Core Higher-Order Session Processes: Tractable Equivalences and Relative Expressiveness⋆

Dimitrios Kouzapas[1,2], Jorge A. Pérez[3], and Nobuko Yoshida[1]

[1] Imperial College London
[2] University of Glasgow
[3] University of Groningen

Abstract. This work proposes tractable bisimulations for the higher-order π-calculus with session primitives ($\mathsf{HO}\pi$) and offers a complete study of the expressivity of its most significant subcalculi. First we develop three typed bisimulations, which are shown to coincide with contextual equivalence. These characterisations demonstrate that observing as inputs only a specific finite set of higher-order values (which inhabit session types) suffices to reason about $\mathsf{HO}\pi$ processes. Next, we identify HO, a minimal, second-order subcalculus of $\mathsf{HO}\pi$ in which higher-order applications/abstractions, name-passing, and recursion are absent. We show that HO can encode $\mathsf{HO}\pi$ extended with higher-order applications and abstractions and that a first-order session π-calculus can encode $\mathsf{HO}\pi$. Both encodings are fully abstract. We also prove that the session π-calculus with passing of shared names cannot be encoded into $\mathsf{HO}\pi$ without shared names. We show that $\mathsf{HO}\pi$, HO, and π are equally expressive; the expressivity of HO enables effective reasoning about typed equivalences for higher-order processes.

⋆ Last Revision: February 9, 2015

Table of Contents

1 Introduction

By combining features from the λ-calculus and the π-calculus, in *higher-order process calculi* exchanged values may contain processes. In this paper, we consider higher-order calculi with *session primitives*, thus enabling the specification of reciprocal exchanges (protocols) for higher-order mobile processes, which can be verified via type-checking using *session types* [19]. The study of higher-order concurrency has received significant attention, from untyped and typed perspectives (see, e.g., [53,48,47,22,35,29,28,24,55]). Although models of session-typed communication with features of higher-order concurrency exist [33,14], their *tractable behavioural equivalences* and *relative expressiveness* remain little understood. Clarifying their status is not only useful for, e.g., justifying non-trivial mobile protocol optimisations, but also for transferring key reasoning techniques between (higher-order) session calculi. Our discovery is that *linearity* of session types plays a vital role to offer new equalities and fully abstract encodability, which to our best knowledge have not been proposed before.

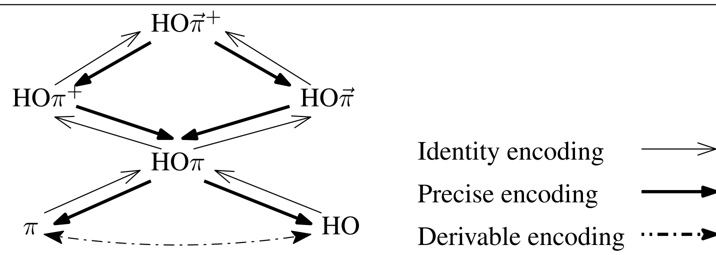
The main higher-order language in our work, denoted $\mathsf{HO}\pi$, extends the higher-order π-calculus [48] with session primitives: it contains constructs for synchronisation on shared names, recursion, name abstractions (i.e., functions from name identifiers to processes, denoted $\lambda x.P$) and applications (denoted $(\lambda x.P)a$); and session communication (value passing and labelled choice using linear names). We study two significant subcalculi of $\mathsf{HO}\pi$, which distil higher- and first-order mobility: the HO-calculus, which is $\mathsf{HO}\pi$ without recursion and name passing, and the session π-calculus (here denoted π), which is $\mathsf{HO}\pi$ without abstractions and applications. While π is, in essence, the calculus in [19], this paper shows that HO is a new core calculus for higher-order session concurrency.

In the first part of the paper, we address tractable behavioural equivalences for $\mathsf{HO}\pi$. A well-studied behavioural equivalence in the higher-order setting is *context bisimilarity* [46], a labelled characterisation of reduction-closed, barbed congruence, which offers an appropriate discriminative power at the price of heavy universal quantifications in output clauses. Obtaining alternative characterisations is thus a recurring issue in the study of higher-order calculi. Our approach shows that protocol specifications given by session types are essential to limit the behaviour of higher-order session processes. Exploiting elementary processes inhabiting session types, this limitation is formally enforced by a refined (typed) labelled transition system (LTS) that narrows down the spectrum of allowed process behaviours, thus enabling tractable reasoning techniques. Two tractable characterisations of bisimilarity are then introduced. Remarkably, using session types we prove that these bisimilarities coincide with context bisimilarity, without using operators for name-matching.

We then move on to assess the expressivity of $\mathsf{HO}\pi$, HO, and π as delineated by typing. We establish strong correspondences between these calculi via type-preserving, fully abstract encodings up to behavioural equalities. While encoding $\mathsf{HO}\pi$ into the π-calculus preserving session types (extending known results for untyped processes) is significant, our main contribution is an encoding of $\mathsf{HO}\pi$ into HO, where name-passing is absent.

We illustrate the essence of encoding name passing into HO: to encode name output, we "pack" the name to be passed around into a suitable abstraction; upon reception, the

Fig. 1 Encodability in Higher-Order Session Calculi. Precise encodings are defined in Definition 5.5.



receiver must "unpack" this object following a precise protocol. More precisely, our encoding of name passing in HO is given as:

$$[\![a!\langle b\rangle.P]\!] = a!\langle \lambda z.\, z?(x).(x\, b)\rangle.[\![P]\!]$$
$$[\![a?(x).Q]\!] = a?(y).(\nu\ s)(y\, s \mid \overline{s}!\langle \lambda x.\, [\![Q]\!]\rangle.\mathbf{0})$$

where a,b are names; s and $\overline{s}$ are linear names (called *session endpoints*); $a!\langle V\rangle.P$ and $a?(x).P$ denote an output and input at a; and $(\nu\ s)(P)$ is hiding. A (deterministic) reduction between endpoints s and $\overline{s}$ guarantees name b is properly unpacked. Encoding a recursive process $\mu X.P$ is also challenging, for the linearity of endpoints in P must be preserved. We encode recursion with non-tail recursive session types; for this we apply recent advances on the theory of session duality [5,6].

We further extend our encodability results to i) $\mathsf{HO}\pi$ with *higher-order* abstractions (denoted $\mathsf{HO}\pi^+$) and to ii) $\mathsf{HO}\pi$ with polyadic name passing and abstraction ($\mathsf{HO}\vec{\pi}$); and to their super-calculus ($\mathsf{HO}\vec{\pi}^+$) (equivalent to the calculus in [33]). A further result shows that shared names strictly add expressive power to session calculi. Figure 1 summarises these results.

Outline / Contributions. This paper is structured as follows:

- Section 2 presents the higher-order session calculus $\mathsf{HO}\pi$ and its subcalculi HO and π.
- Section 3 gives the type system and states type soundness for $\mathsf{HO}\pi$ and its variants.
- Section 4 develops *higher-order* and *characteristic* bisimilarities, our two tractable characterisations of contextual equivalence which alleviate the issues of context bisimilarity [46]. These relations are shown to coincide in $\mathsf{HO}\pi$ (Theorem 4.1).
- Section 5 defines *precise (typed) encodings* by extending encodability criteria studied for untyped processes (e.g. [16,28]).
- Section 6 and Section 7 gives encodings of $\mathsf{HO}\pi$ into HO and of $\mathsf{HO}\pi$ into π. These encodings are shown to be *precise* (Proposition 6.6 and Proposition 6.10). Mutual encodings between π and HO are derivable; all these calculi are thus equally expressive. Exploiting determinacy and typed equivalences, we also prove the non-encodability of shared names into linear names (Theorem 7.1).
- Section 8 studies extensions of $\mathsf{HO}\pi$. We show that both $\mathsf{HO}\pi^+$ (the extension with higher-order applications) and $\mathsf{HO}\vec{\pi}$ (the extension with polyadicity) are encodable in $\mathsf{HO}\pi$ (Proposition 8.4 and Proposition 8.8). This connects our work to the existing higher-order session calculus in [33] (here denoted $\mathsf{HO}\vec{\pi}^+$).

- Section 9 reviews related works. The appendix collects proofs of the main results.

2 The Higher-Order Session π-Calculus ($\mathsf{HO}\pi$)

We introduce the *Higher-Order Session π-Calculus* ($\mathsf{HO}\pi$). $\mathsf{HO}\pi$ includes both name- and abstraction-passing operators as well as recursion; it corresponds to a subcalculus of the language studied by Mostrous and Yoshida in [33,35]. Following the literature [22], for simplicity of the presentation we concentrate on the second-order call-by-value $\mathsf{HO}\pi$. (In Section 8 we consider the extension of $\mathsf{HO}\pi$ with general higher-order abstractions and polyadicity in name-passing/abstractions.) We also introduce two subcalculi of $\mathsf{HO}\pi$. In particular, we define the core higher-order session calculus (HO), which includes constructs for shared name synchronisation and constructs for session establishment/communication and (monadic) name-abstraction, but lacks name-passing and recursion.

Although minimal, in Section 5 the abstraction-passing capabilities of $\mathsf{HO}\pi$ will prove expressive enough to capture key features of session communication, such as delegation and recursion.

2.1 Syntax

The syntax for $\mathsf{HO}\pi$ processes is given in Figure 2.

Identifiers. We use $a,b,c,\ldots$ to range over shared names, and $s,\overline{s},\ldots$ to range over session names whereas $m,n,t,\ldots$ range over shared or session names. We define dual session endpoints $\overline{s}$, with the dual operator defined as $\overline{\overline{s}} = s$ and $\overline{a} = a$. Intuitively, names s and $\overline{s}$ are dual *endpoints*. Name and abstraction variables are uniformly denoted with $x,y,z,\ldots$; we reserve k for name variables and we sometimes write $\underline{x}$ for abstraction variables. Recursive variables are denoted with $X, Y \ldots$. An abstraction $\lambda x.\, P$ is a process P with bound variable x. Symbols $u,v,\ldots$ range over names or variables. Furthermore we use $V, W, \ldots$ to denote transmittable values; either channels u, v or abstractions.

Terms. The name-passing constructs of $\mathsf{HO}\pi$ include the π-calculus prefixes for sending and receiving values V. Process $u!\langle V\rangle.P$ denotes the output of value V over channel u, with continuation P; process $u?(x).P$ denotes the input prefix on channel u of a value that it is going to be substituted on variable x in continuation P. Recursion is expressed by the primitive recursor $\mu X.P$, which binds the recursive variable X in process P. Process $V\,u$ is the application process; it binds channel u on the abstraction V. Prefix $u \triangleleft l.P$ selects label l on channel u and then behaves as P. Given $i \in I$ process $u \triangleright \{l_i : P_i\}_{i\in I}$ offers a choice on labels l_i with continuation P_i. The calculus also includes standard constructs for the inactive process $\mathbf{0}$, parallel composition $P_1 \mid P_2$, and name restriction $(\nu\ n)P$. Session name restriction $(\nu\ s)P$ simultaneously binds endpoints s and $\overline{s}$ in P. We use $\mathtt{fv}(P)$ and $\mathtt{fn}(P)$ to denote a set of free variables and names, respectively; and assume V in $u!\langle V\rangle.P$ does not include free recursive variables X. Furthremore, a well-formed process relies on assumptions for guarded recursive processes. If $\mathtt{fv}(P) = \emptyset$, we call P *closed*. We write $\mathcal{P}$ for the set of all well-formed processes.

Fig. 2 Syntax for $\mathsf{HO}\pi$ (The definition of HO lacks the constructs in grey)

$$
\begin{array}{llll}
\text{(Processes)} & P,Q & ::= & u!\langle V\rangle.P \mid u?(x).P \mid V\,u \\
 & & \mid & u \triangleleft l.P \mid u \triangleright \{l_i : P_i\}_{i\in I} \mid \mathbf{0} \\
 & & \mid & P \mid Q \mid (\nu\, n)P \mid X \mid \mu X.P
\end{array}
\qquad
\begin{array}{llll}
\text{(Names)} & n,m,t & ::= & a,b \mid s,\overline{s} \\
\text{(Identifiers)} & u,v & ::= & n \mid x,y,z,k \\
\text{(Values)} & V,Q & ::= & u \mid \lambda x.\,P
\end{array}
$$

2.2 Sub-calculi

We identify two main sub-calculi of $\mathsf{HO}\pi$ that will form the basis of our study:

Definition 2.1 (Sub-calculi of $\mathsf{HO}\pi$). *We let* $\mathsf{C} \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$ *with:*

- Core higher-order session calculus (HO)*: The sub-calculus* HO *uses only abstraction passing, i.e., values in Figure 2 are defined as in the non-gray syntax;* $V \ ::= \ \lambda x.\,P$ *and does not use the primitive recursion constructs,* X *and* $\mu X.P$*.*
- Session π-calculus (π)*: The sub-calculus* π *uses only name-passing constructs, i.e., values in Figure 2 are defined as* $V \ ::= \ u$*, and does not use applications* $x\,u$*.*

We write $\mathsf{C}^{-\mathsf{sh}}$ *to denote a sub-calculus without shared names, i.e., identifiers in Figure 2 are defined as* $u,v \ ::= \ s,\overline{s}$*.*

Thus, while π is essentially the standard session π-calculus as defined in the literature [19,13], HO can be related to a subcalculus of higher-order process calculi as studied in the untyped [48,50,22] and typed settings [33,34,35]. In Section 6 we show that $\mathsf{HO}\pi$, HO, and π have the same expressivity.

2.3 Operational Semantics

The operational semantics for $\mathsf{HO}\pi$ is standardly given as a *reduction relation*, supported by a structural congruence relation, denoted $\equiv$. Structural congruence is the least congruence that satisfies the commutative monoid $(\mathcal{P}, \mid, \mathbf{0})$:

$$P \mid \mathbf{0} \equiv P \qquad P_1 \mid P_2 \equiv P_2 \mid P_1 \qquad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$$

satisfies α-conversion:

$$P_1 \equiv_\alpha P_2 \ \text{ implies } \ P_1 \equiv P_2$$

and furthermore, satisfies the rules:

$$n \notin \mathtt{fn}(P_1) \ \text{ implies } \ P_1 \mid (\nu\, n)P_2 \equiv (\nu\, n)(P_1 \mid P_2)$$

$$(\nu\, n)\mathbf{0} \equiv \mathbf{0} \qquad (\nu\, n)(\nu\, m)P \equiv (\nu\, m)(\nu\, n)P \qquad \mu X.P \equiv P\{^{\mu X.P}/X\}$$

The first rule is describes scope opening for names. Restricting of a name in an inactive process has no effect. Furthermore, we can permute name restrictions. Recursion is defined in structural congruence terms; a recursive term $\mu X.P$ is structurally equivalent to its unfolding.

Fig. 3 Reduction semantics for $\mathsf{HO}\pi$.

$$(\lambda x.\, P)\, u \longrightarrow P\{^{u}/x\} \qquad [\text{App}]$$

$$n!\langle V\rangle.P \mid \overline{n}?(x).Q \longrightarrow P \mid Q\{^{V}/x\} \qquad [\text{Pass}]$$

$$n \triangleleft l_j.Q \mid \overline{n} \triangleright \{l_i : P_i\}_{i\in I} \longrightarrow Q \mid P_j \;\; (j \in I) \qquad [\text{Sel}]$$

$$\frac{P \longrightarrow P'}{(\nu\, n)P \longrightarrow (\nu\, n)P'}\ [\text{Sess}] \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}\ [\text{Par}] \qquad \frac{P \equiv\longrightarrow\equiv P'}{P \longrightarrow P'}\ [\text{Cong}]$$

Structural congruence is extended to support values, i.e., is the least congruence over processes and values that satisfies $\cong$ for processes and, furthermore:

$$\lambda x.\, P_1 \equiv_\alpha \lambda y.\, P_2 \ \text{ implies } \ \lambda x.\, P_1 \equiv \lambda y.\, P_2 \qquad\qquad P_1 \equiv P_2 \text{ implies } \lambda x.\, P_1 \equiv \lambda x.\, P_2$$

This way, abstraction values are congruent up-to α-conversion. Furthermore, two congruent processes can construct congruent abstractions.

Figure 3 defines the operational semantics for the $\mathsf{HO}\pi$. [App] is a name application. Rule [Pass] defines value passing where value V is being send on channel n to its dual endpoint $\overline{n}$ (for shared interactions $\overline{n} = n$). As a result of the value passing reduction the continuation of the receiving process substitutes the receiving variable x with V. Rule [Sel] is the standard rule for labelled choice/selection; given an index set I, a process selects label $l_j, j \in I$ on channel n over a set of labels $\{l_i\}_{i\in I}$ that are offered by a parallel process on the dual session endpoint $\overline{n}$. Remaining rules define congruence with respect to parallel composition (rule [Par]) and name restriction (rule [Ses]). Rule [Cong] defines closure under structural congruence. We write $\longrightarrow^*$ for a multi-step reduction.

3 Session Types for $\mathsf{HO}\pi$

In this section we define a session typing system for $\mathsf{HO}\pi$ and establish its main properties. We use as a reference the type system for higher-order session processes developed by Mostrous and Yoshida [33,34,35]. Our system is simpler than that in [33], in order to distil the key features of higher-order communication in a session-typed setting.

3.1 Syntax

We define the syntax of session types for $\mathsf{HO}\pi$.

Definition 3.1 (Syntax of Types). *The syntax of types is defined on the types for sessions S, and the types for values U:*

$$\begin{array}{llcl}
\textit{(value)} & U & ::= & \boxed{C} \mid L \\
\textit{(name)} & C & ::= & S \mid \langle S\rangle \mid \langle L\rangle \\
\textit{(abstr)} & L & ::= & C{\rightarrow}\diamond \mid C{\multimap}\diamond \\
\textit{(session)} & S,T & ::= & !\langle U\rangle;S \mid ?(U);S \mid \oplus\{l_i : S_i\}_{i\in I} \mid \&\{l_i : S_i\}_{i\in I} \\
 & & \mid & \mu \mathsf{t}.S \mid \mathsf{t} \mid \mathsf{end}
\end{array}$$

Types for Values. Types for values range over symbol U which includes first-order types C and higher-order types L. First-order types C are used to type names; session types S type session names and shared types $\langle S\rangle$ or $\langle L\rangle$ type shared names that carry session values and higher-order values, respectively. Higher-order types L are used to type abstraction values; $C{\rightarrow}\diamond$ and $C{\multimap}\diamond$ denote shared and linear abstraction types, respectively.

Session Types. The syntax of session types S follows the usual (binary) session types with recursion [19,13]. An *output type* $!\langle U\rangle;S$ is assigned to a name that first sends a value of type U and then follows the type described by S. Dually, the *input type* $?(U);S$ is assigned to a name that first receives a value of type U and then continues as S. Session types for labelled choice and selection, written $\&\{l_i : S_i\}_{i\in I}$ and $\oplus\{l_i : S_i\}_{i\in I}$, respectively, require a set of types $\{S_i\}_{i\in I}$ that correspond to a set of labels $\{i \in I\}_{i\in I}$. *Recursive session types* are defined using the primitive recursor. We require type variables to be *guarded*; this means, e.g., that type $\mu\mathsf{t.t}$ is not allowed. Type end is the termination type. We let T to be the set of all well-formed types and ST to be the set of all well-formed session types.

Types of HO exclude C from value types of $\mathsf{HO}\pi$; the types of π exclude L. From each $\mathsf{C} \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$, $\mathsf{C}^{-\mathsf{sh}}$ excludes shared name types ($\langle S\rangle$ and $\langle L\rangle$), from name type C.

Remark 3.1 (Restriction on Types for Values). The syntax for value types is restricted to disallow types of the form:

- $\langle\langle U\rangle\rangle$: shared names cannot carry shared names; and
- $U{\rightarrow}\diamond$: abstractions do not bind higher-order variables.

The difference between the syntax of process in $\mathsf{HO}\pi$ with the syntax of processes in [33,35] is also reflected on the two corresponding type syntax; the type structure in [33,35], supports the arrow types of the form $U \rightarrow T$ and $U \multimap T$, where T denotes an arbitrary type of a term (i.e. a value or a process).

3.2 Duality

Duality is defined following the co-inductive approach, as in [13,5]. We first require the notion of type equivalence.

Definition 3.2 (Type Equivalence). *Define function $F(\Re) : \mathsf{T} \longrightarrow \mathsf{T}$:*

$$\begin{aligned}
F(\Re) = {} & \{(\mathsf{end}, \mathsf{end})\} \\
& \cup \{(\langle S\rangle, \langle T\rangle) \mid S \Re T\} \cup \{(\langle L_1\rangle, \langle L_2\rangle) \mid L_1 \Re L_2\} \\
& \cup \{(C_1{\rightarrow}\diamond, C_2{\rightarrow}\diamond), (C_1{\multimap}\diamond, C_2{\multimap}\diamond) \mid C_1 \Re C_2\} \\
& \cup \{(!\langle U_1\rangle;S, !\langle U_2\rangle;T), (?(U_1);S, ?(U_1);T) \mid U_1 \Re U_2, S \Re T\} \\
& \cup \{(\oplus\{l_i : S_i\}_{i\in I}, \oplus\{l_i : T_i\}_{i\in I}) \mid S_i \Re T_i\} \\
& \cup \{(\&\{l_i : S_i\}_{i\in I}, \&\{l_i : T_i\}_{i\in I}) \mid S_i \Re T_i\} \\
& \cup \{(S, T) \mid S\{^{\mu\mathsf{t}.S}/_{\mathsf{t}}\} \Re T)\} \\
& \cup \{(S, T) \mid S \Re T\{^{\mu\mathsf{t}.T}/_{\mathsf{t}}\})\}
\end{aligned}$$

Standard arguments ensure that F is monotone, thus the greatest fixed point of F exists. Let type equivalence be defined as $\mathsf{iso} = \nu X.F(X)$.

In essence, type equivalence is a co-inductive definition that equates types up-to recursive unfolding. We may now define the duality relation in terms of type equivalence.

Definition 3.3 (Duality). *Define function $F(\mathfrak{R}) : \mathsf{ST} \longrightarrow \mathsf{ST}$:*

$$\begin{aligned}
F(\mathfrak{R}) = \{&(\mathsf{end}, \mathsf{end})\} \\
\cup\, \{&(!\langle U_1\rangle;S\,, ?(U_2);T)\,, (?(U);S\,, !\langle U\rangle;T) \;\mid\; U_1 \;\mathsf{iso}\; U_2, S \;\mathfrak{R}\; T\} \\
\cup\, \{&(\oplus\{l_i : S_i\}_{i\in I}\,, \&\{l_i : T_i\}_{i\in I}) \;\mid\; S_i \;\mathfrak{R}\; T_i\} \\
\cup\, \{&(\&\{l_i : S_i\}_{i\in I}\,, \oplus\{l_i : T_i\}_{i\in I}) \;\mid\; S_i \;\mathfrak{R}\; T_i\} \\
\cup\, \{&(S\,, T) \;\mid\; S\{^{\mu \mathsf{t}.S}/\mathsf{t}\} \;\mathfrak{R}\; T)\} \\
\cup\, \{&(S\,, T) \;\mid\; S \;\mathfrak{R}\; T\{^{\mu \mathsf{t}.T}/\mathsf{t}\})\}
\end{aligned}$$

Standard arguments ensure that F is monotone, thus the greatest fixed point of F exists. Let duality be defined as $\mathsf{dual} = \nu X.F(X)$.

Duality is applied co-inductively to session types up-to recursive unfolding. Dual session types are prefixed on dual session type constructors that carry equivalent types (! is dual to ? and $\oplus$ is dual to &).

3.3 Type Environments and Judgements

Following [33,35], we define the typing environments.

Definition 3.4 (Typing environment). *We define the* shared type environment *Γ, the* linear type environment *Λ, and the* session type environment *Δ as:*

$$\begin{array}{lcl}
\textit{(Shared)}\ \Gamma & ::= & \emptyset \mid \Gamma \cdot x : C{\rightarrow}\diamond \mid \Gamma \cdot u : \langle S\rangle \mid \Gamma \cdot u : \langle L\rangle \mid \Gamma \cdot X : \Delta \\
\textit{(Linear)}\ \Lambda & ::= & \emptyset \mid \Lambda \cdot x : C{\multimap}\diamond \\
\textit{(Session)}\ \Delta & ::= & \emptyset \mid \Delta \cdot u : S
\end{array}$$

We further require:

i. Domains of Γ, Λ, Δ are pairwise distinct.
ii. Weakening, contraction and exchange apply to shared environment Γ.
iii. Exchange applies to linear environments Λ and Δ.

We define typing judgements for values V and processes P:

$$\Gamma; \Lambda; \Delta \vdash V \triangleright U \qquad\qquad \Gamma; \Lambda; \Delta \vdash P \triangleright \diamond$$

The first judgement asserts that under environment $\Gamma; \Lambda; \Delta$ values V have type U, whereas the second judgement asserts that under environment $\Gamma; \Lambda; \Delta$ process P has the typed process type $\diamond$.

Fig. 4 Typing Rules for $\mathsf{HO}\pi$.

$$[\text{Sess}]\ \Gamma;\emptyset;\{u:S\}\vdash u\triangleright S \qquad [\text{Sh}]\ \Gamma\cdot u:U;\emptyset;\emptyset\vdash u\triangleright U \qquad [\text{LVar}]\ \Gamma;\{x:C{\multimap}\diamond\};\emptyset\vdash x\triangleright C{\multimap}\diamond$$

$$[\text{Prom}]\ \frac{\Gamma;\emptyset;\emptyset\vdash V\triangleright C{\multimap}\diamond}{\Gamma;\emptyset;\emptyset\vdash V\triangleright C{\rightarrow}\diamond} \qquad [\text{EProm}]\ \frac{\Gamma;\Lambda\cdot x:C{\multimap}\diamond;\Delta\vdash P\triangleright\diamond}{\Gamma\cdot x:C{\rightarrow}\diamond;\Lambda;\Delta\vdash P\triangleright\diamond}$$

$$[\text{Abs}]\ \frac{\Gamma;\Lambda;\Delta_1\vdash P\triangleright\diamond \quad \Gamma;\emptyset;\Delta_2\vdash x\triangleright C}{\Gamma;\Lambda;\Delta_1\backslash\Delta_2\vdash \lambda x.P\triangleright C{\multimap}\diamond}$$

$$[\text{App}]\ \frac{U=C{\multimap}\diamond\vee C{\rightarrow}\diamond \quad \Gamma;\Lambda;\Delta_1\vdash V\triangleright U \quad \Gamma;\emptyset;\Delta_2\vdash u\triangleright C}{\Gamma;\Lambda;\Delta_1\cdot\Delta_2\vdash V\,u\triangleright\diamond}$$

$$[\text{Send}]\ \frac{\Gamma;\Lambda_1;\Delta_1\vdash P\triangleright\diamond \quad \Gamma;\Lambda_2;\Delta_2\vdash V\triangleright U \quad u:S\in\Delta_1\cdot\Delta_2}{\Gamma;\Lambda_1\cdot\Lambda_2;((\Delta_1\cdot\Delta_2)\backslash\{u:S\})\cdot u:!\langle U\rangle;S\vdash u!\langle V\rangle.P\triangleright\diamond}$$

$$[\text{Rcv}]\ \frac{\Gamma;\Lambda_1;\Delta_1\cdot u:S\vdash P\triangleright\diamond \quad \Gamma;\Lambda_2;\Delta_2\vdash x\triangleright C}{\Gamma\backslash x;\Lambda_1\backslash\Lambda_2;\Delta_1\backslash\Delta_2\cdot u:?(C);S\vdash u?(x).P\triangleright\diamond}$$

$$[\text{Req}]\ \frac{\begin{array}{c}\Gamma;\emptyset;\emptyset\vdash u\triangleright U_1 \quad \Gamma;\Lambda;\Delta_1\vdash P\triangleright\diamond \quad \Gamma;\emptyset;\Delta_2\vdash V\triangleright U_2\\ (U_1=\langle S\rangle\Leftrightarrow U_2=S)\vee(U_1=\langle L\rangle\Leftrightarrow U_2=L)\end{array}}{\Gamma;\Lambda;\Delta_1\cdot\Delta_2\vdash u!\langle V\rangle.P\triangleright\diamond}$$

$$[\text{Acc}]\ \frac{\begin{array}{c}\Gamma;\emptyset;\emptyset\vdash u\triangleright U_1 \quad \Gamma;\Lambda_1;\Delta_1\vdash P\triangleright\diamond \quad \Gamma;\Lambda_2;\Delta_2\vdash x\triangleright U_2\\ (U_1=\langle S\rangle\Leftrightarrow U_2=S)\vee(U_1=\langle L\rangle\Leftrightarrow U_2=L)\end{array}}{\Gamma;\Lambda_1\backslash\Lambda_2;\Delta_1\backslash\Delta_2\vdash u?(x).P\triangleright\diamond}$$

$$[\text{Bra}]\ \frac{\forall i\in I \quad \Gamma;\Lambda;\Delta\cdot u:S_i\vdash P_i\triangleright\diamond}{\Gamma;\Lambda;\Delta\cdot u:\&\{l_i:S_i\}_{i\in I}\vdash u\triangleright\{l_i:P_i\}_{i\in I}\triangleright\diamond} \qquad [\text{Sel}]\ \frac{\Gamma;\Lambda;\Delta\cdot u:S_j\vdash P\triangleright\diamond \quad j\in I}{\Gamma;\Lambda;\Delta\cdot u:\oplus\{l_i:S_i\}_{i\in I}\vdash u\triangleleft l_j.P\triangleright\diamond}$$

$$[\text{Res}]\ \frac{\Gamma\cdot a:\langle S\rangle;\Lambda;\Delta\vdash P\triangleright\diamond}{\Gamma;\Lambda;\Delta\vdash(\nu\,a)P\triangleright\diamond} \qquad [\text{ResS}]\ \frac{\Gamma;\Lambda;\Delta\cdot s:S_1\cdot\overline{s}:S_2\vdash P\triangleright\diamond \quad S_1\ \mathsf{dual}\ S_2}{\Gamma;\Lambda;\Delta\vdash(\nu\,s)P\triangleright\diamond}$$

$$[\text{Par}]\ \frac{\Gamma;\Lambda_1;\Delta_1\vdash P_1\triangleright\diamond \quad \Gamma;\Lambda_2;\Delta_2\vdash P_2\triangleright\diamond}{\Gamma;\Lambda_1\cdot\Lambda_2;\Delta_1\cdot\Delta_2\vdash P_1\mid P_2\triangleright\diamond} \qquad [\text{End}]\ \frac{\Gamma;\Lambda;\Delta\vdash P\triangleright T \quad u\notin\mathsf{dom}(\Gamma,\Lambda,\Delta)}{\Gamma;\Lambda;\Delta\cdot u:\mathsf{end}\vdash P\triangleright\diamond}$$

$$[\text{Nil}]\ \Gamma;\emptyset;\emptyset\vdash\mathbf{0}\triangleright\diamond \qquad [\text{RVar}]\ \Gamma\cdot X:\Delta;\emptyset;\Delta\vdash X\triangleright\diamond \qquad [\text{Rec}]\ \frac{\Gamma\cdot X:\Delta;\emptyset;\Delta\vdash P\triangleright\diamond}{\Gamma;\emptyset;\Delta\vdash\mu X.P\triangleright\diamond}$$

3.4 Typing Rules

The type relation is defined in Figure 4. Rule [Session] requires the minimal session environment Δ to type session u with type S. Rule [LVar] requires the minimal linear environment Λ to type higher-order variable x with type $C{\rightarrow}\diamond$. Rule [Shared] assigns the value type U to shared names or shared variables u if the map $u:U$ exists in environment Γ. Rule [Shared] also requires that the linear environment is empty. The type $C{\rightarrow}\diamond$ for shared higher-order values V is derived using rule [Prom], where we require a value with linear type to be typed without a linear environment present in order to be used as a shared type. Rule [EProm] allows to freely use a linear type variable as shared type variable. Abstraction values are typed with rule [Abs]. The key type for an abstraction is the type for the bound variables of the abstraction, i.e., for bound variable with type C the abstraction has type $C{\multimap}\diamond$. The dual of abstraction typing is application typing governed by rule [App], where we expect the type C of an application name u to match the type $C{\multimap}\diamond$ or $C{\rightarrow}\diamond$ of the application variable x.

A process prefixed with a session send operator $u!\langle V\rangle.P$ is typed using rule [Send]. The type U of a send value V should appear as a prefix on the session type $!\langle U\rangle;S$ of s. Rule [Rcv] defines the typing for the reception of values $u?(V).P$. The type U of a receive value should appear as a prefix on the session type $?(U);S$ of u. We use a similar approach with session prefixes to type interaction between shared channels as defined in rules [Req] and [Acc], where the type of the sent/received object (S and L, respectively) should match the type of the sent/received subject ($\langle S\rangle$ and $\langle L\rangle$, respectively). Select and branch prefixes are typed using the rules [Sel] and [Bra] respectively. Both rules prefix the session type with the selection type $\oplus\{l_i : S_i\}_{i\in I}$ and $\&\{l_i : S_i\}_{i\in I}$.

The creation of a shared name a requires to add its type in environment Γ as defined in rule [Res]. Creation of a session name s creates two endpoints with dual types and adds them to the session environment Δ as defined in rule [ResS]. Rule [Par] concatenates the linear environment of the parallel components of a parallel operator to create a type for the composed process. The disjointness of environments Λ and Δ is implied. Rule [End] allows a form of weakening for the session environment Δ, provided that the name added in Δ has the inactive type $\mathtt{end}$. The inactive process $\mathbf{0}$ has an empty linear environment. The recursive variable is typed directly from the shared environment Γ as in rule [RVar]. The recursive operator requires that the body of a recursive process matches the type of the recursive variable as in rule [Rec].

3.5 Type Soundness

Type safety result are instances of more general statements already proved by Mostrous and Yoshida [33,35] in the asynchronous case.

Lemma 3.1 (Substitution Lemma - Lemma C.10 in [35]).

1. $\Gamma;\Lambda;\Delta\cdot x : S \vdash P \triangleright \diamond$ *and* $u \notin \mathtt{dom}(\Gamma,\Lambda,\Delta)$ *implies* $\Gamma;\Lambda;\Delta\cdot u : S \vdash P\{^{u}\!/x\} \triangleright \diamond$.
2. $\Gamma\cdot x : \langle U\rangle;\Lambda;\Delta \vdash P \triangleright \diamond$ *and* $a \notin \mathtt{dom}(\Gamma,\Lambda,\Delta)$ *implies* $\Gamma\cdot a : \langle U\rangle;\Lambda;\Delta \vdash P\{^{a}\!/x\} \triangleright \diamond$.
3. *If* $\Gamma;\Lambda_1\cdot x : C{\multimap}\diamond;\Delta_1 \vdash P \triangleright \diamond$ *and* $\Gamma;\Lambda_2;\Delta_2 \vdash V \triangleright C{\multimap}\diamond$ *with* $\Lambda_1\cdot\Lambda_2$ *and* $\Delta_1\cdot\Delta_2$ *defined, then* $\Gamma;\Lambda_1\cdot\Lambda_2;\Delta_1\cdot\Delta_2 \vdash P\{^{V}\!/x\} \triangleright \diamond$.
4. $\Gamma\cdot x : C{\rightarrow}\diamond;\Lambda;\Delta \vdash P \triangleright \diamond$ *and* $\Gamma;\emptyset;\emptyset \vdash V \triangleright C{\rightarrow}\diamond$ *implies* $\Gamma;\Lambda;\Delta \vdash P\{^{V}\!/x\} \triangleright \diamond$.

Proof. By induction on the typing for P, with a case analysis on the last used rule. □

We are interested in session environments which are *balanced*:

Definition 3.5 (Balanced Session Environment). *We say that session environment* Δ *is* balanced *if* $s : S_1, \overline{s} : S_2 \in \Delta$ *implies* S_1 dual S_2.

The type soundness relies on the following auxiliary definition:

Definition 3.6 (Session Environment Reduction). *The reduction relation $\longrightarrow$ on session environments is defined as:*

$$\Delta \cdot s :!\langle U\rangle; S_1 \cdot \overline{s} :?(U); S_2 \longrightarrow \Delta \cdot s : S_1 \cdot \overline{s} : S_2$$
$$\Delta \cdot s : \oplus\{l_i : S_i\}_{i\in I} \cdot \overline{s} : \&\{l_i : S'_i\}_{i\in I} \longrightarrow \Delta \cdot s : S_k \cdot \overline{s} : S'_k, \quad k \in I$$

We write $\longrightarrow^*$ for the multistep environment reduction.

We now state the main soundness result as an instance of type soundness from the system in [33]. It is worth noticing that in [33] has a slightly richer definition of structural congruence. Also, their statement for subject reduction relies on an ordering on typing associated to queues and other runtime elements. Since we are dealing with synchronous semantics we can omit such an ordering. The type soundness result implies soundness for the sub-calculi HO, π, and $\mathsf{C}^{-\mathsf{sh}}$

Theorem 3.1 (Type Soundness - Theorem 7.3 in [35]).

1. *(Subject Congruence) $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ and $P \equiv P'$ implies $\Gamma;\emptyset;\Delta \vdash P' \triangleright \diamond$.*
2. *(Subject Reduction) $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ with balanced Δ and $P \longrightarrow P'$ implies $\Gamma;\emptyset;\Delta' \vdash P' \triangleright \diamond$ and either (i) $\Delta = \Delta'$ or (ii) $\Delta \longrightarrow \Delta'$ with Δ' balanced.*

Proof. See Appendix A (Page 53). □

4 Behavioural Semantics for $\mathsf{HO}\pi$

We develop a theory for observational equivalence over session typed $\mathsf{HO}\pi$ processes. The theory follows the principles laid by the previous work of the authors [27,26,25]. We introduce three different bisimilarities and prove that all of them coincide with typed, reduction-closed, barbed congruence.

4.1 Labelled Transition Semantics

Labels. We define an (early) typed labelled transition system $P_1 \stackrel{\ell}{\longrightarrow} P_2$ (LTS for short) over untyped processes. Later on, using the *environmental* transition semantics, we can define a typed transition relation to formalise how a process interacts with a process in its environment. The interaction is defined on action ℓ:

$$\ell \;::=\; \tau \;\mid\; (\nu\,\tilde{m})n!\langle V\rangle \;\mid\; n?\langle V\rangle \;\mid\; n\oplus l \;\mid\; n\&l$$

The internal action is defined by label τ. Output action $(\nu\,\tilde{m})n!\langle V\rangle$ denotes the output of value V over name n with a possibly empty set of names $\tilde{m}$ being restricted (we may write $n!\langle V\rangle$ when $\tilde{m}$ is empty). Dually, the action for the value input is $n?\langle V\rangle$. We also define actions for selecting a label l, $n\oplus l$ and branching on a label n, $s\&l$. $\mathtt{fn}(\ell)$ and $\mathtt{bn}(\ell)$ denote sets of free/bound names in ℓ, resp.

The dual action relation is the symmetric relation $\asymp$ that satisfies the rules:

$$n\oplus l \asymp \overline{n}\&l \qquad (\nu\,\tilde{m}')n!\langle V\rangle \asymp \overline{n}?\langle V\rangle$$

Dual actions occur on subjects that are dual between them and carry the same object. Thus, output actions are dual to input actions and select actions is dual to branch actions.

Fig. 5 The Untyped (Early) Labelled Transition System.

$$(\lambda x. P)u \xrightarrow{\tau} P\{^{u}/x\}\ \langle\text{App}\rangle \qquad n!\langle V\rangle.P \xrightarrow{n!\langle V\rangle} P\ \langle\text{Out}\rangle \qquad n?(x).P \xrightarrow{n?\langle V\rangle} P\{^{V}/x\}\ \langle\text{In}\rangle$$

$$s \triangleleft l.P \xrightarrow{s\oplus l} P\ \langle\text{Sel}\rangle \qquad \frac{j \in I}{s \triangleright \{l_i : P_i\}_{i\in I} \xrightarrow{s\& l_j} P_j}\ \langle\text{Bra}\rangle$$

$$\frac{P \xrightarrow{\ell} P' \quad n \notin \mathtt{fn}(\ell)}{(\nu\, n)P \xrightarrow{\ell} (\nu\, n)P'}\ \langle\text{Res}\rangle \qquad \frac{P \equiv_\alpha P'' \quad P'' \xrightarrow{\ell} P'}{P \xrightarrow{\ell} P'}\ \langle\text{Alpha}\rangle \qquad \frac{P\{^{\mu X.P}/X\} \xrightarrow{\ell} P'}{\mu X.P \xrightarrow{\ell} P'}\ \langle\text{Rec}\rangle$$

$$\frac{P \xrightarrow{(\nu\, \tilde{m})n!\langle V\rangle} P' \quad m \in \mathtt{fn}(V)}{(\nu\, m)P \xrightarrow{(\nu\, m\cdot\tilde{m})n!\langle V\rangle} P'}\ \langle\text{Scope}\rangle \qquad \frac{P \xrightarrow{\ell_1} P' \qquad Q \xrightarrow{\ell_2} Q' \qquad \ell_1 \asymp \ell_2}{P \mid Q \xrightarrow{\tau} (\nu\, \mathtt{bn}(\ell_1) \cup \mathtt{bn}(\ell_2))(P' \mid Q')}\ \langle\text{Tau}\rangle$$

$$\frac{P \xrightarrow{\ell} P' \quad \mathtt{bn}(\ell) \cap \mathtt{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\ell} P' \mid Q}\ \langle\text{LPar}\rangle \qquad \frac{Q \xrightarrow{\ell} Q' \quad \mathtt{bn}(\ell) \cap \mathtt{fn}(P) = \emptyset}{P \mid Q \xrightarrow{\ell} P \mid Q'}\ \langle\text{RPar}\rangle$$

LTS over Untyped Processes. The labelled transition system (LTS) over untyped processes is defined in Figure 5. We write $P_1 \xrightarrow{\ell} P_2$ with the usual meaning. The rules are standard [27,26]. An application requires a silent step τ to substitute the application name over the application abstraction as defined in rule $\langle\text{App}\rangle$. A process with a send prefix can interact with the environment with a send action that carries a value V as in rule $\langle\text{Out}\rangle$. Dually, in rule $\langle\text{In}\rangle$ an input prefixed process can observe a receive action of a value V. Select and branch prefixed processes observe the select and branch actions in rules $\langle\text{Sel}\rangle$ and $\langle\text{Bra}\rangle$, respectively, and proceed according to the labels observed. Rule $\langle\text{Res}\rangle$ closes the LTS under the name creation operator provided that the restricted name does not occur free in the observable action. If a restricted name occurs free in an output action then the name is added as in the bound name list of the action and the continuation process performs scope opening as described in rule $\langle\text{Scope}\rangle$. Rules $\langle\text{LPar}\rangle$ and $\langle\text{RPar}\rangle$ close the LTS under the parallel operator provided that the observable action does not shared any bound names with the parallel processes. Rule $\langle\text{Tau}\rangle$ states that if two parallel processes can perform dual actions then the two actions can synchronise to observe an internal transition. Finally, rule $\langle\text{Alpha}\rangle$ closes the LTS under alpha-renaming and rule $\langle\text{Rec}\rangle$ handles recursion unfolding.

4.2 Environmental Labelled Transition System

Figure 6 defines a labelled transition relation between a triple of environments, denoted $(\Gamma_1, \Lambda_1, \Delta_1) \xrightarrow{\ell} (\Gamma_2, \Lambda_2, \Delta_2)$. It extends the transition systems in [27,26] to higher-order sessions.

Input Actions are defined by [SRv] and [ShRv] (n session or shared name respectively $n?\langle V\rangle$). We require the value V has the same type as name s and a, respectively. Furthermore we expect the resulting type tuple to contain the values that consist with value

Fig. 6 Labelled Transition Semantics for Typed Environments.

$$[\mathrm{SRv}]\ \frac{\overline{s} \notin \mathtt{dom}(\Delta) \quad \Gamma;\Lambda';\Delta' \vdash V \triangleright U}{(\Gamma;\Lambda;\Delta\cdot s:?(U);S) \xrightarrow{s?\langle V\rangle} (\Gamma;\Lambda\cdot\Lambda';\Delta\cdot\Delta'\cdot s:S)}$$

$$[\mathrm{ShRv}]\ \frac{\Gamma;\emptyset;\emptyset \vdash a \triangleright \langle U\rangle \quad \Gamma;\Lambda';\Delta' \vdash V \triangleright U}{(\Gamma;\Lambda;\Delta) \xrightarrow{a?\langle V\rangle} (\Gamma;\Lambda\cdot\Lambda';\Delta\cdot\Delta')}$$

$$[\mathrm{SSnd}]\ \frac{\begin{array}{c}\overline{s} \notin \mathtt{dom}(\Delta) \qquad \Gamma\cdot\Gamma';\Lambda';\Delta' \vdash V \triangleright U \qquad \tilde{m} = m_1 \ldots m_n \\ \Gamma';\emptyset;\Delta_i \vdash m_i \triangleright U_i \qquad \Gamma';\emptyset;\Delta'_i \vdash \overline{m}_i \triangleright U'_i \qquad \Lambda' \subseteq \Lambda \qquad (\Delta_1 \backslash \bigcup_i \Delta_i) \subseteq (\Delta\cdot s:S)\end{array}}{(\Gamma;\Lambda;\Delta\cdot s:!\langle U\rangle;S) \xrightarrow{(\nu\,\tilde{m})s!\langle V\rangle} (\Gamma\cdot\Gamma';\Lambda\backslash\Lambda';(\Delta\cdot s:S\cdot \bigcup_i \Delta'_i)\backslash\Delta')}$$

$$[\mathrm{ShSnd}]\ \frac{\begin{array}{c}\Gamma\cdot\Gamma'\cdot a:\langle U\rangle;\Lambda';\Delta' \vdash V \triangleright U \qquad \tilde{m} = m_1 \ldots m_n \\ \Gamma';\emptyset;\Delta_i \vdash m_i \triangleright U_i \qquad \Gamma';\emptyset;\Delta'_i \vdash \overline{m}_i \triangleright U_i \qquad \Lambda' \subseteq \Lambda \qquad (\Delta_1 \backslash \bigcup_i \Delta_i) \subseteq \Delta\end{array}}{(\Gamma\cdot a:\langle U\rangle;\Lambda;\Delta) \xrightarrow{(\nu\,\tilde{m})a!\langle V\rangle} (\Gamma\cdot\Gamma'\cdot a:\langle U\rangle;\Lambda\backslash\Lambda';(\Delta\cdot \bigcup_i \Delta'_i)\backslash\Delta')}$$

$$[\mathrm{Sel}]\ \frac{\overline{s} \notin \mathtt{dom}(\Delta) \quad j \in I}{(\Gamma;\Lambda;\Delta\cdot s:\oplus\{l_i:S_i\}_{i\in I}) \xrightarrow{s\oplus l_j} (\Gamma;\Lambda;\Delta\cdot s:S_j)}$$

$$[\mathrm{Bra}]\ \frac{\overline{s} \notin \mathtt{dom}(\Delta) \quad j \in I}{(\Gamma;\Lambda;\Delta\cdot s:\&\{l_i:T_i\}_{i\in I}) \xrightarrow{s\& l_j} (\Gamma;\Lambda;\Delta\cdot s:S_j)}$$

$$[\mathrm{Tau}]\ \frac{\Delta_1 \longrightarrow \Delta_2 \vee \Delta_1 = \Delta_2}{(\Gamma;\Lambda;\Delta_1) \xrightarrow{\tau} (\Gamma;\Lambda;\Delta_2)}$$

V. The condition $\overline{s} \notin \mathtt{dom}(\Delta)$ in [SRv] ensures that the dual name $\overline{s}$ should not be present in the session environment, since if it were present the only communication that could take place is the interaction between the two endpoints (using [Tau] below).

