$

for some context Γ and session types S, T. By inversion on typing on (4) we have:

$$\frac{\Gamma, x : S \vdash_{\mathtt{ST}} P'}{\Gamma, x : !T.S, y : T \vdash_{\mathtt{ST}} \overline{x}\langle y\rangle.P'}$$

By Definitions 4.2 and Figure 6, we must show:

$$f_x : !^0_\kappa[[\![T]\!]_{\mathsf{su}}, [\![\overline{S}]\!]_{\mathsf{su}}] \,;\, \big(y : [\![T]\!]_{\mathsf{su}} \;\mid\; [\![\Gamma]\!]_f\big) \;\vdash^1_{\mathtt{KB}} (\nu c)\overline{f_x}\langle y, c\rangle.[\![P']\!]_{f,\{x\mapsto c\}}$$

By induction hypothesis we have:

$$[\![\Gamma]\!]_{f'}, f'_x : [\![S]\!]_{\mathsf{su}} \;\vdash^1_{\mathtt{KB}} [\![P']\!]_{f'}$$

Let $f = f' \setminus \{x \mapsto c\}$. We can re-write the above judgement as follows:

$$[\![\Gamma]\!]_f, c : [\![S]\!]_{\mathsf{su}} \;\vdash^1_{\mathtt{KB}} [\![P']\!]_{f,\{x\mapsto c\}} \tag{5}$$

By applying rule (Tπ-VAR) to derive $c : [\![\overline{S}]\!]_{\mathsf{su}}$ and $y : [\![T]\!]_{\mathsf{su}}$ and (Tπ-OUT) on (5) we have

$$\frac{y : [\![T]\!]_{\mathsf{su}} \;\vdash^1_{\mathtt{KB}} y : [\![T]\!]_{\mathsf{su}} \qquad c : [\![\overline{S}]\!]_{\mathsf{su}} \;\vdash^1_{\mathtt{KB}} c : [\![\overline{S}]\!]_{\mathsf{su}} \qquad [\![\Gamma]\!]_f, c : [\![S]\!]_{\mathsf{su}} \;\vdash^1_{\mathtt{KB}} [\![P']\!]_{f,\{x\mapsto c\}}}{f_x : !^0_\kappa[[\![T]\!]_{\mathsf{su}}, [\![\overline{S}]\!]_{\mathsf{su}}] \,;\, \big(y : [\![T]\!]_{\mathsf{su}} \mid c : [\![\overline{S}]\!]_{\mathsf{su}} \;\mid\; [\![\Gamma]\!]_f, c : [\![S]\!]_{\mathsf{su}}\big) \;\vdash^1_{\mathtt{KB}} \overline{f_x}\langle y, c\rangle.[\![P']\!]_{f,\{x\mapsto c\}}} \tag{6}$$

Last, by applying rule (Tπ-RES) and Proposition 4.5 on the conclusion of (6) we have:

$$\frac{f_x : !^0_\kappa[[\![T]\!]_{\mathsf{su}}, [\![\overline{S}]\!]_{\mathsf{su}}] \,;\, \big(y : [\![T]\!]_{\mathsf{su}} \mid c : [\![\overline{S}]\!]_{\mathsf{su}} \;\mid\; [\![\Gamma]\!]_f, c : [\![S]\!]_{\mathsf{su}}\big) \;\vdash^1_{\mathtt{KB}} \overline{f_x}\langle y, c\rangle.[\![P']\!]_{f,\{x\mapsto c\}} \qquad \mathsf{rel}([\![\overline{S}]\!]_{\mathsf{su}} \mid [\![S]\!]_{\mathsf{su}})}{f_x : !^0_\kappa[[\![T]\!]_{\mathsf{su}}, [\![\overline{S}]\!]_{\mathsf{su}}] \,;\, \big(y : [\![T]\!]_{\mathsf{su}} \;\mid\; [\![\Gamma]\!]_f\big) \;\vdash^1_{\mathtt{KB}} (\nu c)\overline{f_x}\langle y, c\rangle.[\![P']\!]_{f,\{x\mapsto c\}}}$$

This concludes the case.

5. $P = x \triangleright \{l_i : P_i\}_{i\in I}$. Then, by assumption and Definitions 4.1 and 4.2, we have:

$$\begin{array}{rcl} \Gamma, x : \&\{l_i : T_i\}_{i\in I} & \vdash_{\mathtt{ST}} & x \triangleright \{l_i : P_i\}_{i\in I} \\ x \triangleright \{l_i : \{\!\{P_i\}\!\}\}_{i\in I} & \vdash_{\mathtt{CH}} & [\![\Gamma]\!]_{\mathsf{c}}, x : \&\{l_i : [\![T_i]\!]_{\mathsf{c}}\}_{i\in I} \end{array} \tag{7}$$

for some context Γ and session types T_i for $i \in I$. By inversion on typing on (7) we have:

$$\frac{\Gamma, x : T_i \vdash_{\mathtt{ST}} P_i \quad \forall i \in I}{\Gamma, x : \&\{l_i : T_i\}_{i\in I} \vdash_{\mathtt{ST}} x \triangleright \{l_i : P_i\}_{i\in I}}$$

By Definitions 4.2 and Figure 6, we must show:

$$f_x : ?^0_\kappa[\langle l_i : [\![T_i]\!]_{\mathsf{su}}\rangle_{i\in I}] \; ; \; [\![\Gamma]\!]_f \vdash^1_{\mathtt{KB}} f_x(y).\, \mathbf{case}\, y\, \mathbf{of}\, \{l_i \_ c \triangleright [\![P_i]\!]_{f,\{x\mapsto c\}}\}_{i\in I}$$

By induction hypothesis we have:

$$[\![\Gamma]\!]_{f'}, f'_x : [\![T_i]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P_i]\!]_{f'} \quad \forall i \in I$$

Let $f = f' \setminus \{x \mapsto c\}$. We can re-write the above judgement as follows:

$$[\![\Gamma]\!]_f, c : [\![T_i]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P_i]\!]_{f,\{x\mapsto c\}} \quad \forall i \in I \tag{8}$$

By applying rules (Tπ-VAR) to obtain $y : \langle l_i : [\![T_i]\!]_{\mathsf{su}}\rangle_{i\in I}$ and (Tπ-CASE) and (Tπ-INP) we conclude:

$$f_x : ?^0_\kappa[\langle l_i : [\![T_i]\!]_{\mathsf{su}}\rangle_{i\in I}] \; ; \; [\![\Gamma]\!]_f \vdash^1_{\mathtt{KB}} f_x(y).\, \mathbf{case}\, y\, \mathbf{of}\, \{l_i \_ c \triangleright [\![P_i]\!]_{f,\{x\mapsto c\}}\}_{i\in I}$$

6. $P = x \triangleleft l_j P_j$. Then, by assumption and Definitions 4.1 and 4.2, we have:

$$\begin{array}{rcl} \Gamma, x : \oplus\{l_i : T_i\}_{i\in I} & \vdash_{\mathtt{ST}} & x \triangleleft l_j.P_j \\ x \triangleleft l_j.\{\!\{P_j\}\!\} & \vdash_{\mathtt{CH}} & [\![\Gamma]\!]_{\mathsf{c}}, x : \oplus\{l_i : [\![T_i]\!]_{\mathsf{c}}\}_{i\in I} \end{array} \tag{9}$$

for some context Γ and session types T_i for $i \in I$. By inversion on typing on (9) we have:

$$\frac{\Gamma, x : T_j \vdash_{\mathtt{ST}} P_j \quad \exists j \in I}{\Gamma, x : \oplus\{l_i : T_i\}_{i\in I} \vdash_{\mathtt{ST}} x \triangleleft l_j.P_j}$$

By Definitions 4.2 and Figure 6, we must show:

$$f_x : !^0_\kappa[\langle l_i : [\![\overline{T_i}]\!]_{\mathsf{su}}\rangle_{i\in I}] \; ; \; [\![\Gamma]\!]_f \vdash^1_{\mathtt{KB}} (\nu c)\overline{f_x}\langle l_j \_ c\rangle.[\![P_j]\!]_{f,\{x\mapsto c\}}$$

By induction hypothesis we have:

$$[\![\Gamma]\!]_{f'}, f'_x : [\![T_j]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P_j]\!]_{f'}$$

Let $f = f' \setminus \{x \mapsto c\}$. We can re-write the above judgement as follows:

$$[\![\Gamma]\!]_f, c : [\![T_j]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P_j]\!]_{f,\{x\mapsto c\}} \tag{10}$$

We use typing rules (Tπ-VAR) and (Tπ-LVAL) to obtain

$$c : [\![\overline{T_j}]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} l_j \_ c : \langle l_i : [\![\overline{T_i}]\!]_{\mathsf{su}}\rangle_{i\in I} \tag{11}$$

