Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

On the relative expressiveness of higher-order session processes

Dimitrios Kouzapas [a,*], Jorge A. Pérez [b,*], Nobuko Yoshida [c,*]

[a] University of Cyprus, Nicosia, Cyprus
[b] University of Groningen & CWI, Amsterdam, the Netherlands
[c] Imperial College London, UK

ARTICLE INFO

Article history:
Received 13 October 2017
Received in revised form 23 December 2018
Accepted 7 June 2019
Available online xxxx

Keywords:
Concurrency
Process calculi
Behavioural types
Session types
Expressiveness

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 HOπ, the higher-order π-calculus in which communications are governed by *session types*. Our main discovery is that HO, a subcalculus of HOπ which lacks name-passing and recursion, can serve as a new core calculus for session-typed higher-order concurrency. We show that HO can encode HOπ fully abstractly (up to typed contextual equivalence) more precisely and efficiently than the first-order session π-calculus (π). Overall, under the discipline of session types, HOπ, HO, and π are equally expressive; however, we show that HOπ is more tightly related to HO than to π.

© 2019 Published by Elsevier Inc.

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 [26,37,22]. In concurrency theory, a vast literature on *expressiveness* also studies compilations (or *encodings*) [31,10,7,20,36]: they are used to transfer reasoning techniques across calculi, and to implement complex programming abstractions using simpler process constructs.

In this work, we study the *relative expressiveness* of HOπ, a *higher-order* process language that integrates message-passing concurrency (including recursion) with functional features. We consider *type-preserving encodings* between source and target calculi coupled with *session types* [12] denoting interaction protocols. Building on untyped frameworks for relative expressiveness [10], we propose type preservation as a new criterion for *precise encodings*. We identify HO, a new core calculus for higher-order session concurrency which lacks name passing and recursion. We show that HO can encode HOπ precisely and efficiently. Requiring type preservation makes this encoding far from trivial: we crucially exploit advances on session type duality [1,4] and recent characterisations of typed contextual equivalence [15,17]. We develop a full hierarchy of variants of HOπ based on precise encodings: our encodings are type-preserving and fully abstract up to typed behavioural equivalences. Fig. 1 illustrates this hierarchy; the variants of HOπ are explained next.

Context In *session-based concurrency*, interactions are organised into *sessions*, basic communication units. Interaction patterns can then be abstracted as *session types* [12], against which specifications may be checked. The 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,

* Corresponding authors.
E-mail addresses: dimitrios.kouzapas@cs.ucy.ac.cy (D. Kouzapas), j.a.perez@rug.nl (J.A. Pérez), n.yoshida@imperial.ac.uk (N. Yoshida).

https://doi.org/10.1016/j.ic.2019.06.002
0890-5401/© 2019 Published by Elsevier Inc.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

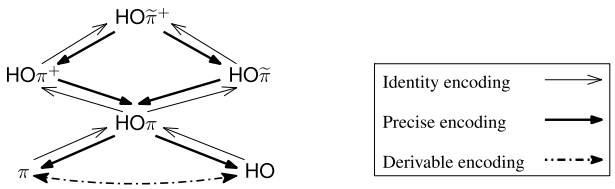

Fig. 1. Encodability in Higher-Order Sessions. Precise encodings are defined in Definition 4.6.

given an index set I, types $\&\{l_i : S_i\}_{i\in I}$ and $\oplus\{l_i : S_i\}_{i\in I}$ define, respectively, external and internal choice constructs for a labelled choice mechanism; types $\mu \mathrm{t}.S$ and $\mathtt{end}$ specify recursive and completed protocols, respectively. By distinguishing between *linear* and *shared names*, session types for the π-calculus describe the intended interactive behaviour of the names in a process [12].

Session-based concurrency has also been cast 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 [27,11]. The higher-order calculus with sessions studied here, called HOπ, can specify protocols involving *code mobility*: it includes constructs for synchronisation along shared names, session communication (value passing, labelled choice) along linear names, recursion and applications. Values in communications can be names but also *first-order* abstractions—functions from name identifiers to processes. (In contrast, HOπ lacks *higher-order* abstractions—functions from processes to processes—but these can be encoded, see below.) Abstractions can be linear or shared, depending on whether they contain linear names or not; their types are denoted $C \multimap \diamond$ and $C \to \diamond$, respectively (C denotes a name).

Expressiveness of HOπ We study the type-preserving, relative expressivity of HOπ. As expected from known literature in the untyped setting [39], the first-order session π-calculus [12] (here denoted π) can encode the higher-order calculus HOπ preserving session types. In this paper, our main discovery concerns the opposite direction: we show that HOπ without name-passing and recursion can serve as a core calculus for higher-order session concurrency. We call this core calculus HO. We show that HO can encode HOπ more efficiently than π. In addition, in the higher-order session typed setting, HO offers more tractable bisimulation techniques than π (cf. §3.3.2).

Challenges and contributions We assess the relative expressiveness of HOπ, HO, and π as delineated by session types. We introduce the notion of *type-preserving encodings*: type information is used 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 also 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* (Definition 4.6), which extends encodability criteria for untyped processes with *full abstraction*. Full abstraction results are stated up to two behavioural equivalences that characterise barbed congruence: *characteristic bisimilarity* ($\approx^{\mathrm{C}}$, introduced in [15]) and *higher-order bisimilarity* ($\approx^{\mathrm{H}}$, introduced in [16] and developed in [17]). Using precise encodings we establish strong correspondences between HOπ and its variants—see below.

Our contributions can be divided in two parts. First, we develop a precise encoding of HOπ into HO (§5.1). 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 [42] only the output capability of names can be sent—a received name cannot be used in later inputs. This is far too limiting in HOπ, 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 HOπ, because linear abstractions cannot be copied/duplicated. Hence, the discipline of session types limits the possibilities for representing infinite behaviours—this holds for even simple forms, such as input-guarded replication.

Our encoding overcomes these two obstacles, as we discuss in §2.

In the second part, we offer additional technical contributions, which include:

(i) the encodability of HO into π (§5.2);
(ii) a non encodability result showing that shared names strictly add expressive power to session calculi (§5.4).
(iii) extensions of our encodability results to richer settings (§6);

In essence, (i) extends known results for untyped processes [39] to the session typed setting. Concerning (iii), we develop extensions of our encodings to

- The extension of HOπ with *higher-order* abstractions (HOπ^+);
- The extension of HOπ with polyadic name passing and abstraction (HO$\tilde{\pi}$);
- The super-calculus of HOπ^+ and HO$\tilde{\pi}$ (denoted HO$\tilde{\pi}^+$), equivalent to the calculus in [27].

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Fig. 1 summarises our encodability results. They connect HOπ with existing higher-order process calculi [27], and highlight the status of HO as the core calculus for session concurrency. Finally, to our knowledge we are the first to prove the non encodability result (ii), exploiting session determinacy and typed equivalences.

Outline §2 overviews key ideas of the precise encoding of HOπ into π. §3 collects background material: §3.1 presents HOπ and its subcalculi (HO and π); §3.2 summarises their session type system; §3.3 presents behavioural equalities for HOπ from [15,17]: barbed congruence, characteristic bisimilarity, and higher-order bisimilarity. §4 defines *precise encodings* by extending encodability criteria for untyped processes. §5 gives precise encodings of HOπ into HO and into π (Theorems 5.1 and 5.2). Mutual encodings between π and HO are derivable; all these calculi are thus equally expressive. Via empirical and formal comparisons between these two precise encodings, in §5.3 we establish that HOπ and HO are more tightly related than HOπ and π (Theorem 5.3). Moreover, we prove the impossibility of encoding communication along shared names using linear names (Theorem 5.4). In §6 we show encodings of HOπ^+ and HO$\widetilde{\pi}$ into HOπ (Theorems 6.1 and 6.2). §7 reviews related works and §8 concludes. Omitted definitions and proofs are in the Appendices (Appendix A and Appendix B).

This paper is an extended and revised version of the homonymous conference paper that appeared in the Proceedings of ESOP'16 [16]. With respect to [16], the current paper provides extended discussions, additional examples, and full technical details. Moreover, it offers a sharper focus on relative expressiveness: a detailed treatment of higher-order bisimilarity (first introduced in [16]) can now be found in our paper [17] (which corresponds to the journal version of [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 (§5.1) should (a) enable the communication of arbitrary names, as required to represent delegation, and (b) address the fact that the linear communication discipline, enforced by session types, limits the possibilities for representing infinite behaviour.

To illustrate our encoding of name passing into HO, we informally introduce some process syntax; formal definitions are given in §3.1. Below, a, b are names and s is a linear session name; name $\overline{s}$ is the dual of s—they are *endpoints* of the same session. Processes $a!\langle V\rangle.P$ and $a?(x).P$ denote output and input at a, respectively; abstractions and applications are denoted $\lambda x.P$ and $(\lambda x.P)\,a$, respectively. Processes $(\nu\, s)(P)$, $P \mid Q$, and $\mathbf{0}$ represent usual forms of name restriction/hiding, parallel composition, and inaction.

In our encoding, we "pack" the name to be sent (denoted b) into an abstraction; upon reception, the receiver "unpacks" this object following a precise protocol on a fresh session (denoted s):

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

Thus, an abstraction containing the name b is first passed around along a. Following this communication, a sequence of (deterministic) reductions between s and $\overline{s}$ guarantees that b is properly unpacked by means of abstraction passing and appropriate applications. Indeed, the above encoding requires three extra reduction steps to mimic a single name communication step in HOπ. Also, notice that 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. Definition 4.4).

As hinted at above, a challenge in encoding recursion is preserving linearity of session names. Roughly speaking, given $\mu X.P$, we encode its recursion body P as an abstraction $\lambda\tilde{x}.\,[\![P]\!]_\sigma$ in which each session name of P (included in set σ) is converted into a name variable in $\tilde{x}$. Since $\lambda\tilde{x}.\,[\![P]\!]_\sigma$ does not mention (linear) session names, we may embed it into a "duplicator" process which implements recursion using higher-order communication [45]. The encoding of the recursion variable X invokes this duplicator in a by-need fashion: it receives $\lambda\tilde{x}.\,[\![P]\!]_\sigma$ and uses two copies of it: one copy allows us to obtain P through the application of the session names in σ; the other allows us to invoke the duplicator when needed. Interestingly, for this encoding to work we require non-tail recursive session types; this exploits recent advances on the theory of duality for session types [1,4].

A plausible encoding that is not precise Our notion of *precise encoding* (Definition 4.6) requires the translation of both process and types; it 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 consistent communication structures described by session types. These requirements are demanding and make our developments far from trivial. In particular, requiring type preservation may rule out other plausible encoding strategies. To illustrate this point, consider the following alternative encoding of name-passing into HO[1]:

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

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

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

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

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

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 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* (cf. Ex. 4.1). 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 $[\![\cdot]\!]^u$ would simply alter the meaning of the session protocol in the source language.

3. Preliminaries

We introduce the *higher-order session π-calculus* (HOπ). We first define syntax, operational semantics, and its sub-calculi (denoted π and HO). Then, a type system and behavioural equivalences for HOπ are recalled in §3.2 and §3.3. HOπ features first-order abstractions and monadic communication; extensions with higher-order abstractions and polyadicity (denoted HOπ^+ and HO$\widetilde{\pi}$, respectively) are discussed in §6. In §3.4 we recall the *Hotel Booking scenario*, a case study for HOπ that we developed in [15,17].

3.1. HOπ: syntax, operational semantics, and subcalculi

Syntax The syntax of HOπ is defined in Fig. 2. HOπ is a subcalculus of the language studied in [27]. It is also a variant of the language that we investigated in [15], which includes higher-order value applications.

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* $V, W, \ldots$ include identifiers $u, v, \ldots$ and abstractions $\lambda x.\, P$ (first- and higher-order values, resp.).

Process terms $P, Q, \ldots$ include usual prefixes for sending and receiving values V. Processes $u \triangleleft l.P$ and $u \triangleright \{l_i : P_i\}_{i\in I}$ are the usual constructs for selection and branching, used to specify labelled deterministic choices within sessions [12]. Process $V\, u$ denotes application; it substitutes name u on the abstraction V. Typing ensures that V is not a name. Recursion $\mu X.P$ binds the recursive variable X in P. Constructs for inaction $\mathbf{0}$, parallel composition $P_1 \mid P_2$, and name restriction $(\nu\, n)P$ are standard.

Notation 1. *We shall write $*P$ to denote a replicated process P, representable as $\mu X.(P \mid X)$.*

Session name restriction $(\nu\, s)P$ simultaneously binds endpoints s and $\overline{s}$ in P. Functions $\mathtt{fv}(P)$, $\mathtt{fn}(P)$, and $\mathtt{fs}(P)$ denote, respectively, the sets of free variables, names, and session names in P, and are defined as expected. 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*.

In a statement, a name (resp. variable) is *fresh* if it is not among the names (resp. variables) of the objects (processes, actions, etc.) of the statement. We shall follow Barendregt's convention: all (session) names and variables in binding occurrences, in any mathematical context, are pairwise distinct but also distinct from free (session) names and variables.

Operational semantics The operational semantics of HOπ is defined in terms of a *reduction relation*, denoted $\longrightarrow$, whose rules are given in Fig. 3 (top). We briefly describe 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 congruence*, noted $\equiv$ and given in Fig. 3 (bottom). We write $\equiv_\alpha$ to denote α-conversion and 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.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

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

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

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

$$P \longrightarrow P' \Rightarrow (\nu\, n)P \longrightarrow (\nu\, n)P' \qquad \text{[Res]}$$

$$P \longrightarrow P' \Rightarrow P \mid Q \longrightarrow P' \mid Q \qquad \text{[Par]}$$

$$P \equiv Q \longrightarrow Q' \equiv P' \Rightarrow P \longrightarrow P' \qquad \text{[Cong]}$$

$$P \mid \mathbf{0} \equiv P \quad P_1 \mid P_2 \equiv P_2 \mid P_1 \quad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3 \quad (\nu\, n)\mathbf{0} \equiv \mathbf{0}$$

$$P \mid (\nu\, n)Q \equiv (\nu\, n)(P \mid Q) \ (n \notin \mathtt{fn}(P)) \quad \mu X.P \equiv P\{^{\mu X.P}/_{X}\} \quad P \equiv Q \text{ if } P \equiv_{\alpha} Q$$

Fig. 3. Operational semantics of HOπ.

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

Fig. 4. Syntax of session types for HOπ.

- The *session π-calculus*, denoted π, lacks higher-order communication but includes recursion; its formal syntax is obtained from Fig. 2 by excluding constructs in boxes.

Let $\mathsf{C} \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$. We write $\mathsf{C}^{-\mathsf{sh}}$ to denote the calculus C without shared names: we delete a, b from n. Thus, languages in $\mathsf{C}^{-\mathsf{sh}}$ feature linear, deterministic behaviour only. In §5 we shall demonstrate that HOπ, HO, and π have the same expressivity, and that C is strictly more expressive than $\mathsf{C}^{-\mathsf{sh}}$.

3.2. Session types for HOπ

We state key definitions and properties for the session type system for HOπ. The considered type system, introduced in [17], distills the key features of [27,28] and so it is simpler.

The syntax of types for HOπ is given in Fig. 4. We write $\diamond$ to denote the process type. 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 [12], with the extension that carried types U may be higher-order. Shared channel types are denoted $\langle S\rangle$ and $\langle L\rangle$.

The type syntax of HO exclude $\boxed{C}$ from value types U; the types of π excludes $\boxed{L}$ and $\boxed{\langle L\rangle}$. Given $\mathsf{C} \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$, the sub-calculus $\mathsf{C}^{-\mathsf{sh}}$ is obtained by excluding shared name types ($\langle S\rangle$ and $\langle L\rangle$), from name type C.

We now define session type duality [1], which builds upon *type equivalence*.

Definition 3.1 *(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:

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

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

Intuitively, duality is obtained by swapping ! by ?, ? by !, $\oplus$ by &, and & by $\oplus$, including the fixed point construction. More formally, we have:

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Definition 3.2 *(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:

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

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

We consider shared, linear, and session *environments*, denoted Γ, Λ, and Δ, resp.:

$$\Gamma ::= \emptyset \mid \Gamma \cdot x : C \to \diamond \mid \Gamma \cdot u : \langle S\rangle \mid \Gamma \cdot u : \langle L\rangle \mid \Gamma \cdot X : \Delta$$

$$\Lambda ::= \emptyset \mid \Lambda \cdot x{:}C \multimap \diamond$$

$$\Delta ::= \emptyset \mid \Delta \cdot u{:}S$$

Γ maps variables and shared names to value types, and recursive variables to session environments; it admits weakening, contraction, and exchange principles. Λ maps variables to linear higher-order types; Δ maps session names to session types. Both Λ and Δ are only subject to exchange. The domains of Γ, Λ, and Δ are assumed pairwise distinct. We write $\Delta_1 \cdot \Delta_2$ for the disjoint union of Δ_1 and Δ_2. We write $\Gamma \backslash x$ to denote the environment obtained from Γ by removing the assignment $x : U \to \diamond$, for some U. Similarly, we write $\Delta_1 \backslash \Delta_2$ and $\Lambda_1 \backslash \Lambda_2$ with the expected reading.