Output Actions are defined by [SSnd] and [ShSnd]. Rule [SSnd] states the conditions for observing action $(\nu\,\tilde{m})s!\langle V\rangle$ on a type tuple $(\Gamma,\Lambda,\Delta\cdot s:S)$. The session environment Δ with $s:S$ should include the session environment of sent value V, *excluding* the session environments of the name n_j in $\tilde{m}$ which restrict the scope of value V. Similarly the linear variable environment Λ' of V should be included in Λ. Scope extrusion of session names in $\tilde{m}$ requires that the dual endpoints of $\tilde{m}$ appear in the resulting session environment. Similarly for shared names in $\tilde{m}$ that are extruded. All free values used for typing V are subtracted from the resulting type tuple. The prefix of session s is consumed by the action. Similarly, an output on a shared name is described by rule [ShSnd] where we require that the name is typed with $\langle U\rangle$. Conditions for the output V are identical to those for rule [SSnd]. We sometimes annotate the output action $(\nu\,\tilde{m})n!\langle V\rangle$ with the type of V as $(\nu\,\tilde{m})n!\langle V:U\rangle$.

Other Actions Rules [Sel] and [Bra] describe actions for select and branch. The only requirements for both rules is that the dual endpoint is not present in the session environment and the action labels are present in the type. Hidden transitions defined by rule [Tau] do not change the session environment or they follow the reduction on session environments (Definition 3.6).

Proposition 4.1 (Environment Transition Weakening). *Consider the LTS for typing environments in Figure 6. If* $(\Gamma_1;\Lambda_1;\Delta_1) \stackrel{\ell}{\longmapsto} (\Gamma_2;\Lambda_2;\Delta_2)$ *then* $(\Gamma_2;\Lambda_1;\Delta_1) \stackrel{\ell}{\longmapsto} (\Gamma_2;\Lambda_2;\Delta_2)$.

Proof. The proof is by case analysis on the definition of $\stackrel{\ell}{\longmapsto}$, exploiting the structural properties (in particular, weakening) of shared environment Γ (cf. Definition 3.4). □

As a direct consequence of Proposition 4.1 we can always make an observation on a type environment without observing a change in the shared environment.

Typed Transition System We define a typed labelled transition system over typed processes, as a combination of the untyped LTS and the LTS for typed environments (cf. Figure 5 and 6):

Definition 4.1 (Typed Transition System). *We write* $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longrightarrow} \Delta_2 \vdash P_2$ *whenever* $P_1 \stackrel{\ell}{\longrightarrow} P_2$, $(\Gamma,\emptyset,\Delta_1) \stackrel{\ell}{\longrightarrow} (\Gamma,\emptyset,\Delta_2)$ *and* $\Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond$.

We extend to $\Longrightarrow$ *and* $\stackrel{\hat{\ell}}{\Longrightarrow}$ *where we write* $\Longrightarrow$ *for the reflexive and transitive closure of* $\longrightarrow$, $\stackrel{\ell}{\Longrightarrow}$ *for the transitions* $\Longrightarrow\stackrel{\ell}{\longrightarrow}\Longrightarrow$ *and* $\stackrel{\hat{\ell}}{\Longrightarrow}$ *for* $\stackrel{\ell}{\Longrightarrow}$ *if* $\ell \neq \tau$ *otherwise* $\Longrightarrow$.

4.3 Reduction-Closed, Barbed Congruence

Equivalent processes require a notion of session type confluence, defined over session environments Δ, following Definition 3.6:

Definition 4.2 (Session Environment Confluence). *We denote* $\Delta_1 \rightleftharpoons \Delta_2$ *whenever* $\exists \Delta$ *such that* $\Delta_1 \longrightarrow^* \Delta$ *and* $\Delta_2 \longrightarrow^* \Delta$.

We define the notion of typed relation over typed processes; it includes properties common to all the equivalence relations that we are going to define:

Definition 4.3 (Typed Relation). *We say that* $\Gamma;\emptyset;\Delta_1 \vdash P_1 \triangleright \diamond \ \Re \ \Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond$ *is a typed relation whenever:*

i) P_1 *and* P_2 *are closed processes;*
ii) Δ_1 *and* Δ_2 *are balanced; and*
iii) $\Delta_1 \rightleftharpoons \Delta_2$.

We write $\Gamma;\Delta_1 \vdash P_1 \ \Re \ \Delta_2 \vdash P_2$ *for* $\Gamma;\emptyset;\Delta_1 \vdash P_1 \triangleright \diamond \ \Re \ \Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond$.

Type relations relate only closed processes (i.e., processes with no free variables) with balanced session environments and the two session environments are confluent.

We define the notions of barb [32] and typed barb:

Definition 4.4 (Barbs). *Let P be a closed process.*

1. *We write $P \downarrow_n$ if $P \equiv (\nu\, \tilde{m})(n!\langle V\rangle.P_2 \mid P_3), n \notin \tilde{m}$. We write $P \Downarrow_n$ if $P \longrightarrow^* \downarrow_n$.*
2. *We write $\Gamma;\emptyset;\Delta \vdash P \downarrow_n$ if $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ with $P \downarrow_n$ and $\overline{n} \notin \Delta$. We write $\Gamma;\emptyset;\Delta \vdash P \Downarrow_n$ if $P \longrightarrow^* P'$ and $\Gamma;\emptyset;\Delta' \vdash P' \downarrow_n$.*

A barb $\downarrow_n$ is an observable on an output prefix with subject n. Similarly a weak barb $\Downarrow_n$ is a barb after a number of reduction steps. Typed barbs $\downarrow_n$ (resp. $\Downarrow_n$) occur on typed processes $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ where we require that whenever n is a session name, then the corresponding dual endpoint $\overline{n}$ is not present in the session type Δ.

To define a congruence relation we define the notion of the context $\mathbb{C}$:

Definition 4.5 (Context). *A context $\mathbb{C}$ is defined on the grammar:*

$$\begin{array}{rcl} \mathbb{C} & ::= & - \mid u!\langle V\rangle.\mathbb{C} \mid u!\langle \lambda x.\mathbb{C}\rangle.P \mid u?(x).\mathbb{C} \mid \mu X.\mathbb{C} \mid (\lambda x.\mathbb{C})u \\ & \mid & (\nu\, n)\mathbb{C} \mid \mathbb{C} \mid P \mid P \mid \mathbb{C} \mid u \triangleleft l.\mathbb{C} \mid k \triangleright \{l_1 : P_1, \cdots, l_i : \mathbb{C}, \cdots, l_n : P_n\} \end{array}$$

Notation $\mathbb{C}[P]$ replaces every hole $-$ in $\mathbb{C}$ with P.

A context is a function that takes a process and returns a new process according to the above syntax.

The first behavioural relation we define is reduction-closed, barbed congruence:

Definition 4.6 (Reduction-closed, Barbed Congruence). *Typed relation $\Gamma;\Delta_1 \vdash P_1 \,\Re\, \Delta_2 \vdash P_2$ is a barbed congruence whenever:*

1. - *If $P_1 \longrightarrow P_1'$ then there exist P_2', Δ_2' such that $P_2 \longrightarrow^* P_2'$ and $\Gamma;\Delta_1' \vdash P_1' \,\Re\, \Delta_2' \vdash P_2'$*
   - *If $P_2 \longrightarrow P_2'$ then there exist P_1', Δ_1' such that $P_1 \longrightarrow^* P_1'$ and $\Gamma;\Delta_1' \vdash P_1' \,\Re\, \Delta_2' \vdash P_2'$*
2. - *If $\Gamma;\emptyset;\Delta_1 \vdash P_1 \downarrow_s$ then $\Gamma;\emptyset;\Delta_2 \vdash P_2 \Downarrow_s$.*
   - *If $\Gamma;\emptyset;\Delta_2 \vdash P_2 \downarrow_s$ then $\Gamma;\emptyset;\Delta_1 \vdash P_1 \Downarrow_s$.*
3. *$\forall \mathbb{C}$, then there exist Δ_1'', Δ_2'' such that $\Gamma;\Delta_1'' \vdash \mathbb{C}[P_1] \,\Re\, \Delta_2'' \vdash \mathbb{C}[P_2]$*

The largest such congruence is denoted with $\cong$.

Reduction-closed, barbed congruence is closed under reduction semantics and preserves barbs under any context, i.e., no barb observer can distinguish between two related processes.

4.4 Context Bisimulation

The second behavioural relation we define is the labelled characterisation of reduction-closed, barbed congruence, called *context bisimulation* [46]:

Definition 4.7 (Context Bisimulation). *Typed relation $\Re$ is a* context bisimulation *if for all $\Gamma;\Delta_1 \vdash P_1 \,\Re\, \Delta_2 \vdash P_2$,*

1. *Whenever $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\, \tilde{m}_1)n!\langle V_1\rangle}{\longrightarrow} \Delta_1' \vdash P_2$ there exist Q_2, V_2, and Δ_2' such that*

$$\Gamma;\Delta_2 \vdash Q_1 \stackrel{(\nu\, \tilde{m}_2)n!\langle V_2\rangle}{\Longrightarrow} \Delta_2' \vdash Q_2$$

and $\forall R$ with $\{x\} = \mathtt{fv}(R)$, then

$$\Gamma;\Delta_1'' \vdash (\nu\, \tilde{m}_1)(P_2 \mid R\{^{V_1}/x\}) \,\Re\, \Delta_2'' \vdash (\nu\, \tilde{m}_2)(Q_2 \mid R\{^{V_2}/x\}).$$

2. *For all* $\Gamma;\Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_1' \vdash P_2$ *such that* $\ell \neq (\nu\, \tilde{m})n!\langle V\rangle$, *there exist* Q_2 *and* Δ_2' *such that*
$$\Gamma;\Delta_2 \vdash Q_1 \stackrel{\hat{\ell}}{\Longrightarrow} \Delta_2' \vdash Q_2$$
and $\Gamma;\Delta_1' \vdash P_2 \;\Re\; \Delta_2' \vdash Q_2$.
3. *The symmetric cases of 1 and 2.*

The Knaster-Tarski theorem ensures that the largest context bisimulation exists, it is called context bisimilarity *and is denoted by* $\approx$.

4.5 Higher-Order Bisimulation and Characteristic Bisimulation ($\approx^H$ / $\approx^C$)

In the general case, contextual bisimulation is a hard relation to compute due to:

i) the universal quantifier over contexts in the output case (Clause 1 in Definition 4.7); and
ii) a higher order input prefix can observe infinitely many different input actions, since infinitely many different processes can match the session type of an input prefix.

To reduce the burden of the contextual bisimulation we take the following two steps:

(a) we replace Clause 1 in Definition 4.7 with a clause involving a more tractable process closure; and
(b) we refine the transition rule for input in the LTS so to define a bisimulation relation without observing infinitely many actions on the same input prefix.

Trigger Processes with Session Communication. Concerning (a), we exploit session types. First observe that closure $R\{^V/_x\}$ in Clause 1 in Definition 4.7 is context bisimilar to the process:

$$P = (\nu\, s)((\lambda z.\, z?(x).R)\, s \mid \overline{s}!\langle V\rangle.\mathbf{0}) \tag{1}$$

In fact, we do have $P \approx R\{^V/_x\}$, since application and session transitions are deterministic. Now let us consider process T_V below, where t is a fresh name:

$$T_V = t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V\rangle.\mathbf{0}) \tag{2}$$

Process T_V can input the class of abstractions $\lambda z.\, z?(x).R$ and can simulate the closure of (1):

$$T_V \xrightarrow{t?\langle \lambda z.\, z?(x).R\rangle} P \approx R\{^V/_x\} \tag{3}$$

Processes such as T_V input a value at a fresh name; we will use this class of ***trigger processes*** to define a refined bisimilarity without the demanding output Clause 1 in Definition 4.7. Given a fresh name t, we write:

$$t \Leftarrow V = t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V\rangle.\mathbf{0})$$

We note that in contrast to previous approaches [50,22] our trigger processes do *not* use recursion or replication. This is crucial to preserve linearity of session names.

Characteristic Processes and Values. Concerning point (b), we limit the possible input abstractions $\lambda x.P$ by exploiting session types. We introduce the key concept of ***characteristic process/value***, which is the simplest process/value that can inhabit a type. As an example, consider $S = ?(S_1 \to \diamond); !\langle S_2 \rangle; \mathsf{end}$. Type S is a session type that first inputs an abstraction (from type S_1 to a process), then outputs a value of type S_2, and terminates. Then, the following process:

$$Q = u?(x).(u!\langle s_2 \rangle.\mathbf{0} \mid x\, s_1)$$

is a characteristic process for S along name u. In fact, it is easy to see that Q is well-typed by session type S. The following definition formalizes this intuition.

Definition 4.8 (Characteristic Process). *Let name u and type U. Then we define the* characteristic process*: $[\![U]\!]^u$ and the* characteristic value *$[\![U]\!]_{\mathsf{c}}$ as:*

$$
\begin{array}{rclcrcl}
[\![?(U);S]\!]^u & \stackrel{\mathsf{def}}{=} & u?(x).([\![S]\!]^u \mid [\![U]\!]^x) & & [\![\langle S \rangle]\!]^u & \stackrel{\mathsf{def}}{=} & u!\langle [\![S]\!]_{\mathsf{c}} \rangle.\mathbf{0} \\
[\![!\langle U \rangle;S]\!]^u & \stackrel{\mathsf{def}}{=} & u!\langle [\![U]\!]_{\mathsf{c}} \rangle.[\![S]\!]^u & & [\![\langle L \rangle]\!]^u & \stackrel{\mathsf{def}}{=} & u!\langle [\![L]\!]_{\mathsf{c}} \rangle.\mathbf{0} \\
[\![\oplus\{l : S\}]\!]^u & \stackrel{\mathsf{def}}{=} & u \triangleleft l.[\![S]\!]^u & [\![C \to \diamond]\!]^x \stackrel{\mathsf{def}}{=} & [\![C \multimap \diamond]\!]^x & \stackrel{\mathsf{def}}{=} & x\,[\![C]\!]_{\mathsf{c}} \\
[\![\&\{l_i : S_i\}_{i \in I}]\!]^u & \stackrel{\mathsf{def}}{=} & u \triangleright \{l_i : [\![S_i]\!]^u\}_{i \in I} & & & & \\
[\![\mathsf{t}]\!]^u & \stackrel{\mathsf{def}}{=} & X_{\mathsf{t}} & & [\![S]\!]_{\mathsf{c}} & \stackrel{\mathsf{def}}{=} & s \quad s \text{ fresh} \\
[\![\mu \mathsf{t}.S]\!]^u & \stackrel{\mathsf{def}}{=} & \mu X_{\mathsf{t}}.[\![S]\!]^u & [\![\langle S \rangle]\!]_{\mathsf{c}} \stackrel{\mathsf{def}}{=} & [\![\langle L \rangle]\!]_{\mathsf{c}} & \stackrel{\mathsf{def}}{=} & a \quad a \text{ fresh} \\
[\![\mathsf{end}]\!]^u & \stackrel{\mathsf{def}}{=} & \mathbf{0} & [\![C \to \diamond]\!]_{\mathsf{c}} \stackrel{\mathsf{def}}{=} & [\![C \multimap \diamond]\!]_{\mathsf{c}} & \stackrel{\mathsf{def}}{=} & \lambda x.[\![C]\!]^x
\end{array}
$$

Proposition 4.2. *Characteristic processes and values are inhabitants of their associated type:*

- $\Gamma; \emptyset; \Delta \cdot u : S \vdash [\![S]\!]^u \triangleright \diamond$
- $U = \langle S \rangle$ *or* $U = \langle L \rangle$ *implies* $\Gamma \cdot u : U; \emptyset; \Delta \vdash [\![U]\!]^u \triangleright \diamond$
- $\Gamma; \emptyset; \Delta \vdash [\![U]\!]_{\mathsf{c}} \triangleright U$

Proof. By induction on the definition of $[\![S]\!]^u$ and $[\![U]\!]^u$. □

Corollary 4.1. *If $\Gamma; \emptyset; \Delta \vdash [\![C]\!]^u \triangleright \diamond$ then $\Gamma; \emptyset; \Delta \vdash u \triangleright C$.*

We use the characteristic value $[\![U]\!]_{\mathsf{c}}$ to limit input transitions. Following the same reasoning as (1)–(3), we can define an alternative trigger process, called ***characteristic trigger process*** with type U to replace Clause 1 in Definition 4.7.

$$t \Leftarrow V : U \stackrel{\mathsf{def}}{=} t?(x).(\nu\, s)([\![?(U); \mathsf{end}]\!]^s \mid \overline{s}!\langle V \rangle.\mathbf{0}) \tag{4}$$

Thus, in contrast to the trigger process in (2), the characteristic trigger process in (4) does not involve a higher-order communication on t.

To refine the input transition system, we need to observe an additional value:

$$\lambda x.\, t?(y).(y\, x)$$

called the ***trigger value***. This is necessary, because it turns out that a characteristic value alone as the observable input is not enough to define a sound bisimulation. Roughly speaking, the trigger value is used to observe/simulate application processes.

The intuition for usage of the trigger is demonstrated in the next example.

Example 4.1. First we demonstrate that observing a characteristic value input alone is not sufficient to define a sound bisimulation closure. Consider typed processes P_1, P_2:

$$P_1 = s?(x).(x\,s_1 \mid x\,s_2) \qquad\qquad P_2 = s?(x).(x\,s_1 \mid s_2?(y).\mathbf{0}) \tag{5}$$

with

$$\Gamma;\emptyset;\Delta\cdot s : ?((?(C);\mathsf{end}){\rightarrow}\diamond);\mathsf{end} \vdash P_i \triangleright \diamond \qquad (i \in \{1,2\}).$$

If the above processes input and substitute over x the characteristic value

$$[\![(?(C);\mathsf{end}){\rightarrow}\diamond]\!]_{\mathsf{c}} = \lambda x.\, x?(y).\mathbf{0}$$

then both processes evolve into:

$$\Gamma;\emptyset;\Delta \vdash s_1?(y).\mathbf{0} \mid s_2?(y).\mathbf{0} \triangleright \diamond$$

therefore becoming context bisimilar. However, the processes in (5) are clearly *not* context bisimilar: there exist many input actions which may be used to distinguish them. For example, if P_1 and P_2 input

$$\lambda x.\, (\nu\, s_3)(a!\langle s_3\rangle.x?(y).\mathbf{0})$$

with $\Gamma;\emptyset;\Delta \vdash s \triangleright \mathsf{end}$, then their derivatives are not bisimilar.

Observing only the characteristic value results in an over-discriminating bisimulation. However, if a trigger value, $\lambda x.\, t?(y).(y\,x)$ is received on s, then we can distinguish processes in (5):

$$\begin{aligned}
\Gamma;\Delta \vdash P_1 &\overset{s?\langle \lambda x.\, t?(y).(y\,x)\rangle}{\Longrightarrow} \Delta' \vdash t?(x).(x\,s_1) \mid t?(x).(x\,s_2)\\
\Gamma;\Delta \vdash P_2 &\overset{s?\langle \lambda x.\, t?(y).(y\,x)\rangle}{\Longrightarrow} \Delta'' \vdash t?(x).(x\,s_1) \mid s_2?(y).\mathbf{0}
\end{aligned}$$

One question that arises here is whether the trigger value is enough to distinguish two processes, hence no need of characteristic values as the input. This is not the case since the trigger value alone also results in an over-discriminating bisimulation relation. In fact the trigger value can be observed on any input prefix of *any type*. For example, consider the following processes:

$$\Gamma;\emptyset;\Delta \vdash (\nu\, s)(n?(x).(x\,s) \mid \overline{s}!\langle \lambda x.\, P\rangle.\mathbf{0}) \triangleright \diamond \tag{6}$$

$$\Gamma;\emptyset;\Delta \vdash (\nu\, s)(n?(x).(x\,s) \mid \overline{s}!\langle \lambda x.\, Q\rangle.\mathbf{0}) \triangleright \diamond \tag{7}$$

if processes in (6)/(7) input the trigger value, we obtain processes:

$$\begin{aligned}
&\Gamma;\emptyset;\Delta' \vdash (\nu\, s)(t?(x).(x\,s) \mid \overline{s}!\langle \lambda x.\, P\rangle.\mathbf{0}) \triangleright \diamond\\
&\Gamma;\emptyset;\Delta' \vdash (\nu\, s)(t?(x).(x\,s) \mid \overline{s}!\langle \lambda x.\, Q\rangle.\mathbf{0}) \triangleright \diamond
\end{aligned}$$

thus we can easily derive a bisimulation closure if we assume a bisimulation definition that allows only trigger value input.

But if processes in (6)/(7) input the characteristic value $\lambda z.\, z?(x).(x\,m)$, then they would become:

$$\begin{aligned}
\Gamma;\emptyset;\Delta \vdash (\nu\, s)(s?(x).(x\,m) \mid \overline{s}!\langle \lambda x.\, P\rangle.\mathbf{0}) &\approx \Delta \vdash P\{m/x\}\\
\Gamma;\emptyset;\Delta \vdash (\nu\, s)(s?(x).(x\,m) \mid \overline{s}!\langle \lambda x.\, Q\rangle.\mathbf{0}) &\approx \Delta \vdash Q\{m/x\}
\end{aligned}$$

which are not bisimilar if $P\{m/x\} \not\approx^{H} Q\{m/x\}$.

We now define the *refined* typed LTS. The new LTS is defined by considering a transition rule for input in which admitted values are trigger or characteristic values: We formalise the restricted input action with the definition of a new environment transition relation:

$$(\Gamma,\Lambda_1,\Delta_1) \stackrel{\ell}{\longmapsto} (\Gamma,\Lambda_2,\Delta_2)$$

The new rule is defined on top of the rules in Figure 6:

Definition 4.9 (Refined Input Environment LTS).

$$[RRv]\quad \frac{(\Gamma_1;\Lambda_1;\Delta_1) \stackrel{n?\langle V\rangle}{\longrightarrow} (\Gamma_2;\Lambda_2;\Delta_2) \qquad \begin{array}{c}(V \equiv \lambda z.\, t?(x).(x\,z) \wedge t\ fresh)\\ \vee\ (V \equiv [\![U]\!]_{\mathsf{c}}) \vee V = m\end{array}}{(\Gamma_1;\Lambda_1;\Delta_1) \stackrel{n?\langle V\rangle}{\longmapsto} (\Gamma_2;\Lambda_1;\Delta_2)}$$

Rule [RRv] refines the input action to carry only a characteristic value (fresh name or abstraction) or a trigger value on a fresh name t. This rule is defined on top of rules [SRv] and [ShRv] in Figure 6. The new environment transition system $\stackrel{\ell}{\longmapsto}$ uses rule [RRv] as input rule. All other defining cases of environment LTS $\stackrel{\ell}{\longmapsto}$ remain the same as in Figure 6.

The new typed relation derived from the $\stackrel{\ell}{\longmapsto}$ environment LTS is defined as:

Definition 4.10 (Restricted Typed Transition). *We write $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longmapsto} \Delta_2 \vdash P_2$ whenever $P_1 \stackrel{\ell}{\longrightarrow} P_2$, $(\Gamma,\emptyset,\Delta_1) \stackrel{\ell}{\longmapsto} (\Gamma,\emptyset,\Delta_2)$ and $\Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond$.*
We extend to $\Longmapsto$ and $\stackrel{\hat{\ell}}{\Longmapsto}$ in the standard way.

Lemma 4.1 (Invariant). *If $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longmapsto} \Delta_2 \vdash P_2$ then $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longrightarrow} \Delta_2 \vdash P_2$.*

Proof. The proof is straightforward from the definition of rule [RRv].

The next definition formalises the notion of a trigger process.

Definition 4.11 (Trigger Process). *Let t, V, and U be a name, a value, and a type, respectively. We have:*

$$\begin{array}{rcl}\textit{Trigger Process}\ \ t \Leftarrow V & \stackrel{\mathsf{def}}{=} & t?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle V\rangle.\mathbf{0})\\ \textit{Characteristic Trigger Process}\ \ t \Leftarrow V : U & \stackrel{\mathsf{def}}{=} & t?(x).(\nu\ s)([\![?(U);\mathsf{end}]\!]^{s} \mid \overline{s}!\langle V\rangle.\mathbf{0})\end{array}$$

The Two Bisimulations. We now define higher-order bisimulation, a more tractable bisimulation for HO and $\mathsf{HO}\pi$. The two bisimulations differ on the fact that they use the different trigger processes: $t \Leftarrow V$ and $t \Leftarrow V : U$.

Definition 4.12 (Higher-Order Bisimulation). *Typed relation $\Re$ is a* higher-Order bisimulation *if for all $\Gamma;\Delta_1 \vdash P_1\ \Re\ \Delta_2 \vdash Q_1$,*

1. *Whenever* $\Gamma;\Delta_1 \vdash P_1 \overset{(\nu\,\tilde{m}_1)n!\langle V_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$ *there exist* Q_2*,* V_2*,* Δ_2' *such that*

$$\Gamma;\Delta_2 \vdash Q_1 \overset{(\nu\,\tilde{m}_2)n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$$

   and, for a fresh t,

$$\Gamma;\Delta_1'' \vdash (\nu\,\tilde{m}_1)(P_2 \mid t \Leftarrow V_1) \;\mathfrak{R}\; \Delta_2'' \vdash (\nu\,\tilde{m}_2)(Q_2 \mid t \Leftarrow V_2).$$

2. *For all* $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$ *such that* $\ell \neq (\nu\,\tilde{m})n!\langle V\rangle$*, there exist* $\exists Q_2$ *and* Δ_2' *such that*

$$\Gamma;\Delta_1 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$$

   and $\Gamma;\Delta_1' \vdash P_2 \;\mathfrak{R}\; \Delta_2' \vdash Q_2$.
3. *The symmetric cases of 1 and 2.*

The Knaster-Tarski theorem ensures that the largest higher-order bisimulation exists; it is called higher-order bisimilarity *and is denoted by* $\approx^H$.

The higher-order bisimulation definition uses higher order input guarded triggers, thus it cannot be used as an equivalence relation for the π sub-calculus. An alternative definition of the bisimulation—based on characteristic output triggers—solves this problem.

Definition 4.13 (Characteristic Bisimulation). *Typed relation* $\mathfrak{R}$ *is a* characteristic bisimulation *if whenever* $\Gamma;\Delta_1 \vdash P_1 \;\mathfrak{R}\; \Delta_2 \vdash Q_1$ *implies:*

1. *Whenever* $\Gamma;\Delta_1 \vdash P_1 \overset{(\nu\,\tilde{m}_1)n!\langle V_1:U\rangle}{\longmapsto} \Delta_1' \vdash P_2$ *there exist* Q_2*,* V_2*, and* Δ_2' *such that*

$$\Gamma;\Delta_2 \vdash Q_1 \overset{(\nu\,\tilde{m}_2)n!\langle V_2:U\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$$

   and, for a fresh t,

$$\Gamma;\Delta_1'' \vdash (\nu\,\tilde{m}_1)(P_2 \mid t \Leftarrow V_1 : U) \;\mathfrak{R}\; \Delta_2'' \vdash (\nu\,\tilde{m}_2)(Q_2 \mid t \Leftarrow V_2 : U).$$

2. *For all* $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$ *such that* $\ell \neq (\nu\,\tilde{m})n!\langle V\rangle$*, there exist* $\exists Q_2$ *and* Δ_2' *such that*

$$\Gamma;\Delta_1 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$$

   and $\Gamma;\Delta_1' \vdash P_2 \;\mathfrak{R}\; \Delta_2' \vdash Q_2$.
3. *The symmetric cases of 1 and 2.*

The Knaster-Tarski theorem ensures that the largest bisimulation exists; it is called characteristic bisimilarity *and is denoted by* $\approx^C$.

The next result clarifies our choice of restricting higher-order input actions with input triggers and characteristic processes: if two processes P and Q are bisimilar under the substitution of the characteristic abstraction and the trigger input, then P and Q are bisimilar under any abstraction substitution.

Lemma 4.2 (Process Substitution). *If*

1. $\Gamma;\Delta'_1 \vdash P\{\lambda z.\, t?(y).(y\, z)/x\} \approx^H \Delta'_2 \vdash Q\{\lambda z.\, t?(y).(y\, z)/x\}$*, for some fresh* t*.*
2. $\Gamma;\Delta''_1 \vdash P\{[\![U]\!]_{\mathsf{c}}/x\} \approx^H \Delta''_2 \vdash Q\{[\![U]\!]_{\mathsf{c}}/x\}$*, for some* U*.*

then $\forall R$ *such that* $\mathtt{fv}(R) = z$

$$\Gamma;\Delta_1 \vdash P\{\lambda z.R/x\} \approx^H \Delta_2 \vdash Q\{\lambda z.R/x\}$$

Proof. The details of the proof can be found in Lemma B.3 (Page 58). □

We now state our main theorem: typed bisimilarities collapse. The following theorem justifies our choices for the bisimulation relations, since they coincide between them and they also coincide with reduction closed, barbed congruence.

Theorem 4.1 (Coincidence). *Relations* $\approx, \approx^C, \approx^H$ *and* $\cong$ *coincide.*

Proof. The full details of the proof are in Appendix B.1. There, the proof is split into a series of lemmas:

- Lemma B.1 establishes $\approx^H = \approx^C$.
- Lemma B.4 exploits the process substitution result (Lemma 4.2) to prove that $\approx^H \subseteq \approx$.
- Lemma B.5 shows that $\approx$ is a congruence which implies $\approx \subseteq \cong$.
- Lemma B.8 shows that $\cong \subseteq \approx^H$, using the technique developed in [18].

The formulation of input triggers in the bisimulation relation allows us to prove the latter result without using a matching operator. □

We now define internal deterministic transitions as those associated to session synchronizations or to β-reductions:

Definition 4.14 (Deterministic Transition). *Let* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *be a balanced* $\mathsf{HO}\pi$ *process. Transition* $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash P'$ *is called:*

- Session transition *whenever the untyped transition* $P \xrightarrow{\tau} P'$ *is derived using rule* $\langle \text{Tau} \rangle$ *(where* $\mathtt{subj}(\ell_1)$ *and* $\mathtt{subj}(\ell_2)$ *in the premise are dual endpoints), possibly followed by uses of* $\langle \text{Alpha} \rangle$*,* $\langle \text{Res} \rangle$*,* $\langle \text{Rec} \rangle$*, or* $\langle \text{Par}_L \rangle / \langle \text{Par}_R \rangle$*.*
- β−transition *whenever the untyped transition* $P \xrightarrow{\tau} P'$ *is derived using rule* $\langle \text{App} \rangle$*, possibly followed by uses of* $\langle \text{Alpha} \rangle$*,* $\langle \text{Res} \rangle$*,* $\langle \text{Rec} \rangle$*, or* $\langle \text{Par}_L \rangle / \langle \text{Par}_R \rangle$*.*

We write $\Gamma;\Delta \vdash P \overset{\tau_{\mathsf{s}}}{\longmapsto} \Delta' \vdash P'$ *and* $\Gamma;\Delta \vdash P \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ *to denote session and* β*-transitions, resp. Also,* $\Gamma;\Delta \vdash P \overset{\tau_{\mathsf{d}}}{\longmapsto} \Delta' \vdash P'$ *denotes either a session transition or a* β*−transition.*

Deterministic transitions imply the τ-inertness property, which is a property that ensures behavioural invariance on deterministic transitions.

Proposition 4.3 (τ-inertness). *Let* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *be a balanced* $\mathsf{HO}\pi$ *process. Then*

- $\Gamma;\Delta \vdash P \overset{\tau_{\mathsf{d}}}{\longmapsto} \Delta' \vdash P'$ *implies* $\Gamma;\Delta \vdash P \approx^H \Delta' \vdash P'$*.*

– $\Gamma;\Delta \vdash P \stackrel{\tau_d}{\Longmapsto} \Delta' \vdash P'$ *implies* $\Gamma;\Delta \vdash P \approx^H \Delta' \vdash P'$.

Proof. The proof for Part 1 relies on the fact that processes of the form $\Gamma;\emptyset;\Delta \vdash s!\langle V\rangle.P_1 \mid \overline{s}?(x).P_2$ cannot have any typed transition observables (for both s and $\overline{s}$ are defined in Δ) and the fact that bisimulation is a congruence. See details in Appendix B.2 (Page 70). The proof for Part 2 is straightforward from Part 1. □

Processes that do not use shared names are inherently deterministic, and so they enjoy τ*-inertness* (in the sense of [17]).

Corollary 4.2 **($\mathsf{C}^{-\mathsf{sh}}$ τ-inertness).** *Let* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *be an* $\mathsf{C}^{-\mathsf{sh}}$ *process.*

– $\Gamma;\Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ *if and only if* $\Gamma;\Delta \vdash P \stackrel{\tau_d}{\longmapsto} \Delta' \vdash P'$.
– $\Gamma;\Delta \vdash P \stackrel{\tau_d}{\longmapsto} \Delta' \vdash P'$ *implies* $\Gamma;\Delta \vdash P \approx^H \Delta' \vdash P'$.

Lemma 4.3 **(Up-to Deterministic Transition).** *Let* $\Gamma;\Delta_1 \vdash P_1 \;\Re\; \Delta_2 \vdash Q_1$ *such that if whenever:*

1. $\forall (\nu\, \tilde{m}_1)n!\langle V_1\rangle$ *such that* $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\, \tilde{m}_1)n!\langle V_1\rangle}{\longmapsto} \Delta_3 \vdash P_3$ *implies that* $\exists Q_2, V_2$ *such that*

$$\Gamma;\Delta_2 \vdash Q_1 \stackrel{(\nu\, \tilde{m}_2)n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$$

and

$$\Gamma;\Delta_3 \vdash P_3 \stackrel{\tau_d}{\Longmapsto} \Delta_1' \vdash P_2$$

and for fresh t*:*

$$\Gamma;\Delta_1'' \vdash (\nu\, \tilde{m}_1)(P_2 \mid t \Leftarrow V_1) \;\Re\; \Delta_2'' \vdash (\nu\, \tilde{m}_2)(Q_2 \mid t \Leftarrow V_2)$$

2. $\forall \ell \neq (\nu\, \tilde{m})n!\langle V\rangle$ *such that* $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longmapsto} \Delta_3 \vdash P_3$ *implies that* $\exists Q_2$ *such that*

$$\Gamma;\Delta_1 \vdash Q_1 \stackrel{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$$

and

$$\Gamma;\Delta_3 \vdash P_3 \stackrel{\tau_d}{\Longmapsto} \Delta_1' \vdash P_2$$

and $\Gamma;\Delta_1' \vdash P_2 \;\Re\; \Delta_2' \vdash Q_2$
3. *The symmetric cases of 1 and 2.*

Then $\Re \subseteq \approx^H$.

Proof. The proof is easy by considering the closure

$$\Re^{\stackrel{\tau_d}{\Longmapsto}} = \{\Gamma;\Delta_1' \vdash P_2, \Delta_2' \vdash Q_1 \;\mid\; \Gamma;\Delta_1 \vdash P_1 \;\Re\; \Delta_2' \vdash Q_1, \Gamma;\Delta_1 \vdash P_1 \stackrel{\tau_d}{\Longmapsto} \Delta_1' \vdash P_2\}$$

We verify that $\Re^{\stackrel{\tau_d}{\Longmapsto}}$ is a bisimulation with the use of Proposition 4.3. □

5 Typed Encodings

This section defines the formal notion of *encoding*, extending to a typed setting existing criteria for untyped processes (as in, e.g. [36,37,38,16,28,54]). We first define a typed calculus parameterised by a syntax, operational semantics, and typing.

Definition 5.1 (Typed Calculus). *A* typed calculus $\mathcal{L}$ *is a tuple:*

$$\langle \mathsf{C}, \mathcal{T}, \overset{\ell}{\longmapsto}, \approx, \vdash \rangle$$

where C *and* $\mathcal{T}$ *are sets of processes and types, respectively; and* $\overset{\ell}{\longmapsto}$, $\approx$, *and* $\vdash$ *denote a transition system, a typed equivalence, and a typing system for* C*, respectively.*

Our notion of encoding considers a mapping on processes, types, and transition labels.

Definition 5.2 (Typed Encoding). *Let* $\mathcal{L}_i = \langle \mathsf{C}_i, \mathcal{T}_i, \overset{\ell}{\longmapsto}_i, \approx_i, \vdash_i \rangle$ *(*$i = 1,2$*) be typed calculi, and let* $\mathcal{A}_i$ *be the set of labels used in relation* $\overset{\ell}{\longmapsto}_i$*. Given mappings* $[\![\cdot]\!] : \mathsf{C}_1 \to \mathsf{C}_2$, $(\!(\cdot)\!) : \mathcal{T}_1 \to \mathcal{T}_2$*, and* $\{\!\{\cdot\}\!\} : \mathcal{A}_1 \to \mathcal{A}_2$*, we write* $\langle [\![\cdot]\!], (\!(\cdot)\!), \{\!\{\cdot\}\!\} \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *to denote the* typed encoding *of* $\mathcal{L}_1$ *into* $\mathcal{L}_2$*.*

We will often assume that $(\!(\cdot)\!)$ extends to typing environments as expected. This way, e.g., $(\!(\Delta \cdot u : S)\!) = (\!(\Delta)\!) \cdot u : (\!(S)\!)$.

We introduce two classes of typed encodings, which serve different purposes. Both consist of syntactic and semantic criteria proposed for untyped processes [37,16,28], here extended to account for (higher-order) session types. First, for stating stronger positive encodability results, we define the notion of *precise* encodings. Then, with the aim of proving strong non-encodability results, precise encodings are relaxed into the weaker *minimal* encodings.

We first state the syntactic criteria. Let σ denote a substitution of names for names (a renaming, in the usual sense). Given environments Δ and Γ, we write $\sigma(\Delta)$ and $\sigma(\Gamma)$ to denote the effect of applying σ on the domains of Δ and Γ (clearly, $\sigma(\Gamma)$ concerns only shared names in Γ: process and recursion variables in Γ are not affected by σ).

Definition 5.3 (Syntax Preserving Encoding). *We say that the typed encoding* $\langle [\![\cdot]\!], (\!(\cdot)\!), \{\!\{\cdot\}\!\} \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *is* syntax preserving *if it is:*

1. Homomorphic wrt parallel*, if* $(\!(\Gamma)\!); \emptyset; (\!(\Delta_1 \cdot \Delta_2)\!) \vdash_1 [\![P_1 \mid P_2]\!] \triangleright \diamond$ *then* $(\!(\Gamma)\!); \emptyset; (\!(\Delta_1)\!) \cdot (\!(\Delta_2)\!) \vdash_2 [\![P_1]\!] \mid [\![P_2]\!] \triangleright \diamond$.
2. Compositional wrt restriction*, if* $(\!(\Gamma)\!); \emptyset; (\!(\Delta)\!) \vdash_1 [\![(\nu\, n)P]\!] \triangleright \diamond$ *then* $(\!(\Gamma)\!); \emptyset; (\!(\Delta)\!) \vdash_2 (\nu\, n)[\![P]\!] \triangleright \diamond$.
3. Name invariant*, if* $(\!(\sigma(\Gamma))\!); \emptyset; (\!(\sigma(\Delta))\!) \vdash_1 [\![\sigma(P)]\!] \triangleright \diamond$ *then* $\sigma((\!(\Gamma)\!)); \emptyset; \sigma((\!(\Delta)\!)) \vdash_2 \sigma([\![P]\!]) \triangleright \diamond$*, for any injective renaming of names* σ*.*

Homomorphism wrt parallel composition (used in, e.g., [37,38]) expresses that encodings should preserve the distributed topology of source processes. This criteria is appropriate for both encodability and non encodability results; in our setting, it admits an elegant formulation, also induced by rules for typed composition. Compositionality wrt restriction is also naturally supported by typing and turns out to be useful in our encodability results (see the following section). Our name invariance criteria follows the one given in [16,28]. Next we define semantic criteria for typed encodings.

Definition 5.4 (Semantic Preserving Encoding). *Let* $\mathcal{L}_i = \langle \mathsf{C}_i, \mathcal{T}_i, \stackrel{\ell}{\longmapsto}, \approx_i, \vdash_i \rangle$ $(i = 1,2)$ *be typed calculi. We say that* $\langle [\![\cdot]\!], (\!(\cdot)\!), \{\!\{\cdot\}\!\} \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *is a* semantic preserving encoding *if it satisfies the properties below. Given a label* $\ell \neq \tau$*, we write* $\mathtt{subj}(\ell)$ *to denote the* subject *of the action.*

1. Type Preservation*: if* $\Gamma; \emptyset; \Delta \vdash_1 P \triangleright \diamond$ *then* $(\!(\Gamma)\!); \emptyset; (\!(\Delta)\!) \vdash_2 [\![P]\!] \triangleright \diamond$*, for any* P *in* C_1.
2. Subject preserving*: if* $\mathtt{subj}(\ell) = u$ *then* $\mathsf{sub}(\{\!\{\ell\}\!\}) = u$.
3. Operational Correspondence*: If* $\Gamma; \emptyset; \Delta \vdash_1 P \triangleright \diamond$ *then*
   (a) *Completeness: If* $\Gamma; \Delta \vdash_1 P \stackrel{\ell_1}{\longmapsto}_1 \Delta' \vdash_1 P'$ *then* $\exists \ell_2, Q, \Delta''$ *s.t.*
   (i) $(\!(\Gamma)\!); (\!(\Delta)\!) \vdash_2 [\![P]\!] \stackrel{\ell_2}{\Longmapsto}_2 (\!(\Delta'')\!) \vdash_2 Q$*, (ii)* $\ell_2 = \{\!\{\ell_1\}\!\}$*, and*
   (iii) $(\!(\Gamma)\!); (\!(\Delta'')\!) \vdash_2 Q \approx_2 (\!(\Delta')\!) \vdash_2 [\![P']\!]$.
   (b) *Soundness: If* $(\!(\Gamma)\!); (\!(\Delta)\!) \vdash_2 [\![P]\!] \stackrel{\ell_2}{\Longmapsto}_2 (\!(\Delta'')\!) \vdash_2 Q$ *then* $\exists \ell_1, P', \Delta'$ *s.t.*
   (i) $\Gamma; \Delta \vdash_1 P \stackrel{\ell_1}{\longmapsto}_1 \Delta' \vdash_1 P'$*, (ii)* $\ell_2 = \{\!\{\ell_1\}\!\}$*, and (iii)* $(\!(\Gamma)\!); (\!(\Delta')\!) \vdash_2 [\![P']\!] \approx_2 (\!(\Delta'')\!) \vdash_2 Q$.
4. Full Abstraction:
   $\Gamma; \Delta_1 \vdash_1 P \approx_1 \Delta_2 \vdash_1 Q$ *if and only if* $(\!(\Gamma)\!); (\!(\Delta_1)\!) \vdash_2 [\![P]\!] \approx_2 (\!(\Delta_2)\!) \vdash_2 [\![Q]\!]$.

Type preservation is a distinguishing criteria in our notion of encoding: it enables us to focus on encodings which retain the communication structures denoted by (session) types. The other semantic criteria build upon analogous definitions in the untyped setting, as we explain now. Operational correspondence, standardly divided into completeness and soundness criteria, is based in the formulation given in [16,28]. Soundness ensures that the source process is mimicked by its associated encoding; completeness concerns the opposite direction. Rather than reductions, completeness and soundness rely on the typed LTS of Definition 4.10; labels are considered up to mapping $\{\!\{\cdot\}\!\}$, which offers flexibility when comparing different subcalculi of $\mathsf{HO}\pi$. We require that $\{\!\{\cdot\}\!\}$ preserves communication subjects, in accordance with the criteria in [28]. It is worth stressing that the operational correspondence statements given in the next section for our encodings are tailored to the specifics of each encoding, and so they are actually stronger than the criteria given above. Finally, following [48,38,57], we consider full abstraction as an encodability criteria: this results into stronger encodability results. From the criteria in Definition 5.3 and Definition 5.4 we have the following derived criteria:

Proposition 5.1 (Derived Criteria). *Let* $\langle [\![\cdot]\!], (\!(\cdot)\!), \{\!\{\cdot\}\!\} \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *be a typed encoding. Suppose the encoding is both operational complete (cf. Definition 5.4-3(a)) and subject preserving (cf. Definition 5.4-2). Then, it is also* barb preserving*, i.e.,* $\Gamma; \Delta \vdash_1 P \downarrow_n$ *implies* $(\!(\Gamma)\!); (\!(\Delta)\!) \vdash_2 [\![P]\!] \Downarrow_n$.

Proof. The proof follows from the definition of barbs, operational completeness, and subject preservation. □

We may now define *precise* and *minimal* typed criteria:

Definition 5.5 (Typed Encodings: Precise and Minimal). *We say that the typed encoding* $\langle [\![\cdot]\!], (\!(\cdot)\!), \{\!\{\cdot\}\!\} \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *is*

(i) precise*, if it is both syntax and semantic preserving (cf. Definition 5.3 and Definition 5.4).*
(ii) minimal*, if it is syntax preserving (cf. Definition 5.3), and operational complete (cf. Definition 5.4-3(a)).*

Precise encodings offer more detailed criteria and used for positive encodability results (Section 6). In contrast, minimal encodings contains only some of the criteria of precise encodings: this reduced notion will be used for the negative result in Section 7.

Further we have:

Proposition 5.2 (Composability of Precise Encodings). *Let* $\langle [\![\cdot]\!]^1, (\!(\cdot)\!)^1, \{\!\{\cdot\}\!\}^1 \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *and* $\langle [\![\cdot]\!]^2, (\!(\cdot)\!)^2, \{\!\{\cdot\}\!\}^2 \rangle : \mathcal{L}_2 \to \mathcal{L}_3$ *be two precise typed encodings. Then their composition, denoted* $\langle [\![\cdot]\!]^2 \circ [\![\cdot]\!]^1, (\!(\cdot)\!)^2 \circ (\!(\cdot)\!)^1, \{\!\{\cdot\}\!\}^2 \circ \{\!\{\cdot\}\!\}^1 \rangle : \mathcal{L}_1 \to \mathcal{L}_3$ *is also a precise encoding.*

Proof. Straightforward application of the definition of each property, with the left-to-right direction of full abstraction being crucial. □

In Section 6 we consider the following concrete instances of typed calculi (cf. Definition 5.1):

Definition 5.6 (Concrete Typed Calculi). *We define the following typed calculi:*

$$\begin{aligned}
\mathcal{L}_{\mathsf{HO}\pi} &= \langle \mathsf{HO}\pi, \mathcal{T}_1, \stackrel{\ell}{\longmapsto}, \approx^H, \vdash \rangle \\
\mathcal{L}_{\mathsf{HO}} &= \langle \mathsf{HO}, \mathcal{T}_2, \stackrel{\ell}{\longmapsto}, \approx^H, \vdash \rangle \\
\mathcal{L}_{\pi} &= \langle \pi, \mathcal{T}_3, \stackrel{\ell}{\longmapsto}, \approx^C, \vdash \rangle
\end{aligned}$$

where: $\mathcal{T}_1$, $\mathcal{T}_2$, *and* $\mathcal{T}_3$ *are sets of types of* $\mathsf{HO}\pi$, HO, *and* π, *respectively; the typing* $\vdash$ *is defined in Figure 4; LTSs are as in Definition 4.10;* $\approx^H$ *is as in Definition 4.12;* $\approx^C$ *is as in Definition 4.13.*

6 Positive Expressiveness Results

In this section we present a study of the expressiveness of $\mathsf{HO}\pi$ and its subcalculi. We present two encodability results:

1. The higher-order name passing communications with recursions ($\mathsf{HO}\pi$) into the higher-order communication without name-passing nor recursions (HO) (Section 6.1).
2. $\mathsf{HO}\pi$ into the first-order name-passing communication with recursions (π) (Section 6.2).

