On the Relative Expressiveness of Higher-Order Session Processes

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

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

Abstract. By integrating constructs from the λ-calculus and the π-calculus, in *higher-order process calculi* exchanged values may contain processes. This paper studies the relative expressiveness of $\mathsf{HO}\pi$, the higher-order π-calculus in which communications are governed by *session types*. Our main discovery is that HO, a subcalculus of $\mathsf{HO}\pi$ which lacks name-passing and recursion, can serve as a new core calculus for session-typed higher-order concurrency. By exploring a new bisimulation for HO, we show that HO can encode $\mathsf{HO}\pi$ fully abstractly (up to typed contextual congruence) more precisely and efficiently than the first-order session π-calculus (π). Overall, under session types, $\mathsf{HO}\pi$, HO, and π are equally expressive; but $\mathsf{HO}\pi$ and HO are more tightly related than $\mathsf{HO}\pi$ and π.

1 Introduction

Type-preserving compilations are important in the design of functional and object-oriented languages: type information has been used to, e.g., justify code optimizations and reason about programs (see, e.g., [24,38,20]). A vast literature on *expressiveness* in concurrency theory (e.g., [28,10,8,18,31]) also studies compilations (or *encodings*): they are used to transfer reasoning techniques from one calculus to another, and to identify constructs which may be implemented using simpler ones. In this work, we study *relative expressiveness* via *type-preserving encodings* for $\mathsf{HO}\pi$, a *higher-order* process language that integrates message-passing concurrency with functional features. We consider source and target calculi coupled with *session types* denoting interaction protocols. Building upon untyped frameworks for relative expressiveness [10], we propose type preservation as a new criteria for *precise encodings*. We identify HO, a new core calculus for higher-order session concurrency without name passing. We show that HO can encode $\mathsf{HO}\pi$ precisely and efficiently. Requiring type preservation makes this encoding far from trivial: our encoding crucially exploits advances on session type duality [2,3] and recent characterisations of typed contextual equivalence [14]. We develop a full hierarchy of variants of $\mathsf{HO}\pi$ based on precise encodings (see Fig. 1): our encodings are type-preserving and fully abstract, up to typed behavioural equalities.

Context In *session-based concurrency*, interactions are organised into *sessions*, basic communication units. Interaction patterns can then be abstracted as expressive *session types* [11], against which specifications may be checked. Session type $?(U);S$ (resp. $!\langle U\rangle;S$) describes a protocol that first receives (resp. sends) a value of type U and then continues as protocol S. Also, given an index set I, types $\&\{l_i : S_i\}_{i\in I}$ and $\oplus\{l_i : S_i\}_{i\in I}$

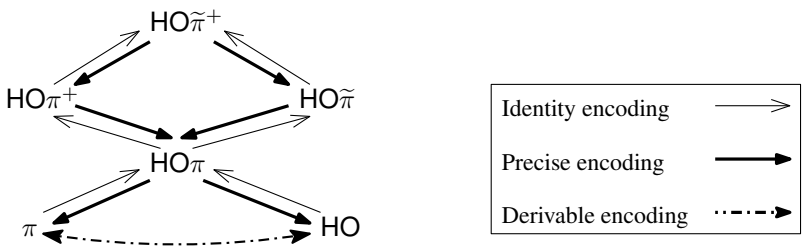

Fig. 1: Encodability in Higher-Order Sessions. Precise encodings are defined in Def. 15.

define a labeled choice mechanism; types μt.S and end denote recursive and completed protocols, respectively. In the (first-order) π-calculus [22], session types describe the intended interactive behaviour of the names/channels in a process.

Session-based concurrency has also been casted in higher-order process calculi which, by combining features from the λ-calculus and the π-calculus, enable the exchange of values that may contain processes [25,9]. The higher-order calculus with sessions studied here, denoted $\mathsf{HO}\pi$, can specify protocols involving *code mobility*: it includes constructs for synchronisation along shared names, session communication (value passing, labelled choice) along linear names, recursion, (first-order) abstractions and applications. That is, values in communications include names but also (first-order) abstractions—functions from name identifiers to processes. (In contrast, we rule out higher-order abstractions—functions from processes to processes.) Abstractions can be linear or shared; their types are denoted $C\multimap\diamond$ and $C\to\diamond$, respectively (C denotes a name). In $\mathsf{HO}\pi$ we may have processes with a session type such as, e.g.,

$$S = \&\{up : ?(C\multimap\diamond); !\langle\mathsf{ok}\rangle; \mathsf{end}\ ,\ down : !\langle C\to\diamond\rangle; !\langle\mathsf{ok}\rangle; \mathsf{end}\ ,\ quit : !\langle\mathsf{bye}\rangle; \mathsf{end}\}$$

that abstracts a server that offers different behaviours to clients: to *upload* a linear function, to *download* a shared function, or to *quit* the protocol. Subsequently, the server sends a message (ok or bye) before closing the session.

Expressiveness of $\mathsf{HO}\pi$ We study the type-preserving, relative expressivity of $\mathsf{HO}\pi$. As expected from known literature in the untyped setting [32], the first-order session π-calculus [11] (here denoted π) can encode $\mathsf{HO}\pi$ preserving session types. In this paper, our *main discovery* is that $\mathsf{HO}\pi$ without name-passing and recursion can serve as a new core calculus for higher-order session concurrency. We call this core calculus HO. We show that HO can encode $\mathsf{HO}\pi$ more efficiently than π. In addition, in the higher-order session typed setting, HO offers more tractable bisimulation techniques than π (cf. § 5.2).

Challenges and Contributions We assess the expressivity of $\mathsf{HO}\pi$, HO, and π as delineated by session types. We introduce *type-preserving encodings*: we use type information to define encodings and to retain the semantics of session protocols. Indeed, not only we require well-typed source processes are encoded into well-typed target processes: we demand that session type constructs (input, output, branching, select) used

to type the source process are preserved by the typing of the target process. This criterion is included in our notion of *precise encoding* (Def. 15), which extends encodability criteria for untyped processes with *full abstraction*. Full abstraction results are stated up to two behavioural equalities that characterise barbed congruence: *characteristic bisimilarity* ($\approx^{\mathsf{C}}$, defined in [14]) and *higher-order bisimilarity* ($\approx^{\mathsf{H}}$), introduced in this work. It turns out that $\approx^{\mathsf{H}}$ offers more direct reasoning than $\approx^{\mathsf{C}}$. Using precise encodings we establish strong correspondences between $\mathsf{HO}\pi$ and its variants—see Fig. 1.

Our main contribution is an encoding of $\mathsf{HO}\pi$ into HO (§ 7.1). Since HO lacks both name-passing and recursion, this encoding involves two *key challenges*:

a. In known (typed) encodings of name-passing into process-passing [36] only the output capability of names can be sent—a received name cannot be used in later inputs. This is far too limiting in $\mathsf{HO}\pi$, where session names may be passed around (*delegation*) and types describe interaction *structures*, rather than "loose" name capabilities.
b. Known encodings of recursion in untyped higher-order calculi do not carry over to session typed calculi such as $\mathsf{HO}\pi$, because linear abstractions cannot be copied/duplicated. Hence, the discipline of session types limits the possibilities for representing infinite behaviours—even simple forms, such as input-guarded replication.

Our encoding overcomes these two obstacles, as we discuss in the following section.

Additional technical contributions include: (i) the encodability of HO into π (§ 7.2); (ii) extensions of our encodability results to richer settings (§ 8); (iii) a non encodability result showing that shared names strictly add expressive power to session calculi (§ 7.4). In essence, (i) extends known results for untyped processes [32] to the session typed setting. Concerning (ii), we develop extensions of our encodings to

- The extension of $\mathsf{HO}\pi$ with *higher-order* abstractions ($\mathsf{HO}\pi^{+}$);
- The extension of $\mathsf{HO}\pi$ with polyadic name passing and abstraction ($\mathsf{HO}\widetilde{\pi}$);
- The super-calculus of $\mathsf{HO}\pi^{+}$ and $\mathsf{HO}\widetilde{\pi}$ ($\mathsf{HO}\widetilde{\pi}^{+}$), equivalent to the calculus in [25].

These encodability results connect $\mathsf{HO}\pi$ with existing higher-order process calculi [25], and further highlight the status of HO as the core calculus for session concurrency. Finally, although (iii) may be somewhat expected, to our knowledge we are the first to prove this separation result, exploiting session determinacy and typed equivalences.

Outline § 2 overviews key ideas of the precise encoding of $\mathsf{HO}\pi$ into π. § 3 presents $\mathsf{HO}\pi$ and its subcalculi (HO and π); § 4 summarises their session type system. § 5 presents behavioural equalities for $\mathsf{HO}\pi$: we recall definitions of barbed congruence and characteristic bisimilarity [14], and introduce higher-order bisimilarity. We show that these three typed relations coincide (Thm. 2). § 6 defines *precise encodings* by extending encodability criteria for untyped processes. § 7 gives precise encodings of $\mathsf{HO}\pi$ into HO and of $\mathsf{HO}\pi$ into π (Thms. 3 and 4). Mutual encodings between π and HO are derivable; all these calculi are thus equally expressive. By means of empirical and formal comparisons between these two precise encodings, in § 7.3 we establish that $\mathsf{HO}\pi$ and HO are more tightly related than $\mathsf{HO}\pi$ and π (Thm. 5). Moreover, we prove the impossibility of encoding communication along shared names using linear names (Thm. 6). In § 8 we show that both $\mathsf{HO}\pi^{+}$ and $\mathsf{HO}\widetilde{\pi}$ are encodable in $\mathsf{HO}\pi$ (Thms. 7 and 8). § 9 collects concluding remarks and reviews related works. The paper is self-contained. ***Omitted definitions and proofs are in the Appendix and in [15].***

2 Overview: Encoding Name Passing Into Process Passing

A Precise Encoding of Name-Passing into Process-Passing As mentioned above, our encoding of HOπ into HO (§ 7.1) should (a) enable the communication of arbitrary names, as required to represent delegation, and (b) address the fact that linearity of session types limits the possibilities for representing infinite behaviour. To encode name passing into HO we "pack" the name to be sent into a suitable abstraction; upon reception, the receiver "unpacks" this object following a precise protocol on a fresh session:

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

Above, a, b are names and s and $\overline{s}$ are linear session names (*endpoints*). Processes $a!\langle V\rangle.P$ and $a?(x).P$ denote output and input at a; abstractions and applications are denoted $\lambda x.P$ and $(\lambda x.P)a$. Processes $(\nu\, s)(P)$ and $\mathbf{0}$ represent hiding and inaction. Thus, following a communication on a, a (deterministic) reduction between s and $\overline{s}$ guarantees that b is properly unpacked by means of abstraction passing and appropriate applications. Observe that HO requires two extra reduction steps to mimic a name communication step in HOπ. Also, observe how an output action in the source process is translated into an output action in the encoded process (and similarly for input). This is key to ensure the preservation of session type operators mentioned above (cf. Def. 13).

To preserve session linearity, we proceed as follows. Given $\mu X.P$, we encode the recursion body P as an abstraction in which free names of P are converted into name variables. The resulting higher-order value is embedded in an input-guarded "duplicator" process [40]. The recursion variable X is then encoded in such a way that it simulates recursion unfolding by invoking the duplicator in a by-need fashion. That is, upon reception, the abstraction representing the recursion body P is duplicated: one copy is used to reconstitute the original recursion body P (through the application of the free names of P); another copy is used to re-invoke the duplicator when needed. Interestingly, for this encoding to work we require non-tail recursive session types; to this end, we apply recent advances on the theory of duality for session types [2,3].

A Plausible Encoding That is Not Precise Our notion of *precise encoding* (Def. 15) requires the translation of both process and types, and admits only process mappings that preserve session types *and* are fully abstract. Thus, our encodings not only exhibit strong behavioural correspondences, but also relate source and target processes with communication structures described by session types. These strict requirements make our developments far from trivial. In particular, requiring type preservation rules out other plausible encoding strategies. To illustrate this point, consider the following encoding of name-passing into HO:[1]

$$\begin{aligned}[\![a!\langle b\rangle.P]\!]^u &= a?(x).(x\,b \mid [\![P]\!]^u)\\ [\![a?(x).Q]\!]^u &= a!\langle \lambda x.\,[\![Q]\!]^u\rangle.\mathbf{0}\end{aligned}$$

Intuitively, the encoding of input takes the initiative by sending an abstraction containing the encoding of its continuation Q; the encoding of output applies this received value

[1] This alternative encoding was suggested by an anonymous reviewer of a previous version of this paper.

$$u,w ::= n \mid x,y,z \qquad n ::= a,b \mid s,\overline{s} \qquad V,W ::= u \mid \boxed{\lambda x.P}$$
$$P,Q ::= u!\langle V\rangle.P \mid u?(x).P \mid u \triangleleft l.P \mid u \triangleright \{l_i : P_i\}_{i\in I} \mid \boxed{V\,u} \mid P \mid Q \mid (\nu n)P \mid \mathbf{0} \mid X \mid \mu X.P$$

Fig. 2: Syntax of HOπ. While HO lacks shaded constructs, π lacks boxed constructs.

to name b. Hence, this mapping entails a "role inversion": outputs are translated into inputs, and inputs are translated into outputs. Although fairly reasonable, we will see that the encoding $[\![\cdot]\!]^u$ is *not type preserving*. Consequently, it is also not *precise*. Since individual prefixes (input, output, branching, select) represent actions in a structured communication sequence (i.e., a protocol abstracted by a session type), the encoding above would simply alter the meaning of the session protocol in the source language.

3 Higher-Order Session π-Calculi

We introduce the *higher-order session π-calculus* (HOπ). We define syntax, operational semantics, and its sub-calculi (π and HO). A type system and behavioural equivalences are introduced in § 4 and § 5. Extensions of HOπ are discussed in § 8.

3.1 HOπ: Syntax, Operational Semantics, Subcalculi

Syntax The syntax of HOπ is defined in Fig. 2. HOπ it is a subcalculus of the language studied in [25]. It is also a variant of the language that we investigated in [14], where higher-order value applications were considered.

Names $a,b,c,\ldots$ (resp. $s,\overline{s},\ldots$) range over shared (resp. session) names. Names $m,n,t,\ldots$ are session or shared names. Dual endpoints are $\overline{n}$ with $\overline{\overline{s}} = s$ and $\overline{a} = a$. Variables are denoted with $x,y,z,\ldots$, and recursive variables are denoted with $X,Y\ldots$. An abstraction $\lambda x.P$ is a process P with name parameter x. *Values* V,W include identifiers $u,v,\ldots$ and abstractions $\lambda x.P$ (first- and higher-order values, resp.).

Terms include π-calculus prefixes for sending and receiving values V. Recursion $\mu X.P$ binds the recursive variable X in process P. Process $V\,u$ is the application which substitutes name u on the abstraction V. Typing ensures that V is not a name. Processes $u \triangleleft l.P$ and $u \triangleright \{l_i : P_i\}_{i\in I}$ are the standard session processes for selecting and branching. Constructs for inaction $\mathbf{0}$, parallel composition $P_1 \mid P_2$, and name restriction $(\nu n)P$ are standard. Session name restriction $(\nu s)P$ simultaneously binds endpoints s and $\overline{s}$ in P. Functions $\mathtt{fv}(P)$ and $\mathtt{fn}(P)$ denote the sets of free variables and names. We assume V in $u!\langle V\rangle.P$ does not include free recursive variables X. If $\mathtt{fv}(P) = \emptyset$, we call P *closed*.

Operational Semantics The *operational semantics* of HOπ is defined in terms of a reduction relation, denoted $\longrightarrow$ and given in Fig. 3 (top). We briefly explain the rules. Rule [App] defines name application. Rule [Pass] defines a shared interaction at n (with $\overline{n} = n$) or a session interaction. Rule [Sel] is the standard rule for labelled choice/selection.Other rules are standard π-calculus rules. Reduction is closed under *structural*

$$(\lambda x.P)\,u \longrightarrow P\{u/x\} \quad [\text{App}] \qquad n!\langle V\rangle.P \mid \overline{n}?(x).Q \longrightarrow P \mid Q\{V/x\} \quad [\text{Pass}]$$
$$n \triangleleft l_j.Q \mid \overline{n} \triangleright \{l_i : P_i\}_{i\in I} \longrightarrow Q \mid P_j \ \ (j \in I) \quad [\text{Sel}] \qquad P \longrightarrow P' \Rightarrow (\nu n)P \longrightarrow (\nu n)P' \quad [\text{Res}]$$
$$P \longrightarrow P' \ \Rightarrow\ P \mid Q \longrightarrow P' \mid Q \quad [\text{Par}] \qquad P \equiv Q \longrightarrow Q' \equiv P' \ \Rightarrow\ P \longrightarrow P' \quad [\text{Cong}]$$

$$P \mid \mathbf{0} \equiv P \quad P_1 \mid P_2 \equiv P_2 \mid P_1 \quad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3 \quad (\nu n)\mathbf{0} \equiv \mathbf{0}$$
$$P \mid (\nu n)Q \equiv (\nu n)(P \mid Q)\ (n \notin \mathtt{fn}(P)) \quad \mu X.P \equiv P\{\mu X.P/X\} \quad P \equiv Q \text{ if } P \equiv_\alpha Q$$

Fig. 3: Operational Semantics of HOπ.

congruence as defined in Fig. 3 (bottom). We assume the expected extension of $\equiv$ to values V. We write $\longrightarrow^*$ for a multi-step reduction.

Subcalculi As motivated in the introduction, we define two *subcalculi* of HOπ.

- The *core higher-order session calculus* (denoted HO), lacks recursion and name passing; its formal syntax is obtained from Fig. 2 by excluding constructs in grey.
- The *session π-calculus* (denoted π), which lacks higher-order constructs (i.e., abstraction passing and application), but includes recursion.

Let $\mathsf{C} \in \{\text{HO}\pi, \text{HO}, \pi\}$. We write $\mathsf{C}^{-\mathsf{sh}}$ for C without shared names (we delete a, b from n). We shall demonstrate in § 7 that HOπ, HO, and π have the same expressivity.

4 Session Types for HOπ

We define a session type system for HOπ and state *type soundness* (Thm. 1), its main property. Our system distills the key features of [25,26] and so it is simpler.

The syntax of types of HOπ follows. We write $\diamond$ to denote the process type.

$$U ::= C \mid L \qquad C ::= S \mid \langle S\rangle \mid \langle L\rangle \qquad L ::= C{\to}\diamond \mid C{\multimap}\diamond$$
$$S ::= !\langle U\rangle;S \mid ?(U);S \mid \oplus\{l_i : S_i\}_{i\in I} \mid \&\{l_i : S_i\}_{i\in I} \mid \mu \mathsf{t}.S \mid \mathsf{t} \mid \mathsf{end}$$

Value type U includes first-order types C and higher-order types L. Types $C{\to}\diamond$ and $C{\multimap}\diamond$ denote *shared* and *linear* higher-order types, respectively. Session types, denoted by S, follow the standard binary session type syntax [11], with the extension that carried types U may be higher-order. Shared channel types are denoted $\langle S\rangle$ and $\langle L\rangle$. Types of HO exclude C from value types of HOπ; the types of π exclude L. From each $\mathsf{C} \in \{\text{HO}\pi, \text{HO}, \pi\}$, $\mathsf{C}^{-\mathsf{sh}}$ excludes shared name types ($\langle S\rangle$ and $\langle L\rangle$), from name type C.

We use the co-inductive definition of *duality* of [2]. We write S_1 dual S_2 if S_1 is the dual of S_2. Intuitively, session type duality is obtained by dualising ! by ?, ? by !, $\oplus$ by &, and & by $\oplus$, including the fixed point construction (see Def. 21 in the Appendix).

We consider environments denoted Γ, Λ, and Δ:

$$\Lambda ::= \emptyset \mid \Lambda \cdot x : C{\multimap}\diamond \qquad \Delta ::= \emptyset \mid \Delta \cdot u : S$$
$$\Gamma ::= \emptyset \mid \Gamma \cdot x : C{\to}\diamond \mid \Gamma \cdot u : \langle S\rangle \mid \Gamma \cdot u : \langle L\rangle \mid \Gamma \cdot X : \Delta$$

$$\text{(Prom)}\ \frac{\Gamma;\emptyset;\emptyset \vdash V \triangleright C \multimap \diamond}{\Gamma;\emptyset;\emptyset \vdash V \triangleright C \to \diamond} \qquad \text{(EProm)}\ \frac{\Gamma;\Lambda \cdot x : C \multimap \diamond;\Delta \vdash P \triangleright \diamond}{\Gamma \cdot x : C \to \diamond;\Lambda;\Delta \vdash P \triangleright \diamond} \qquad \text{(Abs)}\ \frac{\Gamma;\Lambda;\Delta_1 \vdash P \triangleright \diamond \quad \Gamma;\emptyset;\Delta_2 \vdash x \triangleright C}{\Gamma \backslash x;\Lambda;\Delta_1 \backslash \Delta_2 \vdash \lambda x.P \triangleright C \multimap \diamond}$$

$$\text{(App)}\ \frac{U = C \multimap \diamond \vee C \to \diamond \quad \Gamma;\Lambda;\Delta_1 \vdash V \triangleright U \quad \Gamma;\emptyset;\Delta_2 \vdash u \triangleright C}{\Gamma;\Lambda;\Delta_1 \cdot \Delta_2 \vdash V\,u \triangleright \diamond} \qquad \text{(Send)}\ \frac{u : S \in \Delta_1 \cdot \Delta_2 \quad \Gamma;\Lambda_1;\Delta_1 \vdash P \triangleright \diamond \quad \Gamma;\Lambda_2;\Delta_2 \vdash V \triangleright U}{\Gamma;\Lambda_1 \cdot \Lambda_2;((\Delta_1 \cdot \Delta_2) \setminus u : S) \cdot u : !\langle U\rangle;S \vdash u!\langle V\rangle.P \triangleright \diamond}$$

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

$$\text{(Req)}\ \frac{\Gamma;\emptyset;\emptyset \vdash u \triangleright U_1 \quad \Gamma;\Lambda;\Delta_1 \vdash P \triangleright \diamond \quad \Gamma;\emptyset;\Delta_2 \vdash V \triangleright U_2 \quad (U_1 = \langle S\rangle \wedge U_2 = S) \vee (U_1 = \langle L\rangle \wedge U_2 = L)}{\Gamma;\Lambda;\Delta_1 \cdot \Delta_2 \vdash u!\langle V\rangle.P \triangleright \diamond} \qquad \text{(Acc)}\ \frac{\Gamma;\emptyset;\emptyset \vdash u \triangleright U_1 \quad \Gamma;\Lambda_1;\Delta_1 \vdash P \triangleright \diamond \quad \Gamma;\Lambda_2;\Delta_2 \vdash x \triangleright U_2 \quad (U_1 = \langle S\rangle \wedge U_2 = S) \vee (U_1 = \langle L\rangle \wedge U_2 = L)}{\Gamma \backslash x;\Lambda_1 \backslash \Lambda_2;\Delta_1 \backslash \Delta_2 \vdash u?(x).P \triangleright \diamond}$$

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

Γ maps variables and shared names to value types, and recursive variables to session environments; it admits weakening, contraction, and exchange principles. Λ is a mapping from variables to linear higher-order types; and Δ is a mapping from session names to session types. Both Λ and Δ are only subject to exchange. We require that the domains of Γ, Λ and Δ are pairwise distinct. $\Delta_1 \cdot \Delta_2$ denotes the disjoint union of Δ_1 and Δ_2. We are interested in *balanced* session environments:

Definition 1 (Balanced). *We say that a session environment Δ is* balanced *if whenever $s : S_1, \overline{s} : S_2 \in \Delta$ then S_1 dual S_2.*

Given the above intuitions for environments, the typing judgements for values V and processes P are self-explanatory. They are denoted $\Gamma;\Lambda;\Delta \vdash V \triangleright U$ and $\Gamma;\Lambda;\Delta \vdash P \triangleright \diamond$.

Selected typing rules are given in Fig. 4; see App. A.2 for a full account. The shared type $C \to \diamond$ is derived using rule (Prom) only if the value has a linear type with an empty linear environment. Rule (EProm) allows us to freely use a linear type variable as shared. Abstraction values are typed with rule (Abs). Application typing is governed by rule (Abs): we expect the type C of an application name u to match the type $C \multimap \diamond$ or $C \to \diamond$ of the application variable x. In rule (Send), the type U of a send value V should appear as a prefix on the session type $!\langle U\rangle;S$ of u. Rule (Rcv) is its dual. We use a similar approach with session prefixes to type interaction between shared names as defined in rules (Req) and (Acc), where the type of the sent/received object (S and L, resp.) should match the type of the sent/received subject ($\langle S\rangle$ and $\langle L\rangle$, resp.).

Definition 2. *We define the relation $\longrightarrow$ on session environments as:*

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

We state type soundness for HOπ; it implies type soundness for HO, π, and $\mathsf{C}^{-\mathsf{sh}}$.

Theorem 1 (Type Soundness). Suppose $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ with Δ balanced. Then $P \longrightarrow P'$ implies $\Gamma;\emptyset;\Delta' \vdash P' \triangleright \diamond$ and $\Delta = \Delta'$ or $\Delta \longrightarrow \Delta'$ with Δ' balanced.

5 Behavioural Theory for HOπ

We first define reduction-closed, barbed congruence ($\cong$, Def. 7) as the reference equivalence relation for HOπ processes. We then define two characterizations of $\cong$: *characteristic* and *higher-order bisimilarities* (denoted $\approx^{\mathsf{C}}$ and $\approx^{\mathsf{H}}$, cf. Defs. 8 and 9).

5.1 Reduction-Closed, Barbed Congruence ($\cong$)

We consider *typed relations* that relate closed terms whose session environments are balanced and confluent:

Definition 3 (Session Environment Confluence). *Let $\longrightarrow^*$ denote multi-step reduction as in Def. 2. We denote $\Delta_1 \rightleftharpoons \Delta_2$ if there exists Δ such that $\Delta_1 \longrightarrow^* \Delta$ and $\Delta_2 \longrightarrow^* \Delta$.*

Definition 4 (Typed Relation). *We say that $\Gamma;\emptyset;\Delta_1 \vdash P_1 \triangleright \diamond \;\mathfrak{R}\; \Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond$ is a* typed relation *whenever P_1 and P_2 are closed; Δ_1 and Δ_2 are balanced; and $\Delta_1 \rightleftharpoons \Delta_2$. We write $\Gamma;\Delta_1 \vdash P_1 \;\mathfrak{R}\; \Delta_2 \vdash P_2$ for the typed relation $\Gamma;\emptyset;\Delta_1 \vdash P_1 \triangleright \diamond \;\mathfrak{R}\; \Gamma;\emptyset;\Delta_2 \vdash P_2 \triangleright \diamond$.*

As usual, a *barb* $\downarrow_n$ is an observable on an output prefix with subject n [23]. A *weak barb* $\Downarrow_n$ is a barb after zero or more reduction steps. Typed barbs $\downarrow_n$ (resp. $\Downarrow_n$) occur on typed processes $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. When n is a session name we require that its dual endpoint $\overline{n}$ is not present in the session environment Δ:

Definition 5 (Barbs). Let P be a closed process. We define:

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

To define a congruence relation, we introduce the family $\mathbb{C}$ of contexts:

Definition 6 (Context). *A context $\mathbb{C}$ is defined as:*

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

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

We define reduction-closed, barbed congruence [12].