By applying rule (Tπ-OUT) on (10) and (11) we have

$$\frac{c : [\![\overline{T_j}]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} l_j \_ c : \langle l_i : [\![\overline{T_i}]\!]_{\mathsf{su}}\rangle_{i\in I} \qquad [\![\Gamma]\!]_f, c : [\![T_j]\!]_{\mathsf{su}} \vdash^1_{\mathtt{KB}} [\![P_j]\!]_{f,\{x\mapsto c\}}}{f_x : !^0_\kappa[\langle l_i : [\![\overline{T_i}]\!]_{\mathsf{su}}\rangle_{i\in I}] \; ; \; \left(c : [\![\overline{T_j}]\!]_{\mathsf{su}} \mid [\![\Gamma]\!]_f, c : [\![T_j]\!]_{\mathsf{su}}\right) \vdash^1_{\mathtt{KB}} \overline{f_x}\langle l_j \_ c\rangle.[\![P_j]\!]_{f,\{x\mapsto c\}}}$$

Last, by applying (Tπ-RES) on the conclusion above we have

$$\frac{f_x : !^0_\kappa[\langle l_i : [\![\overline{T_i}]\!]_{\mathsf{su}}\rangle_{i\in I}] \; ; \; \left(c : [\![\overline{T_j}]\!]_{\mathsf{su}} \mid [\![\Gamma]\!]_f, c : [\![T_j]\!]_{\mathsf{su}}\right) \vdash^1_{\mathtt{KB}} \overline{f_x}\langle l_j \_ c\rangle.[\![P_j]\!]_{f,\{x\mapsto c\}}}{f_x : !^0_\kappa[\langle l_i : [\![\overline{T_i}]\!]_{\mathsf{su}}\rangle_{i\in I}] \; ; \; [\![\Gamma]\!]_f \vdash^1_{\mathtt{KB}} (\nu c)\overline{f_x}\langle l_j \_ c\rangle.[\![P_j]\!]_{f,\{x\mapsto c\}}}$$

This concludes the case.

7. $P = (\nu xy)(P_1 \mid P_2)$: Then, by assumption and Definitions 4.1 and 4.2, we have:

$$\begin{array}{rcl} \Gamma_1 \circ \Gamma_2 & \vdash_{\mathtt{ST}} & (\nu xy)(P_1 \mid P_2) \\ (\nu w)(\{\!\{P_1\}\!\}[w/x] \mid \{\!\{P_2\}\!\}[w/y]) & \vdash_{\mathtt{CH}} & [\![\Gamma_1]\!]_{\mathsf{c}}, [\![\Gamma_2]\!]_{\mathsf{c}} \end{array} \tag{12}$$

for some contexts Γ_1, Γ_2. By inversion on typing on (12) we have:

$$\dfrac{\dfrac{\Gamma_1, x : T \vdash_{\mathtt{ST}} P_1 \quad \Gamma_2, y : \overline{T} \vdash_{\mathtt{ST}} P_2}{\Gamma_1, x : T \circ \Gamma_2, y : \overline{T} \vdash_{\mathtt{ST}} P_1 \mid P_2}}{\Gamma_1 \circ \Gamma_2 \vdash_{\mathtt{ST}} (\nu xy)(P_1 \mid P_2)}$$

for some session type T.

Notice that, since $P \in \mathscr{L}$, then channel xy – or w – is the only one shared between processes P_1 and P_2. This implies that Γ_1 and Γ_2 are disjoint. Hence, by the definition of $\circ$ we can replace $\Gamma_1 \circ \Gamma_2$ with simply Γ_1, Γ_2. By Definitions 4.2 and Figure 6, we must show:

$$[\![\Gamma_1]\!]_f, [\![\Gamma_2]\!]_f \ \vdash^1_{\mathtt{KB}} (\nu w)([\![P_1]\!]_{f,\{x \mapsto w\}} \mid [\![P_2]\!]_{f,\{y \mapsto w\}})$$

By induction hypothesis we have:

$$[\![\Gamma_1]\!]_{f'}, f'_x : [\![T]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_1]\!]_{f'} \quad \text{and} \quad [\![\Gamma_2]\!]_{f'}, f'_y : [\![\overline{T}]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_2]\!]_{f'}$$

Let $f = f' \setminus \{x, y \mapsto w\}$, hence we have $f'_x = f'_y = w$. We can now re-write the above judgements as follows:

$$[\![\Gamma_1]\!]_f, w : [\![T]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_1]\!]_{f,\{x \mapsto w\}} \quad \text{and} \quad [\![\Gamma_2]\!]_f, w : [\![\overline{T}]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_2]\!]_{f,\{y \mapsto w\}} \tag{13}$$

By applying rule ($\mathrm{T}\pi$-PAR_1) on the conclusion of (13) we have the following derivation:

$$\dfrac{[\![\Gamma_1]\!]_f, w : [\![T]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_1]\!]_{f,\{x \mapsto w\}} \qquad [\![\Gamma_2]\!]_f, w : [\![\overline{T}]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_2]\!]_{f,\{y \mapsto w\}}}{[\![\Gamma_1]\!]_f, [\![\Gamma_2]\!]_f, w : [\![T]\!]_{\mathsf{su}} \mid [\![\overline{T}]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_1]\!]_{f,\{x \mapsto w\}} \mid [\![P_2]\!]_{f,\{y \mapsto w\}}}$$

Since the only channel that processes P_1 and P_2 share is w, it is sufficient to write the composition of typing contexts $[\![\Gamma_1]\!]_f$ and $[\![\Gamma_2]\!]_f$ with ',' instead of parallel composition; we put $\mid$ in the composition of the types for channel w, namely $[\![T]\!]_{\mathsf{su}} \mid [\![\overline{T}]\!]_{\mathsf{su}}$.

By applying rule ($\mathrm{T}\pi$-RES) and by using Proposition 4.5, to show that session type duality implies, through encodings, the reliability predicate $\mathsf{rel}(\cdot)$ we have the following derivation:

$$\dfrac{[\![\Gamma_1]\!]_f, [\![\Gamma_2]\!]_f, w : [\![T]\!]_{\mathsf{su}} \mid [\![\overline{T}]\!]_{\mathsf{su}} \ \vdash^1_{\mathtt{KB}} [\![P_1]\!]_{f,\{x \mapsto w\}} \mid [\![P_2]\!]_{f,\{y \mapsto w\}} \qquad \mathsf{rel}([\![T]\!]_{\mathsf{su}} \mid [\![\overline{T}]\!]_{\mathsf{su}})}{[\![\Gamma_1]\!]_f, [\![\Gamma_2]\!]_f \vdash^1_{\mathtt{KB}} (\nu w)([\![P_1]\!]_{f,\{x \mapsto w\}} \mid [\![P_2]\!]_{f,\{y \mapsto w\}})}$$

8. $P = P_1 \mid P_2$: This case is similar as the previous one. By assumption we have

$$\begin{array}{rcl} \Gamma_1 \circ \Gamma_2 & \vdash_{\mathtt{ST}} & P_1 \mid P_2 \\ \{\!\{P_1\}\!\}, \{\!\{P_2\}\!\} & \vdash_{\mathtt{CH}} & [\![\Gamma_1]\!]_{\mathsf{c}}, [\![\Gamma_2]\!]_{\mathsf{c}} \end{array} \tag{14}$$

and inversion on typing on (14) we infer:

$$\dfrac{\Gamma_1 \vdash_{\mathtt{ST}} P_1 \quad \Gamma_2 \vdash_{\mathtt{ST}} P_2}{\Gamma_1 \circ \Gamma_2 \vdash_{\mathtt{ST}} P_1 \mid P_2}$$

By Definitions 4.2 and Figure 6, we must show:

$$[\![\Gamma_1]\!]_f, [\![\Gamma_2]\!]_f \vdash^1_{\mathtt{KB}} [\![P_1]\!]_f \mid [\![P_2]\!]_f$$

Since by induction hypothesis we have both:

$$[\![\Gamma_1]\!]_f \vdash^1_{\mathtt{KB}} [\![P_1]\!]_f \quad \text{and} \quad [\![\Gamma_2]\!]_f \vdash^1_{\mathtt{KB}} [\![P_2]\!]_f$$

the thesis proceeds easily by using rule Tπ-PAR$_1$ (Figure 5).

□

Lemma A.3. *If $P \in \mathscr{K}_1$ then $P \in \mathscr{L}$.*

Proof (Sketch). By structural induction on P. We briefly comment on the several cases:

1. $P = \mathbf{0}$: Straightforward.
2. $P = [x \leftrightarrow y]$: Straightforward.
3. $P = x(y).P'$: Then by assumption and Definitions 6 and 4.2, we have

$$\begin{array}{rcl} \Gamma, x : ?T.S & \vdash_{\mathtt{ST}} & x(y).P' \\ [\![\Gamma]\!]_f, f_x : ?^0_\kappa[[\![T]\!]_{\mathsf{su}}, [\![S]\!]_{\mathsf{su}}] & \vdash^1_{\mathtt{KB}} & f_x(y,c).[\![P']\!]_{f,\{x \mapsto c\}} \end{array} \tag{15}$$

for some context Γ and session types S and T. By inversion on typing on 15 we have:

$$\frac{\Gamma, x : S, y : T \vdash_{\mathtt{ST}} P'}{\Gamma, x : ?T.S \vdash_{\mathtt{ST}} x(y).P'}$$

We must show:

$$x(y).\{\!\{P'\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}}, x : [\![T]\!]_{\mathsf{c}} \parr [\![S]\!]_{\mathsf{c}}$$

By induction hypothesis we have:

$$\{\!\{P'\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}}, y : [\![T]\!]_{\mathsf{c}} \tag{16}$$

and therefore the thesis follows easily from (16), using rule (T$\parr$) (Figure 3).