In each case we show that the encoding is precise.

We often omit H and C from $\approx^H$ and $\approx^C$ for simplicity of the notations.

Remark 6.1 (Polyadic $\mathsf{HO}\pi$*).* We can assume a semantic preserving encoding from the polyadic $\mathsf{HO}\pi$ to the monadic $\mathsf{HO}\pi$. Polyadic $\mathsf{HO}\pi$ assumes a polyadic extension of the $\mathsf{HO}\pi$ semantics that defines values as $V \;::=\; \tilde{u} \;\mid\; \lambda\tilde{x}.P$ and input prefix as $n?(\tilde{x}).P$. See Section 8.2 for the full definition of polyadic $\mathsf{HO}\pi$.

6.1 Encoding $\mathsf{HO}\pi$ into HO

We show that the subcalculus HO is expressive enough to represent the the full $\mathsf{HO}\pi$ calculus.

The main challenge is to encode (1) name passing and (2) recursions. Name passing involves *packing* a name value as an abstraction send it and it and then substitute on the receiving using a name appication. The encoding on the recursion semantics are more complex; A process is encoded as an abstraction with no free names (i.e a shared abstraction). We then use higher-order passing to pass the process and duplicate the process. One copy of the process is used to reconstitute the original process and the other is used to enable another duplicator procedure. We handle the transformation of a process into a linear abstraction with the definition of an auxiliary mapping from processes with free names to processes without free names (but with free variables) (Definition 6.2). We first require an auxiliary definition:

Definition 6.1. *Let* $(\!|\cdot|\!) : 2^{\mathcal{N}} \longrightarrow \mathcal{V}^{\omega}$ *be a map of sequences lexicographically ordered names to sequences of variables, defined inductively as:*

$$(\!|\epsilon|\!) = \epsilon \qquad\qquad (\!|n \cdot \tilde{m}|\!) = x_n \cdot (\!|\tilde{m}|\!)$$

Given a process P, we write $\mathsf{ofn}(P)$ to denote the *sequence* of free names of P, lexicographically ordered.

The following auxiliary mapping transforms processes with free names into abstractions and it is used in Definition 6.3.

Definition 6.2. *Let* σ *be a set of session names. Define* $\llbracket \cdot \rrbracket_{\sigma} : \mathsf{HO}\pi \to \mathsf{HO}\pi$ *as in Figure 7.*

Given a process P with $\mathtt{fn}(P) = m_1, \cdots, m_n$, we are interested in its associated (polyadic) abstraction, which is defined as $\lambda x_1, \cdots, x_n. \llbracket P \rrbracket_{\emptyset}$, where $(\!|m_j|\!) = x_j$, for all $j \in \{1, \ldots, n\}$. This transformation from processes into abstractions can be reverted by using abstraction and application with an appropriate sequence of session names:

Proposition 6.1. *Let* P *be a* $\mathsf{HO}\pi$ *process with* $\tilde{n} = \mathsf{ofn}(P)$*. Also, suppose* $\tilde{x} = (\!|\tilde{n}|\!)$*. Then* $P \equiv x\,\tilde{n}\{\lambda\tilde{x}. \llbracket P \rrbracket_{\emptyset}/x\}$.

Proof. The proof is an easy induction on the map $\llbracket P \rrbracket_{\emptyset}$. We show a case since other cases are similar.
- Case: $\llbracket n!\langle m \rangle.P \rrbracket_{\emptyset} = x_n!\langle x_m \rangle.\llbracket P \rrbracket_{\emptyset}$
We rewrite substitution as: $x\,\tilde{n}\{\lambda\tilde{x}.\, x_n!\langle y_m \rangle.\llbracket P \rrbracket_{\emptyset}/x\} \equiv (x_n!\langle y_m \rangle.P)\{\tilde{x}/\tilde{n}\}$
If consider that $x_n, y_m \in (\!|\tilde{n}|\!)$ then from the definition of $(\!|\cdot|\!)$ we get that $n, m \in \tilde{n}$. Furthermore by the fact that $\tilde{n}$ and $(\!|\tilde{n}|\!)$ are ordered, substitution becomes: $n!\langle m \rangle.\llbracket P \rrbracket_{\emptyset}\{\tilde{x}/\tilde{n}\}$.
The rest of the cases are similar. $\square$

We are now ready to define the encoding of $\mathsf{HO}\pi$ into strict process-passing. Note that we assume polyadicity in abstraction and application. Given a session environment $\Delta = \{n_1 : S_1, \ldots, n_m : S_m\}$, in the following definition we write $\tilde{S}_{\Delta}$ to stand for $S_1, \ldots, S_m$.

Fig. 7 The auxiliary map (cf. Definition 6.2) used in the encoding of $\mathsf{HO}\pi$ into HO (Definition 6.3).

$$
\begin{array}{rcl}
\llbracket (\nu\, n)P \rrbracket_\sigma & ::= & (\nu\, n)\llbracket P \rrbracket_{\sigma\cdot n} \\
\llbracket n!\langle \lambda x.\, Q\rangle.P \rrbracket_\sigma & ::= & \begin{cases} x_n!\langle \lambda x.\, \llbracket Q \rrbracket_\sigma\rangle.\llbracket P \rrbracket_\sigma \;\; n \notin \sigma \\ n!\langle \lambda x.\, \llbracket Q \rrbracket_\sigma\rangle.\llbracket P \rrbracket_\sigma \;\; n \in \sigma \end{cases} \\
\llbracket n?(X).P \rrbracket_\sigma & ::= & \begin{cases} x_n?(X).\llbracket P \rrbracket_\sigma \;\; n \notin \sigma \\ n?(X).\llbracket P \rrbracket_\sigma \;\; n \in \sigma \end{cases} \\
\llbracket n \triangleleft l.P \rrbracket_\sigma & ::= & \begin{cases} x_n \triangleleft l.\llbracket P \rrbracket_\sigma \;\; n \notin \sigma \\ n \triangleleft l.\llbracket P \rrbracket_\sigma \;\; n \in \sigma \end{cases} \\
\llbracket n \triangleright \{l_i : P_i\}_{i\in I} \rrbracket_\sigma & ::= & \begin{cases} x_n \triangleright \{l_i : \llbracket P_i \rrbracket_\sigma\}_{i\in I} \;\; n \notin \sigma \\ n \triangleright \{l_i : \llbracket P_i \rrbracket_\sigma\}_{i\in I} \;\; n \in \sigma \end{cases} \\
\llbracket x\, n \rrbracket_\sigma & ::= & \begin{cases} x\, x_n \;\; n \notin \sigma \\ x\, n \;\; n \in \sigma \end{cases} \\
\llbracket (\lambda x.\, P)\, n \rrbracket_\sigma & ::= & \begin{cases} (\lambda x.\, \llbracket P \rrbracket_\sigma)\, x_n \;\; n \notin \sigma \\ (\lambda x.\, \llbracket P \rrbracket_\sigma)\, n \;\; n \in \sigma \end{cases} \\
\llbracket \mathbf{0} \rrbracket_\sigma & ::= & \mathbf{0} \\
\llbracket P \mid Q \rrbracket_\sigma & ::= & \llbracket P \rrbracket_\sigma \mid \llbracket Q \rrbracket_\sigma
\end{array}
$$

Definition 6.3 (Encoding $\mathsf{HO}\pi$ into HO). *Let f be a function from recursion variables to sequences of name variables. Define the typed encoding $\langle \llbracket\cdot\rrbracket^1_f, (\!(\cdot)\!)^1, \{\!\{\cdot\}\!\}^1 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\mathsf{HO}}$, where mappings $\llbracket\cdot\rrbracket^1$, $(\!(\cdot)\!)^1$, $\{\!\{\cdot\}\!\}^1$ are as in Figure 8. We assume that the mapping $(\!(\cdot)\!)^1$ on types is extended to session environments Δ and shared environments Γ as follows:*

$$
\begin{array}{rcl}
(\!(\Delta \cdot s : S)\!)^1 & = & (\!(\Delta)\!)^1 \cdot s : (\!(S)\!)^1 \\
(\!(\Gamma \cdot u : \langle S\rangle)\!)^1 & = & (\!(\Gamma)\!)^1 \cdot u : \langle (\!(S)\!)^1 \rangle \\
(\!(\Gamma \cdot u : \langle L\rangle)\!)^1 & = & (\!(\Gamma)\!)^1 \cdot u : \langle (\!(L)\!)^1 \rangle \\
(\!(\Gamma \cdot X : \Delta)\!)^1 & = & (\!(\Gamma)\!)^1 \cdot x : (\tilde{S}_\Delta, S^*) {\to} \diamond \qquad (\textit{where } S^* = \mu \mathsf{t}.?((\tilde{S}_\Delta, \mathsf{t}) {\to} \diamond); \mathsf{end})
\end{array}
$$

Note that Δ in $X : \Delta$ is mapped to a non-tail recursive session type. Non-tail recursive session types have been studied in [6,5]; to our knowledge, this is the first application in the context of higher-order session types. For a simplicity of the presentation, we use the polyadic name abstraction and passing. Polyadic semantics will be formally encoded into HO in Section 8.2.
We explain the mapping in Figure 6.3, focusing on *name passing* ($\llbracket u!\langle w\rangle.P \rrbracket^1_f$ and $\llbracket u?(x).P \rrbracket^1_f$), and *recursion* ($\llbracket \mu X.P \rrbracket^1_f$ and $\llbracket X \rrbracket^1_f$).

Name passing A name w is being passed as an input guarded abstraction; the abstraction receives a higher-order value and continues with the application of w over the received higher-order value. On the receiver side $u?(x).P$ the encoding realises a mechanism that i) receives the input guarded abstraction, then ii) applies it on a fresh session endpoint s, and iii) uses the dual endpoint $\overline{s}$ to send the continuation P as the abstraction $\lambda x.\, P$. Then name substitution is achieved via name application.

Fig. 8 Typed encoding of $\mathsf{HO}\pi$ into HO (cf. Defintion 6.3).

Terms

$$\llbracket u!\langle v\rangle.P\rrbracket^1_f \stackrel{\mathsf{def}}{=} u!\langle \lambda z.\, z?(x).(x\,v)\rangle.\llbracket P\rrbracket^1_f \qquad \llbracket u?(k).Q\rrbracket^1_f \stackrel{\mathsf{def}}{=} u?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle \lambda x.\,\llbracket Q\rrbracket^1_f\rangle.\mathbf{0})$$

$$\llbracket u!\langle \lambda x.\,Q\rangle.P\rrbracket^1_f \stackrel{\mathsf{def}}{=} u!\langle \lambda x.\,\llbracket Q\rrbracket^1_f\rangle.\llbracket P\rrbracket^1_f \qquad \llbracket u?(\underline{x}).P\rrbracket^1_f \stackrel{\mathsf{def}}{=} u?(\underline{x}).\llbracket P\rrbracket^1_f$$

$$\llbracket \mu X.P\rrbracket^1_f \stackrel{\mathsf{def}}{=} (\nu\ s)(s?(x).\llbracket P\rrbracket^1_{f,\{X\to\tilde{n}\}} \mid \overline{s}!\langle \lambda(\llparenthesis\tilde{n}\rrparenthesis, y).\ y?(z_X).\llbracket\llbracket P\rrbracket^1_{f,\{X\to\tilde{n}\}}\rrbracket_\emptyset\rangle.\mathbf{0}) \quad \tilde{n} = \mathsf{ofn}(P)$$

$$\llbracket X\rrbracket^1_f \stackrel{\mathsf{def}}{=} (\nu\ s)(z_X\,(\tilde{n}, s) \mid \overline{s}!\langle \lambda(\llparenthesis\tilde{n}\rrparenthesis, y).\ z_X\,(\llparenthesis\tilde{n}\rrparenthesis, y)\rangle.\mathbf{0}) \quad \tilde{n} = f(X)$$

$$\llbracket s \triangleleft l.P\rrbracket^1_f \stackrel{\mathsf{def}}{=} s \triangleleft l.\llbracket P\rrbracket^1_f \qquad \llbracket s \triangleright \{l_i : P_i\}_{i\in I}\rrbracket^1_f \stackrel{\mathsf{def}}{=} s \triangleright \{l_i : \llbracket P_i\rrbracket^1_f\}_{i\in I}$$

$$\llbracket x\,u\rrbracket^1_f \stackrel{\mathsf{def}}{=} x\,u \qquad \llbracket (\lambda x.\,P)\,u\rrbracket^1_f \stackrel{\mathsf{def}}{=} (\lambda x.\,\llbracket P\rrbracket^1_f)\,u$$

$$\llbracket P \mid Q\rrbracket^1_f \stackrel{\mathsf{def}}{=} \llbracket P\rrbracket^1_f \mid \llbracket Q\rrbracket^1_f \qquad \llbracket (\nu\ n)P\rrbracket^1_f \stackrel{\mathsf{def}}{=} (\nu\ n)\llbracket P\rrbracket^1_f$$

$$\llbracket \mathbf{0}\rrbracket^1_f \stackrel{\mathsf{def}}{=} \mathbf{0}$$

Types

$$\llparenthesis C\rrparenthesis^1_{\mathsf{v}} \stackrel{\mathsf{def}}{=} \begin{cases} (?(\llparenthesis C\rrparenthesis^1 {\multimap} \diamond);\mathsf{end}){\multimap}\diamond & \text{if } C = S \\ (?(\llparenthesis C\rrparenthesis^1 {\to} \diamond);\mathsf{end}){\multimap}\diamond & \text{otherwise} \end{cases}$$

$$\llparenthesis C{\multimap}\diamond\rrparenthesis^1_{\mathsf{v}} \stackrel{\mathsf{def}}{=} \llparenthesis C\rrparenthesis^1{\multimap}\diamond \qquad \llparenthesis C{\to}\diamond\rrparenthesis^1_{\mathsf{v}} \stackrel{\mathsf{def}}{=} \llparenthesis C\rrparenthesis^1{\to}\diamond$$

$$\llparenthesis \langle S\rangle\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \langle \llparenthesis S\rrparenthesis^1\rangle \qquad \llparenthesis \langle L\rangle\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \langle \llparenthesis L\rrparenthesis^1_{\mathsf{v}}\rangle$$

$$\llparenthesis !\langle U\rangle;S\rrparenthesis^1 \stackrel{\mathsf{def}}{=} !\langle \llparenthesis U\rrparenthesis^{\mathsf{v}}\rangle;\llparenthesis S\rrparenthesis^1 \qquad \llparenthesis ?(U);S\rrparenthesis^1 \stackrel{\mathsf{def}}{=} ?(\llparenthesis U\rrparenthesis^{\mathsf{v}});\llparenthesis S\rrparenthesis^1$$

$$\llparenthesis \oplus\{l_i : S_i\}_{i\in I}\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \oplus\{l_i : \llparenthesis S_i\rrparenthesis^1\}_{i\in I} \qquad \llparenthesis \&\{l_i : S_i\}_{i\in I}\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \&\{l_i : \llparenthesis S_i\rrparenthesis^1\}_{i\in I}$$

$$\llparenthesis \mathsf{t}\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \mathsf{t} \qquad \llparenthesis \mu\mathsf{t}.S\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \mu\mathsf{t}.\llparenthesis S\rrparenthesis^1$$

$$\llparenthesis \mathsf{end}\rrparenthesis^1 \stackrel{\mathsf{def}}{=} \mathsf{end}$$

Labels

$$\{\!\{(\nu\ \tilde{m}_1)n!\langle m\rangle\}\!\}^1 \stackrel{\mathsf{def}}{=} (\nu\ \tilde{m}_1)n!\langle \lambda z.\,z?(x).x\,m\rangle \qquad \{\!\{n?\langle m\rangle\}\!\}^1 \stackrel{\mathsf{def}}{=} n?\langle \lambda z.\,z?(x).x\,m\rangle$$

$$\{\!\{(\nu\ \tilde{m})n!\langle \lambda x.\,P\rangle\}\!\}^1 \stackrel{\mathsf{def}}{=} (\nu\ \tilde{m})n!\langle \lambda x.\,\llbracket P\rrbracket^1_\emptyset\rangle \qquad \{\!\{n?\langle \lambda x.\,P\rangle\}\!\}^1 \stackrel{\mathsf{def}}{=} n?\langle \lambda x.\,\llbracket P\rrbracket^1_\emptyset\rangle$$

$$\{\!\{n \oplus l\}\!\}^1 \stackrel{\mathsf{def}}{=} n \oplus l \qquad \{\!\{n\&l\}\!\}^1 \stackrel{\mathsf{def}}{=} n\&l$$

$$\{\!\{\tau\}\!\}^1 \stackrel{\mathsf{def}}{=} \tau$$

Recursion The encoding of a recursive process $\mu X.P$ is delicate, for it must preserve the linearity of session endpoints. To this end, we: i) record a mapping from recursive variable X to process variables z_X; ii) encode the recursion body P as a name abstraction in which free names of P are converted into name variables; iii) this higher-order value is embedded in an input-guarded "duplicator" process; and iv) make the encoding of process variable x to simulate recursion unfolding by invoking the duplicator in a by-need fashion, i.e., upon reception, abstraction $\llbracket P\rrbracket_\sigma$ is duplicated with one copy used to reconstitute the encoded recursion body P through the application of $\mathtt{fn}(P)$ and another copy used to re-invoke the duplicator when needed.

Proposition 6.2 (Type Preservation, $\mathsf{HO}\pi$ into HO). *Let P be a $\mathsf{HO}\pi$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then $\llparenthesis\Gamma\rrparenthesis^1;\emptyset;\llparenthesis\Delta\rrparenthesis^1 \vdash \llbracket P\rrbracket^1_f \triangleright \diamond$.*

Proof. By induction on the inference $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. Details in Proposition C.1 (Page 71). ◻

The following proposition formalizes our strategy for encoding recursive definitions as passing of polyadic abstractions:

Proposition 6.3 (Operational Correspondence for Recursive Processes). *Let P and P_1 be $\mathsf{HO}\pi$ processes s.t. $P = \mu X.P'$ and $P_1 = P'\{^{\mu X.P'}/X\} \equiv P$.*

If $\Gamma;\Delta \vdash P \xrightarrow{\ell} \Gamma;\Delta' \vdash P'$ then, there exist processes R_1, R_2, R_3, action ℓ', and mappings f, f_1, such that:

(i) $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash P \xrightarrow{\tau} (\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P']\!]^1\{^{R_3}/X\} = R_1$;

(ii) $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash R_1 \stackrel{\ell'}{\Longrightarrow} (\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash R_2$, *with* $\ell' = \{\!\{\ell\}\!\}^1$;

(iii) $R_3 = \lambda\tilde{m}.z?(x).\llbracket [\![P']\!]^1_{f_1} \rrbracket_\sigma$, *with* $\tilde{m} = \mathsf{ofn}(P'), z)$ *and* $f_1 = f, \{X \to \mathsf{ofn}(P')\}$.

Proof (Sketch). Part (1) follow directly from the definition of typed encoding for processes $[\![\cdot]\!]^1_f$ (Definition 6.3), observing that the reduction occurs along a restricted name, and so the session environment remains unchanged. Part (2) relies on Proposition 6.4. Part (3) is immediate from Definition 6.3. $\square$

The following proposition formalises completeness and soundness results for the encoding of $\mathsf{HO}\pi$ into HO. Recall that deterministic transitions τ_{s} and τ_β have been defined in Definition 4.14.

Proposition 6.4 (Operational Correspondence, $\mathsf{HO}\pi$ into HO). *Let P be a $\mathsf{HO}\pi$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then:*

1. *Suppose $\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'$. Then we have:*
   a) *If $\ell_1 \in \{(\nu\,\tilde{m})n!\langle m\rangle, (\nu\,\tilde{m})n!\langle \lambda x.\,Q\rangle, s \oplus l, s \& l\}$ then $\exists \ell_2$ s.t.*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash [\![P']\!]^1_f$ *and* $\ell_2 = \{\!\{\ell_1\}\!\}^1$.
   b) *If $\ell_1 = n?\langle \lambda y.\,Q\rangle$ and $P' = P_0\{^{\lambda y.\,Q}/x\}$ then $\exists \ell_2$ s.t.*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash [\![P_0]\!]^1_f\{^{\lambda y.\,[\![Q]\!]^1_\emptyset}/x\}$ *and* $\ell_2 = \{\!\{\ell_1\}\!\}^1$.
   c) *If $\ell_1 = n?\langle m\rangle$ and $P' = P_0\{^m/x\}$ then $\exists \ell_2, R$ s.t.*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash R$, *with* $\ell_2 = \{\!\{\ell_1\}\!\}^1$,
      and $(\!(\Gamma)\!)^1;(\!(\Delta')\!)^1 \vdash R \xrightarrow{\tau_\beta}\xrightarrow{\tau_{\mathsf{s}}}\xrightarrow{\tau_\beta} (\!(\Delta')\!)^1 \vdash [\![P_0]\!]^1_f\{^m/x\}$.
   d) *If $\ell_1 = \tau$ and $P' \equiv (\nu\,\tilde{m})(P_1 \mid P_2\{^m/x\})$ then $\exists R$ s.t.*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\tau} (\!(\Delta)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R)$, *and*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R) \xrightarrow{\tau_\beta}\xrightarrow{\tau_{\mathsf{s}}}\xrightarrow{\tau_\beta} (\!(\Delta)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^m/x\})$.
   e) *If $\ell_1 = \tau$ and $P' \equiv (\nu\,\tilde{m})(P_1 \mid P_2\{^{\lambda y.\,Q}/x\})$ then*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\tau} (\!(\Delta_1)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^{\lambda y.\,[\![Q]\!]^1_\emptyset}/x\})$.
   f) *If $\ell_1 = \tau$ and $P' \not\equiv (\nu\,\tilde{m})(P_1 \mid P_2\{^m/x\}) \wedge P' \not\equiv (\nu\,\tilde{m})(P_1 \mid P_2\{^{\lambda y.\,Q}/x\})$ then*
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\tau} (\!(\Delta'_1)\!)^1 \vdash [\![P']\!]^1_f$.
2. *Suppose $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash Q$. Then we have:*
   a) *If $\ell_2 \in \{(\nu\,\tilde{m})n!\langle \lambda z.\,z?(x).(x\,m)\rangle, (\nu\,\tilde{m})n!\langle \lambda x.\,R\rangle, s \oplus l, s \& l\}$ then $\exists \ell_1, P'$ s.t.*
      $\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'$, $\ell_1 = \{\!\{\ell_2\}\!\}^1$, *and* $Q = [\![P']\!]^1_f$.

b) If $\ell_2 = n?\langle \lambda y.R\rangle$ then either:

(i) $\exists \ell_1, x, P', P''$ *s.t.*
$\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'\{\lambda y.P''/x\}$, $\ell_1 = \{\!\{\ell_2\}\!\}^1$, $[\![P'']\!]^1_\emptyset = R$, *and* $Q = [\![P']\!]^1_f$.

(ii) $R \equiv y?(x).(x\,m)$ *and* $\exists \ell_1, z, P'$ *s.t.*
$\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'\{m/z\}$, $\ell_1 = \{\!\{\ell_2\}\!\}^1$, *and*
$(\!(\Gamma)\!)^1; (\!(\Delta')\!)^1 \vdash Q \xrightarrow{\tau_\beta}\xrightarrow{\tau_s}\xrightarrow{\tau_\beta} (\!(\Delta'')\!)^1 \vdash [\![P'\{m/z\}]\!]^1_f$

c) If $\ell_2 = \tau$ then $\Delta' = \Delta$ and either

(i) $\exists P'$ *s.t.* $\Gamma;\Delta \vdash P \xrightarrow{\tau} \Delta \vdash P'$, *and* $Q = [\![P']\!]^1_f$.

(ii) $\exists P_1, P_2, x, m, Q'$ *s.t.* $\Gamma;\Delta \vdash P \xrightarrow{\tau} \Delta \vdash (\nu\, \tilde{m})(P_1 \mid P_2\{m/x\})$, *and*
$(\!(\Gamma)\!)^1; (\!(\Delta)\!)^1 \vdash Q \xrightarrow{\tau_\beta}\xrightarrow{\tau_s}\xrightarrow{\tau_\beta} (\!(\Delta)\!)^1 \vdash [\![P_1]\!]^1_f \mid [\![P_2\{m/x\}]\!]^1_f$

Proof. The proof is a mechanical induction on the labelled Transition System. Parts (1) and (2) are proved separetely. The most demanding cases for the proof can be found in Proposition C.2 (page 74). □

Proposition 6.5 (Full Abstraction, HOπ into HO). *Let P_1, Q_1 be HOπ processes. $\Gamma;\Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1$ if and only if $(\!(\Gamma)\!)^1; (\!(\Delta_1)\!)^1 \vdash [\![P_1]\!]^1_f \approx^H (\!(\Delta_2)\!)^1 \vdash [\![Q_1]\!]^1_f$.*

Proof. The proof for the soundness direction considers closure that can be shown to be a bisimulation following the soundness direction of Operational Correspondence (Proposition 6.4). Whenever needed the proof makes use of the τ-inertness result (Proposition 4.3).

The proof for the completness direction also considers a closure shown to be a bisimulation up-to deterministic transition (Proposition 4.3) following the completeness direction of Operational Correspondence (Proposition 6.4).

Details of the proof can be found in Proposition C.3 (page 76). □

Proposition 6.6 (Precise encoding of HOπ into HO). *The encoding from $\mathcal{L}_{\mathsf{HO}\pi}$ to $\mathcal{L}_{\mathsf{HO}}$ is precise.*

Proof. Syntactic requirements are easily derivable from the definition of the mappings in Figure 8. Semantic requirements are a consequence of Proposition 6.2, Proposition 6.4, and Proposition 6.5. □

Example 6.1 (Encode $\mu X.a!\langle m\rangle.X$ into HO).
Mapping: Term mapping of HOπ process $\mu X.a!\langle m\rangle.X$ into a HO process. We note initially $f = \emptyset$. The first application of the mapping will give:

$$\begin{aligned}[\![\mu X.a!\langle m\rangle.X]\!]^1 &= (\nu\, s_1)(s_1?(x).[\![a!\langle m\rangle.x]\!]^1_{x \to x_a x_m} \mid \\ &\quad \overline{s_1}!\langle \lambda(x_a, x_m, z).\, z?(x).[\![\,[\![a!\langle m\rangle.x]\!]^1_{x \to x_a x_m}]\!]_\emptyset\rangle.\mathbf{0})\end{aligned}$$

with

$$\begin{aligned}[\![a!\langle m\rangle.x]\!]^1_{x \to x_a x_m} &= a!\langle \lambda z.\, z?(x).(x\,m)\rangle.[\![x]\!]^1_{x \to x_a x_m} \\ &= a!\langle \lambda z.\, z?(x).(x\,m)\rangle.(\nu\, s_2)(x\,(a, m, s_2) \mid \overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0})\end{aligned}$$

Furthermore:

$$
\begin{aligned}
&\llbracket \llbracket a!\langle m\rangle.x\rrbracket^1_{x\to x_a x_m}\rrbracket_\emptyset \\
&\qquad = \llbracket a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\ s_2)(x\,(a,m,s_2) \mid \overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0})\rrbracket_\emptyset \\
&\qquad = x_a!\langle \lambda z.\, z?(x).(x\, x_m)\rangle.\llbracket(\nu\ s_2)(x\,(a,m,s_2) \mid \overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0})\rrbracket_\emptyset \\
&\qquad = x_a!\langle \lambda z.\, z?(x).(x\, x_m)\rangle.(\nu\ s_2)(x\,(x_a,x_m,s_2) \mid \overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0})
\end{aligned}
$$

The whole encoding would be:

$$
\begin{aligned}
&V = \lambda(x_a,x_m,z).\, z?(x).x_a!\langle \lambda z.\, z?(x).(x\, x_m)\rangle.(\nu\ s_2)(x\,(x_a,x_m,s_2) \mid \overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0}) \\
&\llbracket \mu X.a!\langle m\rangle.X\rrbracket^1 \equiv \\
&(\nu\ s_1)(\overline{s_1}!\langle V\rangle.\mathbf{0} \mid s_1?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\ s_2)(\overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0}) \mid x\,(a,m,s_2))
\end{aligned}
$$

Transition Semantics: We can observe $\llbracket \mu X.a!\langle m\rangle.X\rrbracket^1$ as:

$$
\begin{aligned}
&\llbracket \mu X.a!\langle m\rangle.X\rrbracket^1 \\
&\equiv \\
&(\nu\ s_1)(\overline{s_1}!\langle V\rangle.\mathbf{0} \mid s_1?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\ s_2)(\overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0}) \mid x\,(a,m,s_2)) \\
&\xrightarrow{\tau} \\
&a!\langle \lambda z.\, z?(x).(x\, m)\rangle. \\
&(\nu\ s_2)(\overline{s_2}!\langle V\rangle.\mathbf{0} \mid s_2?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\ s_3)(\overline{s_3}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0}) \mid x\,(a,m,s_3)) \\
&\equiv_\alpha \\
&a!\langle \lambda z.\, z?(x).(x\, m)\rangle. \\
&(\nu\ s_1)(\overline{s_1}!\langle V\rangle.\mathbf{0} \mid s_1?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\ s_2)(\overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0}) \mid x\,(a,m,s_2)) \\
&\equiv \\
&a!\langle \lambda z.\, z?(x).(x\, m)\rangle.\llbracket \mu X.a!\langle m\rangle.X\rrbracket^1 \\
&\xrightarrow{a!\langle \lambda z.\, z?(x).(x\, m)\rangle} \\
&\llbracket \mu X.a!\langle m\rangle.X\rrbracket^1
\end{aligned}
$$

Typing Semantics: We further show that $\llbracket \mu X.a!\langle m\rangle.X\rrbracket^1$ is typable:

$$
\frac{\begin{array}{l}
\Gamma;\emptyset;\emptyset \vdash a \triangleright U_1 = \langle ?(U_2 \multimap \diamond);\mathsf{end}\multimap\diamond\rangle \\
\Gamma;\emptyset;\emptyset \vdash m \triangleright U_2 \\
\Gamma;\emptyset; s_2 :\vdash s_2 : ?(L);\mathsf{end} \vdash s_2 \triangleright ?(L);\mathsf{end} \\
\Gamma;\emptyset;\emptyset \vdash x \triangleright (U_1, U_2, ?(L);\mathsf{end}) \to \diamond
\end{array}}{\Gamma;\emptyset; s_2 : ?(L);\mathsf{end} \vdash x\,(a,m,s_2) \triangleright \diamond} \tag{8}
$$

$$
\frac{\dfrac{\begin{array}{l}
\Gamma\cdot x_a : U_1 \cdot x_m : U_2;\emptyset;\emptyset \vdash x_a \triangleright U_1 = \langle ?(U_2 \multimap \diamond);\mathsf{end}\multimap\diamond\rangle \\
\Gamma\cdot x_a : U_1 \cdot x_m : U_2;\emptyset;\emptyset \vdash x_m \triangleright U_2 \\
\Gamma;\emptyset; z : ?(L);\mathsf{end} \vdash z \triangleright ?(L);\mathsf{end} \\
\Gamma;\emptyset;\emptyset \vdash x \triangleright (U_1, U_2, ?(L);\mathsf{end}) \to \diamond
\end{array}}{\Gamma\cdot x_a : U_1 \cdot x_m : U_2;\emptyset; z : ?(L);\mathsf{end} \vdash x\,(x_a,x_m,z) \triangleright \diamond}}{\Gamma;\emptyset;\emptyset \vdash \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z) \triangleright (U_1, U_2, ?(L);\mathsf{end}) \to \diamond} \tag{9}
$$

$$
\frac{\begin{array}{l}
\text{Result (9)} \\
\Gamma;\emptyset;\overline{s_2} : !\langle (U_1, U_2, ?(L);\mathsf{end}) \to \diamond\rangle;\mathsf{end} \vdash \overline{s_2} \triangleright !\langle (U_1, U_2, ?(L);\mathsf{end}) \to \diamond\rangle;\mathsf{end}
\end{array}}{\Gamma;\emptyset;\overline{s_2} : !\langle (U_1, U_2, ?(L);\mathsf{end}) \to \diamond\rangle;\mathsf{end} \vdash \overline{s_2}!\langle \lambda(x_a,x_m,z).\, x\,(x_a,x_m,z)\rangle.\mathbf{0} \triangleright \diamond} \tag{10}
$$

$$\frac{\text{Result (8)} \quad \text{Result (10)} \quad \Delta = s_2 : ?(L);\mathsf{end} \cdot \overline{s_2} : !\langle (U_1, U_2, ?(L);\mathsf{end}) {\rightarrow} \diamond \rangle;\mathsf{end}}{\Gamma;\emptyset;\Delta \vdash \overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0} \mid x\,(a, m, s_2) \triangleright \diamond} \quad (11)$$

$$\frac{\begin{array}{l}\text{Result (11)} \quad ?(L);\mathsf{end} \text{ dual } !\langle (U_1, U_2, ?(L);\mathsf{end}) {\rightarrow} \diamond \rangle;\mathsf{end} \\ L = (U_1, U_2, ?(L);\mathsf{end}) {\rightarrow} \diamond \text{ implies} \\ ?(L);\mathsf{end} = \mu \mathsf{t}.?((U_1, U_2, \mathsf{t}) {\rightarrow} \diamond);\mathsf{end}\end{array}}{\Gamma;\emptyset;\emptyset \vdash (\nu\, s_2)(\overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0} \mid x\,(a, m, s_2) \triangleright \diamond)} \quad (12)$$

$$\frac{\begin{array}{l}\text{Result (12)} \\ \Gamma;\emptyset;\emptyset \vdash a \triangleright \langle ?(U_2 {\multimap} \diamond);\mathsf{end} {\multimap} \diamond \rangle \\ \Gamma;\emptyset;\emptyset \vdash \lambda z.\, z?(x).(x\, m) \triangleright ?(U_2 {\multimap} \diamond);\mathsf{end} {\multimap} \diamond\end{array}}{\Gamma;\emptyset;\emptyset \vdash a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\, s_2)(\overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0} \mid x\,(a, m, s_2)) \triangleright \diamond} \quad (13)$$

$$\frac{\begin{array}{l}\text{Result (13)} \quad \Gamma' = \Gamma \backslash x \\ \Gamma;\emptyset;\emptyset \vdash x \triangleright (U_1, U_2, \mu \mathsf{t}.?((U_1, U_2, t) {\rightarrow} \diamond);\mathsf{end}) {\rightarrow} \diamond \\ \Gamma';\emptyset;\Delta \vdash s_1 \triangleright ?((U_1, U_2, \mu \mathsf{t}.?((U_1, U_2, \mathsf{t}) {\rightarrow} \diamond);\mathsf{end}) {\rightarrow} \diamond);\mathsf{end}\end{array}}{\begin{array}{l}\Gamma';\emptyset;\Delta_1 \vdash \\ \quad s_1?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle.(\nu\, s_2)(\overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0} \mid x\,(a, m, s_2)) \triangleright \diamond\end{array}} \quad (14)$$

$$\frac{\begin{array}{l}V = \lambda(x_a, x_m, z).\, z?(x).x_a!\langle \lambda z.\, z?(x).(x\, x_m)\rangle. \\ \quad (\nu\, s_2)(x\,(x_a, x_m, s_2) \mid \overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0}) \\ \Gamma';\emptyset;\emptyset \vdash V \triangleright (U_1, U_2, \mu \mathsf{t}.?((U_1, U_2, \mathsf{t}) {\rightarrow} \diamond);\mathsf{end}) {\rightarrow} \diamond \\ \Gamma';\emptyset;\Delta_2 \vdash \overline{s_1} \triangleright !\langle (U_1, U_2, \mu \mathsf{t}.?((U_1, U_2, t) {\rightarrow} \diamond);\mathsf{end}) {\rightarrow} \diamond \rangle;\mathsf{end}\end{array}}{\Gamma';\emptyset;\Delta_2 \vdash \overline{s_1}!\langle V\rangle.\mathbf{0} \triangleright \diamond} \quad (15)$$

$$\dfrac{\dfrac{\text{Result (14)} \quad \text{Result (15)}}{\begin{array}{l}\Gamma;\emptyset;\Delta_1 \cdot \Delta_2 \vdash \overline{s_1}!\langle V\rangle.\mathbf{0} \mid s_1?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle. \\ \qquad (\nu\, s_2)(\overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0}) \mid x\,(a, m, s_2) \triangleright \diamond\end{array}}}{\begin{array}{l}\Gamma;\emptyset;\emptyset \vdash (\nu\, s_1)(\overline{s_1}!\langle V\rangle.\mathbf{0} \mid s_1?(x).a!\langle \lambda z.\, z?(x).(x\, m)\rangle. \\ \qquad (\nu\, s_2)(\overline{s_2}!\langle \lambda(x_a, x_m, z).\, x\,(x_a, x_m, z)\rangle.\mathbf{0}) \mid x\,(a, m, s_2)) \triangleright \diamond\end{array}}$$

□

6.2 From $\mathsf{HO}\pi$ to π

We now discuss the encodability of HO into π where we essentially follow the representability result put forward by Sangiorgi [45,50], but casted in the setting of session-typed communications. Intuitively, the strategy represents the exchange of a process with the exchange of a freshly generated *trigger name*. Trigger names are used to activate copies of the process, which now becomes a persistent resource represented by an input-guarded replication. In our calculi, a session name is a linear resource and cannot be replicated. Consider the following (naive) adaptation of Sangiorgi's strategy in

Fig. 9 Typed encoding of HOπ to π (Definition 6.4). Mappings $[\![\cdot]\!]^3$, $(\!(\cdot)\!)^3$, and $\{\!\{\cdot\}\!\}^3$ are homomorphisms for the other processes/types/labels.

$$
\begin{aligned}
[\![u!\langle\lambda x.\,Q\rangle.P]\!]^2 &\stackrel{\mathtt{def}}{=} \begin{cases}(\nu\, a)(u!\langle a\rangle.([\![P]\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2)) & s \notin \mathtt{fn}(Q)\\ (\nu\, s)(u!\langle \overline{s}\rangle.([\![P]\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2)) & \text{otherwise}\end{cases}\\
[\![u?(x).P]\!]^2 &\stackrel{\mathtt{def}}{=} u?(x).[\![P]\!]^2\\
[\![x\,u]\!]^2 &\stackrel{\mathtt{def}}{=} (\nu\, s)(x!\langle s\rangle.\overline{s}!\langle u\rangle.\mathbf{0})\\
[\![(\lambda x.\,P)\,u]\!]^2 &\stackrel{\mathtt{def}}{=} (\nu\, s)(s?(x).[\![P]\!]^2 \mid \overline{s}!\langle u\rangle.\mathbf{0})\\
\\
(\!(!\langle S\to\diamond\rangle;S_1)\!)^2 &\stackrel{\mathtt{def}}{=} !\langle\langle ?((\!(S)\!)^2);\mathsf{end}\rangle\rangle;(\!(S_1)\!)^2\\
(\!(?(S\to\diamond);S_1)\!)^2 &\stackrel{\mathtt{def}}{=} ?(\langle ?((\!(S)\!)^2);\mathsf{end}\rangle);(\!(S_1)\!)^2\\
(\!(!\langle S\multimap\diamond\rangle;S_1)\!)^2 &\stackrel{\mathtt{def}}{=} !\langle ?((\!(S)\!)^2);\mathsf{end}\rangle;(\!(S_1)\!)^2\\
(\!(?(S\multimap\diamond);S_1)\!)^2 &\stackrel{\mathtt{def}}{=} ?(?((\!(S)\!)^2);\mathsf{end});(\!(S_1)\!)^2\\
\{\!\{(\nu\,\tilde{m}')n!\langle\lambda x.\,P\rangle\}\!\}^2 &\stackrel{\mathtt{def}}{=} (\nu\, m)n!\langle m\rangle\\
\{\!\{n?\langle\lambda x.\,P\rangle\}\!\}^2 &\stackrel{\mathtt{def}}{=} n?\langle m\rangle \quad m \text{ fresh}
\end{aligned}
$$

which session names are used are triggers and exchanged processes would be have to used exactly once:

$$
\begin{aligned}
[\![u!\langle\lambda x.\,Q\rangle.P]\!]^n &\stackrel{\mathtt{def}}{=} (\nu\, s)(u!\langle s\rangle.([\![P]\!]^n \mid \overline{s}?(x).[\![Q]\!]^n))\\
[\![u?(x).P]\!]^n &\stackrel{\mathtt{def}}{=} u?(x).[\![P]\!]^n\\
[\![x\,u]\!]^n &\stackrel{\mathtt{def}}{=} x!\langle u\rangle.\mathbf{0}
\end{aligned}
$$

with the remaining HOπ constructs being mapped homomorphically. Although $[\![\cdot]\!]^n$ captures the correct semantics when dealing with systems that allow only linear abstractions, it suffers from non-typability in the presence of shared abstractions. For instance, mapping for $P = n!\langle\lambda x.\,x!\langle m\rangle.\mathbf{0}\rangle.\mathbf{0} \mid \overline{n}?(x).(x\,s_1 \mid x\,s_2)$ would be:

$$
[\![P]\!]^n \stackrel{\mathtt{def}}{=} (\nu\, s)(n!\langle s\rangle.\overline{s}?(x).x!\langle m\rangle.\mathbf{0} \mid \overline{n}?(x).(x!\langle s_1\rangle.\mathbf{0} \mid x!\langle s_2\rangle.\mathbf{0}))
$$

The above process is non typable since processes ($x!\langle s_1\rangle.\mathbf{0}$ and $x!\langle s_2\rangle.\mathbf{0}$) cannot be put in parallel because they do not have disjoint session environments.

The correct approach would be to use replicated shared names as triggers instead of session names, when dealing with shared abstractions. Below we write $*\,P$ as a shorthand notation for $\mu X.(P \mid X)$.

Definition 6.4 (Encoding HOπ to π). *Define encoding* $\langle [\![\cdot]\!]^2, (\!(\cdot)\!)^2, \{\!\{\cdot\}\!\}^2\rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_\pi$ *with mappings* $[\![\cdot]\!]^2$, $(\!(\cdot)\!)^2$, $\{\!\{\cdot\}\!\}^2$ *as in Figure 9.*

Proposition 6.7 (Type Preservation, HOπ into π). *Let P be a* HOπ *process. If* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *then* $(\!(\Gamma)\!)^2;\emptyset;(\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \triangleright \diamond$.

Proof. By induction on the inference $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. Details in Proposition C.4 (Page 79). ◻

Remark 6.2. As stated in [48, Lem. 5.2.2], due to the replicated trigger, operational correspondence in Definition 5.4 is refined to prove full abstraction: e.g., completeness of the case $\ell_1 \neq \tau$, is changed as follows. Suppose:

$$\Gamma; \Delta \vdash P \stackrel{\ell_1}{\longmapsto} \Delta' \vdash P'$$

If $\ell_1 = (\nu\, \tilde{m})n!\langle \lambda x.R\rangle$, then

$$(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} (\!(\Delta')\!)^2 \vdash Q$$

where $\ell_2 = (\nu a)n!\langle a\rangle$ and $Q = [\![P' \mid *\, a?(y).y?(x).R]\!]^2$.
Similarly, if $\ell_1 = n?\langle \lambda x.R\rangle$, then

$$(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} (\!(\Delta')\!)^2 \vdash Q$$

where $\ell_2 = n!\langle a\rangle$ and $[\![P']\!]^2 \approx^H (\nu\, a)(Q \mid *\, a?(y).y?(x).[\![R]\!]^2)$. Soundness is stated in a symmetric way.

This last remark is stated formally in the next proposition:

Proposition 6.8 (Operational Correspondence, $\mathsf{HO}\pi$ into π). *Let P be an $\mathsf{HO}\pi$ process such that $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$.*

1. *Suppose $\Gamma; \Delta \vdash P \stackrel{\ell_1}{\longmapsto} \Delta' \vdash P'$. Then we have:*
   a) *If $\ell_1 = (\nu\, \tilde{m})n!\langle \lambda x.Q\rangle$, then $\exists \Gamma', \Delta'', R$ where either:*
      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} \Gamma' \cdot (\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$
      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} (\!(\Gamma)\!)^2; \Delta'' \vdash [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$
   b) *If $\ell_1 = n?\langle \lambda y.Q\rangle$ then $\exists R$ where either*
      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} \Gamma'; (\!(\Delta'')\!)^2 \vdash R$*, for some* Γ' *and*
        $(\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \approx^H (\!(\Delta'')\!)^2 \vdash (\nu\, a)(R \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} (\!(\Gamma)\!)^2; (\!(\Delta'')\!)^2 \vdash R$*, and*
        $(\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \approx^H (\!(\Delta'')\!)^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$
   c) *If $\ell_1 = \tau$ then either:*
      - $\exists R$ *such that*

        $$\begin{aligned}&(\!(\Gamma)\!)^2; \emptyset;\ (\!(\Delta)\!)^2\ \vdash [\![P]\!]^2\\ &\quad \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^2 \vdash (\nu\, \tilde{m})([\![P_1]\!]^2 \mid (\nu\, a)([\![P_2]\!]^2\{a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2))\end{aligned}$$

      - $\exists R$ *such that*

        $$\begin{aligned}&(\!(\Gamma)\!)^2; \emptyset;\ (\!(\Delta)\!)^2\ \vdash [\![P]\!]^2\\ &\quad \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^2 \vdash (\nu\, \tilde{m})([\![P_1]\!]^2 \mid (\nu\, s)([\![P_2]\!]^2\{\bar{s}/x\} \mid s?(y).y?(x).[\![Q]\!]^2))\end{aligned}$$

      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\tau}{\longmapsto} (\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2$
      - $\ell_1 = \tau_\beta$ *and* $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\tau_{\mathsf{s}}}{\longmapsto} (\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2$

d) *If* $\ell_1 \in \{n \oplus l, n\&l\}$ *then*
$\exists \ell_2 = \{\!\{\ell_1\}\!\}^2$ *such that* $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} (\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2$.