Definition 7 (Barbed Congruence). *Typed relation $\Gamma;\Delta_1 \vdash P \;\mathfrak{R}\; \Delta_2 \vdash Q$ is a* reduction-closed, barbed congruence *whenever:*

1) If $P \longrightarrow P'$ then there exist Q', Δ_1', Δ_2' such that $Q \longrightarrow^* Q'$ and $\Gamma;\Delta_1' \vdash P' \;\mathfrak{R}\; \Delta_2' \vdash Q'$;
2) If $\Gamma;\Delta_1 \vdash P \downarrow_n$ then $\Gamma;\Delta_2 \vdash Q \Downarrow_n$;
3) For all $\mathbb{C}$, there exist Δ_1'', Δ_2'' such that $\Gamma;\Delta_1'' \vdash \mathbb{C}[P] \;\mathfrak{R}\; \Delta_2'' \vdash \mathbb{C}[Q]$;
4) The symmetric cases of 1 and 2.

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

5.2 Two Equivalence Relations: $\approx^{\mathsf{C}}$ and $\approx^{\mathsf{H}}$

A Typed Labelled Transition System In [14] we have characterised reduction-closed, barbed congruence for HOπ via a typed relation called *characteristic bisimilarity*. Its definition uses a *typed* labelled transition system (LTS) informed by session types. Transitions in this LTS are denoted $\Gamma;\emptyset;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P' \triangleright \Delta'$. (Weak transitions, defined as expected, are denoted $\Gamma;\emptyset;\Delta \vdash P \overset{\ell}{\Longmapsto} \Delta' \vdash P' \triangleright \Delta'$.) The main intuition is that the transitions of a typed process should be enabled by its associated typing environment:

$$\text{if } P \overset{\ell}{\longmapsto} P' \text{ and } (\Gamma,\Delta) \overset{\ell}{\longmapsto} (\Gamma,\Delta') \text{ then } \Gamma;\emptyset;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P' \triangleright \Delta'$$

As an example of how types enable transitions, consider the rule for input:

$$\frac{\overline{s} \notin \mathrm{dom}(\Delta) \qquad \Gamma;\Lambda';\Delta' \vdash V \triangleright U \qquad V = m \vee V \equiv (\!|U|\!)_{\mathsf{c}} \vee V \equiv \lambda x.t?(y).(y\,x) \text{ with } t \text{ fresh}}{(\Gamma;\Lambda;\Delta \cdot s:?(U);S) \overset{s?\langle V\rangle}{\longmapsto} (\Gamma;\Lambda\cdot\Lambda';\Delta\cdot\Delta'\cdot s:S)}$$

This rule states that a session channel environment can input a value if the channel is typed with an input prefix and the input value is either a name m, a *characteristic value* $(\!|U|\!)_{\mathsf{c}}$, or a *trigger value* (the abstraction $\lambda x.t?(y).(y\,x)$). A characteristic value is the simplest process that inhabits a type (here, the type U carried by the input prefix). The above rule is used to limit the input actions that can be observed from a session input prefix. For details of the labelled transition system and the characteristic process definition see App. B and [14]. Moreover, we define a *(first-order) trigger process*:

$$t \Leftarrow V:U \overset{\mathtt{def}}{=} t?(x).(\nu\, s)([\![?(U);\mathsf{end}]\!]^{s} \mid \overline{s}!\langle V\rangle.\mathbf{0}) \tag{1}$$

The trigger process $t \Leftarrow V:U$ is is defined as a process input prefixed on a fresh name t: it applies a value on the *characteristic process* $[\![?(U);\mathsf{end}]\!]^{s}$ (see [14] for details).

Characterisations of $\cong$ We now define *characteristic* and *higher-order* bisimilarities. Observe that higher-order bisimilarity is a new typed equality, while characteristic bisimilarity was introduced in [14] (Def. 14).

Definition 8 (Characteristic Bisimilarity). *A typed relation* $\mathfrak{R}$ *is a* characteristic bisimulation *if for all* $\Gamma;\Delta_1 \vdash P_1\ \mathfrak{R}\ \Delta_2 \vdash Q_1$

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, 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 fresh* t,

$$\Gamma;\Delta_1'' \vdash (\nu\tilde{m}_1)(P_2 \mid t \Leftarrow V_1:U_1)\ \mathfrak{R}\ \Delta_2'' \vdash (\nu\tilde{m}_2)(Q_2 \mid t \Leftarrow V_2:U_2)$$

2) *For all* $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$ *such that* ℓ *is not an output, there exist* Q_2, Δ_2' *such that* $\Gamma;\Delta_2 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$ *and* $\Gamma;\Delta_1' \vdash P_2\ \mathfrak{R}\ \Delta_2' \vdash Q_2$*; and*
3) *The symmetric cases of 1 and 2.*

The largest such bisimulation is called characteristic bisimilarity *and denoted by* $\approx^{\mathsf{C}}$.

Interestingly, for reasoning about $\mathsf{HO}\pi$ processes not in π we can exploit the simpler *higher-order bisimilarity*. We replace triggers as in (1) with *higher-order triggers*:

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

We may then define:

Definition 9 (Higher-Order Bisimilarity). *Higher-order bisimilarity, denoted by $\approx^{\mathsf{H}}$, is defined by replacing Clause (1) in Def. 8 with the following clause:*

Whenever $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu \tilde{m_1})n!\langle V_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$ *then there exist* Q_2, V_2, Δ_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, for fresh* t,
$\Gamma;\Delta_1'' \vdash (\nu \tilde{m_1})(P_2 \mid t \hookleftarrow V_1)\ \mathfrak{R}\ \Delta_2'' \vdash (\nu \tilde{m_2})(Q_2 \mid t \hookleftarrow V_2)$

We state the following important result, which attests the significance of $\approx^{\mathsf{H}}$:

Theorem 2. *Typed relations $\cong$, $\approx^{\mathsf{H}}$, and $\approx^{\mathsf{C}}$ coincide for $\mathsf{HO}\pi$ processes.*

Proof. Coincidence of $\cong$ and $\approx^{\mathsf{C}}$ was established in [14]. Coincidence of $\approx^{\mathsf{H}}$ with $\cong$ and $\approx^{\mathsf{C}}$ is a new result: see [15] for details. □

Remark 1 (Comparison between $\approx^{\mathsf{H}}$ and $\approx^{\mathsf{C}}$). The key difference between $\approx^{\mathsf{H}}$ and $\approx^{\mathsf{C}}$ is in the trigger process considered. Because of the application in (2), $\approx^{\mathsf{H}}$ cannot be used to reason about processes in π. In contrast, $\approx^{\mathsf{C}}$ is more general: it can uniformly input characteristic, first- or higher-order values. This convenience comes at a price: the definition of (1) requires information on the type of V; in contrast, the higher-order trigger (2) is more generic and simple, as it works independently of the given type.

An up-to technique In our setting, processes that do not use shared names are deterministic. The following up-to technique, based on determinacy properties, will be useful in proofs (§ 7). Recall that $\Gamma;\Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ denotes an internal (typed) transition.

Notation 1 (Deterministic Transitions) *We shall distinguish two kinds of internal transitions:* session transitions*, denoted $\Gamma;\Delta \vdash P \stackrel{\tau_s}{\longmapsto} \Delta' \vdash P'$, and* β-transitions*, denoted $\Gamma;\Delta \vdash P \stackrel{\tau_\beta}{\longmapsto} \Delta' \vdash P'$. Intuitively, $\stackrel{\tau_s}{\longmapsto}$ results from a session communication (i.e., synchronization between two dual endpoints); $\stackrel{\tau_\beta}{\longmapsto}$ results from an application. We write $\Gamma;\Delta \vdash P \stackrel{\tau_d}{\longmapsto} \Delta' \vdash P'$ to denote either a session transition or a β-transition. Formal definitions for $\stackrel{\tau_\beta}{\longmapsto}$ and $\stackrel{\tau_s}{\longmapsto}$ rely on an LTS for $\mathsf{HO}\pi$; see [15] for details.*

We have the following determinacy properties; see App. B.5 for details.

Lemma 1 (τ-Inertness). *(1) Let $\Gamma;\Delta \vdash P \stackrel{\tau_d}{\longmapsto} \Delta' \vdash P'$ be a deterministic transition, with balanced Δ. Then $\Gamma;\Delta \vdash P \cong \Delta' \vdash P'$ with $\Delta \longrightarrow^* \Delta'$ balanced. (2) Let P be an $\mathsf{HO}\pi^{-\mathsf{sh}}$ process. Assume $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. Then $P \longrightarrow^* P'$ implies $\Gamma;\Delta \vdash P \cong \Delta' \vdash P'$ with $\Delta \longrightarrow^* \Delta'$.*

Using the above determinacy properties, we can state the following up-to technique. We write $\stackrel{\tau_d}{\Longmapsto}$ to denote a (possibly empty) sequence of deterministic steps $\stackrel{\tau_d}{\longmapsto}$.

Lemma 2 (Up-to Deterministic Transition). *Let $\Gamma;\Delta_1 \vdash P_1 \ \mathfrak{R} \ \Delta_2 \vdash Q_1$ such that if whenever:*

1. *$\forall (\nu \tilde{m}_1) n!\langle V_1\rangle$ such that $\Gamma;\Delta_1 \vdash P_1 \overset{(\nu \tilde{m}_1) n!\langle V_1\rangle}{\longmapsto} \Delta_3 \vdash P_3$ implies that $\exists Q_2, V_2$ such that $\Gamma;\Delta_2 \vdash Q_1 \overset{(\nu \tilde{m}_2) n!\langle V_2\rangle}{\Longrightarrow} \Delta_2' \vdash Q_2$ and $\Gamma;\Delta_3 \vdash P_3 \overset{\tau_d}{\Longrightarrow} \Delta_1' \vdash P_2$ and for fresh t: $\Gamma;\Delta_1'' \vdash (\nu \tilde{m}_1)(P_2 \mid t \hookleftarrow V_1) \ \mathfrak{R} \ \Delta_2'' \vdash (\nu \tilde{m}_2)(Q_2 \mid t \hookleftarrow V_2)$.*
2. *$\forall \ell \neq (\nu \tilde{m}) n!\langle V\rangle$ such that $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_3 \vdash P_3$ implies that $\exists Q_2$ such that $\Gamma;\Delta_1 \vdash Q_1 \overset{\hat{\ell}}{\Longrightarrow} \Delta_2' \vdash Q_2$ and $\Gamma;\Delta_3 \vdash P_3 \overset{\tau_d}{\Longrightarrow} \Delta_1' \vdash P_2$ and $\Gamma;\Delta_1' \vdash P_2 \ \mathfrak{R} \ \Delta_2' \vdash Q_2$.*
3. *The symmetric cases of 1 and 2.*

Then $\mathfrak{R} \subseteq \approx^{\mathrm{H}}$.

6 Encodability Criteria for Typed Encodings

Here we define the formal notion of *encoding* by extending to a typed setting existing criteria for untyped processes (as in, e.g., [27,28,29,10,18,8,41,30]). We first define a typed calculus parameterised by a syntax, operational semantics, and typing. Based on this definition, later on we define concrete instances of (higher-order) typed calculi.

Definition 10 (Typed Calculus). *A* typed calculus *$\mathcal{L}$ is a tuple $\langle \mathsf{C}, \mathcal{T}, \overset{\tau}{\longmapsto}, \approx, \vdash\rangle$ where C and $\mathcal{T}$ are sets of processes and types, respectively; also, $\overset{\tau}{\longmapsto}$, $\approx$, and $\vdash$ denote a transition system, a typed equivalence, and a typing system for C, respectively.*

As we explain later, we write $\overset{\tau}{\longmapsto}$ to denote an operational semantics defined in terms of τ-transitions (to characterise reductions). Our notion of encoding considers mappings on processes and types; these are denoted $[\![\cdot]\!]$ and $(\!|\cdot|\!)$, respectively:

Definition 11 (Typed Encoding). *Consider typed calculi $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \overset{\tau}{\longmapsto}_1, \approx_1, \vdash_1\rangle$ and $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \overset{\tau}{\longmapsto}_2, \approx_2, \vdash_2\rangle$. Given mappings $[\![\cdot]\!] : \mathsf{C}_1 \to \mathsf{C}_2$ and $(\!|\cdot|\!) : \mathcal{T}_1 \to \mathcal{T}_2$, we write $\langle [\![\cdot]\!], (\!|\cdot|\!)\rangle : \mathcal{L}_1 \to \mathcal{L}_2$ to denote the* typed encoding *of $\mathcal{L}_1$ into $\mathcal{L}_2$.*

We assume that $(\!|\cdot|\!)$ extends to typing environments, e.g., $(\!|\Delta \cdot u : S|\!) = (\!|\Delta|\!) \cdot u : (\!|S|\!)$. We introduce syntactic criteria for typed encodings. Let σ denote a substitution of names for names (a renaming, as usual). 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 recursive variables in Γ are not affected by σ).

Definition 12 (Syntax Preservation). *We say that type encoding $\langle [\![\cdot]\!], (\!|\cdot|\!)\rangle : \mathcal{L}_1 \to \mathcal{L}_2$ is* syntax preserving *if it is:*

1. Homomorphic wrt parallel*, if $(\!|\Gamma|\!); \emptyset; (\!|\Delta_1 \cdot \Delta_2|\!) \vdash_2 [\![P_1 \mid P_2]\!] \triangleright \diamond$ then $(\!|\Gamma|\!); \emptyset; (\!|\Delta_1|\!) \cdot (\!|\Delta_2|\!) \vdash_2 [\![P_1]\!] \mid [\![P_2]\!] \triangleright \diamond$.*
2. Compositional wrt restriction*, if $(\!|\Gamma|\!); \emptyset; (\!|\Delta|\!) \vdash_2 [\![(\nu n)P]\!] \triangleright \diamond$ then $(\!|\Gamma|\!); \emptyset; (\!|\Delta|\!) \vdash_2 (\nu n)[\![P]\!] \triangleright \diamond$.*

3. Name invariant*, if* $(\!|\sigma(\Gamma)|\!);\emptyset;(\!|\sigma(\Delta)|\!) \vdash_2 [\![\sigma(P)]\!] \triangleright \diamond$ *then* $\sigma((\!|\Gamma|\!));\emptyset;\sigma((\!|\Delta|\!)) \vdash_2 \sigma([\![P]\!]) \triangleright \diamond$*, for any injective renaming of names* σ*.*

Homomorphism wrt parallel (used in, e.g., [28,29]) expresses that encodings should preserve the distributed topology of source processes. This criterion is appropriate for both encodability and non encodability results; in our setting, it is induced by rules for typed composition. Compositionality wrt restriction is also supported by typing and is useful in our encodability results (§ 7). The name invariance criterion follows [10,18].

We now state *type preservation*, a static criterion on the mapping $(\!|\cdot|\!) : \mathcal{T}_1 \to \mathcal{T}_2$: it ensures that type operators are preserved. In our setting, we have five type operators: input, output, recursion (binary operators); selection and branching (n-ary operators). Observe that the source and target languages that we shall consider share these (session) type operators. Type preservation enables us to focus on mappings $(\!|\cdot|\!)$ in which a session type operator is always translated into itself. In turn, this is key to retain the meaning of structured protocols across typed encodings.

Definition 13 (Type Preservation). *The typed encoding* $\langle [\![\cdot]\!], (\!|\cdot|\!) \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *is* type preserving *if for every k-ary type operator* op *in* $\mathcal{T}_1$ *it holds that*

$$(\!|\mathsf{op}(T_1, \cdots, T_k)|\!) = \mathsf{op}((\!|T_1|\!), \cdots, (\!|T_k|\!))$$

This way, e.g., consider a mapping of types $(\!|\cdot|\!)_u$ such that $(\!|!\langle U\rangle;S|\!)_u = ?((\!|U|\!)_u);(\!|S|\!)_u$ and $(\!|?(U);S|\!)_u = !\langle(\!|U|\!)_u\rangle;(\!|S|\!)_u$. This mapping exchanges input and output session type operators and therefore does not satisfy the type preservation criteria above.

Next we define semantic criteria:

Definition 14 (Semantic Preservation). *Consider two typed calculi* $\mathcal{L}_1$ *and* $\mathcal{L}_2$*, defined as* $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \overset{\tau}{\longmapsto}_1, \approx_1, \vdash_1 \rangle$ *and* $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \overset{\tau}{\longmapsto}_2, \approx_2, \vdash_2 \rangle$*. We say that the encoding* $\langle [\![\cdot]\!], (\!|\cdot|\!) \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *is* semantic preserving *if it satisfies the properties below.*

1. Type Soundness*: 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. Barb Preserving*: if* $\Gamma;\Delta \vdash_1 P \downarrow_n$ *then* $(\!|\Gamma|\!);(\!|\Delta|\!) \vdash_2 [\![P]\!] \Downarrow_n$
3. Operational Correspondence*: If* $\Gamma;\emptyset;\Delta \vdash_1 P \triangleright \diamond$ *then*
  (a) Completeness: If $\Gamma;\Delta \vdash_1 P \overset{\tau}{\longmapsto}_1 \Delta' \vdash_1 P'$ *then* $\exists Q, \Delta''$ *s.t.*
    (i) $(\!|\Gamma|\!);(\!|\Delta|\!) \vdash_2 [\![P]\!] \Longrightarrow_2 (\!|\Delta''|\!) \vdash_2 Q$ *and (ii)* $(\!|\Gamma|\!);(\!|\Delta''|\!) \vdash_2 Q \approx_2 (\!|\Delta'|\!) \vdash_2 [\![P']\!]$*.*
  (b) Soundness: If $(\!|\Gamma|\!);(\!|\Delta|\!) \vdash_2 [\![P]\!] \Longrightarrow_2 (\!|\Delta'|\!) \vdash_2 Q$ *then* $\exists P', \Delta''$ *s.t.*
    (i) $\Gamma;\Delta \vdash_1 P \overset{\tau}{\longmapsto}_1 \Delta'' \vdash_1 P'$ *and (ii)* $(\!|\Gamma|\!);(\!|\Delta''|\!) \vdash_2 [\![P']\!] \approx_2 (\!|\Delta'|\!) \vdash_2 Q$*.*
4. Full Abstraction: $\Gamma;\Delta \vdash_1 P \approx_1 \Delta' \vdash_1 Q$ *if and only if* $(\!|\Gamma|\!);(\!|\Delta|\!) \vdash_2 [\![P]\!] \approx_2 (\!|\Delta'|\!) \vdash_2 [\![Q]\!]$*.*

Together with type preservation (Def. 13), type soundness is a distinguishing criterion in our notion of encoding: it enables us to focus on encodings which retain the communication structures denoted by session types. Operational correspondence, standardly divided into completeness and soundness, is based on [10,18]; it relies on τ-labeled transitions (reductions). Completeness ensures that a step of the source process is mimicked by a step of its associated encoding; soundness is its converse. Above, operational correspondence is stated in generic terms. It is worth stressing that the operational correspondence statements for our encodings are tailored to the specifics of each encoding,

and so they are actually stronger than the criteria given above (see Props. 3, 6, 10, 13 and [15] for details). Finally, following [32,29,45], we consider full abstraction as an encodability criterion: this leads to stronger encodability results.

We introduce *precise* and *minimal* encodings. While we state strong positive encodability results in terms of *precise* encodings, to prove the non-encodability result in § 7.4, we appeal to the weaker *minimal* encodings.

Definition 15 (Typed Encodings: Precise and Minimal). *We say that the typed encoding* $\langle [\![\cdot]\!], (\!|\cdot|\!) \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ *is* precise*, if it is syntax, type, and semantic preserving (Defs. 12, 13, 14). We say that the encoding is* minimal*, if it is syntax preserving (Def. 12), barb preserving (Def. 14-2), and operationally complete (Def. 14-3(a)).*

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

7 Expressiveness Results

We present two encodability results: (1) higher-order communication with name-passing and recursion ($\mathsf{HO}\pi$) into higher-order communication without name-passing nor recursion (HO) (§ 7.1); and (2) $\mathsf{HO}\pi$ into the first-order calculus with name-passing with recursion (π) (§ 7.2). In § 7.3 we compare these two encodings. Also, in § 7.4 we state our impossibility result for shared/linear names. We consider the typed calculi (cf. Def. 10):

$$\mathcal{L}_{\mathsf{HO}\pi} = \langle \mathsf{HO}\pi, \mathcal{T}_1, \overset{\tau}{\longmapsto}, \approx^{\mathsf{H}}, \vdash \rangle \quad \mathcal{L}_{\mathsf{HO}} = \langle \mathsf{HO}, \mathcal{T}_2, \overset{\tau}{\longmapsto}, \approx^{\mathsf{H}}, \vdash \rangle \quad \mathcal{L}_{\pi} = \langle \pi, \mathcal{T}_3, \overset{\tau}{\longmapsto}, \approx^{\mathsf{C}}, \vdash \rangle$$

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 Fig. 10. The LTSs follow the intuitions given in § 5.2. Moreover, $\approx^{\mathsf{H}}$ is as in Def. 9, and $\approx^{\mathsf{C}}$ is as in Def. 8.

7.1 From $\mathsf{HO}\pi$ to HO

We show that HO is expressive enough to represent the full $\mathsf{HO}\pi$-calculus. The main challenges are to encode (1) name passing and (2) recursion, for which we only use abstraction passing. As explained in § 2, for (1), we pass an abstraction which enables to use the name upon application. For (2), we copy a process upon reception; passing around linear abstractions is delicate because they cannot be copied. To handle linearity, we define a mapping $\llbracket \cdot \rrbracket_{\sigma}$. from processes with free names to processes without free names (but with free variables instead):

Definition 16 (Auxiliary Mapping). *Let* $\| \cdot \| : 2^{\mathcal{N}} \longrightarrow \mathcal{V}^{\omega}$ *denote a map of sequences of lexicographically ordered names to sequences of variables, defined inductively as:* $\|\epsilon\| = \epsilon$ *and* $\|n \cdot \tilde{m}\| = x_n \cdot \|\tilde{m}\|$*. Also, let* σ *be a set of session names. Fig. 5 defines an auxiliary mapping* $\llbracket \cdot \rrbracket_{\sigma} : \mathsf{HO} \to \mathsf{HO}$*.*

This way, given an HO process P (with free session names given by $\mathtt{fn}(P) = \{m_1, \cdots, m_k\}$), we are interested in the abstraction $\lambda x_1 \cdots x_n . \llbracket P \rrbracket_{\emptyset}$, where $\|m_j\| = x_j$, for all $j \in \{1, \ldots, k\}$.

$$\llbracket \mathbf{0} \rrbracket_\sigma \stackrel{\text{def}}{=} \mathbf{0} \qquad \llbracket n!\langle \lambda x.\, Q\rangle.P \rrbracket_\sigma \stackrel{\text{def}}{=} u!\langle \lambda x.\, \llbracket Q \rrbracket_\sigma\rangle.\llbracket P \rrbracket_\sigma \qquad \llbracket (\nu\, n)P \rrbracket_\sigma \stackrel{\text{def}}{=} (\nu n)\llbracket P \rrbracket_{\sigma\cdot n}$$

$$\llbracket P \mid Q \rrbracket_\sigma \stackrel{\text{def}}{=} \llbracket P \rrbracket_\sigma \mid \llbracket Q \rrbracket_\sigma \qquad \llbracket x n \rrbracket_\sigma \stackrel{\text{def}}{=} x\, u \qquad \llbracket (\lambda x.Q)\, n \rrbracket_\sigma \stackrel{\text{def}}{=} (\lambda x.\llbracket Q \rrbracket_\sigma)\, u$$

$$\llbracket n?(x).P \rrbracket_\sigma \stackrel{\text{def}}{=} u?(x).\llbracket P \rrbracket_\sigma \quad \llbracket n \triangleleft l.P \rrbracket_\sigma \stackrel{\text{def}}{=} u \triangleleft l.\llbracket P \rrbracket_\sigma \quad \llbracket n \triangleright \{l_i : P_i\}_{i\in I} \rrbracket_\sigma \stackrel{\text{def}}{=} u \triangleright \{l_i : \llbracket P_i \rrbracket_\sigma\}_{i\in I}$$

In all cases: $u = n$ if $n \in \sigma$; otherwise $u = x_n$.

Fig. 5: Auxiliary mapping used to encode HOπ into HO (Def. 16).

Types:

$$\lfloor S \rfloor^1 \stackrel{\text{def}}{=} (?(\langle\!\langle S \rangle\!\rangle^1 \multimap \diamond); \mathsf{end}) \multimap \diamond \qquad \lfloor \langle S \rangle \rfloor^1 \stackrel{\text{def}}{=} (?(\langle\langle\!\langle S \rangle\!\rangle^1\rangle \to \diamond); \mathsf{end}) \multimap \diamond$$

$$\lfloor \langle L \rangle \rfloor^1 \stackrel{\text{def}}{=} (?(\langle\langle\!\langle L \rangle\!\rangle^1\rangle \to \diamond); \mathsf{end}) \multimap \diamond \qquad \lfloor C \multimap \diamond \rfloor^1 \stackrel{\text{def}}{=} \langle\!\langle C \rangle\!\rangle^1 \multimap \diamond \qquad \lfloor C \to \diamond \rfloor^1 \stackrel{\text{def}}{=} \langle\!\langle C \rangle\!\rangle^1 \to \diamond$$

$$\langle\!\langle \langle S \rangle \rangle\!\rangle^1 \stackrel{\text{def}}{=} \langle \langle\!\langle S \rangle\!\rangle^1 \rangle \qquad \langle\!\langle \langle L \rangle \rangle\!\rangle^1 \stackrel{\text{def}}{=} \langle \langle\!\langle L \rangle\!\rangle^1 \rangle$$

$$\langle\!\langle !\langle U \rangle; S \rangle\!\rangle^1 \stackrel{\text{def}}{=} !\langle \lfloor U \rfloor^1 \rangle; \langle\!\langle S \rangle\!\rangle^1 \qquad \langle\!\langle ?(U); S \rangle\!\rangle^1 \stackrel{\text{def}}{=} ?(\lfloor U \rfloor^1); \langle\!\langle S \rangle\!\rangle^1$$

$$\langle\!\langle \oplus\{l_i : S_i\}_{i\in I} \rangle\!\rangle^1 \stackrel{\text{def}}{=} \oplus\{l_i : \langle\!\langle S_i \rangle\!\rangle^1\}_{i\in I} \qquad \langle\!\langle \&\{l_i : S_i\}_{i\in I} \rangle\!\rangle^1 \stackrel{\text{def}}{=} \&\{l_i : \langle\!\langle S_i \rangle\!\rangle^1\}_{i\in I}$$

$$\langle\!\langle \mathsf{t} \rangle\!\rangle^1 \stackrel{\text{def}}{=} \mathsf{t} \qquad \langle\!\langle \mu \mathsf{t}.S \rangle\!\rangle^1 \stackrel{\text{def}}{=} \mu \mathsf{t}.\langle\!\langle S \rangle\!\rangle^1 \qquad \langle\!\langle \mathsf{end} \rangle\!\rangle^1 \stackrel{\text{def}}{=} \mathsf{end}$$

Terms:

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

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

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

$$\llbracket \mathbf{0} \rrbracket^1_f \stackrel{\text{def}}{=} \mathbf{0} \qquad \llbracket (\nu n)P \rrbracket^1_f \stackrel{\text{def}}{=} (\nu n)\llbracket P \rrbracket^1_f$$

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

$$\llbracket P \mid Q \rrbracket^1_f \stackrel{\text{def}}{=} \llbracket P \rrbracket^1_f \mid \llbracket Q \rrbracket^1_f$$

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

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

Above $\mathtt{fn}(P)$ denotes a lexicographically ordered sequence of free names in P. The input bound variable x is annotated by a type to distinguish first- and higher-order cases.