4. $P = \overline{x}\langle y \rangle.P'$: Then by assumption and Definitions 6 and 4.2, we have

$$\begin{array}{rcl} \Gamma, x : !T.S, y : T & \vdash_{\mathtt{ST}} & \overline{x}\langle y \rangle.P' \\ [\![\Gamma]\!]_f, f_x : !^0_\kappa[[\![T]\!]_{\mathsf{su}}, [\![S]\!]_{\mathsf{su}}] & \vdash^1_{\mathtt{KB}} & (\nu c) f_x\langle y, c \rangle.[\![P']\!]_{f,\{x \mapsto c\}} \end{array} \tag{17}$$

for some context Γ and session types S and T. By inversion on typing on (18) we have:

$$\frac{\Gamma, x : S \vdash_{\mathtt{ST}} P'}{\Gamma, x : !T.S, y : T \vdash_{\mathtt{ST}} \overline{x}\langle y \rangle.P'}$$

By induction hypothesis we have:

$$\{\!\{P'\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}}$$

and then we can easily infer:

$$\frac{\{\!\{P'\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}} \qquad [y \leftrightarrow z] \vdash_{\mathtt{CH}} y : [\![T]\!]_{\mathsf{c}}, z : [\![\overline{T}]\!]_{\mathsf{c}}}{\overline{x}(z).([z \leftrightarrow y] \mid \{\!\{P'\}\!\}) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}}, x : [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}, y : [\![T]\!]_{\mathsf{c}}}$$

and therefore the thesis follows.

5. $P = (\nu xy)(P_1 \mid P_2)$: Then by assumption and Definitions 6 and 4.2, we have:

$$\begin{array}{rcl} \Gamma_1 \circ \Gamma_2 & \vdash_{\mathtt{ST}} & (\nu xy)(P_1 \mid P_2) \\ [\![\Gamma_1]\!]_f, [\![\Gamma_2]\!]_f & \vdash^1_{\mathtt{KB}} & (\nu w)([\![P_1]\!]_f[w/x] \mid [\![P_2]\!]_f[w/y]) \end{array} \tag{18}$$

for some contexts Γ_1, Γ_2. By inversion on typing on (18) we have:

$$\dfrac{\dfrac{\Gamma_1, x : S \vdash_{\mathtt{ST}} P_1 \quad \Gamma_2, y : \overline{S} \vdash_{\mathtt{ST}} P_2}{\Gamma_1, x : S \circ \Gamma_2, y : \overline{S} \vdash_{\mathtt{ST}} P_1 \mid P_2}}{\Gamma_1 \circ \Gamma_2 \vdash_{\mathtt{ST}} (\nu xy)(P_1 \mid P_2)}$$

for session type S. By induction hypothesis we have both

$$\{\!\{P_1\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}} \quad \text{and} \quad \{\!\{P_2\}\!\} \vdash_{\mathtt{CH}} [\![\Gamma_2]\!]_{\mathsf{c}}, y : [\![\overline{S}]\!]_{\mathsf{c}}$$

and the thesis follows easily from Lemma 3.10 and rule (Tcut).

6. $P = P_1 \mid P_2$: Similar to the previous case, but using rule (Tmix) (rather than on (Tcut)) for composition (see Figure 3).

□

A.3 Proof of Lemma 5.3

We repeat the statement in Page 12:

Lemma A.4. *Let T be a session type. For all $P \in \{\!|T|\!\}^x$, we have: $P \vdash_{\mathtt{CH}} x : [\![T]\!]_{\mathsf{c}}$*

Proof. By induction on the structure of session type T.

1. Case $\{\!|\mathbf{end}|\!\}^x = \{P \mid P \vdash_{\mathtt{CH}} x{:}\bullet\}$. We conclude since $[\![\mathbf{end}]\!]_{\mathsf{c}} = \bullet$.
2. Case $\{\!|?T.S|\!\}^x = \{x(y).P \mid P \vdash_{\mathtt{CH}} y{:}[\![T]\!]_{\mathsf{c}}, x{:}[\![S]\!]_{\mathsf{c}}\}$. Let $x(y).P \in \{\!|?T.S|\!\}^x$, then by rule (T⅋) we have

$$\dfrac{P \vdash_{\mathtt{CH}} y{:}[\![T]\!]_{\mathsf{c}}, x{:}[\![S]\!]_{\mathsf{c}}}{x(y).P \vdash_{\mathtt{CH}} x{:}[\![T]\!]_{\mathsf{c}} \parr [\![S]\!]_{\mathsf{c}}}$$

   By the encoding of types in Figure 7, we have $[\![?T.S]\!]_{\mathsf{c}} = [\![T]\!]_{\mathsf{c}} \parr [\![S]\!]_{\mathsf{c}}$, which concludes the case.
3. Case $\{\!|!T.S|\!\}^x = \{\overline{x}(y).(P \mid Q) \mid P \in \{\!|\overline{T}|\!\}^y \wedge Q \in \{\!|S|\!\}^x\}$. By rule (T⊗) and by induction hypothesis we have

$$\dfrac{P \vdash_{\mathtt{CH}} y : [\![\overline{T}]\!]_{\mathsf{c}} \qquad Q \vdash_{\mathtt{CH}} x : [\![S]\!]_{\mathsf{c}}}{\overline{x}(y).(P \mid Q) \vdash_{\mathtt{CH}} x : [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}}$$

   By the encoding of types in Figure 7, we have $[\![!T.S]\!]_{\mathsf{c}} = [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}$ which concludes the case.
4. Case $\{\!|\&\{l_i : S_i\}_{i\in I}|\!\}^x = \{x \triangleright \{l_i : P_i\}_{i\in I} \mid \forall i \in I.\ P_i \in \{\!|S_i|\!\}^x\}$. By rule (T&) and induction hypothesis we have

$$\dfrac{P_i \vdash_{\mathtt{CH}} x : [\![S_i]\!]_{\mathsf{c}} \qquad \forall i \in I}{x \triangleright \{l_i : P_i\}_{i\in I} \vdash_{\mathtt{CH}} x : \&\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i\in I}}$$

   By the encoding of types in Figure 7, we have $[\![\&\{l_i : S_i\}_{i\in I}]\!]_{\mathsf{c}} = \&\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i\in I}$, which concludes the case.

5. Case $\{\!|\oplus\{l_i : S_i\}_{i\in I}|\!\}^x = \bigcup_{i\in I}\{x \triangleleft l_j.P_j \mid P_j \in \{\!|S_i|\!\}^x\}$. For any process P_j in the set, by rule (T$\oplus$) and induction hypothesis we have

$$\frac{P_j \vdash_{\mathtt{CH}} x : [\![S_j]\!]_{\mathsf{c}}}{x \triangleleft l_j.P_j \vdash_{\mathtt{CH}} x{:}\oplus\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i\in I}}$$

By the encoding of types in Figure 7, we have $[\![\oplus\{l_i : S_i\}_{i\in I}]\!]_{\mathsf{c}} = \oplus\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i\in I}$ which concludes this case.

□

A.4 Proof of Theorem 5.8

We start with an auxiliary lemma.

Lemma A.5 (Substitution in C-Types)**.** *If $P \vdash_{\mathtt{CH}} \Delta, x : A$ then $P[z/x] \vdash_{\mathtt{CH}} \Delta, z{:}A$.*

We repeat the statement in Page 14:

Theorem A.6 (Rewriting is Type Preserving)**.** *Let $(\Gamma \vdash_{\mathtt{ST}} P) \in \mathscr{K}_n$. Then, $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathsf{c}}$.*

Proof. The proof proceeds by induction on the derivation $\Gamma \vdash_{\mathtt{ST}} P$. We will use inversion on typing in order to determine the session typing context Γ.

1. $\Gamma \vdash_{\mathtt{ST}} \mathbf{0}$. Then $\Gamma = \{x : \mathbf{end}\}$ and

$$\frac{}{x : \mathbf{end} \vdash_{\mathtt{ST}} \mathbf{0}}\ (\text{T-NIL})$$

By Definition 5.7, $(\!|x : \mathbf{end} \vdash_{\mathtt{ST}} \mathbf{0}|\!) = \mathbf{0}$. Then by applying rule (T-**1**) we have

$$\frac{}{\mathbf{0} \vdash_{\mathtt{CH}} x : \bullet}$$

The the thesis follows by encoding of types $[\![\cdot]\!]_{\mathsf{c}}$ (given in Figure 7) and Definition 3.9, which ensure $[\![x : \mathbf{end}]\!]_{\mathsf{c}} = x : \bullet$.