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

Fig. 5 gives the typing rules. We now describe some of them; see [17] 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 shared type variable as linear. 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 of the application variable x (i.e., $C \multimap \diamond$ or $C \to \diamond$). In Rule (Send), the type U of value V should appear as a prefix in the session type $!\langle U\rangle; S$ of u. Rule (Rcv) is its dual. Rules (Req) and (Acc) type interaction along shared names; 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.).

We close this section by stating *type soundness* for HOπ, as established in [17]; it implies type soundness for HO, π, and $\mathsf{C}^{-\mathsf{sh}}$. We require two auxiliary definitions. First, we focus on *balanced* session environments:

Definition 3.3 *(Balanced environments).* 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 (cf. Definition 3.2).

Second, we define a notion of reduction for session environments:

Definition 3.4. We define the relation $\longrightarrow$ on session environments Δ 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 write $\longrightarrow^*$ to denote multi-step reduction.

We then have:

Theorem 3.1 *(Type soundness [17]). 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.*

3.3. Behavioural theory for HOπ

We first define reduction-closed, barbed congruence ($\cong$, Definition 3.9) as the reference equivalence relation for HOπ processes. We then recall two characterisations of $\cong$: *characteristic* and *higher-order bisimilarities* (denoted $\approx^{\mathrm{C}}$ and $\approx^{\mathrm{H}}$, cf. Definitions 3.12 and 3.11). We refer to Appendix A for omitted definitions, and to our previous paper [17] for a detailed treatment of these behavioural equivalences.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

$$\text{(Sess)}\ \Gamma;\emptyset;\{u:S\}\vdash u\triangleright S \qquad \text{(Sh)}\ \Gamma\cdot u:U;\emptyset;\emptyset\vdash u\triangleright U$$

$$\text{(LVar)}\ \Gamma;\{x:C\multimap\diamond\};\emptyset\vdash x\triangleright C\multimap\diamond \qquad \text{(RVar)}\ \Gamma\cdot X:\Delta;\emptyset;\Delta\vdash X\triangleright\diamond$$

$$\text{(Abs)}\ \frac{\Gamma;\Lambda;\Delta_1\vdash P\triangleright\diamond\quad\Gamma;\emptyset;\Delta_2\vdash x\triangleright C}{\Gamma\backslash x;\Lambda;\Delta_1\backslash\Delta_2\vdash\lambda x.P\triangleright C\multimap\diamond} \qquad \text{(App)}\ \frac{\Gamma;\Lambda;\Delta_1\vdash V\triangleright C\rightsquigarrow\diamond\quad\rightsquigarrow\in\{\multimap,\to\}\quad\Gamma;\emptyset;\Delta_2\vdash u\triangleright C}{\Gamma;\Lambda;\Delta_1\cdot\Delta_2\vdash V\,u\triangleright\diamond}$$

$$\text{(Prom)}\ \frac{\Gamma;\emptyset;\emptyset\vdash V\triangleright C\multimap\diamond}{\Gamma;\emptyset;\emptyset\vdash V\triangleright C\to\diamond} \qquad \text{(EProm)}\ \frac{\Gamma;\Lambda\cdot x:C\multimap\diamond;\Delta\vdash P\triangleright\diamond}{\Gamma\cdot x:C\to\diamond;\Lambda;\Delta\vdash P\triangleright\diamond} \qquad \text{(End)}\ \frac{\Gamma;\Lambda;\Delta\vdash P\triangleright T\quad u\notin\mathrm{dom}(\Gamma,\Lambda,\Delta)}{\Gamma;\Lambda;\Delta\cdot u:\mathtt{end}\vdash P\triangleright\diamond}$$

$$\text{(Rec)}\ \frac{\Gamma\cdot X:\Delta;\emptyset;\Delta\vdash P\triangleright\diamond}{\Gamma;\emptyset;\Delta\vdash\mu X.P\triangleright\diamond} \qquad \text{(Par)}\ \frac{\Gamma;\Lambda_i;\Delta_i\vdash P_i\triangleright\diamond\quad i=1,2}{\Gamma;\Lambda_1\cdot\Lambda_2;\Delta_1\cdot\Delta_2\vdash P_1\mid P_2\triangleright\diamond} \qquad \text{(Nil)}\ \frac{}{\Gamma;\emptyset;\emptyset\vdash\mathbf{0}\triangleright\diamond}$$

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

$$\text{(Req)}\ \frac{\Gamma;\Lambda;\Delta_1\vdash P\triangleright\diamond\quad\Gamma;\emptyset;\emptyset\vdash u\triangleright\langle\mathcal{U}\rangle\quad\Gamma;\emptyset;\Delta_2\vdash V\triangleright\mathcal{U}\quad\mathcal{U}\in\{S,L\}}{\Gamma;\Lambda;\Delta_1\cdot\Delta_2\vdash u!\langle V\rangle.P\triangleright\diamond}$$

$$\text{(Rcv)}\ \frac{\Gamma;\Lambda_1;\Delta_1\cdot u:S\vdash P\triangleright\diamond\quad\Gamma;\Lambda_2;\Delta_2\vdash x\triangleright U}{\Gamma\backslash x;\Lambda_1\backslash\Lambda_2;\Delta_1\backslash\Delta_2\cdot u:?(U);S\vdash u?(x).P\triangleright\diamond} \qquad \text{(Acc)}\ \frac{\Gamma;\Lambda_1;\Delta_1\vdash P\triangleright\diamond\quad\Gamma;\emptyset;\emptyset\vdash u\triangleright\langle\mathcal{U}\rangle\quad\Gamma;\Lambda_2;\Delta_2\vdash x\triangleright\mathcal{U}\quad\mathcal{U}\in\{S,L\}}{\Gamma\backslash x;\Lambda_1\backslash\Lambda_2;\Delta_1\backslash\Delta_2\vdash u?(x).P\triangleright\diamond}$$

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

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

Fig. 5. Typing rules for HOπ.

3.3.1. Reduction-closed, barbed congruence ($\cong$)

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

Definition 3.5 *(Session environment confluence).* We denote $\Delta_1 \rightleftharpoons \Delta_2$ if there exists Δ such that $\Delta_1 \longrightarrow^* \Delta$ and $\Delta_2 \longrightarrow^* \Delta$.

Definition 3.6 *(Typed relation).* We say that $\Gamma;\emptyset;\Delta_1\vdash P\triangleright\diamond\ \Re\ \Gamma;\emptyset;\Delta_2\vdash Q\triangleright\diamond$ is a *typed relation* whenever P and Q are closed; Δ_1 and Δ_2 are balanced; and $\Delta_1 \rightleftharpoons \Delta_2$.

We write $\Gamma;\Delta_1\vdash P\ \Re\ \Delta_2\vdash Q$ for the typed relation $\Gamma;\emptyset;\Delta_1\vdash P\triangleright\diamond\ \Re\ \Gamma;\emptyset;\Delta_2\vdash Q\triangleright\diamond$.

A *barb* $\downarrow_n$ is an observable on an output or selection prefix with subject n [25]. Notice that observing output barbs is enough to (indirectly) observe input actions. 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 3.7 *(Untyped and typed barbs).* Let P be a closed process. We define:

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

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

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Definition 3.8 *(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 [13].

Definition 3.9 *(Barbed congruence).* Typed relation $\Gamma;\Delta_1 \vdash P \;\Re\; \Delta_2 \vdash Q$ 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 $\Gamma;\Delta_1' \vdash P' \;\Re\; \Delta_2' \vdash Q'$;
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 $\Gamma;\Delta_1'' \vdash \mathbb{C}[P] \;\Re\; \Delta_2'' \vdash \mathbb{C}[Q]$;
4. The symmetric cases of 1 and 2.

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

3.3.2. Two equivalence relations: $\approx^{\mathrm{H}}$ and $\approx^{\mathrm{C}}$

In [15,17] we have characterised reduction-closed, barbed congruence for HOπ via two typed relations, called *characteristic bisimilarity* and *higher-order bisimilarity*. Their definition uses a *typed* labelled transition system (LTS) on processes, informed by session types [18], whose key notions are summarized next. We will be working with closed process terms, i.e., processes without free variables.

A typed labelled transition system The typed LTS describes the interaction of well-typed processes with their environment. We shall focus on well-typed processes whose type judgements have an empty Λ, i.e., an empty environment for linear higher-order types. Given this, we write

$$\Gamma;\Delta \vdash P \xrightarrow{\ell} \Delta' \vdash P'$$

to denote a (strong) transition with action label ℓ (cf. Definition 3.10 below).

Formally, the typed LTS is obtained by coupling an untyped LTS on processes, whose transitions are denoted $P \xrightarrow{\ell} P'$ with a labelled transition relation on typing environments, whose transitions are denoted $(\Gamma,\Delta) \xrightarrow{\ell} (\Gamma,\Delta')$ (see Definition Appendix A.2). These auxiliary LTSs are given in Fig. A.14 and Fig. A.15, respectively. The key idea is that the transitions of a typed process should be enabled by its associated typing:

$$\text{if } P \xrightarrow{\ell} P' \text{ and } (\Gamma,\Delta) \xrightarrow{\ell} (\Gamma,\Delta') \text{ then } \Gamma;\Delta \vdash P \xrightarrow{\ell} \Delta' \vdash P'.$$

The LTS on untyped processes, the LTS on typing environments, and the typed LTS share the same set of action labels:

Definition 3.10 *(Action labels).* The set of action labels for HOπ, ranged over by $\ell, \ell', \ldots$, is defined as follows:

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

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). 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 $n\&l$, respectively. We write $\mathtt{fn}(\ell)$ and $\mathtt{bn}(\ell)$ to denote the sets of free/bound names in ℓ, respectively.

Remark 3.1 *(Type Annotations (1)).* We sometimes annotate process actions with their type. In particular, given a value V of type U, we may write label $(\nu\,\widetilde{m})n!\langle V\rangle$ as $(\nu\,\widetilde{m})n!\langle V : U\rangle$.

The sets of actions for HO and π is derived from the above syntax, in line with the syntax of values V in Fig. 2. This way, e.g., $(\nu\,\widetilde{m})n!\langle\lambda x.\, P\rangle$ is an action label for HO but not for π; similarly, $s?\langle n\rangle$ is an action label for π but not for HO.

A refined typed LTS The characterisation of barbed congruence relies on a *refined* typed LTS on typing environments. Intuitively, the objective is to have a more stringent rule for input transitions, given as follows:

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

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

$$
\begin{array}{rcl}
(\!|?(U);S|\!)^u & \stackrel{\text{def}}{=} & u?(x).(t!\langle u\rangle.\mathbf{0} \mid (\!|U|\!)^x) \\
(\!|!\langle U\rangle;S|\!)^u & \stackrel{\text{def}}{=} & u!\langle (\!|U|\!)_{\mathrm{c}}\rangle.t!\langle u\rangle.\mathbf{0} \\
(\!|\oplus\{l:S\}|\!)^u & \stackrel{\text{def}}{=} & u \triangleleft l.t!\langle u\rangle.\mathbf{0} \\
(\!|\&\{l_i:S_i\}_{i\in I}|\!)^u & \stackrel{\text{def}}{=} & u \triangleright \{l_i : t_i!\langle u\rangle.\mathbf{0}\}_{i\in I} \\
(\!|\mu \mathrm{t}.S|\!)^u & \stackrel{\text{def}}{=} & (\!|S\{\mathrm{end}/\mathrm{t}\}|\!)^u \\
(\!|\mathrm{end}|\!)^u & \stackrel{\text{def}}{=} & \mathbf{0} \\
(\!|\langle S\rangle|\!)^u & \stackrel{\text{def}}{=} & u!\langle (\!|S|\!)_{\mathrm{c}}\rangle.t!\langle u\rangle.\mathbf{0} \\
(\!|\langle L\rangle|\!)^u & \stackrel{\text{def}}{=} & u!\langle (\!|L|\!)_{\mathrm{c}}\rangle.t!\langle u\rangle.\mathbf{0} \\
(\!|U \to \diamond|\!)^u & \stackrel{\text{def}}{=} & u\,(\!|U|\!)_{\mathrm{c}} \\
(\!|U \multimap \diamond|\!)^u & \stackrel{\text{def}}{=} & u\,(\!|U|\!)_{\mathrm{c}}
\end{array}
\qquad
\begin{array}{rcl}
(\!|S|\!)_{\mathrm{c}} & \stackrel{\text{def}}{=} & s \ \ (s \text{ fresh}) \\
(\!|\langle S\rangle|\!)_{\mathrm{c}} & \stackrel{\text{def}}{=} & a \ \ (a \text{ fresh}) \\
(\!|\langle L\rangle|\!)_{\mathrm{c}} & \stackrel{\text{def}}{=} & a \ \ (a \text{ fresh}) \\
(\!|U \to \diamond|\!)_{\mathrm{c}} & \stackrel{\text{def}}{=} & \lambda x.(\!|U|\!)^x \\
(\!|U \multimap \diamond|\!)_{\mathrm{c}} & \stackrel{\text{def}}{=} & \lambda x.(\!|U|\!)^x
\end{array}
$$

(t fresh in all cases)

Fig. 6. Characteristic processes (left) and characteristic values (right).

This rule states that a session environment can input a value if such a value is typed with an input prefix and is either a name m, a *characteristic value* $(\!|U|\!)_{\mathrm{c}}$, 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. The definition of characteristic processes and values is given in Fig. 6.

This refined LTS on typing environments in turn gives rise to a different, refined LTS on processes (cf. Definition Appendix A.5). Note the different notation for standard and refined transitions: $\xrightarrow{s?\langle V\rangle}$ and $\stackrel{s?\langle V\rangle}{\longmapsto}$. In the refined LTS, weak transitions are as expected: we write $\Mapsto$ for the reflexive, transitive closure of $\stackrel{\tau}{\longmapsto}$, $\stackrel{\ell}{\Longmapsto}$ for $\Mapsto\stackrel{\ell}{\longmapsto}\Mapsto$, and $\stackrel{\hat{\ell}}{\Longmapsto}$ for $\stackrel{\ell}{\Longmapsto}$ if $\ell \neq \tau$ and $\Mapsto$ otherwise. Further details on the typed LTSs are given in Appendix A and [17].

Characterising $\cong$ We now recall the definition of *higher-order bisimilarity* and *characteristic bisimilarity*, as jointly introduced in [17]. These bisimilarity relations use two different *trigger processes*:

$$
t \hookleftarrow_{\mathrm{H}} V \stackrel{\text{def}}{=} \begin{cases} t?(x).(\nu\, s)(s?(y).(x\,y) \mid \overline{s}!\langle V\rangle.\mathbf{0}) & \text{if } V \text{ is a first-order value} \\ t?(x).(\nu\, s)(s?(y).(y\,x) \mid \overline{s}!\langle V\rangle.\mathbf{0}) & \text{if } V \text{ is a higher-order value} \end{cases} \tag{1}
$$

$$
t \Leftarrow_{\mathrm{C}} V : U \stackrel{\text{def}}{=} t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid \overline{s}!\langle V\rangle.\mathbf{0}) \tag{2}
$$

The process in (1) is called *higher-order trigger process*, while process in (2) is called *characteristic trigger process*. Notice that while in (1) there is a higher-order input on t, in (2) the variable x does not play any rôle. Process $(\!|U|\!)^y$ is the *characteristic process* of type U, implemented along name y. We use higher-order trigger processes to define *higher-order bisimilarity*:

Definition 3.11 *(Higher-order bisimilarity).* A typed relation $\Re$ is a *higher-order bisimulation* if for all $\Gamma;\Delta_1 \vdash P_1\ \Re\ \Delta_2 \vdash Q_1$

1) Whenever $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\,\widetilde{m_1})n!\langle V_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$, there exist Q_2, V_2, Δ_2' such that $\Gamma;\Delta_2 \vdash Q_1 \stackrel{(\nu\,\widetilde{m_2})n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$ and, for a fresh t,

$$\Gamma;\Delta_1'' \vdash (\nu\,\widetilde{m_1})(P_2 \mid t \hookleftarrow_{\mathrm{H}} V_1)\ \Re\ \Delta_2'' \vdash (\nu\,\widetilde{m_2})(Q_2 \mid t \hookleftarrow_{\mathrm{H}} V_2)$$

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

The largest such bisimulation is called *higher-order bisimilarity*, denoted by $\approx^{\mathrm{H}}$.

We exploit characteristic trigger processes to define *characteristic bisimilarity*:

Definition 3.12 *(Characteristic bisimilarity).* A typed relation $\Re$ is a *characteristic bisimulation* if for all $\Gamma;\Delta_1 \vdash P_1\ \Re\ \Delta_2 \vdash Q_1$,

1) Whenever $\Gamma;\Delta_1 \vdash P_1 \stackrel{(\nu\,\widetilde{m_1})n!\langle V_1:U_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$ then there exist Q_2, V_2, Δ_2' such that $\Gamma;\Delta_2 \vdash Q_1 \stackrel{(\nu\,\widetilde{m_2})n!\langle V_2:U_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$ and, for a fresh t,

$$\Gamma;\Delta_1'' \vdash (\nu\,\widetilde{m_1})(P_2 \mid t \Leftarrow_{\mathrm{C}} V_1 : U_1)\ \Re\ \Delta_2'' \vdash (\nu\,\widetilde{m_2})(Q_2 \mid t \Leftarrow_{\mathrm{C}} V_2 : U_2)$$

2) For all $\Gamma;\Delta_1 \vdash P_1 \stackrel{\ell}{\mapsto} \Delta_1' \vdash P_2$ such that ℓ is not an output, there exist Q_2, Δ_2' such that $\Gamma;\Delta_2 \vdash Q_1 \stackrel{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$ and $\Gamma;\Delta_1' \vdash P_2\ \Re\ \Delta_2' \vdash Q_2$; and

3) The symmetric cases of 1 and 2.