Fig. 6: Encoding of HOπ into HO (Def. 17).

Definition 17 (Typed Encoding of HOπ into HO). *Let f be a map from process variables to sequences of name variables. The typed encoding $\langle \llbracket \cdot \rrbracket^1_f, \langle\!\langle \cdot \rangle\!\rangle^1 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\mathsf{HO}}$ is defined in Fig. 6. We assume that the mapping $\langle\!\langle \cdot \rangle\!\rangle^1$ on types is extended to session environments Δ and shared environments Γ homomorphically with:*

$$\langle\!\langle \Gamma \cdot X : \Delta \rangle\!\rangle^1 = \langle\!\langle \Gamma \rangle\!\rangle^1 \cdot z_X : (S_1, \ldots, S_m, S^*) \to \diamond$$

with $\Delta = \{n_i : S_i\}_{1\le i\le m}$ and $S^* = \mu \mathsf{t}.?((S_1, \ldots, S_m, \mathsf{t}) \to \diamond); \mathsf{end}$.

Note that Δ in $X : \Delta$ is mapped to a non-tail recursive session type with variable z_X (see recursion mappings in Fig. 6). Non-tail recursive session types have been studied

in [3,2]; to our knowledge, this is the first application in the context of higher-order session types. For simplicity of the presentation, we use name abstractions with polyadicity. A precise encoding of polyadicity into HO is given in § 8.

Key elements in Fig. 6 are encodings of *name passing* ($[\![u!\langle w\rangle.P]\!]^1_f$ and $[\![u?(x).P]\!]^1_f$) and *recursion* ($[\![\mu X.P]\!]^1_f$ and $[\![X]\!]^1_f$). As already motivated in § 2, a name w is passed as an input-guarded abstraction; on the receiver side, the encoding realises a mechanism that i) receives the abstraction; ii) applies to it a fresh endpoint s; iii) uses the dual endpoint $\overline{s}$ to send the continuation P as an abstraction. Thus, name substitution is achieved via name application. As for recursion, to encode $\mu X.P$ we first record a mapping from recursive variable X to process variable z_X. Then, using $[\![\cdot]\!]_\sigma$ in Def. 16, we encode the recursion body P as a name abstraction in which free names of P are converted into name variables. (Notice that P is first encoded into HO and then transformed using mapping $[\![\cdot]\!]_\sigma$.) Subsequently, this higher-order value is embedded in an input-guarded "duplicator" process. We encode X in such a way that it simulates recursion unfolding by invoking the duplicator in a by-need fashion. That is, upon reception, the HO abstraction encoding P is duplicated: one copy is used to reconstitute the original recursion body P (through the application of $\mathtt{fn}(P)$); another copy is used to re-invoke the duplicator when needed. We illustrate the encoding by means of an example.

Example 1 (The Encoding $[\![\cdot]\!]^1_f$ At Work). Let $P = \mu X.a!\langle m\rangle.X$ be an HOπ process. Its associated encoding into HO is as follows—we note that initially $f = \emptyset$.

$$
\begin{aligned}
[\![P]\!]^1_f &= (\nu\, s_1)(s_1?(x).[\![a!\langle m\rangle.X]\!]^1_{f'} \mid \overline{s_1}!\langle\lambda(x_a,x_m,z).\, z?(x).[\![[\![a!\langle m\rangle.X]\!]^1_{f'}]\!]_\emptyset\rangle.\mathbf{0})\\
[\![a!\langle m\rangle.X]\!]^1_{f'} &= a!\langle\lambda z.\, z?(x).(x\,m)\rangle.[\![X]\!]^1_{f'}\\
&= a!\langle\lambda z.\, z?(x).(x\,m)\rangle.(\nu\, s_2)(x(a,m,s_2) \mid \overline{s_2}!\langle\lambda(x_a,x_m,z).\, x(x_a,x_m,z)\rangle.\mathbf{0})\\
[\![[\![a!\langle m\rangle.X]\!]^1_{f'}]\!]_\emptyset &= [\![a!\langle\lambda z.\, z?(x).(x\,m)\rangle.(\nu\, s_2)(x(a,m,s_2) \mid \overline{s_2}!\langle\lambda(x_a,x_m,z).\, x(x_a,x_m,z)\rangle.\mathbf{0})]\!]_\emptyset\\
&= x_a!\langle\lambda z.\, z?(x).(x\,x_m)\rangle.[\![(\nu\, s_2)(x(a,m,s_2) \mid \overline{s_2}!\langle\lambda(x_a,x_m,z).\, x(x_a,x_m,z)\rangle.\mathbf{0})]\!]_\emptyset\\
&= x_a!\langle\lambda z.\, z?(x).(x\,x_m)\rangle.(\nu\, s_2)(x(x_a,x_m,s_2) \mid \overline{s_2}!\langle\lambda(x_a,x_m,z).\, x(x_a,x_m,z)\rangle.\mathbf{0})
\end{aligned}
$$

where $f' = X \to x_a x_m$. That is, by writing V to denote the process

$$\lambda(x_a,x_m,z).\, z?(x).x_a!\langle\lambda z.\, z?(x).(x\,x_m)\rangle.(\nu\, s_2)(x(x_a,x_m,s_2) \mid \overline{s_2}!\langle\lambda(x_a,x_m,z).\, x(x_a,x_m,z)\rangle.\mathbf{0})$$

we would have that $P = \mu X.a!\langle m\rangle.X$ is mapped into the HO process

$$(\nu\, s_1)(s_1?(x).a!\langle\lambda z.\, z?(x).(x\,m)\rangle.(\nu\, s_2)(x(a,m,s_2) \mid \overline{s_2}!\langle\lambda(x_a,x_m,z).\, x(x_a,x_m,z)\rangle.\mathbf{0}) \mid \overline{s_1}!\langle V\rangle.\mathbf{0})$$

We illustrate the behaviour of the encoded process:

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

where $\ell = a!\langle \lambda z.\, z?(x).(x\, m)\rangle$ is an output action. For type preservation/soundness see [15].

We now describe the properties of the encoding. Directly from Fig. 6 we may state:

Proposition 2 (HOπ into HO: Type Preservation). *The encoding from $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_{\mathsf{HO}}$ (cf. Def. 17) is type preserving.*

Now, we state operational correspondence with respect to reductions; the full statement (and proof) can be found in [15].

Proposition 3 (HOπ into HO: Operational Correspondence - Excerpt). *Let P be an* HOπ *process such that $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$.*

1. *Completeness: Suppose $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash P'$. Then we have:*
   a) *If $P' \equiv (\nu\, \tilde{m})(P_1 \mid P_2\{^m/_x\})$ then $\exists R$ s.t.*
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\, \tilde{m})([\![P_1]\!]^1_f \mid R)$, *and*
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash (\nu\, \tilde{m})([\![P_1]\!]^1_f \mid R) \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\, \tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^m/_x\})$.
   b) *If $P' \equiv (\nu\, \tilde{m})(P_1 \mid P_2\{^{\lambda y.\, Q}/_x\})$ then*
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta_1|\!)^1 \vdash (\nu\, \tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^{\lambda y.\, [\![Q]\!]^1_\emptyset}/_x\})$.
   c) *If $P' \not\equiv (\nu\, \tilde{m})(P_1 \mid P_2\{^m/_x\}) \wedge P' \not\equiv (\nu\, \tilde{m})(P_1 \mid P_2\{^{\lambda y.\, Q}/_x\})$ then*
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta'_1|\!)^1 \vdash [\![P']\!]^1_f$.
2. *Soundness: Suppose $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^1 \vdash Q$. Then $\Delta' = \Delta$ and either*
   a) *$\exists P'$ s.t. $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta \vdash P'$, and $Q = [\![P']\!]^1_f$.*
   b) *$\exists P_1, P_2, x, m, Q'$ s.t. $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta \vdash (\nu\, \tilde{m})(P_1 \mid P_2\{^m/_x\})$, and*
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash Q \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta|\!)^1 \vdash [\![P_1]\!]^1_f \mid [\![P_2\{^m/_x\}]\!]^1_f$

Observe how we can explicitly distinguish the role of finite, deterministic reductions (Not. 1) in both soundness and completeness statements.

Using operational correspondence, we can show *full abstraction*:

Proposition 4 (HOπ into HO: Full Abstraction). *Let P_1, Q_1 be* HOπ *processes. $\Gamma;\Delta_1 \vdash P_1 \approx^{\mathrm{H}} \Delta_2 \vdash Q_1$ if and only if $(\!|\Gamma|\!)^1;(\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \approx^{\mathrm{H}} (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]^1_f$.*

We may state the main result of this section. See [15] for details.

Theorem 3 (Precise Encoding of HOπ into HO). *The encoding from $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_{\mathsf{HO}}$ (cf. Def. 17) is precise.*

7.2 From HOπ to π

We now discuss the encodability of HO into π. We closely follow Sangiorgi's encoding [33,36], which represents the exchange of a process with the exchange of a fresh *trigger name*. Trigger names may then be used to activate copies of the process, which becomes a persistent resource represented by an input-guarded replication. We cast this strategy in the setting of session-typed communications. In the presence of linear session names (which cannot be replicated), our approach uses replicated names as triggers for shared resources and non-replicated names for linear resources (cf. $[\![u!\langle \lambda x.\, Q\rangle.P]\!]^2$).

Types:

$$(\!|!\langle S \multimap \diamond\rangle; S_1|\!)^2 \stackrel{\text{def}}{=} !\langle\langle ?((\!|S|\!)^2); \mathsf{end}\rangle\rangle; (\!|S_1|\!)^2 \qquad (\!|?(S \multimap \diamond); S_1|\!)^2 \stackrel{\text{def}}{=} ?(\langle ?((\!|S|\!)^2); \mathsf{end}\rangle); (\!|S_1|\!)^2$$

Terms:

$$[\![u!\langle\lambda x.Q\rangle.P]\!]^2 \stackrel{\text{def}}{=} \begin{cases} (\nu a)(u!\langle a\rangle.([\![P]\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2)) & (s \notin \mathtt{fn}(Q)) \\ (\nu a)(u!\langle a\rangle.([\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2)) & (\text{otherwise}) \end{cases}$$

$$[\![u?(x).P]\!]^2 \stackrel{\text{def}}{=} u?(x).[\![P]\!]^2$$

$$[\![x\,u]\!]^2 \stackrel{\text{def}}{=} (\nu s)(x!\langle s\rangle.\overline{s}!\langle u\rangle.\mathbf{0})$$

$$[\![(\lambda x.P)\,u]\!]^2 \stackrel{\text{def}}{=} (\nu s)(s?(x).[\![P]\!]^2 \mid \overline{s}!\langle u\rangle.\mathbf{0})$$

Notice: $*P$ means $\mu X.(P \mid X)$. Elided mappings are homomorphic.

Fig. 7: Encoding of HOπ into π (Def. 18).

Definition 18 (Typed Encoding of HOπ into π). *The typed encoding* $\langle [\![\cdot]\!]^2, (\!|\cdot|\!)^2\rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_\pi$ *is defined in Fig. 7.*

Observe how $[\![(\lambda x.P)\,u]\!]^2$ naturally induces a name substitution. We describe key properties of this encoding. First, type preservation and operational correspondence:

Proposition 5 (HOπ into π: Type Preservation). *The encoding from* $\mathcal{L}_{\mathsf{HO}\pi}$ *into* $\mathcal{L}_\pi$ *(cf. Def. 18) is type preserving.*

Proposition 6 (HOπ into π: Operational Correspondence - Excerpt). *Let P be an* HOπ *process such that* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$.

1. *Completeness: Suppose* $\Gamma;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$. *Then either:*
   a) *If* $\ell = \tau$ *then one of the following holds:*
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash (\nu\tilde{m})([\![P_1]\!]^2 \mid (\nu a)([\![P_2]\!]^2\{a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2))$, *for some* P_1, P_2, Q;
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash (\nu\tilde{m})([\![P_1]\!]^2 \mid (\nu s)([\![P_2]\!]^2\{\overline{s}/x\} \mid s?(y).y?(x).[\![Q]\!]^2))$, *for some* P_1, P_2, Q;
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$
   b) *If* $\ell = \tau_\beta$ *then* $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$.
2. *Suppose* $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash R$.
   Then $\exists P'$ *such that* $P \overset{\tau}{\longmapsto} P'$ *and* $(\!|\Gamma|\!)^2;(\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathsf{H}} (\!|\Delta'|\!)^2 \vdash R$.

We can show full abstraction, type preservation, and preciseness:

Proposition 7 (HOπ to π: Full Abstraction). *Let* P_1, Q_1 *be* HOπ *processes.* $\Gamma;\Delta_1 \vdash P_1 \approx^{\mathsf{H}} \Delta_2 \vdash Q_1$ *if and only if* $(\!|\Gamma|\!)^2;(\!|\Delta_1|\!)^2 \vdash [\![P_1]\!]^2 \approx^{\mathsf{C}} (\!|\Delta_2|\!)^2 \vdash [\![Q_1]\!]^2$.

Theorem 4 (Precise Encoding of HOπ into π). *The encoding from* $\mathcal{L}_{\mathsf{HO}\pi}$ *into* $\mathcal{L}_\pi$ *(cf. Def. 18) is precise.*

7.3 Comparing Precise Encodings

The precise encodings reported in § 7.1 and § 7.2 confirm that HO and π constitute two important sources of expressiveness in HOπ. This naturally begs the question: which of the two sub-calculi is more tightly related to HOπ? We empirically and formally argue that when compared to π, HO is more economical and satisfies tighter correspondences.

Empirical Comparison: Reduction Steps We first contrast the way in which (a) the encoding from HOπ to HO (§ 7.1) translates processes with name passing; (b) the encoding from HOπ to π (§ 7.2) translates processes with abstraction passing. Consider the HOπ processes:

$$P_1 = s!\langle a\rangle.\mathbf{0} \mid \overline{s}?(x).(x!\langle s_1\rangle.\mathbf{0} \mid \ldots \mid x!\langle s_n\rangle.\mathbf{0}) \qquad P_2 = s!\langle \lambda x.P\rangle.\mathbf{0} \mid \overline{s}?(x).(x\,s_1 \mid \ldots \mid x\,s_n)$$

Observe that P_1 features *pure* name passing (no abstraction-passing), whereas P_2 involves *pure* abstraction passing (no name passing). In both cases, the intended communication on s leads to n usages of the communication object (name a in P_1, abstraction $\lambda x.P$ in P_2). Consider now the reduction steps from P_1 and P_2:

$$\begin{array}{l} P_1 \overset{\tau}{\longmapsto} a!\langle s_1\rangle.\mathbf{0} \mid \ldots \mid a!\langle s_n\rangle.\mathbf{0} \\ P_2 \overset{\tau}{\longmapsto} (\lambda x.P)\,s_1 \mid \ldots \mid (\lambda x.P)\,s_n \quad \underbrace{\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\cdots\overset{\tau_\beta}{\longmapsto}}_{n} \quad P\{s_1/x\} \mid \ldots \mid P\{s_1/x\} \end{array}$$

By considering the encoding of P_1 into HO we obtain:

$$\begin{array}{rcl} [\![P_1]\!]^1_f & = & s!\langle \lambda z.z?(y).y\,a\rangle.\mathbf{0} \mid \\ & & \qquad \overline{s}?(x).(\nu\, t)(x\,t \mid \overline{t}!\langle \lambda x.(x!\langle \lambda z.z?(y).y\,s_1\rangle.\mathbf{0} \mid \ldots \mid x!\langle \lambda z.z?(y).y\,s_n\rangle.\mathbf{0})\rangle.\mathbf{0}) \\ & \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} & (\nu\, t)(t?(y).y\,a \mid \overline{t}!\langle \lambda x.(x!\langle \lambda z.z?(y).y\,s_1\rangle.\mathbf{0} \mid \ldots \mid x!\langle \lambda z.z?(y).y\,s_n\rangle.\mathbf{0})\rangle.\mathbf{0}) \\ & \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} & a!\langle \lambda z.z?(y).y\,s_1\rangle.\mathbf{0} \mid \ldots \mid a!\langle \lambda z.z?(y).y\,s_n\rangle.\mathbf{0} \end{array}$$

Now, we encode P_2 into π:

$$\begin{array}{rcl} [\![P_2]\!]^2 & = & (\nu\, b)(s!\langle b\rangle.\mathbf{0} \mid *\, b?(y).y?(x).P) \mid \\ & & \qquad \overline{s}?(x).((\nu\, s)(x!\langle s\rangle.\overline{s}!\langle s_1\rangle.\mathbf{0}) \mid \ldots \mid (\nu\, s)(x!\langle s\rangle.\overline{s}!\langle s_n\rangle.\mathbf{0})) \\ & \overset{\tau_s}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_s}{\longmapsto} & (\nu\, b)(*\, b?(y).y?(x).P \mid P\{s_1/x\} \mid \ldots \mid (\nu\, s)(b!\langle s\rangle.\overline{s}!\langle s_n\rangle.\mathbf{0})) \\ & \Longmapsto_{2*(n-1)} & (\nu\, b)(*\, b?(y).y?(x).P \mid P\{s_1/x\} \mid \ldots \mid P\{s_n/x\}) \end{array}$$

It is clear that encoding P_1 into HO is more economical than encoding P_2 into π. Not only moving to a pure higher-order setting requires less reduction steps than in the first-order concurrency of π; in the presence of shared names, moving to a first-order setting brings the need of setting up and handling replicated processes which will eventually lead to garbage processes. In contrast, the mechanism present in HO works efficiently regardless of the linear or shared properties of the name that is "packed" into the abstraction. The use of β-transitions guarantees local synchronizations, which are arguably more economical than point-to-point, session synchronizations.

It is useful to move our comparison to a purely linear setting. Consider processes:

$$Q_1 = s'!\langle s\rangle.\mathbf{0} \mid \overline{s'}?(x).x!\langle a\rangle.\mathbf{0} \overset{\tau}{\longmapsto} s!\langle a\rangle.\mathbf{0} \qquad Q_2 = s!\langle\lambda x.P\rangle.\mathbf{0} \mid \overline{s}?(x).x\,a \overset{\tau}{\longmapsto}\overset{\tau}{\longmapsto} P\{^a/_x\}$$

Q_1 is a π process; Q_2 is an HO processs. If we consider their encodings into HO and π, respectively, we obtain:

$$\begin{array}{rcl}
[\![Q_1]\!]^1_f & = & s'!\langle\lambda z.z?(y).y\,s\rangle.\mathbf{0} \mid \overline{s'}?(x).(\nu\,t)(x\,t \mid \bar{t}!\langle\lambda x.x!\langle\lambda z.z?(y).y\,a\rangle.\mathbf{0}\rangle.\mathbf{0}) \\
 & \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} & (\nu\,t)(t?(y).y\,s \mid \bar{t}!\langle\lambda x.x!\langle\lambda z.z?(y).y\,a\rangle.\mathbf{0}\rangle.\mathbf{0}) \\
 & \overset{\tau_s}{\longmapsto} & \lambda x.x!\langle\lambda z.z?(y).y\,a\rangle.\mathbf{0}\,s \overset{\tau_\beta}{\longmapsto} s!\langle\lambda z.z?(y).y\,a\rangle.\mathbf{0} \\
[\![Q_2]\!]^2 & = & (\nu\,t)(s!\langle t\rangle.\mathbf{0} \mid \bar{t}?(y).y?(x).P) \mid \overline{s}?(x).(\nu\,s)(x!\langle s\rangle.\overline{s}!\langle a\rangle.\mathbf{0}) \\
 & \overset{\tau_s}{\longmapsto}\overset{\tau_s}{\longmapsto} & (\nu\,s)(s?(x).P \mid \overline{s}!\langle a\rangle.\mathbf{0}) \overset{\tau_s}{\longmapsto} P\{^a/_x\}
\end{array}$$

In this case, the encoding $[\![\cdot]\!]^2$ is more efficient: it induces less reduction steps. Therefore considering a fragment of HOπ without shared communications (linearity only) has consequences in terms of reduction steps. Notice that we prove that linear communications do not suffice to encode shared communications (§ 7.4).

Formal Comparison: Labelled Transition Correspondence We now formally establish differences between $[\![\cdot]\!]^1_f$ and $[\![\cdot]\!]^2$. To this end, we introduce an extra encodability criteria: a form of operational correspondence for *visible actions*. We shall write $\ell_1, \ell_2, \ldots$ to denote actions different from τ and $\overset{\ell}{\longmapsto}$ to denote a LTS. As actions from different calculi may be different, we also consider a mapping $\{\!|\cdot|\!\}$ on action labels.

Definition 19 (Labelled Correspondence / Tight Encodings). *Consider typed calculi $\mathcal{L}_1$ and $\mathcal{L}_2$, defined as $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \overset{\ell_1}{\longmapsto}_1, \approx_1, \vdash_1\rangle$ and $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \overset{\ell_2}{\longmapsto}_2, \approx_2, \vdash_2\rangle$. The encoding $\langle [\![\cdot]\!], (\!|\cdot|\!)\rangle : \mathcal{L}_1 \to \mathcal{L}_2$ satisfies* labelled operational correspondence *if it satisfies:*

1. *If $\Gamma;\Delta \vdash_1 P \overset{\ell_1}{\longmapsto}_1 \Delta' \vdash_1 P'$ then $\exists Q, \Delta'', \ell_2$ s.t. (i) $(\!|\Gamma|\!);(\!|\Delta|\!) \vdash_2 [\![P]\!] \overset{\ell_2}{\Longmapsto}_2 (\!|\Delta''|\!) \vdash_2 Q$; (ii) $\ell_2 = \{\!|\ell_1|\!\}$; (iii) $(\!|\Gamma|\!);(\!|\Delta''|\!) \vdash_2 Q \approx_2 (\!|\Delta'|\!) \vdash_2 [\![P']\!]$.*
2. *If $(\!|\Gamma|\!);(\!|\Delta|\!) \vdash_2 [\![P]\!] \overset{\ell_2}{\Longmapsto}_2 (\!|\Delta'|\!) \vdash_2 Q$ then $\exists P', \Delta'', \ell_1$ s.t. (i) $\Gamma;\Delta \vdash_1 P \overset{\ell_1}{\longmapsto}_1 \Delta'' \vdash_1 P'$; (ii) $\ell_2 = \{\!|\ell_1|\!\}$; (iii) $(\!|\Gamma|\!);(\!|\Delta''|\!) \vdash_2 [\![P']\!] \approx_2 (\!|\Delta'|\!) \vdash_2 Q$.*

A tight encoding *is a typed encoding which is precise (Def. 15) and that also satisfies labelled operational correspondence as above.*

We have the following result, which attests that HOπ and HO are more tightly related than HOπ and π:

Theorem 5. *While the encoding of* HOπ *into* HO *(Def. 17) is tight, the encoding of* HOπ *into* π *(Def. 18) is not tight.*

We substantiate the above claim by showing that the encoding $[\![\cdot]\!]^1_f$ enjoys labelled operational correspondence, whereas $[\![\cdot]\!]^2$ does not. Consider the following mapping:

$$\begin{array}{rcl}
\{\!|(\nu\tilde{m_1})n!\langle m\rangle|\!\}^1 & \overset{\text{def}}{=} & (\nu\tilde{m_1})n!\langle\lambda z.\,z?(x).x\,m\rangle \qquad \{\!|n?\langle m\rangle|\!\}^1 \overset{\text{def}}{=} n?\langle\lambda z.\,z?(x).x\,m\rangle \\
\{\!|(\nu\tilde{m})n!\langle\lambda x.P\rangle|\!\}^1 & \overset{\text{def}}{=} & (\nu\tilde{m})n!\langle\lambda x.[\![P]\!]^1_\emptyset\rangle \qquad \{\!|n?\langle\lambda x.P\rangle|\!\}^1 \overset{\text{def}}{=} n?\langle\lambda x.[\![P]\!]^1_\emptyset\rangle \\
\{\!|n\oplus l|\!\}^1 & \overset{\text{def}}{=} & n\oplus l \qquad \{\!|n\&l|\!\}^1 \overset{\text{def}}{=} n\&l
\end{array}$$

Then the following result, a complement of Prop. 3, holds:

Proposition 8 (Labelled Transition Correspondence, HOπ into HO). *Let P be an HOπ process. If $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ then:*

1. *Suppose $\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'$. 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.*
   $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash [\![P']\!]^1_f$ *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.*
   $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash [\![P_0]\!]^1_f\{^{\lambda y.[\![Q]\!]^1_\emptyset}/_x\}$ *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. $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash R$,*
   with $\ell_2 = \{\!|\ell_1|\!\}^1$, and $(\!|\Gamma|\!)^1;(\!|\Delta'|\!)^1 \vdash R \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta'|\!)^1 \vdash [\![P_0]\!]^1_f\{^m/_x\}$.
2. *Suppose $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash Q$. Then we have:*
   a) *If $\ell_2 \in \{(\nu\tilde{m})n!\langle\lambda z.\, z?(x).(xm)\rangle, (\nu\tilde{m})n!\langle\lambda x.R\rangle, s\oplus l, s\&l\}$ then $\exists \ell_1, P'$ s.t.*
   $\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'$, $\ell_1 = \{\!|\ell_2|\!\}^1$, *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.*
      $\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'\{^{\lambda y.P''}/_x\}$, $\ell_1 = \{\!|\ell_2|\!\}^1$, $[\![P'']\!]^1_\emptyset = R$, *and* $Q = [\![P']\!]^1_f$.
      (ii) $R \equiv y?(x).(xm)$ *and* $\exists \ell_1, z, P'$ *s.t.* $\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'\{^m/_z\}$,
      $\ell_1 = \{\!|\ell_2|\!\}^1$, *and* $(\!|\Gamma|\!)^1;(\!|\Delta'|\!)^1 \vdash Q \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta''|\!)^1 \vdash [\![P'\{^m/_z\}]\!]^1_f$

The analog of Prop. 8 does not hold for the encoding of HOπ into π. Consider the HOπ process:

$$\Gamma;\emptyset;\Delta \vdash s!\langle\lambda x.P\rangle.\mathbf{0} \triangleright \diamond \overset{s!\langle\lambda x.P\rangle}{\longmapsto} \emptyset \vdash \mathbf{0} \not\longmapsto$$

with $\lambda x.P$ being a linear value. We translated into π process:

$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta|\!)^2 \vdash (\nu a)(s!\langle a\rangle.\mathbf{0} \mid a?(y).y?(x).P) \triangleright \diamond \overset{s!\langle a\rangle}{\longmapsto} \Delta' \vdash a?(y).y?(x).P \triangleright \diamond \overset{a?\langle V\rangle}{\longmapsto} \ldots$$

The resulting processes have a mismatch both in the typing environment ($\Delta' \neq (\!|\emptyset|\!)^2$) and in the actions that they can subsequently observe: the first process cannot perform any action, while the second process can performs actions of the encoding of $\lambda x.P$.