2. $\Gamma \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P'$. Then $\Gamma = \Gamma', x : !T.S, v : T$ and

$$\frac{\Gamma', x : S \vdash_{\mathtt{ST}} P'}{\Gamma', x : !T.S, v : T \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P'}\ (\text{T-OUT})$$

By Definition 5.7, $(\!|\Gamma \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P'|\!) = \overline{x}(z).\big([v \leftrightarrow z] \mid (\!|\Gamma', x : S \vdash_{\mathtt{ST}} P'|\!)\big)$. By typing rule (T-id) and by Lemma 3.10 we have

$$\frac{}{[v \leftrightarrow z] \vdash_{\mathtt{CH}} v : [\![T]\!]_{\mathsf{c}}, z : [\![\overline{T}]\!]_{\mathsf{c}}} \tag{19}$$

By induction hypothesis we have that $(\!|\Gamma', x : S \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}}$

By applying typing rule (T-$\otimes$) on the induction hypothesis and (19) we have that

$$\frac{[v \leftrightarrow z] \vdash_{\mathtt{CH}} v : [\![T]\!]_{\mathsf{c}}, z : [\![\overline{T}]\!]_{\mathsf{c}} \qquad (\!|\Gamma', x : S \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}}}{\overline{x}(z).\big([v \leftrightarrow z] \mid (\!|\Gamma', x : S \vdash_{\mathtt{ST}} P|\!)\big) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}, v : [\![T]\!]_{\mathsf{c}}}$$

where, by encoding of types $[\![\cdot]\!]_{\mathsf{c}}$ (given in Figure 7) we have $[\![!T.S]\!]_{\mathsf{c}} = [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}$ and by Definition 3.9, we have $[\![\Gamma', x : !T.S, v : T]\!]_{\mathsf{c}} = [\![\Gamma']\!]_{\mathsf{c}}, x : [\![\overline{T}]\!]_{\mathsf{c}} \otimes [\![S]\!]_{\mathsf{c}}, v : [\![T]\!]_{\mathsf{c}}$.

3. $\Gamma \vdash_{\mathtt{ST}} x(y:T).P'$. Then $\Gamma = \Gamma', x : ?T.S$ and

$$\frac{\Gamma', x : S, y : T \vdash_{\mathtt{ST}} P'}{\Gamma', x : ?T.S \vdash_{\mathtt{ST}} x(y:T).P'} \text{ (T-In)}$$

By Definition 5.7, we have $(\!|\Gamma \vdash_{\mathtt{ST}} x(y:T).P'|\!) = x(y).(\!|\Gamma', x : S, y : T \vdash_{\mathtt{ST}} P'|\!)$. By induction hypothesis we have $(\!|\Gamma', x : S, y : T \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}}, y : [\![T]\!]_{\mathsf{c}}$. By typing rule (T-⅋) on the induction hypothesis we have

$$\frac{(\!|\Gamma', x : S, y : T \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S]\!]_{\mathsf{c}}, y : [\![T]\!]_{\mathsf{c}}}{x(y).(\!|\Gamma', x : S, y : T \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![T]\!]_{\mathsf{c}} \parr [\![S]\!]_{\mathsf{c}}}$$

where by encoding of types $[\![\cdot]\!]_{\mathsf{c}}$ (given in Figure 7) we have that $[\![?T.S]\!]_{\mathsf{c}} = [\![T]\!]_{\mathsf{c}} \parr [\![S]\!]_{\mathsf{c}}$ and by Definition 3.9, we have $[\![\Gamma', x : ?T.S]\!]_{\mathsf{c}} = [\![\Gamma']\!]_{\mathsf{c}}, x : [\![T]\!]_{\mathsf{c}} \parr [\![S]\!]_{\mathsf{c}}$.

4. $P = x \lhd l_j.P'$: Then $\Gamma = \Gamma', x : \oplus\{l_i : S_i\}_{i \in I}$ and

$$\frac{\Gamma', x : S_j \vdash_{\mathtt{ST}} P' \qquad j \in I}{\Gamma', x : \oplus\{l_i : S_i\}_{i \in I} \vdash_{\mathtt{ST}} x \lhd l_j.P'} \text{ (T-Sel)}$$

By Definition 5.7 we have that $(\!|\Gamma \vdash_{\mathtt{ST}} x \lhd l_j.P'|\!) = x \lhd l_j.(\!|\Gamma', x : S_j \vdash_{\mathtt{ST}} P'|\!)$. By induction hypothesis we have $(\!|\Gamma', x : S_j \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S_j]\!]_{\mathsf{c}}$. By applying the typing rule (T-⊕) on the induction hypothesis we have:

$$\frac{(\!|\Gamma', x : S_j \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S_j]\!]_{\mathsf{c}} \qquad j \in I}{x \lhd l_j.(\!|\Gamma', x : S_j \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : \oplus\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i \in I}}$$

By encoding of types $[\![\cdot]\!]_{\mathsf{c}}$ (given in Figure 7) and Definition 3.9, we have $[\![\Gamma', x : \oplus\{l_i : S_i\}_{i \in I}]\!]_{\mathsf{c}} = [\![\Gamma']\!]_{\mathsf{c}}, x : \oplus\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i \in I}$, which concludes the case.

5. $P = x \rhd \{l_i : P_i\}_{i \in I}$: Then $\Gamma = \Gamma', x : \&\{l_i : S_i\}_{i \in I}$ and

$$\frac{\Gamma', x : S_i \vdash_{\mathtt{ST}} P_i \qquad \forall i \in I}{\Gamma', x : \&\{l_i : S_i\}_{i \in I} \vdash_{\mathtt{ST}} x \rhd \{l_i : P_i\}_{i \in I}} \text{ (T-Bra)}$$

By Definition 5.7, we have

$$(\!|\Gamma \vdash_{\mathtt{ST}} x \rhd \{l_i : P_i\}_{i \in I}|\!) = x \rhd \{l_i : (\!|\Gamma', x : S_i \vdash_{\mathtt{ST}} P_i|\!)\}_{i \in I}$$

By induction hypothesis we have for all $i \in I$

$$(\!|\Gamma', x : S_i \vdash_{\mathtt{ST}} P_i|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S_i]\!]_{\mathsf{c}} \tag{20}$$

By applying typing rule (T-&) on the induction hypothesis (20) we have

$$\frac{(\!|\Gamma', x : S_i \vdash_{\mathtt{ST}} P_i|\!) \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : [\![S_i]\!]_{\mathsf{c}}}{x \rhd \{l_i : (\!|\Gamma', x : S_i \vdash_{\mathtt{ST}} P_i|\!)\}_{i \in I} \vdash_{\mathtt{CH}} [\![\Gamma']\!]_{\mathsf{c}}, x : \&\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i \in I}}$$

where by encoding of types $[\![\cdot]\!]_{\mathsf{c}}$ (given in Figure 7) we have $[\![\&\{l_i : S_i\}_{i \in I}]\!]_{\mathsf{c}} = \&\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i \in I}$ and by Definition 3.9, we have $[\![\Gamma', x : \&\{l_i : S_i\}_{i \in I}]\!]_{\mathsf{c}} = [\![\Gamma']\!]_{\mathsf{c}}, x : \&\{l_i : [\![S_i]\!]_{\mathsf{c}}\}_{i \in I}$. This concludes the case.

6. $\Gamma \vdash_{\mathtt{ST}} (\nu \widetilde{x}\widetilde{y} : \widetilde{S})(Q \mid R)$. Then $\Gamma = \Gamma_1 \circ \Gamma_2$, where $\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q$ and $\Gamma_2, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} R$, and

$$\dfrac{\dfrac{\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q \qquad \Gamma_2, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} R}{\Gamma_1 \circ \Gamma_2, \widetilde{x} : \widetilde{S}, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} Q \mid R}\ (\text{T-Par})}{\Gamma_1 \circ \Gamma_2 \vdash_{\mathtt{ST}} (\nu \widetilde{x}\widetilde{y} : \widetilde{S})(Q \mid R)}\ (\text{T-Res})$$

Notice that, since the only restricted names in P are $\widetilde{x}\widetilde{y}$ and the restriction creates co-variables, it means that $\Gamma_1 \cap \Gamma_2 = \emptyset$. Hence, by the definition of $\circ$ we have that $\Gamma_1 \circ \Gamma_2 = \Gamma_1, \Gamma_2$.
By Definition 5.7, we have that $(\!|\Gamma \vdash_{\mathtt{ST}} (\nu \widetilde{x}\widetilde{y} : \widetilde{S})(Q \mid R)|\!)$ is the process

$$(\nu k)\big(\mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q|\!)\big[\widetilde{z}/\widetilde{x}\big]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \parallel_k \mathscr{C}_{\widetilde{z}:\widetilde{\overline{S}}}\big[(\!|\Gamma_2, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} R|\!)\big[\widetilde{z}/\widetilde{y}\big]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)$$