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

We state the following important coincidence result:

Theorem 3.2 *([17]). Typed relations $\cong$, $\approx^{\mathrm{H}}$, and $\approx^{\mathrm{C}}$ coincide for $\mathsf{HO}\pi$ processes.*

Remark 3.2 *(Differences between $\approx^{\mathrm{H}}$ and $\approx^{\mathrm{C}}$).* Although $\approx^{\mathrm{H}}$ and $\approx^{\mathrm{C}}$ are conceptually similar, they differ in the kind of trigger process considered. Because of the application in $t \leftarrow_{\mathrm{H}} V$ (cf. (1)), $\approx^{\mathrm{H}}$ cannot be used to reason about first-order session processes (i.e., processes without higher-order features). In contrast, $\approx^{\mathrm{C}}$ is more general: it can uniformly input characteristic, first- or higher-order values.

An up-to technique As mentioned above, processes that do not use shared names (e.g., those in languages in $\mathsf{C}^{-\mathsf{sh}}$) are deterministic. Internal transitions associated to session interactions or β-reductions are deterministic. To define an auxiliary proof technique that exploits determinacy we require some auxiliary definitions. Recall that $\Gamma; \Delta \vdash P \overset{\tau}{\longmapsto} \Delta' \vdash P'$ denotes an internal (typed) transition.

The following up-to technique, based on determinacy properties, will be useful in proofs (§5).

Notation 2 *(Deterministic transitions). We distinguish two kinds of τ-transitions:* session transitions, *noted $\Gamma; \Delta \vdash P \overset{\tau_{\mathrm{s}}}{\longmapsto} \Delta' \vdash P'$, and* β-transitions, *noted $\Gamma; \Delta \vdash P \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P'$. Intuitively, $\overset{\tau_{\mathrm{s}}}{\longmapsto}$ results from a session communication (i.e., synchronization between two dual endpoints), while $\overset{\tau_\beta}{\longmapsto}$ results from an application. We write $\Gamma; \Delta \vdash P \overset{\tau_{\mathrm{d}}}{\longmapsto} \Delta' \vdash P'$ to denote a session transition or a β-transition. See §A.4 and [17] for formal definitions of $\overset{\tau_\beta}{\longmapsto}$ and $\overset{\tau_{\mathrm{s}}}{\longmapsto}$.*

We have the following determinacy property:

Lemma 3.1 *(τ-Inertness [17]). Suppose $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$ with balanced Δ.*

1) If $\Gamma; \Delta \vdash P \overset{\tau_{\mathrm{d}}}{\longmapsto} \Delta' \vdash P'$ then $\Gamma; \Delta \vdash P \approx^{\mathrm{H}} \Delta' \vdash P'$, with $\Delta \longrightarrow^* \Delta'$.
2) If P is an $\mathsf{HO}\pi^{-\mathsf{sh}}$ process, and $P \longrightarrow^* P'$ then $\Gamma; \Delta \vdash P \approx^{\mathrm{H}} \Delta' \vdash P'$, with $\Delta \longrightarrow^* \Delta'$.

We use Lemma 3.1 to prove Theorem 5.4, the negative result stated in §5.4. This property also enables us to define the following up-to technique, useful in full abstraction proofs. We write $\overset{\tau_{\mathrm{d}}}{\Longmapsto}$ to denote a (possibly empty) sequence of deterministic steps $\overset{\tau_{\mathrm{d}}}{\longmapsto}$. We can finally state:

Lemma 3.2 *(Up-to deterministic transition [17]). Let $\Gamma; \Delta_1 \vdash P_1 \mathrel{\Re} \Delta_2 \vdash Q_1$ such that if whenever:*

1. *$\forall (\nu\, \tilde{m}_1) n!\langle V_1\rangle$ such that $\Gamma; \Delta_1 \vdash P_1 \overset{(\nu\, \tilde{m}_1) n!\langle V_1\rangle}{\longmapsto} \Delta_3 \vdash P_3$ 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 $\Gamma; \Delta_3 \vdash P_3 \overset{\tau_{\mathrm{d}}}{\Longmapsto} \Delta_1' \vdash P_2$ and for fresh t:*
   $\Gamma; \Delta_1'' \vdash (\nu\, \tilde{m}_1)(P_2 \mid t \leftarrow_{\mathrm{H}} V_1) \mathrel{\Re} \Delta_2'' \vdash (\nu\, \tilde{m}_2)(Q_2 \mid t \leftarrow_{\mathrm{H}} V_2)$.
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 $\Gamma; \Delta_3 \vdash P_3 \overset{\tau_{\mathrm{d}}}{\Longmapsto} \Delta_1' \vdash P_2$ and $\Gamma; \Delta_1' \vdash P_2 \mathrel{\Re} \Delta_2' \vdash Q_2$.*
3. *The symmetric cases of 1 and 2.*

Then $\Re \subseteq \approx^{\mathrm{H}}$.

3.4. The hotel booking scenario

We recall the case study for $\mathsf{HO}\pi$ that we developed in our previous works [15,17]: a specification of a *hotel booking scenario*. The scenario involves a Client process that wants to book a hotel room. Client narrows the choice down to two hotels, and requires a quote from the two in order to decide. The round-trip time (RTT) required for taking quotes from the two hotels is not optimal, so the client sends mobile processes to both hotels to automatically negotiate and book a room.

Fig. 7 presents two possible $\mathsf{HO}\pi$ implementations of this scenario. For convenience, we write $\texttt{if}\ e\ \texttt{then}\ P_1\ \texttt{else}\ P_2$ to denote a conditional process that executes P_1 or P_2 depending on boolean expression e (this process is encodable using labelled choice). The first implementation, given by process Client_1, sends two abstractions with body P_{xy}, one to each hotel, using sessions s_1 and s_2. In P_{xy}, name x is meant to be instantiated by the hotel as the negotiating endpoint, whereas name

$$\mathsf{Client}_1 \stackrel{\text{def}}{=} (\nu\, h_1, h_2)(s_1!\langle\lambda x.\, P_{xy}\{^{h_1}/_{y}\}\rangle.s_2!\langle\lambda x.\, P_{xy}\{^{h_2}/_{y}\}\rangle.\mathbf{0} \mid \overline{h_1}?(x).\overline{h_2}?(y).\mathtt{if}\ x \leq y\ \mathtt{then}\ (\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0}\ \mathtt{else}\ \overline{h_1} \triangleleft \mathsf{reject}.\overline{h_2} \triangleleft \mathsf{accept}.\mathbf{0}))$$

$$P_{xy} \stackrel{\text{def}}{=} x!\langle\mathsf{room}\rangle.x?(\mathit{quote}).y!\langle\mathit{quote}\rangle.y \triangleright \left\{ \begin{array}{l} \mathsf{accept} : x \triangleleft \mathsf{accept}.x!\langle\mathsf{credit}\rangle.\mathbf{0}\,, \\ \mathsf{reject} : x \triangleleft \mathsf{reject}.\mathbf{0} \end{array} \right\}$$

$$\mathsf{Client}_2 \stackrel{\text{def}}{=} (\nu\, h)(s_1!\langle\lambda x.\, Q_1\{^{h}/_{y}\}\rangle.s_2!\langle\lambda x.\, Q_2\{^{\overline{h}}/_{y}\}\rangle.\mathbf{0})$$

$$Q_1 \stackrel{\text{def}}{=} x!\langle\mathsf{room}\rangle.x?(\mathit{quote}_1).y!\langle\mathit{quote}_1\rangle.y?(\mathit{quote}_2).R_x$$

$$Q_2 \stackrel{\text{def}}{=} x!\langle\mathsf{room}\rangle.x?(\mathit{quote}_1).y?(\mathit{quote}_2).y!\langle\mathit{quote}_1\rangle.R_x$$

$$R_x \stackrel{\text{def}}{=} \mathtt{if}\ \mathit{quote}_1 \leq \mathit{quote}_2\ \mathtt{then}\ (x \triangleleft \mathsf{accept}.x!\langle\mathsf{credit}\rangle.\mathbf{0}\ \mathtt{else}\ x \triangleleft \mathsf{reject}.\mathbf{0})$$

Fig. 7. Two implementations of the Hotel Booking scenario in HOπ [17].

y is used to interact with Client_1. Intuitively, process P_{xy}: (i) sends the room requirements to the hotel; (ii) receives a quote from the hotel; (iii) sends the quote to Client_1; (iv) expects a choice from Client_1 whether to accept or reject the offer; (v) if the choice is accept then it informs the hotel and performs the booking; otherwise, if the choice is reject then it informs the hotel and ends the session. Client_1 instantiates two copies of P_{xy} as abstractions on session x. It uses fresh endpoints h_1, h_2 to substitute channel y in P_{xy}. This enables communication with the mobile code(s): Client_1 uses the dual endpoints $\overline{h_1}$ and $\overline{h_2}$ to receive the negotiation result from the two remote instances of P and then inform the two processes for the final booking decision.

In the second implementation, given by process Client_2, the two mobile processes reach an agreement by interacting with each other (rather than with the client). Processes Q_1 and Q_2 negotiate a quote from the hotel in the same fashion as process P_{xy} in Client_1. The key difference with respect to P_{xy} is that y is used for interaction between process Q_1 and Q_2. Both processes send their quotes to each other and then internally follow the same logic to reach to a decision. Process Client_2 then uses sessions s_1 and s_2 to send the two instances of Q_1 and Q_2 to the two hotels, using them as abstractions on name x. It further substitutes the two endpoints of a fresh channel h to channels y respectively, in order for the two instances to communicate with each other.

To illustrate the type system of HOπ, we give types to the client processes. Assume

$$S = !\langle\mathsf{quote}\rangle; \&\{\mathsf{accept} : \mathtt{end}, \mathsf{reject} : \mathtt{end}\}$$

$$U = !\langle\mathsf{room}\rangle; ?(\mathsf{quote}); \oplus\{\mathsf{accept} : !\langle\mathsf{credit}\rangle; \mathtt{end}, \mathsf{reject} : \mathtt{end}\}$$

where quote, room, and credit are (first-order) base types. We then have:

$$\emptyset; \emptyset; y : S \vdash \lambda x.\, P_{xy} \triangleright U \multimap \diamond$$

$$\emptyset; \emptyset; s_1 : !\langle U \multimap \diamond\rangle; \mathtt{end} \cdot s_2 : !\langle U \multimap \diamond\rangle; \mathtt{end} \vdash \mathsf{Client}_1 \triangleright \diamond$$

$$\emptyset; \emptyset; y : !\langle\mathsf{quote}\rangle; ?(\mathsf{quote}); \mathtt{end} \vdash \lambda x.\, Q_i \triangleright U \multimap \diamond \quad (i = 1, 2)$$

$$\emptyset; \emptyset; s_1 : !\langle U \multimap \diamond\rangle; \mathtt{end} \cdot s_2 : !\langle U \multimap \diamond\rangle; \mathtt{end} \vdash \mathsf{Client}_2 \triangleright \diamond$$

4. Correctness criteria for typed encodings

We define the formal notion of *encoding* by extending to a typed setting existing encodability criteria for untyped processes, as put forward in, e.g., [29,31,35,10,20,7,46,34]. We first define a *typed calculus* parametrised by a process syntax, an operational semantics, and a type system. Based on this definition, in §5 and §6 we will define concrete instances of (higher-order) typed calculi.

4.1. Basic definitions

Definition 4.1 *(Typed calculus).* A *typed calculus* $\mathcal{L}$ is a tuple $\langle \mathsf{C}, \mathcal{T}, \mapsto, \approx, \vdash\rangle$ where C and $\mathcal{T}$ are sets of processes and types, respectively; also, $\mapsto$, $\approx$, and $\vdash$ denote a transition system (over an underlying set of actions, denoted $\mathcal{A}$), a typed equivalence, and a typing system for C, respectively.

Most elements of the formal notion of typed calculus are self-explanatory. Concerning the operational semantics, we shall assume a notion of transition system in which transitions are labelled with elements from a finite set of actions $\mathcal{A}$, which contains at least the unobservable action τ. We will often be interested in τ-transitions, denoted $\stackrel{\tau}{\mapsto}$, which characterise

reductions. Nevertheless, to state more precise forms of operational correspondence, we will sometimes find it convenient to use transitions of the form $\overset{\ell}{\mapsto}$, where $\ell \in \mathcal{A}$ and $\ell \neq \tau$ (i.e., visible transitions).

Our notion of encoding considers mappings on both processes and types; these are denoted $[\![\cdot]\!]$ and $(\!|\cdot|\!)$, respectively:

Definition 4.2 *(Typed encoding).* Consider two typed calculi $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \mapsto_1, \approx_1, \vdash_1 \rangle$ and $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \mapsto_2, \approx_2, \vdash_2 \rangle$. 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 $\mathcal{L}_1$ (the *source calculus*) into $\mathcal{L}_2$ (the *target calculus*). Mapping $(\!|\cdot|\!)$ on types extends to typing environments in the expected way.

When considering forms of operational correspondence with visible actions, our notion of typed encoding shall include mappings $[\![\cdot]\!]$ and $(\!|\cdot|\!)$, but also a mapping $\{\!|\cdot|\!\} : \mathcal{A}_1 \to \mathcal{A}_2$ describing how visible actions in the source calculus $\mathcal{L}_1$ are mapped in the target calculus $\mathcal{L}_2$.

We now 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. In the case of HOπ and its variants, $\sigma(\Gamma)$ clearly concerns only shared names in Γ: process and recursive variables in Γ are not affected by σ.

Definition 4.3 *(Syntax preservation).* We say that the typed 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., [31,35]) expresses that translations 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 the typing rule for parallel composition (cf. Rule (Par) in Fig. 5). Compositionality wrt restriction is also supported by typing and is useful in our encodability results (§5). The name invariance criterion follows [10,20].

We now state *type preservation*, a static criterion on the mapping $(\!|\cdot|\!) : \mathcal{T}_1 \to \mathcal{T}_2$: it ensures that a typed operator is always translated into itself. The source and target calculi that we consider here share five (session) type operators: input, output, recursion (binary operators); selection and branching (n-ary operators). As such, type preservation is key to retain the meaning of structured protocols: as session types operators abstract communication behaviour, type preserving encodings help us maintain behaviour across translations.

Definition 4.4 *(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 $\mathtt{op}$ in $\mathcal{T}_1$ it holds that

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

Example 4.1. Following the discussion in §2, let $(\!|\cdot|\!)_u$ be a mapping on session types such that

$$(\!|!\langle U\rangle; S|\!)_u = ?((\!|U|\!)_u); (\!|S|\!)_u$$
$$(\!|?(U); S|\!)_u = !\langle (\!|U|\!)_u\rangle; (\!|S|\!)_u$$

and other type operators are translated homomorphically. Since $(\!|\cdot|\!)_u$ translates the output type operator into an input type operator (and viceversa), it does not satisfy type preservation.

Next we define semantic criteria for typed encodings. Recall that (un)typed barbs have been defined in Definition 3.7.

Definition 4.5 *(Semantic preservation).* Consider typed calculi $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \mapsto_1, \approx_1, \vdash_1 \rangle$ and $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \mapsto_2, \approx_2, \vdash_2 \rangle$. We say that the typed 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 $(\!|\Gamma|\!); \emptyset; (\!|\Delta|\!) \vdash_2 [\![P]\!] \triangleright \diamond$.
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 $\Gamma; \Delta \vdash_1 P \overset{\tau}{\mapsto}_1 \Delta' \vdash_1 P'$ then $\exists Q, \Delta''$ such that
      (i) $(\!|\Gamma|\!); (\!|\Delta|\!) \vdash_2 [\![P]\!] \mapsto\!\!\!\Rightarrow_2 (\!|\Delta''|\!) \vdash_2 Q$ and
      (ii) $(\!|\Gamma|\!); (\!|\Delta'|\!) \vdash_2 [\![P']\!] \approx_2 (\!|\Delta''|\!) \vdash_2 Q$.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

(b) Soundness: If $(\!|\Gamma|\!); (\!|\Delta|\!) \vdash_2 [\![P]\!] \Rightarrow_2 (\!|\Delta'|\!) \vdash_2 Q$ then $\exists P', Q', \Delta'', \Delta'''$ such that
(i) $\Gamma; \Delta \vdash_1 P \overset{\tau}{\mapsto}_1 \Delta'' \vdash_1 P'$,
(ii) $(\!|\Gamma|\!); (\!|\Delta'|\!) \vdash_2 Q \Rightarrow_2 (\!|\Delta'''|\!) \vdash_2 Q'$ and
(iii) $(\!|\Gamma|\!); (\!|\Delta''|\!) \vdash_2 [\![P']\!] \approx_2 (\!|\Delta'''|\!) \vdash_2 Q'$.

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 (Definition 4.4), type soundness is a distinguishing encodability criterion. Barb preservation, related to success sensitiveness in [10], is convenient in our developments as all considered calculi have the same notion of barb. Operational correspondence, standardly divided into completeness and soundness, is also based on [10]; it relies on τ-transitions (reductions). Completeness ensures that a step of the source process is mimicked by a step of its associated encoding. Soundness is the converse of completeness; the formulation given above is called *weak soundness* in [36].