2. *Suppose* $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} (\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash R$.
   a) *If* $\ell_2 = (\nu\, m)n!\langle m\rangle$ *then either*
      - $\exists P'$ *such that* $P \stackrel{(\nu\, m)n!\langle m\rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2$.
      - $\exists Q, P'$ *such that* $P \stackrel{n!\langle \lambda x.\, Q\rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$
      - $\exists Q, P'$ *such that* $P \stackrel{n!\langle \lambda x.\, Q\rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$
   b) *If* $\ell_2 = n?\langle m\rangle$ *then either*
      - $\exists P'$ *such that* $P \stackrel{n?\langle m\rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2$.
      - $\exists Q, P'$ *such that* $P \stackrel{n?\langle \lambda x.\, Q\rangle}{\longmapsto} P'$
        and $(\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \approx^H (\!(\Delta')\!)^2 \vdash (\nu\, a)(R \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
      - $\exists Q, P'$ *such that* $P \stackrel{n?\langle \lambda x.\, Q\rangle}{\longmapsto} P'$
        and $(\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \approx^H (\!(\Delta')\!)^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$
   c) *If* $\ell_2 = \tau$ *then* $\exists P'$ *such that* $P \stackrel{\tau}{\longmapsto} P'$ *and* $(\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \approx^H (\!(\Delta')\!)^2 \vdash R$.
   d) *If* $\ell_2 \notin \{n!\langle m\rangle, n \oplus l, n\&l\}$ *then* $\exists \ell_1$ *such that* $\ell_1 = \{\!\{\ell_2\}\!\}^2$ *and*
      $\Gamma; \Delta \vdash P \stackrel{\ell_1}{\longmapsto} \Gamma; \Delta \vdash P'$.

Proof. The proof is done by induction on the labelled transition system considering Definition 6.4. The most demaning cases are Part 1b and Part 2b where we require a further induction to proof bisimulation closure.

Details of the proof of the most demanding cases can be found in Proposition C.5 (page 34). □

Proposition 6.9 (Full Abstraction, From HOπ to π). *Let* P_1, Q_1 *be* HOπ *processes.* $\Gamma;\Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1$ *if and only if* $(\!(\Gamma)\!)^2; (\!(\Delta_1)\!)^2 \vdash [\![P_1]\!]^2 \approx^C (\!(\Delta_2)\!)^2 \vdash [\![Q_1]\!]^2$.

Proof. Proof follows directly from Proposition 6.8. The cases of Proposition 6.8 are used to create a bisimulation closure to prove the the soundness direction and a bisimulation up to determinate transition (Lemma 4.3) to prove the completeness direction. □

Proposition 6.10 (Precise encoding of HOπ into π). *The encoding from* $\mathcal{L}_{\mathsf{HO}\pi}$ *to* $\mathcal{L}_\pi$ *is precise.*

Proof. Syntactic requirements are easily derivable from the definition of the mappings in Figure 9. Semantic requirements are a consequence of Proposition 6.7, Proposition 6.8, and Proposition 6.9. □

7 Negative Encodability Results

As most session calculi, HOπ includes communication on both shared and linear channels. The former enables non determinism and unrestricted behavior; the latter allows to represent deterministic and linear communication structures. The expressive power of shared names is also illustrated by our encoding from $\mathsf{HO}\pi$ into π (Definition 6.4). Shared and linear channels are fundamentally different; still, to the best of our knowledge, the status of shared communication, in terms of expressiveness, has not been formalized for session calculi.

The above begs the question: can we represent shared name interaction using session name interaction? In this section we prove that shared names actually add expressiveness to $\mathsf{HO}\pi$, for their behavior cannot be represented using purely deterministic processes. To this end, we show the non existence of a minimal encoding (cf. Definition 5.5(ii)) of shared name communication into linear communication. Recall that minimal encodings preserve barbs (Proposition 5.1).

Theorem 7.1. *Let* $\mathsf{C}_1, \mathsf{C}_2 \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$. *There is no typed, minimal encoding from* $\mathcal{L}_{\mathsf{C}_1}$ *into* $\mathcal{L}_{\mathsf{C}_2^{-\mathsf{sh}}}$

Proof. Assume, towards a contradiction, that such a typed encoding indeed exists. Consider the π process

$$P = \overline{a}\langle s\rangle.\mathbf{0} \mid a(x).n \triangleleft l_1.\mathbf{0} \mid a(x).m \triangleleft l_2.\mathbf{0} \qquad \text{(with } n \neq m\text{)}$$

such that $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. From process P we have:

$$\Gamma;\Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash n \triangleleft l_1.\mathbf{0} \mid a(x).m \triangleleft l_2.\mathbf{0} = P_1 \tag{16}$$
$$\Gamma;\Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash m \triangleleft l_2.\mathbf{0} \mid a(x).n \triangleleft l_1.\mathbf{0} = P_2 \tag{17}$$

Thus, by definition of typed barb we have:

$$\Gamma;\Delta' \vdash P_1 \downarrow_n \wedge \Gamma;\Delta' \vdash P_1 \not\downarrow_m \tag{18}$$
$$\Gamma;\Delta' \vdash P_2 \downarrow_m \wedge \Gamma;\Delta' \vdash P_2 \not\downarrow_n \tag{19}$$

Consider now the $\mathsf{HO}\pi^{-\mathsf{sh}}$ process $[\![P]\!]$. By our assumption of operational completeness (Definition 5.4-2(a)), from (16) with (17) we infer that there exist $\mathsf{HO}\pi^{-\mathsf{sh}}$ processes S_1 and S_2 such that:

$$(\!(\Gamma)\!);(\!(\Delta)\!) \vdash [\![P]\!] \stackrel{\tau_{\mathsf{s}}}{\Longmapsto} (\!(\Delta')\!) \vdash S_1 \approx [\![P_1]\!] \tag{20}$$
$$(\!(\Gamma)\!);(\!(\Delta)\!) \vdash [\![P]\!] \stackrel{\tau_{\mathsf{s}}}{\Longmapsto} (\!(\Delta')\!) \vdash S_2 \approx [\![P_2]\!] \tag{21}$$

By our assumption of barb preservation, from (18) with (19) we infer:

$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_1]\!] \Downarrow_n \wedge (\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_1]\!] \not\Downarrow_m \tag{22}$$
$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_2]\!] \Downarrow_m \wedge (\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_2]\!] \not\Downarrow_n \tag{23}$$

By definition of $\approx$, by combining (20) with (22) and (21) with (23), we infer barbs for S_1 and S_2:

$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash S_1 \Downarrow_n \wedge (\!(\Gamma)\!);(\!(\Delta')\!) \vdash S_1 \not\Downarrow_m \tag{24}$$
$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash S_2 \Downarrow_m \wedge (\!(\Gamma)\!);(\!(\Delta')\!) \vdash S_2 \not\Downarrow_n \tag{25}$$

That is, S_1 and $[\![P_1]\!]$ (resp. S_2 and $[\![P_2]\!]$) have the same barbs. Now, by τ-inertness (Proposition 4.3), we have both

$$(\!(\Gamma)\!);(\!(\Delta)\!) \vdash S_1 \approx (\!(\Delta')\!) \vdash [\![P]\!] \tag{26}$$

$$(\!(\Gamma)\!);(\!(\Delta)\!) \vdash S_2 \approx (\!(\Delta')\!) \vdash [\![P]\!] \tag{27}$$

Combining (26) with (27), by transitivity of $\approx$, we have

$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash S_1 \approx (\!(\Delta')\!) \vdash S_2 \tag{28}$$

In turn, from (28) we infer that it must be the case that:

$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_1]\!] \Downarrow_n \wedge (\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_1]\!] \Downarrow_m$$
$$(\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_2]\!] \Downarrow_m \wedge (\!(\Gamma)\!);(\!(\Delta')\!) \vdash [\![P_2]\!] \Downarrow_n$$

which clearly contradict (22) and (23) above. □

8 Extensions of $\mathsf{HO}\pi$

This section studies (i) the extension of $\mathsf{HO}\pi$ with higher-order applications/abstractions (denoted $\mathsf{HO}\pi^+$), and (ii) the extension of $\mathsf{HO}\pi$ with polyadicity (denoted $\mathsf{HO}\vec{\pi}$). In both cases, we detail required modifications in the syntax and types, and describe further encodability results.

8.1 Encoding $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$

The $\mathsf{HO}\pi$ calculus is purposefully minimal and allows only name applications/abstractions (also referred to as *first-order* applications/abstractions). We now introduce $\mathsf{HO}\pi^+$, the extension of $\mathsf{HO}\pi$ with higher-order applications. We show that $\mathsf{HO}\pi^+$ has a precise encoding into $\mathsf{HO}\pi$ (Proposition 8.4). Therefore, since typed encodings are composable (Proposition 5.2), $\mathsf{HO}\pi^+$ has a precise encoding to HO and π. In turn, this latter result implies that HO is powerful enough to express full higher-order semantics.

Modifications in Syntax, Reduction Semantics, and Types. The syntax of $\mathsf{HO}\pi^+$ processes is obtained from the syntax for processes given in Figure 2 by replacing $V\,u$ with $W\,V$. Reduction is then defined by the rules in Figure 3, excepting rule [App], which is replaced by the following rule

$$[\text{App}^+] \qquad (\lambda x.\,P)\,V \longrightarrow P\{^V/_x\}$$

The syntax of types in Figure 3.1 is generalized by including

$$L \ ::= \ U{\rightarrow}\diamond \ \mid \ U{\multimap}\diamond$$

instead of $L \ ::= \ C{\rightarrow}\diamond \ \mid \ C{\multimap}\diamond$. Definitions of type equivalence/duality and typing environments (Γ and Λ) are straightforward extensions of Definition 3.2, Definition 3.3, and

Definition 3.4, respectively. The typing rules of Figure 4 are then modified accordingly: most significant changes are required in rules [Abs] and [App] (for typing abstractions and applications, respectively), which for $\mathsf{HO}\pi^+$ processes are modified as follows:

$$[\mathrm{Abs}^+]\ \frac{\Gamma;\Lambda;\Delta_1 \vdash P \triangleright \diamond \quad \Gamma;\emptyset;\Delta_2 \vdash x \triangleright U}{\Gamma;\Lambda;\Delta_1\backslash\Delta_2 \vdash \lambda x.\, P \triangleright U{\multimap}\diamond}$$

$$[\mathrm{App}^+]\ \frac{U = U'{\multimap}\diamond \vee U'{\rightarrow}\diamond \quad \Gamma;\Lambda;\Delta_1 \vdash V \triangleright U \quad \Gamma;\emptyset;\Delta_2 \vdash W \triangleright U'}{\Gamma;\Lambda;\Delta_1 \cdot \Delta_2 \vdash V\, W \triangleright \diamond}$$

With these modifications we can now state the extension of Theorem 3.1:

Theorem 8.1 (Type Soundness for $\mathsf{HO}\pi^+$).

1. *(Subject Congruence)* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *and* $P \equiv P'$ *implies* $\Gamma;\emptyset;\Delta \vdash P' \triangleright \diamond$.
2. *(Subject Reduction)* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *with balanced* Δ *and* $P \longrightarrow P'$ *implies* $\Gamma;\emptyset;\Delta' \vdash P' \triangleright \diamond$ *and either (i)* $\Delta = \Delta'$ *or (ii)* $\Delta \longrightarrow \Delta'$ *with* Δ' *balanced.*

Proof. Part (1) is as for $\mathsf{HO}\pi$ processes. Part (2) is also as before, but requires the expected generalization of parts (3) and (4) of the substitution lemma (Lemma 3.1). We describe the analysis when the reduction is inferred by rule [App$^+$]. We have

$$P = (\lambda x.\, Q)\, V \longrightarrow Q\{V/x\} = P'$$

Suppose $\Gamma;\, \emptyset;\, \Delta \vdash (\lambda x.\, Q)\, V \triangleright \diamond$. We examine one possible way in which this assumption can be derived; other cases are similar or simpler:

$$\dfrac{\dfrac{\Gamma, x : L_1{\multimap}\diamond;\, \emptyset;\, \Delta \vdash Q \triangleright \diamond \quad \Gamma, x : L_1{\multimap}\diamond;\, \emptyset;\, \emptyset \vdash x \triangleright L_1{\multimap}\diamond}{\Gamma;\, \emptyset;\, \Delta \vdash \lambda x.\, Q \triangleright (L_1{\rightarrow}\diamond){\rightarrow}\diamond} \qquad \dfrac{}{\Gamma;\, \emptyset;\, \emptyset \vdash V \triangleright L_1{\rightarrow}\diamond}}{\Gamma;\, \emptyset;\, \Delta \vdash (\lambda x.\, Q)\, V \triangleright \diamond}$$

Then, by combining premise $\Gamma, x : L_1{\multimap}\diamond;\, \emptyset;\, \Delta \vdash Q \triangleright \diamond$ with the extended formulation of Lemma 3.1(4), we obtain $\Gamma;\, \emptyset;\, \Delta \vdash Q\{V/x\} \triangleright \diamond$, as desired. ◻

As for the behavioural semantics of $\mathsf{HO}\pi^+$, modifications are as expected. The set of action labels remains the same. In the untyped LTS, rule $\langle\mathrm{App}\rangle$ is replaced with rule $\lambda x.\, P\, V \xrightarrow{\tau} P\{V/x\}$. Definition 4.8 (characteristic processes) now includes

$$[\![U{\rightarrow}\diamond]\!]^x \stackrel{\mathsf{def}}{=} [\![U{\multimap}\diamond]\!]^x \stackrel{\mathsf{def}}{=} x[\![U]\!]_{\mathsf{c}}$$
$$[\![U{\rightarrow}\diamond]\!]_{\mathsf{c}} \stackrel{\mathsf{def}}{=} [\![U{\multimap}\diamond]\!]_{\mathsf{c}} \stackrel{\mathsf{def}}{=} \lambda x.\, [\![U]\!]^x$$

instead of $[\![C{\rightarrow}\diamond]\!]^x \stackrel{\mathsf{def}}{=} [\![C{\multimap}\diamond]\!]^x \stackrel{\mathsf{def}}{=} x[\![C]\!]_{\mathsf{c}}$ and $[\![C{\rightarrow}\diamond]\!]_{\mathsf{c}} \stackrel{\mathsf{def}}{=} [\![C{\multimap}\diamond]\!]_{\mathsf{c}} \stackrel{\mathsf{def}}{=} \lambda x.\, [\![C]\!]^x$, respectively. The rest of the definitions for the behavioural semantics is kept unchanged.

Encoding $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$. We now present an encoding from $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$.

Definition 8.1 (Encoding from $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let* $\mathcal{L}_{\mathsf{HO}\pi^+} = \langle \mathsf{HO}\pi^+, \mathcal{T}_4, \stackrel{\ell}{\longmapsto}, \approx^H, \vdash\rangle$ *where* $\mathcal{T}_4$ *is a set of types of* $\mathsf{HO}\pi^+$*; the typing* $\vdash$ *is defined in Figure 4 with extended rules [Abs] and [App]. Then, mapping* $\langle [\![\cdot]\!]^3, (\!(\cdot)\!)^3, \{\!\!\{\cdot\}\!\!\}^3 \rangle : \mathcal{L}_{\mathsf{HO}\pi^+} \to \mathcal{L}_{\mathsf{HO}\pi}$ *is defined in Figure 10.*

Fig. 10 Encoding of $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$ (cf. Definition 8.1). We assume that the rest of the encoding is homomorphic on the syntax of processes, types and labels, respectively.

$$
\begin{aligned}
[\![x\,(\lambda y.\,P)]\!]^3 &\stackrel{\text{def}}{=} (\nu\ s)(x\,s \mid \overline{s}!\langle \lambda y.\,[\![P]\!]^3\rangle.\mathbf{0})\\
[\![(\lambda x.\,P)\,(\lambda y.\,Q)]\!]^3 &\stackrel{\text{def}}{=} (\nu\ s)(s?(x).[\![P]\!]^3 \mid \overline{s}!\langle \lambda y.\,[\![Q]\!]^3\rangle.\mathbf{0})\\
[\![u!\langle \lambda \underline{x}.\,Q\rangle.P]\!]^3 &\stackrel{\text{def}}{=} u!\langle \lambda z.\,z?(\underline{x}).[\![Q]\!]^3\rangle.[\![P]\!]^3\\
[\![u!\langle \lambda k.\,Q\rangle.P]\!]^3 &\stackrel{\text{def}}{=} u!\langle \lambda k.\,[\![Q]\!]^3\rangle.[\![P]\!]^3
\end{aligned}
$$

$$
\begin{aligned}
(\!(L{\rightarrow}\diamond)\!)^3 &\stackrel{\text{def}}{=} (?((\!(L)\!)^3);\mathsf{end}){\rightarrow}\diamond\\
(\!(L{\multimap}\diamond)\!)^3 &\stackrel{\text{def}}{=} (?((\!(L)\!)^3);\mathsf{end}){\multimap}\diamond\\
(\!(!\langle L{\rightarrow}\diamond\rangle;S)\!)^3 &\stackrel{\text{def}}{=} !\langle(\!(L{\rightarrow}\diamond)\!)^3\rangle;(\!(S)\!)^3\\
(\!(!\langle L{\multimap}\diamond\rangle;S)\!)^3 &\stackrel{\text{def}}{=} !\langle(\!(L{\multimap}\diamond)\!)^3\rangle;(\!(S)\!)^3\\
(\!(?(L{\rightarrow}\diamond);S)\!)^3 &\stackrel{\text{def}}{=} ?((\!(L{\rightarrow}\diamond)\!)^3);(\!(S)\!)^3\\
(\!(?(L{\multimap}\diamond);S)\!)^3 &\stackrel{\text{def}}{=} ?((\!(L{\multimap}\diamond)\!)^3);(\!(S)\!)^3
\end{aligned}
$$

$$
\begin{aligned}
\{\!\{(\nu\ \tilde{m})n!\langle \lambda k.\,P\rangle\}\!\}^3 &\stackrel{\text{def}}{=} (\nu\ \tilde{m})n!\langle \lambda x.\,[\![P]\!]^3\rangle\\
\{\!\{n?\langle \lambda k.\,P\rangle\}\!\}^3 &\stackrel{\text{def}}{=} n?\langle \lambda x.\,[\![P]\!]^3\rangle\\
\{\!\{(\nu\ \tilde{m})n!\langle \lambda \underline{x}.\,P\rangle\}\!\}^3 &\stackrel{\text{def}}{=} (\nu\ \tilde{m})n!\langle \lambda z.\,z?(x).[\![P]\!]^3\rangle\\
\{\!\{n?\langle \lambda \underline{x}.\,P\rangle\}\!\}^3 &\stackrel{\text{def}}{=} n?\langle \lambda z.\,z?(x).[\![P]\!]^3\rangle
\end{aligned}
$$

Proposition 8.1 (Type Preservation. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let P be a $\mathsf{HO}\pi^+$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then $(\!(\Gamma)\!)^3;\emptyset;(\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \triangleright \diamond$.*

Proof. The proof is a mechanical induction on the structure of P. Details of the proof in Proposition C.6 (page 84). ◻

Proposition 8.2 (Operational Correspondence. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$).

1. *Let $\Gamma;\emptyset;\Delta \vdash P$. $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ implies*
   a) *If $\ell \in \{(\nu\ \tilde{m})n!\langle \lambda x.\,Q\rangle, n?\langle \lambda x.\,Q\rangle\}$ then $(\!(\Gamma)\!)^3;(\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \stackrel{\ell'}{\longmapsto} (\!(\Delta')\!)^3 \vdash [\![P']\!]^3$ with $\{\!\{\ell\}\!\}^3 = \ell'$.*
   b) *If $\ell \notin \{(\nu\ \tilde{m})n!\langle \lambda x.\,Q\rangle, n?\langle \lambda x.\,Q\rangle, \tau\}$ then $(\!(\Gamma)\!)^3;(\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \stackrel{\ell}{\longmapsto} (\!(\Delta')\!)^3 \vdash [\![P']\!]^3$.*
   c) *If $\ell = \tau_\beta$ then $(\!(\Gamma)\!)^3;(\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \stackrel{\tau}{\longmapsto} \Delta'' \vdash R$ and $(\!(\Gamma)\!)^3(\!(\Delta')\!)^3[\![P']\!]^3 \approx^H \Delta'' R$.*
   d) *If $\ell = \tau$ and $\ell \neq \tau_\beta$ then $(\!(\Gamma)\!)^3;(\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^3 \vdash [\![P']\!]^3$.*
2. *Let $\Gamma;\emptyset;\Delta \vdash P$. $(\!(\Gamma)\!)^3;(\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \stackrel{\ell}{\longmapsto} (\!(\Delta'')\!)^3 \vdash Q$ implies*
   a) *If $\ell \in \{(\nu\ \tilde{m})n!\langle \lambda x.\,Q\rangle, n?\langle \lambda x.\,Q\rangle, \tau\}$ then $\Gamma;\Delta \vdash P \stackrel{\ell'}{\longmapsto} \Delta' \vdash P'$ with $\{\!\{\ell'\}\!\}^3 = \ell$ and $Q \equiv [\![P']\!]^3$.*
   b) *If $\ell \notin \{(\nu\ \tilde{m})n!\langle \lambda x.R\rangle, n?\langle \lambda x.R\rangle, \tau\}$ then $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ and $Q \equiv [\![P']\!]^3$.*
   c) *If $\ell = \tau$ then either $\Gamma;\Delta \vdash \Delta \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ with $Q \equiv [\![P']\!]^3$ or $\Gamma;\Delta \vdash \Delta \stackrel{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^3;(\!(\Delta'')\!)^3 \vdash Q \stackrel{\tau_\beta}{\longmapsto} (\!(\Delta'')\!)^3 \vdash [\![P']\!]^3$.*

Proof. The proof is an induction on the labelled transition system. The most interesting cases can be found in Proposition C.7 (page 85). □

Proposition 8.3 (Full Abstraction. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let P, Q $\mathsf{HO}\pi^+$ processes with $\Gamma;\emptyset;\Delta_1 \vdash P \triangleright \diamond$ and $\Gamma;\emptyset;\Delta_2 \vdash Q \triangleright \diamond$.*
Then $\Gamma;\Delta_1 \vdash P \approx^H \Delta_2 \vdash Q$ if and only if $(\!(\Gamma)\!)^3;(\!(\Delta_1)\!)^3 \vdash [\![P]\!]^3 \approx^H (\!(\Delta_2)\!)^3 \vdash [\![Q]\!]^3$

Proof. **Soundness Direction.**
We create the closure

$$\mathfrak{R} = \{\Gamma;\Delta_1 \vdash P\ ,\ \Delta_2 \vdash Q \ \mid\ (\!(\Gamma)\!)^3;(\!(\Delta_1)\!)^3 \vdash [\![P]\!]^3 \approx^H (\!(\Delta_2)\!)^3 \vdash [\![Q]\!]^3\}$$

It is straightforward to show that $\mathfrak{R}$ is a bisimulation if we follow Part 2 of Proposition 8.2 for subcases a and b. In subcase c we make use of Proposition 4.3.
Completeness Direction.
We create the closure

$$\mathfrak{R} = \{(\!(\Gamma)\!)^3;(\!(\Delta_1)\!)^3 \vdash [\![P]\!]^3\ ,\ (\!(\Delta_2)\!)^3 \vdash [\![Q]\!]^3 \ \mid\ \Gamma;\Delta_1 \vdash P \approx^H \Delta_2 \vdash Q\}$$

We show that $\mathfrak{R}$ is a bisimulation up to deterministic transitions by following Part 1 of Proposition 8.2. The proof is straightforward for subcases a), b) and d). In subcase c) we make use of Lemma 4.3. □

Proposition 8.4 (Precise encoding of $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$). *The encoding from $\mathcal{L}_{\mathsf{HO}\pi^+}$ to $\mathcal{L}_{\mathsf{HO}\pi}$ is precise.*

Proof. Syntactic requirements are easily derivable from the definition of the mappings in Figure 10. Semantic requirements are a consequence of Proposition 8.1, Proposition 8.2, and Proposition 8.3. □

8.2 Polyadic $\mathsf{HO}\pi$

Embedding polyadic name passing into the monadic name passing is well-studied in the literature. Using the linear typing, the preciseness (full abstraction) can be obtained [57]. Here we describe an encoding of $\mathsf{HO}\vec{\pi}$ into $\mathsf{HO}\pi$.

Modifications in Syntax, Reduction Semantics, and Types. The syntax of $\mathsf{HO}\vec{\pi}$ processes is obtained from the syntax for processes given in Figure 2 by considering values

$$V \ ::=\ \tilde{u} \ \mid\ \lambda \tilde{x}.P$$

and input prefixes $n?(\tilde{x}).P$. Thus, polyadicity arises both in (session) communications and abstractions. Reduction is then defined by the rules in Figure 3, excepting rules [App] and [Pass] which are replaced by rules

$$\begin{array}{lll}
[\mathrm{App}^p] & (\lambda \tilde{x}.P)\,\tilde{u} \longrightarrow P\{\tilde{u}/\tilde{x}\} & |\tilde{x}| = |\tilde{u}| \\
[\mathrm{Pass}^p] & n!\langle V\rangle.P_1 \mid \overline{n}?(\tilde{x}).P_2 \longrightarrow P_1 \mid P_2\{V/\tilde{x}\} & |V| = |\tilde{x}|
\end{array}$$

The syntax of types in Figure 3.1 is modified to include

$$
\begin{array}{rcl}
L & ::= & \tilde{C}\!\rightarrow\!\diamond \;\mid\; \tilde{C}\!\multimap\!\diamond \\
U & ::= & \tilde{C} \;\mid\; L
\end{array}
$$

instead of $L \;::=\; C\!\rightarrow\!\diamond \;\mid\; C\!\multimap\!\diamond$ and $U \;::=\; C \;\mid\; L$, respectively.

Definitions of type equivalence/duality and typing environments (Γ and Λ) are straightforward extensions of Definition 3.2, Definition 3.3, and Definition 3.4, respectively. Following [33,35] the type system for $\mathsf{HO}\vec{\pi}$ disallows polyadicity along shared names. Based on these modifications, the typing rules of Figure 4 are adapted in the expected way. In order to type polyadic values, we rely on the following rule:

$$
[\text{Pol}]\;\; \frac{V = a_i \ldots a_n \qquad \Gamma;\Lambda_i;\Delta_i \vdash u_i \triangleright C_i \qquad U = C_1 \ldots C_n}{\Gamma;\bigcup_{i\in I}\Lambda_i;\bigcup_{i\in I}\Delta_i \vdash V \triangleright U}
$$

Other rules are adjusted in the expected way, in order to accommodate polyadic values. Notice, however, that rules [Req] and [Acc] are kept unchanged, as they are used to type monadic exchanges along shared name prefixes. We now state type soundness for $\mathsf{HO}\vec{\pi}$; the proof is straightforward and omitted, for it follows closely the proof detailed in Appendix A.

Theorem 8.2 (Type Soundness for $\mathsf{HO}\vec{\pi}$).

1. *(Subject Congruence)* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *and* $P \equiv P'$ *implies* $\Gamma;\emptyset;\Delta \vdash P' \triangleright \diamond$.
2. *(Subject Reduction)* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *with balanced* Δ *and* $P \longrightarrow P'$ *implies* $\Gamma;\emptyset;\Delta' \vdash P' \triangleright \diamond$ *and either (i)* $\Delta = \Delta'$ *or (ii)* $\Delta \longrightarrow \Delta'$ *with* Δ' *balanced.*

As for the behavioral semantics for $\mathsf{HO}\vec{\pi}$, the set of action labels is kept unchanged. In fact, as V now stands for $\tilde{u}$ and $\lambda\tilde{x}.\,P$, labels $(\nu\,\tilde{m})n!\langle V\rangle$ and $n?\langle V\rangle$ require no modification. The LTS for $\mathsf{HO}\vec{\pi}$ is as for $\mathsf{HO}\pi$, excepting rule $\langle\text{App}\rangle$ which is replaced with the rule:

$$
(\lambda\tilde{x}.\,P)\,\tilde{u} \xrightarrow{\tau} P\{\tilde{u}/\tilde{x}\}
$$

The characteristic process and characteristic value definition (Definition 4.8) is extended to include the cases:

$$
\begin{array}{rcl}
[\![C_1 \ldots C_n]\!]^{u_1\cdots u_n} & \stackrel{\mathsf{def}}{=} & [\![C_1]\!]^{x_1} \mid \ldots \mid [\![C_n]\!]^{x_n} \\
[\![U_1 \ldots U_n]\!]_{\mathsf{c}} & \stackrel{\mathsf{def}}{=} & [\![U_1]\!]_{\mathsf{c}}, \ldots, [\![U_n]\!]_{\mathsf{c}}
\end{array}
$$

Thus, a polyadic type is inhabited by process whose parallel components inhabit type the individual components of the polyadic type. A polyadic value type is inhabited by a list of values which inhabit the individual components of the polyadic value. The rest of the behavioural semantics remains unchanged.

Encoding $\mathsf{HO}\vec{\pi}$ into $\mathsf{HO}\pi$. We slightly modify Definition 5.4 to capture that a label ℓ may be mapped into a sequence of labels $\tilde{\ell}$. Also, Definition 5.4 stays as the same assuming that if $P \stackrel{\ell}{\longmapsto} P'$ and $\{\!|\ell|\!\} = \{\ell_1, \ell_2, \cdots, \ell_m\}$ then $[\![P]\!] \stackrel{\{\!|\ell|\!\}}{\Longmapsto} [\![P']\!]$ should be understood as $[\![P]\!] \stackrel{\ell_1}{\Longmapsto} P_1 \stackrel{\ell_2}{\Longmapsto} P_2 \cdots \stackrel{\ell_m}{\Longmapsto} P_m = [\![P']\!]$, for some $P_1, P_2, \ldots, P_m$.

Fig. 11 Encoding of $\mathsf{HO}\vec{\pi}$ into $\mathsf{HO}\pi$ (cf. Definition 8.2). We assume that the rest of the encoding is homomorphic on the syntax of processes, types and labels, respectively.

Terms

$$
\begin{aligned}
[\![n!\langle u_1,\dots,u_n\rangle.P]\!]^4 &\stackrel{\text{def}}{=} n!\langle u_1\rangle.\dots;n!\langle u_n\rangle.[\![P]\!]^4\\
[\![n?(x_1,\dots,x_n).P]\!]^4 &\stackrel{\text{def}}{=} n?(x_1).\dots;n?(x_n).[\![P]\!]^4\\
[\![n!\langle\lambda x_1,\dots,x_n.\,Q\rangle.P]\!]^4 &\stackrel{\text{def}}{=} n!\langle\lambda z.\,z?(x_1).\dots;z?(x_n).[\![Q]\!]^4\rangle.[\![P]\!]^4\\
[\![x\,(u_1,\dots,u_n)]\!]^4 &\stackrel{\text{def}}{=} (\nu\ s)(x\,s \mid \overline{s}!\langle u_1\rangle.\dots;\overline{s}!\langle u_1\rangle.\mathbf{0})\\
[\![(\lambda x.\,P)\,(u_1,\dots,u_n)]\!]^4 &\stackrel{\text{def}}{=} (\nu\ s)((\lambda x.\,[\![P]\!]^4)\,s \mid \overline{s}!\langle u_1\rangle.\dots;\overline{s}!\langle u_1\rangle.\mathbf{0})
\end{aligned}
$$

Types

$$
\begin{aligned}
(\!(\,(C_1,\dots,C_n){\multimap}\diamond)\!)^4 &\stackrel{\text{def}}{=} (?(C_1);\dots;?(C_n);\mathsf{end}){\multimap}\diamond\\
(\!(\,(C_1,\dots,C_n){\to}\diamond)\!)^4 &\stackrel{\text{def}}{=} (?(C_1);\dots;?(C_n);\mathsf{end}){\to}\diamond\\
(\!(!\langle L\rangle;S)\!)^4 &\stackrel{\text{def}}{=} !\langle(\!(L)\!)^4\rangle;(\!(S)\!)^4\\
(\!(?(L);S)\!)^4 &\stackrel{\text{def}}{=} ?((\!(L)\!)^4);(\!(S)\!)^4\\
(\!(!\langle C_1,\dots,C_n\rangle;S)\!)^4 &\stackrel{\text{def}}{=} !\langle C_1\rangle;\dots;!\langle C_n\rangle;(\!(S)\!)^4\\
(\!(?(C_1,\dots,C_n);S)\!)^4 &\stackrel{\text{def}}{=} ?(C_1);\dots;!\langle C_n\rangle;(\!(S)\!)^4
\end{aligned}
$$

Labels

$$
\begin{aligned}
\{\!\{(\nu\ \tilde{m}')n!\langle m_1,\dots,m_n\rangle\}\!\}^4 &\stackrel{\text{def}}{=} (\nu\ \tilde{m_1}')n!\langle m_1\rangle\dots(\nu\ \tilde{m_n}')n!\langle m_n\rangle \qquad \begin{array}{l}\tilde{m_i}' = m_i \Leftrightarrow m_i \in \tilde{m}' \wedge\\ \tilde{m_i}' = \emptyset \Leftrightarrow m_i \notin \tilde{m}'\end{array}\\
\{\!\{n?\langle m_1,\dots,m_n\rangle\}\!\}^4 &\stackrel{\text{def}}{=} n?\langle m_1\rangle\dots n?\langle m_n\rangle\\
\{\!\{(\nu\ \tilde{m})n!\langle\lambda x_1,\dots,x_n.\,P\rangle\}\!\}4 &\stackrel{\text{def}}{=} (\nu\ \tilde{m})n!\langle\lambda z.\,z?(x_1).\dots;z?(x_n).[\![P]\!]^4\rangle\\
\{\!\{n?\langle\lambda x_1,\dots,x_n.\,P\rangle\}\!\}4 &\stackrel{\text{def}}{=} n?\langle\lambda z.\,z?(x_1).\dots;z?(x_n).[\![P]\!]^4\rangle\\
\{\!\{\tau_\beta\}\!\}^4 &\stackrel{\text{def}}{=} \tau_\beta,\tau_{\mathsf{s}},\dots,\tau_{\mathsf{s}}\\
\{\!\{\tau\}\!\}^4 &\stackrel{\text{def}}{=} \tau,\dots,\tau
\end{aligned}
$$

Let $\mathcal{L}_{\mathsf{HO}\vec{\pi}} = \langle \mathsf{HO}\vec{\pi}, \mathcal{T}_5, \stackrel{\ell}{\longmapsto}, \approx^H, \vdash\rangle$ where $\mathcal{T}_5$ is a set of types of $\mathsf{HO}\pi^+$; the typing $\vdash$ is defined in Figure 4 with polyadic types.

Definition 8.2 (Encoding from $\mathsf{HO}\vec{\pi}$ to $\mathsf{HO}\pi$). *Encoding $\langle [\![\cdot]\!]^4, (\!(\cdot)\!)^4, \{\!\{\cdot\}\!\}^4\rangle : \mathcal{L}_{\mathsf{HO}\vec{\pi}} \to \mathcal{L}_{\mathsf{HO}\pi}$ to be defined as in Figure 11.*

Proposition 8.5 (Type Preservation. From $\mathsf{HO}\vec{\pi}$ to $\mathsf{HO}\pi$). *Let P be a $\mathsf{HO}\vec{\pi}$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then $(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond$.*

Proof. By induction on the inference $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. See Proposition C.8 (Page 87) for details. □

Proposition 8.6 (Operational Correspondence. From $\mathsf{HO}\vec{\pi}$ to $\mathsf{HO}\pi$).

1. *Let $\Gamma;\emptyset;\Delta \vdash P$. Then $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ implies*

a) If $\ell = (\nu\ \tilde{m}')n!\langle\tilde{m}\rangle$ then $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell_1}{\longmapsto} \dots \stackrel{\ell_n}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P]\!]^4$ with $\{\!\{\ell\}\!\}^4 = \ell_1 \dots \ell_n$.

b) If $\ell = n?\langle\tilde{m}\rangle$ then $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell_1}{\longmapsto} \dots \stackrel{\ell_n}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P]\!]^4$ with $\{\!\{\ell\}\!\}^4 = \ell_1 \dots \ell_n$.

c) If $\ell \in \{(\nu\ \tilde{m})n!\langle\lambda\tilde{x}.R\rangle, n?\langle\lambda\tilde{x}.R\rangle\}$ then $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell'}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$ with $\{\!\{\ell\}\!\}^4 = \ell'$.

d) If $\ell \in \{n \oplus l, n\&l\}$ then $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$.

e) If $\ell = \tau_\beta$ then either $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\tau_\beta}{\longmapsto}\stackrel{\tau_{\mathsf{s}}}{\longmapsto} \dots \stackrel{\tau_{\mathsf{s}}}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$ with $\{\!\{\ell\}\!\} = \tau_\beta, \tau_{\mathsf{s}} \dots \tau_{\mathsf{s}}$.

f) If $\ell = \tau$ then $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\tau}{\longmapsto} \dots \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$ with $\{\!\{\ell\}\!\}^4 = \tau \dots \tau$.

2. Let $\Gamma; \emptyset; \Delta \vdash P$. $(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell_1}{\longmapsto} (\!(\Delta_1)\!)^4 \vdash P_1$ implies

a) If $\ell \in \{n?\langle m\rangle, n!\langle m\rangle, (\nu\ m)n!\langle m\rangle\}$ then $\Gamma; \Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^4; (\!(\Delta_1)\!)^4 \vdash P_1 \stackrel{\ell_2}{\longmapsto} \dots \stackrel{\ell_n}{\longmapsto} (\!(\Delta')\!)^4 \vdash (\!(P')\!)^4$ with $\{\!\{\ell\}\!\}^4 = \ell_1 \dots \ell_n$.

b) If $\ell \in \{(\nu\ \tilde{m})n!\langle\lambda x.R\rangle, n?\langle\lambda x.R\rangle\}$ then $\Gamma; \Delta \vdash P \stackrel{\ell'}{\longmapsto} \Delta' \vdash P'$ with $\{\!\{\ell'\}\!\}^4 = \ell$ and $P_1 \equiv [\![P']\!]^4$.

c) If $\ell \in \{n \oplus l, n\&l\}$ then $\Gamma; \Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ and $P_1 \equiv [\![P']\!]^4$.

d) If $\ell = \tau_\beta$ then $\Gamma; \Delta \vdash P \stackrel{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^4; (\!(\Delta_1)\!)^4 \vdash P_1 \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \dots \stackrel{\tau_{\mathsf{s}}}{\longmapsto} (\!(\Delta')\!)^4 \vdash (\!(P')\!)^4$ with $\{\!\{\ell\}\!\}^4 = \tau_\beta, \tau_{\mathsf{s}} \dots \tau_{\mathsf{s}}$.

e) If $\ell = \tau$ then $\Gamma; \Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^4; (\!(\Delta_1)\!)^4 \vdash P_1 \stackrel{\tau}{\longmapsto} \dots \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^4 \vdash (\!(P')\!)^4$ with $\{\!\{\ell\}\!\}^4 = \tau \dots \tau$.

Proof. We present the proof for the dyadic case in Proposition C.9 (Page 88). The polyadic case proof is an generalisation of the dyadic case proof. □

Proposition 8.7 (Full Abstraction. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let P, Q $\mathsf{HO}\vec{\pi}$ process with $\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond$ and $\Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond$. $\Gamma; \Delta_1 \vdash P \approx^H \Delta_2 \vdash Q$ if and only if $(\!(\Gamma)\!)^4; (\!(\Delta_1)\!)^4 \vdash [\![P]\!]^4 \approx^H (\!(\Delta_2)\!)^4 \vdash [\![Q]\!]^4$*

Proof. The proof for both direction is a consequence of Operational Correspondence, Proposition 8.6.

Soundness Direction.

We create the closure

$$\Re = \{\Gamma; \Delta_1 \vdash P\ ,\ \Delta_2 \vdash Q \mid (\!(\Gamma)\!)^4; (\!(\Delta_1)\!)^4 \vdash [\![P]\!]^4 \approx^H (\!(\Delta_2)\!)^4 \vdash [\![Q]\!]^4\}$$

It is straightforward to show that $\Re$ is a bisimulation if we follow Part 2 of Proposition 8.6.

Completeness Direction.

We create the closure

$$\Re = \{(\!(\Gamma)\!)^4; (\!(\Delta_1)\!)^4 \vdash [\![P]\!]^4\ ,\ (\!(\Delta_2)\!)^4 \vdash [\![Q]\!]^4 \mid \Gamma; \Delta_1 \vdash P \approx^H \Delta_2 \vdash Q\}$$

We show that $\Re$ is a bisimulation up to deterministic transitions by following Part 1 of Proposition 8.6. □

Proposition 8.8 (Precise encoding of $\mathsf{HO}\pi^{+}$ into $\mathsf{HO}\pi$). *The encoding from $\mathcal{L}_{\mathsf{HO}\vec{\pi}}$ to $\mathcal{L}_{\mathsf{HO}\pi}$ is precise.*

Proof. Syntactic requirements are easily derivable from the definition of the mappings in Figure 11. Semantic requirements are a consequence of Proposition 8.5, Proposition 8.6, and Proposition 8.7. ◻

9 Related Work

Expressiveness in Concurrency. There is a vast literature on expressiveness studies for process calculi; we refer to [39] for a survey (see also [40, § 2.3]). In particular, the expressive power of the π-calculus has received much attention. Studies cover, e.g., relationships between first-order and higher-order concurrency (see, e.g., [48,47]), comparisons between synchronous and asynchronous communication (see, e.g., [7,37,2]), and (non)encodability issues for different choice operators (see, e.g., [36,42]). To substantiate claims related to (relative) expressive power, early works appealed to different definitions of encoding. Later on, proposals of abstract frameworks which formalise the notion of encoding and state associated syntactic and semantic criteria were put forward; recent proposals are [16,12,54]. These frameworks are applicable to different calculi, and have shown useful to clarify known results and to derive new ones. Our formulation of (precise) typed encoding (Definition 5.5) builds upon existing proposals (including [37,16,28]) in order to account for the session type systems associated to the process languages under comparison.

Expressiveness of Higher-Order Process Calculi. Early expressiveness studies for higher-order calculi are [52,48]; more recent works include [8,28,29,55,56]. Due to the close relationship between higher-order process calculi and functional calculi, works devoted to encoding (variants of) the λ-calculus into (variants of) the π-calculus (see, e.g., [45,11,58,3,51]) are also worth mentioning. The work [48] gives an encoding of the higher-order π-calculus into the first-order π-calculus which is fully abstract with respect to reduction-closed, barbed congruence. A basic form of input/output types is used in [49], where the encoding in [48] is casted in the asynchronous setting, with output and applications coalesced in a single construct. Building upon [49], a simply typed encoding for synchronous processes is given in [50]; the reverse encoding (i.e., first-order communication into higher-order processes) is also studied there for an asynchronous, localised π-calculus (only the output capability of names can be sent around). The work [47] studies hierarchies for calculi with *internal* first-order mobility and with higher-order mobility without name-passing (similarly as the subcalculus HO). The hierarchies are based on expressivity: formally defined according to the order of types needed in typing, they describe different "degrees of mobility". Via fully abstract encodings, it is shown that that name- and process-passing calculi with equal order of types have the same expressiveness. With respect to these previous results, our approach based on session types has several important consequences and allows us to derive new results. Our study reinforces the intuitive view of "encodings as protocols", namely session protocols which enforce precise linear and shared disciplines for names, a distinction not investigated in [48,49]. In turn, the linear/shared distinction is central in proper definitions of trigger processes, which are essential to encodings and behavioural equivalences. More interestingly, we showed that HO, a minimal higher-order session calculus (no name passing, only first-order application) suffices to encode π (the session calculus with name passing) but also $\mathsf{HO}\pi$ and its extension with higher-order applications (denoted $\mathsf{HO}\pi^{+}$). Thus, using session types all these calculi are shown to be equally expressive with fully abstract encodings. To our knowledge, these are the first expressiveness results of this kind.

Other related works are [8,55,29]. The paper [8] proposes a fully abstract, continuation-passing style encoding of the π-calculus into Homer, a rich higher-order process calculus with explicit locations, local names, and nested locations. The work [55] studies the encodability of the higher-order π-calculus (extended with a relabelling operator) into the first-order π-calculus; encodings in the reverse direction are also proposed, following [52]. A minimal calculus of higher-order concurrency is studied in [29]: it lacks restriction, name passing, output prefix (so communication is asynchronous), and constructs for infinite behaviour. Nevertheless, this calculus (a sublanguage of HO) is shown to be Turing complete. Moreover, strong bisimilarity is decidable and coincides with reduction-closed, barbed congruence.

Building upon [53], the work [55] studies the (non)encodability of the π-calculus into a higher-order π-calculus with a powerful name relabelling operator, which is shown to be essential in encoding name-passing. A core higher-order calculus is studied in [29]: it lacks restriction, name passing, output prefix and constructs for infinite behaviour. This calculus has a simple notion of bisimilarity which coincides with reduction-closed, barbed congruence. The absence of restriction plays a key role in the characterisations in [29]; hence, our characterisation of contextual equivalence for HO (which has restriction) cannot be derived from that in [29].

In [28] the core calculus in [29] is extended with restriction, synchronous communication, and polyadicity. It is shown that synchronous communication can encode asynchronous communication, and that process passing polyadicity induces a hierarchy in expressive power. The paper [56] complements [28] by studying the expressivity of second-order process abstractions. Polyadicity is shown to induce an expressiveness hierarchy; also, by adapting the encoding in [48], process abstractions are encoded into name abstractions. In contrast, we give a fully abstract encoding of $\mathsf{HO}\vec{\pi}^{+}$ into HO that preserves session types; this improves [28,56] by enforcing linearity disciplines on process behaviour. The focus of [28,56] is on the expressiveness of untyped, higher-order processes; they do not address tractable equivalences for processes (such as higher-order and characteristic bisimulations) which only require observation of finite higher-order values, whose formulations rely on session types.

Session Typed Processes. The works [10,9] study encodings of binary session calculi into a linearly typed π-calculus. While [10] gives a precise encoding of π into a linear calculus (an extension of [3]), the work [9] gives the operational correspondence (without full abstraction, cf. Definition 5.3-4) for the first- and higher-order π-calculi into [23]. They investigate an embeddability of two different typing systems; by the result of [10], $\mathsf{HO}\pi^{+}$ is encodable into the linearly typed π-calculi.

The syntax of $\mathsf{HO}\pi$ is a subset of that in [33,35]. The work [33] develops a full higher-order session calculus with process abstractions and applications; it admits the type $U = U_1 \to U_2 \ldots U_n \to \diamond$ and its linear type U^1 which corresponds to $\tilde{U} \to \diamond$ and $\tilde{U} \multimap \diamond$ in a super-calculus of $\mathsf{HO}\pi^+$ and $\mathsf{HO}\vec{\pi}$. Our results show that the calculus in [33] is not only expressed but also reasoned in HO (with limited form of arrow types, $C \to \diamond$ and $C \multimap \diamond$), via precise encodings. None of the above works proposes tractable bisimulations for higher-order processes.

Other Works on Typed Behavioural Equivalences. Since types can limit contexts (environments) where processes can interact, typed equivalences usually offer *coarse* semantics than untyped semantics. The work [43] demonstrated the IO-subtyping can equate the optimal encoding of the λ-calculus by Milner which was not i n the untyped polyadic π-calculus [31]. After [43], many works on typed π-calculi have investigated correctness of encodings of known concurrent and sequential calculi in order to examine semantic effects of proposed typing systems.

The type discipline closely related to session types is a family of linear typing systems. The work [23] first proposed a linearly typed reduction-closed, barbed congruence and reasoned a tail-call optimisation of higher-order functions which are encoded as processes. The work [57] had used a bisimulation of graph-based types to prove the full abstraction of encodings of the polyadic synchronous π-calculus into the monadic synchronous π-calculus. Later typed equivalences of a family of linear and affine calculi [3,58,4] were used to encode PCF [44,30], the simply typed λ-calculi with sums and products, and system F [15] fully abstractly (a fully abstract encoding of the λ-calculi was an open problem in [31]). The work [59] proposed a new bisimilarity method associated with linear type structure and strong normalisation. It presented applications to reason secrecy in programming languages. A subsequent work [20] adapted these results to a practical direction. It proposes new typing systems for secure higher-order and multi-threaded programming languages. In these works, typed properties, linearity and liveness, play a fundamental role in the analysis. In general, linear types are suitable to encode "sequentiality" in the sense of [21,1].