7.4 A Negative Result

As most session calculi, HOπ includes communication on both shared and linear names. The former enables non determinism and unrestricted behaviour; the latter allows to represent deterministic and linear communication structures. The expressive power of shared names is also illustrated by our encoding from HOπ into π (Fig. 7). This result begs the question: can we represent shared name interaction using session name interaction? Here we prove that shared names actually add expressiveness to HOπ: we show the non existence of a minimal encoding (cf. Def. 15) of shared name communication into linear communication (see App. D for details of the proof).

Theorem 6. *There is no minimal encoding from π to $\mathsf{HO}\pi^{-\mathsf{sh}}$. Hence, for any $\mathsf{C}_1, \mathsf{C}_2 \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$, there is no minimal encoding from $\mathcal{L}_{\mathsf{C}_1}$ into $\mathcal{L}_{\mathsf{C}_2^{-\mathsf{sh}}}$.*

By Def. 17 and 18 and Prop. 3 and 4, we have:

Corollary 1. *Let $\mathsf{C}_1, \mathsf{C}_2 \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$. There exists a precise encoding from $\mathcal{L}_{\mathsf{C}_1^{-\mathsf{sh}}}$ into $\mathcal{L}_{\mathsf{C}_2^{-\mathsf{sh}}}$.*

8 Extensions

Here we extend $\mathsf{HO}\pi$ in two directions: (i) $\mathsf{HO}\pi^+$ extends $\mathsf{HO}\pi$ with higher-order applications/abstractions; (ii) $\mathsf{HO}\widetilde{\pi}$ extends $\mathsf{HO}\pi$ with polyadicity. In both cases, we detail the required modifications in the syntax and types. The two extensions may be combined into $\mathsf{HO}\widetilde{\pi}^+$: the polyadic extension of $\mathsf{HO}\pi^+$.

$\mathsf{HO}\pi$ with Higher-Order Abstractions ($\mathsf{HO}\pi^+$) and with Polyadicity ($\mathsf{HO}\widetilde{\pi}$) We first introduce $\mathsf{HO}\pi^+$, the extension of $\mathsf{HO}\pi$ with higher-order abstractions and applications. This is the calculus that we studied in [14]. The syntax of $\mathsf{HO}\pi^+$ is obtained from Fig. 2 by extending $V\,u$ to $V\,W$, where W is a higher-order value. As for the reduction semantics, we keep the rules in Fig. 3, except for [App] which is replaced by

$$(\lambda x.P)\,V \longrightarrow P\{^{V}/_{x}\}$$

The syntax of types is modified as follows:

$$L \;::=\; U{\to}\diamond \;\mid\; U{\multimap}\diamond$$

These types can be easily accommodated in the type system in Fig. 10: we replace C by U in [Abs] and C by U' in [App]. Subject reduction (Thm. 1) holds for $\mathsf{HO}\pi^+$ (cf. [14])

The calculus $\mathsf{HO}\widetilde{\pi}$ extends $\mathsf{HO}\pi$ with polyadic name passing $\tilde{n}$ and $\lambda\tilde{x}.Q$ in the syntax of value V. The operational semantics is kept unchanged, with the expected use of the simultaneous substitution $\{^{\tilde{V}}/_{\tilde{x}}\}$. The type syntax is extended to:

$$L \;::=\; \tilde{C}{\to}\diamond \;\mid\; \tilde{C}{\multimap}\diamond \qquad S \;::=\; !\langle\tilde{U}\rangle;S \;\mid\; ?(\tilde{U});S \;\mid\; \cdots$$

As in [25,26], the type system for $\mathsf{HO}\widetilde{\pi}$ disallows a shared name that directly carries polyadic shared names.

By combining $\mathsf{HO}\pi^+$ and $\mathsf{HO}\widetilde{\pi}$ into a single calculus we obtain $\mathsf{HO}\widetilde{\pi}^+$: the extension of $\mathsf{HO}\pi$ allows *both* higher-order abstractions/aplications and polyadicity.

Precise Encodings of $\mathsf{HO}\pi^+$ and $\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$ We give encodings of $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$ and into $\mathsf{HO}\widetilde{\pi}$, and show that they are precise. We also exploit encoding composition (Prop. 1) to encode $\mathsf{HO}\widetilde{\pi}^+$ into HO and π. We consider the following typed calculi (cf. Def. 10):

- $\mathcal{L}_{\mathsf{HO}\pi^+} = \langle \mathsf{HO}\pi^+, \mathcal{T}_4, \overset{\ell}{\longmapsto}, \approx^{\mathtt{H}}, \vdash \rangle$, where $\mathcal{T}_4$ is a set of types of $\mathsf{HO}\pi^+$; the typing $\vdash$ is defined in Fig. 10 with extended rules [Abs] and [App].

Types :

$(\!|L{\to}\diamond|\!)^3 \stackrel{\text{def}}{=} ?((\!|L|\!)^3);\mathsf{end}{\to}\diamond \qquad (\!|!\langle L{\to}\diamond\rangle;S|\!)^3 \stackrel{\text{def}}{=} !\langle(\!|L{\to}\diamond|\!)^3\rangle;(\!|S|\!)^3$

$(\!|L{\multimap}\diamond|\!)^3 \stackrel{\text{def}}{=} ?((\!|L|\!)^3);\mathsf{end}{\multimap}\diamond \qquad (\!|?(L{\to}\diamond);S|\!)^3 \stackrel{\text{def}}{=} ?((\!|L{\to}\diamond|\!)^3);(\!|S|\!)^3$

Terms :

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

Mappings for elided processes and types are homomorphic.

Fig. 8: Encoding of $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$.

- $\mathcal{L}_{\mathsf{HO}\widetilde{\pi}} = \langle \mathsf{HO}\widetilde{\pi}, \mathcal{T}_5, \stackrel{\ell}{\longmapsto}, \approx^{\mathrm{H}}, \vdash\rangle$, where $\mathcal{T}_5$ is the set of types of $\mathsf{HO}\widetilde{\pi}$; the typing $\vdash$ is defined in Fig. 10 with polyadic types.

First, the typed encoding $\langle [\![\cdot]\!]^3, (\!|\cdot|\!)^3\rangle : \mathsf{HO}\pi^+ \to \mathsf{HO}\pi$ is defined in Fig. 8. It satisfies the following properties:

Proposition 9 ($\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$: Type Preservation). *The encoding from $\mathcal{L}_{\mathsf{HO}\pi^+}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ (cf. Fig. 8) is type preserving.*

Proposition 10 (Operational Correspondence: From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$- Excerpt). *Let P be an $\mathsf{HO}\pi^+$ process such that $\Gamma;\emptyset;\Delta \vdash P$.*

1. *Completeness: $\Gamma;\Delta \vdash P \stackrel{\ell}{\longmapsto} \Delta' \vdash P'$ implies*
   a) *If $\ell = \tau_\beta$ then $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \stackrel{\tau}{\longmapsto} \Delta'' \vdash R$ and $(\!|\Gamma|\!)^3;(\!|\Delta'|\!)^3 \vdash [\![P']\!]^3 \approx^{\mathrm{H}} \Delta'' \vdash R$, for some R;*
   b) *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. *Soundness: $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \stackrel{\tau}{\longmapsto} (\!|\Delta''|\!)^3 \vdash Q$ implies either*
   a) *$\Gamma;\Delta \vdash P \stackrel{\tau}{\longmapsto} \Delta' \vdash P'$ with $Q \equiv [\![P']\!]^3$*
   b) *$\Gamma;\Delta \vdash P \stackrel{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^3;(\!|\Delta''|\!)^3 \vdash Q \stackrel{\tau_\beta}{\longmapsto} (\!|\Delta''|\!)^3 \vdash [\![P']\!]^3$.*

Proposition 11 (Full Abstraction. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let P, Q be $\mathsf{HO}\pi^+$ processes with $\Gamma;\emptyset;\Delta_1 \vdash P \triangleright \diamond$ and $\Gamma;\emptyset;\Delta_2 \vdash Q \triangleright \diamond$.*
Then $\Gamma;\Delta_1 \vdash P \approx^{\mathrm{H}} \Delta_2 \vdash Q$ if and only if $(\!|\Gamma|\!)^3;(\!|\Delta_1|\!)^3 \vdash [\![P]\!]^3 \approx^{\mathrm{H}} (\!|\Delta_2|\!)^3 \vdash [\![Q]\!]^3$

Using the above propositions, Thms. 3 and 4, and Prop. 1, we derive the following:

Theorem 7 (Encoding $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$). *The encoding from $\mathcal{L}_{\mathsf{HO}\pi^+}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ (cf. Fig. 8) is precise. Hence, the encodings from $\mathcal{L}_{\mathsf{HO}\pi^+}$ to $\mathcal{L}_{\mathsf{HO}}$ and $\mathcal{L}_\pi$ are also precise.*

Second, we define the typed encoding $\langle [\![\cdot]\!]^4, (\!|\cdot|\!)^4\rangle : \mathsf{HO}\widetilde{\pi} \to \mathsf{HO}\pi$ in Fig. 9. For simplicity, we give the dyadic case (tuples of length 2); the general case is as expected. Then encoding of $\mathsf{HO}\widetilde{\pi}$ satisfies the following properties:

Types :

$$\begin{array}{rcl}
(\!|!\langle S_1, S_2\rangle; S|\!)^4 & \stackrel{\text{def}}{=} & !\langle (\!|S_1|\!)^4\rangle; !\langle (\!|S_2|\!)^4\rangle; (\!|S|\!)^4 \\
(\!|!\langle L\rangle; S|\!)^4 & \stackrel{\text{def}}{=} & !\langle (\!|L|\!)^4\rangle; (\!|S|\!)^4 \\
(\!|(C_2, C_2)\to\diamond|\!)^4 & \stackrel{\text{def}}{=} & (?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathsf{end})\to\diamond \\
(\!|(C_1, C_2)\multimap\diamond|\!)^4 & \stackrel{\text{def}}{=} & (?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathsf{end})\multimap\diamond
\end{array}$$

Terms :

$$\begin{array}{rcl}
[\![u!\langle u_1, u_2\rangle.P]\!]^4 & \stackrel{\text{def}}{=} & u!\langle u_1\rangle.u!\langle u_2\rangle.[\![P]\!]^4 \\
[\![u!\langle\lambda(x_1, x_2).Q\rangle.P]\!]^4 & \stackrel{\text{def}}{=} & u!\langle\lambda z.z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P]\!]^4 \\
[\![x(u_1, u_2)]\!]^4 & \stackrel{\text{def}}{=} & (\nu s)(x\,s \mid \overline{s}!\langle u_1\rangle.\overline{s}!\langle u_2\rangle.\mathbf{0}) \\
[\![(\lambda(x_1, x_2).P)(u_1, u_2)]\!]^4 & \stackrel{\text{def}}{=} & (\nu s)(s?(x_1).s?(x_2).[\![P]\!]^4 \mid \overline{s}!\langle u_1\rangle.\overline{s}!\langle u_2\rangle.\mathbf{0})
\end{array}$$

The input cases are defined as the output cases by replacing ! by ?. Elided mappings for processes and types are homomorphic.

Fig. 9: Encoding of $\mathsf{HO}\widetilde{\pi}$ (dyadic case) into $\mathsf{HO}\pi$.

Proposition 12 ($\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$: Type Preservation). *The encoding from $\mathcal{L}_{\mathsf{HO}\widetilde{\pi}}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ (cf. Fig. 9) is type preserving.*

Proposition 13 (Operational Correspondence: From $\mathsf{HO}\widetilde{\pi}$ to $\mathsf{HO}\pi$- Excerpt). *Let $\Gamma;\emptyset;\Delta \vdash P$.*

1. *Completeness: $\Gamma;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$ implies*
   a) *If $\ell = \tau_\beta$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$*
   b) *If $\ell = \tau$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\tau}{\longmapsto}\overset{\tau}{\longmapsto}\overset{\tau}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$*
2. *Soundness: $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\longmapsto} (\!|\Delta_1|\!)^4 \vdash P_1$ implies*
   a) *If $\ell = \tau_\beta$ then $\Gamma;\Delta \vdash P \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau_s}{\longmapsto}\overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$*
   b) *If $\ell = \tau$ then $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau}{\longmapsto}\overset{\tau}{\longmapsto}\overset{\tau}{\longmapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$*

Proposition 14 (Full Abstraction: From $\mathsf{HO}\widetilde{\pi}$ to $\mathsf{HO}\pi$). *Let P, Q be $\mathsf{HO}\widetilde{\pi}$ processes with $\Gamma;\emptyset;\Delta_1 \vdash P \triangleright \diamond$ and $\Gamma;\emptyset;\Delta_2 \vdash Q \triangleright \diamond$. Then we have:*
$\Gamma;\Delta_1 \vdash P \approx^{\mathrm{H}} \Delta_2 \vdash Q$ if and only if $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \approx^{\mathrm{H}} (\!|\Delta_2|\!)^4 \vdash [\![Q]\!]^4$.

Using the above propositions, Thms. 3 and 4, and Prop. 1, we derive the following:

Theorem 8 (Encoding of $\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$). *The encoding from $\mathcal{L}_{\mathsf{HO}\widetilde{\pi}}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ (cf. Fig. 9) is precise. Hence, the encodings from $\mathcal{L}_{\mathsf{HO}\widetilde{\pi}}$ to $\mathcal{L}_{\mathsf{HO}}$ and $\mathcal{L}_\pi$ are also precise.*

By combining Thms. 7 and 8, we can extend preciseness to the super-calculus $\mathsf{HO}\widetilde{\pi}^+$.

9 Concluding Remarks and Related Work

We have thoroughly studied the expressivity of $\mathsf{HO}\pi$, the higher-order π-calculus with sessions. Unlike most previous works on the expressivity of (higher-order) process calculi, we have carried out our study in the setting of *session types*. Types not only delineate and enable encodings; they inform the techniques required to reason about such

encodings. Our results cover a wide spectrum of features intrinsic to higher-order concurrency: pure process-passing (with first- and higher-order abstractions), pure name-passing, polyadicity, linear/shared communication (see Fig. 1). Remarkably, the discipline embodied by session types turns out to be fundamental to show that all these languages are equally expressive, up to strong behavioural correspondences. Indeed, although our encodings may be exploited in an untyped setting, session type information is critical in the proofs of key properties for preciseness, in particular full abstraction.

Related Work There is a vast literature on expressiveness for process calculi, both first- and higher-order. In the untyped setting, the relative expressiveness of name-passing calculi with respect to higher-order languages is well-known. Our study recasts some classic results [32] into a session typed setting, and offers new encodability results. Our study also stresses the view of "encodings as protocols", namely session protocols which enforce linear and shared disciplines for names, a distinction not explored in previous works. This distinction is key in our technical developments: it enables us to obtain refined operational correspondence statements (cf. Props. 3, 6, 10, and 13). Remarkably, we showed that HO suffices to encode π, the first-order session calculus [11]. To our knowledge, this is a new result for session typed calculi: its significance is stressed by the demanding encodability criteria considered here, in particular full abstraction up to higher-order/characteristic bisimilarity ($\approx^{\mathsf{H}}/\approx^{\mathsf{C}}$, cf. Props. 4 and 7). As such, this new result is relevant in a broader setting, for known encodings of name-passing into higher-order calculi [36,4,21,42,44] either require heavy limitations in source/target languages, do not consider types, and/or fail to satisfy strong encodability criteria (see below). We also show that HO can encode $\mathsf{HO}\pi$ and its extension with higher-order applications ($\mathsf{HO}\pi^{+}$). Thus, all these calculi are equally expressive with fully abstract encodings (up to $\approx^{\mathsf{H}}/\approx^{\mathsf{C}}$). To our knowledge, these are the first expressivity results of this kind.

Early works on (relative) expressiveness appealed to different notions of encoding. Later on, proposals of abstract frameworks which formalise the notion of encoding and state associated syntactic/semantic criteria were put forward; recent proposals include [10,8,41,30,31]. These frameworks have been used to clarify known results and to derive new ones. Our formulation of precise encoding (Def. 15) builds upon existing proposals (e.g., [28,10,18]) in order to account for the session types associated to $\mathsf{HO}\pi$.

Early expressiveness studies for higher-order calculi are [39,32]; more recent works include [4,18,19,42,43]. Due to the close relationship between higher-order process calculi and functional calculi, encodings of (variants of) the λ-calculus into (variants of) the π-calculus (see, e.g., [33,7,46,1,37]) are also related. The well-known encoding of the higher-order π-calculus into the π-calculus by Sangiorgi [32] is fully abstract with respect to reduction-closed, barbed congruence. We have shown in § 7.2 that the analogue of Sangiorgi's encoding for the session typed setting also satisfies full abstraction (up to $\approx^{\mathsf{H}}/\approx^{\mathsf{C}}$, cf. Prop. 6). A basic form of input/output types is used in [35], where the encoding in [32] is casted in the asynchronous setting, with output and applications coalesced in a single construct. Building upon [35], a simply typed encoding for synchronous processes is given in [36]; 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 [34]

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 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.

Other related works are [4,21,42,19]. The paper [4] proposes a fully abstract, continuation-passing style encoding of the π-calculus into Homer, a higher-order calculus with explicit locations, local names, and nested locations. The paper [21] presents a *reflective* calculus with a "quoting" operator: names are quoted processes and represent the code of a process; name-passing then becomes a way of passing the code of a process. This reflective calculus can encode both first- and higher-order π-calculus. Building upon [40], the work [42] studies the (non)encodability of the untyped π-calculus into a higher-order π-calculus with a powerful name relabelling operator, which is essential to encode name-passing. In a recent development, the paper [44] defines an encoding of the (untyped) π-calculus without relabeling. The encoding in [44] is quite different from the one in § 7.1: in [44] names are encoded using polyadic name abstractions (called *pipes*); guarded replication (rather than recursion) enables infinite behaviours. While our encoding satisfies full abstraction (Prop. 4), the encoding in [44] does not: only divergence-reflection and operational correspondence (soundness and completeness) properties are established. Soundness is stated up-to *pipe-bisimilarity*, an equivalence tailored to the encoding strategy; the authors of [44] describe this result as "weak".

A minimal calculus of higher-order concurrency is studied in [19]: it lacks restriction, name passing, output prefix, and replication/recursion. Still, this sublanguage of HO is Turing equivalent. In [18] this core calculus is extended with restriction, output prefix, and polyadicity: it is shown that synchronous communication can encode asynchronous communication, and that process passing polyadicity induces an expressiveness hierarchy. The paper [43] complements [18] by studying the expressivity of second-order process abstractions. Polyadicity is shown to induce an expressiveness hierarchy; also, by adapting the encoding in [32], process abstractions are encoded into name abstractions. In contrast, we give a fully abstract encoding of $\mathsf{HO}\widetilde{\pi}^+$ into HO that preserves session types; this improves [18,43] by enforcing linearity disciplines on process behaviour. The focus of [18,42,43,44] is on untyped, higher-order processes; they do not address communication disciplined by (session) type systems.

Within session types, the works [6,5] study encodings of binary session calculi into a linearly typed π-calculus. While [6] gives a precise encoding of π into a linear calculus (an extension of [1]), the work [5] gives operational correspondence (without full abstraction) for the first- and higher-order π-calculi into [13]. By the result of [6], $\mathsf{HO}\pi^+$ is encodable into the linearly typed π-calculi. The syntax of $\mathsf{HO}\pi$ is a subset of that in [25,26]. The work [25] 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 $\mathsf{HO}\widetilde{\pi}$. Our results show that the calculus in [25] 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.

Acknowledgments We benefited from feedback from the users of the Moca mailing list, in particular Greg Meredith and Xu Xian.

References

1. M. Berger, K. Honda, and N. Yoshida. Sequentiality and the π-calculus. In *Proc. TLCA'01*, volume 2044 of *LNCS*, pages 29–45, 2001.
2. G. Bernardi, O. Dardha, S. J. Gay, and D. Kouzapas. On duality relations for session types. In *Prof. of TGC*, volume 8902 of *LNCS*, pages 51–66. Springer, 2014.
3. V. Bono and L. Padovani. Typing copyless message passing. *LMCS*, 8(1), 2012.
4. M. Bundgaard, T. T. Hildebrandt, and J. C. Godskesen. A cps encoding of name-passing in higher-order mobile embedded resources. *Theor. Comput. Sci.*, 356(3):422–439, 2006.
5. O. Dardha, E. Giachino, and D. Sangiorgi. Session types revisited. In *Proc. of PPDP'12*, pages 139–150. ACM, 2012.
6. R. Demangeon and K. Honda. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR*, volume 6901 of *LNCS*, pages 280–296. Springer, 2011.
7. Y. Fu. Variations on mobile processes. *Theor. Comput. Sci.*, 221(1-2):327–368, 1999.
8. Y. Fu and H. Lu. On the expressiveness of interaction. *Theor. Comput. Sci.*, 411(11-13):1387–1451, 2010.
9. S. J. Gay and V. T. Vasconcelos. Linear type theory for asynchronous session types. *J. Funct. Program.*, 20(1):19–50, 2010.
10. D. Gorla. Towards a unified approach to encodability and separation results for process calculi. *Inf. Comput.*, 208(9):1031–1053, 2010.
11. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138. Springer, 1998.
12. K. Honda and N. Yoshida. On reduction-based process semantics. *TCS*, 151(2):437–486, 1995.
13. N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the Pi-Calculus. *TOPLAS*, 21(5):914–947, Sept. 1999.
14. D. Kouzapas, J. A. Pérez, and N. Yoshida. Characteristic Bisimulation for Higher-Order Session Processes. In *CONCUR 2015*, volume 42 of *LIPIcs*, pages 398–411, Dagstuhl, Germany, 2015.
15. D. Kouzapas, J. A. Pérez, and N. Yoshida. Full version of this paper. Technical report, Imperial College / Univ. of Groningen, 2015. `http://arxiv.org/abs/1502.02585`.
16. D. Kouzapas and N. Yoshida. Globally governed session semantics. *LMCS*, 10(4), 2014.
17. D. Kouzapas, N. Yoshida, R. Hu, and K. Honda. On asynchronous eventful session semantics. *MSCS*, 2015.
18. I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In *ICALP*, volume 6199, pages 442–453, 2010.
19. I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. *Inf. Comput.*, 209(2):198–226, 2011.
20. C. League, Z. Shao, and V. Trifonov. Type-preserving compilation of Featherweight Java. *ACM Trans. Program. Lang. Syst.*, 24(2):112–152, 2002.
21. L. G. Meredith and M. Radestock. A reflective higher-order calculus. *Electr. Notes Theor. Comput. Sci.*, 141(5):49–67, 2005.
22. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I and II. *Info.& Comp.*, 100(1), 1992.
23. R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th ICALP*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.
24. J. G. Morrisett, D. Walker, K. Crary, and N. Glew. From System F to typed assembly language. *ACM Trans. Program. Lang. Syst.*, 21(3):527–568, 1999.

25. D. Mostrous and N. Yoshida. Two session typing systems for higher-order mobile processes. In *TLCA*, volume 4583 of *LNCS*, pages 321–335. Springer, 2007.
26. D. Mostrous and N. Yoshida. Session Typing and Asynchronous Subtying for Higher-Order π-Calculus. *Inf. Comput.*, 241:227–263, 2015.
27. U. Nestmann. What is a "good" encoding of guarded choice? *Inf. Comput.*, 156(1-2):287–319, 2000.
28. C. Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *MSCS*, 13(5):685–719, 2003.
29. C. Palamidessi, V. A. Saraswat, F. D. Valencia, and B. Victor. On the expressiveness of linearity vs persistence in the asychronous pi-calculus. In *Proc. of LICS 2006*, pages 59–68, 2006.
30. K. Peters, U. Nestmann, and U. Goltz. On distributability in process calculi. In *Proc. of ESOP 2013*, volume 7792 of *LNCS*, pages 310–329. Springer, 2013.
31. K. Peters and R. J. van Glabbeek. Analysing and comparing encodability criteria. In *Proc. of EXPRESS/SOS 2015*, volume 190 of *EPTCS*, pages 46–60, 2015.
32. D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher Order Paradigms*. PhD thesis, University of Edinburgh, 1992.
33. D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. In *7th LICS Conf.*, pages 102–109. IEEE Computer Society Press, 1992.
34. D. Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *TCS*, 167(2):235–274, 1996.
35. D. Sangiorgi. Asynchronous process calculi: the first- and higher-order paradigms. *Theor. Comput. Sci.*, 253(2):311–350, 2001.
36. D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
37. D. Sangiorgi and X. Xu. Trees from functions as processes. In *Proc. of CONCUR 2014*, volume 8704, pages 78–92. Springer, 2014.
38. Z. Shao and A. W. Appel. A type-based compiler for standard ML. In *Proc. of PLDI'95*, pages 116–129. ACM, 1995.
39. B. Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Dept. of Comp. Sci., Imperial College, 1990.
40. B. Thomsen. Plain CHOCS: A Second Generation Calculus for Higher Order Processes. *Acta Informatica*, 30(1):1–59, 1993.
41. R. J. van Glabbeek. Musings on encodings and expressiveness. In *Proc. of EXPRESS/SOS 2012*, volume 89 of *EPTCS*, pages 81–98, 2012.
42. X. Xu. Distinguishing and relating higher-order and first-order processes by expressiveness. *Acta Informatica*, 49(7-8):445–484, 2012.
43. X. Xu, Q. Yin, and H. Long. On the expressiveness of parameterization in process-passing. In *WS-FM*, volume 8379 of *LNCS*, pages 147–167. Springer, 2014.
44. X. Xu, Q. Yin, and H. Long. On the computation power of name parameterization in higher-order processes. In *Proc. of ICE 2015*, volume 189 of *EPTCS*, pages 114–127, 2015.
45. N. Yoshida. Graph types for monadic mobile processes. In *FSTTCS*, volume 1180 of *LNCS*, pages 371–386. Springer, 1996.
46. N. Yoshida, M. Berger, and K. Honda. Strong normalisation in the pi -calculus. *Inf. Comput.*, 191(2):145–202, 2004.

Table of Contents

A Appendix: the Typing System of HOπ

In this appendix we formally define *type equivalence* and *duality*. We also present and describe our typing rules, given in Fig. 10.