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 Propositions 5.2, 5.5, 6.2, and 6.5). In particular, we will consider forms of operational correspondence that account also for visible actions, relying on a mapping $\{\!|\cdot|\!\}$ on actions, as already explained (cf. Definition 4.7 below). Finally, following [39,35,51], we consider full abstraction as an encodability criterion: this leads to stronger encodability results.

4.2. Precise, minimal, and tight encodings

We may now introduce *precise*, *minimal*, and *tight* encodings. While we state strong positive encodability results in terms of *precise* encodings, to prove the non-encodability result in § 5.4, we appeal to the weaker *minimal* encodings. Also, to compare two precise encodings in § 5.3 here we introduce the notion of *tight* encodings.

Definition 4.6 *(Typed encodings: precise and minimal).* Let $\langle [\![\cdot]\!], (\!|\cdot|\!) \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ be a typed encoding.

- We say that the typed encoding is *precise*, if it is syntax, type, and semantic preserving (Definitions 4.3, 4.4, 4.5).
- We say that the typed encoding is *minimal*, if it is syntax preserving (Definition 4.3), barb preserving (Definition 4.5(2)), and operationally complete (Definition 4.5(3)(a)).

The following property, concerning composability of precise encodings, will come in handy in § 6. It follows closely a similar property established in [9] for (untyped) valid encodings between languages with equivalences which are reduction-closed.

Proposition 4.1 *(Composability). Assume typed calculi $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ whose typed equivalences ($\approx_1$, $\approx_2$, and $\approx_3$, respectively) are reduction-closed. 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 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.*

Proof. The proof follows directly from the definitions, and is very similar to the proof of Proposition 10 in [9]. □

We now introduce the notion of *tight encodings*, which refine precise encodings with extra correctness criterion: a form of operational correspondence for *visible actions*. As already motivated above, we write ℓ_1, ℓ_2 to denote actions different from τ, and $\overset{\ell}{\mapsto}$ (resp. $\overset{\ell}{\Longmapsto}$) to denote a (weak) visible transition; recall that $\{\!|\cdot|\!\}$ stands for a mapping on action labels.

Definition 4.7 *(Labelled correspondence / tight encodings).* Consider typed calculi $\mathcal{L}_1$ and $\mathcal{L}_2$, defined as $\mathcal{L}_1 = \langle \mathsf{C}_1, \mathcal{T}_1, \mapsto_1, \approx_1, \vdash_1 \rangle$ and $\mathcal{L}_2 = \langle \mathsf{C}_2, \mathcal{T}_2, \mapsto_2, \approx_2, \vdash_2 \rangle$. The encoding $\langle [\![\cdot]\!], (\!|\cdot|\!) \rangle : \mathcal{L}_1 \to \mathcal{L}_2$ satisfies *labelled operational correspondence* if it satisfies:

1. If $\Gamma; \Delta \vdash_1 P \overset{\ell_1}{\mapsto}_1 \Delta' \vdash_1 P'$ then $\exists Q$, Δ'', ℓ_2 such that:
(i) $(\!|\Gamma|\!); (\!|\Delta|\!) \vdash_2 [\![P]\!] \overset{\ell_2}{\Longmapsto}_2 (\!|\Delta''|\!) \vdash_2 Q$; (ii) $\ell_2 = \{\!|\ell_1|\!\}$; and
(iii) $(\!|\Gamma|\!); (\!|\Delta''|\!) \vdash_2 Q \approx_2 (\!|\Delta'|\!) \vdash_2 [\![P']\!]$.
2. If $(\!|\Gamma|\!); (\!|\Delta|\!) \vdash_2 [\![P]\!] \overset{\ell_2}{\Longmapsto}_2 (\!|\Delta'|\!) \vdash_2 Q$ then $\exists P'$, Q', Δ'', Δ''', ℓ_1 such that:
(i) $\Gamma; \Delta \vdash_1 P \overset{\ell_1}{\mapsto}_1 \Delta'' \vdash_1 P'$; (ii) $\ell_2 = \{\!|\ell_1|\!\}$; (iii) $(\!|\Gamma|\!); (\!|\Delta'|\!) \vdash_2 Q \Rightarrow_2 (\!|\Delta'''|\!) \vdash_2 Q'$ (iv) $(\!|\Gamma|\!); (\!|\Delta''|\!) \vdash_2 [\![P']\!] \approx_2 (\!|\Delta'''|\!) \vdash_2 Q'$.

A *tight encoding* is a typed encoding which is precise (Definition 4.6) and that also satisfies labelled operational correspondence as above.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

$$\begin{aligned}
\llbracket w!\langle\lambda x.\, Q\rangle.P \rrbracket_\sigma &\stackrel{\text{def}}{=} u!\langle\lambda x.\, \llbracket Q \rrbracket_{\sigma\cdot x}\rangle.\llbracket P \rrbracket_\sigma & \llbracket w \triangleright \{l_i : P_i\}_{i\in I} \rrbracket_\sigma &\stackrel{\text{def}}{=} u \triangleright \{l_i : \llbracket P_i \rrbracket_\sigma\}_{i\in I}\\
\llbracket w?(x).P \rrbracket_\sigma &\stackrel{\text{def}}{=} u?(x).\llbracket P \rrbracket_\sigma & \llbracket w \triangleleft l.P \rrbracket_\sigma &\stackrel{\text{def}}{=} u \triangleleft l.\llbracket P \rrbracket_\sigma\\
\llbracket (\nu\, n)P \rrbracket_\sigma &\stackrel{\text{def}}{=} (\nu\, n)\llbracket P \rrbracket_{\sigma\cdot n} & \llbracket (\lambda x.Q)\, w \rrbracket_\sigma &\stackrel{\text{def}}{=} (\lambda x.\llbracket Q \rrbracket_{\sigma\cdot x})\, u\\
\llbracket P \mid Q \rrbracket_\sigma &\stackrel{\text{def}}{=} \llbracket P \rrbracket_\sigma \mid \llbracket Q \rrbracket_\sigma & \llbracket x\, w \rrbracket_\sigma &\stackrel{\text{def}}{=} x\, u\\
\llbracket \mathbf{0} \rrbracket_\sigma &\stackrel{\text{def}}{=} \mathbf{0}
\end{aligned}$$

$$\text{In all cases: } u = \begin{cases} x_n & \text{if } w \text{ is a name } n \text{ and } n \notin \sigma \ (x \text{ fresh})\\ w & \text{otherwise: } w \text{ is a variable or a name } n \text{ and } n \in \sigma \end{cases}$$

Fig. 8. Auxiliary mapping used to encode HOπ into HO (Definition 5.1).

This way, the notion of labelled correspondence complements/generalizes the notions of operational soundness and completeness given in Definition 4.5, which is restricted to τ-labelled transitions.

5. Expressiveness results for HOπ, HO, and π

In this section, we present two precise encodings: (1) higher-order communication with recursion and name-passing (HOπ) into higher-order communication without name-passing nor recursion (HO) (§5.1); and (2) HOπ into the first-order calculus with name-passing with recursion (π) (§5.2). We then compare these encodings (§5.3). Moreover, in §5.4 we state our impossibility result for shared/linear names. We consider the following typed calculi, which result as three instances of Definition 4.1:

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

where $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ are sets of types of HOπ, HO, and π, respectively. The typing $\vdash$ is defined in §3.2. The LTSs follow the intuitions given in §3.3.2. The set of actions $\mathcal{A}_{\mathsf{HO}\pi}$ is as in Definition 3.10; the sets of actions $\mathcal{A}_{\mathsf{HO}}$ and $\mathcal{A}_\pi$ are obtained from $\mathcal{A}_{\mathsf{HO}\pi}$ as expected, considering the differences in the syntax of values V. Moreover, higher-order and characteristic bisimilarities $\approx^{\mathrm{H}}$ and $\approx^{\mathrm{C}}$ are as in Definition 3.11 and Definition 3.12.

Remark 5.1 *(Type Annotations (2)).* In encodings, we sometimes type-annotate bound variables in order to distinguish first- and higher-order values and processes. This way, e.g., we may write $u?(x{:}C).P$ and $u?(x{:}L).P$ to denote first- and higher-order input prefixed processes, respectively.

5.1. Precise encoding of HOπ into HO

HO is expressive enough to precisely encode HOπ. As discussed above, 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 the auxiliary mappings $\|\cdot\|$ and $\big\|\!\big[\cdot \big]\!\big\|_\sigma$: the former maps sequences of session names into sequences of variables; the second maps processes with free names to processes without free names (but with free variables instead):

Definition 5.1 *(Auxiliary mappings).* We define mappings $\|\cdot\|$ and $\big\|\!\big[\cdot \big]\!\big\|_\sigma$ as follows:

- $\|\cdot\| : 2^{\mathcal{N}} \longrightarrow \mathcal{V}^\omega$ is a map of sequences of lexicographically ordered names to sequences of variables, defined inductively as:

$$\begin{aligned}
\|\epsilon\| &= \epsilon\\
\|n\cdot\tilde{m}\| &= x_n \cdot \|\tilde{m}\| \quad (x \text{ fresh})
\end{aligned}$$

- Given a set of session names and variables σ, the map $\big\|\!\big[\cdot \big]\!\big\|_\sigma : \mathsf{HO} \to \mathsf{HO}$ is as in Fig. 8.

Let P be an HOπ process with $\mathtt{fn}(P) = \{n_1, \cdots, n_k\}$. Intuitively, our encoding $[\![\cdot]\!]^1_f$ exploits the abstraction $\lambda x_1, \cdots, x_k.\, \big\|\!\big[[\![P]\!]^1_f \big]\!\big\|_\emptyset$, where $x_j = \|n_j\|$, for all $j \in \{1, \ldots, k\}$:

Terms:

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

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

$$\llbracket u!\langle\lambda x.\, Q\rangle.P\rrbracket^1_f \stackrel{\text{def}}{=} u!\langle\lambda x.\, \llbracket Q\rrbracket^1_f\rangle.\llbracket P\rrbracket^1_f$$

$$\llbracket u?(x{:}L).P\rrbracket^1_f \stackrel{\text{def}}{=} u?(x).\llbracket P\rrbracket^1_f$$

$$\llbracket s \triangleleft l.P\rrbracket^1_f \stackrel{\text{def}}{=} s \triangleleft l.\llbracket P\rrbracket^1_f$$

$$\llbracket s \triangleright \{l_i{:}P_i\}_{i\in I}\rrbracket^1_f \stackrel{\text{def}}{=} s \triangleright \{l_i : \llbracket P_i\rrbracket^1_f\}_{i\in I}$$

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

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

$$\llbracket x\, u\rrbracket^1_f \stackrel{\text{def}}{=} x\, u$$

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

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

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

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

Above, $\mathtt{fn}(P)$ is a lexicographically ordered sequence of free names in P. Map $\big\|\cdot\big\|_\sigma$ is given in Definition 5.1 and Fig. 8.

Types:

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

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

$$\lfloor C \to \diamond\rfloor^1 \stackrel{\text{def}}{=} \langle\!|C|\!\rangle^1 \to \diamond$$

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

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

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

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

$$\langle\!|\mathtt{end}|\!\rangle^1 \stackrel{\text{def}}{=} \mathtt{end}$$

Fig. 9. Encoding of HOπ into HO (Definition 5.2).

Definition 5.2 *(Typed encoding of* HOπ *into* HO*).* Let f be a map from process variables to sequences of name variables. The typed encoding $\langle\llbracket\cdot\rrbracket^1 f, \langle\!|\cdot|\!\rangle^1\rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\mathsf{HO}}$ is given in Fig. 9. Mapping $\langle\!|\cdot|\!\rangle^1$ on types homomorphically extends to environments Δ and Γ, with

$$\langle\!|\Gamma \cdot X : \{n_i : S_i\}_{1\le i\le m}|\!\rangle^1 = \langle\!|\Gamma|\!\rangle^1 \cdot z_X : (\langle\!|S_1|\!\rangle^1, \ldots, \langle\!|S_m|\!\rangle^1, S^*) \to \diamond$$

where S^* is defined as $\mu\mathrm{t}.?((\langle\!|S_1|\!\rangle^1, \ldots, \langle\!|S_m|\!\rangle^1, \mathrm{t}) \to \diamond);\, \mathtt{end}$.

Observe that the encoding of types $\langle\!|\cdot|\!\rangle^1$ depends on an auxiliary encoding for value types, denoted $\lfloor\cdot\rfloor^1$. Notice also that Δ in $X : \Delta$ is mapped to a non-tail recursive session type with variable z_X. Non-tail recursive session types were studied in [4,1]; to our knowledge, this is the first application in the context of higher-order session types. For convenience, we use polyadic name abstractions $\lambda x_1, \ldots, x_k.\, P$, with $k \ge 2$ (sometimes also denoted as $\lambda(x_1, \ldots, x_k).\, P$). A precise encoding of polyadicity into HOπ is given in § 6.2 (see also Corollary 6.2 to its extension to HO).

Key elements in Fig. 9 are encodings of *name passing* ($\llbracket u!\langle w\rangle.P\rrbracket^1_f$ and $\llbracket u?(x).P\rrbracket^1_f$) and *recursion* ($\llbracket\mu X.P\rrbracket^1_f$ and $\llbracket X\rrbracket^1_f$). As motivated in § 2, a name w is passed as an input-guarded abstraction; on the receiver side, the encoding 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; here, we assume that for each recursive variable X_i there is a fresh variable z_{X_i}. Then, using the auxiliary mapping $\big\|\cdot\big\|_\sigma$ in Definition 5.1, 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 $\big\|\cdot\big\|_\sigma$.) 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 recover 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 two examples: the first illustrates our strategy for encoding recursion, while the second illustrates the strategy for first-order session communication.

Example 5.1 *(Encoding recursion).* Let $\mu X.a!\langle m\rangle.X$ be an HOπ process. Its encoding into HO is given next; notice that $f = \emptyset$ and $f' = X \to a\,m$.

$$\begin{aligned}
[\![\mu X.a!\langle m\rangle.X]\!]^1_f &= (\nu\, s_1)\big(\overline{s_1}!\langle\lambda(x_a, x_m, y_1).\, y_1?(z_X).\big\|[\![a!\langle m\rangle.X]\!]^1_{f'}\big\|_{\emptyset}\rangle.\mathbf{0} \mid s_1?(z_X).[\![a!\langle m\rangle.X]\!]^1_{f'}\big)\\
[\![a!\langle m\rangle.X]\!]^1_{f'} &= a!\langle\lambda z_1.\, z_1?(x).(x m)\rangle.(\nu\, s_2)(z_X\,(a, m, s_2) \mid \overline{s_2}!\langle z_X\rangle.\mathbf{0})\\
\big\|[\![a!\langle m\rangle.X]\!]^1_{f'}\big\|_{\emptyset} &= x_a!\langle\lambda z_1.\, z_1?(x).(x\, x_m)\rangle.(\nu\, s_2)(z_X\,(x_a, x_m, s_2) \mid \overline{s_2}!\langle z_X\rangle.\mathbf{0})
\end{aligned}$$

This way, by writing V to denote the abstraction

$$\lambda(x_a, x_m, y_1).\, y_1?(z_X).x_a!\langle\lambda z_1.\, z_1?(x).(x\, x_m)\rangle.(\nu\, s_2)(z_X\,(x_a, x_m, s_2) \mid \overline{s_2}!\langle z_X\rangle.\mathbf{0})$$

we would have

$$[\![\mu X.a!\langle m\rangle.X]\!]^1_f = (\nu\, s_1)\big(\overline{s_1}!\langle V\rangle.\mathbf{0} \mid s_1?(z_X).a!\big\langle\lambda z_1.\, z_1?(x).(x m)\big\rangle.(\nu\, s_2)(z_X\,(a, m, s_2) \mid \overline{s_2}!\langle z_X\rangle.\mathbf{0})\big)$$

Next we illustrate the behaviour of $[\![\mu X.a!\langle m\rangle.X]\!]^1_f$; below ℓ stands for $a!\langle\lambda z.\, z?(x).(x m)\rangle$.

$$\begin{aligned}
[\![\mu X.a!\langle m\rangle.X]\!]^1_f &\overset{\tau}{\mapsto} a!\langle\lambda z_1.\, z_1?(x).(x m)\rangle.(\nu\, s_2)(V\,(a, m, s_2) \mid \overline{s_2}!\langle V\rangle.\mathbf{0})\\
&\overset{\ell}{\mapsto} (\nu\, s_2)(V\,(a, m, s_2) \mid \overline{s_2}!\langle V\rangle.\mathbf{0})\\
&\overset{\tau}{\mapsto} (\nu\, s_2)\big(s_2?(z_X).a!\big\langle\lambda z_1.\, z_1?(x).(x m)\big\rangle.(\nu\, s_3)(z_X\,(a, m, s_3) \mid \overline{s_3}!\langle z_X\rangle.\mathbf{0}) \mid \overline{s_2}!\langle V\rangle.\mathbf{0}\big)\\
&\equiv (\nu\, s_2)\big(\overline{s_2}!\langle V\rangle.\mathbf{0} \mid s_2?(z_X).a!\big\langle\lambda z_1.\, z_1?(x).(x m)\big\rangle.(\nu\, s_3)(z_X\,(a, m, s_3) \mid \overline{s_3}!\langle z_X\rangle.\mathbf{0})\big)\\
&\equiv_\alpha [\![\mu X.a!\langle m\rangle.X]\!]^1_f
\end{aligned}$$

Example 5.2 *(Encoding a hotel booking client).* The HOπ process Client_2 (cf. Fig. 7) is one possible implementation for the hotel booking scenario described in § 3.4. Its encoding in HO is as follows:

$$\begin{aligned}
[\![\mathsf{Client}_2]\!]^1_f &= [\![(\nu\, h)(s_1!\langle\lambda x.\, Q_1\{^h/_y\}\rangle.s_2!\langle\lambda x.\, Q_2\{^{\overline{h}}/_y\}\rangle.\mathbf{0})]\!]^1_f\\
&= (\nu\, h)(s_1!\langle\lambda x.\, [\![Q_1\{^h/_y\}]\!]^1_f\rangle.s_2!\langle\lambda x.\, [\![Q_2\{^{\overline{h}}/_y\}]\!]^1_f\rangle.\mathbf{0})
\end{aligned}$$

where $[\![Q_1]\!]^1_f$ and $[\![Q_2]\!]^1_f$ are given in Fig. 10.