Typed Behavioural Equivalences. This work follows the principles for session type behavioural semantics in [27,26,41] where a bisimulation is defined on a LTS that assumes a session typed observer. Our theory for higher-order session types differentiates from the work in [27,26], which considers the first-order binary and multiparty session types, respectively. The work [41] gives a behavioural theory for a logically motivated language of binary sessions without shared names.

Our approach for the higher-order builds upon techniques by Sangiorgi [48,46] and Jeffrey and Rathke [22]. The work [48] introduced the first fully-abstract encoding from the higher-order π-calculus into the π-calculus. Sangiorgi's encoding is based on the idea of a replicated input-guarded process (called a trigger process). We use a similar replicated triggered process to encode $\mathsf{HO}\pi$ into π (Definition 6.4). Operational correspondence for the triggered encoding is shown using a context bisimulation with first-order labels. To deal with the issue of context bisimilarity, Sangiorgi proposes *normal bisimilarity*, a tractable equivalence without universal quantification. To prove that context and normal bisimilarities coincide, [48] uses triggered processes. Triggered bisimulation is also defined on first-order labels where the contextual bisimulation is restricted to arbitrary trigger substitution. This characterisation of context bisimilarity was refined in [22] for calculi with recursive types, not addressed in [46,48] and relevant in our work. The bisimulation in [22] is based on an LTS which is extended with trigger meta-notation. As in [46,48], the LTS in [22] observes first-order triggered values instead of higher-order values, offering a more direct characterisation of contextual equivalence and lifting the restriction to finite types.

We contrast the approach in [22] and our approach based on higher-order and characteristic bisimilarities. Below we use the notations adopted in [22].

i) The work [22] extends the first-order LTS for a trigger interaction whereas our work uses the higher-order LTS.
ii) The output of a higher-order value $\lambda x. Q$ on name n in [22] requires the output of a fresh trigger name t (notation τ_t) on channel n and then the introduction of a replicated triggered process (notation $(t \Leftarrow (x)Q)$). Hence we have:

$$P \xrightarrow{(\nu\ t)n!\langle \tau_t \rangle} P' \mid (t \Leftarrow (x)Q) \xrightarrow{t?\langle v \rangle} P' \mid (x)Q\, v \mid (t \Leftarrow (x)Q)$$

In our characteristic bisimulation, we only observe an output of a value that can be either first- or higher-order as follows:

$$P \overset{n!\langle V \rangle}{\longmapsto} P'$$

with $V \equiv \lambda x. Q$ or $V = m$.
A non-replicated triggered process $(t \Leftarrow V)$ appears in the parallel context of the acting process when we compare two processes for behavioural equality (cf. Definition 4.13). Using the LTS in Definition 4.1 we can obtain:

$$\begin{array}{rll} P' \mid t \Leftarrow \lambda x. Q & \xrightarrow{\lambda z. z?(y).* t?(x).(y\, x)} & P' \mid (\nu\ s)(s?(y).* t?(x).(y\, x) \mid s!\langle \lambda x. Q \rangle.\mathbf{0}) \\ & \xrightarrow{\tau} & P' \mid * t?(y).((\lambda x. Q)\, y) \end{array}$$

that simulates the approach in [22].
In addition, the output of the characteristic bisimulation differentiates from the approach in [22] as listed below:

– The typed LTS predicts the case of linear output values and will never allow replication of such a value; if V is linear the input action would have no replication operator, as $\lambda z. z?(y).t?(x).(y\, x)$.
– The characteristic bisimulation introduces a uniform approach not only for higher-order values but for first-order values as well, i.e. triggered process can accept any process that can substitute a first-order value as well. This is derived from the fact that the $\mathsf{HO}\pi$-calculus makes no use of a matching operator, in contrast to the calculus defined in [22]) where name matching is crucial to prove completeness of the bisimilarity relation. Instead of a matching operator, we use types: a characteristic value inhabiting a type enables the simplest form of interactions with the environment.

- Our $\mathsf{HO}\pi$-calculus requires only first-order applications. Higher-order applications, as in [22], are presented as an extension in the $\mathsf{HO}\pi^+$ calculus.
- Our trigger process is non-replicated. It guards the output value with a higher-order input prefix. The functionality of the input is then used to simulate the contextual bisimilarity that subsumes the replicated trigger approach (cf. Section 4.5). The transformation of an output action as an input action allows for treating an output using the restricted LTS (Definition 4.10):

$$P' \mid t \Leftarrow \lambda x.\, Q \overset{t?\langle \lambda x.\, [\![U]\!]^x \rangle}{\longmapsto} P' \mid (\nu\ s)([\![U]\!]^x\, s \mid \overline{s}!\langle \lambda x.\, Q \rangle.\mathbf{0})$$

iii) The input of a higher-order value in the [22] requires the input of a meta-syntactic fresh trigger, which then substituted on the application variable, thus the meta-syntax is extended to represent applications, e.g.:

$$n?(x).P \overset{n?\langle \tau_k \rangle}{\longrightarrow} ((\lambda x.\, P)\, \tau_k) \overset{\tau}{\longrightarrow} P\{\tau_k/x\}$$

Every instance of process variable x in P being substituted with trigger value τ_k to give an application of the form $(\tau_k\, x)$. In contrast the approach in the characteristic bisimulation observes the triggered value $\lambda z.\, t?(x).(x\, z)$ as an input instead of the meta-syntactic trigger:

$$n?(x).P \overset{n?\langle \lambda z.\, t?(x).(x\, z) \rangle}{\longmapsto} P\{\lambda z.\, t?(x).(x\, z)/x\}$$

Every instance of process variable x in P is substituted to give application of the form $(\lambda z.\, t?(x).(x\, z))\, v$ Note that in the characteristic bisimulation, we can also observe a characteristic process as an input.

iv) Triggered applications in [22] are observed as an output of the application value over the fresh trigger name:

$$\tau_k\, v \overset{k!\langle v \rangle}{\longrightarrow} \mathbf{0}$$

In contrast in the characteristic bisimulation we have two kind of applications: i) the trigger value application allows us to simulate an application on a fresh trigger name. ii) the characteristic value application allows us to inhabit an application value and observe the interaction its interaction with the environment as below:

$$(\lambda z.\, t?(x).(x\, z))\, v \overset{\tau}{\longrightarrow} t?(x).(x\, v) \overset{t?\langle \lambda x.\, [\![U]\!]^x \rangle}{\longrightarrow} (\lambda x.\, [\![U]\!]^x)\, v \overset{\tau}{\longrightarrow} [\![U]\!]^x\{v/x\}$$

References

1. Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *TCS*, 163:409–470, 2000.
2. Romain Beauxis, Catuscia Palamidessi, and Frank D. Valencia. On the asynchronous nature of the asynchronous pi-calculus. In *Concurrency, Graphs and Models*, volume 5065 of *LNCS*, pages 473–492. Springer, 2008.
3. Martin Berger, Kohei Honda, and Nobuko Yoshida. Sequentiality and the π-calculus. In *Proc. TLCA'01*, volume 2044 of *LNCS*, pages 29–45, 2001.

4. Martin Berger, Kohei Honda, and Nobuko Yoshida. Genericity and the pi-calculus. *Acta Inf.*, 42(2-3):83–141, 2005.
5. Giovanni Bernardi, Ornela Dardha, Simon J. Gay, and Dimitrios Kouzapas. On duality relations for session types. In *TGC*, LNCS, 2014. To appear.
6. Viviana Bono and Luca Padovani. Typing copyless message passing. *LMCS*, 8(1), 2012.
7. Gérard Boudol. Asynchrony and the π-calculus (note). Technical report, Rapport de Recherche 1702, INRIA, Sophia-Antipolis, 1992.
8. Mikkel Bundgaard, Thomas T. Hildebrandt, and Jens Chr. Godskesen. A cps encoding of name-passing in higher-order mobile embedded resources. *Theor. Comput. Sci.*, 356(3):422–439, 2006.
9. Ornela Dardha, Elena Giachino, and Davide Sangiorgi. Session types revisited. In *Proc. of the 14th symposium on Principles and practice of declarative programming*, PPDP'12, pages 139–150, New York, NY, USA, 2012. ACM.
10. Romain Demangeon and Kohei Honda. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR*, volume 6901 of *LNCS*, pages 280–296. Springer, 2011.
11. Yuxi Fu. Variations on mobile processes. *Theor. Comput. Sci.*, 221(1-2):327–368, 1999.
12. Yuxi Fu and Hao Lu. On the expressiveness of interaction. *Theor. Comput. Sci.*, 411(11-13):1387–1451, 2010.
13. Simon Gay and Malcolm Hole. Subtyping for Session Types in the Pi-Calculus. *Acta Informatica*, 42(2/3):191–225, 2005.
14. Simon J. Gay and Vasco Thudichum Vasconcelos. Linear type theory for asynchronous session types. *J. Funct. Program.*, 20(1):19–50, 2010.
15. Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. CUP, 1989.
16. Daniele Gorla. Towards a unified approach to encodability and separation results for process calculi. *Inf. Comput.*, 208(9):1031–1053, 2010.
17. Jan Friso Groote and M. P. A. Sellink. Confluence for process verification. *Theor. Comput. Sci.*, 170(1-2):47–81, 1996.
18. Matthew Hennessy. *A Distributed Pi-Calculus*. CUP, 2007.
19. Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138. Springer, 1998.
20. Kohei Honda and Nobuko Yoshida. A uniform type structure for secure information flow. *TOPLAS*, 29(6), 2007.
21. J. Martin E. Hyland and C. H. Luke Ong. On full abstraction for PCF. *Inf. & Comp.*, 163:285–408, 2000.
22. Alan Jeffrey and Julian Rathke. Contextual equivalence for higher-order pi-calculus revisited. *Logical Methods in Computer Science*, 1(1), 2005.
23. Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the Pi-Calculus. *TOPLAS*, 21(5):914–947, September 1999.
24. Vasileios Koutavas and Matthew Hennessy. A testing theory for a higher-order cryptographic language. In *ESOP*, volume 6602 of *LNCS*, pages 358–377, 2011.
25. Dimitrios Kouzapas. *A study of bisimulation theory for session types*. PhD thesis, Department of Computing, Imperial College London, June 2013.
26. Dimitrios Kouzapas and Nobuko Yoshida. Globally governed session semantics. *LMCS*, 10(4), 2014.
27. Dimitrios Kouzapas, Nobuko Yoshida, Raymond Hu, and Kohei Honda. On asynchronous eventful session semantics. *MSCS*, 2015.
28. Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, and Alan Schmitt. On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In *Proc. of ICALP 2010*, volume 6199, pages 442–453. Springer, 2010.

29. Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, and Alan Schmitt. On the expressiveness and decidability of higher-order process calculi. *Inf. Comput.*, 209(2):198–226, 2011.
30. Robin Milner. Fully abstract models of typed lambda-calculi. *TCS*, 4(1):1 – 22, 1977.
31. Robin Milner. Functions as processes. *MSCS*, 2(2):119–141, 1992.
32. Robin Milner and Davide Sangiorgi. Barbed bisimulation. volume 623 of *LNCS*, pages 685–695, 1992.
33. Dimitris Mostrous and Nobuko Yoshida. Two session typing systems for higher-order mobile processes. In *TLCA'07*, volume 4583 of *LNCS*, pages 321–335. Springer, 2007.
34. Dimitris Mostrous and Nobuko Yoshida. Session-based communication optimisation for higher-order mobile processes. In *TLCA'09*, volume 5608 of *LNCS*, pages 203–218. Springer, 2009.
35. Dimitris Mostrous and Nobuko Yoshida. Session Typing and Asynchronous Subtying for Higher-Order π-Calculus. *Info.& Comp.*, 2015. To appear.
36. Uwe Nestmann. What is a "good" encoding of guarded choice? *Inf. Comput.*, 156(1-2):287–319, 2000. A preliminary version appeared in EXPRESS'97.
37. Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.
38. Catuscia Palamidessi, Vijay A. Saraswat, Frank D. Valencia, and Björn Victor. On the expressiveness of linearity vs persistence in the asychronous pi-calculus. In *Proc. of LICS 2006*, pages 59–68, 2006.
39. Joachim Parrow. Expressiveness of process algebras. *Electr. Notes Theor. Comput. Sci.*, 209:173–186, 2008.
40. Jorge A. Pérez. *Higher-Order Concurrency: Expressiveness and Decidability Results*. PhD thesis, University of Bologna, 2010.
41. Jorge A. Pérez, Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logical relations and observational equivalences for session-based concurrency. *Inf. Comput.*, 239:254–302, 2014.
42. Kirstin Peters, Uwe Nestmann, and Ursula Goltz. On distributability in process calculi. In *Proc. of ESOP 2013*, volume 7792, pages 310–329. Springer, 2013.
43. B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *MSCS*, 6(5):409–454, 1996.
44. Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223 – 255, 1977.
45. D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *7th LICS Conf.*, pages 102–109. IEEE Computer Society Press, 1992.
46. D. Sangiorgi. Bisimulation for Higher-Order Process Calculi. *Inf. & Comp.*, 131(2):141–178, 1996.
47. D. Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *TCS*, 167(2):235–274, 1996.
48. Davide Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher Order Paradigms*. PhD thesis, University of Edinburgh, 1992.
49. Davide Sangiorgi. Asynchronous process calculi: the first- and higher-order paradigms. *Theor. Comput. Sci.*, 253(2):311–350, 2001.
50. Davide Sangiorgi and David Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
51. Davide Sangiorgi and Xian Xu. Trees from functions as processes. In *Proc. of CONCUR 2014*, volume 8704, pages 78–92. Springer, 2014.
52. Bent Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Dept. of Comp. Sci., Imperial College, 1990.
53. Bent Thomsen. Plain CHOCS: A Second Generation Calculus for Higher Order Processes. *Acta Informatica*, 30(1):1–59, 1993.

54. Rob J. van Glabbeek. Musings on encodings and expressiveness. In *Proc. of EXPRESS/SOS 2012*, volume 89 of *EPTCS*, pages 81–98, 2012.
55. Xian Xu. Distinguishing and relating higher-order and first-order processes by expressiveness. *Acta Informatica*, 49(7-8):445–484, 2012.
56. Xian Xu, Qiang Yin, and Huan Long. On the expressiveness of parameterization in process-passing. In *Proc. of WS-FM 2013*, volume 8379, pages 147–167. Springer, 2014.
57. Nobuko Yoshida. Graph types for monadic mobile processes. In *FSTTCS*, volume 1180 of *LNCS*, pages 371–386. Springer, 1996.
58. Nobuko Yoshida, Martin Berger, and Kohei Honda. Strong normalisation in the π-calculus. *Inf. Comput.*, 191(2):145–202, 2004.
59. Nobuko Yoshida, Kohei Honda, and Martin Berger. Linearity and bisimulation. In *FoSSaCs02*, volume 2303 of *LNCS*, pages 417–433. Springer, 2002.

A Type Soundness

We state type soundness of our system. As our typed process framework is a sub-calculus of that considered by Mostrous and Yoshida, the proof of type soundness requires notions and properties which are specific instances of those already shown in [35]. We begin by stating weakening and strengthening lemmas, which have standard proofs.

Lemma A.1 (Weakening - Lemma C.2 in [35]).

– If $\Gamma;\Lambda;\Delta \vdash P \triangleright \diamond$ and $x \notin \mathtt{dom}(\Gamma,\Lambda,\Delta)$ then $\Gamma \cdot x : S \rightarrow\diamond;\Lambda;\Delta \vdash P \triangleright \diamond$

Lemma A.2 (Strengthening - Lemmas C.3 and C.4 in [35]).

– If $\Gamma \cdot x : S \rightarrow\diamond;\Lambda;\Delta \vdash P \triangleright \diamond$ and $x \notin \mathtt{fpv}(P)$ then $\Gamma;\Lambda;\Delta \vdash P \triangleright \diamond$
– If $\Gamma;\Lambda;\Delta \cdot s : \mathtt{end} \vdash P \triangleright \diamond$ and $s \notin \mathtt{fn}(P)$ then $\Gamma;\Lambda;\Delta \vdash P \triangleright \diamond$

Lemma A.3 (Substitution Lemma - Lemma C.10 in [35]).

1. Suppose $\Gamma;\Lambda;\Delta \cdot x : S \vdash P \triangleright \diamond$ and $s \notin \mathtt{dom}(\Gamma,\Lambda,\Delta)$. Then $\Gamma;\Lambda;\Delta \cdot s : S \vdash P\{^{s}/_{x}\} \triangleright \diamond$.
2. Suppose $\Gamma \cdot x : \langle U\rangle;\Lambda;\Delta \vdash P \triangleright \diamond$ and $a \notin \mathtt{dom}(\Gamma,\Lambda,\Delta)$. Then $\Gamma \cdot a : \langle U\rangle;\Lambda;\Delta \vdash P\{^{a}/_{x}\} \triangleright \diamond$.
3. Suppose $\Gamma;\Lambda_1 \cdot x : C{\multimap}\diamond;\Delta_1 \vdash P \triangleright \diamond$ and $\Gamma;\Lambda_2;\Delta_2 \vdash V \triangleright C{\multimap}\diamond$ with Λ_1,Λ_2 and Δ_1,Δ_2 defined. Then $\Gamma;\Lambda_1 \cdot \Lambda_2;\Delta_1 \cdot \Delta_2 \vdash P\{^{V}/_{x}\} \triangleright \diamond$.
4. Suppose $\Gamma \cdot x : C{\rightarrow}\diamond;\Lambda;\Delta \vdash P \triangleright \diamond$ and $\Gamma;\emptyset;\emptyset \vdash V \triangleright C{\rightarrow}\diamond$. Then $\Gamma;\Lambda;\Delta \vdash P\{^{V}/_{x}\} \triangleright \diamond$.

Proof. In all four parts, we proceed by induction on the typing for P, with a case analysis on the last applied rule. □

We now state the instance of type soundness that we can derive from [35]. It is worth noticing the definition of structural congruence in [35] is richer. Also, their statement for subject reduction relies on an ordering on typings associated to queues and other runtime elements (such extended typings are denoted Δ in [35]). Since we are working with synchronous communication we can omit such an ordering.

We now repeat the statement of Theorem 3.1 in Page 12:

Theorem A.1 (Type Soundness - Theorem 3.1).

1. *(Subject Congruence) Suppose $\Gamma;\Lambda;\Delta \vdash P \triangleright \diamond$. Then $P \equiv P'$ implies $\Gamma;\Lambda;\Delta \vdash P' \triangleright \diamond$.*
2. *(Subject Reduction) Suppose $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ with balanced Δ. Then $P \longrightarrow P'$ implies $\Gamma;\emptyset;\Delta' \vdash P' \triangleright \diamond$ and $\Delta = \Delta'$ or $\Delta \longrightarrow \Delta'$.*

Proof. Part (1) is standard, using weakening and strengthening lemmas. Part (2) proceeds by induction on the last reduction rule used. Below, we give some details:

1. Case [App]: Then we have

$$P = (\lambda x.\, Q)\, u \longrightarrow Q\{^{u}/_{x}\} = P'$$

Suppose $\Gamma; \emptyset; \Delta \vdash (\lambda x. Q)\, u \triangleright \diamond$. We examine one possible way in which this assumption can be derived; other cases are similar or simpler:

$$\dfrac{\dfrac{\Gamma; \emptyset; \Delta \cdot \{x : S\} \vdash Q \triangleright \diamond \quad \Gamma'; \emptyset; \{x : S\} \vdash x \triangleright S}{\Gamma; \emptyset; \Delta \vdash \lambda x. Q \triangleright S \multimap \diamond} \qquad \dfrac{}{\Gamma; \emptyset; \{u : S\} \vdash u \triangleright S}}{\Gamma; \emptyset; \Delta \cdot u : S \vdash (\lambda x. Q)\, u \triangleright \diamond}$$

Then, by combining premise $\Gamma; \emptyset; \Delta \cdot \{x : S\} \vdash Q \triangleright \diamond$ with the substitution lemma (Lemma 3.1(1)), we obtain $\Gamma; \emptyset; \Delta \cdot u : S \vdash Q\{^{u}/x\} \triangleright \diamond$, as desired.

2. Case [Pass]: There are several sub-cases, depending on the type of the communication subject n and the type of the object V. We analyze two representative sub-cases:
   (a) n is a shared name and V is a name v. Then we have the following reduction:

$$P = n!\langle v \rangle.Q_1 \mid n?(x).Q_2 \longrightarrow Q_1 \mid Q_2\{^{v}/x\} = P'$$

By assumption, we have the following typing derivation:

$$\dfrac{(29) \quad (30)}{\Gamma; \emptyset; \Delta_1 \cdot \{v : S\} \cdot \Delta_3 \vdash n!\langle v \rangle.Q_1 \mid n?(x).Q_2 \triangleright \diamond}$$

where (29) and (30) are as follows:

$$\dfrac{\Gamma' \cdot n : \langle S \rangle; \emptyset; \emptyset \vdash n \triangleright \langle S \rangle \quad \Gamma; \emptyset; \Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma; \emptyset; \{v : S\} \vdash v \triangleright S}{\Gamma; \emptyset; \Delta_1 \cdot \{v : S\} \vdash n!\langle v \rangle.Q_1 \triangleright \diamond} \tag{29}$$

$$\dfrac{\Gamma' \cdot n : \langle S \rangle; \emptyset; \emptyset \vdash n \triangleright \langle S \rangle \quad \Gamma; \emptyset; \Delta_3 \cdot x : S \vdash Q_2 \triangleright \diamond}{\Gamma; \emptyset; \Delta_3 \vdash n?(x).Q_2 \triangleright \diamond} \tag{30}$$

Now, by applying Lemma 3.1(1) on $\Gamma; \emptyset; \Delta_3 \cdot x : S \vdash Q_2 \triangleright \diamond$ we obtain

$$\Gamma; \emptyset; \Delta_3 \cdot v : S \vdash Q_2\{^{v}/x\} \triangleright \diamond$$

and the case is completed by using rule [Par] with this judgment:

$$\dfrac{\Gamma; \emptyset; \Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma; \emptyset; \Delta_3 \cdot v : S \vdash Q_2\{^{v}/x\} \triangleright \diamond}{\Gamma; \emptyset; \Delta_1 \cdot \Delta_3 \cdot v : S \vdash Q_1 \mid Q_2\{^{v}/x\} \triangleright \diamond}$$

Observe how in this case the session environment does not reduce.

   (b) n is a shared name and V is a higher-order value. Then we have the following reduction:

$$P = n!\langle V \rangle.Q_1 \mid n?(x).Q_2 \longrightarrow Q_1 \mid Q_2\{^{V}/x\} = P'$$

By assumption, we have the following typing derivation (below, we write L to stand for $C \to \diamond$ and Γ to stand for $\Gamma' \setminus \{x : L\}$).

$$\dfrac{(31) \quad (32)}{\Gamma; \emptyset; \Delta_1 \cdot \Delta_3 \vdash n!\langle v \rangle.Q_1 \mid n?(x).Q_2 \triangleright \diamond}$$

where (31) and (32) are as follows:

$$\dfrac{\Gamma; \emptyset; \emptyset \vdash n \triangleright \langle L \rangle \quad \Gamma; \emptyset; \Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma; \emptyset; \emptyset \vdash V \triangleright L}{\Gamma; \emptyset; \Delta_1 \vdash n!\langle V \rangle.Q_1 \triangleright \diamond} \tag{31}$$

$$\dfrac{\Gamma'; \emptyset; \emptyset \vdash n \triangleright \langle L \rangle \quad \Gamma'; \emptyset; \Delta_3 \vdash Q_2 \triangleright \diamond \quad \Gamma'; \emptyset; \emptyset \vdash x \triangleright L}{\Gamma; \emptyset; \Delta_3 \vdash n?(x).Q_2 \triangleright \diamond} \tag{32}$$

Now, by applying Lemma 3.1(4) on $\Gamma' \setminus \{x : L\}; \emptyset; \Delta_3 \vdash Q_2 \triangleright \diamond$ and $\Gamma; \emptyset; \emptyset \vdash V \triangleright L$ we obtain

$$\Gamma; \emptyset; \Delta_3 \vdash Q_2\{^V/_x\} \triangleright \diamond$$

and the case is completed by using rule [Par] with this judgment:

$$\frac{\Gamma;\emptyset;\Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma; \emptyset; \Delta_3 \vdash Q_2\{^V/_x\} \triangleright \diamond}{\Gamma;\emptyset;\Delta_1 \cdot \Delta_3 \vdash Q_1 \mid Q_2\{^V/_x\} \triangleright \diamond}$$

Observe how in this case the session environment does not reduce.

3. Case [Sel]: The proof is standard, the session environment reduces.
4. Cases [Par] and [Res]: The proof is standard, exploiting induction hypothesis.
5. Case [Cong]: follows from Theorem 3.1 (1).

□

B Behavioural Semantics

We present the proofs for the theorems in Section 4.

B.1 Proof of Theorem 4.1

We split the proof of Theorem 4.1 (Page 22) into several lemmas:

- Lemma B.1 establishes $\approx^H = \approx^C$.s
- Lemma B.4 exploits the process substitution result (Lemma 4.2) to prove that $\approx^H \subseteq \approx$.
- Lemma B.5 shows that $\approx$ is a congruence which implies $\approx \subseteq \cong$.
- Lemma B.8 shows that $\cong \subseteq \approx^H$.

We now proceed to state and proof these lemmas, together with some auxiliary results.

Lemma B.1. $\approx^H = \approx^C$.

Proof. We only prove the direction $\approx^H \subseteq \approx^C$. The direction $\approx^C \subseteq \approx^H$ is similar.
Consider

$$\mathfrak{R} = \{\Gamma;\Delta_1 \vdash P \,,\, \Delta_2 \vdash Q \;\mid\; \Gamma;\Delta_1 \vdash P \approx^H \Delta_2 \vdash Q\}$$

We show that $\mathfrak{R}$ is a characteristic bisimulation. The proof does a case analysis on the transition label ℓ.
- Case $\ell = (\nu\, \tilde{m}_1)n!\langle V_1\rangle$ is the non-trivial case.
If

$$\Gamma;\Delta_1 \vdash P \stackrel{(\nu\, \tilde{m}_1)n!\langle V_1\rangle}{\longmapsto} \Delta_1' \vdash P' \tag{33}$$

then $\exists Q, V_2$ such that

$$\Gamma;\Delta_2 \vdash Q \stackrel{(\nu\, \tilde{m}_2)n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q' \tag{34}$$

and for fresh t:

$$\begin{array}{r}\Gamma;\emptyset;\,\Delta_1' \vdash (\nu\,\tilde{m}_1)(P' \mid t?(x).(\nu\,s)(x\,s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\\ \approx^H \Delta_2 \vdash (\nu\,\tilde{m}_2)(Q' \mid t?(x).(\nu\,s)(x\,s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}))\end{array}$$

From the last typed pair we can derive that for $\Gamma;\emptyset;\Delta \vdash V_1 \triangleright U$:

$$\begin{array}{r}\Gamma;\emptyset;\,\Delta_1' \vdash (\nu\,\tilde{m}_1)(P' \mid t?(x).(\nu\,s)(x\,s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\\ \overset{t?\langle [\![?(U);\mathsf{end}]\!]^x\rangle}{\longmapsto} \Delta_1'' \vdash (\nu\,\tilde{m}_1)(P' \mid (\nu\,s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\end{array}$$

implies

$$\begin{array}{r}\Gamma;\emptyset;\,\Delta_2' \vdash (\nu\,\tilde{m}_2)(Q' \mid t?(x).(\nu\,s)(x\,s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}))\\ \overset{t?\langle [\![?(U);\mathsf{end}]\!]^x\rangle}{\longmapsto} \Delta_2'' \vdash (\nu\,\tilde{m}_2)(Q' \mid (\nu\,s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}))\end{array}$$

and $\Gamma;\emptyset;\Delta' \vdash V_2 \triangleright U$.
Transition (33) implies transition (34). It remains to show that for fresh t:

$$\begin{array}{r}\Gamma;\emptyset;\,\Delta_1' \vdash (\nu\,\tilde{m}_1)(P' \mid t?(x).(\nu\,s)((\!(?(U);\mathsf{end})\!)^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\\ \approx^H \Delta_2 \vdash (\nu\,\tilde{m}_2)(Q' \mid t?(x).(\nu\,s)((\!(?(U);\mathsf{end})\!)^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}))\end{array}$$

The freshness of t implies that

$$\begin{array}{r}\Gamma;\emptyset;\,\Delta_1' \vdash (\nu\,\tilde{m}_1)(P' \mid t?(x).(\nu\,s)((\!(?(U);\mathsf{end})\!)^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\\ \overset{t?\langle m'\rangle}{\longmapsto} \Delta_1'' \vdash (\nu\,\tilde{m}_1)(P' \mid (\nu\,s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\end{array}$$

and

$$\begin{array}{r}\Gamma;\emptyset;\,\Delta_2' \vdash (\nu\,\tilde{m}_2)(Q' \mid t?(x).(\nu\,s)((\!(?(U);\mathsf{end})\!)^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}))\\ \overset{t?\langle m'\rangle}{\longmapsto} \Delta_2'' \vdash (\nu\,\tilde{m}_2)(Q' \mid (\nu\,s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}))\end{array}$$

which coincides with the transitions for $\approx^H$.
- The rest of the cases are trivial.
The direction $\approx^C \subseteq \approx^H$ is very similar to the direction $\approx^H \subseteq \approx^C$: it requires a case analysis on the transition label ℓ. Again the non-trivial case is $\ell = (\nu\,\tilde{m}_1)n!\langle V_1\rangle$. □

The next lemma implies a process substitution lemma as a corollary. Given two processes that are bisimilar under trigerred substitution and characteristic process substitution, we can prove that they are bisimilar under every process substitution. This result is the key result for proving the soundness of the bisimulation.

Lemma B.2 (Linear Process Substitution). If

1. $\mathtt{fpv}(P_2) = \mathtt{fpv}(Q_2) = \{x\}$.
2. $\Gamma; x : U; \Delta_1''' \vdash P_2 \triangleright \diamond$ and $\Gamma; x : U; \Delta_2''' \vdash Q_2 \triangleright \diamond$.
3. $\Gamma;\Delta_1' \vdash (\nu\,\tilde{m}_1)(P_1 \mid P_2\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}) \approx^H \Delta_2' \vdash (\nu\,\tilde{m}_2)(Q_1 \mid Q_2\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\})$, for some fresh t.
4. $\Gamma;\Delta_1'' \vdash (\nu\,\tilde{m}_1)(P_1 \mid P_2\{(\!(U)\!)\mathsf{c}/x\}) \approx^H \Delta_2'' \vdash (\nu\,\tilde{m}_2)(Q_1 \mid Q_2\{(\!(U)\!)\mathsf{c}/x\})$, for some U.

then $\forall R$ such that $\mathtt{fv}(R) = \tilde{x}$

$$\Gamma;\Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{\lambda \tilde{x}.R}/x\}) \approx^H \Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{\lambda \tilde{x}.R}/x\})$$

Proof. We create a bisimulation closure:

$$\begin{aligned}
\Re = \{&\Gamma;\Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{\lambda \tilde{x}.R}/x\}), \Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{\lambda \tilde{x}.R}/x\}) \ \mid \\
&\quad \forall R \text{ such that } \mathtt{fv}(R) = \tilde{x}, \mathtt{fpv}(P_2) = \mathtt{fpv}(Q_2) = \{x\} \\
&\quad \Gamma; x : U; \Delta_1''' \vdash P_2 \triangleright \diamond, \Gamma; x : U; \Delta_2''' \vdash Q_2 \triangleright \diamond \\
&\quad \text{for fresh } t, \\
&\quad \Gamma;\Delta_1' \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/x\}) \approx^H \Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/x\}), \\
&\quad \Gamma;\Delta_1'' \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{[\![U]\!]_{\mathtt{c}}}/x\}) \approx^H \Delta_2'' \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{[\![U]\!]_{\mathtt{c}}}/x\}) \text{ for some } U \\
\}&
\end{aligned}$$

We show that $\Re$ is a bisimulation up-to $\beta-$transition (Lemma 4.3).
We do a case analysis on the transition:

$$\Gamma;\Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1\{{}^{\lambda \tilde{x}.R}/x\} \mid P_2\{{}^{\lambda \tilde{x}.R}/x\}) \xrightarrow{\ell_1} \Delta_1' \vdash P_1'$$

- Case: $P_2 \neq x\, \tilde{n}$ for some $\tilde{n}$.

$$\Gamma;\Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{\lambda \tilde{x}.R}/x\}) \overset{\ell_1}{\longmapsto} \Delta_1' \vdash (\nu\, \tilde{m_1'})(P_1 \mid P_2'\{{}^{\lambda \tilde{x}.R}/x\})$$

From the latter transition we obtain that

$$\begin{aligned}
&\Gamma;\emptyset;\, \Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/x\}) \\
&\overset{\ell_1}{\longmapsto} \Delta_1' \vdash P' \equiv (\nu\, \tilde{m}_1)(P_1' \mid P_2'\{{}^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/x\})
\end{aligned}$$

which implies

$$\begin{aligned}
&\Gamma;\emptyset;\, \Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/x\}) \\
&\overset{\ell_2}{\Longmapsto} \Delta_2' \vdash Q' \equiv (\nu\, \tilde{m}_2)(Q_1' \mid Q_2'\{{}^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/x\}) && (35)\\
&\qquad \Gamma;\Delta_1' \vdash P' \mid C_1 \approx^H \Delta_2' \vdash Q' \mid C_2 && (36)
\end{aligned}$$

Furthermore, we have:

$$\Gamma;\Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid P_2\{{}^{[\![U]\!]_{\mathtt{c}}}/x\}) \overset{\ell_1}{\longmapsto} \Delta_1' \vdash P'' \equiv (\nu\, \tilde{m_1'})(P_1' \mid P_2'\{{}^{[\![U]\!]_{\mathtt{c}}}/x\})$$

which implies

$$\begin{aligned}
&\Gamma;\emptyset;\, \Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{[\![U]\!]_{\mathtt{c}}}/x\}) \\
&\overset{\ell_2}{\Longmapsto} \Delta_2' \vdash Q'' \equiv (\nu\, \tilde{m}_2{}')(Q_1' \mid Q_2'\{{}^{[\![U]\!]_{\mathtt{c}}}/x\}) && (37)\\
&\qquad \Gamma;\Delta_1' \vdash P'' \mid C_1 \approx^H \Delta_2' \vdash Q'' \mid C_2 && (38)
\end{aligned}$$

From (35) and (37) we obtain that $\forall R$ with $\mathtt{fv}(R) = \tilde{x}$:

$$\Gamma;\Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_1 \mid Q_2\{{}^{\lambda\tilde{x}.R}/x\}) \overset{\ell_2}{\Longmapsto} \Delta'_2 \vdash (\nu\, \tilde{m}_2{}')(Q'_1 \mid Q'_2\{{}^{\lambda\tilde{x}.R}/x\})$$

The case concludes if we combine (36) and (38), to obtain that $\forall R$ with $\mathtt{fv}(R) = \tilde{x}$

$$\Gamma';\Delta''_1 \vdash (\nu\, \tilde{m}_1{}')(P'_1 \mid P'_2\{{}^{\lambda\tilde{x}.R}/x\}) \mid C_1 \ \Re\ \Delta''_2 \vdash (\nu\, \tilde{m}_2{}')(Q_1 \mid Q'_2\{{}^{\lambda\tilde{x}.R}/x\}) \mid C_2$$

- Case: $P_2 = x\,\tilde{n}$ for some $\tilde{n}$.
$\forall R$ with $\mathtt{fv}(R) = \tilde{x}$

$$\begin{aligned} &\Gamma;\emptyset;\, \Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid (x\,\tilde{n})\{{}^{\lambda\tilde{x}.R}/x\}) \\ \overset{\tau_\beta}{\longmapsto}\ &\Delta'_1 \vdash (\nu\, \tilde{m'_1})(P_1 \mid R\{{}^{\tilde{n}}/\tilde{x}\}) \end{aligned}$$

From the latter transition we get that:

$$\begin{aligned} &\Gamma;\emptyset;\, \Delta_1 \vdash (\nu\, \tilde{m}_1)(P_1 \mid x\,\tilde{n}\{{}^{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})}/x\}) \\ \overset{\tau_\beta}{\longmapsto}\overset{t?\langle\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})\rangle}{\longmapsto}\ &\Delta'_1 \vdash (\nu\, \tilde{m'_1})(P_1 \mid x\,\tilde{n}\{{}^{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})}/x\}) \end{aligned} \tag{39}$$

and t' a fresh name. From the freshness of t, the determinacy of the application transition and the fact that x is linear in Q_2 it has to be the case that:

$$\begin{aligned} &\Gamma;\emptyset;\Delta'_2 \vdash (\nu\, \tilde{m'_2})(Q_1 \mid Q_2\{{}^{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})}/x\}) \\ \Longmapsto\ &\qquad (\nu\, \tilde{m'_2})(Q''_1 \mid Q_3 \mid x\,\tilde{m}\{{}^{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})}/x\}) \\ \overset{\tau_\beta}{\longmapsto}\overset{t?\langle\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})\rangle}{\longmapsto}\ &\Delta''_2 \vdash (\nu\, \tilde{m'_2})(Q'_1 \mid x\,\tilde{m}\{{}^{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})}/x\}) \end{aligned}$$

and

$$\begin{aligned} &\Gamma;\emptyset;\, \Delta'_1 \vdash (\nu\, \tilde{m'_1})(P_1 \mid x\,\tilde{n}\{{}^{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})}/x\}) \\ \approx^H\ &\Delta'_2 \vdash (\nu\, \tilde{m}_2{}')(Q'_1 \mid x\,\tilde{m}\{{}^{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})}/x\}) \end{aligned} \tag{40}$$

From the latter transition we can conclude that $\forall R$ with $\mathtt{fv}(R) = \{x\}$:

$$\begin{aligned} &\Gamma;\emptyset;\Delta'_2 \vdash (\nu\, \tilde{m'_2})(Q_1 \mid Q_2\{{}^{\lambda\tilde{x}.R}/x\}) \\ \Longmapsto\ &\qquad (\nu\, \tilde{m'_2})(Q'_1 \mid x\,\tilde{m}\{{}^{\lambda\tilde{x}.R}/x\}) \\ \overset{\tau_\beta}{\longmapsto}\ &\Delta''_2 \vdash (\nu\, \tilde{m'_2})(Q'_1 \mid R\{{}^{\tilde{m}}/\tilde{x}\}) \end{aligned}$$

From the definition of $\mathcal{S}$ and (40), we also conclude that

$$\Gamma;\Delta'_1 \vdash (\nu\, \tilde{m'_1})(P_1 \mid R\{{}^{\tilde{n}}/\tilde{x}\}) \overset{\tau_\beta}{\longmapsto}\ \Re\ \overset{\tau_\beta}{\longleftarrow} \Delta'_2 \vdash (\nu\, \tilde{m}_2{}')(Q'_1 \mid R\{{}^{\tilde{m}}/\tilde{x}\})$$

□

We can generalise the result of the linear process substitution lemma to prove process substitution (Lemma 4.2). Intuitively, we can subsequently apply linear process substitution to achieve process substitution.

Lemma B.3 (Process Substitution). If

1. $\Gamma;\Delta_1' \vdash P\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \approx^H \Delta_2 \vdash Q\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}$ for some fresh t.
2. $\Gamma;\Delta_1'' \vdash P\{[\![U]\!]_{\mathsf{c}}/x\} \approx^H \Delta_2'' \vdash Q\{[\![U]\!]_{\mathsf{c}}/x\}$ for some U.

then $\forall R$ such that $\mathtt{fv}(R) = \tilde{x}$

$$\Gamma;\Delta_1 \vdash P\{\lambda\tilde{x}.R/x\} \approx^H \Delta_2 \vdash Q\{\lambda\tilde{x}.R/x\}$$

Proof. We define a closure $\Re$ using the normal form of P and Q

$$\begin{aligned}\Re = \{&\Gamma;\Delta_1 \vdash (\nu\,\tilde{m}_1)(P_1\{\lambda\tilde{x}.R/x\} \mid P_2\{\lambda\tilde{x}.R/x\}), \Delta_2 \vdash (\nu\,\tilde{m}_2)(Q_1\{\lambda\tilde{x}.R/x\} \mid Q_2\{\lambda\tilde{x}.R/x\}) \mid \\ &\forall R \text{ such that } \mathtt{fv}(R) = \tilde{x}, \\ &\text{for fresh } t, \begin{array}{l}\Gamma;\emptyset;\,\Delta_1' \vdash (\nu\,\tilde{m}_1)(P_1\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid P_2\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}) \\ \approx^H \Delta_2' \vdash (\nu\,\tilde{m}_2)(Q_1\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid Q_2\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\})\end{array} \\ &\text{for some } U, \begin{array}{l}\Gamma;\emptyset;\,\Delta_1'' \vdash (\nu\,\tilde{m}_1)(P_1\{[\![U]\!]_{\mathsf{c}}/x\} \mid P_2\{[\![U]\!]_{\mathsf{c}}/x\}) \\ \approx^H \Delta_2'' \vdash (\nu\,\tilde{m}_2)(Q_1\{[\![U]\!]_{\mathsf{c}}/x\} \mid Q_2\{[\![U]\!]_{\mathsf{c}}/x\})\end{array} \\ \}&\end{aligned}$$

We show that $\Re$ is a bisimulation up to β−transition (Lemma 4.3).
- Case: $P_2 \neq x\,\tilde{n}$ for some $\tilde{n}$.

$$\begin{aligned}&\Gamma;\emptyset;\,\Delta_1 \vdash (\nu\,\tilde{m}_1)(P_1\{\lambda\tilde{x}.R/x\} \mid P_2\{\lambda\tilde{x}.R/x\}) \\ &\overset{\ell_1}{\longmapsto} \Delta_1' \vdash (\nu\,\tilde{m}_1')(P_1\{\lambda\tilde{x}.R/x\} \mid P_2'\{\lambda\tilde{x}.R/x\})\end{aligned} \tag{41}$$

The case is similar to the first case of Lemma B.2.
- Case: $P_2 = x\,\tilde{n}$ for some $\tilde{n}$.

$$\begin{aligned}&\Gamma;\emptyset;\,\Delta_1 \vdash (\nu\,\tilde{m}_1)(P_1\{\lambda\tilde{x}.R/x\} \mid x\,\tilde{n}\{\lambda\tilde{x}.R/x\}) \\ &\overset{\tau_\beta}{\longmapsto} \Delta_1' \vdash (\nu\,\tilde{m}_1')(P_1\{\lambda\tilde{x}.R/x\} \mid R\{\tilde{n}/\tilde{x}\})\end{aligned}$$

From the latter transition we get that:

$$\begin{aligned}&\Gamma;\emptyset;\,\Delta_1 \vdash (\nu\,\tilde{m}_1)(P_1\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid x\,\tilde{n}\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}) \\ \overset{\tau_\beta}{\longmapsto}\overset{t?\langle\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})\rangle}{\longmapsto}\ &\Delta_1' \vdash (\nu\,\tilde{m}_1{}')(P_1\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid y\,\tilde{n}\{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})/y\})\end{aligned} \tag{42}$$

and t' a fresh name. From the freshness of t and the determinacy of the application transition it has to be the case that:

$$\begin{aligned}&\Gamma;\emptyset;\Delta_2' \vdash (\nu\,\tilde{m}_2{}')(Q_1\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid Q_2\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}) \\ \Longmapsto\ &(\nu\,\tilde{m}_2{}')(Q_1'\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid Q_2'\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid x\,\tilde{m}\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}) \\ \overset{\tau_\beta}{\longmapsto}\overset{t?\langle\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})\rangle}{\longmapsto}\ &\Delta_2'' \vdash (\nu\,\tilde{m}_2{}')((Q_1' \mid Q_2')\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid y\,\tilde{m}\{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})/y\})\end{aligned}$$

Let Q_3 such that

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta \vdash (\nu\,\tilde{m}_2{}')(Q_1 \mid Q_3)\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\}\{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})/y\} \\ &\Longmapsto \Delta' \vdash (\nu\,\tilde{m}_2{}')((Q_1' \mid Q_2')\{\lambda\tilde{x}.\,t?(y).(y\,\tilde{x})/x\} \mid y\,\tilde{m}\{\lambda\tilde{x}.\,t'?(y).(y\,\tilde{x})/y\})\end{aligned}$$

From Lemma B.2 we get that $\forall R$ with $\mathtt{fv}(R) = \tilde{x}$

$$\begin{array}{rl} \Gamma;\emptyset;\,\Delta_1''' & \vdash (\nu\, \tilde{m_1}')(P_1\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\} \mid y\, \tilde{n}\{^{\lambda \tilde{x}. R}/_y\}) \\ \approx^H \;\Delta' & \vdash (\nu\, \tilde{m_2}')((Q_1 \mid Q_3)\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\}\{^{\lambda \tilde{x}. R}/_y\}) \end{array}$$

From (41) we get that

$$\begin{array}{rl} \Gamma;\emptyset;\;\Delta' & \vdash (\nu\, \tilde{m_1}')((Q_1 \mid Q_3)\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\}\{^{\lambda \tilde{x}. R}/_y\}) \\ \Longrightarrow\stackrel{\tau_\beta}{\longmapsto} \Delta'' & \vdash (\nu\, \tilde{m_2}')((Q_1' \mid Q_2')\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\} \mid R\{^{\tilde{m}}/_{\tilde{x}}\}) \end{array}$$

and from the definition of $\Re$

$$\begin{array}{c} \Gamma;\emptyset;\,\Delta_1'' \vdash (\nu\, \tilde{m_1}')(P_1\{^{\lambda \tilde{x}. R}/_x\} \mid y\, \tilde{n}\{^{\lambda \tilde{x}. R}/_y\}) \\ \stackrel{\tau_\beta}{\longmapsto}\; \Re \;\stackrel{\tau_\beta}{\longleftarrow} \Delta_2'' \vdash (\nu\, \tilde{m_2}')((Q_1' \mid Q_2')\{^{\lambda \tilde{x}. R}/_x\} \mid y\, \tilde{m}\{^{\lambda \tilde{x}. R}/_y\}) \end{array}$$

as required. ◻

Lemma B.4. $\approx^H \subseteq \approx$

Proof. Let

$$\Gamma; \Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1$$

The proof is divided on cases on the label ℓ for the transition:

$$\Gamma; \Delta_1 \vdash P_1 \stackrel{\ell}{\longmapsto} \Delta_1' \vdash P_2 \tag{43}$$

- Case: $\ell \notin \{(\nu\, \tilde{m}_1)n!\langle \lambda \tilde{x}. P\rangle, (\nu\, \tilde{m_1}')n!\langle \tilde{m}_1\rangle, n?\langle \lambda \tilde{x}. P\rangle\}$
For the latter ℓ and transition in (43) we conclude that:

$$\Gamma; \Delta_2 \vdash Q_1 \stackrel{\ell}{\Longrightarrow} \Delta_2' \vdash Q_2$$

and

$$\Gamma; \Delta_1' \vdash P_2 \approx^H \Delta_2' \vdash Q_2$$

The above premise and conclusion coincides with defining cases for ℓ in $\approx$.
- Case: $\ell = n?\langle \lambda \tilde{x}. P\rangle$
Transition in (43) concludes:

$$\begin{array}{l} \Gamma; \Delta_1 \vdash P_1 \stackrel{n?\langle \lambda \tilde{x}. [\![U]\!]^{\tilde{x}}\rangle}{\longmapsto} \Delta_1' \vdash P_2\{^{\lambda \tilde{x}. [\![U]\!]^{\tilde{x}}}/_x\} \\ \Gamma; \Delta_1 \vdash P_1 \stackrel{n?\langle \lambda \tilde{x}.\, t?(y).(y\, \tilde{x})\rangle}{\longmapsto} \Delta_1'' \vdash P_2\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\} \end{array}$$

The last two transitions imply:

$$\begin{array}{l} \Gamma; \Delta_2 \vdash Q_1 \stackrel{n?\langle \lambda \tilde{x}. [\![U]\!]^{\tilde{x}}\rangle}{\Longrightarrow} \Delta_2' \vdash Q_2\{^{\lambda \tilde{x}. [\![U]\!]^{\tilde{x}}}/_x\} \\ \Gamma; \Delta_2 \vdash Q_1 \stackrel{n?\langle \lambda \tilde{x}.\, t?(y).(y\, \tilde{x})\rangle}{\Longrightarrow} \Delta_2'' \vdash Q_2\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\} \end{array}$$

and

$$\begin{array}{l} \Gamma; \Delta_1' \vdash P_2\{^{\lambda \tilde{x}. [\![U]\!]^{\tilde{x}}}/_x\} \approx^H \Delta_2' \vdash Q_2\{^{\lambda \tilde{x}. [\![U]\!]^{\tilde{x}}}/_x\} \\ \Gamma; \Delta_1'' \vdash P_2\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\} \approx^H \Delta_2'' \vdash Q_2\{^{\lambda \tilde{x}.\, t?(y).(y\, \tilde{x})}/_x\} \end{array}$$

To conlude from (4.2) that $\forall R$ with $\mathtt{fv}(R) = \tilde{x}$

$$\Gamma; \Delta'_1 \vdash P_2\{^{\lambda\tilde{x}.R}/x\} \approx^H \Delta'_2 \vdash Q_2\{^{\lambda\tilde{x}.R}/x\}$$

as required.
- Case: $\ell = (\nu\, \tilde{m}_1)n!\langle \lambda\tilde{x}.P\rangle$
From transition (43) we conclude:

$$\Gamma; \Delta_2 \vdash Q_1 \stackrel{(\nu\, \tilde{m}_2)n!\langle \lambda\tilde{x}.Q\rangle}{\Longrightarrow} \Delta'_2 \vdash Q_2$$

and for fresh t

$$\begin{array}{l} \Gamma; \emptyset; \Delta'_1 \vdash (\nu\, \tilde{m}_1)(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda\tilde{x}.P\rangle.\mathbf{0})) \\ \approx^H \Delta'_2 \vdash (\nu\, \tilde{m}_2)(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda\tilde{x}.Q\rangle.\mathbf{0})) \end{array}$$

From the previous case we can conclude that $\forall R$ with $\mathtt{fpv}(R) = \{x\}$:

$$\begin{array}{rl} & \Gamma; \emptyset; \Delta'_1 \vdash (\nu\, \tilde{m}_1)(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda\tilde{x}.P\rangle.\mathbf{0})) \\ \stackrel{t?\langle \lambda z.z?(x).R\rangle}{\longrightarrow} & (\nu\, \tilde{m}_1)(P_2 \mid (\nu\, s)(s?(x).R \mid \overline{s}!\langle \lambda\tilde{x}.P\rangle.\mathbf{0})) \\ \stackrel{\tau}{\longrightarrow} & \Delta''_1 \vdash (\nu\, \tilde{m}_1)(P_2 \mid R\{^{\lambda\tilde{x}.P}/x\}) \end{array}$$

and

$$\begin{array}{rl} & \Gamma; \emptyset; \Delta'_2 \vdash (\nu\, \tilde{m}_2)(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda\tilde{x}.Q\rangle.\mathbf{0})) \\ \stackrel{t?\langle \lambda z.z?(x).R\rangle}{\longrightarrow} & (\nu\, \tilde{m}_2)(Q_2 \mid (\nu\, s)(s?(x).R \mid \overline{s}!\langle \tilde{x}Q\rangle.\mathbf{0})) \\ \stackrel{\tau}{\longrightarrow} & \Delta''_2 \vdash (\nu\, \tilde{m}_2)(Q_2 \mid R\{^{\lambda\tilde{x}.Q}/x\}) \end{array}$$

and furthermore it is easy to see that $\forall R$ with $\mathtt{fpv}(R) = X$:

$$\Gamma; \Delta''_1 \vdash (\nu\, \tilde{m}_1)(P_2 \mid R\{^{\lambda\tilde{x}.P}/x\}) \approx^H \Delta_2 \vdash (\nu\, \tilde{m}_2)(Q_2 \mid R\{^{\lambda\tilde{x}.Q}/x\})$$

as required by the definition of $\approx$.
- Case: $\ell = (\nu\, \tilde{m}_1{}')n!\langle \tilde{m}_1\rangle$
The last case shares a similar argumentation with the previous case. ◻

Lemma B.5. $\approx \subseteq \cong$.