By induction hypothesis on the premises of the typing rule (T-Par) we have that

$$(\!|\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q|\!) \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}}, \widetilde{x} : [\![\widetilde{S}]\!]_{\mathsf{c}}$$

and also by Lemma 3.10

$$(\!|\Gamma_2, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} R|\!) \vdash_{\mathtt{CH}} [\![\Gamma_2]\!]_{\mathsf{c}}, \widetilde{y} : [\![\widetilde{\overline{S}}]\!]_{\mathsf{c}}$$

By applying Lemma A.5 on the induction hypothesis for x and y we have

$$(\!|\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q|\!)\big[\widetilde{z}/\widetilde{x}\big] \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}}, \widetilde{z} : [\![\widetilde{S}]\!]_{\mathsf{c}}$$

and

$$(\!|\Gamma_2, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} R|\!)\big[\widetilde{z}/\widetilde{y}\big] \vdash_{\mathtt{CH}} [\![\Gamma_2]\!]_{\mathsf{c}}, \widetilde{z} : [\![\widetilde{\overline{S}}]\!]_{\mathsf{c}}$$

By using the catalysers on the two judgements above to obtain a closure on z and by applying Lemma 5.4 we have:

$$\mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q|\!)\big[\widetilde{z}/\widetilde{x}\big]\big] \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}} \tag{21}$$

and

$$\mathscr{C}_{\widetilde{z}:\widetilde{\overline{S}}}\big[(\!|\Gamma_2, \widetilde{y} : \widetilde{\overline{S}} \vdash_{\mathtt{ST}} R|\!)\big[\widetilde{z}/\widetilde{y}\big]\big] \vdash_{\mathtt{CH}} [\![\Gamma_2]\!]_{\mathsf{c}} \tag{22}$$

By applying Lemma 5.3 we have that $\{\!|T_i|\!\}^{w_i} \vdash_{\mathtt{CH}} w_i : [\![T_i]\!]_{\mathsf{c}}$ for all $w_i : T_i \in \Gamma_2$ and $\{\!|T_i'|\!\}^{w_i'} \vdash_{\mathtt{CH}} w_i' : [\![T_i']\!]_{\mathsf{c}}$ for all $(w_i' : T_i') \in \Gamma_1$. Finally, by applying the typing rule (T-Mix) we have:

$$\dfrac{\{\!|T_i|\!\}^{w_i} \vdash_{\mathtt{CH}} w_i : [\![T_i]\!]_{\mathsf{c}} \qquad \forall w_i : T_i \in \Gamma_2}{\prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \vdash_{\mathtt{CH}} [\![\Gamma_2]\!]_{\mathsf{c}}}\ (\text{T-mix}) \tag{23}$$

and

$$\dfrac{\{\!|T_i'|\!\}^{w_i'} \vdash_{\mathtt{CH}} w_i' : [\![T_i']\!]_{\mathsf{c}} \qquad \forall (w_i' : T_i') \in \Gamma_1}{\prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'} \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}}}\ (\text{T-mix}) \tag{24}$$

where $\widetilde{w} : [\![\widetilde{T}]\!]_{\mathsf{c}} = [\![\Gamma_2]\!]_{\mathsf{c}}$ and $w' : [\![T']\!]_{\mathsf{c}} = [\![\Gamma_1]\!]_{\mathsf{c}}$. By applying (T-mix) on (21) and the conclusion of (23) we have:

$$(\text{T-mix})\ \dfrac{\mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q|\!)\big[\widetilde{z}/\widetilde{x}\big]\big] \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}} \qquad \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \vdash_{\mathtt{CH}} [\![\Gamma_2]\!]_{\mathsf{c}}}{\mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1, \widetilde{x} : \widetilde{S} \vdash_{\mathtt{ST}} Q|\!)\big[\widetilde{z}/\widetilde{x}\big]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \vdash_{\mathtt{CH}} [\![\Gamma_1]\!]_{\mathsf{c}}, [\![\Gamma_2]\!]_{\mathsf{c}}} \tag{25}$$

and by applying (T-mix) on (22) and the conclusion of (24) we have:

$$\text{(T-mix)}\ \frac{\mathscr{C}_{\widetilde{z}:\widetilde{\widetilde{S}}}\big[(\!|\Gamma_2,\widetilde{y}:\widetilde{\widetilde{S}}\vdash_{\mathtt{ST}} R|\!)[\widetilde{z}/\widetilde{y}]\big]\vdash_{\mathtt{CH}}[\![\Gamma_2]\!]_{\mathsf{c}} \qquad \prod_{(w_i':T_i')\in\Gamma_1}\{\!|T_i'|\!\}^{w_i'}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}}}{\mathscr{C}_{\widetilde{z}:\widetilde{\widetilde{S}}}\big[(\!|\Gamma_2,\widetilde{y}:\widetilde{\widetilde{S}}\vdash_{\mathtt{ST}} R|\!)[\widetilde{z}/\widetilde{y}]\big]\mid\prod_{(w_i':T_i')\in\Gamma_1}\{\!|T_i'|\!\}^{w_i'}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}}} \tag{26}$$

By applying typing rule (T-$\bot$) on k and (T-&) on the conclusions of 25 and 26, and by letting

$$\mathscr{Q}=\mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1,\widetilde{x}:\widetilde{S}\vdash_{\mathtt{ST}} Q|\!)[\widetilde{z}/\widetilde{x}]\big]\mid\prod_{(w_i:T_i)\in\Gamma_2}\{\!|T_i|\!\}^{w_i}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}},k:\bullet$$

and

$$\mathscr{R}=\mathscr{C}_{\widetilde{z}:\widetilde{\widetilde{S}}}\big[(\!|\Gamma_2,\widetilde{y}:\widetilde{\widetilde{S}}\vdash_{\mathtt{ST}} R|\!)[\widetilde{z}/\widetilde{y}]\big]\mid\prod_{(w_i':T_i')\in\Gamma_1}\{\!|T_i'|\!\}^{w_i'}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}},k:\bullet$$

we have the following derivation:

$$\text{(T-\&)}\ \frac{\begin{array}{c}\mathscr{C}_{\widetilde{z}:\widetilde{S}}\big[(\!|\Gamma_1,\widetilde{x}:\widetilde{S}\vdash_{\mathtt{ST}} Q|\!)[\widetilde{z}/\widetilde{x}]\big]\mid\prod_{(w_i:T_i)\in\Gamma_2}\{\!|T_i|\!\}^{w_i}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}},k:\bullet\\ \mathscr{C}_{\widetilde{z}:\widetilde{\widetilde{S}}}\big[(\!|\Gamma_2,\widetilde{y}:\widetilde{\widetilde{S}}\vdash_{\mathtt{ST}} R|\!)[\widetilde{z}/\widetilde{y}]\big]\mid\prod_{(w_i':T_i')\in\Gamma_1}\{\!|T_i'|\!\}^{w_i'}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}},k:\bullet\end{array}}{k\triangleright\{l_1:\mathscr{Q},l_2:\mathscr{R}\}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}},k:\&\{l_1:\bullet,l_2:\bullet\}} \tag{27}$$

By applying (T-$\otimes$) we have

$$\frac{\mathbf{0}\vdash_{\mathtt{CH}}k:\bullet \qquad j\in\{1,2\}}{k\triangleleft\mathtt{inx}.\mathbf{0}\vdash_{\mathtt{CH}}k:\oplus\{\mathtt{inl}:\bullet,\mathtt{inr}:\bullet\}}$$

We conclude by applying (T-cut)

$$\frac{k\triangleleft\mathtt{inx}.\mathbf{0}\vdash_{\mathtt{CH}}k:\oplus\{\mathtt{inl}:\bullet,\mathtt{inr}:\bullet\} \qquad k\triangleright\{l_1:\mathscr{Q},l_2:\mathscr{R}\}\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}},k:\&\{\mathtt{inl}:\bullet,\mathtt{inr}:\bullet\}}{(\nu k)\big(k\triangleleft\mathtt{inx}.\mathbf{0}\mid k\triangleright\{\mathtt{inl}:\mathscr{Q},\mathtt{inr}:\mathscr{R}\}\big)\vdash_{\mathtt{CH}}[\![\Gamma_1]\!]_{\mathsf{c}},[\![\Gamma_2]\!]_{\mathsf{c}}}$$

□

A.5 Proof of Theorem 5.10

We repeat the statement in Page 14:

Theorem A.7 (Operational Correspondence)**.** *Let $P\in\mathscr{K}_n$ such that $\Gamma\vdash_{\mathtt{ST}}P$ for some Γ. Then we have:*

I) If $P\to P'$ then there exist Q, Q' s.t. (i) $(\!|\Gamma\vdash_{\mathtt{ST}}P|\!)\to^{\mathtt{inx}}\to^*\equiv Q$; (ii) $Q\doteqdot Q'$; (iii) $(\!|\Gamma\vdash_{\mathtt{ST}}P'|\!)\to^{\mathtt{inx}}Q'$.