A.1 Type Equivalence and Duality

Definition 20 (Type Equivalence). *Let* ST *a set of closed session types. Two types S and S' are said to be* isomorphic *if a pair (S,S') is in the largest fixed point of the monotone function $F : \mathcal{P}(\mathsf{ST}\times\mathsf{ST}) \to \mathcal{P}(\mathsf{ST}\times\mathsf{ST})$ defined by:*

$$\begin{aligned}
F(\mathfrak{R}) = \{&(\mathsf{end},\mathsf{end})\}\\
&\cup\{(!\langle U_1\rangle;S_1, !\langle U_2\rangle;S_2) \mid (S_1,S_2),(U_1,U_2)\in\mathfrak{R}\}\\
&\cup\{(?(U_1);S_1, ?(U_2);S_2) \mid (S_1,S_2),(U_1,U_2)\in\mathfrak{R}\}\\
&\cup\{(\&\{l_i:S_i\}_{i\in I}, \&\{l_i:S'_i\}_{i\in I}) \mid \forall i\in I.(S_i,S'_i)\in\mathfrak{R}\}\\
&\cup\{(\oplus\{l_i:S_i\}_{i\in I}, \oplus\{l_i:S'_i\}_{i\in I}) \mid \forall i\in I.(S_i,S'_i)\in\mathfrak{R}\}\\
&\cup\{(\mu\mathsf{t}.S, S') \mid (S\{^{\mu\mathsf{t}.S}/_{\mathsf{t}}\}, S')\in\mathfrak{R}\}\\
&\cup\{(S, \mu\mathsf{t}.S') \mid (S, S'\{^{\mu\mathsf{t}.S'}/_{\mathsf{t}}\})\in\mathfrak{R}\}
\end{aligned}$$

Standard arguments ensure that F is monotone, thus the greatest fixed point of F exists. We write $S_1 \sim S_2$ if $(S_1,S_2)\in\mathfrak{R}$.

Definition 21 (Duality). *Let* ST *a set of closed session types. Two types S and S' are said to be* dual *if a pair (S,S') is in the largest fixed point of the monotone function $F : \mathcal{P}(\mathsf{ST}\times\mathsf{ST}) \to \mathcal{P}(\mathsf{ST}\times\mathsf{ST})$ defined by:*

$$\begin{aligned}
F(\mathfrak{R}) = \{&(\mathsf{end},\mathsf{end})\}\\
&\cup\{(!\langle U_1\rangle;S_1, ?(U_2);S_2) \mid (S_1,S_2)\in\mathfrak{R},\ U_1\sim U_2\}\\
&\cup\{(?(U_1);S_1, !\langle U_2\rangle;S_2) \mid (S_1,S_2)\in\mathfrak{R},\ U_1\sim U_2\}\\
&\cup\{(\oplus\{l_i:S_i\}_{i\in I}, \&\{l_i:S'_i\}_{i\in I}) \mid \forall i\in I.(S_i,S'_i)\in\mathfrak{R}\}\\
&\cup\{(\&\{l_i:S_i\}_{i\in I}, \oplus\{l_i:S'_i\}_{i\in I}) \mid \forall i\in I.(S_i,S'_i)\in\mathfrak{R}\}\\
&\cup\{(\mu\mathsf{t}.S, S') \mid (S\{^{\mu\mathsf{t}.S}/_{\mathsf{t}}\}, S')\in\mathfrak{R}\}\\
&\cup\{(S, \mu\mathsf{t}.S') \mid (S, S'\{^{\mu\mathsf{t}.S'}/_{\mathsf{t}}\})\in\mathfrak{R}\}
\end{aligned}$$

Standard arguments ensure that F is monotone, thus the greatest fixed point of F exists. We write S_1 dual S_2 if $(S_1,S_2)\in\mathfrak{R}$.

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

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

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

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

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

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

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

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

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

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

$$[\text{Par}]\ \frac{\Gamma;\Lambda_i;\Delta_i\vdash P_i\triangleright\diamond \quad i=1,2}{\Gamma;\Lambda_1\cdot\Lambda_2;\Delta_1\cdot\Delta_2\vdash P_1\mid P_2\triangleright\diamond} \qquad [\text{End}]\ \frac{\Gamma;\Lambda;\Delta\vdash P\triangleright T \quad u\notin\mathsf{dom}(\Gamma,\Lambda,\Delta)}{\Gamma;\Lambda;\Delta\cdot u:\mathsf{end}\vdash P\triangleright\diamond}$$

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

Fig. 10: Complete Typing Rules for HOπ.

A.2 Typing Rules

The typing system is defined in Fig. 10. Rules [Sess, Sh, LVar] are name and variable introduction rules. The shared type $C\to\diamond$ is derived using rule [Prom] only if the value has a linear type with an empty linear environment. Rule [EProm] allows us to freely use a linear type variable as shared. Abstraction values are typed with rule [Abs]. Application typing is governed by rule [App]: we expect the type C of an application name u to match the type $C\multimap\diamond$ or $C\to\diamond$ of the application variable x.

In rule [Send], the type U of a send value V should appear as a prefix on the session type $!\langle U\rangle;S$ of u. Rule [Rcv] is its dual. We use a similar approach with session prefixes to type interaction between shared names 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). Rules for selection and branching, denoted [Sel] and [Bra], are standard.

A shared name creation a creates and restricts a in environment Γ as defined in rule [Res]. Creation of a session name s creates and restricts two endpoints with dual types in rule [ResS]. Rule [Par] combines the environments Λ and Δ of the components of a parallel process; the disjointness of environments Λ and Δ is implied. Rule [End] adds the names with type end in Δ. The recursion requires that the body process matches the type of the recursive variable as in rule [Rec]. The recursive variable is typed directly from the shared environment Γ as in rule [RVar]. The inactive process $\mathbf{0}$ is typed with no linear environments as in rule [Nil].

B Behavioural Semantics

We present the theory of Labelled Transition Semantics as presented in [14, Sec. 4].

We begin by defining an (early) labelled transition system (LTS) on untyped processes (§ B.1). Then, using the *environmental* transition semantics (§ B.2), we define a typed LTS to formalise how a typed process interacts with a typed observer.

B.1 Labelled Transition System for Processes

Interaction is defined on action labels ℓ:

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

Label τ defines internal actions. Action $(\nu \widetilde{m})n!\langle V \rangle$ denotes the sending of value V over channel n with a possible empty set of restricted names $\widetilde{m}$ (we may write $n!\langle V \rangle$ when $\widetilde{m}$ is empty). Dually, the action for value reception is $n?\langle V \rangle$. Actions for select and branch on a label l are denoted $n \oplus l$ and $n\&l$, resp. We write $\mathtt{fn}(\ell)$ and $\mathtt{bn}(\ell)$ to denote the sets of free/bound names in ℓ, resp. Given $\ell \neq \tau$, we write $\mathtt{subj}(\ell)$ to denote the *subject* of ℓ.

Dual actions occur on subjects that are dual between them and carry the same object; thus, output is dual to input and selection is dual to branching. Formally, duality on actions is the symmetric relation $\asymp$ that satisfies: (i) $n \oplus l \asymp \overline{n}\&l$ and (ii) $(\nu \widetilde{m})n!\langle V \rangle \asymp \overline{n}?\langle V \rangle$.

The LTS over *untyped processes* is given in Fig. 11. We write $P_1 \xrightarrow{\ell} P_2$ with the usual meaning. The rules are standard [17,16]. A process with an output prefix can interact with the environment with an output action that carries a value V (rule $\langle \text{Snd} \rangle$). Dually, in rule $\langle \text{Rv} \rangle$ a receiver process can observe an input of an arbitrary value V. Select and branch processes observe the select and branch actions in rules $\langle \text{Sel} \rangle$ and $\langle \text{Bra} \rangle$, resp. Rule $\langle \text{Res} \rangle$ closes the LTS under restriction if the restricted name does not occur free in the observable action. If a restricted name occurs free in the carried value of an output action, the process performs scope opening (rule $\langle \text{New} \rangle$). Rule $\langle \text{Rec} \rangle$ handles recursion unfolding. Rule $\langle \text{Tau} \rangle$ states that two parallel processes which perform dual actions can synchronise by an internal transition. Rules $\langle \text{Par}_L \rangle$/$\langle \text{Par}_R \rangle$ and $\langle \text{Alpha} \rangle$ close the LTS under parallel composition and α-renaming.

$$\langle \text{App} \rangle\ (\lambda x.P)\,V \xrightarrow{\tau} P\{^{V}/_{x}\} \qquad \langle \text{Snd} \rangle\ n!\langle V\rangle.P \xrightarrow{n!\langle V\rangle} P \qquad \langle \text{Rv} \rangle\ n?(x).P \xrightarrow{n?\langle V\rangle} P\{^{V}/_{x}\}$$

$$\langle \text{Sel} \rangle\ s \lhd l.P \xrightarrow{s \oplus l} P \qquad \langle \text{Bra} \rangle\ s \rhd \{l_i : P_i\}_{i\in I} \xrightarrow{s\&l_j} P_j\ (j \in I)$$

$$\langle \text{Alpha} \rangle \frac{P \equiv_\alpha Q \quad Q \xrightarrow{\ell} P'}{P \xrightarrow{\ell} P'} \qquad \langle \text{Res} \rangle \frac{P \xrightarrow{\ell} P' \quad n \notin \mathtt{fn}(\ell)}{(\nu n)P \xrightarrow{\ell} (\nu n)P'}$$

$$\langle \text{New} \rangle \frac{P \xrightarrow{(\nu \widetilde{m})n!\langle V\rangle} P' \quad m \in \mathtt{fn}(V)}{(\nu m)P \xrightarrow{(\nu m\cdot \widetilde{m}')n!\langle V\rangle} P'}$$

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

$$\langle \text{Tau} \rangle \frac{P \xrightarrow{\ell_1} P' \qquad Q \xrightarrow{\ell_2} Q' \qquad \ell_1 \asymp \ell_2}{P \mid Q \xrightarrow{\tau} (\nu\, \mathtt{bn}(\ell_1) \cup \mathtt{bn}(\ell_2))(P' \mid Q')} \qquad \langle \text{Rec} \rangle \frac{P\{\mu X.P/X\} \xrightarrow{\ell} P'}{\mu X.P \xrightarrow{\ell} P'}$$

Fig. 11: The Untyped LTS for HOπ processes. We omit rule $\langle \text{Par}_R \rangle$.

B.2 Environmental Labelled Transition System

Fig. 12 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 LTSs in [17,16] to higher-order sessions. Notice that due to weakening we have $(\Gamma', \Lambda_1, \Delta_1) \overset{\ell}{\longmapsto} (\Gamma', \Lambda_2, \Delta_2)$ if $(\Gamma, \Lambda_1, \Delta_1) \overset{\ell}{\longmapsto} (\Gamma', \Lambda_2, \Delta_2)$.

Input Actions are defined by rules [SRv] and [ShRv]. In rule [SRv] the type of value V and the type of the object associated to the session type on s should coincide. The resulting type tuple must contain the environments associated to V. The dual endpoint $\overline{s}$ cannot be present in the session environment: if it were present the only possible communication would be the interaction between the two endpoints (cf. rule [Tau]). Rule [ShRv] is for shared names and follows similar principles.

Output Actions are defined by rules [SSnd] and [ShSnd]. Rule [SSnd] states the conditions for observing action $(\nu \widetilde{m})s!\langle V\rangle$ on a type tuple $(\Gamma, \Lambda, \Delta \cdot s : S)$. The session environment Δ with $s : S$ should include the session environment of the sent value V, *excluding* the session environments of names m_j in $\widetilde{m}$ which restrict the scope of value V. Analogously, the linear variable environment Λ' of V should be included in Λ. Scope extrusion of session names in $\widetilde{m}$ requires that the dual endpoints of $\widetilde{m}$ should appear in the resulting session environment. Similarly for shared names in $\widetilde{m}$ 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. Rule [ShSnd] is for output actions on shared names: the name must be typed with $\langle U\rangle$; conditions on V are identical to those on rule [SSnd].

Other Actions Rules [Sel] and [Bra] describe actions for select and branch. Rule [Tau] defines internal transitions: it keeps the session environment unchanged or reduces it (Def. 2).

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

$$[\text{SSnd}]\ \frac{\begin{array}{c}\Gamma\cdot\Gamma';\Lambda';\Delta' \vdash V \triangleright U \quad \Gamma';\emptyset;\Delta_j \vdash m_j \triangleright U_j \ \ \overline{s} \notin \mathtt{dom}(\Delta) \\ \Delta'\setminus \cup_j \Delta_j \subseteq (\Delta\cdot s : S) \ \ \Gamma';\emptyset;\Delta'_j \vdash \overline{m}_j \triangleright U'_j \ \ \Lambda' \subseteq \Lambda\end{array}}{(\Gamma;\Lambda;\Delta\cdot s : !\langle U\rangle;S) \xrightarrow{(\nu\,\widetilde{m})s!\langle V\rangle} (\Gamma\cdot\Gamma';\Lambda\backslash\Lambda';(\Delta\cdot s : S\cdot\cup_j\Delta'_j)\backslash\Delta')}$$

$$[\text{ShSnd}]\ \frac{\begin{array}{c}\Gamma\cdot\Gamma';\Lambda';\Delta' \vdash V \triangleright U \ \ \Gamma';\emptyset;\Delta_j \vdash m_j \triangleright U_j \ \ \Gamma;\emptyset;\emptyset \vdash a \triangleright \langle U\rangle \\ \Delta'\setminus \cup_j \Delta_j \subseteq \Delta \qquad \Gamma';\emptyset;\Delta'_j \vdash \overline{m}_j \triangleright U'_j \ \ \Lambda' \subseteq \Lambda\end{array}}{(\Gamma;\Lambda;\Delta) \xrightarrow{(\nu\,\widetilde{m})a!\langle V\rangle} (\Gamma\cdot\Gamma';\Lambda\backslash\Lambda';(\Delta\cdot\cup_j\Delta'_j)\backslash\Delta')}$$

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

$$[\text{Bra}]\ \frac{\overline{s} \notin \mathtt{dom}(\Delta) \quad j \in I}{(\Gamma;\Lambda;\Delta\cdot s : \&\{l_i : T_i\}_{i\in I}) \xrightarrow{s\&l_j} (\Gamma;\Lambda;\Delta\cdot s : S_j)} \qquad [\text{Tau}]\ \frac{\Delta_1 \longrightarrow \Delta_2 \vee \Delta_1 = \Delta_2}{(\Gamma;\Lambda;\Delta_1) \xrightarrow{\tau} (\Gamma;\Lambda;\Delta_2)}$$

Fig. 12: Labelled Transition System for Typed Environments.

Example 2. Consider environment $(\Gamma;\emptyset;s : !\langle !\langle S\rangle;\mathsf{end}\multimap\diamond\rangle;\mathsf{end}\cdot s' : S)$ and typed value

$$\Gamma;\emptyset;s' : S\cdot m : ?(\mathsf{end});\mathsf{end} \vdash V \triangleright !\langle \mathsf{end}\rangle;\mathsf{end}\multimap\diamond \quad \text{with} \quad V = \lambda x.\, x!\langle s'\rangle.m?(z).\mathbf{0}$$

Let $\Delta'_1 = \{\overline{m} : !\langle \mathsf{end}\rangle;\mathsf{end}\}$ and $U = !\langle !\langle S\rangle;\mathsf{end}\multimap\diamond\rangle;\mathsf{end}$. Then by [SSnd], we can derive:

$$(\Gamma;\emptyset;s : !\langle !\langle S\rangle;\mathsf{end}\multimap\diamond\rangle;\mathsf{end}\cdot s' : S) \xrightarrow{(\nu\, m)s!\langle V\rangle} (\Gamma;\emptyset;s : \mathsf{end})$$

Our typed LTS combines the LTSs in Fig. 11 and Fig. 12.

Definition 22 (Typed Transition System). A *typed transition relation* is a typed relation $\Gamma;\Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_2 \vdash P_2$ where: (1) $P_1 \xrightarrow{\ell} P_2$ and (2) $(\Gamma,\emptyset,\Delta_1) \xrightarrow{\ell} (\Gamma,\emptyset,\Delta_2)$ with $\Gamma;\emptyset;\Delta_i \vdash P_i \triangleright \diamond$ $(i = 1,2)$. We extend to $\Longrightarrow$ and $\overset{\hat{\ell}}{\Longrightarrow}$ where we write $\Longrightarrow$ for the reflexive and transitive closure of $\longrightarrow$, $\overset{\ell}{\Longrightarrow}$ for the transitions $\Longrightarrow\xrightarrow{\ell}\Longrightarrow$, and $\overset{\hat{\ell}}{\Longrightarrow}$ for $\overset{\ell}{\Longrightarrow}$ if $\ell \neq \tau$ otherwise $\Longrightarrow$.

B.3 Characteristic Processes and Values

We define characteristic processes/values:

Definition 23 (Characteristic Process and Values). Let u and U be a name and a type, respectively. Fig. 13 defines the *characteristic process* $(\!|U|\!)^u$ and the *characteristic value* $(\!|U|\!)_{\mathsf{c}}$.

Proposition 15. *Let S be a session type. Then $\Gamma;\emptyset;\Delta\cdot s : S \vdash (\!|S|\!)^s \triangleright \diamond$. Also, let $\langle U\rangle$ be a first-order (channel) type. Then $\Gamma\cdot a : \langle U\rangle;\emptyset;\Delta \vdash (\!|\langle U\rangle|\!)^a \triangleright \diamond$.*

The following example motivates the refined LTS explained in the introduction.

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

$$
[\![S]\!]_{\mathsf{c}} \stackrel{\text{def}}{=} s \ \ (s \text{ fresh}) \qquad [\![\langle S\rangle]\!]_{\mathsf{c}} \stackrel{\text{def}}{=} [\![\langle L\rangle]\!]_{\mathsf{c}} \stackrel{\text{def}}{=} a \ \ (a \text{ fresh}) \qquad [\![U\to\diamond]\!]_{\mathsf{c}} \stackrel{\text{def}}{=} [\![U\multimap\diamond]\!]_{\mathsf{c}} \stackrel{\text{def}}{=} \lambda x.[\![U]\!]^x
$$

Fig. 13: Characteristic Processes (top) and Values (bottom).

B.4 Refined Labelled Transition Semantics

We define the *refined* typed LTS by considering a transition rule for input in which admitted values are trigger or characteristic values or names:

Definition 24 (Refined Typed Labelled Transition Relation). *We define the environment transition rule for input actions using the input rules in Fig. 12:*

$$
[RRcv]\ \frac{(\Gamma_1;\Lambda_1;\Delta_1) \xrightarrow{n?\langle V\rangle} (\Gamma_2;\Lambda_2;\Delta_2) \quad V = m \vee V \equiv [\![U]\!]_{\mathsf{c}} \vee V \equiv \lambda x.t?(y).(y\,x) \text{ with } t \text{ fresh}}{(\Gamma_1;\Lambda_1;\Delta_1) \overset{n?\langle V\rangle}{\longmapsto} (\Gamma_2;\Lambda_2;\Delta_2)}
$$

Rule [RRcv] *is defined on top of rules* [SRv] *and* [ShRv] *in Fig. 12. We use the non-receiving rules in Fig. 12 together with rule* [RRcv] *to define* $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_2 \vdash P_2$ *as in Def. 22.*

Notice that $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_2 \vdash P_2$ (refined transition) implies $\Gamma;\Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_2 \vdash P_2$ (ordinary transition). Below we sometimes write $\overset{(\nu\tilde{m})n!\langle V:U\rangle}{\longmapsto}$ when the type of V is U.

Lemma 3. $\approx^{\mathsf{H}} = \approx^{\mathsf{C}} = \cong$.

Proof. A detail analysis of the proof see [15]. Also the proof for $\approx^{\mathsf{C}} = \cong$ can be found in [14].
Here we only prove the direction $\approx^{\mathsf{H}} \subseteq \approx^{\mathsf{C}}$. The direction $\approx^{\mathsf{C}} \subseteq \approx^{\mathsf{H}}$ is similar.
Consider

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

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

$$
\Gamma;\Delta_1 \vdash P \overset{(\nu m\tilde{m}_1)n!\langle V_1\rangle}{\longmapsto} \Delta_1' \vdash P' \tag{3}
$$

then $\exists Q, V_2$ such that

$$
\Gamma;\Delta_2 \vdash Q \overset{(\nu m\tilde{m}_2)n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q' \tag{4}
$$

and for fresh t:

$$\begin{array}{rl} \Gamma;\emptyset;\Delta_1' & \approx^{\mathsf{H}} \Delta_2 \vdash (\nu \tilde{m_1})(P' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ & \approx^{\mathsf{H}} \quad (\nu \tilde{m_2})(Q' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \end{array}$$

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

$$\begin{array}{rl} \Gamma;\emptyset;\Delta_1' & \overset{t?\langle [\![?(U);\mathsf{end}]\!]^x\rangle}{\longmapsto} \Delta_1'' \vdash (\nu \tilde{m_1})(P' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ & \overset{t?\langle [\![?(U);\mathsf{end}]\!]^x\rangle}{\longmapsto} \quad (\nu \tilde{m_1})(P' \mid (\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \end{array}$$

implies

$$\begin{array}{rl} \Gamma;\emptyset;\Delta_2' & \overset{t?\langle [\![?(U);\mathsf{end}]\!]^x\rangle}{\longmapsto} \Delta_2'' \vdash (\nu \tilde{m_2})(Q' \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \\ & \overset{t?\langle [\![?(U);\mathsf{end}]\!]^x\rangle}{\longmapsto} \quad (\nu \tilde{m_2})(Q' \mid (\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \end{array}$$

and $\Gamma;\emptyset;\Delta' \vdash V_2 \triangleright U$.
Transition (3) implies transition (4). It remains to show that for fresh t:

$$\begin{array}{rl} \Gamma;\emptyset;\Delta_1' & \approx^{\mathsf{H}} \Delta_2 \vdash (\nu \tilde{m_1})(P' \mid t?(x).(\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ & \approx^{\mathsf{H}} \quad (\nu \tilde{m_2})(Q' \mid t?(x).(\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \end{array}$$

The freshness of t implies that

$$\begin{array}{rl} \Gamma;\emptyset;\Delta_1' & \overset{t?\langle m'\rangle}{\longmapsto} \Delta_1'' \vdash (\nu \tilde{m_1})(P' \mid t?(x).(\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ & \overset{t?\langle m'\rangle}{\longmapsto} \quad (\nu \tilde{m_1})(P' \mid (\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \end{array}$$

and

$$\begin{array}{rl} \Gamma;\emptyset;\Delta_2' & \overset{t?\langle m'\rangle}{\longmapsto} \Delta_2'' \vdash (\nu \tilde{m_2})(Q' \mid t?(x).(\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \\ & \overset{t?\langle m'\rangle}{\longmapsto} \quad (\nu \tilde{m_2})(Q' \mid (\nu\, s)([\![?(U);\mathsf{end}]\!]^s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \end{array}$$

which coincides with the transitions for $\approx^{\mathsf{H}}$.
- The rest of the cases are trivial.
The direction $\approx^{\mathsf{C}} \subseteq \approx^{\mathsf{H}}$ is very similar to the direction $\approx^{\mathsf{H}} \subseteq \approx^{\mathsf{C}}$: it requires a case analysis on the transition label ℓ. Again the non-trivial case is $\ell = (\nu \tilde{m_1})n!\langle V_1\rangle$. □

B.5 τ-inertness

We prove Part 1 of Prop. 1.

Proposition 16 (τ-inertness). Let balanced $\mathsf{HO}\pi$ process $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. $\Gamma;\Delta \vdash P \overset{\tau_d}{\longmapsto} \Delta' \vdash P'$ implies $\Gamma;\Delta \vdash P \approx \Delta' \vdash P'$.

Proof. The proof is done by induction on the structure of $\overset{\tau}{\longrightarrow}$ which coincides the reduction $\longrightarrow$.
Basic step:

- Case: $P = (\lambda x.P)n$:

$$\Gamma;\Delta \vdash (\lambda x.P)n \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P\{^n/_x\}$$

Bisimulation requirements hold since, there is no other transition to observe than $\overset{\tau_\beta}{\longmapsto}$.
- Case: $P = s!\langle V\rangle.P_1 \mid \overline{s}?(x).P_2$:

$$\Gamma;\Delta \vdash s!\langle V\rangle.P_1 \mid \overline{s}?(x).P_2 \overset{\tau_s}{\longmapsto} \Delta' \vdash P_1 \mid P_2$$

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 $\Gamma;\Delta \vdash P \approx \Delta' \vdash P'$.
- 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 $\Gamma_1;\Delta_1 \vdash P_1 \approx \Delta_2 \vdash P_2$.
Induction Step:
- Case: $P = (\nu\, s)P_1$

$$\Gamma;\Delta \vdash (\nu\, s)P_1 \overset{\tau_s}{\longmapsto} \Delta' \vdash (\nu\, s)P_2$$

From the induction hypothesis and the fact that bisimulation is a congruence we get that $\Gamma;\Delta \vdash P \approx \Delta' \vdash P'$.
- Case: $P = P_1 \mid P_3$

$$\Gamma;\Delta \vdash P_1 \mid P_3 \overset{\tau_s}{\longmapsto} \Delta' \vdash P_2 \mid P_3$$

From the induction hypothesis and the fact that bisimulation is a congruence we get that $\Gamma;\Delta \vdash P \approx \Delta' \vdash P'$.
- Case: $P \equiv P_1$

From the induction hypothesis and the fact that bisimulation is a congruence and structural congruence preserves $\approx$ we get that $\Gamma;\Delta \vdash P \approx \Delta' \vdash P'$.

□

Lemma 4 (Up-to Deterministic Transition). *Let $\Gamma;\Delta_1 \vdash P_1 \ \mathfrak{R}\ \Delta_2 \vdash Q_1$ such that if whenever:*

1. *$\forall(\nu\tilde{m_1})n!\langle V_1\rangle$ such that $\Gamma;\Delta_1 \vdash P_1 \overset{(\nu\tilde{m_1})n!\langle V_1\rangle}{\longmapsto} \Delta_3 \vdash P_3$ implies that $\exists Q_2, V_2$ such that $\Gamma;\Delta_2 \vdash Q_1 \overset{(\nu\tilde{m_2})n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$ and $\Gamma;\Delta_3 \vdash P_3 \overset{\tau_d}{\Longmapsto} \Delta_1' \vdash P_2$ and for fresh t: $\Gamma;\Delta_1'' \vdash (\nu\tilde{m_1})(P_2 \mid t \hookleftarrow V_1)\ \mathfrak{R}\ \Delta_2'' \vdash (\nu\tilde{m_2})(Q_2 \mid t \hookleftarrow V_2)$.*
2. *$\forall \ell \neq (\nu\tilde{m})n!\langle V\rangle$ such that $\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_3 \vdash P_3$ implies that $\exists Q_2$ such that $\Gamma;\Delta_1 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$ and $\Gamma;\Delta_3 \vdash P_3 \overset{\tau_d}{\Longmapsto} \Delta_1' \vdash P_2$ and $\Gamma;\Delta_1' \vdash P_2\ \mathfrak{R}\ \Delta_2' \vdash Q_2$.*
3. *The symmetric cases of 1 and 2.*

Then $\mathfrak{R} \subseteq \approx^{\mathrm{H}}$.

Proof. The proof is easy by considering the closure

$$\mathfrak{R}^{\overset{\tau_d}{\Longmapsto}} = \{\Gamma;\Delta_1' \vdash P_2, \Delta_2' \vdash Q_1 \ \mid\ \Gamma;\Delta_1 \vdash P_1\ \mathfrak{R}\ \Delta_2' \vdash Q_1, \Gamma;\Delta_1 \vdash P_1 \overset{\tau_d}{\Longmapsto} \Delta_1' \vdash P_2\}$$

We verify that $\mathfrak{R}^{\overset{\tau_d}{\Longmapsto}}$ is a bisimulation with the use of Prop. 1. □

C Expressiveness Results

In this section we give the proofs for the expressiveness results stated in Section 7 and in Section 8.