We now state the properties of the encoding. We start with type preservation and type soundness:

Proposition 5.1 *(HOπ into HO: type preservation and type soundness). The encoding from $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_{\mathsf{HO}}$ (cf. Definition 5.2) is type preserving (cf. Definition 4.4) and type sound (cf. Definition 4.5(1)).*

Proof. Type preservation follows directly from Fig. 9. Type soundness is shown by induction on the inference of $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. See Proposition Appendix B.1 (Page 35) in B.1. □

We now state a generalised form of operational correspondence, which includes τ-labelled transitions (reductions) but also visible actions. To this end, we define a mapping on action labels:

Definition 5.3. Given the typed encoding $\langle [\![\cdot]\!]^1_f, (\!|\cdot|\!)^1\rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\mathsf{HO}}$ (cf. Definition 5.2), the mapping on actions $\{\!|\cdot|\!\}^1 : \mathcal{A}_{\mathsf{HO}\pi} \to \mathcal{A}_{\mathsf{HO}}$ is defined as follows:

$$\begin{aligned}
\{\!|(\nu\tilde{m})n!\langle m\rangle|\!\}^1 &\overset{\mathtt{def}}{=} (\nu\tilde{m})n!\langle\lambda z.\, z?(x).(x m)\rangle\\
\{\!|n?\langle m\rangle|\!\}^1 &\overset{\mathtt{def}}{=} n?\langle\lambda z.\, z?(x).(x m)\rangle\\
\{\!|(\nu\tilde{m})n!\langle\lambda x.\, P\rangle|\!\}^1 &\overset{\mathtt{def}}{=} (\nu\tilde{m})n!\langle\lambda x.\, [\![P]\!]^1_{\emptyset}\rangle\\
\{\!|n?\langle\lambda x.\, P\rangle|\!\}^1 &\overset{\mathtt{def}}{=} n?\langle\lambda x.\, [\![P]\!]^1_{\emptyset}\rangle
\end{aligned}$$

and as an homomorphism for other actions $\ell \in \mathcal{A}_{\mathsf{HO}\pi}$.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

$$
\begin{aligned}
[\![Q_1]\!]^1_f &= [\![x!\langle \mathsf{room}\rangle.x?(quote_1).y!\langle quote_1\rangle.y?(quote_2).R_x]\!]^1_f \\
&= x!\langle \lambda z_1.\ z_1?(x_1).(x_1\ \mathsf{room})\rangle.[\![x?(quote_1).y!\langle quote_1\rangle.y?(quote_2).R_x]\!]^1_f \\
&= x!\langle \lambda z_1.\ z_1?(x_1).(x_1\ \mathsf{room})\rangle.x?(y_1).(\nu\, s_1)\big(y_1\, s_1 \mid \\
&\quad \overline{s_1}!\langle \lambda quote_1.\ [\![y!\langle quote_1\rangle.y?(quote_2).R_x]\!]^1_f\rangle.\mathbf{0}\big) \\
&= x!\langle \lambda z_1.\ z_1?(x_1).(x_1\ \mathsf{room})\rangle.x?(y_1).(\nu\, s_1)\big(y_1\, s_1 \mid \\
&\quad \overline{s_1}!\langle \lambda quote_1.\ y!\langle \lambda z_2.\ z_2?(x_2).(x_2\ quote_1)\rangle.[\![y?(quote_2).R_x]\!]^1_f\rangle.\mathbf{0}\big) \\
&= x!\langle \lambda z_1.\ z_1?(x_1).(x_1\ \mathsf{room})\rangle.x?(y_1).(\nu\, s_1)\big(y_1\, s_1 \mid \\
&\quad \overline{s_1}!\langle \lambda quote_1.\ y!\langle \lambda z_2.\ z_2?(x_2).(x_2\ quote_1)\rangle.y?(u_1).(\nu\, s_2)(u_1\, s_2 \mid \overline{s_2}!\langle \lambda quote_2.\ [\![R_x]\!]^1_f\rangle.\mathbf{0})\rangle.\mathbf{0}\big) \\
[\![Q_2]\!]^1_f &= [\![x!\langle \mathsf{room}\rangle.x?(quote_1).y?(quote_2).y!\langle quote_1\rangle.R_x]\!]^1_f \\
&= x!\langle \lambda z_1.\ z_1?(x_1).(x_1\ \mathsf{room})\rangle.x?(y_1).(\nu\, s_1)\big(y_1\, s_1 \mid \\
&\quad \overline{s_1}!\langle \lambda quote_1.\ y?(u_1).(\nu\, s_2)(u_1\, s_2 \mid \overline{s_2}!\langle \lambda quote_2.\ y!\langle \lambda z_2.\ z_2?(x_2).(x_2\ quote_1)\rangle.[\![R_x]\!]^1_f\rangle.\mathbf{0})\rangle.\mathbf{0}\big)
\end{aligned}
$$

with

$$
\begin{aligned}
[\![R_x]\!]^1_f &= [\![\mathtt{if}\ quote_1 \le quote_2\ \mathtt{then}\ (x \triangleleft \mathsf{accept}.x!\langle \mathsf{credit}\rangle.\mathbf{0}\ \mathtt{else}\ x \triangleleft \mathsf{reject}.\mathbf{0})]\!]^1_f \\
&= \mathtt{if}\ quote_1 \le quote_2\ \mathtt{then}\ [\![(x \triangleleft \mathsf{accept}.x!\langle \mathsf{credit}\rangle.\mathbf{0}\ \mathtt{else}\ x \triangleleft \mathsf{reject}.\mathbf{0})]\!]^1_f \\
&= \mathtt{if}\ quote_1 \le quote_2\ \mathtt{then}\ (x \triangleleft \mathsf{accept}.x!\langle \lambda z.\ z?(x).(x\,\mathsf{credit})\rangle.\mathbf{0}\ \mathtt{else}\ x \triangleleft \mathsf{reject}.\mathbf{0})
\end{aligned}
$$

Fig. 10. Encodings of the hotel booking clients (Example 5.2).

We then have:

Proposition 5.2 *(Operational 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 such that $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash R$, with $\ell_2 = (\!|\ell_1|\!)^1$, and $(\!|\Gamma|\!)^1;(\!|\Delta'|\!)^1 \vdash R \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta'|\!)^1 \vdash [\![P_0]\!]^1_f\{^m/_x\}$.*
   d) *If $\ell_1 = \tau$ and $P \equiv (\nu\,\tilde{m})(n!\langle m\rangle.P_1 \mid n?(x).P_2)$ and $P' = (\nu\,\tilde{m})(P_1 \mid P_2\{^m/_x\})$ then $\exists R$ such that $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R)$, and $(\!|\Gamma|\!)^1;(\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R) \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^m/_x\})$.*
   e) *If $\ell_1 = \tau$ and $P \equiv (\nu\,\tilde{m})(n!\langle \lambda y.\ Q\rangle.P_1 \mid n?(x).P_2)$ and $P' = (\nu\,\tilde{m})(P_1 \mid P_2\{^{\lambda y.\ Q}/_x\})$ 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 $P \equiv (\nu\,\tilde{m})((\lambda x.\ P_1)\ V)$ and $P' = (\nu\,\tilde{m})(P_1\{^V/_x\})$ 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 $(\!|\Gamma|\!)^1;(\!|\Delta'|\!)^1 \vdash Q \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta''|\!)^1 \vdash [\![P'\{^m/_z\}]\!]^1_f$*

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

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

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

Proof. By transition induction. See Proposition Appendix B.2 (Page 37) in B.1. □

In the above proposition, it is worth observing how we can explicitly distinguish the role of finite, deterministic reductions ($\overset{\tau_s}{\longmapsto}$ and $\overset{\tau_\beta}{\longmapsto}$, cf. Not. 2) in soundness and completeness statements.

The typed operational correspondence given above is an important component in the proof of *full abstraction*, which we state next.

Proposition 5.3 *(*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* $(\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \approx^{\mathrm{H}} (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]^1_f$.

Proof. The proof of both directions proceeds coinductively. See Proposition Appendix B.3 (Page 40) in B.1. □

We may state the main result of this section:

Theorem 5.1 *(Precise encoding of* HOπ *into* HO*). The encoding from* $\mathcal{L}_{\mathsf{HO}\pi}$ *into* $\mathcal{L}_{\mathsf{HO}}$ *(cf. Definition 5.2) is precise.*

Proof. According to Definition 4.6, preciseness includes syntax-, type-, and semantics-preservation. Syntax preservation follows immediately from the definition of the encoding. Type preservation follows from Proposition 5.1 (Page 16). Semantics-preservation follows from Proposition 5.2 (Page 17) and Proposition 5.3 (Page 18). □

5.2. Precise encoding of HOπ into π

We now discuss the precise encodability of HOπ into π; the only non trivial issue is encoding higher-order communication, which is present in HOπ but not in π. We closely follow Sangiorgi's encoding [39,42], which represents the exchange of a process/abstraction by passing around a fresh *trigger name*. Trigger names may then be used to activate copies of the abstraction, which becomes a persistent resource represented by an input-guarded replication.

The process mapping $[\![\cdot]\!]^2$, which we now informally discuss, casts this strategy in the setting of session-typed communications. In the presence of session names (which are linear and cannot be replicated), our approach uses replicated names as triggers for shared resources and non-replicated names for linear resources. The encoding of abstraction sending therefore distinguishes two cases:

$$[\![u!\langle\lambda x.\, Q\rangle.P]\!]^2 \overset{\mathrm{def}}{=} \begin{cases} (\nu\, a)(u!\langle a\rangle.([\![P]\!]^2 \mid *a?(y).y?(x).[\![Q]\!]^2)) & \text{if } \mathtt{fs}(Q) = \emptyset \\ (\nu\, a)(u!\langle a\rangle.([\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2)) & \text{otherwise} \end{cases}$$

where $*P$ stands for $\mu X.(P \mid X)$ (Not. 1). In the first case, if the abstraction body does not contain (linear) session names then it can be safely represented as a persistent server accessible via a (fresh) trigger name a, which is sent in place of the abstraction. The second case covers the case in which the abstraction to be passed around is linear: the server on a should be invoked exactly once—it cannot be persistent. In this scheme, the encoding of abstraction reception simply expects a trigger name:

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

The mechanism for representing abstraction passing with name passing is completed in the encoding of name application. There are two cases:

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

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

Thus, in both cases we first establish a fresh session s with the server representing the abstraction body; the name to be applied (u) is then passed around using s. Observe how this encoding naturally induces the name substitution expected from a name application. We may now define:

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Terms:

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

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

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

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

Types:

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

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

Elided mappings are homomorphic.

Fig. 11. Encoding of HOπ into π (Definition 5.4).

Definition 5.4 *(Typed encoding of* HOπ *into* π*).* The typed encoding $\big\langle [\![\cdot]\!]^2, (\!|\cdot|\!)^2 \big\rangle : \mathcal{L}_{\text{HO}\pi} \to \mathcal{L}_\pi$ is defined in Fig. 11.

Example 5.3 *(Encoding* Client_1 *and* Client_2*).* The Hotel Booking scenario is described in § 3.4 (and Fig. 7) as the HOπ processes Client_1 and Client_2. We first encode Client_1 in π is as follows:

$$\begin{aligned}
[\![\mathsf{Client}_1]\!]^2 &= [\![(\nu\, h_1, h_2)(s_1!\langle\lambda x.\, P_{xy}\{^{h_1}/_{y}\}\rangle.s_2!\langle\lambda x.\, P_{xy}\{^{h_2}/_{y}\}\rangle.\mathbf{0} \mid \\
&\qquad \overline{h_1}?(x).\overline{h_2}?(y).\mathtt{if}\ x \leq y\ \mathtt{then} \\
&\qquad\qquad (\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0}\ \mathtt{else}\ \overline{h_1} \triangleleft \mathsf{reject}.\overline{h_2} \triangleleft \mathsf{accept}.\mathbf{0}))]\!]^2 \\
&= (\nu\, h_1, h_2)([\![s_1!\langle\lambda x.\, P_{xy}\{^{h_1}/_{y}\}\rangle.s_2!\langle\lambda x.\, P_{xy}\{^{h_2}/_{y}\}\rangle.\mathbf{0}]\!]^2 \mid \\
&\qquad \overline{h_1}?(x).\overline{h_2}?(y).\mathtt{if}\ x \leq y\ \mathtt{then} \\
&\qquad\qquad (\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0}\ \mathtt{else}\ \overline{h_1} \triangleleft \mathsf{reject}.\overline{h_2} \triangleleft \mathsf{accept}.\mathbf{0})) \\
&= (\nu\, h_1, h_2)\big((\nu\, a_1)(s_1!\langle a_1\rangle.([\![s_2!\langle\lambda x.\, P_{xy}\{^{h_2}/_{y}\}\rangle.\mathbf{0}]\!]^2 \mid a_1?(y).y?(x).[\![P_{xy}\{^{h_1}/_{y}\}]\!]^2)) \mid \\
&\qquad \overline{h_1}?(x).\overline{h_2}?(y).\mathtt{if}\ x \leq y\ \mathtt{then} \\
&\qquad\qquad (\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0}\ \mathtt{else}\ \overline{h_1} \triangleleft \mathsf{reject}.\overline{h_2} \triangleleft \mathsf{accept}.\mathbf{0})\big) \\
&= (\nu\, h_1, h_2)\big((\nu\, a_1)(s_1!\langle a_1\rangle.(\nu\, a_2)(s_2!\langle a_2\rangle. \\
&\qquad (\mathbf{0} \mid a_2?(y).y?(x).[\![P_{xy}\{^{h_2}/_{y}\}]\!]^2)) \mid a_1?(y).y?(x).[\![P_{xy}\{^{h_1}/_{y}\}]\!]^2) \mid \\
&\qquad \overline{h_1}?(x).\overline{h_2}?(y).\mathtt{if}\ x \leq y\ \mathtt{then} \\
&\qquad\qquad (\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0}\ \mathtt{else}\ \overline{h_1} \triangleleft \mathsf{reject}.\overline{h_2} \triangleleft \mathsf{accept}.\mathbf{0})\big)
\end{aligned}$$

where $[\![P_{xy}]\!]^2 = P_{xy}$, for it does not involve higher-order communication. Similarly, the encoding of Client_2 is as follows:

$$\begin{aligned}
[\![\mathsf{Client}_2]\!]^2 &= [\![(\nu\, h)(s_1!\langle\lambda x.\, Q_1\{^{h}/_{y}\}\rangle.s_2!\langle\lambda x.\, Q_2\{^{\overline{h}}/_{y}\}\rangle.\mathbf{0})]\!]^2 \\
&= (\nu\, h)\big((\nu\, a_1)(s_1!\langle a_1\rangle.([\![s_2!\langle\lambda x.\, Q_2\{^{\overline{h}}/_{y}\}\rangle.\mathbf{0}]\!]^2 \mid a_1?(y).y?(x).[\![Q_1\{^{h}/_{y}\}]\!]^2))\big) \\
&= (\nu\, h)\big((\nu\, a_1)(s_1!\langle a_1\rangle.(\nu\, a_2)(s_2!\langle a_2\rangle.\mathbf{0} \mid \\
&\qquad a_2?(y).y?(x).[\![Q_2\{^{\overline{h}}/_{y}\}]\!]^2) \mid a_1?(y).y?(x).[\![Q_1\{^{h}/_{y}\}]\!]^2)\big)
\end{aligned}$$

where $[\![Q_1]\!]^2 = Q_1$ and $[\![Q_2]\!]^2 = Q_2$ for they do not involve higher-order communication.

We state the properties of this encoding. First, type preservation, type soundness, and operational correspondence, which requires a mapping on action labels.

Proposition 5.4 *(*HOπ *into* π*: type preservation and type soundness). The encoding from* $\mathcal{L}_{\text{HO}\pi}$ *into* $\mathcal{L}_\pi$ *(cf. Definition 5.4) is type preserving (cf. Definition 4.4) and type sound (cf. Definition 4.5(1)).*

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Proof. Type preservation follows directly from Fig. 11. Type soundness is proven by induction on the inference $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. See Proposition Appendix B.4 (Page 42) in B.2. □

Definition 5.5. Given the typed encoding $\langle [\![\cdot]\!]^2, (\!|\cdot|\!)^2 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_\pi$ (cf. Definition 5.4), the mapping on actions $\{\!|\cdot|\!\}^2 : \mathcal{A}_{\mathsf{HO}\pi} \to \mathcal{A}_\pi$ is defined as follows:

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

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

and as an homomorphism for other actions $\ell \in \mathcal{A}_{\mathsf{HO}\pi}$.