II) If $(\!|\Gamma\vdash_{\mathtt{ST}}P|\!)\to^{\mathtt{inx}}\to^*\equiv Q$ then there exists P' s.t. $P\to P'$ and $Q\doteqdot(\!|\Gamma\vdash_{\mathtt{ST}}P'|\!)$.

Proof. I) The proof is done by induction on the length of the derivation $P\to P'$.

1. Case (R-Com):

$$P\triangleq(\nu xy:S')(\overline{x}\langle v\rangle.P_1\mid y(t:T).P_2)\to(\nu xy:S'')(P_1\mid P_2[v/t])\triangleq P'$$

Since $\Gamma \vdash_{\mathtt{ST}} P$, by inversion let $S' = !T.S$ for some S and $S'' = S$. Then, again by inversion $\Gamma = (\Gamma_1, v : T) \circ \Gamma_2$ meaning that

$$\frac{\Gamma_1, x : S \vdash_{\mathtt{ST}} P_1}{\Gamma_1, v : T, x : !T.S \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P_1}$$

and

$$\frac{\Gamma_2, y : \overline{S}, t : T \vdash_{\mathtt{ST}} P_2}{\Gamma_2, y : ?T.\overline{S} \vdash_{\mathtt{ST}} y(t : T).P_2}$$

By Definition 5.7 we have

$$\begin{aligned}
(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) &= (\!|\Gamma \vdash_{\mathtt{ST}} (\nu xy : !T.S)(\overline{x}\langle v\rangle.P_1 \mid y(t : T).P_2)|\!) \\
&= (\nu k)\big(\mathscr{C}_{z:!T.S}\big[(\!|\Gamma_1, x : !T.S, v : T \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P_1|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\qquad \|_k\, \mathscr{C}_{z:?T.\overline{S}}\big[(\!|\Gamma_2, y : ?T.\overline{S} \vdash_{\mathtt{ST}} y(t : T).P_2|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\{\Gamma_1, v:T\}} \{\!|T_i'|\!\}^{w_i'}\big) \\
&= (\nu k)\big(\mathscr{C}_{z:!T.S}\big[(\!|\Gamma_1, z : !T.S, v : T \vdash_{\mathtt{ST}} \overline{z}\langle v\rangle.P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\qquad \|_k\, \mathscr{C}_{z:?T.\overline{S}}\big[(\!|\Gamma_2, z : ?T.\overline{S} \vdash_{\mathtt{ST}} z(t : T).P_2[z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\{\Gamma_1, v:T\}} \{\!|T_i'|\!\}^{w_i'}\big)
\end{aligned}$$

By the reduction rules given in Fig. 1:

- If $\mathtt{inx} = \mathtt{inl}$ we have the following reduction:

$$\begin{aligned}
(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) &\to \mathscr{C}_{z:!T.S}\big[(\!|\Gamma_1, z : !T.S, v : T \vdash_{\mathtt{ST}} \overline{z}\langle v\rangle.P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&= \mathscr{C}_{z:!T.S}\big[\overline{z}(w).\big([v \leftrightarrow w] \mid (\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\equiv (\nu z)\big((\nu w)(\overline{z}\langle w\rangle.\big([v \leftrightarrow w] \mid (\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big) \mid z(s).Q)\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i}
\end{aligned}$$

  By Definition 5.1 we have that $z(s).Q \in \{\!|?T.\overline{S}|\!\}^z$ with $Q \vdash_{\mathtt{CH}} s{:}[\![T]\!]_{\mathtt{c}}, z{:}[\![\overline{S}]\!]_{\mathtt{c}}$. By applying rules (R-Com), (R-ChRes) and (R-Fwd) given in Fig. 1 we have the following reductions:

$$\begin{aligned}
&\to\to (\nu z)\big((\nu w)([v \leftrightarrow w] \mid (\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!) \mid Q\{w/s\})\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\to (\nu z)\big((\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!) \mid Q\{v/w\}\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \triangleq \mathscr{P}_1
\end{aligned}$$

  By Theorem 5.8 we have that $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$. By hypothesis $P \to P'$, then by subject

reduction $(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$. and the rewriting of P' is as follows.

$$\begin{aligned}(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) &= (\!|\Gamma \vdash_{\mathtt{ST}} (\nu xy : S)(P_1 \mid P_2[v/t])|\!) \\ &= (\nu k)\big(\mathscr{C}_{z:S}\big[(\!|\Gamma_1, x : S \vdash_{\mathtt{ST}} P_1|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2,v:T\}} \{\!|T_i|\!\}^{w_i} \\ &\qquad \|_k\, \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, y : \overline{S}, v : T \vdash_{\mathtt{ST}} P_2[v/t]|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big) \\ &= (\nu k)\big(\mathscr{C}_{z:S}\big[(\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2,v:T\}} \{\!|T_i|\!\}^{w_i} \\ &\qquad \|_k\, \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, z : \overline{S}, v : T \vdash_{\mathtt{ST}} P_2[v/t][z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)\end{aligned}$$

Since `inl` was chosen, then we have the following reduction, where $R_{\overline{z}} \in \{\!|\overline{S}|\!\}^z$:

$$\begin{aligned}&\to \mathscr{C}_{z:S}\big[(\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2,v:T\}} \{\!|T_i|\!\}^{w_i} \\ &= (\nu z)\big((\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!) \mid R_{\overline{z}}\big) \mid \prod_{(w_i:T_i)\in\{\Gamma_2,v:T\}} \{\!|T_i|\!\}^{w_i} \triangleq \mathscr{P}_1'\end{aligned}$$

Recall that process $\mathscr{P}_1$ is well-typed in $[\![\Gamma]\!]_{\mathtt{c}}$; process $\mathscr{P}_1'$ is also well-typed in $[\![\Gamma]\!]_{\mathtt{c}}$. Then, $\mathscr{P}_1 \doteqdot \mathscr{P}_1'$ and P_1 is a subterm of both $\mathscr{P}_1$ and $\mathscr{P}_1'$.

- If $\mathtt{inx} = \mathtt{inr}$ we have the following reduction:

$$\begin{aligned}(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) &\to \mathscr{C}_{z:?T.\overline{S}}\big[(\!|\Gamma_2, z : ?T.\overline{S} \vdash_{\mathtt{ST}} z(t : T).P_2[z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\{\Gamma_1,v:T\}} \{\!|T_i'|\!\}^{w_i'} \\ &= \mathscr{C}_{z:?T.\overline{S}}\big[z(t).(\!|\Gamma_2, z : \overline{S}, t : T \vdash_{\mathtt{ST}} P_2[z/y]|\!)\big] \mid \mid \prod_{(w_i':T_i')\in\{\Gamma_1,v:T\}} \{\!|T_i'|\!\}^{w_i'} \\ &\equiv (\nu z)\big(z(t).(\!|\Gamma_2, z : \overline{S}, t : T \vdash_{\mathtt{ST}} P_2[z/y]|\!) \mid (\nu u)\overline{z}\langle u\rangle.(R \mid Q)\big) \mid \prod_{(w_i':T_i')\in\{\Gamma_1,v:T\}} \{\!|T_i'|\!\}^{w_i'}\end{aligned}$$

By Definition 5.1 we have that $\overline{z}(u).(R \mid Q) \in \{\!|!T.S|\!\}^z$ with $R \in \{\!|\overline{T}|\!\}^u$ and $Q \in \{\!|S|\!\}^z$. By applying rule (R-Com) given in Fig. 1 we have the following reductions:

$$\to\equiv (\nu z)\big((\nu u)(\!|\Gamma_2, z : \overline{S}, u : T \vdash_{\mathtt{ST}} P_2[z/y][u/t]|\!) \mid R \mid Q)\big) \mid \prod_{(w_i':T_i')\in\{\Gamma_1,v:T\}} \{\!|T_i'|\!\}^{w_i'} \triangleq \mathscr{P}_2$$

By Theorem 5.8 process $(\nu z)\big((\nu u)(\!|\Gamma_2, z : \overline{S}, u : T \vdash_{\mathtt{ST}} P_2[z/y][u/t]|\!) \mid R \mid Q)\big)$ is well-typed in $[\![\Gamma_2]\!]_{\mathtt{c}}$, since restrictions of z and u remove them from the typing environment $\Gamma_2, z : \overline{S}, u : T$. The encoding of P' is the same as before; we recall it here for simplicity:

$$\begin{aligned}(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) &= (\!|\Gamma \vdash_{\mathtt{ST}} (\nu xy : S)(P_1 \mid P_2[v/t])|\!) \\ &= (\nu k)\big(\mathscr{C}_{z:S}\big[(\!|\Gamma_1, x : S \vdash_{\mathtt{ST}} P_1|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2,v:T\}} \{\!|T_i|\!\}^{w_i} \\ &\qquad \|_k\, \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, y : \overline{S}, v : T \vdash_{\mathtt{ST}} P_2[v/t]|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big) \\ &= (\nu k)\big(\mathscr{C}_{z:S}\big[(\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2,v:T\}} \{\!|T_i|\!\}^{w_i} \\ &\qquad \|_k\, \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, z : \overline{S}, v : T \vdash_{\mathtt{ST}} P_2[v/t][z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)\end{aligned}$$

Since `inr` was chosen, then we have the following reduction, where $R_{\overline{z}} \in \{\!|S|\!\}^z$:

$$\begin{aligned}&\to \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, z:\overline{S}, v:T \vdash_{\mathrm{ST}} P_2[v/t][z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\\ &= (\nu z)\big((\!|\Gamma_2, z:\overline{S}, v:T \vdash_{\mathrm{ST}} P_2[v/t][z/y]|\!) \mid R_{\overline{z}}\big) \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'} \triangleq \mathscr{P}_2'\end{aligned}$$

Process $\mathscr{P}_2$ and process $\mathscr{P}_2'$ are well-typed in $[\![\Gamma]\!]_{\mathsf{c}}$. Then $\mathscr{P}_2 \doteqdot \mathscr{P}_2'$ and P_2 is a subterm of both $\mathscr{P}_2$ and $\mathscr{P}_2'$.