For the purpose of proving Operational Correspondence in this section we prove a stronger result than the result suggested in Def. 14(2). We state the requirements below.

Definition 25 (Operational Correspondence). Consider typed calculi $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \overset{\ell_1}{\longmapsto}_1, \approx_1, \vdash_1 \rangle$ and $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \overset{\ell_2}{\longmapsto}_2, \approx_2, \vdash_2 \rangle$. Let $\mathcal{A}_i$ be the set of labels from relation $\overset{\ell_i}{\longmapsto}$ ($i = 1,2$) and let mapping $\{\!|\cdot|\!\} : \mathcal{A}_1 \to \mathcal{A}_2$.

We say that $\langle [\![\cdot]\!], (\!|\cdot|\!) \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ is a *operational corresponce* if it satisfies the property: If $\Gamma; \emptyset; \Delta \vdash_1 P \triangleright \diamond$ then

1. Completeness: If $\Gamma; \Delta \vdash_1 P \overset{\ell_1}{\longmapsto}_1 \Delta' \vdash_1 P'$ then
   $\exists Q, \Delta''$ s.t. (i) $(\!|\Gamma|\!); (\!|\Delta|\!) \vdash_2 [\![P]\!] \overset{\ell_2}{\Longmapsto}_2 (\!|\Delta''|\!) \vdash_2 Q$ (ii) $\ell_2 = \{\!|\ell_1|\!\}$, and
   (ii) $(\!|\Gamma|\!); (\!|\Delta''|\!) \vdash_2 Q \approx_2 (\!|\Delta'|\!) \vdash_2 [\![P']\!]$.
2. Soundness: If $(\!|\Gamma|\!); (\!|\Delta|\!) \vdash_2 [\![P]\!] \overset{\ell_2}{\Longmapsto}_2 (\!|\Delta'|\!) \vdash_2 Q$ then
   $\exists P', \Delta''$ s.t. (i) $\Gamma; \Delta \vdash_1 P \overset{\ell_1}{\longmapsto}_1 \Delta'' \vdash_1 P'$ (ii) $\ell_2 = \{\!|\ell_1|\!\}$, and
   (ii) $(\!|\Gamma|\!); (\!|\Delta''|\!) \vdash_2 [\![P']\!] \approx_2 (\!|\Delta'|\!) \vdash_2 Q$.

Proposition 17. *Consider typed calculi* $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \overset{\ell_1}{\longmapsto}_1, \approx_1, \vdash_1 \rangle$ *and* $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \overset{\ell_2}{\longmapsto}_2, \approx_2, \vdash_2 \rangle$. *Let* $\mathcal{A}_i$ *be the set of labels from relation* $\overset{\ell_i}{\longmapsto}$ ($i = 1,2$) *and let mapping* $\{\!|\cdot|\!\} : \mathcal{A}_1 \to \mathcal{A}_2$.

If $\{\!|\tau|\!\} = \tau$ *then the requirements of Def. 25 imply the requirements of Def. 14(2).*

Proof. The proof is trivial by substituting the τ action on labels ℓ_1 and ℓ_2 in Def. 25 to get Def. 14(2).

C.1 Properties for encoding $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_{\mathsf{HO}}$

In this section we prove Thm. 3, in Page 16 that requires that encoding $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_{\mathsf{HO}}$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated in Prop. 18.
- Operational Correspondence, stated in Prop. 19. Note that we prove a stronger operational correspondence condition, as in Def. 25, than the condition suggested in Def. 14(2).
- Full Abstraction, stated in Prop. 20.

Proposition 18 (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:

$$\frac{\Gamma;\emptyset;\Delta\cdot k : S \vdash P' \triangleright \diamond \quad \Gamma;\emptyset;\{k' : S_1\} \vdash k' \triangleright S_1}{\Gamma;\emptyset;\Delta\cdot k' : S_1 \cdot k :!\langle S_1\rangle;S \vdash k!\langle k'\rangle.P' \triangleright \diamond}$$

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:

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{(\!|\Gamma|\!)^1;\{x : (\!|S_1|\!)^1 \multimap \diamond\};\emptyset \vdash x \triangleright (\!|S_1|\!)^1 \multimap \diamond \qquad (\!|\Gamma|\!)^1;\emptyset;\{k' : (\!|S_1|\!)^1\} \vdash k' \triangleright (\!|S_1|\!)^1}{(\!|\Gamma|\!)^1;\{x : (\!|S_1|\!)^1 \multimap \diamond\};k' : (\!|S_1|\!)^1 \vdash xk' \triangleright \diamond}}{(\!|\Gamma|\!)^1;\{x : (\!|S_1|\!)^1 \multimap \diamond\};k' : (\!|S_1|\!)^1 \cdot z : \mathsf{end} \vdash xk' \triangleright \diamond}}{(\!|\Gamma|\!)^1;\emptyset;k' : (\!|S_1|\!)^1 \cdot z :?((\!|S_1|\!)^1 \multimap \diamond);\mathsf{end} \vdash z?(x).(xk') \triangleright \diamond}}{(\!|\Gamma|\!)^1;\emptyset;k' : (\!|S_1|\!)^1 \vdash \lambda z.z?(x).(xk') \triangleright U_1} \quad (5)$$

$$\frac{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k : (\!|S|\!)^1 \vdash [\![P']\!]^1 \triangleright \diamond \qquad (\!|\Gamma|\!)^1;\emptyset;k' : (\!|S_1|\!)^1 \vdash \lambda z.z?(x).(xk') \triangleright U_1 \quad (5)}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k' : (\!|S_1|\!)^1 \cdot k :!\langle U_1\rangle;(\!|S|\!)^1 \vdash k!\langle \lambda z.z?(x).(xk')\rangle.[\![P']\!]^1 \triangleright \diamond}$$

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

$$(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k : (\!|S|\!)^1\cdot x : (\!|S_1|\!)^1 \vdash [\![Q]\!]^1 \triangleright \diamond$$

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:

$$\frac{(\!|\Gamma|\!)^1;\{X : U_1\};\emptyset \vdash X \triangleright U_1 \qquad (\!|\Gamma|\!)^1;\emptyset;\cdot s :?((\!|S_1|\!)^1 \multimap \diamond);\mathsf{end} \vdash s \triangleright ?((\!|S_1|\!)^1 \multimap \diamond);\mathsf{end}}{(\!|\Gamma|\!)^1;\{X : U_1\};\cdot s :?((\!|S_1|\!)^1 \multimap \diamond);\mathsf{end} \vdash x\,s \triangleright \diamond} \quad (6)$$

$$\frac{\dfrac{(\!|\Gamma|\!)^1;\emptyset;\emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!|\Gamma|\!)^1;\emptyset;\overline{s} : \mathsf{end} \vdash \mathbf{0} \triangleright \diamond} \qquad \dfrac{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k : (\!|S|\!)^1 x : (\!|S_1|\!)^1 \vdash [\![Q]\!]^1 \triangleright \diamond}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k : (\!|S|\!)^1 \vdash \lambda x.[\![Q]\!]^1 \triangleright (\!|S_1|\!)^1 \multimap \diamond}}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k : (\!|S|\!)^1\cdot \overline{s} :!\langle(\!|S_1|\!)^1 \multimap \diamond\rangle;\mathsf{end} \vdash \overline{s}!\langle\lambda x.[\![Q]\!]^1\rangle.\mathbf{0} \triangleright \diamond} \quad (7)$$

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1;\{X:U_1\};\cdot s:?((\!|S_1|\!)^1\multimap\diamond);\mathsf{end}\vdash x\,s\triangleright\diamond \quad (6)}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k:(\!|S|\!)^1\cdot\overline{s}:!\langle(\!|S_1|\!)^1\multimap\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda x.[\![Q]\!]^1\rangle.\mathbf{0}\triangleright\diamond \quad (7)}}{(\!|\Gamma|\!)^1;\{X:U_1\};(\!|\Delta|\!)^1\cdot k:(\!|S|\!)^1\cdot s:?((\!|S_1|\!)^1\multimap\diamond);\mathsf{end}\cdot\overline{s}:!\langle(\!|S_1|\!)^1\multimap\diamond\rangle;\mathsf{end}\vdash x\,s\mid\overline{s}!\langle\lambda x.[\![Q]\!]^1\rangle.\mathbf{0}\triangleright\diamond}\ (8)$$

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1;\{X:U_1\};(\!|\Delta|\!)^1\cdot k:(\!|S|\!)^1\cdot s:?((\!|S_1|\!)^1\multimap\diamond);\mathsf{end}\cdot\overline{s}:!\langle(\!|S_1|\!)^1\multimap\diamond\rangle;\mathsf{end}\vdash x\,s\mid\overline{s}!\langle\lambda x.[\![Q]\!]^1\rangle.\mathbf{0}\triangleright\diamond \quad (8)}{(\!|\Gamma|\!)^1;\{X:U_1\};(\!|\Delta|\!)^1\cdot k:(\!|S|\!)^1\vdash(\nu\,s)(x\,s\mid\overline{s}!\langle\lambda x.[\![Q]\!]^1\rangle.\mathbf{0})\triangleright\diamond}}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1\cdot k:?(U_1);(\!|S|\!)^1\vdash k?(x).(\nu\,s)(x\,s\mid\overline{s}!\langle\lambda x.[\![Q]\!]^1\rangle.\mathbf{0})\triangleright\diamond}$$

In the second sub-case, x stands for a shared name. Then we have the following typing in the source language:

$$\dfrac{\Gamma\cdot x:\langle S_1\rangle;\emptyset;\Delta\cdot k:S\vdash Q\triangleright\diamond}{\Gamma;\emptyset;\Delta\cdot k:?(\langle S_1\rangle);S\vdash k?(x).Q\triangleright\diamond}$$

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 $[\![X]\!]^1_f$ 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 $\Gamma=\Gamma'\cdot X:\tilde{T}\to\diamond$, where

$$\tilde{T}=(\tilde{S},S^*)\qquad S^*=?(A);\mathsf{end}\qquad A=\mu\mathrm{t}.(\tilde{S},?(\mathrm{t});\mathsf{end})$$

$$\dfrac{\dfrac{}{\Gamma;\emptyset;\emptyset\vdash z_X\triangleright\tilde{T}\to\diamond}\quad\dfrac{\Gamma;\emptyset;\{n_i:S_i\}\vdash n_i\triangleright S_i}{\Gamma;\emptyset;\{s:S^*\}\vdash s\triangleright S^*}}{\Gamma;\emptyset;\Delta_{\tilde{n}},s:?(\tilde{T}\to\diamond);\mathsf{end}\vdash z_X(\tilde{n},s)\triangleright\diamond}\qquad(9)$$

$$\dfrac{\dfrac{\Gamma;\emptyset;\emptyset\vdash\mathbf{0}\triangleright\diamond}{\Gamma;\emptyset;\overline{s}:\mathsf{end}\vdash\mathbf{0}\triangleright\diamond}\quad\dfrac{\dfrac{\dfrac{\Gamma;\emptyset;\{x_i:S_i\}\vdash x_i\triangleright S_i}{\Gamma;\emptyset;\{z:S^*\}\vdash z\triangleright S^*}\quad\dfrac{}{\Gamma;\emptyset;\emptyset\vdash z_X\triangleright\tilde{T}\to\diamond}}{\Gamma;\emptyset;\Delta_{\tilde{x}},z:S^*\vdash z_X(\tilde{x},z)\triangleright\diamond}}{\Gamma;\emptyset;\emptyset\vdash\lambda(\tilde{x},z).\ z_X(\tilde{x},z)\triangleright\tilde{T}\to\diamond}}{\Gamma;\emptyset;\overline{s}:!\langle\tilde{T}\to\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},z).\ z_X(\tilde{x},z)\rangle.\mathbf{0}\triangleright\diamond}\qquad(10)$$

$$\dfrac{\dfrac{\Gamma;\emptyset;\Delta_{\tilde{n}},s:?(\tilde{T}\to\diamond);\mathsf{end}\vdash z_X(\tilde{n},s)\triangleright\diamond\quad(9)\qquad\Gamma;\emptyset;\overline{s}:!\langle\tilde{T}\to\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},z).\ z_X(\tilde{x},z)\rangle.\mathbf{0}\triangleright\diamond\quad(10)}{\Gamma;\emptyset;\Delta_{\tilde{n}},s:?(\tilde{T}\to\diamond);\mathsf{end},\overline{s}:!\langle\tilde{T}\to\diamond\rangle;\mathsf{end}\vdash z_X(\tilde{n},s)\mid\overline{s}!\langle\lambda(\tilde{x},z).\ x(\tilde{x},z)\rangle.\mathbf{0}\triangleright\diamond}}{\Gamma;\emptyset;\Delta_{\tilde{n}}\vdash(\nu\,s)(z_X(\tilde{n},s)\mid\overline{s}!\langle\lambda(\tilde{x},z).\ z_X(\tilde{x},z)\rangle.\mathbf{0})\triangleright\diamond}$$

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 $[\![P]\!]^1_{f,\{X\to\tilde{n}\}}$ and $\tilde{x}$ to stand for $[\![\mathsf{ofn}(P)]\!]$.

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1\cdot z_X:\tilde{T}\to\diamond;\emptyset;(\!|\Delta_{\tilde{n}}|\!)^1\vdash R\triangleright\diamond}{(\!|\Gamma|\!)^1\cdot z_X:\tilde{T}\to\diamond;\emptyset;(\!|\Delta_{\tilde{n}}|\!)^1,s:\mathsf{end}\vdash R\triangleright\diamond}}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_{\tilde{n}}|\!)^1,s:?(\tilde{T}\to\diamond);\mathsf{end}\vdash s?(z_X).R\triangleright\diamond} \tag{11}$$

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1;\emptyset;\emptyset\vdash\mathbf{0}\triangleright\diamond}{(\!|\Gamma|\!)^1;\emptyset;\overline{s}:\mathsf{end}\vdash\mathbf{0}\triangleright\diamond} \quad \dfrac{\dfrac{\dfrac{(\!|\Gamma|\!)^1\cdot z_X:\tilde{T}\to\diamond;\emptyset;(\!|\Delta_{\tilde{x}}|\!)^1\vdash\{\!\{R\}\!\}^{\emptyset}\triangleright\diamond}{(\!|\Gamma|\!)^1\cdot z_X:\tilde{T}\to\diamond;\emptyset;(\!|\Delta_{\tilde{x}}|\!)^1,y:\mathsf{end}\vdash\{\!\{R\}\!\}^{\emptyset}\triangleright\diamond}}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_{\tilde{x}}|\!)^1,y:?(A);\mathsf{end}\vdash y?(z_X).\{\!\{R\}\!\}^{\emptyset}\triangleright\diamond}}{(\!|\Gamma|\!)^1;\emptyset;\emptyset\vdash\lambda(\tilde{x},y).\,y?(z_X).\{\!\{R\}\!\}^{\emptyset}\triangleright\tilde{T}\to\diamond}}{(\!|\Gamma|\!)^1;\emptyset;\overline{s}:!\langle\tilde{T}\to\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},y).\,y?(z_X).\{\!\{R\}\!\}^{\emptyset}\rangle.\mathbf{0}\triangleright\diamond} \tag{12}$$

$$\dfrac{\begin{array}{c}(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_{\tilde{n}}|\!)^1,s:?(\tilde{T}\to\diamond);\mathsf{end}\vdash s?(z_X).R\triangleright\diamond \qquad (11)\\ (\!|\Gamma|\!)^1;\emptyset;\overline{s}:!\langle\tilde{T}\to\diamond\rangle;\mathsf{end}\vdash\overline{s}!\langle\lambda(\tilde{x},y).\,y?(z_X).\{\!\{R\}\!\}^{\emptyset}\rangle.\mathbf{0}\triangleright\diamond \quad (12)\end{array}}{\dfrac{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_{\tilde{n}}|\!)^1,s:?(\tilde{T}\to\diamond);\mathsf{end},\overline{s}:!\langle\tilde{T}\to\diamond\rangle;\mathsf{end}\vdash s?(z_X).R\mid\overline{s}!\langle\lambda(\tilde{x},y).\,y?(z_X).\{\!\{R\}\!\}^{\emptyset}\rangle.\mathbf{0}\triangleright\diamond}{(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_{\tilde{n}}|\!)^1\vdash(\nu\,s)(s?(z_X).R\mid\overline{s}!\langle\lambda(\tilde{x},y).\,y?(z_X).\{\!\{R\}\!\}^{\emptyset}\rangle.\mathbf{0})\triangleright\diamond}}$$

□

Before we prove operational correspondence we define mapping from $\{\!|\cdot|\!\}^1:\mathcal{A}\to\mathcal{A}$ where $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$:

Definition 26 ($\{\!|\cdot|\!\}^1:\mathcal{A}\to\mathcal{A}$). Let $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$ then we define:

$$\{\!|(\nu\tilde{m})n!\langle m\rangle|\!\}^1\stackrel{\mathsf{def}}{=}(\nu\tilde{m})n!\langle\lambda z.\,z?(x).(x\,m)\rangle\qquad\{\!|n?\langle m\rangle|\!\}^1\stackrel{\mathsf{def}}{=}n?\langle\lambda z.\,z?(x).(x\,m)\rangle$$
$$\{\!|(\nu\tilde{m})n!\langle\lambda x.P\rangle|\!\}^1\stackrel{\mathsf{def}}{=}(\nu\tilde{m})n!\langle\lambda x.[\![P]\!]^1_{\emptyset}\rangle$$
$$\{\!|n?\langle\lambda x.P\rangle|\!\}^1\stackrel{\mathsf{def}}{=}n?\langle\lambda x.[\![P]\!]^1_{\emptyset}\rangle\qquad\{\!|n\oplus l|\!\}^1\stackrel{\mathsf{def}}{=}n\oplus l\qquad\{\!|n\&l|\!\}^1\stackrel{\mathsf{def}}{=}n\&l\qquad\{\!|\tau|\!\}^1\stackrel{\mathsf{def}}{=}\tau$$

We now state and prove a detailed version of the operational corresponce in Def. 25.

Proposition 19 (Operational Correspondence, HOπ into HO). Let P be a HOπ process. If $\Gamma;\emptyset;\Delta\vdash P\triangleright\diamond$ then:

1. Suppose $\Gamma;\Delta\vdash P\overset{\ell_1}{\longmapsto}\Delta'\vdash P'$. 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.
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1\vdash[\![P]\!]^1_f\overset{\ell_2}{\longmapsto}(\!|\Delta'|\!)^1\vdash[\![P']\!]^1_f$ 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.
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1\vdash[\![P]\!]^1_f\overset{\ell_2}{\longmapsto}(\!|\Delta'|\!)^1\vdash[\![P_0]\!]^1_f\{^{\lambda y.[\![Q]\!]^1_{\emptyset}}/_x\}$ 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.
      $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1\vdash[\![P]\!]^1_f\overset{\ell_2}{\longmapsto}(\!|\Delta'|\!)^1\vdash R$, with $\ell_2=\{\!|\ell_1|\!\}^1$,
      and $(\!|\Gamma|\!)^1;(\!|\Delta'|\!)^1\vdash R\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto}(\!|\Delta'|\!)^1\vdash[\![P_0]\!]^1_f\{^m/_x\}$.

d) If $\ell_1 = \tau$ and $P' \equiv (\nu\,\tilde{m})(P_1 \mid P_2\{^m/_x\})$ then $\exists R$ s.t.
$(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R)$, and
$(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R) \overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^m/_x\})$.

e) If $\ell_1 = \tau$ and $P' \equiv (\nu\,\tilde{m})(P_1 \mid P_2\{^{\lambda y.\,Q}/_x\})$ then
$(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta_1|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^{\lambda y.\,[\![Q]\!]^1_\emptyset}/_x\})$.

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
$(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta'_1|\!)^1 \vdash [\![P']\!]^1_f$.

2. Suppose $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash Q$. 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.
$\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'$, $\ell_1 = \{\!|\ell_2|\!\}^1$, 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.
$\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'\{^{\lambda y.\,P''}/_x\}$, $\ell_1 = \{\!|\ell_2|\!\}^1$, $[\![P'']\!]^1_\emptyset = R$, and $Q = [\![P']\!]^1_f$.

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

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

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

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

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:

$$\Gamma;\Delta_0 \cdot n{:}S \cdot s : !\langle S\rangle;S_1 \vdash s!\langle n\rangle.P' \overset{\ell_1}{\longmapsto} \Delta_0 \cdot s{:}S_1 \vdash P'$$

The encoding of the source judgment for P is as follows:

$$(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0 \cdot n{:}S \cdot s : !\langle S\rangle;S_1|\!)^1 \vdash [\![s!\langle n\rangle.P']\!]^1 \triangleright \diamond$$

which, using Def. 17 can be expressed as

$$(\!|\Gamma|\!)^{\mathsf{p}};\emptyset;(\!|\Delta_0|\!) \cdot n{:}(\!|S|\!)^1 \cdot s : !\langle ?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond\rangle;(\!|S_1|\!)^1 \vdash s!\langle\lambda z.\, z?(x).(x\,n)\rangle.[\![P']\!]^1 \triangleright \diamond$$

Now, $\{\!|\ell_1|\!\}^1 = s!\langle \lambda z.\, z?(x).xn \rangle$. We may infer the following transition for $[\![P]\!]^1$:

$$\begin{array}{rl} & (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta|\!)^1 \vdash s!\langle \lambda z.\, z?(x).(xn)\rangle.[\![P']\!]^1 \triangleright \diamond \\ \overset{\{\!|\ell_1|\!\}^1}{\longmapsto} & (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0|\!)^1 \cdot s : (\!|S_1|\!)^1 \vdash [\![P']\!]^1 \triangleright \diamond \\ = & (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0 \cdot s : S_1|\!)^1 \vdash [\![P']\!]^1 \triangleright \diamond \end{array}$$

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:

$$\begin{array}{l} (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0 \cdot n : ?(S);S_1|\!)^1 \vdash [\![P]\!]^1 \triangleright \diamond \\ = (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0|\!)^1 \cdot n : ?(?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond);(\!|S_1|\!)^1 \vdash n?(x).(\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.[\![P']\!]^1\rangle.\mathbf{0}) \triangleright \diamond \end{array}$$

Now, $\{\!|\ell_1|\!\}^1 = n?\langle \lambda z.\, z?(x).(xm)\rangle$ and it is immediate to infer the following transition for $[\![P]\!]^1$:

$$\begin{array}{l} (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0|\!)^1 \cdot n : ?(?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond);(\!|S_1|\!)^1 \vdash n?(x).(\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.[\![P']\!]^1\rangle.\mathbf{0}) \triangleright \diamond \\ \overset{\{\!|\ell_1|\!\}^1}{\longmapsto} (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0|\!)^1 \cdot n : (\!|S_1|\!)^1 \cdot m : (\!|S|\!)^1 \vdash (\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.[\![P']\!]^1\rangle.\mathbf{0})\{^{\lambda z.\, z?(x).(xm)}/_x\} \triangleright \diamond \end{array}$$

Let us write R to stand for process $(\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.[\![P']\!]^1\rangle.\mathbf{0})\{^{\lambda z.\, z?(x).(xm)}/_x\}$. We then have:

$$\begin{array}{rl} R & \overset{\tau}{\longrightarrow} (\nu\, s)(s?(x).(xm) \mid \overline{s}!\langle \lambda x.[\![P']\!]^1\rangle.\mathbf{0}) \\ & \overset{\tau}{\longrightarrow} (\lambda x.[\![P']\!]^1)\, m \mid \mathbf{0} \\ & \overset{\tau}{\longrightarrow} [\![P']\!]^1\{^m/_x\} \end{array}$$

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 $\ell_2 = n!\langle \lambda z.\, z?(x).(xm)\rangle$ (the case $\ell_2 = (\nu\, m)n!\langle \lambda z.\, z?(x).(xm)\rangle$ is similar). Then we have:

$$(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_0|\!)^1 \cdot n : !\langle ?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond\rangle;(\!|S_1|\!)^1 \vdash n!\langle \lambda z.\, z?(x).(xm)\rangle.[\![P']\!]^1 \triangleright \diamond$$

for some S, S_1, and Δ_0. We may infer the following typed transition for $[\![P]\!]^1$:

$$\begin{array}{l} (\!|\Gamma|\!)^1;(\!|\Delta_0|\!)^1 \cdot n : !\langle ?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond\rangle;(\!|S_1|\!)^1 \vdash n!\langle \lambda z.\, z?(x).(xm)\rangle.[\![P']\!]^1 \\ \overset{\ell_2}{\longmapsto} (\!|\Gamma|\!)^1;(\!|\Delta_0|\!)^1 \cdot n : (\!|S_1|\!)^1 \vdash [\![P']\!]^1 \end{array}$$

Now, in the source term P we can infer the following transition

$$\Gamma; \Delta_0 \cdot n : !\langle S \rangle; S_1 \vdash n!\langle m \rangle.P' \overset{n!\langle m \rangle}{\longmapsto} \Gamma; \Delta_0 \cdot n : S_1 \vdash P'$$

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

$$(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta_0|\!)^1 \cdot n : ?(?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond); (\!|S_1|\!)^1 \vdash n?(x).(\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1 \rangle.\mathbf{0}) \triangleright \diamond$$

for some S, S_1, Δ_0. We may infer the following typed transitions for $[\![P]\!]^1$:

$$\begin{aligned}
& (\!|\Gamma|\!)^1; (\!|\Delta_0|\!)^1 \cdot n : ?(?((\!|S|\!)^1 \multimap \diamond); \mathsf{end} \multimap \diamond); (\!|S_1|\!)^1 \vdash n?(x).(\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1 \rangle.\mathbf{0}) \\
\overset{\ell_2}{\longmapsto}\ & (\!|\Gamma|\!)^1; (\!|\Delta_0|\!)^1 \cdot n : (\!|S_1|\!)^1 \cdot m : (\!|S_1|\!)^1 \vdash (\nu\, s)((x\, s) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1 \rangle.\mathbf{0})\{^{\lambda z.\, z?(x).x\, m}/_x\} \\
=\ & (\!|\Gamma|\!)^1; (\!|\Delta_0|\!)^1 \cdot n : (\!|S_1|\!)^1 \cdot m : (\!|S|\!)^1 \vdash (\nu\, s)(s?(x).(x\, m) \mid \overline{s}!\langle \lambda x.\, [\![P']\!]^1 \rangle.\mathbf{0}) \\
\overset{\tau}{\longmapsto}\ & (\!|\Gamma|\!)^1; (\!|\Delta_0|\!)^1 \cdot n : (\!|S_1|\!)^1 \cdot m : (\!|S|\!)^1 \vdash (\lambda x.\, [\![P']\!]^1)\, m \\
\overset{\tau}{\longmapsto}\ & (\!|\Gamma|\!)^1; (\!|\Delta_0|\!)^1 \cdot n : (\!|S_1|\!)^1 \cdot m : (\!|S|\!)^1 \vdash [\![P']\!]^1\{^m/_x\}
\end{aligned}$$

Now, in the source term P we can infer the following transition

$$\Gamma; \Delta_0 \cdot n : ?(S); S_1 \vdash n?(x).P' \overset{n?\langle m \rangle}{\longmapsto} \Gamma; \Delta_0 \cdot n : S_1 \cdot m : S \vdash P'\{^m/_x\}$$

and the thesis follows.