We now state operational correspondence:

Proposition 5.5 *(Operational correspondence, $\mathsf{HO}\pi$ into π). Let P be an $\mathsf{HO}\pi$ process such that $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$.*

1. *Suppose $\Gamma; \Delta \vdash P \stackrel{\ell_1}{\longmapsto} \Delta' \vdash P'$. Then we have:*

   a) *If $\ell_1 = (\nu\,\tilde{m})n!\langle\lambda x.\, Q\rangle$, then $\exists \Gamma', \Delta''$ where either:*

   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\{\ell_1\}^2} \Gamma' \cdot (\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \mid *a?(y).y?(x).[\![Q]\!]^2$ *(if* $\mathtt{fs}(Q) = \emptyset$*)*
   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\{\ell_1\}^2} (\!|\Gamma|\!)^2; \Delta'' \vdash [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$ *(otherwise)*

   b) *If $\ell_1 = n?\langle\lambda y.\, Q\rangle$ then $\exists R$ where either*

   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\{\ell_1\}^2} \Gamma'; (\!|\Delta''|\!)^2 \vdash R$, *for some* Γ' *and* $(\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta''|\!)^2 \vdash (\nu\, a)(R \mid *a?(y).y?(x).[\![Q]\!]^2)$ *(if* $\mathtt{fs}(Q) = \emptyset$*)*
   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\{\ell_1\}^2} (\!|\Gamma|\!)^2; (\!|\Delta''|\!)^2 \vdash R$, *and* $(\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta''|\!)^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$ *(otherwise)*

   c) *If $\ell_1 = \tau$, with $\tau \neq \tau_\beta$ then one of the following holds:*

   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\tau} (\!|\Delta'|\!)^2 \vdash (\nu\,\tilde{m})([\![P_1]\!]^2 \mid (\nu\, a)([\![P_2]\!]^2\{^a/_x\} \mid *a?(y).y?(x).[\![Q]\!]^2))$, *for some* P_1, P_2, Q *(with* $\mathtt{fs}(Q) = \emptyset$*);*
   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\tau} (\!|\Delta'|\!)^2 \vdash (\nu\,\tilde{m})([\![P_1]\!]^2 \mid (\nu\, s)([\![P_2]\!]^2\{^{\overline{s}}/_x\} \mid s?(y).y?(x).[\![Q]\!]^2))$, *for some* P_1, P_2, Q *(with* $\mathtt{fs}(Q) \neq \emptyset$*);*
   - $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\tau} (\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$

   d) *If $\ell_1 = \tau_\beta$ then $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\tau_{\mathsf{s}}} (\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$*

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

2. *Suppose $(\!|\Gamma|\!)^2; (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \xmapsto{\ell_2} (\!|\Delta'|\!)^2 \vdash R$.*

   a) *If $\ell_2 = (\nu\, m)n!\langle m\rangle$ then one of the following holds:*

   - *$\exists P'$ such that $P \xmapsto{(\nu\, m)n!\langle m\rangle} P'$ and $R = [\![P']\!]^2$;*
   - *$\exists Q, P'$ such that $P \xmapsto{n!\langle\lambda x.\, Q\rangle} P'$ and $R = [\![P']\!]^2 \mid *a?(y).y?(x).[\![Q]\!]^2$ and $\mathtt{fs}(Q) = \emptyset$;*
   - *$\exists Q, P'$ such that $P \xmapsto{n!\langle\lambda x.\, Q\rangle} P'$ and $R = [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$ and $\mathtt{fs}(Q) \neq \emptyset$;*

   b) *If $\ell_2 = n?\langle m\rangle$ then one of the following holds:*

   - *$\exists P'$ such that $P \xmapsto{n?\langle m\rangle} P'$ and $R = [\![P']\!]^2$;*
   - *$\exists Q, P'$ such that $P \xmapsto{n?\langle\lambda x.\, Q\rangle} P'$*
     and $(\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta'|\!)^2 \vdash (\nu\, a)(R \mid *a?(y).y?(x).[\![Q]\!]^2)$ and $\mathtt{fs}(Q) = \emptyset$;

- $\exists Q, P'$ *such that* $P \xmapsto{n?\langle \lambda x.\, Q\rangle} P'$
  and $(\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta'|\!)^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$ *and* $\mathtt{fs}(Q) \neq \emptyset$.

c) If $\ell_2 = \tau$ *then* $\exists P'$ *such that* $P \xmapsto{\tau} P'$ *and* $(\!|\Gamma|\!)^2; (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta'|\!)^2 \vdash R$.
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*
$\Gamma;\ \Delta \vdash P \xmapsto{\ell_1} \Gamma;\ \Delta \vdash P'$.

Proof. By transition induction. See Proposition Appendix B.5 (Page 44) in B.2. □

Some comments on the completeness properties given by Proposition 5.5 (Page 20) are in order. Items 1(a), 1(b), and 1(e) describe the way in which the encoding mimicks source visible transitions (output, input, and labelled choice/selection, respectively). As discussed above, the encoding of output sets up a potentially persistent server to represent the body of the abstraction being exchanged. The statement in 1(a) formalises the fact that after an output transition in the source process this server has not been yet invoked/used on the target side, and so it appears as a residual context $(*a?(y).y?(x).[\![Q]\!]^2$ or $s?(y).y?(x).[\![Q]\!]^2)$ in parallel to the encoding of the continuation of the output ($[\![P']\!]^2$). Similarly, the statement in 1(b) formalises the fact that after an input transition the resulting process R should be placed in an appropriate context containing the server representing the abstraction body. Together, R and its server are behaviourally equivalent to $[\![P']\!]^2$. Items 1(c) and 1(d) state correspondences for internal actions, in the sense of Definition 4.5. In particular, the first two sub-items in 1(c) describe how a source reduction due to abstraction passing is matched: in our encoding this is mimicked by exchanging the trigger names; the third sub-item covers other possibilities for source reductions.

Exploiting the above properties (type preservation, typed operational correspondence), we can show that our typed encoding is fully abstract and precise.

Proposition 5.6 *(*$\mathsf{HO}\pi$ *to* π*: full abstraction). Let* P_1, Q_1 *be* $\mathsf{HO}\pi$ *processes.* $\Gamma; \Delta_1 \vdash P_1 \approx^{\mathrm{H}} \Delta_2 \vdash Q_1$ *if and only if* $(\!|\Gamma|\!)^2; (\!|\Delta_1|\!)^2 \vdash [\![P_1]\!]^2 \approx^{\mathrm{C}} (\!|\Delta_2|\!)^2 \vdash [\![Q_1]\!]^2$.

Proof. The proof of both directions proceeds coinductively. See Proposition Appendix B.6 (Page 46) in B.2. □

We may now finally state:

Theorem 5.2 *(Precise encoding of* $\mathsf{HO}\pi$ *into* π*). The encoding from* $\mathcal{L}_{\mathsf{HO}\pi}$ *into* $\mathcal{L}_\pi$ *(cf. Definition 5.4) is precise.*

Proof. According to Definition 4.6, preciseness includes syntax-, type-, and semantics-preservation. Syntax preservation follows immediately from the definition of the encoding. Type preservation follows from Proposition 5.4 (Page 19). Semantics-preservation follows from Proposition 5.5 (Page 20) and Proposition 5.6 (Page 21). □

5.3. Comparing two precise encodings

The precise encodings in §5.1 and §5.2 confirm that HO and π constitute two important sources of expressiveness in $\mathsf{HO}\pi$. This naturally begs the question: which of the two sub-calculi is more tightly related to $\mathsf{HO}\pi$? We argue, both empirically and formally, 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 $\mathsf{HO}\pi$ to HO, denoted $[\![\cdot]\!]^1_f$ (§5.1), translates processes with name passing;
b) the encoding from $\mathsf{HO}\pi$ to π, denoted $[\![\cdot]\!]^2$ (§5.2), translates processes with abstraction passing.

Consider the $\mathsf{HO}\pi$ processes:

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

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

$$P_1 \xmapsto{\tau} a!\langle s_1\rangle.\mathbf{0} \mid \ \ldots \ \mid a!\langle s_n\rangle.\mathbf{0}$$
$$P_2 \xmapsto{\tau} (\lambda x.\, R)\, s_1 \mid \ \ldots \ \mid (\lambda x.\, R)\, s_n \quad \underbrace{\xmapsto{\tau_\beta} \xmapsto{\tau_\beta} \cdots \xmapsto{\tau_\beta}}_{n} \quad R\{s_1/x\} \mid \ \ldots \ \mid R\{s_n/x\}$$

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

We now encode P_1 into HO and P_2 into π and contrast the results. First, by considering the encoding of P_1 into HO (following mapping $[\![\cdot]\!]^1_f$ in Fig. 9) we obtain:

$$\begin{aligned}
[\![P_1]\!]^1_f \quad &= \quad s!\langle \lambda z.\, z?(x).x\, a\rangle.\mathbf{0} \mid \\
&\qquad \overline{s}?(y).(\nu\, s_0)\big(y\, s_0 \mid \overline{s_0}!\langle \lambda x.\, (x!\langle V_1\rangle.\mathbf{0} \mid \ldots \mid x!\langle V_n\rangle.\mathbf{0})\rangle.\mathbf{0}\big) \\
&\xrightarrow{\tau_s}\xrightarrow{\tau_\beta} (\nu\, s_0)(s_0?(x).x\, a \mid \overline{s_0}!\langle \lambda x.\, (x!\langle V_1\rangle.\mathbf{0} \mid \ldots \mid x!\langle V_n\rangle.\mathbf{0})\rangle.\mathbf{0}) \\
&\xrightarrow{\tau_s}\xrightarrow{\tau_\beta} a!\langle V_1\rangle.\mathbf{0} \mid \ldots \mid a!\langle V_n\rangle.\mathbf{0}
\end{aligned}$$

where we write V_i to stand for $\lambda z.\, z?(x_i).x_i\, s_i$. Now, we encode P_2 into π (following mapping $[\![\cdot]\!]^2$ in Fig. 11):

$$\begin{aligned}
[\![P_2]\!]^2 \quad &= \quad (\nu\, a)(s!\langle a\rangle.(\mathbf{0} \mid *a?(y).y?(x).[\![R]\!]^2)) \mid \\
&\qquad \overline{s}?(x).\big((\nu\, s_0)(x!\langle s_0\rangle.\overline{s_0}!\langle s_1\rangle.\mathbf{0}) \mid \ldots \mid (\nu\, s_0)(x!\langle s_0\rangle.\overline{s_0}!\langle s_n\rangle.\mathbf{0})\big) \\
&\xrightarrow{\tau_s} \equiv \quad (\nu\, a)\big(*a?(y).y?(x).[\![R]\!]^2 \mid \\
&\qquad\qquad (\nu\, s_0)(a!\langle s_0\rangle.\overline{s_0}!\langle s_1\rangle.\mathbf{0}) \mid \ldots \mid (\nu\, s_0)(a!\langle s_0\rangle.\overline{s_0}!\langle s_n\rangle.\mathbf{0})\big) \\
&\xrightarrow{\tau_s}\xrightarrow{\tau_s} \quad (\nu\, a)\big(*a?(y).y?(x).[\![R]\!]^2 \mid [\![R]\!]^2\{s_1/x\} \mid \\
&\qquad\qquad (\nu\, s_0)(a!\langle s_0\rangle.\overline{s_0}!\langle s_2\rangle.\mathbf{0}) \mid \ldots \mid (\nu\, s_0)(a!\langle s_0\rangle.\overline{s_0}!\langle s_n\rangle.\mathbf{0})\big) \\
&\Mapsto_{2*(n-1)} (\nu\, a)\big(*a?(y).y?(x).[\![R]\!]^2 \mid [\![R]\!]^2\{s_1/x\} \mid [\![R]\!]^2\{s_2/x\} \mid \ldots \mid [\![R]\!]^2\{s_n/x\}\big)
\end{aligned}$$

Clearly, 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 (stuck) processes (such as $*a?(y).y?(x).[\![R]\!]^2$ above). 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 and Q_2:

$$\begin{aligned}
Q_1 &= s'!\langle s\rangle.\mathbf{0} \mid \overline{s'}?(x).x!\langle a\rangle.\mathbf{0} \\
&\xrightarrow{\tau} s!\langle a\rangle.\mathbf{0} \\
Q_2 &= s!\langle \lambda x.\, R\rangle.\mathbf{0} \mid \overline{s}?(x).x\, a \\
&\xrightarrow{\tau}\xrightarrow{\tau} R\{a/x\}
\end{aligned}$$

Q_1 is a π process and Q_2 is an HO process. If we consider the encoding of Q_1 into HO and of Q_2 into π, respectively, we obtain:

$$\begin{aligned}
[\![Q_1]\!]^1_f \quad &= \quad s'!\langle \lambda z.\, z?(x).x\, s\rangle.\mathbf{0} \mid \overline{s'}?(y).(\nu\, s_0)(y\, s_0 \mid \overline{s_0}!\langle \lambda x.\, x!\langle \lambda z.\, z?(y).y\, a\rangle.\mathbf{0}\rangle.\mathbf{0}) \\
&\xrightarrow{\tau_s}\xrightarrow{\tau_\beta} (\nu\, s_0)(s_0?(x).x\, s \mid \overline{s_0}!\langle \lambda x.\, x!\langle \lambda z.\, z?(y).y\, a\rangle.\mathbf{0}\rangle.\mathbf{0}) \\
&\xrightarrow{\tau_s} \quad (\lambda x.\, x!\langle \lambda z.\, z?(y).y\, a\rangle.\mathbf{0})\, s \\
&\xrightarrow{\tau_\beta} \quad s!\langle \lambda z.\, z?(y).y\, a\rangle.\mathbf{0} \\
[\![Q_2]\!]^2 \quad &= \quad (\nu\, a_1)(s!\langle a_1\rangle.(\mathbf{0} \mid \overline{a_1}?(y).y?(x).[\![R]\!]^2)) \mid \overline{s}?(x).(\nu\, s_0)(x!\langle s_0\rangle.\overline{s_0}!\langle a\rangle.\mathbf{0}) \\
&\xrightarrow{\tau_s}\xrightarrow{\tau_s} (\nu\, s_0)(s_0?(x).[\![R]\!]^2 \mid \overline{s_0}!\langle a\rangle.\mathbf{0}) \\
&\xrightarrow{\tau_s} \quad [\![R]\!]^2\{a/x\}
\end{aligned}$$

In this case, the encoding $[\![\cdot]\!]^2$ is more efficient because it induces less reduction steps. Therefore, considering a fragment of HOπ without shared communications (linearity only) has consequences in terms of reduction steps. These apparent benefits of encoding $[\![\cdot]\!]^2$ over encoding $[\![\cdot]\!]^1_f$ in the presence of linearity should, however, be considered in a broader setting, for in §5.4 we prove that linear resources do not suffice to encode shared communications. Therefore, in the general case featuring linear and shared communication not only the benefits of $[\![\cdot]\!]^2$ over $[\![\cdot]\!]^1_f$ could not be obtained, but the drawbacks mentioned in the comparison between $[\![P_1]\!]^1_f$ and $[\![P_2]\!]^2$ (i.e., the garbage processes generated by $[\![\cdot]\!]^2$) could well be more prominent. This observation may be used to informally argue that $[\![\cdot]\!]^1_f$ is "better than" $[\![\cdot]\!]^2$ (or, alternatively, that HOπ is closer to HO than to π); next, we develop a formal argument to substantiate this claim.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Formal comparison: labelled transition correspondence To formally state that HOπ and HO are more closely related than HOπ and π, we may distinguish the precise encodings $[\![\cdot]\!]^1_f$ and $[\![\cdot]\!]^2$ depending on whether they are also tight encodings or not (cf. Definition 4.7):

Theorem 5.3 *(HO tightly encodes HOπ). While the encoding of HOπ into HO (Definition 5.2) is tight, the encoding of HOπ into π (Definition 5.4) is not tight.*

Proof (Sketch). The proof proceeds by showing that the encoding $[\![\cdot]\!]^1_f$ enjoys labelled operational correspondence, whereas $[\![\cdot]\!]^2$ does not. Recall that a labelled operational correspondence for $[\![\cdot]\!]^1_f$ has been already stated in Proposition 5.2 (Page 17). The analog of Proposition 5.2 (Page 17) does not hold for the encoding $[\![\cdot]\!]^2$ of HOπ into π. Consider the HOπ process:

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

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

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

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 perform actions of the encoding of $\lambda x.\, P$. □

5.4. A negative result

As most session calculi, HOπ includes communication on both shared and linear names. Shared names enable non deterministic, unrestricted behaviour; linear names represent deterministic communication structures. The expressiveness of shared names is also illustrated by our encoding from HOπ into π (Fig. 11). This result begs the question: can we represent interaction along shared names using linear names only? It turns out that shared names strictly add expressiveness to HOπ: next we prove the non existence of a minimal encoding of interaction along shared names using linear names.