Proof. We prove that $\approx$ satisfies the defining properties of $\cong$. Let

$$\Gamma; \Delta_1 \vdash P_1 \approx \Delta_2 \vdash P_2$$

Reduction Closed:

$$\Gamma; \Delta_1 \vdash P_1 \longrightarrow \Delta'_1 \vdash P'_1$$

implies that $\exists P'_2$ such that

$$\begin{array}{r} \Gamma; \Delta_2 \vdash P_2 \Longrightarrow \Delta'_2 \vdash P'_2 \\ \Gamma; \Delta_1 \vdash P'_1 \approx \Delta'_2 \vdash P'_2 \end{array}$$

Same argument hold for the symmetric case, thus $\approx$ is reduction closed.

Barb Preservation:

$$\Gamma;\emptyset;\Delta_1 \vdash P_1 \triangleright \diamond \downarrow_n$$

implies that

$$\begin{aligned} &P \cong (\nu\, \tilde{m})(n!\langle V_1\rangle.P_3 \mid P_4) \\ &\overline{n} \notin \Delta_1 \end{aligned}$$

From the definition of $\approx$ we get that

$$\Gamma;\Delta_1 \vdash (\nu\, \tilde{m})(n!\langle V_1\rangle.P_3 \mid P_4) \stackrel{(\nu\, s_1)m!\langle V_1\rangle}{\longrightarrow} \Delta_1' \vdash (\nu\, \tilde{m'})(P_3 \mid P_4)$$

implies

$$\Gamma;\Delta_2 \vdash P_2 \stackrel{(\nu\, m_2)n!\langle V_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2'$$

From the last result we get that

$$\Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond \Downarrow_n$$

as required.

Congruence:

The congruence property requires that we check that $\approx$ is preserved under any context.
The most interesting context case is parallel composition.
We construct a congruence relation. Let

$$\begin{aligned} \mathcal{S} = \{&(\Gamma;\emptyset;\Delta_1\cdot\Delta_3 \vdash (\nu\, \tilde{n_1})(P_1 \mid R) \triangleright \diamond, \Gamma;\emptyset;\Delta_2\cdot\Delta_3 \vdash (\nu\, \tilde{n_2})(P_2 \mid R)) \mid \\ &\Gamma;\Delta_1 \vdash P_1 \approx \Delta_2 \vdash P_2, \forall \Gamma;\emptyset;\Delta_3 \vdash R \triangleright \diamond \\ &\} \end{aligned}$$

We need to show that the above congruence is a bisimulation. To show that $\mathcal{S}$ is a bisimulation we do a case analysis on the structure of the $\stackrel{\ell}{\longrightarrow}$ transition.

- Case:

$$\Gamma;\Delta_1\cdot\Delta_3 \vdash (\nu\, \tilde{n_1})(P_1 \mid R) \stackrel{\ell}{\longrightarrow} \Delta_1'\cdot\Delta_3 \vdash (\nu\, \tilde{n_1'})(P_1' \mid R)$$

The case is divided into three subcases:
Subcase i: $\ell \notin \{(\nu\, \tilde{m})n!\langle\lambda\tilde{x}.\,Q\rangle, (\nu\, m\tilde{m}_1)n!\langle\tilde{m}_1\rangle\}$
From the definition of typed transition we get:

$$\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longrightarrow} \Delta_1' \vdash P_1'$$

which implies that

$$\Gamma;\Delta_1 \vdash P_2 \stackrel{\ell}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{44}$$

$$\Gamma;\Delta_1' \vdash P_1' \approx \Delta_2'' \vdash P_2' \tag{45}$$

From transition in (44) we conclude that

$$\Gamma;\Delta_2\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2)(P_2 \mid R) \stackrel{\ell}{\Longrightarrow} \Delta_2'\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}')(P_2' \mid R)$$

Furthermore from (45) and the definition of $\mathcal{S}$ we conlude that

$$\Gamma;\Delta_1'\cdot\Delta_3 \vdash (\nu\, \tilde{n}_1{}')(P_1' \mid R)\ \mathcal{S}\ \Delta_2'\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}')(P_2' \mid R)$$

Subcase ii: $\ell = (\nu\, \tilde{m}_1)n!\langle\lambda\tilde{x}.\, Q_1\rangle$
From the definition of typed transition we get

$$\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\, \tilde{m}_1)n!\langle\lambda\tilde{x}.\, Q_1\rangle}{\longrightarrow} \Delta_1' \vdash P_1'$$

which implies that

$$\Gamma;\Delta_1 \vdash P_2 \stackrel{(\nu\, \tilde{m}_2)n!\langle\lambda\tilde{x}.\, Q_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{46}$$

$$\forall Q, \{x\} \in \mathtt{fpv}(Q)$$

$$\Gamma;\Delta_1'' \vdash (\nu\, \tilde{n}_1{}'')(P_1' \mid Q\{^{\lambda\tilde{x}.\, Q_1}/x\}) \ \approx\ \Delta_2'' \vdash (\nu\, \tilde{n}_2{}'')(P_2' \mid Q\{^{\lambda\tilde{x}.\, Q_2}/x\}) \tag{47}$$

From transition (46) conclude that

$$\Gamma;\Delta_2\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2)(P_2 \mid R) \stackrel{(\nu\, \tilde{m}_2)n!\langle\lambda\tilde{x}.\, Q_2\rangle}{\Longrightarrow} \Delta_2'\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}')(P_2' \mid R)$$

Furthermore from (47) we conlude that $\forall Q$ with $\{x\} = \mathtt{fpv}(Q)$

$$\Gamma;\Delta_1''\cdot\Delta_3 \vdash (\nu\, \tilde{n}_1{}'')(P_1' \mid Q\{^{(\tilde{x})Q_1}/x\} \mid R)\ \mathcal{S}\ \Delta_2''\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}'')(P_2' \mid Q\{^{\lambda\tilde{x}.\, Q_2}/x\} \mid R)$$

- Subcase iii: $\ell = (\nu\, m\tilde{m}_1)n!\langle\tilde{m}_1\rangle$
From the definition of typed transition we get that

$$\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\, m\tilde{m}_1)n!\langle\tilde{m}_1\rangle}{\longrightarrow} \Delta_1' \vdash P_1'$$

which implies that $\exists P_2', s_2$ such that

$$\Gamma;\Delta_1 \vdash P_2 \stackrel{(\nu\, m\tilde{m}_2)n!\langle\tilde{m}_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{48}$$

$$\forall Q, x = \mathtt{fn}(Q),$$

$$\Gamma;\Delta_1'' \vdash (\nu\, \tilde{n}_1)(P_1' \mid Q\{^{\tilde{m}_1}/\tilde{x}\}) \ \approx\ \Delta_2'' \vdash (\nu\, \tilde{n}_2)(P_2' \mid Q\{^{\tilde{m}_2}/\tilde{x}\}) \tag{49}$$

From transition (48) conclude that

$$\Gamma;\Delta_2\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}')(P_2 \mid R) \stackrel{(\nu\, m\tilde{m}_2)n!\langle\tilde{m}_2\rangle}{\Longrightarrow} \Delta_2'\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}''')(P_2' \mid R)$$

Furthermore from (49) we conlude that $\forall Q, x = \mathtt{fn}(Q)$

$$\Gamma;\Delta_1''\cdot\Delta_3 \vdash (\nu\, \tilde{n}_1{}'')(P_1' \mid Q\{^{\tilde{m}_1}/\tilde{x}\} \mid R)\ \mathcal{S}\ \Delta_2''\cdot\Delta_3 \vdash (\nu\, \tilde{n}_2{}'')(P_2' \mid Q\{^{\tilde{m}_2}/\tilde{x}\} \mid R)$$

- Case:

$$\Gamma;\Delta_1 \cdot \Delta_3 \vdash (\nu\, \tilde{m}_1)(P_1 \mid R) \xrightarrow{\ell} \Delta_1 \cdot \Delta_3' \vdash (\nu\, \tilde{m}_1{}')(P_1 \mid R')$$

This case is divided into three subcases:
Subcase i: $\ell \notin \{(\nu\, \tilde{m})n!\langle \lambda\tilde{x}.\, Q\rangle, (\nu\, m\tilde{m}_1)n!\langle \tilde{m}_1\rangle\}$
From the LTS we get that:

$$\Gamma;\Delta_3 \vdash R \xrightarrow{\ell} \Delta_3' \vdash R'$$

Which in turn implies

$$\Gamma;\Delta_2 \cdot \Delta_3 \vdash (\nu\, \tilde{m}_2)(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta_3' \vdash (\nu\, \tilde{m}_2{}')(P_2 \mid R')$$

From the definition of $\mathcal{S}$ we conclude that

$$\Gamma;\Delta_1 \cdot \Delta_3' \vdash (\nu\, \tilde{m}_1{}')(P_1 \mid R')\ \mathcal{S}\ \Delta_2 \cdot \Delta_3'' \vdash (\nu\, \tilde{m}_2{}')(P_2 \mid R')$$

as required.
Subcase ii: $\ell = (\nu\, \tilde{m}_1)n!\langle \lambda\tilde{x}.\, Q\rangle$
From the LTS we get that:

$$\Gamma;\Delta_3 \vdash R \xrightarrow{\ell} \Delta_3' \vdash R' \tag{50}$$

$$\forall R_1, \{x\} = \mathtt{fpv}(R_1),$$

$$\Gamma;\emptyset;\Delta_3'' \vdash (\nu\, \tilde{m}')(R' \mid R_1\{^{\lambda\tilde{x}.\, Q}/x\}) \triangleright \diamond \tag{51}$$

From (50) we get that

$$\Gamma;\Delta_2 \cdot \Delta_3 \vdash (\nu\, \tilde{m}_2{}')(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta_3' \vdash (\nu\, \tilde{m}_2)(P_2 \mid R')$$

Furthermore from (51) and the definition of $\mathcal{S}$ we conclude that $\forall R_1$ with $\{x\} \in \mathtt{fpv}(R_1)$

$$\Gamma;\Delta_1 \cdot \Delta_3'' \vdash (\nu\, \tilde{m}_1)(P_1 \mid (\nu\, \tilde{m}')(R' \mid R_1\{^{\lambda\tilde{x}.\, Q}/x\}))\ \mathcal{S}\ \Delta_2 \cup \Delta_3'' \vdash (\nu\, \tilde{m}_2)(P_2 \mid (\nu\, \tilde{m}')(R' \mid R_1\{^{\lambda\tilde{x}.\, Q}/x\}))$$

as required.
Subcase iii: $\ell = (\nu\, m\tilde{m})n!\langle \tilde{m}\rangle$
From the typed LTS we get that:

$$\Gamma;\Delta_3 \vdash R \xrightarrow{\ell} \Delta_3' \vdash R' \tag{52}$$

$$\forall Q, \tilde{x} = \mathtt{fn}(Q),$$

$$\Gamma;\emptyset;\Delta_3'' \vdash (\nu\, \tilde{m}')(R' \mid Q\{^{\tilde{m}}/\tilde{x}\}) \triangleright \diamond \tag{53}$$

From (52), we obtain that

$$\Gamma;\Delta_2 \cdot \Delta_3 \vdash (\nu\, \tilde{m}_2)(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta_3' \vdash (\nu\, \tilde{m}_2)(P_2 \mid R')$$

Furthermore from (53) and the definition of $\mathcal{S}$ we conclude that $\forall Q, \tilde{x} = \mathtt{fn}(Q)$

$$\Gamma;\Delta_1 \cdot \Delta_3'' \vdash (\nu\, \tilde{m}_1)(P_1 \mid (\nu\, \tilde{m})(R' \mid Q\{^{\tilde{m}'}/\tilde{x}\}))\ \mathcal{S}\ \Delta_2 \cdot \Delta_3'' \vdash (\nu\, \tilde{m}_2)(P_2 \mid (\nu\, \tilde{m}')(R' \mid Q\{^{\tilde{m}}/\tilde{x}\}))$$

as required.
- Case:

$$\Gamma;\Delta_1\cdot\Delta_3 \vdash (\nu\,\tilde{m}_1)(P_1 \mid R) \longrightarrow \Delta_1'\cdot\Delta_3' \vdash (\nu\,\tilde{m}_1{}')(P_1' \mid R')$$

This case is divided into three subcases:
Subcase i: $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\longrightarrow} \Delta_1' \vdash P_1'$ and $\ell \notin \{(\nu\,\tilde{m})n!\langle\lambda\tilde{x}.\,Q\rangle, (\nu\,m\tilde{m}_1)n!\langle\tilde{m}_1\rangle\}$ implies

$$\Gamma;\Delta_3 \vdash R \stackrel{\bar{\ell}}{\longrightarrow} \Delta_3 \vdash R' \tag{54}$$

$$\Gamma;\Delta_2 \vdash P_2 \stackrel{\hat{\ell}}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{55}$$

$$\Gamma;\Delta_1' \vdash P_1' \approx \Delta_2' \vdash P_2' \tag{56}$$

From (54) and (55) we get

$$\Gamma;\Delta_2\cdot\Delta_3 \vdash (\nu\,\tilde{m}_2)(P_2 \mid R) \Longrightarrow \Delta_2'\cdot\Delta_3' \vdash (\nu\,\tilde{m}_2{}')(P_2' \mid R')$$

From (56) and the definition of ($\mathcal{S}$) we get that

$$\Gamma;\Delta_1'\cdot\Delta_3' \vdash (\nu\,\tilde{m}_1{}')(P_1' \mid R')\ \mathcal{S}\ \Delta_2'\cdot\Delta_3 \vdash (\nu\,\tilde{m}_2{}')(P_2' \mid R')$$

as required.
Subcase ii: $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\,\tilde{m}_1)n!\langle\lambda\tilde{x}.\,Q_1\rangle}{\longrightarrow} \Delta_1' \vdash P_1'$ implies

$$\Gamma;\Delta_3 \vdash R \stackrel{n?\langle\lambda\tilde{x}.\,Q_1\rangle}{\longrightarrow} \Delta_3' \vdash R'\{\lambda\tilde{x}.\,Q_1/x\} \tag{57}$$

$$\Gamma;\Delta_1\cdot\Delta_3 \vdash (\nu\,\tilde{m}_1)(P_1 \mid R) \longrightarrow \Delta_1'\cdot\Delta_3' \vdash (\nu\,\tilde{m}_1{}'')(P_1' \mid R'\{\lambda\tilde{x}.\,Q_1/x\})$$

$$\Gamma;\Delta_2 \vdash P_2 \stackrel{(\nu\,\tilde{m}_2)n!\langle\lambda\tilde{x}.\,Q_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{58}$$

$$\forall Q, \{x\} = \mathtt{fpv}(Q),$$

$$\Gamma;\Delta_1'' \vdash (\nu\,\tilde{m}_1{}')(P_1' \mid Q\{\lambda\tilde{x}.\,Q_1/x\})\ \approx\ \Delta_2'' \vdash (\nu\,\tilde{m}_2{}')(P_2' \mid Q\{\lambda\tilde{x}.\,Q_2/x\}) \tag{59}$$

From (57) and the Substitution Lemma (Lemma 3.1) we obtain that

$$\Gamma;\Delta_3 \vdash R \stackrel{n?\langle\lambda\tilde{x}.\,Q_2\rangle}{\longrightarrow} \Delta_3'' \vdash R'\{\lambda\tilde{x}.\,Q_2/x\}$$

to combine with (58) and get

$$\Gamma;\Delta_2\cdot\Delta_3 \vdash (\nu\,\tilde{m}_2)(P_2 \mid R) \Longrightarrow \Delta_2'\cdot\Delta_3'' \vdash (\nu\,\tilde{m}_2{}'')(P_2' \mid R'\{\lambda\tilde{x}.\,Q_2/X\})$$

In result in (59), set Q as R' to obtain:

$$\Gamma;\Delta_1'' \vdash (\nu\,\tilde{m}_1{}')(P_1' \mid R'\{\lambda\tilde{x}.\,Q_1/x\})\ \mathcal{S}\ \Delta_2''(\nu\,\tilde{m}_2{}')(P_2' \mid R'\{\lambda\tilde{x}.\,Q_2/x\}) \vdash$$

Subcase iii: $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\,m\tilde{m}_1)n!\langle\tilde{m}_1\rangle}{\longrightarrow} \Delta_1' \vdash P_1'$

$$\Gamma;\Delta_3 \vdash R \stackrel{n?\langle\tilde{m}_1\rangle}{\longrightarrow} \Delta_3' \vdash R'\{\tilde{m}_1/\tilde{x}\} \tag{60}$$

$$\Gamma;\Delta_1\cup\Delta_3 \vdash (\nu\,\tilde{m}_1)(P_1 \mid R) \longrightarrow \Delta_1'\cup\Delta_3' \vdash (\nu\,\tilde{m}_1{}'')(P_1' \mid R'\{s_1/x\})$$

$$\Gamma;\Delta_2 \vdash P_2 \stackrel{(\nu\,m\tilde{m}_2)n!\langle\tilde{m}_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{61}$$

$$\forall Q, \{x\} = \mathtt{fpv}(Q),$$

$$\Gamma;\Delta_1'' \vdash (\nu\,\tilde{m}_1{}')(P_1' \mid Q\{\tilde{m}_1/\tilde{x}\})\ \approx\ \Delta_2'' \vdash (\nu\,\tilde{m}_2{}')(P_2' \mid Q\{\tilde{m}_2/\tilde{x}\}) \tag{62}$$

From (60) and the Substitution Lemma (Lemma 3.1) we get that

$$\Gamma;\Delta_3 \vdash R \overset{n?\langle \tilde{m_2}\rangle}{\longrightarrow} \Delta_3'' \vdash R'\{^{\tilde{m_2}}/\tilde{x}\}$$

to combine with (61) and get

$$\Gamma;\Delta_2 \cdot \Delta_3 \vdash (\nu\ \tilde{m_2})(P_2 \mid R) \Longrightarrow \Delta_2' \cdot \Delta_3'' \vdash (\nu\ \tilde{m_2}'')(P_2' \mid R'\{^{\tilde{m_2}}/\tilde{x}\})$$

Set Q as R' in result in (62) to obtain

$$\Gamma;\Delta_1'' \vdash (\nu\ \tilde{m_1}')(P_1' \mid R'\{^{\tilde{m_1}}/\tilde{x}\})\ \mathcal{S}\ \Delta_2'' \vdash (\nu\ \tilde{m_2}')(P_2' \mid R'\{^{\tilde{m_2}}/\tilde{x}\})$$

□

We prove the result $\cong \subseteq \approx^H$ following the technique developed in [18] and refined for session types in [27,26].

Definition B.1 (Definibility). *Let $\Gamma;\emptyset;\Delta_1 \vdash P \triangleright \diamond$. A visible action ℓ is* definable *whenever there exists (testing) process $\Gamma;\emptyset;\Delta_2 \vdash T\langle \ell, succ\rangle \triangleright \diamond$ with succ fresh name such that:*

- *If $\Gamma;\Delta_1 \vdash P \overset{\ell}{\longrightarrow} \Delta_1' \vdash P'$ and $\ell \in \{n \oplus \ell, n\&\ell, n?\langle \tilde{m}\rangle, n?\langle \lambda\tilde{x}.\,Q\rangle\}$ then:*

$$P \mid T\langle \ell, succ\rangle \longrightarrow P' \mid succ!\langle \overline{m}\rangle.\mathbf{0} \text{ and } \Gamma;\emptyset;\Delta_1' \cdot \Delta_2' \vdash P' \mid succ!\langle \overline{m}\rangle.\mathbf{0}$$

- *If $\Gamma;\Delta_1 \vdash P \overset{(\nu\ \tilde{m})n!\langle V\rangle}{\longrightarrow} \Delta_1' \vdash P'$, t fresh and $\tilde{m}' \subseteq \tilde{m}$ then:*

$$P \mid T\langle (\nu\ \tilde{m})n!\langle V\rangle, succ\rangle \longrightarrow (\nu\ \tilde{m})(P' \mid t?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle V\rangle.\mathbf{0}) \mid succ!\langle \overline{n}, \tilde{m}'\rangle.\mathbf{0})$$
$$\Gamma;\emptyset;\Delta_1' \cdot \Delta_2' \vdash (\nu\ \tilde{m})(P' \mid t?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle V\rangle.\mathbf{0}) \mid succ!\langle \overline{n}, \tilde{m}'\rangle.\mathbf{0}) \triangleright \diamond$$

- *Let $\ell \in \{n \oplus \ell, n\&\ell, n?\langle \tilde{m}\rangle, n?\langle (\tilde{x})Q\rangle\}$. If $P \mid T\langle \ell, succ\rangle \longrightarrow Q$ with $\Gamma;\emptyset;\Delta \vdash Q \triangleright \diamond \downarrow_{succ}$ then $\Gamma;\Delta_1 \vdash P \overset{\ell}{\Longrightarrow} \Delta_1' \vdash P'$ and $Q \equiv P' \mid succ!\langle \overline{n}\rangle.\mathbf{0}$.*
- *If $P \mid T\langle (\nu\ \tilde{m})n!\langle V\rangle, succ\rangle \longrightarrow Q$ with $\Gamma;\emptyset;\Delta \vdash Q \triangleright \diamond \downarrow_{succ}$ then $\Gamma;\Delta_1 \vdash P \overset{(\nu\ \tilde{m})n!\langle V\rangle}{\Longrightarrow} \Delta_1' \vdash P'$ and $Q \equiv (\nu\ \tilde{m})(P' \mid t?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle V\rangle.\mathbf{0}) \mid succ!\langle \overline{n}, \tilde{m}'\rangle.\mathbf{0})$ with t fresh and $\tilde{m}' \subseteq \tilde{m}$.*

We first show that every visible action ℓ is *definable*.

Lemma B.6 (Definibility). *Every action ℓ is definable.*

Proof. We define $T\langle \ell, \text{succ}\rangle$:

- $T\langle n?\langle V\rangle, \text{succ}\rangle = \overline{n}!\langle V\rangle.\text{succ}!\langle \overline{n}\rangle.\mathbf{0}$.
- $T\langle n\&l, \text{succ}\rangle = \overline{n} \triangleleft l.\text{succ}!\langle \overline{n}\rangle.\mathbf{0}$.
- $T\langle (\nu\ \tilde{m}')n!\langle \tilde{m}\rangle, \text{succ}\rangle = \overline{n}?(\tilde{x}).(t?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle \tilde{x}\rangle.\mathbf{0}) \mid \text{succ}!\langle \overline{n}, \tilde{m}''\rangle.\mathbf{0})$ with $\tilde{m}'' \subseteq \tilde{m}'$.

- $T\langle(\nu\,\tilde{m})n!\langle\lambda\tilde{x}.\,Q\rangle,\mathrm{succ}\rangle = \overline{n}?(y).(t?(x).(\nu\ s)(x\,s \mid \overline{s}!\langle\lambda\tilde{x}.\,(y\,\tilde{x})\rangle.\mathbf{0}) \mid \mathrm{succ}!\langle\overline{n},\tilde{m}'\rangle.\mathbf{0})$ with $\tilde{m}' \subseteq \tilde{m}$.
- $T\langle n \oplus l,\mathrm{succ}\rangle = \overline{n} \triangleright \{l : \mathrm{succ}!\langle\overline{n}\rangle.\mathbf{0}), l_i : (\nu\ a)(a?(y).\mathrm{succ}!\langle\overline{n}\rangle.\mathbf{0})\}_{i\in I}$.

Assuming a process

$$\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$$

it is straightforward to verify that $\forall\ell$, ℓ is definable. □

Lemma B.7 (Extrusion). If

$$\Gamma;\Delta_1' \vdash (\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \cong \Delta_2 \vdash (\nu\,\tilde{m}_2{}')(Q \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_2{}''\rangle.\mathbf{0})$$

then

$$\Gamma;\Delta_1 \vdash P \cong \Delta_2 \vdash Q$$

Proof. Let

$$\begin{aligned}\mathcal{S} = \{&\Gamma;\emptyset;\Delta_1 \vdash P \triangleright \diamond, \Gamma;\emptyset;\Delta_2 \vdash Q \triangleright \diamond \ \mid\\ &\Gamma;\Delta_1' \vdash (\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \cong \Delta_2 \vdash (\nu\,\tilde{m}_2{}')(Q \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_2{}''\rangle.\mathbf{0})\\ &\}\end{aligned}$$

We show that $\mathcal{S}$ is a congruence.
Reduction closed:
$P \longrightarrow P'$ implies $(\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \longrightarrow (\nu\,\tilde{m}_1{}')(P' \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0})$ implies from the freshness of succ $(\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \twoheadrightarrow (\nu\,\tilde{m}_1{}')(Q' \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_2{}''\rangle.\mathbf{0})$. which implies $Q \twoheadrightarrow Q'$ as required.
Barb Preserving:
Let $\Gamma;\emptyset;\Delta_1 \vdash P \downarrow_s$. We analyse two cases.
- Case: $s \neq n$.
$\Gamma;\emptyset;\Delta_1 \vdash P \downarrow_s$ implies

$$\Gamma;\emptyset;\Delta_1' \vdash (\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \downarrow_s$$

implies $\Gamma;\emptyset;\Delta_2' \vdash (\nu\,\tilde{m}_2{}')(Q \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_2{}''\rangle.\mathbf{0}) \Downarrow_s$ implies from the freshness of succ that $\Gamma;\emptyset;\Delta_2 \vdash Q \Downarrow_s$ as required.
- Case: $s = n$ and $\Gamma;\emptyset;\Delta_1 \vdash P \downarrow_n$
We compose with $\overline{\mathrm{succ}}?(x,\tilde{y}).T\langle\ell,\mathrm{succ}'\rangle$ with $\mathtt{subj}(\ell) = x$ to get

$$\Gamma;\emptyset;\Delta_1' \vdash (\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \mid \overline{\mathrm{succ}}?(x,\tilde{y}).T\langle\ell,\mathrm{succ}'\rangle$$

Which implies from the fact that $\Gamma;\emptyset;\Delta_1 \vdash P \downarrow_n$ that

$$(\nu\,\tilde{m}_1{}')(P \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0}) \mid \overline{\mathrm{succ}}?(x,\tilde{y}).T\langle\ell,\mathrm{succ}'\rangle \twoheadrightarrow (\nu\,\tilde{m}_1{}')(P' \mid \mathrm{succ}'!\langle\overline{n},\tilde{m}_1{}''\rangle.\mathbf{0})$$

and furthermore

$$(\nu\,\tilde{m}_2{}')(Q \mid \mathrm{succ}!\langle\overline{n},\tilde{m}_2{}''\rangle.\mathbf{0}) \mid \overline{\mathrm{succ}}?(x,\tilde{y}).T\langle\ell,\mathrm{succ}'\rangle \twoheadrightarrow (\nu\,\tilde{m}_2{}')(Q' \mid \mathrm{succ}'!\langle\overline{n},\tilde{m}_2{}''\rangle.\mathbf{0})$$

The last reduction implies that $\Gamma;\emptyset;\Delta_2 \vdash Q \Downarrow_n$ as required.
Congruence: The key case of congruence is parallel composition. We define relation C as

$$\begin{aligned}C = \{&\Gamma;\emptyset;\Delta_1 \cdot \Delta_3 \vdash P \mid R \triangleright \diamond, \Gamma;\emptyset;\Delta_2 \cdot \Delta_3 \vdash Q \mid R \triangleright \diamond \ \mid\\ &\forall R,\\ &\Gamma;\Delta_1' \vdash (\nu\, \tilde{m_1}')(P \mid \mathrm{succ}!\langle \overline{n}, \tilde{m_1}'' \rangle.\mathbf{0}) \cong \Delta_2' \vdash (\nu\, \tilde{m_2}')(Q \mid \mathrm{succ}!\langle \overline{n}, \tilde{m_2}'' \rangle.\mathbf{0})\}\end{aligned}$$

We show that C is a congruence.
We distinguish two cases:
- Case: $\overline{n}, \tilde{m_1}'', \tilde{m_2}'' \notin \mathtt{fn}(R)$
From the definition of C we can deduce that $\forall R$:

$$\Gamma;\Delta_1'' \vdash (\nu\, \tilde{m_1}')(P \mid \mathrm{succ}!\langle \overline{n}, \tilde{m_1}'' \rangle.\mathbf{0}) \mid R \cong \Delta_2'' \vdash (\nu\, \tilde{m_2}')(Q \mid \mathrm{succ}!\langle \overline{n}, \tilde{m_2}'' \rangle.\mathbf{0}) \mid R$$

The conclusion is then trivial.
- Case: $\tilde{s} = \{\overline{n}, \tilde{m_1}''\} \cap \{\overline{n}, \tilde{m_2}''\} \in \mathtt{fn}(R)$
From the definition of C we can deduce that $\forall R^{y_1}$ such that $R = R^{y_1}\{\tilde{s}/\tilde{y_1}\}$ and succ' fresh and $\{\tilde{y}\} = \{\tilde{y_1}\} \cup \{\tilde{y_2}\}$:

$$\begin{aligned}&\Gamma;\emptyset;\, \Delta_1'' \vdash (\nu\, \tilde{m_1}')(P \mid \mathrm{succ}!\langle \overline{n}, \tilde{m_1}'' \rangle.\mathbf{0}) \mid \overline{\mathrm{succ}}?(\tilde{y}).(R^{y_1} \mid \mathrm{succ}'!\langle \tilde{y_2} \rangle.\mathbf{0})\\ &\quad \cong \Delta_2'' \vdash (\nu\, \tilde{m_2}')(Q \mid \mathrm{succ}!\langle \overline{n}, \tilde{m_2}'' \rangle.\mathbf{0}) \mid \overline{\mathrm{succ}}?(\tilde{y}).(R^{y_1} \mid \mathrm{succ}'!\langle \tilde{y_2} \rangle.\mathbf{0})\end{aligned}$$

Applying reduction closeness to the above pair we get:

$$\Gamma;\Delta_1'' \vdash (\nu\, \tilde{m_1}')(P \mid R \mid \mathrm{succ}'!\langle \tilde{s_2} \rangle.\mathbf{0}) \cong \Delta_2'' \vdash (\nu\, \tilde{m_2}')(Q \mid R \mid \mathrm{succ}'!\langle \tilde{s_2} \rangle.\mathbf{0})$$

The conclusion then follows. □

Lemma B.8. $\cong \subseteq \approx^H$.

Proof. Let

$$\Gamma;\Delta_1 \vdash P_1 \cong \Delta_2 \vdash P_2$$

We distinguish two cases:
- Case:

$$\Gamma;\Delta_1 \vdash P_1 \xrightarrow{\tau} \Delta_1' \vdash P_1'$$

The result follows the reduction closeness property of $\cong$ since

$$\Gamma;\Delta_2 \vdash P_2 \stackrel{\tau}{\Longrightarrow} \Delta_2' \vdash P_2'$$

and

$$\Gamma;\Delta_1' \vdash P_1' \cong \Delta_2' \vdash P_2'$$

- Case:

$$\Gamma;\Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_1' \vdash P_1' \tag{63}$$

We choose test $T\langle \ell, \mathrm{succ}\rangle$ to get

$$\Gamma;\Delta_1 \cdot \Delta_3 \vdash P_1 \mid T\langle \ell, \mathrm{succ}\rangle \cong \Delta_2 \cdot \Delta_3 \vdash P_2 \mid T\langle \ell, \mathrm{succ}\rangle \tag{64}$$

From this point we distinguish three subcases:
Subcase i: $\ell \in \{n?\langle \tilde{m}\rangle, n?\langle \lambda \tilde{x}.\, Q\rangle, n \oplus l, n \& l\}$
By reducing (63), we obtain

$$\begin{array}{l} P_1 \mid T\langle \ell, \mathrm{succ}\rangle \longrightarrow P_1' \mid \mathrm{succ}!\langle \overline{n}\rangle.\mathbf{0} \\ \Gamma;\emptyset;\Delta_1' \cdot \Delta_3' \vdash P_1' \mid \mathrm{succ}!\langle \overline{n}\rangle.\mathbf{0} \downarrow_{\mathrm{succ}} \end{array}$$

implies from (64)

$$\Gamma;\emptyset;\Delta_2 \cdot \Delta_3 \vdash P_2 \mid T\langle \ell, \mathrm{succ}\rangle \Downarrow_{\mathrm{succ}}$$

implies from Lemma B.6,

$$\begin{array}{l} \Gamma;\Delta_2 \vdash P_2 \stackrel{\ell}{\Longrightarrow} \Delta_2' \vdash P_2' \\ P_2 \mid T\langle \ell, \mathrm{succ}\rangle \twoheadrightarrow P_2' \mid \mathrm{succ}!\langle \overline{n}\rangle.\mathbf{0} \end{array}$$

and

$$\Gamma;\Delta_1' \cdot \Delta_3' \vdash P_1' \mid \mathrm{succ}!\langle \overline{n}\rangle. \cong \Delta_2' \cdot \Delta_3' \vdash P_2' \mid \mathrm{succ}!\langle \overline{n}\rangle.\mathbf{0}$$

We then apply Lemma B.7 to get

$$\Gamma;\Delta_1' \vdash P_1' \cong \Delta_2' \vdash P_2'$$

as required.
Subcase ii: $\ell = (\nu\, \tilde{m}_1)n!\langle \lambda \tilde{x}.\, Q_1\rangle$
Note that $T\langle (\nu\, \tilde{m}_1)n!\langle (\tilde{x})Q_1\rangle, \mathrm{succ}\rangle = T\langle (\nu\, \tilde{m}_2)n!\langle \lambda \tilde{x}.\, Q_2\rangle, \mathrm{succ}\rangle$
Transition in (63) becomes

$$\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\, \tilde{m}_1)n!\langle \lambda \tilde{x}.\, Q_1\rangle}{\longrightarrow} \Delta_1' \vdash P_1' \tag{65}$$

If we use the test process $T\langle (\nu\, \tilde{m}_1)n!\langle (\tilde{x})Q_1\rangle, \mathrm{succ}\rangle$ we reduce to:

$$\begin{array}{l} P_1 \mid T\langle (\nu\, \tilde{m}_1)n!\langle \lambda \tilde{x}.\, Q_1\rangle, \mathrm{succ}\rangle \longrightarrow (\nu\, m_1)(P_1' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda \tilde{x}.\, Q_1\rangle.\mathbf{0})) \mid \mathrm{succ}!\langle \overline{n}, \tilde{m}_1{}'\rangle.\mathbf{0} \\ \Gamma;\emptyset;\Delta_1' \cdot \Delta_3' \vdash (\nu\, m_1)(P_1' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda \tilde{x}.\, Q_1\rangle.\mathbf{0})) \mid \mathrm{succ}!\langle \overline{n}, \tilde{m}_1{}'\rangle.\mathbf{0} \downarrow_{\mathrm{succ}} \end{array}$$

implies from (64)

$$\Gamma;\emptyset;\Delta_2 \cdot \Delta_3 \vdash P_2 \mid T\langle (\nu\, \tilde{m}_2)n!\langle \lambda \tilde{x}.\, Q_2\rangle, \mathrm{succ}\rangle \Downarrow_{\mathrm{succ}}$$

implies from Lemma B.6

$$\begin{array}{l} \Gamma;\Delta_2 \vdash P_2 \stackrel{(\nu\, \tilde{m}_2)n!\langle \lambda \tilde{x}.\, Q_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \\ P_2 \mid T\langle \ell, \mathrm{succ}\rangle \twoheadrightarrow (\nu\, m_2)(P_2' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda \tilde{x}.\, Q_2\rangle.\mathbf{0})) \mid \mathrm{succ}!\langle \overline{n}, \tilde{m}_2{}'\rangle.\mathbf{0} \end{array} \tag{66}$$

and

$$\begin{array}{l}\Gamma;\emptyset;\Delta_1'\cdot\Delta_3'\vdash (\nu\, m_1)(P_1' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle\lambda\tilde{x}.\, Q_1\rangle.\mathbf{0})) \mid \mathrm{succ}!\langle\overline{n},\tilde{m_1}'\rangle.\mathbf{0}\\ \quad\cong \Delta_2'\cdot\Delta_3'\vdash (\nu\, m_2)(P_2' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle\lambda\tilde{x}.\, Q_2\rangle.\mathbf{0})) \mid \mathrm{succ}!\langle\overline{n},\tilde{m_2}'\rangle.\mathbf{0}\end{array}$$

We then apply Lemma B.7 to get

$$\begin{array}{l}\Gamma;\emptyset;\Delta_1'\vdash (\nu\, m_1)(P_1' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle\lambda\tilde{x}.\, Q_1\rangle.\mathbf{0}))\\ \quad\cong \Delta_2'\vdash (\nu\, m_2)(P_2' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle\lambda\tilde{x}.\, Q_2\rangle.\mathbf{0}))\end{array}$$

as required.
-Case: $\ell = (\nu\,\tilde{s})n!\langle\tilde{m}\rangle$
Follows similar arguments as the previous case. □

Theorem B.1 (Concidence).

1. $\approx\, = \,\approx^H$.
2. $\approx\, = \,\cong$.

Proof. Lemma B.1 proves $\approx^H = \approx^C$. Lemma B.8 proves $\cong\, \subseteq\, \approx^H$. Lemma B.4 proves $\approx^H \subseteq\, \approx$. Lemma B.5 proves $\approx\, \subseteq\, \cong$.
From the above results, we conclude $\cong\, \subseteq\, \approx^H = \approx^C \subseteq\, \approx\, \subseteq\, \cong$. □

B.2 τ-inertness

We prove Part 1 of Proposition 4.3.

Proposition B.1 (τ-inertness). Let balanced $\mathsf{HO}\pi$ process $\Gamma;\emptyset;\Delta\vdash P\triangleright\diamond$. $\Gamma;\Delta\vdash P \stackrel{\tau_{\mathsf{d}}}{\longmapsto} \Delta'\vdash P'$ implies $\Gamma;\Delta\vdash P \approx^H \Delta'\vdash P'$.

Proof. The proof is done by induction on the structure of $\stackrel{\tau}{\longrightarrow}$ which coincides the reduction $\longrightarrow$.
Basic step:
- Case: $P = (\lambda x.\, P)\, n$:

$$\Gamma;\Delta\vdash(\lambda x.\, P)\, n \stackrel{\tau_\beta}{\longmapsto} \Delta'\vdash P\{n/x\}$$

Bisimulation requirements hold since, there is no other transition to observe than $\stackrel{\tau_\beta}{\longmapsto}$.
- Case: $P = s!\langle V\rangle.P_1 \mid \overline{s}?(x).P_2$:

$$\Gamma;\Delta\vdash s!\langle V\rangle.P_1 \mid \overline{s}?(x).P_2 \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \Delta'\vdash P_1 \mid P_2$$

The proof follows from the fact that we can only observe a τ action on typed process $\Gamma;\emptyset;\Delta\vdash P\triangleright\diamond$. Actions $s!\langle V\rangle$ and $\overline{s}?\langle V\rangle$ are forbiden by the LTS for typed environments. It is easy to conclude then that $\Gamma;\Delta\vdash P\approx^H \Delta'\vdash P'$.
- Case: $P = s \triangleleft l.P_1 \mid \overline{s}\triangleright\{l_i : P_i\}_{i\in I}$
Similar arguments as the previous case.
Induction hypothesis:
If $P_1 \longrightarrow P_2$ then $\Gamma_1;\Delta_1\vdash P_1 \approx^H \Delta_2\vdash P_2$.

Induction Step:
- Case: $P = (\nu\, s)P_1$

$$\Gamma;\Delta \vdash (\nu\, s)P_1 \overset{\tau_{\mathsf{s}}}{\longmapsto} \Delta' \vdash (\nu\, s)P_2$$

From the induction hypothesis and the fact that bisimulation is a congruence we get that $\Gamma;\Delta \vdash P \approx^H \Delta' \vdash P'$.
- Case: $P = P_1 \mid P_3$

$$\Gamma;\Delta \vdash P_1 \mid P_3 \overset{\tau_{\mathsf{s}}}{\longmapsto} \Delta' \vdash P_2 \mid P_3$$

From the induction hypothesis and the fact that bisimulation is a congruence we get that $\Gamma;\Delta \vdash P \approx^H \Delta' \vdash P'$.
- Case: $P \equiv P_1$

From the induction hypothesis and the fact that bisimulation is a congruence and structural congruence preserves $\approx^H$ we get that $\Gamma;\Delta \vdash P \approx^H \Delta' \vdash P'$.

□

C Expressiveness Results

C.1 Properties for $\langle [\![\cdot]\!]^1_f, (\!(\cdot)\!)^1, \{\!\{\cdot\}\!\}^1 \rangle : \mathsf{HO}\pi \to \mathsf{HO}$

We repeat the statement of Proposition 6.2, as in Page 29:

Proposition C.1 (Type Preservation, $\mathsf{HO}\pi$ into HO). *Let P be a $\mathsf{HO}\pi$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then $(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \triangleright \diamond$.*

Proof. By induction on the inference of $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$.