□

Proposition 20 (Full Abstraction, HOπ into HO). $\Gamma; \Delta_1 \vdash P_1 \approx \Delta_2 \vdash Q_1$ if and only if $(\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \approx (\!|\Delta_2|\!)^1 \vdash [\![Q_2]\!]^1_f$.

Proof. **Proof of Soundness Direction.**
Let

$$\mathfrak{R} = \{\Gamma; \Delta_1 \vdash P_1 \approx \Delta_2 \vdash Q_1 \ \mid\ (\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \approx (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]^1_f\}$$

The proof considers a case analysis on the transition $\overset{\ell}{\longmapsto}$ and uses the soundness direction of operational correspondence (cf. Prop. 19). We give an interesting case. The others are similar of easier.
- Case: $\ell = (\nu\, \tilde{m_1}')n!\langle m_1 \rangle$.
Prop. 19 implies that

$$\Gamma; \Delta_1 \vdash P_1 \overset{(\nu\, \tilde{m_1}')n!\langle m_1 \rangle}{\longmapsto} \Delta_1' \vdash P_2$$

implies

$$(\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \overset{(\nu\, \tilde{m_1}')n!\langle \lambda z.\, z?(x).(x\, m_1) \rangle}{\longmapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]^1_f$$

that in combination with the definition of $\mathfrak{R}$ we get

$$(\!|\Gamma|\!)^1; (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]^1_f \overset{(\nu\, \tilde{m_2}')n!\langle \lambda z.\, z?(x).(x\, m_2) \rangle}{\Longmapsto} (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]^1_f \tag{13}$$

and

$$\begin{array}{rcl}(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_1'|\!)^1 & \approx & (\!|\Delta_2'|\!)^1 \vdash (\nu\tilde{m_1}')([\![P_2]\!]_f^1 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle x\rangle.\mathbf{0})s\lambda z.\, z?(x).(x\, m_1)) \\ & \approx & (\nu\tilde{m_2}')([\![Q_2]\!]_f^1 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle x\rangle.\mathbf{0})s\lambda z.\, z?(x).(x\, m_2))\end{array}$$

We rewrite the last result as

$$\begin{array}{rcl}(\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_1'|\!)^1 & \approx & (\!|\Delta_2'|\!)^1 \vdash [\![(\nu\tilde{m_1}')(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_1)]\!]_f^1 \\ & \approx & [\![(\nu\tilde{m_2}')(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_2)]\!]_f^1\end{array}$$

to conclude that

$$\begin{array}{rcl}\Gamma;\emptyset;\Delta_1' & \mathfrak{R} & \Delta_2' \vdash (\nu\tilde{m_1}')(P_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_1) \\ & \mathfrak{R} & (\nu\tilde{m_2}')(Q_2 \mid t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_2)\end{array}$$

as required

Proof of Completeness Direction.

Let

$$\mathfrak{R} = \{(\!|\Gamma|\!)^1;(\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]_f^1, (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]_f^1 \;\mid\; \Gamma;\Delta_1 \vdash P_1 \approx \Delta_2 \vdash Q_1\}$$

We show that $\mathfrak{R} \subset \approx$ 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 Prop. 19 implies that

$$(\!|\Gamma|\!)^1;(\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]_f^1 \overset{\ell}{\longmapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]_f^1$$

implies

$$\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$$

From the latter transition and the definition of $\mathfrak{R}$ we imply

$$\Gamma;\Delta_2 \vdash Q_1 \overset{\ell}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{14}$$

$$\Gamma;\Delta_1' \vdash P_2 \approx \Delta_2' \vdash Q_2 \tag{15}$$

From 14 and Prop. 19 we get

$$(\!|\Gamma|\!)^1;(\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]_f^1 \overset{\ell}{\Longmapsto} (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]_f^1$$

Furthermore, from 15 and the definition of $\mathfrak{R}$ we get

$$(\!|\Gamma|\!)^1;(\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]_f^1 \;\mathfrak{R}\; (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]_f^1$$

as required.

- Case: $\ell = (\nu\tilde{m})n!\langle\lambda x.\, P\rangle$

There are two subcases:

-Subcase:

The proof of Prop. 19 implies that

$$(\!|\Gamma|\!)^1;(\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]_f^1 \overset{\ell}{\longmapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]_f^1$$

implies

$$\Gamma;\Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$$

where the proof is similar with the previous case.
- Subcase:
The proof of Prop. 19 implies that

$$(\!|\Gamma|\!)^1;(\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]_f^1 \overset{(\nu \tilde{m_1}')n!\langle\lambda z.z?(x).(x m_1)\rangle}{\longmapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]_f^1$$

implies

$$\Gamma;\Delta_1 \vdash P_1 \overset{(\nu \tilde{m_1}')n!\langle m_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$$

From the latter transition and the definition of $\mathfrak{R}$ we imply

$$\Gamma;\Delta_2 \vdash Q_1 \overset{(\nu \tilde{m_2}')n!\langle m_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{16}$$

and

$$\begin{aligned} &\Gamma;\emptyset;\Delta_1' \vdash (\nu \tilde{m_1}')(P_2 \mid t?(x).(\nu s)(x\,s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_1) \\ &\quad\approx \quad \Delta_2' \vdash (\nu \tilde{m_2}')(Q_2 \mid t?(x).(\nu s)(x\,s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_2) \end{aligned} \tag{17}$$

From (16) and Prop. 19 we get

$$(\!|\Gamma|\!)^1;(\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]_f^1 \overset{(\nu \tilde{m_2}')n!\langle\lambda z.z?(x).(x m_2)\rangle}{\Longmapsto} (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]_f^1$$

Furthermore, from (17) and the definition of $\mathfrak{R}$ we get

$$\begin{array}{ccc} (\!|\Gamma|\!)^1;\emptyset;(\!|\Delta_1'|\!)^1 & \mathfrak{R} & (\!|\Delta_2'|\!)^1 \vdash [\![(\nu \tilde{m_1}')(P_2 \mid t?(x).(\nu s)(x\,s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_1)]\!]_f^1 \\ & \mathfrak{R} & [\![(\nu \tilde{m_2}')(Q_2 \mid t?(x).(\nu s)(x\,s \mid \overline{s}!\langle x\rangle.\mathbf{0})sm_2)]\!]_f^1 \end{array}$$

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 Prop. 19 implies that

$$(\!|\Gamma|\!)^1;(\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]_f^1 \overset{n?\langle\lambda z.z?(x).(x s)\rangle}{\longmapsto} (\!|\Delta_1''|\!)^1 \vdash R$$

implies

$$\Gamma;\Delta_1 \vdash P_1 \overset{n?\langle m_1\rangle}{\longmapsto} \Delta_1' \vdash P_2 \tag{18}$$

and

$$(\!|\Gamma|\!)^1;(\!|\Delta_1''|\!)^1 \vdash R \overset{\tau_s}{\longmapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]_f^1 \tag{19}$$

From the transition (18) and the definition of $\mathfrak{R}$ we imply

$$\Gamma;\Delta_2 \vdash Q_1 \overset{n?\langle m_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{20}$$

$$\Gamma;\Delta_1' \vdash P_2 \approx \Delta_2' \vdash Q_2 \tag{21}$$

From (20) and Prop. 19 we get

$$(\!|\Gamma|\!)^1;(\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]_f^1 \overset{n?\langle\lambda z.z?(x).(x\,s)\rangle}{\Longmapsto} (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]_f^1$$

Furthermore, from 21 and the definition of $\mathfrak{R}$ we get

$$(\!|\Gamma|\!)^1;(\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]_f^1 \;\mathfrak{R}\; (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]_f^1$$

If we consider result (19) we get:

$$(\!|\Gamma|\!)^1;(\!|\Delta_1''|\!)^1 \vdash R \overset{\tau_s}{\longmapsto} \;\mathfrak{R}\; (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]_f^1$$

where following Lem. 2 we show that R is a bisimulation an up to $\overset{\tau_s}{\Longmapsto}$. □

C.2 Properties for encoding $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_\pi$

In this section we prove Thm. 4, in Page 17 that requires that encoding $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_\pi$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated in Prop. 21.
- Operational Correspondence, stated in Prop. 22. Note that we prove a stronger operational correspondence condition, as in Def. 25, than the condition suggested in Def. 14(2).
- Full Abstraction, stated in Prop. 23.

Proposition 21 (Type Preservation, $\mathsf{HO}\pi$ into π). Let P be a $\mathsf{HO}\pi$ 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 $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$.

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:

$$\frac{\Gamma;\emptyset;\Delta_1 \cdot k : S \vdash P \triangleright \diamond \quad \dfrac{\Gamma;\emptyset;\Delta_2 \cdot x : S_1 \vdash Q \triangleright \diamond}{\Gamma;\emptyset;\Delta_2 \vdash \lambda x.Q \triangleright S_1 \multimap \diamond}}{\Gamma;\emptyset;\Delta_1 \cdot \Delta_2 \cdot k :!\langle S_1 \multimap \diamond\rangle; S \vdash k!\langle\lambda x.Q\rangle.P \triangleright \diamond}$$

   This way, by IH we have

$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta_2|\!)^2, x : (\!|S_1|\!)^2 \vdash [\![Q]\!]^2 \triangleright \diamond$$

Let us write U_1 to stand for $\langle ?((\!|S_1|\!)^2); \mathsf{end}\rangle$. The corresponding typing in the target language is as follows:

$$(\!|\Gamma_1|\!)^2 = (\!|\Gamma|\!)^2 \cup a : \langle ?((\!|S_1|\!)^2); \mathsf{end}\rangle$$
$$(\!|\Gamma_2|\!)^2 = (\!|\Gamma_1|\!)^2 \cup X : (\!|\Delta_2|\!)^2$$

Also $(*)$ stands for $(\!|\Gamma_1|\!)^2; \emptyset; \emptyset \vdash a \triangleright U_1$; $(**)$ stands for $(\!|\Gamma_2|\!)^2; \emptyset; \emptyset \vdash a \triangleright U_1$; and $(***)$ stands for $(\!|\Gamma_2|\!)^2; \emptyset; \emptyset \vdash X \triangleright \diamond$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\!|\Gamma_2|\!)^2; \emptyset; (\!|\Delta_2|\!)^2, x : (\!|S_1|\!)^2 \vdash [\![Q]\!]^2 \triangleright \diamond}}{(\!|\Gamma_2|\!)^2; \emptyset; (\!|\Delta_2|\!)^2, y : \mathsf{end}, x : (\!|S_1|\!)^2 \vdash [\![Q]\!]^2 \triangleright \diamond}}{(\!|\Gamma_2|\!)^2; \emptyset; (\!|\Delta_2|\!)^2, y : ?((\!|S_1|\!)^2); \mathsf{end} \vdash y?(x).[\![Q]\!]^2 \triangleright \diamond} \quad \dfrac{}{(**)}}{(***) \qquad (\!|\Gamma_2|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \vdash a?(y).y?(x).[\![Q]\!]^2 \triangleright \diamond}}{(\!|\Gamma_2|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \vdash a?(y).y?(x).[\![Q]\!]^2 \mid X \triangleright \diamond}}{(\!|\Gamma_1|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \vdash \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond} \tag{22}$$

$$\dfrac{(\!|\Gamma_1|\!)^2; \emptyset; (\!|\Delta_1|\!)^2, k : (\!|S|\!)^2 \vdash [\![P]\!]^2 \triangleright \diamond \quad (\!|\Gamma_1|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \vdash \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond \quad (22)}{(\!|\Gamma_1|\!)^2; \emptyset; (\!|\Delta_1, \Delta_2|\!)^2, k : (\!|S|\!)^2 \vdash [\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond} \tag{23}$$

$$\dfrac{\dfrac{(\!|\Gamma_1|\!)^2; \emptyset; \emptyset \vdash a \triangleright U_1 \quad (\!|\Gamma_1|\!)^2; \emptyset; (\!|\Delta_1, \Delta_2|\!)^2, k : (\!|S|\!)^2 \vdash [\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X) \triangleright \diamond \quad (23)}{(\!|\Gamma_1|\!)^2; \emptyset; (\!|\Delta_1, \Delta_2|\!)^2, k : !\langle U_1\rangle; (\!|S|\!)^2 \vdash k!\langle a\rangle.([\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X)) \triangleright \diamond}}{(\!|\Gamma|\!)^2; \emptyset; (\!|\Delta_1, \Delta_2|\!)^2, k : !\langle U_1\rangle; (\!|S|\!)^2 \vdash (\nu a)(k!\langle a\rangle.([\![P]\!]^2 \mid \mu X.(a?(y).y?(x).[\![Q]\!]^2 \mid X))) \triangleright \diamond}$$

In the second case, $\lambda x.Q$ has a shared type. We have the following typing in the source language:

$$\dfrac{\Gamma; \emptyset; \Delta \cdot k : S \vdash P \triangleright \diamond \quad \dfrac{\dfrac{\Gamma; \emptyset; \cdot x : S_1 \vdash Q \triangleright \diamond}{\Gamma; \emptyset; \emptyset \vdash \lambda x.Q \triangleright S_1 \multimap \diamond}}{\Gamma; \emptyset; \emptyset \vdash \lambda x.Q \triangleright S_1 \to \diamond}}{\Gamma; \emptyset; \Delta \cdot k : !\langle S_1 \to \diamond\rangle; S \vdash k!\langle \lambda x.Q\rangle.P \triangleright \diamond}$$

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:

$$\dfrac{\Gamma \cdot x : S_1 \to \diamond; \emptyset; \Delta \cdot k : S \vdash P \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot k : ?(S_1 \to \diamond); S \vdash k?(x).P \triangleright \diamond}$$

The corresponding typing in the target language is as follows:

$$\dfrac{\dfrac{}{(\!|\Gamma|\!)^2 \cdot x : \langle ?((\!|S_1|\!)^2); \mathsf{end}\rangle; \emptyset; \Delta \cdot k : (\!|S|\!)^2 \vdash (\!|P|\!)^2 \triangleright \diamond}}{(\!|\Gamma|\!)^2; \emptyset; (\!|\Delta|\!)^2 \cdot k : ?(\langle ?((\!|S_1|\!)^2); \mathsf{end}\rangle); (\!|S|\!)^2 \vdash k?(x).[\![P]\!]^2 \triangleright \diamond}$$

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 $P = xk$. 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:

$$\frac{\Gamma; \{x : S_1 \multimap \diamond\}; \emptyset \vdash X \triangleright S_1 \multimap \diamond \quad \Gamma; \emptyset; \{k : S_1\} \vdash k \triangleright S_1}{\Gamma; \{x : S_1 \multimap \diamond\}; k : S_1 \vdash xk \triangleright \diamond}$$

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:

$$\frac{\dfrac{(\!|\Gamma_1|\!)^2; \emptyset; \emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!|\Gamma_1|\!)^2; \emptyset; \overline{s} : \mathsf{end} \vdash \mathbf{0} \triangleright \diamond} \quad (\!|\Gamma_1|\!)^2; \emptyset; \{k : (\!|S_1|\!)^2\} \vdash k \triangleright (\!|S_1|\!)^2}{(\!|\Gamma_1|\!)^2; \emptyset;\ k : (\!|S_1|\!)^2, \overline{s} : !\langle (\!|S_1|\!)^2\rangle; \mathsf{end} \vdash \overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond} \tag{24}$$

$$\frac{\dfrac{(\!|\Gamma_1|\!)^2; \emptyset;\ k : (\!|S_1|\!)^2, \overline{s} : !\langle (\!|S_1|\!)^2\rangle; \mathsf{end} \vdash \overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond \quad (24)}{(\!|\Gamma_1|\!)^2; \emptyset; \emptyset \vdash x \triangleright \langle !\langle (\!|S_1|\!)^2\rangle; \mathsf{end}\rangle}}{\dfrac{(\!|\Gamma_1|\!)^2; \emptyset; k : (\!|S_1|\!)^2, s : ?((\!|S_1|\!)^2); \mathsf{end}, \overline{s} : !\langle (\!|S_1|\!)^2\rangle; \mathsf{end} \vdash x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond}{(\!|\Gamma_1|\!)^2; \emptyset; k : (\!|S_1|\!)^2 \vdash (\nu\, s)(x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0}) \triangleright \diamond}}$$

In the second case, x is shared, and we have the following typing in the source language:

$$\frac{\Gamma \cdot x : S_1 \multimap \diamond; \emptyset; \emptyset \vdash x \triangleright S_1 \to \diamond \quad \Gamma; \emptyset; k : S_1 \vdash k \triangleright S_1}{\Gamma \cdot x : S_1 \to \diamond; \emptyset; k : S_1 \vdash xk \triangleright \diamond}$$

The associated typing in the target language is obtained similarly as in the first case. □

Before we prove operational correspondence we define mapping from $\{\!|\cdot|\!\}^2 : \mathcal{A} \to \mathcal{A}$ where $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$:

Definition 27 **($\{\!|\cdot|\!\}^2 : \mathcal{A} \to \mathcal{A}$).** Let $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$ then we define:

$$\{\!|(\nu \tilde{m})n!\langle \lambda x.P\rangle|\!\}^2 \overset{\mathtt{def}}{=} (\nu\, m)n!\langle m\rangle \qquad\qquad \{\!|n?\langle \lambda x.P\rangle|\!\}^2 \overset{\mathtt{def}}{=} n?\langle m\rangle \qquad m \text{ fresh}$$

and homomorphic for all other cases of $\ell \in \mathcal{A}$.

We now state and prove a detailed and extended version of the operational corresponce in Def. 25.

Proposition 22 (Operational Correspondence, HOπ into π). *Let P be an HOπ process such that $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$.*

1. *Suppose $\Gamma;\Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'$. Then we have:*
   a) *If $\ell_1 = (\nu\tilde{m})n!\langle\lambda x.Q\rangle$, then $\exists\Gamma',\Delta''$ where either:*
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\!|\ell_1|\!\}^2}{\longmapsto} \Gamma'\cdot(\!|\Gamma|\!)^2;(\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\!|\ell_1|\!\}^2}{\longmapsto} (\!|\Gamma|\!)^2;\Delta'' \vdash [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$
   b) *If $\ell_1 = n?\langle\lambda y.Q\rangle$ then $\exists R$ where either*
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\!|\ell_1|\!\}^2}{\longmapsto} \Gamma';(\!|\Delta''|\!)^2 \vdash R$, *for some Γ' and* $(\!|\Gamma|\!)^2;(\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx (\!|\Delta''|\!)^2 \vdash (\nu a)(R \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\!|\ell_1|\!\}^2}{\longmapsto} (\!|\Gamma|\!)^2;(\!|\Delta''|\!)^2 \vdash R$, *and* $(\!|\Gamma|\!)^2;(\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx (\!|\Delta''|\!)^2 \vdash (\nu s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$
   c) *If $\ell_1 = \tau$ then either:*
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash (\nu\tilde{m})([\![P_1]\!]^2 \mid (\nu a)([\![P_2]\!]^2\{a/x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2))$, *for some P_1, P_2, Q;*
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash (\nu\tilde{m})([\![P_1]\!]^2 \mid (\nu s)([\![P_2]\!]^2\{\overline{s}/x\} \mid s?(y).y?(x).[\![Q]\!]^2))$, *for some P_1, P_2, Q;*
      - $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Gamma|\!)^2;(\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$
   d) *If $\ell_1 = \tau_\beta$ then $(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau_s}{\longmapsto} (\!|\Gamma|\!)^2;(\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$*
   e) *If $\ell_1 \in \{n\oplus l, n\&l\}$ then*

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

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

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 Def. 18 we have that
$[\![P]\!]^2 = (\nu a)(n!\langle a\rangle.[\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2)$
Then

$$\Gamma;\emptyset;\Delta \vdash P \overset{n!\langle\lambda x.Q\rangle}{\longmapsto} \Delta' \vdash P'$$
$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{(\nu a)n!\langle a\rangle}{\longmapsto} (\!|\Delta|\!)^2 \vdash [\![P']\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2$$

and from Def. 18

$$\{\!|n!\langle\lambda x.Q\rangle|\!\} = (\nu a)n!\langle a\rangle$$

as required.
- Subcase: $P = n!\langle\lambda x.Q\rangle.P'$ and also from Def. 18 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 Def. 18 we have that $[\![P]\!]^2 = n?(x).[\![P']\!]^2$
Then

$$\Gamma;\emptyset;\Delta \vdash P \overset{n?\langle\lambda x.Q\rangle}{\longmapsto} \Delta' \vdash P'\{^{\lambda x.Q}/_x\}$$
$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{n?\langle a\rangle}{\longrightarrow} (\!|\Delta''|\!)^2 \vdash R\{^a/_x\}$$

with

$$\{\!|n?\langle\lambda x.Q\rangle|\!\}^2 = n?\langle a\rangle$$

It remains to show that

$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta'|\!)^2 \vdash [\![P'\{^{\lambda x.Q}/_x\}]\!]^2 \approx (\!|\Delta''|\!)^2 \vdash (\nu a)(R\{^a/_x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

The proof is an induction on the syntax structure of P'. Suppose $P' = xm$, then:

$$[\![xm\{^{\lambda x.Q}/_x\}]\!]^2 = [\![Q\{^m/_x\}]\!]^2$$
$$(\nu a)(R\{^a/_x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2) = (\nu a)((\nu s)(x!\langle s\rangle.\overline{s}!\langle m\rangle.\mathbf{0})\{^a/_x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

The second term can be deterministically reduced as:

$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta''|\!)^2 \overset{\tau}{\longmapsto}\overset{\tau_s}{\longmapsto} (\!|\Delta''|\!)^2 \vdash (\nu a)((\nu s)(x!\langle s\rangle.\overline{s}!\langle m\rangle.\mathbf{0})\{^a/_x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$
$$\overset{\tau}{\longmapsto}\overset{\tau_s}{\longmapsto} \qquad (\nu a)([\![Q\{^m/_x\}]\!]^2 \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

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:

$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta'|\!)^2 \vdash [\![(P_1 \mid P_2)\{^{\lambda x.Q}/_x\}]\!]^2 \approx (\!|\Delta''|\!)^2 \vdash (\nu a)([\![P_1 \mid P_2]\!]^2\{^a/_x\} \mid *\, a?(y).y?(x).[\![Q]\!]^2)$$

We know that

$$(\!|\Gamma|\!)^2;(\!|\Delta_1|\!)^2 \vdash [\![P_1\{^{\lambda x.\,Q}/_x\}]\!]^2 \approx (\!|\Delta_1''|\!)^2 \vdash (\nu a)([\![P_1]\!]^2\{^a/_x\} \mid * a?(y).y?(x).[\![Q]\!]^2)$$
$$(\!|\Gamma|\!)^2;(\!|\Delta_2|\!)^2 \vdash [\![P_2\{^{\lambda x.\,Q}/_x\}]\!]^2 \approx (\!|\Delta_1''|\!)^2 \vdash (\nu a)([\![P_2]\!]^2\{^a/_x\} \mid * a?(y).y?(x).[\![Q]\!]^2)$$

We conclude from the congruence of $\approx$.
- The rest of the cases for Part 1 are easy to follow using Def. 18.
- Part 2.
The proof for Part 2 is straightforward following Def. 18. We give some distinctive cases:
- Case $P = n!\langle \lambda x.\, Q\rangle.P'$

$$\frac{\Gamma;\Delta \vdash P \overset{n!\langle \lambda x.\, Q\rangle}{\longmapsto} \Delta' \vdash P'}{(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{(\nu a)n!\langle a\rangle}{\longmapsto} (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \mid * a?(y).y?(s).[\![Q]\!]^2}$$

as required.
- Case $P = n?(x).P'$

$$\frac{\Gamma;\Delta \vdash P \overset{n?\langle \lambda x.\, Q\rangle}{\longmapsto} \Delta' \vdash P'\{^{\lambda x.}/_Q\}x}{(\!|\Gamma|\!)^2;(\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{n?\langle a\rangle}{\longmapsto} (\!|\Delta''|\!)^2 \vdash [\![P']\!]^2\{^a/_x\}}$$

We now use a similar argumentation as the input case in Part 1 to prove that:

$$\Gamma;\Delta' \vdash P'\{^{\lambda x.\, Q}/_x\} \approx (\!|\Delta''|\!)^2 \vdash (\nu a)([\![P']\!]^2\{^a/_x\} \mid * a?(y).y?(x).[\![Q]\!]^2)$$

□

Proposition 23 (Full Abstraction, From $\mathsf{HO}\pi$ to π). *Let P_1, Q_1 be $\mathsf{HO}\pi$ processes. $\Gamma;\Delta_1 \vdash P_1 \approx^{\mathsf{H}} \Delta_2 \vdash Q_1$ if and only if $(\!|\Gamma|\!)^2;(\!|\Delta_1|\!)^2 \vdash [\![P_1]\!]^2 \approx^{\mathsf{C}} (\!|\Delta_2|\!)^2 \vdash [\![Q_1]\!]^2$.*

Proof. Proof follows directly from Prop. 22. The cases of Prop. 22 are used to create a bisimulation closure to prove the the soundness direction and a bisimulation up to determinate transition (Lem. 2) to prove the completeness direction. □

C.3 Properties for encoding $\mathcal{L}_{\mathsf{HO}\pi^+}$ into $\mathcal{L}_{\mathsf{HO}\pi}$

In this section we prove Thm. 7, in Page 22 that requires that encoding $\mathcal{L}_{\mathsf{HO}\pi^+}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated in Prop. 24.
- Operational Correspondence, stated in Prop. 25. Note that we prove a stronger operational correspondence condition, as in Def. 25, than the condition suggested in Def. 14(2).
- Full Abstraction, stated in Prop. 26.