Theorem 5.4. *There is no minimal encoding from π to HO$\pi^{-\mathsf{sh}}$.*

Proof. Assume, towards a contradiction, that such a typed minimal encoding indeed exists. Recall that a minimal encoding is syntax preserving, barb preserving, and operationally complete (cf. Definition 4.6). Consider the π process

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

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

$$\Gamma;\Delta \vdash P \xmapsto{\tau} \Delta' \vdash n \triangleleft l_1.\mathbf{0} \mid a(x).m \triangleleft l_2.\mathbf{0} = P_1 \tag{3}$$

$$\Gamma;\Delta \vdash P \xmapsto{\tau} \Delta' \vdash m \triangleleft l_2.\mathbf{0} \mid a(x).n \triangleleft l_1.\mathbf{0} = P_2 \tag{4}$$

Thus, by definition of typed barb (cf. Definition 3.7) we have:

$$\Gamma;\Delta' \vdash P_1 \downarrow_n \wedge \Gamma;\Delta' \vdash P_1 \not\downarrow_m \tag{5}$$

$$\Gamma;\Delta' \vdash P_2 \not\downarrow_n \wedge \Gamma;\Delta' \vdash P_2 \downarrow_m \tag{6}$$

Consider now the HO$\pi^{-\mathsf{sh}}$ process $[\![P]\!]$. By our assumption of operational completeness (Definition 4.5-3(a)), from (3) with (4) we infer that there exist HO$\pi^{-\mathsf{sh}}$ processes S_1 and S_2 such that:

$$(\!|\Gamma|\!);(\!|\Delta|\!) \vdash [\![P]\!] \xRightarrow{\tau_s} (\!|\Delta'|\!) \vdash S_1 \approx^{\mathrm{H}} [\![P_1]\!] \tag{7}$$

$$(\!|\Gamma|\!);(\!|\Delta|\!) \vdash [\![P]\!] \xRightarrow{\tau_s} (\!|\Delta'|\!) \vdash S_2 \approx^{\mathrm{H}} [\![P_2]\!] \tag{8}$$

By our assumption of barb preservation, from (5) with (6) we infer:

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_1]\!] \Downarrow_n \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_1]\!] \not\Downarrow_m \tag{9}$$

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_2]\!] \not\Downarrow_n \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash [\![P_2]\!] \Downarrow_m \tag{10}$$

By definition of $\approx$, by combining (7) with (9) and (8) with (10), we infer barbs for S_1 and S_2:

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_1 \Downarrow_n \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_1 \not\Downarrow_m \tag{11}$$

$$(\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_2 \Downarrow_m \wedge (\!|\Gamma|\!);(\!|\Delta'|\!) \vdash S_2 \not\Downarrow_n \tag{12}$$

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

That is, S_1 and $[\![P_1]\!]$ (resp. S_2 and $[\![P_2]\!]$) have the same barbs. Now, by τ-inertness (Lemma 3.1), we have both

$$(\!|\Gamma|\!); (\!|\Delta'|\!) \vdash S_1 \approx^{\mathrm{H}} (\!|\Delta|\!) \vdash [\![P]\!] \tag{13}$$

$$(\!|\Gamma|\!); (\!|\Delta'|\!) \vdash S_2 \approx^{\mathrm{H}} (\!|\Delta|\!) \vdash [\![P]\!] \tag{14}$$

Combining (13) with (14), by transitivity of $\approx^{\mathrm{H}}$, we infer

$$(\!|\Gamma|\!); (\!|\Delta'|\!) \vdash S_1 \approx^{\mathrm{H}} (\!|\Delta'|\!) \vdash S_2 \tag{15}$$

In turn, from (15) we infer that it must be the case that:

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

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

which clearly contradict (9) and (10) above. We therefore conclude that a minimal encoding from π to $\mathsf{HO}\pi^{-\mathsf{sh}}$ does not exist. □

We then have:

Corollary 5.1. *Let* $\mathsf{C}_1, \mathsf{C}_2 \in \{\mathsf{HO}\pi, \mathsf{HO}, \pi\}$.

(a) There is no minimal encoding from $\mathcal{L}_{\mathsf{C}_1}$ *into* $\mathcal{L}_{\mathsf{C}_2^{-\mathsf{sh}}}$.
(b) There is a precise encoding of $\mathcal{L}_{\mathsf{C}_1^{-\mathsf{sh}}}$ *in* $\mathcal{L}_{\mathsf{C}_2^{-\mathsf{sh}}}$.

Proof. Part (a) is immediate from Theorem 5.4. Part (b) follows from the definitions of the typed encodings of $\mathsf{HO}\pi$ into HO (cf. Definition 5.2) and of HO into π (cf. Definition 5.4), which work uniformly for linear and shared names, as well as from the preciseness results for such encodings (cf. Proposition 5.1 (Page 18) and Proposition 5.2 (Page 21)). □

6. Extensions: $\mathsf{HO}\pi$ with higher-order abstractions and with polyadicity

We now extend $\mathsf{HO}\pi$ in two orthogonal ways: $\mathsf{HO}\pi^+$ extends $\mathsf{HO}\pi$ with higher-order applications/abstractions, while $\mathsf{HO}\widetilde{\pi}$ extends $\mathsf{HO}\pi$ with polyadicity. In both cases, we detail the required modifications in syntax and types. By combining $\mathsf{HO}\pi^+$ and $\mathsf{HO}\widetilde{\pi}$ into a single calculus we obtain $\mathsf{HO}\widetilde{\pi}^+$: the extension of $\mathsf{HO}\pi$ with *both* higher-order abstractions/applications and polyadicity (cf. Corollary 6.1 and Corollary 6.2)

We present precise encodings of $\mathsf{HO}\pi^+$ and $\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$. We then use the encodings of $\mathsf{HO}\pi$ into HO and π in the previous section, together with encoding composability (Proposition 4.1 (Page 13)), to relate HO and π with the super-calculus $\mathsf{HO}\widetilde{\pi}^+$, which subsumes both $\mathsf{HO}\pi^+$ and $\mathsf{HO}\widetilde{\pi}$.

6.1. Precise encoding of $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$

We first introduce $\mathsf{HO}\pi^+$, the extension of $\mathsf{HO}\pi$ with higher-order abstractions and applications. This is the calculus whose (typed) behavioural theory we studied in [15,17]. The syntax of $\mathsf{HO}\pi^+$ is obtained from Fig. 2 by replacing $V\,u$ with $V\,W$ in the syntax of processes, where W is a higher-order value. As for the reduction semantics, we keep the rules in Fig. 3, except for Rule [App], which is replaced by

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

Example 6.1. The following is a simple $\mathsf{HO}\pi^+$ process with its corresponding reductions:

$$\begin{aligned} s!\langle \lambda x.\, Q\rangle.\mathbf{0} \mid \overline{s}?(y).(\lambda z.\,(z\, s_1)\, y) &\longrightarrow (\lambda z.\,(z\, s_1))\,(\lambda x.\, Q) \\ &\longrightarrow (\lambda x.\, Q)\, s_1 \\ &\longrightarrow Q\,\{^{s_1}/_{x}\} \end{aligned}$$

Above, the additional expressivity of $\mathsf{HO}\pi^+$ with respect to $\mathsf{HO}\pi$ is in the ability of applying a function such as $\lambda z.\,(z\, s_1)$ to an argument such as $\lambda x.\, Q$, which is not a name but another function.

The syntax of types in Fig. 4 is modified as follows:

$$L \;::=\; U \to \diamond \;\mid\; U \multimap \diamond.$$

Values and Terms:

$$\{\!|x|\!\}^3 \stackrel{\text{def}}{=} x$$

$$\{\!|\lambda x : L.\, P|\!\}^3 \stackrel{\text{def}}{=} \lambda z.\, z?(x).[\![P]\!]^3$$

$$[\![u!\langle\lambda x : L.\, Q\rangle.P]\!]^3 \stackrel{\text{def}}{=} u!\langle\{\!|\lambda x.\, Q|\!\}^3\rangle.[\![P]\!]^3$$

$$[\![u?(x).P]\!]^3 \stackrel{\text{def}}{=} u?(x).[\![P]\!]^3$$

$$[\![(x : L)\, V]\!]^3 \stackrel{\text{def}}{=} (\nu\, s)(x\, s \mid \overline{s}!\langle\{\!|V|\!\}^3\rangle.\mathbf{0})$$

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

Types:

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

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

Mappings for elided processes and types are homomorphic.

Fig. 12. Encoding of HOπ^+ into HOπ.

These types can be easily accommodated in the type system in §3.2: in Fig. 5, we replace C by U in Rule (Abs) and C by U' in Rule (App). With these extensions, subject reduction (Theorem 3.1) holds for HOπ^+ (cf. [15])

We give an encoding of HOπ^+ into HOπ and show that it is precise. We may then use encoding composition (Proposition 4.1 (Page 13)) to encode HOπ^+ into HO and π. We consider the following typed calculus (cf. Definition 4.1):

$$\mathcal{L}_{\mathsf{HO}\pi^+} = \langle \mathsf{HO}\pi^+, \mathcal{T}_4, \stackrel{\ell}{\mapsto}, \approx^{\mathrm{H}}, \vdash\rangle$$

where $\mathcal{T}_4$ is a set of types of HOπ^+; the typing $\vdash$ is defined in §3.2 with Rules (Abs) and (App) modified as explained above. Formally, the set $\mathcal{A}_{\mathsf{HO}\pi^+}$ coincides with $\mathcal{A}_{\mathsf{HO}\pi}$, for the syntax of values V is the same in both languages. However, by considering the refined actions given by type-annotated values (cf. Remark 5.1), we have that $\mathcal{A}_{\mathsf{HO}\pi^+}$ includes output and input actions of the form $(\nu\,\tilde{m})n!\langle\lambda x{:}L.\, P\rangle$ and $n?\langle\lambda x{:}L.\, P\rangle$, whereas $\mathcal{A}_{\mathsf{HO}\pi}$ includes only labels of the form $(\nu\,\tilde{m})n!\langle\lambda x{:}C.\, P\rangle$ and $n?\langle\lambda x{:}C.\, P\rangle$.

Definition 6.1 *(Typed encoding of* HOπ^+ *into* HOπ*).* The typed encoding $\left\langle [\![\cdot]\!]^3, (\!|\cdot|\!)^3\right\rangle : \mathcal{L}_{\mathsf{HO}\pi^+} \to \mathcal{L}_{\mathsf{HO}\pi}$ is defined in Fig. 12.

We consider mappings for terms and types, denoted $[\![\cdot]\!]^3$ and $(\!|\cdot|\!)^3$, respectively. Since now functions can be applied to (higher-order) values, we have also an auxiliary mapping on values, denoted $\{\!|\cdot|\!\}^3$. We illustrate the essence of these mappings by means of an example.

Example 6.2. We translate the simple process from Ex. 6.1, underlining the parts of the translation which are expanded/modified from one line to the following:

$$\begin{aligned}
&[\![s!\langle\lambda x.\, Q\rangle.\mathbf{0} \mid \overline{s}?(y).(\lambda z.\,(z\, s_1)\, y)]\!]^3 \\
=\ & s!\langle\underline{\{\!|\lambda x.\, Q|\!\}^3}\rangle.\underline{[\![\mathbf{0}]\!]^3} \mid \overline{s}?(y).\underline{[\![(\lambda z.\,(z\, s_1)\, y)]\!]^3} \\
=\ & s!\langle\lambda w.\, w?(x).[\![Q]\!]^3\rangle.\mathbf{0} \mid \overline{s}?(y).(\nu\, s_0)(s_0?(z).\underline{[\![(z\, s_1)]\!]^3} \mid \overline{s_0}!\langle\underline{\{\!|y|\!\}^3}\rangle.\mathbf{0}) \\
=\ & s!\langle\lambda w.\, w?(x).[\![Q]\!]^3\rangle.\mathbf{0} \mid \overline{s}?(y).(\nu\, s_0)(s_0?(z).(\nu\, s_2)(z\, s_2 \mid \overline{s_2}!\langle\underline{\{\!|s_1|\!\}^3}\rangle.\mathbf{0}) \mid \overline{s_0}!\langle y\rangle.\mathbf{0}) \\
=\ & s!\langle\lambda w.\, w?(x).[\![Q]\!]^3\rangle.\mathbf{0} \mid \overline{s}?(y).(\nu\, s_0)(s_0?(z).(\nu\, s_2)(z\, s_2 \mid \overline{s_2}!\langle s_1\rangle.\mathbf{0}) \mid \overline{s_0}!\langle y\rangle.\mathbf{0}) \\
\longrightarrow\ & \mathbf{0} \mid (\nu\, s_0)(s_0?(z).(\nu\, s_2)(z\, s_2 \mid \overline{s_2}!\langle s_1\rangle.\mathbf{0}) \mid \overline{s_0}!\langle\lambda w.\, w?(x).[\![Q]\!]^3\rangle.\mathbf{0}) \\
\longrightarrow\ & \mathbf{0} \mid (\nu\, s_2)((\lambda w.\, w?(x).[\![Q]\!]^3)\, s_2 \mid \overline{s_2}!\langle s_1\rangle.\mathbf{0}) \mid \mathbf{0} \\
\longrightarrow\ & \mathbf{0} \mid (\nu\, s_2)(s_2?(x).[\![Q]\!]^3 \mid \overline{s_2}!\langle s_1\rangle.\mathbf{0}) \mid \mathbf{0} \\
\longrightarrow\equiv\ & [\![Q]\!]^3\{^{s_1}/_x\}
\end{aligned}$$

This typed encoding satisfies the following properties:

Proposition 6.1 *(*HOπ^+ *into* HOπ*: type preservation and type soundness). The encoding from* $\mathcal{L}_{\mathsf{HO}\pi^+}$ *into* $\mathcal{L}_{\mathsf{HO}\pi}$ *(cf. Fig. 12) is type preserving (cf. Definition 4.4) and type sound (cf. Definition 4.5(1)).*

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Proof. Type preservation follows directly from Fig. 12. Type soundness is shown by induction on the inference of $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. See Proposition Appendix B.7 (Page 47) in B.3. □

Before proving operational correspondence we define a mapping on action labels:

Definition 6.2. Given the typed encoding $\langle [\![\cdot]\!]^3, (\!|\cdot|\!)^3 \rangle : \mathcal{L}_{\mathsf{HO}\pi^+} \to \mathcal{L}_{\mathsf{HO}\pi}$ (cf. Definition 6.1), the mapping on actions $\{\!|\cdot|\!\}^3 : \mathcal{A}_{\mathsf{HO}\pi^+} \to \mathcal{A}_{\mathsf{HO}\pi}$ is defined as follows:

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

$$\{\!|n?\langle\lambda x{:}L.\,P\rangle|\!\}^3 \stackrel{\text{def}}{=} n?\langle\lambda z{:}(\!|L|\!)^3.\,z?(x).[\![P]\!]^3\rangle$$

and as an homomorphism for all other actions $\ell \in \mathcal{A}_{\mathsf{HO}\pi^+}$.

We may now state a labelled form of operational correspondence, as well as full abstraction:

Proposition 6.2 *(Operational correspondence. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). Let $\Gamma; \emptyset; \Delta \vdash P$ be an $\mathsf{HO}\pi^+$ process.*

1. $\Gamma; \Delta \vdash P \overset{\ell}{\mapsto} \Delta' \vdash P'$ *implies*

   a) *If* $\ell \in \{(\nu\,\tilde{m})n!\langle\lambda x.\,Q\rangle, n?\langle\lambda x.\,Q\rangle\}$ *then* $(\!|\Gamma|\!)^3; (\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\ell'}{\mapsto} (\!|\Delta'|\!)^3 \vdash [\![P']\!]^3$ *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* $(\!|\Gamma|\!)^3; (\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\ell}{\mapsto} (\!|\Delta'|\!)^3 \vdash [\![P']\!]^3$.
   c) *If* $\ell = \tau_\beta$ *then* $(\!|\Gamma|\!)^3; (\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\tau}{\mapsto} \Delta'' \vdash R$ *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* $(\!|\Gamma|\!)^3; (\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\tau}{\mapsto} (\!|\Delta'|\!)^3 \vdash [\![P']\!]^3$.