2. Case (R-Case):

$$P \triangleq (\nu xy:S')(x \triangleleft l_j.Q \mid y \triangleright \{l_i : R_i\}_{i\in I}) \to (\nu xy:S'')(Q \mid R_j) \triangleq P'$$

Since $\Gamma \vdash_{\mathrm{ST}} P$, by inversion let $S' = \oplus\{l_i : S_i\}_{i\in I}$ for some S and $S'' = S_j$. Then, again by inversion $\Gamma = \Gamma_1 \circ \Gamma_2$ and

$$\frac{\Gamma_1, x:S_j \vdash Q \quad \exists j \in I}{\Gamma_1, x:\oplus\{l_i:S_i\}_{i\in I} \vdash x \triangleleft l_j.Q}$$

and

$$\frac{\Gamma_2, y:\overline{S_i} \vdash R_i \quad \forall i \in I}{\Gamma_2, y:\&\{l_i:\overline{S_i}\}_{i\in I} \vdash y \triangleright \{l_i : R_i\}_{i\in I}}$$

By Definition 5.7 we have

$$\begin{aligned}(\!|\Gamma \vdash_{\mathrm{ST}} P|\!) =& (\!|\Gamma \vdash_{\mathrm{ST}} (\nu xy:S')(x \triangleleft l_j.Q \mid y \triangleright \{l_i : R_i\}_{i\in I})|\!)\\ =& (\nu k)\big(\mathscr{C}_{z:S'}\big[(\!|\Gamma_1, x:S' \vdash_{\mathrm{ST}} x \triangleleft l_j.Q|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i}\\ &\|_k \mathscr{C}_{z:\overline{S'}}\big[(\!|\Gamma_2, y:\overline{S'} \vdash_{\mathrm{ST}} y \triangleright \{l_i : R_i\}_{i\in I}|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)\end{aligned}$$

By the reduction rules given in Fig. 1:

- If $\mathtt{inx} = \mathtt{inl}$ we have the following reduction:

$$\begin{aligned}(\!|\Gamma \vdash_{\mathrm{ST}} P|\!) &\to \mathscr{C}_{z:S'}\big[(\!|\Gamma_1, x:S' \vdash_{\mathrm{ST}} x \triangleleft l_j.Q|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i}\\ &\equiv (\nu z)\big((\!|\Gamma_1, z:S' \vdash_{\mathrm{ST}} z \triangleleft l_j.Q[z/x]|\!) \mid z \triangleright \{l_i : R_i'\}_{i\in I}\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i}\end{aligned}$$

By Definition 5.1, $z \triangleright \{l_i : R_i'\}_{i\in I} \in \{\!|\&\{l_i : \overline{S_i}\}_{i\in I}|\!\}^z$ and for all $i \in I$ we have $R_i' \in \{\!|\overline{S}|\!\}^z$. In particular $R_j' \in \{\!|\overline{S_j}|\!\}^z$. By applying rules (R-Res) and (R-Case) the process performs the following reduction:

$$\to\to (\nu z)\big((\!|\Gamma_1, z:S_j \vdash_{\mathrm{ST}} Q[z/x]|\!) \mid R_j'\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \triangleq \mathscr{Q}$$

Since z is restricted, the last process is well-typed in $\Gamma_1 \circ \Gamma_2$.

By Theorem 5.8 we have that $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$. By hypothesis $P \to P'$, then by subject reduction $(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$, and the rewriting of P' is as follows.

$$\begin{aligned}(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) &= (\!|\Gamma \vdash_{\mathtt{ST}} (\nu xy : S_j)(Q \mid R_j)|\!) \\ &= (\nu k)\big(\mathscr{C}_{z:S_j}\big[(\!|\Gamma_1, x : S_j \vdash_{\mathtt{ST}} Q|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\ &\quad \|_k \mathscr{C}_{z:\overline{S_j}}\big[(\!|\Gamma_2, y : \overline{S_j} \vdash_{\mathtt{ST}} R_j|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)\end{aligned}$$

Since `inl` was chosen, then we have the following reduction where $R_{\overline{z}} \in \{\!|\overline{S_j}|\!\}^z$:

$$\begin{aligned}&\to \mathscr{C}_{z:S_j}\big[(\!|\Gamma_1, z : S_j \vdash_{\mathtt{ST}} Q[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\ &= (\nu z)\big((\!|\Gamma_1, z : S_j \vdash_{\mathtt{ST}} Q[z/x]|\!) \mid R_{\overline{z}}\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} = \mathscr{Q}\end{aligned}$$

- If $\mathtt{inx} = \mathtt{inr}$, since by Definition 5.1 we have that $\{\!| \oplus \{l_i : S_i\}_{i\in I} |\!\}^z = \bigcup_{i\in I} \big\{z \triangleleft l_i.P_i \mid P_i \in \{\!|S_i|\!\}^z\big\}$, then there exists an index $j \in I$. Recall that l_j is the label selected in process P. We hence use label l_j for the reduction below

$$\begin{aligned}(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) &\to \mathscr{C}_{z:\overline{S'}}\big[(\!|\Gamma_2, y : \overline{S'} \vdash_{\mathtt{ST}} y \triangleright \{l_i : R_i\}_{i\in I}|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'} \\ &\equiv (\nu z)\big((\!|\Gamma_2, z : \overline{S'} \vdash_{\mathtt{ST}} z \triangleright \{l_i : R_i\}_{i\in I}[z/y]|\!) \mid z \triangleleft l_j.R'\big) \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\end{aligned}$$

By Definition 5.1 $z \triangleleft l_j.R' \in \{\!| \oplus \{l_i : S_i\}_{i\in I} |\!\}^z$ for $j \in I$ and $R' \in \{\!|S_j|\!\}^z$. By applying rules (R-Res) and (R-Case) the process performs the following reduction:

$$\to\to (\nu z)\big((\!|\Gamma_2, z : \overline{S_k} \vdash_{\mathtt{ST}} R_k[z/y]|\!) \mid R'\big) \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'} \triangleq \mathscr{R}$$

Since z is restricted, the last process is well-typed in $\Gamma_1 \circ \Gamma_2$.
By Theorem 5.8 we have that $(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$. By hypothesis $P \to P'$, then by subject reduction $(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) \vdash_{\mathtt{CH}} [\![\Gamma]\!]_{\mathtt{c}}$, and the rewriting of P' is presented above.
Since `inr` was chosen, then we have the following reduction where $R_{\overline{z}} \in \{\!|S_j|\!\}^z$

$$\begin{aligned}(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) &\to \mathscr{C}_{z:\overline{S_j}}\big[(\!|\Gamma_2, z : \overline{S_j} \vdash_{\mathtt{ST}} R_j[z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big) \\ &= (\nu z)\big((\!|\Gamma_2, z : \overline{S_j} \vdash_{\mathtt{ST}} R_j[z/y]|\!) \mid R_{\overline{z}}\big) \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)\end{aligned}$$

II) The proof is done by induction on the structure of P.