1. Case $P = k!\langle n\rangle.P'$. There are two sub-cases. In the first sub-case $n = k'$ (output of a linear channel). Then we have the following typing in the source language:

$$\frac{\Gamma;\emptyset;\Delta\cdot k : S \vdash P' \triangleright \diamond \qquad \Gamma;\emptyset;\{k' : S_1\} \vdash k' \triangleright S_1}{\Gamma;\emptyset;\Delta\cdot k' : S_1 \cdot k :!\langle S_1\rangle;S \vdash k!\langle k'\rangle.P' \triangleright \diamond}$$

Thus, by IH we have

$$(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta)\!)^1\cdot k : (\!(S)\!)^1 \vdash [\![P']\!]^1 \triangleright \diamond$$

Let us write U_1 to stand for $?((\!(S_1)\!)^1{\multimap}\diamond);\mathsf{end}{\multimap}\diamond$. The corresponding typing in the target language is as follows:

$$\frac{\dfrac{\dfrac{\dfrac{(\!(\Gamma)\!)^1;\{x : (\!(S_1)\!)^1{\multimap}\diamond\};\emptyset \vdash x \triangleright (\!(S_1)\!)^1{\multimap}\diamond \qquad (\!(\Gamma)\!)^1;\emptyset;\{k' : (\!(S_1)\!)^1\} \vdash k' \triangleright (\!(S_1)\!)^1}{(\!(\Gamma)\!)^1;\{x : (\!(S_1)\!)^1{\multimap}\diamond\};k' : (\!(S_1)\!)^1 \vdash x\,k' \triangleright \diamond}}{(\!(\Gamma)\!)^1;\{x : (\!(S_1)\!)^1{\multimap}\diamond\};k' : (\!(S_1)\!)^1\cdot z : \mathsf{end} \vdash x\,k' \triangleright \diamond}}{(\!(\Gamma)\!)^1;\emptyset;k' : (\!(S_1)\!)^1\cdot z :?((\!(S_1)\!)^1{\multimap}\diamond);\mathsf{end} \vdash z?(x).(x\,k') \triangleright \diamond}}{(\!(\Gamma)\!)^1;\emptyset;k' : (\!(S_1)\!)^1 \vdash \lambda z.z?(x).(x\,k') \triangleright U_1} \quad (67)$$

$$\frac{(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta)\!)^1\cdot k : (\!(S)\!)^1 \vdash [\![P']\!]^1 \triangleright \diamond \qquad (\!(\Gamma)\!)^1;\emptyset;k' : (\!(S_1)\!)^1 \vdash \lambda z.z?(x).(x\,k') \triangleright U_1 \ (67)}{(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta)\!)^1\cdot k' : (\!(S_1)\!)^1\cdot k :!\langle U_1\rangle;(\!(S)\!)^1 \vdash k!\langle \lambda z.z?(x).(x\,k')\rangle.[\![P']\!]^1 \triangleright \diamond}$$

In the second sub-case, we have $n = a$ (output of a shared name). Then we have the following typing in the source language:

$$\frac{\Gamma \cdot a : \langle S_1 \rangle; \emptyset; \Delta \cdot k : S \vdash P' \triangleright \diamond \quad \Gamma \cdot a : \langle S_1 \rangle; \emptyset; \emptyset \vdash a \triangleright S_1}{\Gamma \cdot a : \langle S_1 \rangle; \emptyset; \Delta \cdot k : !\langle \langle S_1 \rangle \rangle; S \vdash k!\langle a \rangle.P' \triangleright \diamond}$$

The typing in the target language is derived similarly as in the first sub-case.

2. Case $P = k?(x).Q$. We have two sub-cases, depending on the type of x. In the first case, x stands for a linear channel. Then we have the following typing in the source language:

$$\frac{\Gamma; \emptyset; \Delta \cdot k : S \cdot x : S_1 \vdash Q \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot k : ?(S_1); S \vdash k?(x).Q \triangleright \diamond}$$

Thus, by IH we have

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \cdot x : \langle\!\langle S_1 \rangle\!\rangle^1 \vdash [\![Q]\!]^1 \triangleright \diamond$$

Let us write U_1 to stand for $?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end} {\multimap} \diamond$. The corresponding typing in the target language is as follows:

$$\frac{\langle\!\langle \Gamma \rangle\!\rangle^1; \{X : U_1\}; \emptyset \vdash X \triangleright U_1 \qquad \langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \cdot s : ?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end} \ \vdash s \triangleright ?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end}}{\langle\!\langle \Gamma \rangle\!\rangle^1; \{X : U_1\}; \cdot s : ?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end} \ \vdash x\, s \triangleright \diamond} \tag{68}$$

$$\frac{\dfrac{\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \emptyset \vdash \mathbf{0} \triangleright \diamond}{\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \overline{s} : \mathsf{end} \vdash \mathbf{0} \triangleright \diamond} \qquad \dfrac{\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 x : \langle\!\langle S_1 \rangle\!\rangle^1 \vdash [\![Q]\!]^1 \triangleright \diamond}{\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \vdash \lambda x. [\![Q]\!]^1 \triangleright \langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \cdot \overline{s} : !\langle \langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond \rangle; \mathsf{end} \vdash \overline{s}!\langle \lambda x. [\![Q]\!]^1 \rangle.\mathbf{0} \triangleright \diamond} \tag{69}$$

$$\frac{\begin{array}{c} \langle\!\langle \Gamma \rangle\!\rangle^1; \{X : U_1\}; \cdot s : ?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end} \ \vdash x\, s \triangleright \diamond \quad (68) \\ \langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \cdot \overline{s} : !\langle \langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond \rangle; \mathsf{end} \vdash \overline{s}!\langle \lambda x. [\![Q]\!]^1 \rangle.\mathbf{0} \triangleright \diamond \ (69) \end{array}}{\langle\!\langle \Gamma \rangle\!\rangle^1; \{X : U_1\}; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \cdot s : ?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end} \cdot \overline{s} : !\langle \langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond \rangle; \mathsf{end} \vdash x\, s \mid \overline{s}!\langle \lambda x. [\![Q]\!]^1 \rangle.\mathbf{0} \triangleright \diamond} \tag{70}$$

$$\frac{\dfrac{\langle\!\langle \Gamma \rangle\!\rangle^1; \{X : U_1\}; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \cdot s : ?(\langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond); \mathsf{end} \cdot \overline{s} : !\langle \langle\!\langle S_1 \rangle\!\rangle^1 {\multimap} \diamond \rangle; \mathsf{end} \vdash x\, s \mid \overline{s}!\langle \lambda x. [\![Q]\!]^1 \rangle.\mathbf{0} \triangleright \diamond \quad (70)}{\langle\!\langle \Gamma \rangle\!\rangle^1; \{X : U_1\}; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : \langle\!\langle S \rangle\!\rangle^1 \vdash (\nu\ s)(x\, s \mid \overline{s}!\langle \lambda x. [\![Q]\!]^1 \rangle.\mathbf{0}) \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^1 \cdot k : ?(U_1); \langle\!\langle S \rangle\!\rangle^1 \vdash k?(x).(\nu\ s)(x\, s \mid \overline{s}!\langle \lambda x. [\![Q]\!]^1 \rangle.\mathbf{0}) \triangleright \diamond}$$

In the second sub-case, x stands for a shared name. Then we have the following typing in the source language:

$$\frac{\Gamma \cdot x : \langle S_1 \rangle; \emptyset; \Delta \cdot k : S \vdash Q \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot k : ?(\langle S_1 \rangle); S \vdash k?(x).Q \triangleright \diamond}$$

The typing in the target language is derived similarly as in the first sub-case.

3. Case $P_0 = X$. Then we have the following typing in the source language:

$$\Gamma \cdot X : \Delta; \emptyset; \emptyset \vdash X \triangleright \diamond$$

Then the typing of $\llbracket X\rrbracket^1_f$ is as follows, assuming $f(X)=\tilde{n}$ and $\tilde{x}=(\!|\tilde{n}|\!)$. Also, we write $\Delta_{\tilde{n}}$ and $\Delta_{\tilde{x}}$ to stand for $n_1:S_1,\ldots,n_m:S_m$ and $x_1:S_1,\ldots,x_m:S_m$, respectively. Below, we assume that $\Gamma=\Gamma'\cdot X:\tilde{T}\rightarrow\diamond$, where

$$\tilde{T}=(\tilde{S},S^*) \qquad S^*=?(A);\mathsf{end} \qquad A=\mu\mathsf{t}.(\tilde{S},?(\mathsf{t});\mathsf{end})$$

$$\dfrac{\dfrac{}{\Gamma;\emptyset;\emptyset\vdash z_X\triangleright\tilde{T}\rightarrow\diamond}\quad\begin{array}{c}\Gamma;\emptyset;\{n_i:S_i\}\vdash n_i\triangleright S_i\\ \Gamma;\emptyset;\{s:S^*\}\vdash s\triangleright S^*\end{array}}{\Gamma;\emptyset;\Delta_{\tilde{n}},s:?(\tilde{T}\rightarrow\diamond);\mathsf{end}\vdash z_X\,(\tilde{n},s)\triangleright\diamond}\tag{71}$$

$$\dfrac{\dfrac{\Gamma;\emptyset;\emptyset\vdash\mathbf{0}\triangleright\diamond}{\Gamma;\emptyset;\overline{s}:\mathsf{end}\vdash\mathbf{0}\triangleright\diamond}\quad\dfrac{\dfrac{\begin{array}{c}\Gamma;\emptyset;\{x_i:S_i\}\vdash x_i\triangleright S_i\\ \Gamma;\emptyset;\{z:S^*\}\vdash z\triangleright S^*\\ \Gamma;\emptyset;\emptyset\vdash z_X\triangleright\tilde{T}\rightarrow\diamond\end{array}}{\Gamma;\emptyset;\Delta_{\tilde{x}},z:S^*\vdash z_X\,(\tilde{x},z)\triangleright\diamond}}{\Gamma;\emptyset;\emptyset\vdash\lambda(\tilde{x},z).\ z_X\,(\tilde{x},z)\triangleright\tilde{T}\rightarrow\diamond}}{\Gamma;\emptyset;\overline{s}:!\langle\tilde{T}\rightarrow\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},z).\ z_X\,(\tilde{x},z)\rangle.\mathbf{0}\triangleright\diamond}\tag{72}$$

$$\dfrac{\dfrac{\begin{array}{c}\Gamma;\emptyset;\Delta_{\tilde{n}},s:?(\tilde{T}\rightarrow\diamond);\mathsf{end}\vdash z_X\,(\tilde{n},s)\triangleright\diamond\quad(71)\\ \Gamma;\emptyset;\overline{s}:!\langle\tilde{T}\rightarrow\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},z).\ z_X\,(\tilde{x},z)\rangle.\mathbf{0}\triangleright\diamond\ (72)\end{array}}{\Gamma;\emptyset;\Delta_{\tilde{n}},s:?(\tilde{T}\rightarrow\diamond);\mathsf{end},\overline{s}:!\langle\tilde{T}\rightarrow\diamond\rangle;\mathsf{end}\vdash z_X\,(\tilde{n},s)\mid\overline{s}!\langle\lambda(\tilde{x},z).\ x(\tilde{x},z)\rangle.\mathbf{0}\triangleright\diamond}}{\Gamma;\emptyset;\Delta_{\tilde{n}}\vdash(\nu\ s)(z_X\,(\tilde{n},s)\mid\overline{s}!\langle\lambda(\tilde{x},z).\ z_X\,(\tilde{x},z)\rangle.\mathbf{0})\triangleright\diamond}$$

4. Case $P_0=\mu X.P$. Then we have the following typing in the source language:

$$\dfrac{\Gamma\cdot X:\Delta;\emptyset;\Delta\vdash P\triangleright\diamond}{\Gamma;\emptyset;\Delta\vdash\mu X.P\triangleright\diamond}$$

Then we have the following typing in the target language —we write R to stand for $\llbracket P\rrbracket^1_{f,\{X\rightarrow\tilde{n}\}}$ and $\tilde{x}$ to stand for $(\!|\mathsf{ofn}(P)|\!)$.

$$\dfrac{\dfrac{(\!(\Gamma)\!)^1\cdot z_X:\tilde{T}\rightarrow\diamond;\emptyset;(\!(\Delta_{\tilde{n}})\!)^1\vdash R\triangleright\diamond}{(\!(\Gamma)\!)^1\cdot z_X:\tilde{T}\rightarrow\diamond;\emptyset;(\!(\Delta_{\tilde{n}})\!)^1,s:\mathsf{end}\vdash R\triangleright\diamond}}{(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta_{\tilde{n}})\!)^1,s:?(\tilde{T}\rightarrow\diamond);\mathsf{end}\vdash s?(z_X).R\triangleright\diamond}\tag{73}$$

$$\dfrac{\dfrac{(\!(\Gamma)\!)^1;\emptyset;\emptyset\vdash\mathbf{0}\triangleright\diamond}{(\!(\Gamma)\!)^1;\emptyset;\overline{s}:\mathsf{end}\vdash\mathbf{0}\triangleright\diamond}\quad\dfrac{\dfrac{\dfrac{(\!(\Gamma)\!)^1\cdot z_X:\tilde{T}\rightarrow\diamond;\emptyset;(\!(\Delta_{\tilde{x}})\!)^1\vdash\llbracket R\rrbracket_{\emptyset}\triangleright\diamond}{(\!(\Gamma)\!)^1\cdot z_X:\tilde{T}\rightarrow\diamond;\emptyset;(\!(\Delta_{\tilde{x}})\!)^1,y:\mathsf{end}\vdash\llbracket R\rrbracket_{\emptyset}\triangleright\diamond}}{(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta_{\tilde{x}})\!)^1,y:?(A);\mathsf{end}\vdash y?(z_X).\llbracket R\rrbracket_{\emptyset}\triangleright\diamond}}{(\!(\Gamma)\!)^1;\emptyset;\emptyset\vdash\lambda(\tilde{x},y).\ y?(z_X).\llbracket R\rrbracket_{\emptyset}\triangleright\tilde{T}\rightarrow\diamond}}{(\!(\Gamma)\!)^1;\emptyset;\overline{s}:!\langle\tilde{T}\rightarrow\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},y).\ y?(z_X).\llbracket R\rrbracket_{\emptyset}\rangle.\mathbf{0}\triangleright\diamond}\tag{74}$$

$$\dfrac{\dfrac{\begin{array}{c}(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta_{\tilde{n}})\!)^1,s:?(\tilde{T}\rightarrow\diamond);\mathsf{end}\vdash s?(z_X).R\triangleright\diamond\quad(73)\\ (\!(\Gamma)\!)^1;\emptyset;\overline{s}:!\langle\tilde{T}\rightarrow\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},y).\ y?(z_X).\llbracket R\rrbracket_{\emptyset}\rangle.\mathbf{0}\triangleright\diamond\ (74)\end{array}}{(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta_{\tilde{n}})\!)^1,s:?(\tilde{T}\rightarrow\diamond);\mathsf{end},\overline{s}:!\langle\tilde{T}\rightarrow\diamond\rangle;\mathsf{end}\vdash s?(z_X).R\mid\overline{s}!\langle\lambda(\tilde{x},y).\ y?(z_X).\llbracket R\rrbracket_{\emptyset}\rangle.\mathbf{0}\triangleright\diamond}}{(\!(\Gamma)\!)^1;\emptyset;(\!(\Delta_{\tilde{n}})\!)^1\vdash(\nu\ s)(s?(z_X).R\mid\overline{s}!\langle\lambda(\tilde{x},y).\ y?(z_X).\llbracket R\rrbracket_{\emptyset}\rangle.\mathbf{0})\triangleright\diamond}$$

□

We repeat the statement of Proposition 6.4, as in Page 30:

Proposition C.2 (Operational Correspondence, $\mathsf{HO}\pi$ into HO). Let P be a $\mathsf{HO}\pi$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then:

1. Suppose $\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'$. Then we have:
   a) If $\ell_1 \in \{(\nu\,\tilde{m})n!\langle m\rangle, (\nu\,\tilde{m})n!\langle \lambda x.\,Q\rangle, s\oplus l, s\&l\}$ then $\exists \ell_2$ s.t.
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash [\![P']\!]^1_f$ and $\ell_2 = \{\!\{\ell_1\}\!\}^1$.
   b) If $\ell_1 = n?\langle \lambda y.\,Q\rangle$ and $P' = P_0\{\lambda y.\,Q/x\}$ then $\exists \ell_2$ s.t.
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash [\![P_0]\!]^1_f\{\lambda y.\,[\![Q]\!]^1_\emptyset/x\}$ and $\ell_2 = \{\!\{\ell_1\}\!\}^1$.
   c) If $\ell_1 = n?\langle m\rangle$ and $P' = P_0\{m/x\}$ then $\exists \ell_2, R$ s.t.
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash R$, with $\ell_2 = \{\!\{\ell_1\}\!\}^1$,
   and $(\!(\Gamma)\!)^1;(\!(\Delta')\!)^1 \vdash R \xrightarrow{\tau_\beta}\xrightarrow{\tau_{\mathsf{s}}}\xrightarrow{\tau_\beta} (\!(\Delta')\!)^1 \vdash [\![P_0]\!]^1_f\{m/x\}$.
   d) If $\ell_1 = \tau$ and $P' \equiv (\nu\,\tilde{m})(P_1 \mid P_2\{m/x\})$ then $\exists R$ s.t.
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\tau} (\!(\Delta)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R)$, and
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R) \xrightarrow{\tau_\beta}\xrightarrow{\tau_{\mathsf{s}}}\xrightarrow{\tau_\beta} (\!(\Delta)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{m/x\})$.
   e) If $\ell_1 = \tau$ and $P' \equiv (\nu\,\tilde{m})(P_1 \mid P_2\{\lambda y.\,Q/x\})$ then
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\tau} (\!(\Delta_1)\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{\lambda y.\,[\![Q]\!]^1_\emptyset/x\})$.
   f) If $\ell_1 = \tau$ and $P' \not\equiv (\nu\,\tilde{m})(P_1 \mid P_2\{m/x\}) \wedge P' \not\equiv (\nu\,\tilde{m})(P_1 \mid P_2\{\lambda y.\,Q/x\})$ then
   $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\tau} (\!(\Delta'_1)\!)^1 \vdash [\![P']\!]^1_f$.
2. Suppose $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash [\![P]\!]^1_f \xrightarrow{\ell_2} (\!(\Delta')\!)^1 \vdash Q$. Then we have:
   a) If $\ell_2 \in \{(\nu\,\tilde{m})n!\langle \lambda z.\,z?(x).(x\,m)\rangle, (\nu\,\tilde{m})n!\langle \lambda x.\,R\rangle, s\oplus l, s\&l\}$ then $\exists \ell_1, P'$ s.t.
   $\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'$, $\ell_1 = \{\!\{\ell_2\}\!\}^1$, and $Q = [\![P']\!]^1_f$.
   b) If $\ell_2 = n?\langle \lambda y.\,R\rangle$ then either:
      (i) $\exists \ell_1, x, P', P''$ s.t.
      $\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'\{\lambda y.\,P''/x\}$, $\ell_1 = \{\!\{\ell_2\}\!\}^1$, $[\![P'']\!]^1_\emptyset = R$, and $Q = [\![P']\!]^1_f$.
      (ii) $R \equiv y?(x).(x\,m)$ and $\exists \ell_1, z, P'$ s.t.
      $\Gamma;\Delta \vdash P \xrightarrow{\ell_1} \Delta' \vdash P'\{m/z\}$, $\ell_1 = \{\!\{\ell_2\}\!\}^1$, and
      $(\!(\Gamma)\!)^1;(\!(\Delta')\!)^1 \vdash Q \xrightarrow{\tau_\beta}\xrightarrow{\tau_{\mathsf{s}}}\xrightarrow{\tau_\beta} (\!(\Delta'')\!)^1 \vdash [\![P'\{m/z\}]\!]^1_f$
   c) If $\ell_2 = \tau$ then $\Delta' = \Delta$ and either
      (i) $\exists P'$ s.t. $\Gamma;\Delta \vdash P \xrightarrow{\tau} \Delta \vdash P'$, and $Q = [\![P']\!]^1_f$.
      (ii) $\exists P_1, P_2, x, m, Q'$ s.t. $\Gamma;\Delta \vdash P \xrightarrow{\tau} \Delta \vdash (\nu\,\tilde{m})(P_1 \mid P_2\{m/x\})$, and
      $(\!(\Gamma)\!)^1;(\!(\Delta)\!)^1 \vdash Q \xrightarrow{\tau_\beta}\xrightarrow{\tau_{\mathsf{s}}}\xrightarrow{\tau_\beta} (\!(\Delta)\!)^1 \vdash [\![P_1]\!]^1_f \mid [\![P_2\{m/x\}]\!]^1_f$

Proof. By transition induction. We consider parts (1) and (2) separately:
Part (1) - Completeness. We consider two representative cases, the rest is similar or simpler:

1. Subcase (a): $P = s!\langle n\rangle.P'$ and $\ell_1 = s!\langle n\rangle$ (the case $\ell_1 = (\nu\ n)s!\langle n\rangle$ is similar). By assumption, P is well-typed. We may have:

$$\frac{\Gamma;\emptyset;\Delta_0\cdot s:S_1 \vdash P' \triangleright \diamond \quad \Gamma;\emptyset;\{n{:}S\} \vdash n \triangleright S}{\Gamma;\emptyset;\Delta_0\cdot n{:}S\cdot s:!\langle S\rangle;S_1 \vdash s!\langle n\rangle.P' \triangleright \diamond}$$

for some S, S_1, Δ_0. We may then have the following transition:

$$\Gamma;\Delta_0\cdot n{:}S\cdot s:!\langle S\rangle;S_1 \vdash s!\langle n\rangle.P' \overset{\ell_1}{\longmapsto} \Gamma;\Delta_0\cdot s{:}S_1 \vdash P'$$

The encoding of the source judgment for P is as follows:

$$\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\cdot n{:}S\cdot s:!\langle S\rangle;S_1\rangle\!\rangle^1 \vdash [\![s!\langle n\rangle.P']\!]^1 \triangleright \diamond$$

which, using Definition 6.3 can be expressed as

$$\langle\!\langle \Gamma\rangle\!\rangle^{\mathsf{p}};\emptyset;\langle\!\langle \Delta_0\rangle\!\rangle\cdot n{:}\langle\!\langle S\rangle\!\rangle^1\cdot s:!\langle ?(\langle\!\langle S\rangle\!\rangle^1{\multimap}\diamond);\mathsf{end}{\multimap}\diamond\rangle;\langle\!\langle S_1\rangle\!\rangle^1 \vdash s!\langle \lambda z.\, z?(x).(x\,n)\rangle.[\![P']\!]^1 \triangleright \diamond$$

Now, $\{\!\{\ell_1\}\!\}^1 = s!\langle \lambda z.\, z?(x).x\,n\rangle$. We may infer the following transition for $[\![P]\!]^1$:

$$\begin{aligned}
&\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta\rangle\!\rangle^1 \vdash s!\langle \lambda z.\, z?(x).(x\,n)\rangle.[\![P']\!]^1 \triangleright \diamond\\
\overset{\{\!\{\ell_1\}\!\}^1}{\longmapsto}\ &\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\rangle\!\rangle^1\cdot s:\langle\!\langle S_1\rangle\!\rangle^1 \vdash [\![P']\!]^1 \triangleright \diamond\\
=\ &\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\cdot s:S_1\rangle\!\rangle^1 \vdash [\![P']\!]^1 \triangleright \diamond
\end{aligned}$$

from which the thesis follows easily.

2. Subcase (c): $P = n?(x).P'$ and $\ell_1 = n?\langle m\rangle$. By assumption P is well-typed. We may have:

$$\frac{\Gamma;\emptyset;\Delta_0\cdot x:S\cdot n:S_1 \vdash P' \triangleright \diamond \quad \Gamma;\emptyset;\{x:S\} \vdash x \triangleright S}{\Gamma;\emptyset;\Delta_0\cdot n:?(S);S_1 \vdash n?(x).P' \triangleright \diamond}$$

for some S, S_1, Δ_0. We may infer the following typed transition:

$$\Gamma;\emptyset;\Delta_0\cdot n:?(S);S_1 \vdash n?(x).P' \triangleright \diamond \overset{n?\langle m\rangle}{\longmapsto} \Gamma;\emptyset;\Delta_0\cdot n:S_1\cdot m:S \vdash P'\{m/x\} \triangleright \diamond$$

The encoding of the source judgment for P is as follows:

$$\begin{aligned}
&\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\cdot n:?(S);S_1\rangle\!\rangle^1 \vdash [\![P]\!]^1 \triangleright \diamond\\
=\ &\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\rangle\!\rangle^1\cdot n:?(?(\langle\!\langle S\rangle\!\rangle^1{\multimap}\diamond);\mathsf{end}{\multimap}\diamond);\langle\!\langle S_1\rangle\!\rangle^1 \vdash n?(x).(\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\,[\![P']\!]^1\rangle.\mathbf{0}) \triangleright \diamond
\end{aligned}$$

Now, $\{\!\{\ell_1\}\!\}^1 = n?\langle \lambda z.\, z?(x).(x\,m)\rangle$ and it is immediate to infer the following transition for $[\![P]\!]^1$:

$$\begin{aligned}
&\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\rangle\!\rangle^1\cdot n:?(?(\langle\!\langle S\rangle\!\rangle^1{\multimap}\diamond);\mathsf{end}{\multimap}\diamond);\langle\!\langle S_1\rangle\!\rangle^1 \vdash n?(x).(\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\,[\![P']\!]^1\rangle.\mathbf{0}) \triangleright \diamond\\
\overset{\{\!\{\ell_1\}\!\}^1}{\longmapsto}\ &\langle\!\langle \Gamma\rangle\!\rangle^1;\emptyset;\langle\!\langle \Delta_0\rangle\!\rangle^1\cdot n:\langle\!\langle S_1\rangle\!\rangle^1\cdot m:\langle\!\langle S\rangle\!\rangle^1 \vdash (\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\,[\![P']\!]^1\rangle.\mathbf{0})\{\lambda z.\, z?(x).(x\,m)/x\} \triangleright \diamond
\end{aligned}$$

Let us write R to stand for process $(\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\,[\![P']\!]^1\rangle.\mathbf{0})\{\lambda z.\, z?(x).(x\,m)/x\}$. We then have:

$$\begin{aligned}
R &\overset{\tau}{\longrightarrow} (\nu\ s)(s?(x).(x\,m) \mid \overline{s}!\langle \lambda x.\,[\![P']\!]^1\rangle.\mathbf{0})\\
&\overset{\tau}{\longrightarrow} (\lambda x.\,[\![P']\!]^1)\,m \mid \mathbf{0}\\
&\overset{\tau}{\longrightarrow} [\![P']\!]^1\{m/x\}
\end{aligned}$$

and so the thesis follows.

Part (2) - Soundness. We consider two representative cases, the rest is similar or simpler:

1. Subcase (a): $P = n!\langle m\rangle.P'$ and $\ell_2 = n!\langle \lambda z.\ z?(x).(x\,m)\rangle$ (the case $\ell_2 = (\nu\, m)n!\langle \lambda z.\ z?(x).(x\,m)\rangle$ is similar). Then we have:

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n :!\langle ?(\langle\!\langle S \rangle\!\rangle^1 \multimap \diamond); \mathsf{end} \multimap \diamond\rangle; \langle\!\langle S_1 \rangle\!\rangle^1 \vdash n!\langle \lambda z.\ z?(x).(x\,m)\rangle.[\![P']\!]^1 \triangleright \diamond$$

for some S, S_1, and Δ_0. We may infer the following typed transition for $[\![P]\!]^1$:

$$\begin{aligned} & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n :!\langle ?(\langle\!\langle S \rangle\!\rangle^1 \multimap \diamond); \mathsf{end} \multimap \diamond\rangle; \langle\!\langle S_1 \rangle\!\rangle^1 \vdash n!\langle \lambda z.\ z?(x).(x\,m)\rangle.[\![P']\!]^1 \\ \stackrel{\ell_2}{\longmapsto}\ & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n : \langle\!\langle S_1 \rangle\!\rangle^1 \vdash [\![P']\!]^1 \end{aligned}$$

Now, in the source term P we can infer the following transition

$$\Gamma; \Delta_0 \cdot n :!\langle S\rangle; S_1 \vdash n!\langle m\rangle.P' \stackrel{n!\langle m\rangle}{\longmapsto} \Gamma; \Delta_0 \cdot n : S_1 \vdash P'$$

and thus the thesis follows easily by noticing that $\{\!\!\{ n!\langle m\rangle \}\!\!\}^1 = n!\langle \lambda z.\ z?(x).(x\,m)\rangle$.

2. Subcase (c): $P = n?(x).P'$ and $\ell_2 = n?\langle \lambda y.\, y?(x).(x\,m)\rangle$. Then we have

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n :?(?(\langle\!\langle S \rangle\!\rangle^1 \multimap \diamond); \mathsf{end} \multimap \diamond); \langle\!\langle S_1 \rangle\!\rangle^1 \vdash n?(x).(\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1\rangle.\mathbf{0}) \triangleright \diamond$$

for some S, S_1, Δ_0. We may infer the following typed transitions for $[\![P]\!]^1$:

$$\begin{aligned} & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n :?(?(\langle\!\langle S \rangle\!\rangle^1 \multimap \diamond); \mathsf{end} \multimap \diamond); \langle\!\langle S_1 \rangle\!\rangle^1 \vdash n?(x).(\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1\rangle.\mathbf{0}) \\ \stackrel{\ell_2}{\longmapsto}\ & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n : \langle\!\langle S_1 \rangle\!\rangle^1 \cdot m : \langle\!\langle S_1 \rangle\!\rangle^1 \vdash (\nu\ s)((x\,s) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1\rangle.\mathbf{0})\{^{\lambda z.\, z?(x).x m}/x\} \\ =\ & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n : \langle\!\langle S_1 \rangle\!\rangle^1 \cdot m : \langle\!\langle S \rangle\!\rangle^1 \vdash (\nu\ s)(s?(x).(x\,m) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1\rangle.\mathbf{0}) \\ \stackrel{\tau}{\longmapsto}\ & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n : \langle\!\langle S_1 \rangle\!\rangle^1 \cdot m : \langle\!\langle S \rangle\!\rangle^1 \vdash (\lambda x.\, [\![P']\!]^1)\, m \\ \stackrel{\tau}{\longmapsto}\ & \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_0 \rangle\!\rangle^1 \cdot n : \langle\!\langle S_1 \rangle\!\rangle^1 \cdot m : \langle\!\langle S \rangle\!\rangle^1 \vdash [\![P']\!]^1\{^m/x\} \end{aligned}$$

Now, in the source term P we can infer the following transition

$$\Gamma; \Delta_0 \cdot n :?(S); S_1 \vdash n?(x).P' \stackrel{n?\langle m\rangle}{\longmapsto} \Gamma; \Delta_0 \cdot n : S_1 \cdot m : S \vdash P'\{^m/x\}$$

and the thesis follows.

$\square$

We repeat the statement of Proposition 6.5, as in Page 31:

Proposition C.3 (Full Abstraction, HOπ into HO). $\Gamma;\Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1$ if and only if $\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \approx^H \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$.

Proof. **Proof of Soundness Direction.**
Let

$$\mathfrak{R} = \{\Gamma;\Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1 \ \mid\ \langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \approx^H \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_1]\!]^1_f\}$$

The proof considers a case analysis on the transition $\stackrel{\ell}{\longmapsto}$ and uses the soundness direction of operational correspondence (cf. Proposition 6.4). We give an interesting case. The others are similar of easier.
- Case: $\ell = (\nu\, \tilde{m_1}')n!\langle m_1\rangle$.
Proposition 6.4 implies that

$$\Gamma; \Delta_1 \vdash P_1 \stackrel{(\nu\, \tilde{m_1}')n!\langle m_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$$

implies

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \stackrel{(\nu\, \tilde{m_1}')n!\langle \lambda z. z?(x).(x\, m_1)\rangle}{\longmapsto} \langle\!\langle \Delta_1' \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f$$

that in combination with the definition of $\Re$ we get

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_1]\!]^1_f \stackrel{(\nu\, \tilde{m_2}')n!\langle \lambda z. z?(x).(x\, m_2)\rangle}{\Longmapsto} \langle\!\langle \Delta_2' \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f \tag{75}$$

and

$$\begin{aligned} \langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset;\, \langle\!\langle \Delta_1' \rangle\!\rangle^1 &\vdash (\nu\, \tilde{m_1}')([\![P_2]\!]^1_f \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda z. z?(x).(x\, m_1)\rangle.\mathbf{0})) \\ \approx^H \langle\!\langle \Delta_2' \rangle\!\rangle^1 &\vdash (\nu\, \tilde{m_2}')([\![Q_2]\!]^1_f \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle \lambda z. z?(x).(x\, m_2)\rangle.\mathbf{0})) \end{aligned}$$

We rewrite the last result as

$$\begin{aligned} \langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset;\, \langle\!\langle \Delta_1' \rangle\!\rangle^1 &\vdash [\![(\nu\, \tilde{m_1}')(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_1\rangle.\mathbf{0}))]\!]^1_f \\ \approx^H \langle\!\langle \Delta_2' \rangle\!\rangle^1 &\vdash [\![(\nu\, \tilde{m_2}')(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_2\rangle.\mathbf{0}))]\!]^1_f \end{aligned}$$

to conclude that

$$\begin{aligned} \Gamma; \emptyset;\, \Delta_1' &\vdash (\nu\, \tilde{m_1}')(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_1\rangle.\mathbf{0})) \\ \Re\;\; \Delta_2' &\vdash (\nu\, \tilde{m_2}')(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_2\rangle.\mathbf{0})) \end{aligned}$$

as required
Proof of Completeness Direction.
Let

$$\Re = \{\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f, \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_1]\!]^1_f \;\mid\; \Gamma; \Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1\}$$

We show that $\Re \subset \approx^H$ by a case analysis on the action ℓ
- Case: $\ell \notin \{(\nu\, \tilde{m})n!\langle \lambda x. P\rangle, n?\langle \lambda x. P\rangle\}$.
The proof of Proposition 6.4 implies that

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \stackrel{\ell}{\longmapsto} \langle\!\langle \Delta_1' \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f$$

implies

$$\Gamma; \Delta_1 \vdash P_1 \stackrel{\ell}{\longmapsto} \Delta_1' \vdash P_2$$

From the latter transition and the definition of $\Re$ we imply

$$\Gamma; \Delta_2 \vdash Q_1 \stackrel{\ell}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{76}$$

$$\Gamma; \Delta_1' \vdash P_2 \approx^H \Delta_2' \vdash Q_2 \tag{77}$$

From 76 and Proposition 6.4 we get

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_1]\!]^1_f \stackrel{\ell}{\Longmapsto} \langle\!\langle \Delta_2' \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$$

Furthermore, from 77 and the definition of $\mathfrak{R}$ we get

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1' \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f \ \mathfrak{R}\ \langle\!\langle \Delta_2' \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$$

as required.
- Case: $\ell = (\nu\, \tilde{m})n!\langle \lambda x.\, P\rangle$
There are two subcases:
-Subcase:
The proof of Proposition 6.4 implies that

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \stackrel{\ell}{\longmapsto} \langle\!\langle \Delta_1' \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f$$

implies

$$\Gamma; \Delta_1 \vdash P_1 \stackrel{\ell}{\longmapsto} \Delta_1' \vdash P_2$$

where the proof is similar with the previous case.
- Subcase:
The proof of Proposition 6.4 implies that

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \stackrel{(\nu\, \tilde{m}_1')n!\langle \lambda z.\, z?(x).(x\, m_1)\rangle}{\longmapsto} \langle\!\langle \Delta_1' \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f$$

implies

$$\Gamma; \Delta_1 \vdash P_1 \stackrel{(\nu\, \tilde{m}_1')n!\langle m_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$$

From the latter transition and the definition of $\mathfrak{R}$ we imply

$$\Gamma; \Delta_2 \vdash Q_1 \stackrel{(\nu\, \tilde{m}_2')n!\langle m_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{78}$$

and

$$\begin{aligned} &\Gamma; \emptyset; \Delta_1' \vdash (\nu\, \tilde{m}_1')(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_1\rangle.\mathbf{0})) \\ \approx^H \quad &\Delta_2' \vdash (\nu\, \tilde{m}_2')(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_2\rangle.\mathbf{0})) \end{aligned} \tag{79}$$

From (78) and Proposition 6.4 we get

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_1]\!]^1_f \stackrel{(\nu\, \tilde{m}_2')n!\langle \lambda z.\, z?(x).(x\, m_2)\rangle}{\Longmapsto} \langle\!\langle \Delta_2' \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$$

Furthermore, from (79) and the definition of $\mathfrak{R}$ we get

$$\begin{aligned} &\langle\!\langle \Gamma \rangle\!\rangle^1; \emptyset;\, \langle\!\langle \Delta_1' \rangle\!\rangle^1 \vdash [\![(\nu\, \tilde{m}_1')(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_1\rangle.\mathbf{0}))]\!]^1_f \\ &\quad \mathfrak{R}\ \ \langle\!\langle \Delta_2' \rangle\!\rangle^1 \vdash [\![(\nu\, \tilde{m}_2')(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle m_2\rangle.\mathbf{0}))]\!]^1_f \end{aligned}$$

as required.

- Case: $\ell = n?\langle \lambda x. P\rangle$
We have two subcases.
- Subcase: Similar with the first subcase of the previous case.
- Subcase: The proof of Proposition 6.4 implies that

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_1 \rangle\!\rangle^1 \vdash [\![P_1]\!]^1_f \stackrel{n?\langle \lambda z. z?(x).(x\,s)\rangle}{\longmapsto} \langle\!\langle \Delta''_1 \rangle\!\rangle^1 \vdash R$$

implies

$$\Gamma; \Delta_1 \vdash P_1 \stackrel{n?\langle m_1 \rangle}{\longmapsto} \Delta'_1 \vdash P_2 \tag{80}$$

and

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta''_1 \rangle\!\rangle^1 \vdash R \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \langle\!\langle \Delta'_1 \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f \tag{81}$$

From the transition (80) and the definition of $\Re$ we imply

$$\Gamma; \Delta_2 \vdash Q_1 \stackrel{n?\langle m_2 \rangle}{\Longmapsto} \Delta'_2 \vdash Q_2 \tag{82}$$

$$\Gamma; \Delta'_1 \vdash P_2 \approx^H \Delta'_2 \vdash Q_2 \tag{83}$$

From (82) and Proposition 6.4 we get

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta_2 \rangle\!\rangle^1 \vdash [\![Q_1]\!]^1_f \stackrel{n?\langle \lambda z. z?(x).(x\,s)\rangle}{\Longmapsto} \langle\!\langle \Delta'_2 \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$$

Furthermore, from 83 and the definition of $\Re$ we get

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta'_1 \rangle\!\rangle^1 \vdash [\![P_2]\!]^1_f \;\Re\; \langle\!\langle \Delta'_2 \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$$

If we consider result (81) we get:

$$\langle\!\langle \Gamma \rangle\!\rangle^1; \langle\!\langle \Delta''_1 \rangle\!\rangle^1 \vdash R \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \;\Re\; \langle\!\langle \Delta'_2 \rangle\!\rangle^1 \vdash [\![Q_2]\!]^1_f$$

where following Lemma 4.3 we show that R is a bisimulation an up to $\stackrel{\tau_{\mathsf{s}}}{\Longmapsto}$. ◻

C.2 Properties for $\langle [\![\cdot]\!]^2, \langle\!\langle\cdot\rangle\!\rangle^2, \{\!\{\cdot\}\!\}^2 \rangle : \mathsf{HO}\pi \to \pi$

We repeat the statement of Proposition 6.7, as in Page 34:

Proposition C.4 (Type Preservation, $\mathsf{HO}\pi$ into π). Let P be a $\mathsf{HO}\pi$ process.
If $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$ then $\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \triangleright \diamond$.

Proof. By induction on the inference $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$.

1. Case $P = k!\langle \lambda x. Q\rangle.P$. Then we have two possibilities, depending on the typing for $\lambda x. Q$. The first case concerns a linear typing, and we have the following typing in the source language:

$$\frac{\Gamma; \emptyset; \Delta_1 \cdot k : S \vdash P \triangleright \diamond \qquad \dfrac{\Gamma; \emptyset; \Delta_2 \cdot x : S_1 \vdash Q \triangleright \diamond}{\Gamma; \emptyset; \Delta_2 \vdash \lambda x. Q \triangleright S_1 \multimap \diamond}}{\Gamma; \emptyset; \Delta_1 \cdot \Delta_2 \cdot k : !\langle S_1 \multimap \diamond \rangle; S \vdash k!\langle \lambda x. Q\rangle.P \triangleright \diamond}$$

This way, by IH we have

$$\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2, x : \langle\!\langle S_1 \rangle\!\rangle^2 \vdash [\![Q]\!]^2 \triangleright \diamond$$

Let us write U_1 to stand for $\langle ?(\langle\!\langle S_1 \rangle\!\rangle^2); \mathsf{end} \rangle$. The corresponding typing in the target language is as follows:

$$\begin{aligned}
\langle\!\langle \Gamma_1 \rangle\!\rangle^2 &= \langle\!\langle \Gamma \rangle\!\rangle^2 \cup a : \langle ?(\langle\!\langle S_1 \rangle\!\rangle^2); \mathsf{end} \rangle \\
\langle\!\langle \Gamma_2 \rangle\!\rangle^2 &= \langle\!\langle \Gamma_1 \rangle\!\rangle^2 \cup X : \langle\!\langle \Delta_2 \rangle\!\rangle^2
\end{aligned}$$

Also $(*)$ stands for $\langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \emptyset \vdash a \triangleright U_1$; $(**)$ stands for $\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \emptyset \vdash a \triangleright U_1$; and $(***)$ stands for $\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \emptyset \vdash X \triangleright \diamond$.

$$\dfrac{\dfrac{}{(***)} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2, x : \langle\!\langle S_1 \rangle\!\rangle^2 \vdash [\![Q]\!]^2 \triangleright \diamond}}{\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2, y : \mathsf{end}, x : \langle\!\langle S_1 \rangle\!\rangle^2 \vdash [\![Q]\!]^2 \triangleright \diamond}}{\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2, y : ?(\langle\!\langle S_1 \rangle\!\rangle^2); \mathsf{end} \vdash y?(x).[\![Q]\!]^2 \triangleright \diamond} \quad \dfrac{}{(**)}}{\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2 \vdash a?(y).y?(x).[\![Q]\!]^2 \triangleright \diamond}}{\dfrac{\langle\!\langle \Gamma_2 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2 \vdash a?(y).y?(x).[\![Q]\!]^2 \mid X \triangleright \diamond}{\langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2 \vdash \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond}} \tag{84}$$

$$\dfrac{\begin{array}{c}\langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_1 \rangle\!\rangle^2, k : \langle\!\langle S \rangle\!\rangle^2 \vdash [\![P]\!]^2 \triangleright \diamond \\ \langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^2 \vdash \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond \quad (84)\end{array}}{\langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_1, \Delta_2 \rangle\!\rangle^2, k : \langle\!\langle S \rangle\!\rangle^2 \vdash [\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond} \tag{85}$$

$$\dfrac{\dfrac{\begin{array}{c}\langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \emptyset \vdash a \triangleright U_1 \\ \langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_1, \Delta_2 \rangle\!\rangle^2, k : \langle\!\langle S \rangle\!\rangle^2 \vdash [\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond \quad (85)\end{array}}{\langle\!\langle \Gamma_1 \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_1, \Delta_2 \rangle\!\rangle^2, k : !\langle U_1 \rangle; \langle\!\langle S \rangle\!\rangle^2 \vdash k!\langle a \rangle.([\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X)) \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta_1, \Delta_2 \rangle\!\rangle^2, k : !\langle U_1 \rangle; \langle\!\langle S \rangle\!\rangle^2 \vdash (\nu\, a)(k!\langle a \rangle.([\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X))) \triangleright \diamond}$$

In the second case, $\lambda x. Q$ has a shared type. We have the following typing in the source language:

$$\dfrac{\Gamma; \emptyset; \Delta \cdot k : S \vdash P \triangleright \diamond \quad \dfrac{\dfrac{\Gamma; \emptyset; \cdot x : S_1 \vdash Q \triangleright \diamond}{\Gamma; \emptyset; \emptyset \vdash \lambda x. Q \triangleright S_1 \multimap \diamond}}{\Gamma; \emptyset; \emptyset \vdash \lambda x. Q \triangleright S_1 \rightarrow \diamond}}{\Gamma; \emptyset; \Delta \cdot k : !\langle S_1 \rightarrow \diamond \rangle; S \vdash k!\langle \lambda x. Q \rangle.P \triangleright \diamond}$$

The corresponding typing in the target language can be derived similarly as in the first case.

2. Case $P = k?(x).P$. Then there are two cases, depending on the type of X. In the first case, we have the following typing in the source language:

$$\dfrac{\Gamma \cdot x : S_1 \rightarrow \diamond; \emptyset; \Delta \cdot k : S \vdash P \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot k : ?(S_1 \rightarrow \diamond); S \vdash k?(x).P \triangleright \diamond}$$

The corresponding typing in the target language is as follows:

$$\dfrac{\dfrac{}{\langle\!\langle \Gamma \rangle\!\rangle^2 \cdot x : \langle ?(\langle\!\langle S_1 \rangle\!\rangle^2); \mathsf{end} \rangle; \emptyset; \Delta \cdot k : \langle\!\langle S \rangle\!\rangle^2 \vdash \langle\!\langle P \rangle\!\rangle^2 \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^2 \cdot k : ?(\langle ?(\langle\!\langle S_1 \rangle\!\rangle^2); \mathsf{end} \rangle); \langle\!\langle S \rangle\!\rangle^2 \vdash k?(x).[\![P]\!]^2 \triangleright \diamond}$$

In the second case, we have the following typing in the source language:

$$\frac{\Gamma; \{x : S_1 \multimap \diamond\}; \emptyset; \Delta \cdot k : S \vdash P \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot k :?(S_1 \multimap \diamond); S \vdash k?(x).P \triangleright \diamond}$$

The corresponding typing in the target language is as follows:

$$\frac{(\!(\Gamma)\!)^2 \cdot x : \langle ?((\!(S_1)\!)^2); \mathsf{end}\rangle; \emptyset; \Delta \cdot k : (\!(S)\!)^2 \vdash (\!(P)\!)^2 \triangleright \diamond}{(\!(\Gamma)\!)^2; \emptyset; (\!(\Delta)\!)^2 \cdot k :?(\langle ?((\!(S_1)\!)^2); \mathsf{end}\rangle); (\!(S)\!)^2 \vdash k?(x).[\![P]\!]^2 \triangleright \diamond}$$

3. Case $P = x\,k$. Also here we have two cases, depending on whether X has linear or shared type. In the first case, x is linear and we have the following typing in the source language:

$$\frac{\Gamma; \{x : S_1 \multimap \diamond\}; \emptyset \vdash X \triangleright S_1 \multimap \diamond \quad \Gamma; \emptyset; \{k : S_1\} \vdash k \triangleright S_1}{\Gamma; \{x : S_1 \multimap \diamond\}; k : S_1 \vdash x\,k \triangleright \diamond}$$

Let us write $(\!(\Gamma_1)\!)^2$ to stand for $(\!(\Gamma)\!)^2 \cdot x : \langle !\langle (\!(S_1)\!)^2\rangle; \mathsf{end}\rangle$. The corresponding typing in the target language is as follows:

$$\frac{\dfrac{(\!(\Gamma_1)\!)^2; \emptyset; \emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!(\Gamma_1)\!)^2; \emptyset; \overline{s} : \mathsf{end} \vdash \mathbf{0} \triangleright \diamond} \quad (\!(\Gamma_1)\!)^2; \emptyset; \{k : (\!(S_1)\!)^2\} \vdash k \triangleright (\!(S_1)\!)^2}{(\!(\Gamma_1)\!)^2; \emptyset;\ k : (\!(S_1)\!)^2, \overline{s} :!\langle (\!(S_1)\!)^2\rangle; \mathsf{end} \vdash \overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond} \tag{86}$$

$$\frac{\dfrac{\begin{array}{c}(\!(\Gamma_1)\!)^2; \emptyset;\ k : (\!(S_1)\!)^2, \overline{s} :!\langle (\!(S_1)\!)^2\rangle; \mathsf{end} \vdash \overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond \quad (86)\\ (\!(\Gamma_1)\!)^2; \emptyset; \emptyset \vdash x \triangleright \langle !\langle (\!(S_1)\!)^2\rangle; \mathsf{end}\rangle\end{array}}{(\!(\Gamma_1)\!)^2; \emptyset; k : (\!(S_1)\!)^2, s :?((\!(S_1)\!)^2); \mathsf{end}, \overline{s} :!\langle (\!(S_1)\!)^2\rangle; \mathsf{end} \vdash x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond}}{(\!(\Gamma_1)\!)^2; \emptyset; k : (\!(S_1)\!)^2 \vdash (\nu\ s)(x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0}) \triangleright \diamond}$$

In the second case, x is shared, and we have the following typing in the source language:

$$\frac{\Gamma \cdot x : S_1 \multimap \diamond; \emptyset; \emptyset \vdash x \triangleright S_1 \rightarrow \diamond \quad \Gamma; \emptyset; k : S_1 \vdash k \triangleright S_1}{\Gamma \cdot x : S_1 \rightarrow \diamond; \emptyset; k : S_1 \vdash x\,k \triangleright \diamond}$$

The associated typing in the target language is obtained similarly as in the first case.
□

We repeat the statement of Proposition 6.8, as in Page 35:

Proposition C.5 (Operational Correspondence, HOπ into π). *Let P be an HOπ process such that $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$.*

1. *Suppose $\Gamma; \Delta \vdash P \stackrel{\ell_1}{\longmapsto} \Delta' \vdash P'$. Then we have:*
   a) *If $\ell_1 = (\nu\, \tilde{m})n!\langle \lambda x.\, Q\rangle$, then $\exists \Gamma', \Delta'', R$ where either:*
      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} \Gamma' \cdot (\!(\Gamma)\!)^2; (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$
      - $(\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} (\!(\Gamma)\!)^2; \Delta'' \vdash [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$

   b) *If $\ell_1 = n?\langle \lambda y.\, Q\rangle$ then $\exists R$ where either*

- $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} \Gamma'; \langle\!\langle \Delta'' \rangle\!\rangle^2 \vdash R$, *for some* Γ' *and* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2 \approx^H \langle\!\langle \Delta'' \rangle\!\rangle^2 \vdash (\nu\, a)(R \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
- $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{\{\!\{\ell_1\}\!\}^2}{\longmapsto} \langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta'' \rangle\!\rangle^2 \vdash R$, *and* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2 \approx^H \langle\!\langle \Delta'' \rangle\!\rangle^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$

c) If $\ell_1 = \tau$ *then either:*

- $\exists R$ *such that*

$$\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset;\ \langle\!\langle \Delta \rangle\!\rangle^2 \ \vdash [\![P]\!]^2 \stackrel{\tau}{\longmapsto} \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash (\nu\, \tilde{m})([\![P_1]\!]^2 \mid (\nu\, a)([\![P_2]\!]^2\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2))$$

- $\exists R$ *such that*

$$\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset;\ \langle\!\langle \Delta \rangle\!\rangle^2 \ \vdash [\![P]\!]^2 \stackrel{\tau}{\longmapsto} \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash (\nu\, \tilde{m})([\![P_1]\!]^2 \mid (\nu\, s)([\![P_2]\!]^2\{^{\bar{s}}/x\} \mid s?(y).y?(x).[\![Q]\!]^2))$$

- $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{\tau}{\longmapsto} \langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2$
- $\ell_1 = \tau_\beta$ *and* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{\tau_s}{\longmapsto} \langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2$

d) If $\ell_1 \in \{n \oplus l, n \& l\}$ *then*

$\exists \ell_2 = \{\!\{\ell_1\}\!\}^2$ *such that* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} \langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2$.