Proposition 24 (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:

$$(\!|\Gamma|\!)^3;(\!|\Lambda_1|\!)^3;(\!|\Delta_1|\!)^3 \cdot u : (\!|S|\!)^3 \vdash [\![P']\!]^3 \triangleright \diamond \tag{25}$$

$$(\!|\Gamma|\!)^3 \cdot \underline{x} : (\!|L|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3 \vdash [\![Q]\!]^3 \triangleright \diamond \tag{26}$$

$$(\!|\Gamma|\!)^3 \cdot \underline{x} : (\!|L|\!)^3;\emptyset;\emptyset \vdash \underline{x} \triangleright (\!|L|\!)^3 \tag{27}$$

The corresponding typing in $\mathsf{HO}\pi$ is as follows:

$$\dfrac{\dfrac{\dfrac{\overline{(26)}}{(\!|\Gamma|\!)^3 \cdot x : (\!|L|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3 \cdot z : \mathsf{end} \vdash [\![Q]\!]^3 \triangleright \diamond} \quad \overline{(27)}}{(\!|\Gamma|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3 \cdot z : ?((\!|L|\!)^3);\mathsf{end} \vdash z?(\underline{x}).[\![Q]\!]^3 \triangleright \diamond}}{\dfrac{\overline{(25)} \quad \dfrac{(28) \quad \overline{(\!|\Gamma|\!)^3;\emptyset;z : ?((\!|L|\!)^3);\mathsf{end} \vdash z \triangleright ?((\!|L|\!)^3);\mathsf{end}}}{(\!|\Gamma|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3 \vdash \lambda z.z?(\underline{x}).[\![Q]\!]^3 \triangleright (?((\!|L|\!)^3);\mathsf{end}) \multimap \diamond}}{(\!|\Gamma|\!)^3;(\!|\Lambda_1|\!)^3 \cdot (\!|\Lambda_2|\!)^3;(\!|\Delta_1|\!)^3 \cdot (\!|\Delta_2|\!)^3 \cdot u : !\langle ?((\!|L|\!)^3);\mathsf{end} \multimap \diamond\rangle;(\!|S|\!)^3 \vdash u!\langle\lambda z.z?(\underline{x}).[\![Q]\!]^3\rangle.[\![P']\!]^3 \triangleright \diamond}} \tag{28}$$

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:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \cdot x : C \to \diamond;\Lambda;\Delta_1 \vdash P \triangleright \diamond}}{\Gamma;\Lambda;\Delta_1 \vdash \lambda x.\,P \triangleright (C \multimap \diamond) \multimap \diamond} \quad \dfrac{\dfrac{}{\Gamma;\emptyset;\Delta_2, y : C \vdash Q \triangleright \diamond}}{\Gamma;\emptyset;\Delta_2 \vdash \lambda y.\,Q \triangleright C \multimap \diamond}}{\Gamma;\Lambda;\Delta_1 \cdot \Delta_2 \vdash (\lambda x.\,P)(\lambda y.\,Q) \triangleright \diamond}$$

Thus, by IH we have:

$$(\!|\Gamma|\!)^3 \cdot x : (\!|C \to \diamond|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3 \vdash [\![P]\!]^3 \triangleright \diamond \tag{29}$$

$$(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_1|\!)^3, y : (\!|C|\!)^3 \vdash [\![Q]\!]^3 \triangleright \diamond \tag{30}$$

The corresponding typing in $\mathsf{HO}\pi$ is as follows — recall that $(\!|C \multimap \diamond|\!)^3 = (\!|C|\!)^3 \multimap \diamond$.

$$\dfrac{\dfrac{\dfrac{\overline{(29)}}{(\!|\Gamma|\!)^3 \cdot x : (\!|C \to \diamond|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3 \cdot s : \mathsf{end} \vdash [\![P]\!]^3 \triangleright \diamond}}{(\!|\Gamma|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3 \cdot s : ?((\!|C \multimap \diamond|\!)^3);\mathsf{end} \vdash s?(x).[\![P]\!]^3 \triangleright \diamond} \quad \dfrac{\dfrac{\dfrac{\dfrac{\overline{(30)}}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3 \cdot y : (\!|C|\!)^3 \vdash [\![Q]\!]^3 \triangleright \diamond}}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3 \vdash \lambda y.\,[\![Q]\!]^3 \triangleright (\!|C \multimap \diamond|\!)^3}}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3 \cdot \overline{s} : \mathsf{end} \vdash \lambda y.\,[\![Q]\!]^3 \triangleright (\!|C \multimap \diamond|\!)^3}}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3 \cdot \overline{s} : !\langle(\!|C \multimap \diamond|\!)^3\rangle;\mathsf{end} \vdash \overline{s}!\langle\lambda y.\,[\![Q]\!]^3\rangle.\mathbf{0} \triangleright \diamond}}{(\!|\Gamma|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3 \cdot (\!|\Delta_2|\!)^3 \cdot s : ?((\!|C \multimap \diamond|\!)^3);\mathsf{end} \cdot \overline{s} : !\langle(\!|C \multimap \diamond|\!)^3\rangle;\mathsf{end} \vdash s?(x).[\![P]\!]^3 \mid \overline{s}!\langle\lambda y.\,[\![Q]\!]^3\rangle.\mathbf{0} \triangleright \diamond}}{(\!|\Gamma|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3 \cdot (\!|\Delta_2|\!)^3 \vdash (\nu\, s)(s?(x).[\![P]\!]^3 \mid \overline{s}!\langle\lambda y.\,[\![Q]\!]^3\rangle.\mathbf{0}) \triangleright \diamond} \tag{31}$$

□

Before we prove operational correspondence we define mapping from $\{\!|\cdot|\!\}^3 : \mathcal{A} \to \mathcal{A}$ where $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$:

Definition 28 **($\{\!|\cdot|\!\}^3 : \mathcal{A} \to \mathcal{A}$).** Let $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$ then we define:

$$\{\!|(\nu\tilde{m})n!\langle\lambda x{:}L.P\rangle|\!\}^3 \stackrel{\texttt{def}}{=} (\nu\tilde{m})n!\langle\lambda z.z?(x).[\![P]\!]^3\rangle \qquad \{\!|n?\langle\lambda x{:}L.P\rangle|\!\}^3 \stackrel{\texttt{def}}{=} n?\langle\lambda z.z?(x).[\![P]\!]^3\rangle$$

and homomorphic for all other cases of $\ell \in \mathcal{A}$.

We now state and prove a detailed version of the operational corresponce in Def. 25.

Proposition 25 (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 $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\ell'}{\longmapsto} (\!|\Delta'|\!)^3 \vdash [\![P']\!]^3$ 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 $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\ell}{\longmapsto} (\!|\Delta'|\!)^3 \vdash [\![P']\!]^3$.*
   c) *If $\ell = \tau_\beta$ then $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\tau}{\longmapsto} \Delta'' \vdash R$ and $(\!|\Gamma|\!)^3;(\!|\Delta'|\!)^3 \vdash [\![P']\!]^3 \approx^{\mathrm{H}} \Delta'' \vdash R$, for some R.*
   d) *If $\ell = \tau$ and $\ell \neq \tau_\beta$ then $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^3 \vdash [\![P']\!]^3$.*
2. *Let $\Gamma;\emptyset;\Delta \vdash P$. $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\ell}{\longmapsto} (\!|\Delta''|\!)^3 \vdash Q$ implies*
   a) *If $\ell \in \{(\nu\tilde{m})n!\langle\lambda x.Q\rangle, n?\langle\lambda x.Q\rangle\}$ then $\Gamma;\Delta \vdash P \overset{\ell'}{\longmapsto} \Delta' \vdash P'$ 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 $\Gamma;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$ and $Q \equiv [\![P']\!]^3$.*
   c) *If $\ell = \tau$ then either $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash P'$ with $Q \equiv [\![P']\!]^3$ or $\Gamma;\Delta \vdash P \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^3;(\!|\Delta''|\!)^3 \vdash Q \overset{\tau_\beta}{\longmapsto} (\!|\Delta''|\!)^3 \vdash [\![P']\!]^3$.*

Proof. 1. The proof of Part 1 does a transition induction and considers the mapping as defined in Fig. 8. We give the most interesting cases.

- Case: $P = (\lambda x.Q_1)\,\lambda x.Q_2$.
  $\Gamma;\Delta \vdash (\lambda x.Q_1)\,\lambda x.Q_2 \overset{\tau_\beta}{\longmapsto} \Delta \vdash Q_1\{\lambda x.Q_2/x\}$ implies

  $$(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash (\nu s)(s?(x).[\![Q_1]\!]^3 \mid \overline{s}!\langle\lambda x.[\![Q_2]\!]^3\rangle.\mathbf{0}) \overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^3 \vdash [\![Q_1]\!]^3\{\lambda x.[\![Q_2]\!]^3/x\}$$

- Case: $P = n!\langle\lambda\underline{x}.Q\rangle.P$
  $\Gamma;\Delta \vdash n!\langle\lambda\underline{x}.Q\rangle.P \overset{n!\langle\lambda x.Q\rangle}{\longmapsto} \Delta \vdash P$ implies
  $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash n!\langle\lambda z.z?(x).[\![Q]\!]^3\rangle.[\![P]\!]^3 \overset{n!\langle\lambda z.z?(x).[\![Q]\!]^3\rangle}{\longmapsto} \Delta \vdash [\![P]\!]^3$
- Other cases are similar.

2. The proof of Part 2 also does a transition induction and considers the mapping as defined in Fig. 8. We give the most interesting cases.
   - Case: $P = (\lambda x.\, Q_1)\,\lambda x.\, Q_2$.

$$\begin{array}{rl} (\!|\Gamma|\!)^3;\emptyset;(\!|\Delta|\!)^3 & \stackrel{\tau_\beta}{\longmapsto} (\!|\Delta'|\!)^3 \vdash (\nu\, s)((\lambda z.\, z?(x).[\![Q]\!]^3)\, s \mid \overline{s}!\langle \lambda x.\, Q_2\rangle.\mathbf{0}) \\ & \stackrel{\tau_\beta}{\longmapsto} \qquad\qquad (\nu\, s)(s?(x).[\![Q]\!]^3 \mid \overline{s}!\langle \lambda x.\, Q_2\rangle.\mathbf{0}) \end{array}$$

     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

$$\begin{array}{rl} (\!|\Gamma|\!)^3;\emptyset;(\!|\Delta|\!)^3 & \stackrel{\tau_s}{\longmapsto} (\!|\Delta'|\!)^3 \vdash (\nu\, s)(s?(x).[\![Q]\!]^3 \mid \overline{s}!\langle \lambda x.\, Q_2\rangle.\mathbf{0}) \\ & \stackrel{\tau_s}{\longmapsto} \qquad\qquad [\![Q_1]\!]^3\{^{\lambda x.\, [\![Q_2]\!]^3}/_x\} \end{array}$$

   - Case: $P = n!\langle \lambda \underline{x}.\, Q\rangle.P$

     $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash n!\langle \lambda z.\, z?(x).[\![Q]\!]^3\rangle.[\![P]\!]^3 \stackrel{n!\langle \lambda z.\, z?(x).[\![Q]\!]^3\rangle}{\longmapsto} \Delta \vdash [\![P]\!]^3$ 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.

□

Proposition 26 (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 $\Gamma;\Delta_1 \vdash P \approx \Delta_2 \vdash Q$ if and only if $(\!|\Gamma|\!)^3;(\!|\Delta_1|\!)^3 \vdash [\![P]\!]^3 \approx (\!|\Delta_2|\!)^3 \vdash [\![Q]\!]^3$

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

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

It is straightforward to show that $\mathfrak{R}$ is a bisimulation if we follow Part 2 of Prop. 25 for subcases a and b. In subcase c we make use of Prop. 1.
Completeness Direction.
We create the closure

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

We show that $\mathfrak{R}$ is a bisimulation up to deterministic transitions by following Part 1 of Prop. 25. The proof is straightforward for subcases a), b) and d). In subcase c) we make use of Lem. 2. □

C.4 Properties for encoding $\mathcal{L}_{\mathsf{HO}\overline{\pi}}$ into $\mathcal{L}_{\mathsf{HO}\pi}$

In this section we prove Thm. 8, in Page 23 that requires that encoding $\mathcal{L}_{\mathsf{HO}\overline{\pi}}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated in Prop. 27.
- Operational Correspondence, stated in Prop. 28. Note that we prove a stronger operational correspondence condition, as in Def. 25, than the condition suggested in Def. 14(2).

– Full Abstraction, stated in Prop. 29.

Proposition 27 (Type Preservation. From $\mathsf{HO}\tilde{\pi}$ to $\mathsf{HO}\pi$). Let P be a $\mathsf{HO}\tilde{\pi}$ 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 Fig. 9 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:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_2|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot x_2 : (\!|C_2|\!)^4 \vdash [\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_2|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot x_2 : (\!|C_2|\!)^4 \cdot z : \mathsf{end} \vdash [\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_2|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot z : ?((\!|C_2|\!)^4); \mathsf{end} \vdash z?(x_2).[\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_2|\!)^4 \cdot z : ?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathsf{end} \vdash z?(x_1).z?(x_2).[\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_2|\!)^4 \vdash \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4 \triangleright ((\!|C_1|\!)^4, (\!|C_2|\!)^4) \multimap \diamond} \qquad (32)$$

$$\dfrac{\dfrac{}{(\!|\Gamma|\!)^{\mathsf{p}};\emptyset;(\!|\Delta_1|\!)^{\mathsf{p}} \cdot k : (\!|S|\!)^{\mathsf{p}} \vdash [\![P']\!]^{\mathsf{p}} \triangleright \diamond}\ (32)}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_1|\!)^4 \cdot (\!|\Delta_2|\!)^4 \cdot n : !\langle (\!|L|\!)^4\rangle; (\!|S|\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond}$$

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 Fig. 9 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:

$$\dfrac{\dfrac{\dfrac{}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_1|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot x_2 : (\!|C_2|\!)^4 \cdot n : (\!|S|\!)^4 \vdash [\![P']\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_1|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot n : ?((\!|C_2|\!)^4); (\!|S|\!)^4 \vdash n?(x_2).[\![P']\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_1|\!)^4 \cdot n : ?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); (\!|S|\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond}$$

□

Before we prove operational correspondence we define mapping from $\{\!|\cdot|\!\}^4 : \mathcal{A} \to \mathcal{A}$ where $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$:

Definition 29 **($\{\!|\cdot|\!\}^4 : \mathcal{A} \to \mathcal{A}$).** Let $\mathcal{A}$ is the set of labels of the relation $\overset{\ell}{\longmapsto}$ then we define:

$$\{\!|(\nu\tilde{m})n!\langle m_1, m_2\rangle|\!\}^4 \stackrel{\texttt{def}}{=} \ell_1, \ell_2 \text{ where } (m_i \in \tilde{m} \Leftrightarrow \ell_i = (\nu m_i)n!\langle m_i\rangle) \vee (m_i \notin \tilde{m} \Leftrightarrow \ell_i = n!\langle m_i\rangle)$$
$$\{\!|(\nu\tilde{m})n!\langle\lambda(x_1,x_2).P\rangle|\!\}^4 \stackrel{\texttt{def}}{=} (\nu\tilde{m})n!\langle\lambda z.z?(x_1).z?(x_2).[\![P]\!]^4\rangle$$

and homomorphic for all other cases of $\ell \in \mathcal{A}$.

Proposition 28 (Operational Correspondence. From HO$\tilde{\pi}$ to HOπ).

1. *Let $\Gamma;\emptyset;\Delta \vdash P$. Then $\Gamma;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$ implies*
   a) *If $\ell = (\nu\tilde{m}')n!\langle\tilde{m}\rangle$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell_1}{\longmapsto} \ldots \overset{\ell_n}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P]\!]^4$ with $\{\!|\ell|\!\}^4 = \ell_1 \ldots \ell_n$.*
   b) *If $\ell = n?\langle\tilde{m}\rangle$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell_1}{\longmapsto} \ldots \overset{\ell_n}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P]\!]^4$ with $\{\!|\ell|\!\}^4 = \ell_1 \ldots \ell_n$.*
   c) *If $\ell \in \{(\nu\tilde{m})n!\langle\lambda\tilde{x}.R\rangle, n?\langle\lambda\tilde{x}.R\rangle\}$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell'}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ with $\{\!|\ell|\!\}^4 = \ell'$.*
   d) *If $\ell \in \{n \oplus l, n\&l\}$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$.*
   e) *If $\ell = \tau_\beta$ then either $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto} \ldots \overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ with $\{\!|\ell|\!\} = \tau_\beta, \tau_s \ldots \tau_s$.*
   f) *If $\ell = \tau$ then $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\tau}{\longmapsto} \ldots \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ with $\{\!|\ell|\!\}^4 = \tau \ldots \tau$.*
2. *Let $\Gamma;\emptyset;\Delta \vdash P$. $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\longmapsto} (\!|\Delta_1|\!)^4 \vdash P_1$ implies*
   a) *If $\ell \in \{n?\langle m\rangle, n!\langle m\rangle, (\nu m)n!\langle m\rangle\}$ then $\Gamma;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4 \vdash P_1 \overset{\ell_2}{\longmapsto} \ldots \overset{\ell_n}{\longmapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$ with $\{\!|\ell|\!\}^4 = \ell_1 \ldots \ell_n$.*
   b) *If $\ell \in \{(\nu\tilde{m})n!\langle\lambda x.R\rangle, n?\langle\lambda x.R\rangle\}$ then $\Gamma;\Delta \vdash P \overset{\ell'}{\longmapsto} \Delta' \vdash P'$ with $\{\!|\ell'|\!\}^4 = \ell$ and $P_1 \equiv [\![P']\!]^4$.*
   c) *If $\ell \in \{n \oplus l, n\&l\}$ then $\Gamma;\Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$ and $P_1 \equiv [\![P']\!]^4$.*
   d) *If $\ell = \tau_\beta$ then $\Gamma;\Delta \vdash P \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau_s}{\longmapsto} \ldots \overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$ with $\{\!|\ell|\!\}^4 = \tau_\beta, \tau_s \ldots \tau_s$.*
   e) *If $\ell = \tau$ then $\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash P'$ and $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau}{\longmapsto} \ldots \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$ with $\{\!|\ell|\!\}^4 = \tau \ldots \tau$.*

Proof. The proof of both parts is by transition induction, following the mapping defined in Fig. 9 . 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

$$\Gamma; \Delta_0 \cdot m_1{:}S_1 \cdot m_2{:}S_2 \cdot n : !\langle S_1, S_2\rangle; S \vdash n!\langle m_1, m_2\rangle.P' \overset{\ell_1}{\longmapsto} \Delta_0 \cdot n{:}S \vdash P'$$

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 Fig. 9 , can be expressed as

$$(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_0|\!) \cdot m_1{:}(\!|S_1|\!)^4 \cdot m_2{:}(\!|S_2|\!)^4 \cdot n : !\langle (\!|S_1|\!)^4\rangle; !\langle (\!|S_2|\!)^4\rangle; (\!|S|\!)^4 \vdash n!\langle m_1\rangle.n!\langle m_2\rangle.[\![P']\!]^4 \triangleright \diamond$$

Now, $\{\!|\ell_1|\!\}^4 = n!\langle m_1\rangle, n!\langle m_2\rangle$. It is immediate to infer the following typed transitions for $[\![P]\!]^4 = n!\langle m_1\rangle.n!\langle m_2\rangle.[\![P']\!]^4$:

$$\begin{aligned}
& (\!|\Gamma|\!)^4; (\!|\Delta_0|\!) \cdot m_1{:}(\!|S_1|\!)^4 \cdot m_2{:}(\!|S_2|\!)^4 \cdot n : !\langle (\!|S_1|\!)^4\rangle; !\langle (\!|S_2|\!)^4\rangle; (\!|S|\!)^4 \vdash n!\langle m_1\rangle.n!\langle m_2\rangle.[\![P']\!]^4 \\
\overset{n!\langle m_1\rangle}{\longmapsto}\ & (\!|\Gamma|\!)^4; (\!|\Delta_0|\!) \cdot m_2{:}(\!|S_2|\!)^4 \cdot n : !\langle (\!|S_2|\!)^4\rangle; (\!|S|\!)^4 \vdash n!\langle m_2\rangle.[\![P']\!]^4 \\
\overset{n!\langle m_2\rangle}{\longmapsto}\ & (\!|\Gamma|\!)^4; (\!|\Delta_0|\!) \cdot n{:}(\!|S|\!)^4 \vdash [\![P']\!]^4 \\
=\ & (\!|\Gamma|\!)^4; (\!|\Delta_0 \cdot n : S|\!)^4 \vdash [\![P']\!]^4
\end{aligned}$$

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:

$$\frac{\Gamma; \emptyset; \Delta_0 \cdot n : S \vdash P' \triangleright \diamond \quad \Gamma; \emptyset; \Delta_1 \vdash \lambda(x_1, x_2).Q \triangleright (C_1, C_2) \multimap \diamond}{\Gamma; \emptyset; \Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) \multimap \diamond\rangle; S \vdash n!\langle \lambda(x_1, x_2).Q\rangle.P' \triangleright \diamond}$$

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:

$$\Gamma; \Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) \multimap \diamond\rangle; S \vdash n!\langle \lambda(x_1, x_2).Q\rangle.P' \overset{\ell_1}{\longmapsto} \Delta_0 \cdot n{:}S \vdash P'$$

The encoding of the source judgment is

$$(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_0 \cdot \Delta_1 \cdot n : !\langle (C_1, C_2) \multimap \diamond\rangle; S|\!)^4 \vdash [\![n!\langle \lambda(x_1, x_2).Q\rangle.P']\!]^4 \triangleright \diamond$$

which, using Fig. 9 , can be equivalently expressed as

$$(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_0 \cdot \Delta_1|\!) \cdot n : !\langle (?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathsf{end}) \multimap \diamond\rangle; (\!|S|\!)^4 \vdash n!\langle \lambda z.z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4 \triangleright \diamond$$

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 $[\![P]\!]^4 = n!\langle \lambda z.z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4$:

$$\begin{aligned}
& (\!|\Gamma|\!)^4; (\!|\Delta_0 \cdot \Delta_1|\!) \cdot n : !\langle (?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathsf{end}) \multimap \diamond\rangle; (\!|S|\!)^4 \vdash n!\langle \lambda z.z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4 \\
\overset{\{\!|\ell_1|\!\}^4}{\longmapsto}\ & (\!|\Gamma|\!)^4; (\!|\Delta_0|\!) \cdot n : (\!|S|\!)^4, \vdash [\![P']\!]^4 \\
=\ & (\!|\Gamma|\!)^4; (\!|\Delta_0 \cdot n : S|\!)^4 \vdash [\![P']\!]^4
\end{aligned}$$

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:

$$\begin{aligned}
&(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\cdot n:?((\!|S_1|\!)^4);?((\!|S_2|\!)^4);(\!|S|\!)^4\cdot \vdash n?(x_1).n?(x_2).[\![P']\!]^4\\
\xrightarrow{n?\langle m_1\rangle}\ &(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\cdot n:?((\!|S_2|\!)^4);(\!|S|\!)^4\cdot m_1:(\!|S_1|\!)^4\vdash n?(x_2).[\![P']\!]^4\{m_1/x_1\}\\
\xrightarrow{n?\langle m_2\rangle}\ &(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\cdot n:(\!|S|\!)^4\cdot m_1:(\!|S_1|\!)^4\cdot m_2:(\!|S_2|\!)^4\vdash [\![P']\!]^4\{m_1/x_1\}\{m_2/x_2\}=Q
\end{aligned}$$

  Observe that the we use substitution 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:

$$\Gamma;\Delta\cdot n:?(S_1,S_2);S\vdash n?(x_1,x_2).P'\xrightarrow{\ell_1}\Delta\cdot n{:}S\cdot m_1:S_1\cdot m_2:S_2\vdash P'\{m_1,m_2/x_1,x_2\}$$

  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:

$$\begin{aligned}
&(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\cdot n:(\!|!\langle(C_1,C_2)\multimap\diamond\rangle;S|\!)^4\vdash n!\langle\lambda z.\,z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4\\
\xrightarrow{\ell'_1}\ &(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\cdot n:(\!|S|\!)^4\vdash [\![P']\!]^4=Q
\end{aligned}$$

  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:

$$\Gamma;\Delta\cdot n:!\langle(C_1,C_2)\multimap\diamond\rangle;S\vdash n!\langle\lambda x_1,x_2.\,Q\rangle.P'\xrightarrow{\ell_1}\Delta\cdot n{:}S\vdash P'$$

  which concludes the proof for this case.

□

Proposition 29 (Full Abstraction. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). *Let P,Q be $\mathsf{HO}\pi^+$ process with $\Gamma;\emptyset;\Delta_1\vdash P\triangleright\diamond$ and $\Gamma;\emptyset;\Delta_2\vdash Q\triangleright\diamond$. $\Gamma;\Delta_1\vdash P\approx^{\mathrm{H}}\Delta_2\vdash Q$ if and only if $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4\vdash [\![P]\!]^4\approx^{\mathrm{H}}(\!|\Delta_2|\!)^4\vdash[\![Q]\!]^4$.*

Proof. The proof for both direction is a consequence of Operational Correspondence, Prop. 28.

Soundness Direction.

We create the closure

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

It is straightforward to show that $\mathfrak{R}$ is a bisimulation if we follow Part 2 of Prop. 28.

Completeness Direction.

We create the closure

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

We show that $\mathfrak{R}$ is a bisimulation up to deterministic transitions by following Part 1 of Prop. 28. □

D Negative Result

Theorem 9. *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:

$$\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash n \triangleleft l_1.\mathbf{0} \mid a(x).m \triangleleft l_2.\mathbf{0} = P_1 \tag{33}$$

$$\Gamma;\Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash m \triangleleft l_2.\mathbf{0} \mid a(x).n \triangleleft l_1.\mathbf{0} = P_2 \tag{34}$$

Thus, by definition of typed barb we have:

$$\Gamma;\Delta' \vdash P_1 \downarrow_n \wedge \Gamma;\Delta' \vdash P_1 \not\downarrow_m \tag{35}$$

$$\Gamma;\Delta' \vdash P_2 \downarrow_m \wedge \Gamma;\Delta' \vdash P_2 \not\downarrow_n \tag{36}$$

Consider now the $\mathsf{HO}\pi^{-\mathsf{sh}}$ process $[\![P]\!]$. By our assumption of operational completeness (Def. 14-2(a)), from (33) with (34) we infer that there exist $\mathsf{HO}\pi^{-\mathsf{sh}}$ processes S_1 and S_2 such that:

$$(\!|\Gamma|\!);(\!|\Delta|\!) \vdash [\![P]\!] \overset{\tau_s}{\Longmapsto} (\!|\Delta'|\!) \vdash S_1 \approx [\![P_1]\!] \tag{37}$$

$$(\!|\Gamma|\!);(\!|\Delta|\!) \vdash [\![P]\!] \overset{\tau_s}{\Longmapsto} (\!|\Delta'|\!) \vdash S_2 \approx [\![P_2]\!] \tag{38}$$

By our assumption of barb preservation, from (35) with (36) we infer:

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_1]\!] \Downarrow_n \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_1]\!] \not\Downarrow_m \tag{39}$$

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_2]\!] \Downarrow_m \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_2]\!] \not\Downarrow_n \tag{40}$$

By definition of $\approx$, by combining (37) with (39) and (38) with (40), we infer barbs for S_1 and S_2:

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_1 \Downarrow_n \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_1 \not\Downarrow_m \tag{41}$$

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_2 \Downarrow_m \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_2 \not\Downarrow_n \tag{42}$$

That is, S_1 and $[\![P_1]\!]$ (resp. S_2 and $[\![P_2]\!]$) have the same barbs. Now, by τ-inertness (Prop. 1), we have both

$$(\!|\Gamma|\!);(\!|\Delta|\!) \vdash S_1 \approx (\!|\Delta'|\!) \vdash [\![P]\!] \tag{43}$$

$$(\!|\Gamma|\!);(\!|\Delta|\!) \vdash S_2 \approx (\!|\Delta'|\!) \vdash [\![P]\!] \tag{44}$$

Combining (43) with (44), by transitivity of $\approx$, we have

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_1 \approx (\!|\Delta'|\!) \vdash S_2 \tag{45}$$

In turn, from (45) 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 (39) and (40) above. □