1. Case $P \triangleq (\nu xy : S')(\overline{x}\langle v\rangle.P_1 \mid y(t : T).P_2)$. Since $\Gamma \vdash_{\mathtt{ST}} P$, by inversion let $S' = !T.S$ for some S and $S'' = S$. Then, again by inversion $\Gamma = (\Gamma_1, v : T) \circ \Gamma_2$ meaning that

$$\frac{\Gamma_1, x : S \vdash_{\mathtt{ST}} P_1}{\Gamma_1, v : T, x : !T.S \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P_1}$$

and

$$\frac{\Gamma_2, y : \overline{S}, t : T \vdash_{\mathtt{ST}} P_2}{\Gamma_2, y : ?T.\overline{S} \vdash_{\mathtt{ST}} y(t:T).P_2}$$

By Definition 5.7 we have

$$\begin{aligned}
(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) &= (\!|\Gamma \vdash_{\mathtt{ST}} (\nu xy : !T.S)(\overline{x}\langle v\rangle.P_1 \mid y(t:T).P_2)|\!) \\
&= (\nu k)\big(\mathscr{C}_{z:!T.S}\big[(\!|\Gamma_1, x : !T.S, v : T \vdash_{\mathtt{ST}} \overline{x}\langle v\rangle.P_1|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\qquad \|_k\, \mathscr{C}_{z:?T.\overline{S}}\big[(\!|\Gamma_2, y : ?T.\overline{S} \vdash_{\mathtt{ST}} y(t:T).P_2|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\{\Gamma_1, v:T\}} \{\!|T_i'|\!\}^{w_i'}\big) \\
&= (\nu k)\big(\mathscr{C}_{z:!T.S}\big[(\!|\Gamma_1, z : !T.S, v : T \vdash_{\mathtt{ST}} \overline{z}\langle v\rangle.P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\qquad \|_k\, \mathscr{C}_{z:?T.\overline{S}}\big[(\!|\Gamma_2, z : ?T.\overline{S} \vdash_{\mathtt{ST}} z(t:T).P_2[z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\{\Gamma_1, v:T\}} \{\!|T_i'|\!\}^{w_i'}\big)
\end{aligned}$$

By the reduction rules given in Fig. 1:

- If $\mathtt{inx} = \mathtt{inl}$ we have the following reduction:

$$\begin{aligned}
(\!|\Gamma \vdash_{\mathtt{ST}} P|\!) &\to \mathscr{C}_{z:!T.S}\big[(\!|\Gamma_1, z : !T.S, v : T \vdash_{\mathtt{ST}} \overline{z}\langle v\rangle.P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&= \mathscr{C}_{z:!T.S}\big[\overline{z}(w).\big([v\leftrightarrow w] \mid (\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big)\big] \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\equiv (\nu z)\big((\nu w)(\overline{z}\langle w\rangle.\big([v\leftrightarrow w] \mid (\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big) \mid z(s).Q)\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i}
\end{aligned}$$

  By Definition 5.1 we have that $z(s).Q \in \{\!|?T.\overline{S}|\!\}^z$ with $Q \vdash_{\mathtt{CH}} s{:}[\![T]\!]_{\mathtt{c}}, z{:}[\![\overline{S}]\!]_{\mathtt{c}}$. By applying rules (R-Com), (R-ChRes) and (R-Fwd) given in Fig. 1 we have the following reductions:

$$\begin{aligned}
&\to\to (\nu z)\big((\nu w)([v\leftrightarrow w] \mid (\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!) \mid Q\{w/s\})\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \\
&\to (\nu z)\big((\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!) \mid Q\{v/w\}\big) \mid \prod_{(w_i:T_i)\in\Gamma_2} \{\!|T_i|\!\}^{w_i} \triangleq \mathscr{Q}
\end{aligned}$$

  By (R-Com) $P \to (\nu xy : S'')(P_1 \mid P_2[v/t]) \triangleq P$. Since $\Gamma \vdash_{\mathtt{ST}} P$, by inversion let $S' = !T.S$ for some S and $S'' = S$. The rewriting of P_1 is as follows:

$$\begin{aligned}
(\!|\Gamma \vdash_{\mathtt{ST}} P'|\!) &= (\!|\Gamma \vdash_{\mathtt{ST}} (\nu xy : S)(P_1 \mid P_2[v/t])|\!) \\
&= (\nu k)\big(\mathscr{C}_{z:S}\big[(\!|\Gamma_1, x : S \vdash_{\mathtt{ST}} P_1|\!)[z/x]\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2, v:T\}} \{\!|T_i|\!\}^{w_i} \\
&\qquad \|_k\, \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, y : \overline{S}, v : T \vdash_{\mathtt{ST}} P_2[v/t]|\!)[z/y]\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big) \\
&= (\nu k)\big(\mathscr{C}_{z:S}\big[(\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2, v:T\}} \{\!|T_i|\!\}^{w_i} \\
&\qquad \|_k\, \mathscr{C}_{z:\overline{S}}\big[(\!|\Gamma_2, z : \overline{S}, v : T \vdash_{\mathtt{ST}} P_2[v/t][z/y]|\!)\big] \mid \prod_{(w_i':T_i')\in\Gamma_1} \{\!|T_i'|\!\}^{w_i'}\big)
\end{aligned}$$

Since `inl` was chosen, then we have the following reduction, where $R_{\overline{z}} \in \{\!|\overline{S}|\!\}^z$:

$$\begin{aligned}
&\to \mathscr{C}_{z:S}\big[(\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!)\big] \mid \prod_{(w_i:T_i)\in\{\Gamma_2, v:T\}} \{\!|T_i|\!\}^{w_i} \\
&= (\nu z)\big((\!|\Gamma_1, z : S \vdash_{\mathtt{ST}} P_1[z/x]|\!) \mid R_{\overline{z}}\big) \mid \prod_{(w_i:T_i)\in\{\Gamma_2, v:T\}} \{\!|T_i|\!\}^{w_i} \triangleq \mathscr{R}
\end{aligned}$$

Recall that process $\mathscr{Q}$ is well-typed in $[\![\Gamma]\!]_{\mathsf{c}}$; process $\mathscr{R}$ is also well-typed in $[\![\Gamma]\!]_{\mathsf{c}}$. Then, $\mathscr{Q} \doteqdot \mathscr{R}$ and P_1 is a subterm of both $\mathscr{Q}$ and $\mathscr{R}$.

□

A.6 Example of the Rewriting Procedure

Consider again the process

$$P_2 = (\nu a_1 b_1)(\nu a_2 b_2)(a_1(x).\,\overline{a_2}\langle x\rangle \mid \overline{b_1}\langle \mathbf{n}\rangle.\,b_2(z))$$

that we used in Lemma 4.3. We have that

$$\begin{aligned}
a_1 : ?\,\mathbf{end}, a_2 : !\,\mathbf{end}, x : \mathbf{end} &\vdash_{\mathtt{ST}} a_1(x).\,\overline{a_2}\langle x\rangle && \star \\
b_1 : !\,\mathbf{end}, b_2 : ?\,\mathbf{end}, \mathbf{n} : \mathbf{end} &\vdash_{\mathtt{ST}} \overline{b_1}\langle \mathbf{n}\rangle.\,b_2(z) && \star\star
\end{aligned}$$

and let $\Gamma_1 = x : \mathbf{end}$, $\Gamma_2 = \mathbf{n} : \mathbf{end}$ and $\Gamma = \Gamma_1 \circ \Gamma_2$. By applying Definition 5.7 we have

$$\begin{aligned}
(\!|\Gamma \vdash_{\mathtt{ST}} P_2|\!) = \;& (\nu k)\big(\{\!|\mathbf{end}|\!\}^{\mathbf{n}} \mid (\nu z_1)(\nu z_2)(\{\!|!\,\mathbf{end}|\!\}^{z_1} \mid \{\!|?\,\mathbf{end}|\!\}^{z_2} \mid (\!|\star|\!)[z_1, z_2/a_1, a_2]) \\
& \|_k \{\!|\mathbf{end}|\!\}^{x} \mid (\nu z_1)(\nu z_2)(\{\!|?\,\mathbf{end}|\!\}^{z_1} \mid \{\!|!\,\mathbf{end}|\!\}^{z_2} \mid (\!|\star\star|\!)[z_1, z_2/b_1, b_2])\big) \\
\equiv\; & (\nu k)\big((\nu z_1)(\nu z_2)(\overline{z_1}(\mathbf{n}).\mathbf{0} \mid z_2(y).\mathbf{0} \mid z_1(x).\overline{z_2}(w)([x \leftrightarrow w] \mid \mathbf{0})) \\
& \|_k (\nu z_1)(\nu z_2)(z_1(y).\mathbf{0} \mid \overline{z_2}(\mathbf{n}).\mathbf{0} \mid \overline{z_1}(v)([\mathbf{n} \leftrightarrow v] \mid z_2(z).\mathbf{0}))\big) \triangleq \mathscr{P}_2
\end{aligned}$$

Now, to illustrate the reduction that process $\mathscr{P}_2$ can perform, suppose `inr` is chosen:

$$\begin{aligned}
\mathscr{P}_2 \to & (\nu z_1)(\nu z_2)(z_1(y).\mathbf{0} \mid \overline{z_2}(\mathbf{n}).\mathbf{0} \mid \overline{z_1}(v)([\mathbf{n} \leftrightarrow v] \mid z_2(z).\mathbf{0}))\big) \\
\equiv\; & (\nu z_1)(\nu z_2)(z_1(y).\mathbf{0} \mid \overline{z_2}(\mathbf{n}).\mathbf{0} \mid (\nu v)\overline{z_1}\langle v\rangle([\mathbf{n} \leftrightarrow v] \mid z_2(z).\mathbf{0}))\big) \\
\equiv\; & (\nu v)(\nu z_1)(\nu z_2)(z_1(y).\mathbf{0} \mid \overline{z_2}(\mathbf{n}).\mathbf{0} \mid \overline{z_1}\langle v\rangle([\mathbf{n} \leftrightarrow v] \mid z_2(z).\mathbf{0}))\big) \to^*
\end{aligned}$$