2. *Suppose* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} \langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash R$.

a) If $\ell_2 = (\nu\, m)n!\langle m \rangle$ *then either*

- $\exists P'$ *such that* $P \stackrel{(\nu\, m)n!\langle m \rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2$.
- $\exists Q, P'$ *such that* $P \stackrel{n!\langle \lambda x.\, Q \rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$
- $\exists Q, P'$ *such that* $P \stackrel{n!\langle \lambda x.\, Q \rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$

b) If $\ell_2 = n?\langle m \rangle$ *then either*

- $\exists P'$ *such that* $P \stackrel{n?\langle m \rangle}{\longmapsto} P'$ *and* $R = [\![P']\!]^2$.
- $\exists Q, P'$ *such that* $P \stackrel{n?\langle \lambda x.\, Q \rangle}{\longmapsto} P'$ *and* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2 \approx^H \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash (\nu\, a)(R \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
- $\exists Q, P'$ *such that* $P \stackrel{n?\langle \lambda x.\, Q \rangle}{\longmapsto} P'$ *and* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2 \approx^H \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$

c) If $\ell_2 = \tau$ *then* $\exists P'$ *such that* $P \stackrel{\tau}{\longmapsto} P'$ *and* $\langle\!\langle \Gamma \rangle\!\rangle^2; \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash [\![P']\!]^2 \approx^H \langle\!\langle \Delta' \rangle\!\rangle^2 \vdash R$.

d) If $\ell_2 \notin \{n!\langle m \rangle, n \oplus l, n \& l\}$ *then* $\exists \ell_1$ *such that* $\ell_1 = \{\!\{\ell_2\}\!\}^2$ *and*

$\Gamma; \Delta \vdash P \stackrel{\ell_1}{\longmapsto} \Gamma; \Delta \vdash P'$.

Proof. The proof is done by transition induction. We conside the two parts separately.
- Part 1
- Basic Step:
- Subcase: $P = n!\langle \lambda x.\, Q \rangle.P'$ and also from Definition 6.4 we have that $[\![P]\!]^2 = (\nu\, a)(n!\langle a \rangle.[\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
Then

$$\Gamma; \emptyset; \Delta \vdash P \stackrel{n!\langle \lambda x.\, Q \rangle}{\longmapsto} \Delta' \vdash P'$$

$$\langle\!\langle \Gamma \rangle\!\rangle^2; \emptyset; \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P]\!]^2 \stackrel{(\nu\, a)n!\langle a \rangle}{\longmapsto} \langle\!\langle \Delta \rangle\!\rangle^2 \vdash [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$$

and from Definition 6.4

$$\{\!\{n!\langle \lambda x.\, Q\rangle\}\!\} = (\nu\, a)n!\langle a\rangle$$

as required.
- Subcase: $P = n!\langle \lambda x.\, Q\rangle.P'$ and also from Definition 6.4 we have that $[\![P]\!]^2 = (\nu\, s)(n!\langle \overline{s}\rangle.[\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2)$ is similar as above.
- Subcase $P = n?(x).P'$.
- From Definition 6.4 we have that $[\![P]\!]^2 = n?(x).[\![P']\!]^2$
Then

$$\begin{array}{rcl} \Gamma; \emptyset; \Delta \vdash P & \stackrel{n?\langle \lambda x.\, Q\rangle}{\longmapsto} & \Delta' \vdash P'\{^{\lambda x.\, Q}/x\} \\ (\!(\Gamma)\!)^2; \emptyset; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 & \stackrel{n?\langle a\rangle}{\longrightarrow} & (\!(\Delta'')\!)^2 \vdash R\{^a/x\} \end{array}$$

with

$$\{\!\{n?\langle \lambda x.\, Q\rangle\}\!\}^2 = n?\langle a\rangle$$

It remains to show that

$$(\!(\Gamma)\!)^2; \emptyset; (\!(\Delta')\!)^2 \vdash [\![P'\{^{\lambda x.\, Q}/x\}]\!]^2 \approx^H (\!(\Delta'')\!)^2 \vdash (\nu\, a)(R\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

The proof is an induction on the syntax structure of P'. Suppose $P' = xm$, then:

$$\begin{array}{rcl} [\![xm\{^{\lambda x.\, Q}/x\}]\!]^2 & = & [\![Q\{^m/x\}]\!]^2 \\ (\nu\, a)(R\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2) & = & (\nu\, a)((\nu\, s)(x!\langle s\rangle.\overline{s}!\langle m\rangle.\mathbf{0})\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2) \end{array}$$

The second term can be deterministically reduced as:

$$\begin{array}{l} (\!(\Gamma)\!)^2; \emptyset;\ (\!(\Delta'')\!)^2 \vdash (\nu\, a)((\nu\, s)(x!\langle s\rangle.\overline{s}!\langle m\rangle.\mathbf{0})\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2) \\ \stackrel{\tau}{\longmapsto}\stackrel{\tau_{\mathsf{s}}}{\longmapsto} (\!(\Delta'')\!)^2 \vdash (\nu\, a)([\![Q\{^m/x\}]\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2) \end{array}$$

which is bisimilar with:

$$[\![Q\{^m/x\}]\!]^2$$

because a is fresh and cannot interact anymore.
An interesting inductive step case is parallel composition. Suppose $P' = P_1 \mid P_2$. We need to show that:

$$(\!(\Gamma)\!)^2; \emptyset; (\!(\Delta')\!)^2 \vdash [\![(P_1 \mid P_2)\{^{\lambda x.\, Q}/x\}]\!]^2 \approx^H (\!(\Delta'')\!)^2 \vdash (\nu\, a)([\![P_1 \mid P_2]\!]^2\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

We know that

$$\begin{array}{l} (\!(\Gamma)\!)^2; (\!(\Delta_1)\!)^2 \vdash [\![P_1\{^{\lambda x.\, Q}/x\}]\!]^2 \approx^H (\!(\Delta''_1)\!)^2 \vdash (\nu\, a)([\![P_1]\!]^2\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2) \\ (\!(\Gamma)\!)^2; (\!(\Delta_2)\!)^2 \vdash [\![P_2\{^{\lambda x.\, Q}/x\}]\!]^2 \approx^H (\!(\Delta''_1)\!)^2 \vdash (\nu\, a)([\![P_2]\!]^2\{^a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2) \end{array}$$

We conclude from the congruence of $\approx^H$.

- The rest of the cases for Part 1 are easy to follow using Definition 6.4.
- Part 2.
The proof for Part 2 is straightforward following Definition 6.4. We give some distinctive cases:
- Case $P = n!\langle \lambda x.\, Q\rangle.P'$

$$\begin{array}{c} \Gamma; \Delta \vdash P \overset{n!\langle \lambda x.\, Q\rangle}{\longmapsto} \Delta' \vdash P' \\ (\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 \overset{(\nu\, a)n!\langle a\rangle}{\longmapsto} (\!(\Delta')\!)^2 \vdash [\![P']\!]^2 \mid *\, a?(y).y?(s).[\![Q]\!]^2 \end{array}$$

as required.
- Case $P = n?(x).P'$

$$\begin{array}{rcl} \Gamma; \Delta \vdash P & \overset{n?\langle \lambda x.\, Q\rangle}{\longmapsto} & \Delta' \vdash P'\{^{\lambda x.}/Q\}x \\ (\!(\Gamma)\!)^2; (\!(\Delta)\!)^2 \vdash [\![P]\!]^2 & \overset{n?\langle a\rangle}{\longmapsto} & (\!(\Delta'')\!)^2 \vdash [\![P']\!]^2\{^{a}/x\} \end{array}$$

We now use a similar argumentation as the input case in Part 1 to prove that:

$$\Gamma; \Delta' \vdash P'\{^{\lambda x.\, Q}/x\} \approx^H (\!(\Delta'')\!)^2 \vdash (\nu\, a)([\![P']\!]^2\{^{a}/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

□

C.3 Properties for $\langle [\![\cdot]\!]^3, (\!(\cdot)\!)^3, \{\!\{\cdot\}\!\}^3 \rangle : \mathsf{HO}\pi^+ \to \mathsf{HO}\pi$

We study the properties of the typed encoding in Definition 8.1 (Page 39).

We repeat the statement of Proposition 8.1, as in Page 40:

Proposition C.6 (Type Preservation. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let P be a $\mathsf{HO}\pi^+$ process. If $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$ then $(\!(\Gamma)\!)^3; \emptyset; (\!(\Delta)\!)^3 \vdash [\![P]\!]^3 \triangleright \diamond$.*

Proof. By induction on the inference of $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. We detail some representative cases:

1. Case $P = u!\langle \lambda \underline{x}.\, Q\rangle.P'$. Then we may have the following typing in $\mathsf{HO}\pi^+$:

$$\dfrac{\dfrac{}{\Gamma; \Lambda_1; \Delta_1 \cdot u : S \vdash P' \triangleright \diamond} \quad \dfrac{\dfrac{}{\Gamma \cdot \underline{x} : L; \Lambda_2; \Delta_2 \vdash Q \triangleright \diamond} \quad \dfrac{}{\Gamma \cdot \underline{x} : L; \emptyset; \emptyset \vdash \underline{x} \triangleright L}}{\Gamma; \Lambda_2; \Delta_2 \vdash \lambda \underline{x} : L.\, Q \triangleright L \multimap \diamond}}{\Gamma; \Lambda_1 \cdot \Lambda_2; \Delta_1 \cdot \Delta_2 \cdot u : !\langle L \multimap \diamond\rangle; S \vdash u!\langle \lambda \underline{x}.\, Q\rangle.P' \triangleright \diamond}$$

Thus, by IH we have:

$$(\!(\Gamma)\!)^3; (\!(\Lambda_1)\!)^3; (\!(\Delta_1)\!)^3 \cdot u : (\!(S)\!)^3 \vdash [\![P']\!]^3 \triangleright \diamond \tag{87}$$

$$(\!(\Gamma)\!)^3 \cdot \underline{x} : (\!(L)\!)^3; (\!(\Lambda_2)\!)^3; (\!(\Delta_2)\!)^3 \vdash [\![Q]\!]^3 \triangleright \diamond \tag{88}$$

$$(\!(\Gamma)\!)^3 \cdot \underline{x} : (\!(L)\!)^3; \emptyset; \emptyset \vdash \underline{x} \triangleright (\!(L)\!)^3 \tag{89}$$

The corresponding typing in $\mathsf{HO}\pi$ is as follows:

$$\dfrac{\dfrac{\overline{(88)}}{\langle\!\langle \Gamma \rangle\!\rangle^3 \cdot x : \langle\!\langle L \rangle\!\rangle^3; \langle\!\langle \Lambda_2 \rangle\!\rangle^3; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot z : \mathsf{end} \vdash [\![Q]\!]^3 \triangleright \diamond} \qquad \overline{(89)}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Lambda_2 \rangle\!\rangle^3; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot z : ?(\langle\!\langle L \rangle\!\rangle^3); \mathsf{end} \vdash z?(\underline{x}).[\![Q]\!]^3 \triangleright \diamond} \tag{90}$$

$$\dfrac{\overline{(87)} \qquad \dfrac{(90) \qquad \overline{\langle\!\langle \Gamma \rangle\!\rangle^3; \emptyset; z : ?(\langle\!\langle L \rangle\!\rangle^3); \mathsf{end} \vdash z \triangleright ?(\langle\!\langle L \rangle\!\rangle^3); \mathsf{end}}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Lambda_2 \rangle\!\rangle^3; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \vdash \lambda z. z?(\underline{x}).[\![Q]\!]^3 \triangleright (?(\langle\!\langle L \rangle\!\rangle^3); \mathsf{end}) \multimap \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Lambda_1 \rangle\!\rangle^3 \cdot \langle\!\langle \Lambda_2 \rangle\!\rangle^3; \langle\!\langle \Delta_1 \rangle\!\rangle^3 \cdot \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot u : !\langle ?(\langle\!\langle L \rangle\!\rangle^3); \mathsf{end} \multimap \diamond \rangle; \langle\!\langle S \rangle\!\rangle^3 \vdash u!\langle \lambda z. z?(\underline{x}).[\![Q]\!]^3 \rangle.[\![P']\!]^3 \triangleright \diamond}$$

2. Case $P = (\lambda x. P)(\lambda y. Q)$. We may have different possibilities for the types of each abstraction. We consider only one of them, as the rest are similar:

$$\dfrac{\dfrac{\overline{\Gamma \cdot x : C \to \diamond; \Lambda; \Delta_1 \vdash P \triangleright \diamond}}{\Gamma; \Lambda; \Delta_1 \vdash \lambda x. P \triangleright (C \multimap \diamond) \multimap \diamond} \qquad \dfrac{\overline{\Gamma; \emptyset; \Delta_2, y : C \vdash Q \triangleright \diamond}}{\Gamma; \emptyset; \Delta_2 \vdash \lambda y. Q \triangleright C \multimap \diamond}}{\Gamma; \Lambda; \Delta_1 \cdot \Delta_2 \vdash (\lambda x. P)(\lambda y. Q) \triangleright \diamond}$$

Thus, by IH we have:

$$\langle\!\langle \Gamma \rangle\!\rangle^3 \cdot x : \langle\!\langle C \to \diamond \rangle\!\rangle^3; \langle\!\langle \Lambda \rangle\!\rangle^3; \langle\!\langle \Delta_1 \rangle\!\rangle^3 \vdash [\![P]\!]^3 \triangleright \diamond \tag{91}$$

$$\langle\!\langle \Gamma \rangle\!\rangle^3; \emptyset; \langle\!\langle \Delta_1 \rangle\!\rangle^3, y : \langle\!\langle C \rangle\!\rangle^3 \vdash [\![Q]\!]^3 \triangleright \diamond \tag{92}$$

The corresponding typing in $\mathsf{HO}\pi$ is as follows — recall that $\langle\!\langle C \multimap \diamond \rangle\!\rangle^3 = \langle\!\langle C \rangle\!\rangle^3 \multimap \diamond$.

$$\dfrac{\dfrac{\overline{(91)}}{\langle\!\langle \Gamma \rangle\!\rangle^3 \cdot x : \langle\!\langle C \to \diamond \rangle\!\rangle^3; \langle\!\langle \Lambda \rangle\!\rangle^3; \langle\!\langle \Delta_1 \rangle\!\rangle^3 \cdot s : \mathsf{end} \vdash [\![P]\!]^3 \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Lambda \rangle\!\rangle^3; \langle\!\langle \Delta_1 \rangle\!\rangle^3 \cdot s : ?(\langle\!\langle C \multimap \diamond \rangle\!\rangle^3); \mathsf{end} \vdash s?(x).[\![P]\!]^3 \triangleright \diamond} \tag{93}$$

$$\dfrac{\dfrac{(93) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{(92)}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot y : \langle\!\langle C \rangle\!\rangle^3 \vdash [\![Q]\!]^3 \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \vdash \lambda y. [\![Q]\!]^3 \triangleright \langle\!\langle C \multimap \diamond \rangle\!\rangle^3}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot \overline{s} : \mathsf{end} \vdash \lambda y. [\![Q]\!]^3 \triangleright \langle\!\langle C \multimap \diamond \rangle\!\rangle^3}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \emptyset; \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot \overline{s} : !\langle \langle\!\langle C \multimap \diamond \rangle\!\rangle^3 \rangle; \mathsf{end} \vdash \overline{s}!\langle \lambda y. [\![Q]\!]^3 \rangle.\mathbf{0} \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Lambda \rangle\!\rangle^3; \langle\!\langle \Delta_1 \rangle\!\rangle^3 \cdot \langle\!\langle \Delta_2 \rangle\!\rangle^3 \cdot s : ?(\langle\!\langle C \multimap \diamond \rangle\!\rangle^3); \mathsf{end} \cdot \overline{s} : !\langle \langle\!\langle C \multimap \diamond \rangle\!\rangle^3 \rangle; \mathsf{end} \vdash s?(x).[\![P]\!]^3 \mid \overline{s}!\langle \lambda y. [\![Q]\!]^3 \rangle.\mathbf{0} \triangleright \diamond}}{\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Lambda \rangle\!\rangle^3; \langle\!\langle \Delta_1 \rangle\!\rangle^3 \cdot \langle\!\langle \Delta_2 \rangle\!\rangle^3 \vdash (\nu\, s)(s?(x).[\![P]\!]^3 \mid \overline{s}!\langle \lambda y. [\![Q]\!]^3 \rangle.\mathbf{0}) \triangleright \diamond}$$

□

We repeat the statement of Proposition 8.2, as in Page 40:

Proposition C.7 (Operational Correspondence. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$).

1. *Let $\Gamma; \emptyset; \Delta \vdash P$. $\Gamma; \Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$ implies*
   a) *If $\ell \in \{(\nu\, \tilde{m})n!\langle \lambda x. Q \rangle, n?\langle \lambda x. Q \rangle\}$ then $\langle\!\langle \Gamma \rangle\!\rangle^3; \langle\!\langle \Delta \rangle\!\rangle^3 \vdash [\![P]\!]^3 \overset{\ell'}{\longmapsto} \langle\!\langle \Delta' \rangle\!\rangle^3 \vdash [\![P']\!]^3$ with $\{\!\{ \ell \}\!\}^3 = \ell'$.*

b) *If* $\ell \notin \{(\nu\,\tilde{m})n!\langle\lambda x.\,Q\rangle, n?\langle\lambda x.\,Q\rangle, \tau\}$ *then* $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash [\![P]\!]^3 \stackrel{\ell}{\longmapsto} \langle\!\langle\Delta'\rangle\!\rangle^3 \vdash [\![P']\!]^3$.

c) *If* $\ell = \tau_\beta$ *then* $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash [\![P]\!]^3 \stackrel{\tau}{\longmapsto} \Delta'' \vdash R$ *and* $\langle\!\langle\Gamma\rangle\!\rangle^3\langle\!\langle\Delta'\rangle\!\rangle^3[\![P']\!]^3 \approx^H \Delta'' R$.

d) *If* $\ell = \tau$ *and* $\ell \neq \tau_\beta$ *then* $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash [\![P]\!]^3 \stackrel{\tau}{\longmapsto} \langle\!\langle\Delta'\rangle\!\rangle^3 \vdash [\![P']\!]^3$.

2. *Let* $\Gamma;\emptyset;\Delta \vdash P$. $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash [\![P]\!]^3 \stackrel{\ell}{\longmapsto} \langle\!\langle\Delta''\rangle\!\rangle^3 \vdash Q$ *implies*

a) *If* $\ell \in \{(\nu\,\tilde{m})n!\langle\lambda x.\,Q\rangle, n?\langle\lambda x.\,Q\rangle, \tau\}$ *then* $\Gamma;\Delta \vdash P \stackrel{\ell'}{\longmapsto} \Delta' \vdash P'$ *with* $\{\!\!\{\ell'\}\!\!\}^3 = \ell$ *and* $Q \equiv [\![P']\!]^3$.

b) *If* $\ell \notin \{(\nu\,\tilde{m})n!\langle\lambda x.\,R\rangle, n?\langle\lambda x.\,R\rangle, \tau\}$ *then* $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ *and* $Q \equiv [\![P']\!]^3$.

c) *If* $\ell = \tau$ *then either* $\Gamma;\Delta \vdash \Delta \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ *with* $Q \equiv [\![P']\!]^3$
or $\Gamma;\Delta \vdash \Delta \stackrel{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ *and* $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta''\rangle\!\rangle^3 \vdash Q \stackrel{\tau_\beta}{\longmapsto} \langle\!\langle\Delta''\rangle\!\rangle^3 \vdash [\![P']\!]^3$.

Proof. 1. The proof of Part 1 does a transition induction and considers the mapping as defined in Definition 8.1. We give the most interesting cases.

- Case: $P = (\lambda x.\,Q_1)\,\lambda x.\,Q_2$.
  $\Gamma;\Delta \vdash (\lambda x.\,Q_1)\,\lambda x.\,Q_2 \stackrel{\tau_\beta}{\longmapsto} \Delta \vdash Q_1\{\lambda x.\,Q_2/x\}$ implies

  $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash (\nu\,s)(s?(x).[\![Q_1]\!]^3 \mid \overline{s}!\langle\lambda x.\,[\![Q_2]\!]^3\rangle.\mathbf{0}) \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \langle\!\langle\Delta'\rangle\!\rangle^3 \vdash [\![Q_1]\!]^3\{\lambda x.\,[\![Q_2]\!]^3/x\}$

- Case: $P = n!\langle\lambda \underline{x}.\,Q\rangle.P$
  $\Gamma;\Delta \vdash n!\langle\lambda \underline{x}.\,Q\rangle.P \stackrel{n!\langle\lambda x.\,Q\rangle}{\longmapsto} \Delta \vdash P$ implies
  $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash n!\langle\lambda z.\,z?(x).[\![Q]\!]^3\rangle.[\![P]\!]^3 \stackrel{n!\langle\lambda z.\,z?(x).[\![Q]\!]^3\rangle}{\longmapsto} \Delta \vdash [\![P]\!]^3$
- Other cases are similar.

2. The proof of Part 2 also does a transition induction and considers the mapping as defined in Definition 8.1. We give the most interesting cases.

- Case: $P = (\lambda x.\,Q_1)\,\lambda x.\,Q_2$.

$$\begin{aligned}\langle\!\langle\Gamma\rangle\!\rangle^3;\emptyset;\ \langle\!\langle\Delta\rangle\!\rangle^3\ &\vdash (\nu\,s)((\lambda z.\,z?(x).[\![Q]\!]^3)\,s \mid \overline{s}!\langle\lambda x.\,Q_2\rangle.\mathbf{0})\\ \stackrel{\tau_\beta}{\longmapsto} \langle\!\langle\Delta'\rangle\!\rangle^3\ &\vdash (\nu\,s)(s?(x).[\![Q]\!]^3 \mid \overline{s}!\langle\lambda x.\,Q_2\rangle.\mathbf{0})\end{aligned}$$

  implies $\Gamma;\Delta \vdash (\lambda x.\,Q_1)\,\lambda x.\,Q_2 \stackrel{\tau_\beta}{\longmapsto} \Delta \vdash Q_1\{\lambda x.\,Q_2/x\}$ and

$$\begin{aligned}\langle\!\langle\Gamma\rangle\!\rangle^3;\emptyset;\ \langle\!\langle\Delta\rangle\!\rangle^3\ &\vdash (\nu\,s)(s?(x).[\![Q]\!]^3 \mid \overline{s}!\langle\lambda x.\,Q_2\rangle.\mathbf{0})\\ \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \langle\!\langle\Delta'\rangle\!\rangle^3\ &\vdash [\![Q_1]\!]^3\{\lambda x.\,[\![Q_2]\!]^3/x\}\end{aligned}$$

- Case: $P = n!\langle\lambda \underline{x}.\,Q\rangle.P$
  $\langle\!\langle\Gamma\rangle\!\rangle^3;\langle\!\langle\Delta\rangle\!\rangle^3 \vdash n!\langle\lambda z.\,z?(x).[\![Q]\!]^3\rangle.[\![P]\!]^3 \stackrel{n!\langle\lambda z.\,z?(x).[\![Q]\!]^3\rangle}{\longmapsto} \Delta \vdash [\![P]\!]^3$ and
  $\Gamma;\Delta \vdash n!\langle\lambda \underline{x}.\,Q\rangle.P \stackrel{n!\langle\lambda \underline{x}.\,Q\rangle}{\longmapsto} \Delta \vdash P$
- Other cases are similar.

$\square$

C.4 Properties for $\langle [\![\cdot]\!]^4, (\!(\cdot)\!)^4, \{\!\{\cdot\}\!\}^4 \rangle : \mathsf{HO}\vec{\pi} \to \mathsf{HO}\pi$

We study the properties of the typed encoding in Definition 8.2 (Page 43).

We repeat the statement of Proposition 8.5, as in Page 43:

Proposition C.8 (Type Preservation. From $\mathsf{HO}\vec{\pi}$ to $\mathsf{HO}\pi$). Let P be a $\mathsf{HO}\vec{\pi}$ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then $(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond$.

Proof. By induction on the inference $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. We examine two representative cases, using biadic communications.

1. Case $P = n!\langle V\rangle.P'$ and $\Gamma;\emptyset;\Delta_1 \cdot \Delta_2 \cdot n :!\langle (C_1,C_2)\multimap\diamond\rangle;S \vdash n!\langle V\rangle.P' \triangleright \diamond$. Then either $V = y$ or $V = \lambda(x_1,x_2).\,Q$, for some Q. The case $V = y$ is immediate; we give details for the case $V = \lambda(x_1,x_2).\,Q$, for which we have the following typing:

$$\dfrac{\dfrac{}{\Gamma;\emptyset;\Delta_1 \cdot n : S \vdash P' \triangleright \diamond} \quad \dfrac{\Gamma;\emptyset;\Delta_2 \cdot x_1 : C_1 \cdot x_2 : C_2 \vdash Q \triangleright \diamond}{\Gamma;\emptyset;\Delta_2 \vdash \lambda(x_1,x_2).\,Q \triangleright (C_1,C_2)\multimap\diamond}}{\Gamma;\emptyset;\Delta_1 \cdot \Delta_2 \cdot n :!\langle (C_1,C_2)\multimap\diamond\rangle;S \vdash k!\langle \lambda(x_1,x_2).\,Q\rangle.P \triangleright \diamond}$$

We now show the typing for $[\![P]\!]^4$. By IH we have both:

$$(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_1)\!)^4 \cdot n : (\!(S)\!)^4 \vdash [\![P']\!]^4 \triangleright \diamond \qquad (\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_2)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot x_2 : (\!(C_2)\!)^4 \vdash [\![Q]\!]^4 \triangleright \diamond$$

Let $L = (C_1,C_2)\multimap\diamond$. By Definition 8.2 we have $(\!(L)\!)^4 = (?((\!(C_1)\!)^4);?((\!(C_2)\!)^4);\mathsf{end})\multimap\diamond$ and $[\![P]\!]^4 = n!\langle \lambda z.\,z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4$. We can now infer the following typing derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_2)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot x_2 : (\!(C_2)\!)^4 \vdash [\![Q]\!]^4 \triangleright \diamond}}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_2)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot x_2 : (\!(C_2)\!)^4 \cdot z : \mathsf{end} \vdash [\![Q]\!]^4 \triangleright \diamond}}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_2)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot z :?((\!(C_2)\!)^4);\mathsf{end} \vdash z?(x_2).[\![Q]\!]^4 \triangleright \diamond}}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_2)\!)^4 \cdot z :?((\!(C_1)\!)^4);?((\!(C_2)\!)^4);\mathsf{end} \vdash z?(x_1).z?(x_2).[\![Q]\!]^4 \triangleright \diamond}}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_2)\!)^4 \vdash \lambda z.\,z?(x_1).z?(x_2).[\![Q]\!]^4 \triangleright ((\!(C_1)\!)^4,(\!(C_2)\!)^4)\multimap\diamond} \tag{94}$$

$$\dfrac{\dfrac{}{(\!(\Gamma)\!)^{\mathsf{p}};\emptyset;(\!(\Delta_1)\!)^{\mathsf{p}} \cdot k : (\!(S)\!)^{\mathsf{p}} \vdash [\![P']\!]^{\mathsf{p}} \triangleright \diamond} \quad (94)}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_1)\!)^4 \cdot (\!(\Delta_2)\!)^4 \cdot n :!\langle (\!(L)\!)^4\rangle;(\!(S)\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond}$$

2. Case $P = n?(x_1,x_2).P'$ and $\Gamma;\emptyset;\Delta_1 \cdot n :?((C_1,C_2));S \vdash n?(x_1,x_2).P' \triangleright \diamond$. We have the following typing derivation:

$$\dfrac{\Gamma;\emptyset;\Delta_1 \cdot n : S \cdot x_1 : C_1 \cdot x_2 : C_2 \vdash P' \triangleright \diamond \quad \Gamma;\emptyset;\vdash x_1,x_2 \triangleright C_1,C_2}{\Gamma;\emptyset;\Delta_1 \cdot n :?((C_1,C_2));S \vdash n?(x_1,x_2).P' \triangleright \diamond}$$

By Definition 8.2 we have $[\![P]\!]^4 = n?(x_1).k?(x_2).[\![P']\!]^4$. By IH we have

$$(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_1)\!)^4 \cdot n : (\!(S)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot x_2 : (\!(C_2)\!)^4 \vdash [\![P']\!]^4 \triangleright \diamond$$

and the following type derivation:

$$\dfrac{\dfrac{\dfrac{}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_1)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot x_2 : (\!(C_2)\!)^4 \cdot n : (\!(S)\!)^4 \vdash [\![P']\!]^4 \triangleright \diamond}}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_1)\!)^4 \cdot x_1 : (\!(C_1)\!)^4 \cdot n :?((\!(C_2)\!)^4);(\!(S)\!)^4 \vdash n?(x_2).[\![P']\!]^4 \triangleright \diamond}}{(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_1)\!)^4 \cdot n :?((\!(C_1)\!)^4);?((\!(C_2)\!)^4);(\!(S)\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond}$$

□

We repeat the statement of Proposition 8.6, as in Page 43:

Proposition C.9 (Operational Correspondence. From $\mathsf{HO}\vec{\pi}$ to $\mathsf{HO}\pi$).

1. *Let $\Gamma;\emptyset;\Delta \vdash P$. Then $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ implies*
   a) *If $\ell = (\nu\,\tilde{m}')n!\langle\tilde{m}\rangle$ then $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell_1}{\longmapsto} \dots \stackrel{\ell_n}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P]\!]^4$ with $\{\!\{\ell\}\!\}^4 = \ell_1 \dots \ell_n$.*
   b) *If $\ell = n?\langle\tilde{m}\rangle$ then $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell_1}{\longmapsto} \dots \stackrel{\ell_n}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P]\!]^4$ with $\{\!\{\ell\}\!\}^4 = \ell_1 \dots \ell_n$.*
   c) *If $\ell \in \{(\nu\,\tilde{m})n!\langle\lambda\tilde{x}.R\rangle, n?\langle\lambda\tilde{x}.R\rangle\}$ then $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell'}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$ with $\{\!\{\ell\}\!\}^4 = \ell'$.*
   d) *If $\ell \in \{n \oplus l, n\&l\}$ then $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$.*
   e) *If $\ell = \tau_\beta$ then either $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\tau_\beta}{\longmapsto}\stackrel{\tau_{\mathsf{s}}}{\longmapsto} \dots \stackrel{\tau_{\mathsf{s}}}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$ with $\{\!\{\ell\}\!\} = \tau_\beta, \tau_{\mathsf{s}} \dots \tau_{\mathsf{s}}$.*
   f) *If $\ell = \tau$ then $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\tau}{\longmapsto} \dots \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^4 \vdash [\![P']\!]^4$ with $\{\!\{\ell\}\!\}^4 = \tau \dots \tau$.*
2. *Let $\Gamma;\emptyset;\Delta \vdash P$. $(\!(\Gamma)\!)^4;(\!(\Delta)\!)^4 \vdash [\![P]\!]^4 \stackrel{\ell_1}{\longmapsto} (\!(\Delta_1)\!)^4 \vdash P_1$ implies*
   a) *If $\ell \in \{n?\langle m\rangle, n!\langle m\rangle, (\nu\,m)n!\langle m\rangle\}$ then $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^4;(\!(\Delta_1)\!)^4 \vdash P_1 \stackrel{\ell_2}{\longmapsto} \dots \stackrel{\ell_n}{\longmapsto} (\!(\Delta')\!)^4 \vdash (\!(P')\!)^4$ with $\{\!\{\ell\}\!\}^4 = \ell_1 \dots \ell_n$.*
   b) *If $\ell \in \{(\nu\,\tilde{m})n!\langle\lambda x.R\rangle, n?\langle\lambda x.R\rangle\}$ then $\Gamma;\Delta \vdash P \stackrel{\ell'}{\longmapsto} \Delta' \vdash P'$ with $\{\!\{\ell'\}\!\}^4 = \ell$ and $P_1 \equiv [\![P']\!]^4$.*
   c) *If $\ell \in \{n \oplus l, n\&l\}$ then $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ and $P_1 \equiv [\![P']\!]^4$.*
   d) *If $\ell = \tau_\beta$ then $\Gamma;\Delta \vdash P \stackrel{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^4;(\!(\Delta_1)\!)^4 \vdash P_1 \stackrel{\tau_{\mathsf{s}}}{\longmapsto} \dots \stackrel{\tau_{\mathsf{s}}}{\longmapsto} (\!(\Delta')\!)^4 \vdash (\!(P')\!)^4$ with $\{\!\{\ell\}\!\}^4 = \tau_\beta, \tau_{\mathsf{s}} \dots \tau_{\mathsf{s}}$.*
   e) *If $\ell = \tau$ then $\Gamma;\Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ and $(\!(\Gamma)\!)^4;(\!(\Delta_1)\!)^4 \vdash P_1 \stackrel{\tau}{\longmapsto} \dots \stackrel{\tau}{\longmapsto} (\!(\Delta')\!)^4 \vdash (\!(P')\!)^4$ with $\{\!\{\ell\}\!\}^4 = \tau \dots \tau$.*

Proof. The proof of both parts is by transition induction, following the mapping defined in Definition 8.1. We consider some representative cases, using biadic communication:

- Case (1(a)), with $P = n!\langle m_1, m_2\rangle.P'$ and $\ell_1 = n!\langle m_1, m_2\rangle$. By assumption, P is well-typed. As one particular possibility, we may have:

$$\frac{\Gamma;\emptyset;\Delta_0 \cdot n : S \vdash P' \triangleright \diamond \quad \Gamma;\emptyset;m_1{:}S_1 \cdot m_2{:}S_2 \vdash m_1, m_2 \triangleright S_1, S_2}{\Gamma;\emptyset;\Delta_0 \cdot m_1{:}S_1 \cdot m_2{:}S_2 \cdot n : !\langle S_1, S_2\rangle;S \vdash n!\langle m_1, m_2\rangle.P' \triangleright \diamond}$$

  for some $\Gamma, S, S_1, S_2, \Delta_0$, such that $\Delta = \Delta_0 \cdot m_1{:}S_1 \cdot m_2{:}S_2 \cdot n : !\langle S_1, S_2\rangle;S$. We may then have the following typed transition

$$\Gamma;\Delta_0 \cdot m_1{:}S_1 \cdot m_2{:}S_2 \cdot n : !\langle S_1, S_2\rangle;S \vdash n!\langle m_1, m_2\rangle.P' \stackrel{\ell_1}{\longmapsto} \Gamma;\Delta_0 \cdot n{:}S \vdash P'$$

  The encoding of the source judgment for P is as follows:

$$(\!(\Gamma)\!)^4;\emptyset;(\!(\Delta_0 \cdot m_1{:}S_1 \cdot m_2{:}S_2 \cdot n : !\langle S_1, S_2\rangle;S)\!)^4 \vdash [\![n!\langle m_1, m_2\rangle.P']\!]^4 \triangleright \diamond$$

which, using Definition 8.1, can be expressed as

$$\langle\!\langle \Gamma \rangle\!\rangle^4; \emptyset; \langle\!\langle \Delta_0 \rangle\!\rangle \cdot m_1{:}\langle\!\langle S_1 \rangle\!\rangle^4 \cdot m_2{:}\langle\!\langle S_2 \rangle\!\rangle^4 \cdot n : !\langle \langle\!\langle S_1 \rangle\!\rangle^4 \rangle; !\langle \langle\!\langle S_2 \rangle\!\rangle^4 \rangle; \langle\!\langle S \rangle\!\rangle^4 \vdash n!\langle m_1 \rangle.n!\langle m_2 \rangle.[\![P']\!]^4 \triangleright \diamond$$

Now, $\{\!\{\ell_1\}\!\}^4 = n!\langle m_1 \rangle, n!\langle m_2 \rangle$. It is immediate to infer the following typed transitions for $[\![P]\!]^4 = n!\langle m_1 \rangle.n!\langle m_2 \rangle.[\![P']\!]^4$:

$$\begin{aligned}
& \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \rangle\!\rangle \cdot m_1{:}\langle\!\langle S_1 \rangle\!\rangle^4 \cdot m_2{:}\langle\!\langle S_2 \rangle\!\rangle^4 \cdot n : !\langle \langle\!\langle S_1 \rangle\!\rangle^4 \rangle; !\langle \langle\!\langle S_2 \rangle\!\rangle^4 \rangle; \langle\!\langle S \rangle\!\rangle^4 \vdash n!\langle m_1 \rangle.n!\langle m_2 \rangle.[\![P']\!]^4 \\
\xmapsto{n!\langle m_1 \rangle}\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \rangle\!\rangle \cdot m_2{:}\langle\!\langle S_2 \rangle\!\rangle^4 \cdot n : !\langle \langle\!\langle S_2 \rangle\!\rangle^4 \rangle; \langle\!\langle S \rangle\!\rangle^4 \vdash n!\langle m_2 \rangle.[\![P']\!]^4 \\
\xmapsto{n!\langle m_2 \rangle}\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \rangle\!\rangle \cdot n{:}\langle\!\langle S \rangle\!\rangle^4 \vdash [\![P']\!]^4 \\
=\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \cdot n : S \rangle\!\rangle^4 \vdash [\![P']\!]^4
\end{aligned}$$

which concludes the proof for this case.

- Case (1(c)) with $P = n!\langle \lambda(x_1, x_2).\, Q \rangle.P'$ and $\ell_1 = n!\langle \lambda(x_1, x_2).\, Q \rangle$. By assumption, P is well-typed. We may have:

$$\frac{\Gamma; \emptyset; \Delta_0 \cdot n : S \vdash P' \triangleright \diamond \qquad \Gamma; \emptyset; \Delta_1 \vdash \lambda(x_1, x_2).\, Q \triangleright (C_1, C_2) {\multimap} \diamond}{\Gamma; \emptyset; \Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) {\multimap} \diamond \rangle; S \vdash n!\langle \lambda(x_1, x_2).\, Q \rangle.P' \triangleright \diamond}$$

for some Γ, S, C_1, C_2, Δ_0, Δ_1, such that $\Delta = \Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) {\multimap} \diamond \rangle; S$. (For simplicity, we consider only the case of a linear function.) We may have the following typed transition:

$$\Gamma; \Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) {\multimap} \diamond \rangle; S \vdash n!\langle \lambda(x_1, x_2).\, Q \rangle.P' \xmapsto{\ell_1} \Gamma; \Delta_0 \cdot n{:}S \vdash P'$$

The encoding of the source judgment is

$$\langle\!\langle \Gamma \rangle\!\rangle^4; \emptyset; \langle\!\langle \Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) {\multimap} \diamond \rangle; S \rangle\!\rangle^4 \vdash [\![n!\langle \lambda(x_1, x_2).\, Q \rangle.P']\!]^4 \triangleright \diamond$$

which, using Definition 8.1, can be equivalently expressed as

$$\langle\!\langle \Gamma \rangle\!\rangle^4; \emptyset; \langle\!\langle \Delta_0 \cdot \Delta_1 \rangle\!\rangle \cdot n : !\langle (?(\langle\!\langle C_1 \rangle\!\rangle^4); ?(\langle\!\langle C_2 \rangle\!\rangle^4); \mathsf{end}) {\multimap} \diamond \rangle; \langle\!\langle S \rangle\!\rangle^4 \vdash n!\langle \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4 \rangle.[\![P']\!]^4 \triangleright \diamond$$

Now, $\{\!\{\ell_1\}\!\}^4 = n!\langle \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4 \rangle$. It is immediate to infer the following typed transition for $[\![P]\!]^4 = n!\langle \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4 \rangle.[\![P']\!]^4$:

$$\begin{aligned}
& \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \cdot \Delta_1 \rangle\!\rangle \cdot n : !\langle (?(\langle\!\langle C_1 \rangle\!\rangle^4); ?(\langle\!\langle C_2 \rangle\!\rangle^4); \mathsf{end}) {\multimap} \diamond \rangle; \langle\!\langle S \rangle\!\rangle^4 \vdash n!\langle \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4 \rangle.[\![P']\!]^4 \\
\xmapsto{\{\!\{\ell_1\}\!\}^4}\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \rangle\!\rangle \cdot n : \langle\!\langle S \rangle\!\rangle^4, \vdash [\![P']\!]^4 \\
=\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta_0 \cdot n : S \rangle\!\rangle^4 \vdash [\![P']\!]^4
\end{aligned}$$

which concludes the proof for this case.

- Case (2(a)), with $P = n?(x_1, x_2).P'$, $[\![P]\!]^4 = n?(x_1).n?(x_2).[\![P']\!]^4$. We have the following typed transitions for $[\![P]\!]^4$, for some S, S_1, S_2, and Δ:

$$\begin{aligned}
& \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta \rangle\!\rangle^4 \cdot n : ?(\langle\!\langle S_1 \rangle\!\rangle^4); ?(\langle\!\langle S_2 \rangle\!\rangle^4); \langle\!\langle S \rangle\!\rangle^4 \cdot \vdash n?(x_1).n?(x_2).[\![P']\!]^4 \\
\xmapsto{n?\langle m_1 \rangle}\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta \rangle\!\rangle^4 \cdot n : ?(\langle\!\langle S_2 \rangle\!\rangle^4); \langle\!\langle S \rangle\!\rangle^4 \cdot m_1 : \langle\!\langle S_1 \rangle\!\rangle^4 \vdash n?(x_2).[\![P']\!]^4\{m_1/x_1\} \\
\xmapsto{n?\langle m_2 \rangle}\ & \langle\!\langle \Gamma \rangle\!\rangle^4; \langle\!\langle \Delta \rangle\!\rangle^4 \cdot n : \langle\!\langle S \rangle\!\rangle^4 \cdot m_1 : \langle\!\langle S_1 \rangle\!\rangle^4 \cdot m_2 : \langle\!\langle S_2 \rangle\!\rangle^4 \vdash [\![P']\!]^4\{m_1/x_1\}\{m_2/x_2\} = Q
\end{aligned}$$

Observe that the substitution lemma (Lemma 3.1(1)) has been used twice. It is then immediate to infer the label for the source transition: $\ell_1 = n?\langle m_1, m_2\rangle$. Indeed, $\{\!\{\ell_1\}\!\}^4 = n?\langle m_1\rangle, n?\langle m_2\rangle$. Now, in the source term P we can infer the following transition:

$$\Gamma;\Delta \cdot n : ?(S_1, S_2); S \vdash n?(x_1, x_2).P' \xmapsto{\ell_1} \Gamma;\Delta \cdot n{:}S \cdot m_1 : S_1 \cdot m_2 : S_2 \vdash P'\{m_1,m_2/x_1,x_2\}$$

which concludes the proof for this case.

- Case (2(b)), with $P = n!\langle\lambda(x_1, x_2).\, Q\rangle.P'$, $[\![P]\!]^4 = n!\langle\lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4$. We have the following typed transition, for some S, C_1, C_2, and Δ:

$$\begin{aligned} &(\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \cdot n : (\!(!\langle(C_1, C_2){\multimap}\diamond\rangle; S)\!)^4 \vdash n!\langle\lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4 \\ \xmapsto{\ell_1'}\ & (\!(\Gamma)\!)^4; (\!(\Delta)\!)^4 \cdot n : (\!(S)\!)^4 \vdash [\![P']\!]^4 = Q \end{aligned}$$

where $\ell_1' = n!\langle\lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4\rangle$. For simplicity, we consider only the case of linear functions. It is then immediate to infer the label for the source transition: $\ell_1 = n!\langle\lambda(x_1, x_2).\, Q\rangle$. Now, in the source term P we can infer the following transition:

$$\Gamma;\Delta \cdot n : !\langle(C_1, C_2){\multimap}\diamond\rangle; S \vdash n!\langle\lambda x_1, x_2.\, Q\rangle.P' \xmapsto{\ell_1} \Gamma;\Delta \cdot n{:}S \vdash P'$$

which concludes the proof for this case.

$\square$