2. $(\!|\Gamma|\!)^3; (\!|\Delta|\!)^3 \vdash [\![P]\!]^3 \overset{\ell}{\mapsto} (\!|\Delta''|\!)^3 \vdash Q$ *implies*

   a) *If* $\ell \in \{(\nu\,\tilde{m})n!\langle\lambda x.\,R\rangle, n?\langle\lambda x.\,R\rangle\}$ *then* $\Gamma; \Delta \vdash P \overset{\ell'}{\mapsto} \Delta' \vdash P'$ *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* $\Gamma; \Delta \vdash P \overset{\ell}{\mapsto} \Delta' \vdash P'$ *and* $Q \equiv [\![P']\!]^3$.
   c) *If* $\ell = \tau$ *then either* $\Gamma; \Delta \vdash P \overset{\tau}{\mapsto} \Delta' \vdash P'$ *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. By transition induction. See Proposition Appendix B.8 (Page 48) in B.3. □

The correspondence is rather tight: in both completeness and soundness directions, the most interesting cases are due to input and output actions (whose label explicitly mentions a value) and to τ_β internal actions in the source process. We may now have:

Proposition 6.3 *(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 $(\!|\Gamma|\!)^3; (\!|\Delta_1|\!)^3 \vdash [\![P]\!]^3 \approx^{\mathrm{H}} (\!|\Delta_2|\!)^3 \vdash [\![Q]\!]^3$.*

Proof. By coinduction. See Proposition Appendix B.9 (Page 49) in B.3. □

Using the above propositions, Theorems 5.1 and 5.2, and Proposition 4.1 (Page 13), we derive the following:

Theorem 6.1 *(Encoding $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$). The encoding from $\mathcal{L}_{\mathsf{HO}\pi^+}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ (cf. Fig. 12) is precise.*

Proof. According to Definition 4.6, preciseness includes syntax-, type-, and semantics-preservation. Syntax preservation follows immediately from the definition of the encoding. Type preservation follows from Proposition 6.1 (Page 25). Semantics-preservation follows from Proposition 6.2 (Page 26) and Proposition 6.3 (Page 26). □

We then have the following corollary:

Corollary 6.1 *(Encodability of $\mathsf{HO}\pi^+$ into $\mathsf{HO}\pi$ and π). Consider the typed encodings*

- $\langle [\![\cdot]\!]^1_f, (\!|\cdot|\!)^1 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\mathsf{HO}}$ *(cf. Definition 5.2)*

- $\langle [\![\cdot]\!]^2, (\!|\cdot|\!)^2 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\pi}$ *(cf. Definition 5.4)*
- $\langle [\![\cdot]\!]^3, (\!|\cdot|\!)^3 \rangle : \mathcal{L}_{\mathsf{HO}\pi^+} \to \mathcal{L}_{\mathsf{HO}\pi}$ *(cf. Definition 6.1)*

Then the following typed encodings are precise:

- $\langle [\![\cdot]\!]^1 \circ [\![\cdot]\!]^3, (\!|\cdot|\!)^1 \circ (\!|\cdot|\!)^3 \rangle : \mathcal{L}_{\mathsf{HO}\pi^+} \to \mathcal{L}_{\mathsf{HO}}$
- $\langle [\![\cdot]\!]^2 \circ [\![\cdot]\!]^3, (\!|\cdot|\!)^2 \circ (\!|\cdot|\!)^3 \rangle : \mathcal{L}_{\mathsf{HO}\pi^+} \to \mathcal{L}_{\pi}$

Proof. Directly from Theorem 5.1, Theorem 5.2, and Theorem 6.1 (which give preciseness for all the involved encodings), using Proposition 4.1 (Page 13). □

6.2. Precise encoding of $\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$

The calculus $\mathsf{HO}\widetilde{\pi}$ extends $\mathsf{HO}\pi$ with polyadicity so as to enable the exchange of tuples of names $\widetilde{n}$ (with fixed length $k \geq 1$) in both session communication and as arguments to function applications. Communication along shared names remains monadic. As such, the syntax of Fig. 2 is modified by considering polyadic first-order applications of the form $\lambda x_1, \ldots, x_k.\, Q$ $(k \geq 1)$ in the syntax of values V; the syntax of processes includes polyadicity in input and output prefixes, as well as in function applications. The operational semantics in Fig. 3 requires only minor modifications to accommodate the simultaneous substitution $\{\widetilde{V}/\widetilde{x}\}$ (for equally sized $\widetilde{u}/\widetilde{V}$ and $\widetilde{x}$) in Rules [App] and [Pass]:

$$(\lambda \widetilde{x}.\, P)\, \widetilde{u} \longrightarrow P\{\widetilde{u}/x\}$$

$$n!\langle \widetilde{V} \rangle.P \mid \overline{n}?(\widetilde{x}).Q \longrightarrow P \mid Q\{\widetilde{V}/\widetilde{x}\}$$

The type syntax in Fig. 4 is extended accordingly, as follows:

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

$$S \;::=\; !\langle \widetilde{U} \rangle; S \;\mid\; ?(\widetilde{U}); S \;\mid\; \cdots$$

As in [27,28], the type system for $\mathsf{HO}\widetilde{\pi}$ disallows polyadicity along shared names. We consider the following typed calculus (cf. Definition 4.1):

$$\mathcal{L}_{\mathsf{HO}\widetilde{\pi}} = \langle \mathsf{HO}\widetilde{\pi}, \mathcal{T}_5, \overset{\ell}{\mapsto}, \approx^{\mathrm{H}}, \vdash \rangle$$

where $\mathcal{T}_5$ is the set of types of $\mathsf{HO}\widetilde{\pi}$; the typing $\vdash$ is defined in § 3.2 with type syntax given above. Also, writing k to denote the arity of $\mathsf{HO}\widetilde{\pi}$, the set of labels $\mathcal{A}_{\mathsf{HO}\widetilde{\pi}}$ extends that in Definition 3.10 with actions $(\nu\, \widetilde{m})n!\langle m_1, \ldots, m_k \rangle$, $n!\langle \lambda x_1, \ldots, x_k.\, P \rangle$, $n?\langle m_1, \ldots, m_k \rangle$, and $n?\langle \lambda x_1, \ldots, x_k.\, P \rangle$.

We now define a typed encoding of $\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$. For simplicity, in definitions and statements we sometimes give the dyadic case (tuples of length 2); the general k-adic case is as expected.

Definition 6.3 *(Typed encoding of $\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$).* The typed encoding $\langle [\![\cdot]\!]^4, (\!|\cdot|\!)^4 \rangle : \mathcal{L}_{\mathsf{HO}\widetilde{\pi}} \to \mathcal{L}_{\mathsf{HO}\pi}$ in Fig. 13.

The encoding is unsurprising: a single polyadic communication of a tuple of length $k > 1$ is translated as k independent monadic communications, exploiting the already private communication medium given by the session name—unlike classical encodings [23], there is no need to create an additional fresh name for carrying out the monadic exchanges. Polyadic first-order abstraction and application appeal to an auxiliary fresh session along which parameters are communicated one by one.

The encoding satisfies the following properties:

Proposition 6.4 *($\mathsf{HO}\widetilde{\pi}$ into $\mathsf{HO}\pi$: type preservation and type soundness). The encoding from $\mathcal{L}_{\mathsf{HO}\widetilde{\pi}}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ (cf. Fig. 13) is type preserving (cf. Definition 4.4) and type sound (cf. Definition 4.5(1)).*

Proof. Type preservation follows directly from Fig. 13. Type soundness is shown by induction on the inference $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. See Proposition Appendix B.10 (Page 49) in B.4. □

In this case, the required mapping on actions maps an action on $\mathcal{A}_{\mathsf{HO}\widetilde{\pi}}$ into a *sequence* of actions in $\mathcal{A}_{\mathsf{HO}\pi}$. This is a natural consequence of dividing a polyadic name communication or application into several independent (monadic) communications:

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

Terms:

$$\llbracket u!\langle u_1,u_2\rangle.P\rrbracket^4 \stackrel{\text{def}}{=} u!\langle u_1\rangle.u!\langle u_2\rangle.\llbracket P\rrbracket^4$$
$$\llbracket u!\langle\lambda x_1,x_2.\,Q\rangle.P\rrbracket^4 \stackrel{\text{def}}{=} u!\langle\lambda z.\,z?(x_1).z?(x_2).\llbracket Q\rrbracket^4\rangle.\llbracket P\rrbracket^4$$
$$\llbracket x(u_1,u_2)\rrbracket^4 \stackrel{\text{def}}{=} (\nu\, s)(xs \mid \overline{s}!\langle u_1\rangle.\overline{s}!\langle u_2\rangle.\mathbf{0})$$
$$\llbracket(\lambda x_1,x_2.\,P)\,(u_1,u_2)\rrbracket^4 \stackrel{\text{def}}{=} (\nu\, s)(s?(x_1).s?(x_2).\llbracket P\rrbracket^4 \mid \overline{s}!\langle u_1\rangle.\overline{s}!\langle u_2\rangle.\mathbf{0})$$

Types:

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

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

Fig. 13. Encoding of $\mathsf{HO}\tilde{\pi}$ (dyadic case) into $\mathsf{HO}\pi$.

Definition 6.4. Given the typed encoding $\langle\llbracket\cdot\rrbracket^4,(\!|\cdot|\!)^4\rangle : \mathcal{L}_{\mathsf{HO}\tilde{\pi}} \to \mathcal{L}_{\mathsf{HO}\pi}$ (cf. Definition 6.3), the mapping on actions $\{\!|\cdot|\!\}^4 : \mathcal{A}_{\mathsf{HO}\tilde{\pi}} \to \mathcal{A}_{\mathsf{HO}\pi}{}^*$ is defined as follows:

$$\{\!|(\nu\,\tilde{m})n!\langle m_1,m_2\rangle|\!\}^4 \stackrel{\text{def}}{=} \ell_1,\ell_2 \quad \text{where} \begin{cases}\ell_i=(\nu\,m_i)n!\langle m_i\rangle & \text{if } m_i\in\tilde{m}\\ \ell_i=n!\langle m_i\rangle & \text{if } m_i\notin\tilde{m}\end{cases}$$
$$\{\!|(\nu\,\tilde{m})n!\langle\lambda x_1,x_2.\,P\rangle|\!\}^4 \stackrel{\text{def}}{=} (\nu\,\tilde{m})n!\langle\lambda z.\,z?(x_1).z?(x_2).\llbracket P\rrbracket^4\rangle$$
$$\{\!|\tau|\!\}^4 \stackrel{\text{def}}{=} \begin{cases}\tau_\beta,\tau_{\mathsf{s}},\tau_{\mathsf{s}} & \text{if } \tau=\tau_\beta\\ \tau,\tau & \text{otherwise}\end{cases}$$
$$\{\!|n\oplus l|\!\}^4 \stackrel{\text{def}}{=} n\oplus l$$
$$\{\!|n\&l|\!\}^4 \stackrel{\text{def}}{=} n\&l$$

The above definition handles the dyadic case. Notice that we distinguish two kinds of internal actions: the first case above results from the translation of function applications; the second case is associated to internal actions arising from the mapping of polyadic name synchronization. We may now state operational correspondence:

Proposition 6.5 *(Operational correspondence. From $\mathsf{HO}\tilde{\pi}$ to $\mathsf{HO}\pi$). Let $\Gamma;\emptyset;\Delta\vdash P$ be an $\mathsf{HO}\tilde{\pi}$ process.*

1. $\Gamma;\Delta\vdash P \overset{\ell}{\mapsto} \Delta'\vdash P'$ *implies*

   a) *If* $\ell=(\nu\,\tilde{m}')n!\langle\tilde{m}\rangle$ *then* $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\ell_1}{\mapsto}\dots\overset{\ell_k}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$ *with* $\{\!|\ell|\!\}^4=\ell_1,\cdots,\ell_k$.
   b) *If* $\ell=n?\langle\tilde{m}\rangle$ *then* $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\ell_1}{\mapsto}\dots\overset{\ell_k}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$ *with* $\{\!|\ell|\!\}^4=\ell_1,\cdots,\ell_k$.
   c) *If* $\ell\in\{(\nu\,\tilde{m})n!\langle\lambda\tilde{x}.\,R\rangle,n?\langle\lambda\tilde{x}.\,R\rangle\}$ *then* $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\ell'}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$ *with* $\{\!|\ell|\!\}^4=\ell'$.
   d) *If* $\ell\in\{n\oplus l,n\&l\}$ *then* $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\ell}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$.
   e) *If* $\ell=\tau_\beta$ *then* $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\tau_\beta}{\mapsto}\overset{\tau_{\mathsf{s}}}{\mapsto}\dots\overset{\tau_{\mathsf{s}}}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$ *with* $\{\!|\ell|\!\}^=\tau_\beta,\underbrace{\tau_{\mathsf{s}},\cdots,\tau_{\mathsf{s}}}_{k}$.
   f) *If* $\ell=\tau$ *then* $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\tau}{\mapsto}\dots\overset{\tau}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$ *with* $\{\!|\ell|\!\}^4=\underbrace{\tau,\cdots,\tau}_{k}$.

2. $(\!|\Gamma|\!)^4;(\!|\Delta|\!)^4\vdash\llbracket P\rrbracket^4 \overset{\ell}{\mapsto}(\!|\Delta_1|\!)^4\vdash P_1$ *implies*

   a) *If* $\ell\in\{n?\langle m\rangle,n!\langle m\rangle,(\nu\,m)n!\langle m\rangle\}$ *then* $\Gamma;\Delta\vdash P\overset{\ell'}{\mapsto}\Delta'\vdash P'$ *and* $(\!|\Gamma|\!)^4;(\!|\Delta_1|\!)^4\vdash P_1\overset{\ell_2}{\mapsto}\dots\overset{\ell_k}{\mapsto}(\!|\Delta'|\!)^4\vdash\llbracket P'\rrbracket^4$ *with* $\{\!|\ell'|\!\}^4=\ell_1,\cdots,\ell_k$ *and* $\ell=\ell_1$.
   b) *If* $\ell\in\{(\nu\,\tilde{m})n!\langle\lambda x.\,R\rangle,n?\langle\lambda x.\,R\rangle\}$ *then* $\Gamma;\Delta\vdash P\overset{\ell'}{\mapsto}\Delta'\vdash P'$ *with* $\{\!|\ell'|\!\}^4=\ell$ *and* $P_1\equiv\llbracket P'\rrbracket^4$.

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

d) If $\ell = \tau_\beta$ *then* $\Gamma; \Delta \vdash P \overset{\tau_\beta}{\longmapsto} \Delta' \vdash P'$ *and* $(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau_s}{\longmapsto} \dots \overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$ *with* $\{\!|\ell|\!\}^4 = \tau_\beta, \underbrace{\tau_s, \cdots, \tau_s}_{k}$.

e) If $\ell = \tau$ *and* $\ell \neq \tau_\beta$ *then* $\Gamma; \Delta \vdash P \overset{\tau}{\mapsto} \Delta' \vdash P'$ *and* $(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau}{\mapsto} \dots \overset{\tau}{\mapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$ *with* $\{\!|\ell|\!\}^4 = \underbrace{\tau, \cdots, \tau}_{k}$.

Proof. The proof of both parts is by transition induction, following the mapping defined in Fig. 13. See Proposition Appendix B.11 (Page 50) in B.4. □

As expected, the above correspondence is most interesting in the cases of input and output actions and of τ_β/τ reductions in the source process. The case of first-order input and output (Items 1(a) and 1(b)) a single source action is reflected as k independent actions from the corresponding target process. In contrast, when the source action is a single τ_β (Item 1(e)) then we have $k+1$ independent actions: the first is a τ_β synchronisation as it corresponds to the application of a fresh session name; the other k actions are τ_s synchronisations, as they correspond to the communication of the k arguments to the function, which are passed around using the session established thanks to the first τ_β action. When the source action is a single regular synchronisation (Item 1(f)) then we have k synchronisations in the target side. The correspondences in the soundness direction follow similar intuitions. We may now state:

Proposition 6.6 *(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, Theorems 5.1 and 5.2, and Proposition 4.1 (Page 13), we derive the following:

Theorem 6.2 *(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. 13) is precise.*

Proof. According to Definition 4.6, preciseness includes syntax-, type-, and semantics-preservation. Syntax preservation follows immediately from the definition of the encoding. Type preservation follows from Proposition 6.4 (Page 27). Semantics-preservation follows from Proposition 6.5 (Page 28) and Proposition 6.6 (Page 29). □

We then have the following corollary:

Corollary 6.2 *(Encodability of* $\mathsf{HO}\widetilde{\pi}$ *into* $\mathsf{HO}\pi$ *and* π*). Consider the typed encodings*

- $\langle [\![\cdot]\!]^1_f, (\!|\cdot|\!)^1 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\mathsf{HO}}$ *(cf. Definition 5.2)*
- $\langle [\![\cdot]\!]^2, (\!|\cdot|\!)^2 \rangle : \mathcal{L}_{\mathsf{HO}\pi} \to \mathcal{L}_{\pi}$ *(cf. Definition 5.4)*
- $\langle [\![\cdot]\!]^4, (\!|\cdot|\!)^4 \rangle : \mathcal{L}_{\mathsf{HO}\widetilde{\pi}} \to \mathcal{L}_{\mathsf{HO}\pi}$ *(cf. Definition 6.3)*

Then the following typed encodings are precise:

- $\langle [\![\cdot]\!]^1 \circ [\![\cdot]\!]^4, (\!|\cdot|\!)^1 \circ (\!|\cdot|\!)^4 \rangle : \mathcal{L}_{\mathsf{HO}\widetilde{\pi}} \to \mathcal{L}_{\mathsf{HO}}$
- $\langle [\![\cdot]\!]^2 \circ [\![\cdot]\!]^4, (\!|\cdot|\!)^2 \circ (\!|\cdot|\!)^4 \rangle : \mathcal{L}_{\mathsf{HO}\widetilde{\pi}} \to \mathcal{L}_{\pi}$

Proof. Directly from Theorem 5.1, Theorem 5.2, and Theorem 6.2 (which give preciseness for all the involved encodings), using Proposition 4.1 (Page 13). □

By combining Theorems 6.1 and 6.2, we can extend preciseness to the super-calculus $\mathsf{HO}\widetilde{\pi}^+$, which subsumes both $\mathsf{HO}\pi^+$ and $\mathsf{HO}\widetilde{\pi}$.

7. Related work

There is a vast literature on expressiveness for process calculi; we refer to [32] and [33, §2.3] for surveys. Our study offers new encodability results and casts known results [39] into a session typed setting. Our work stresses the view of "encodings as protocols", namely session protocols which enforce linear and shared disciplines for names, a distinction little explored in previous works. This distinction enables us to obtain refined operational correspondence results (cf. Propositions 5.2, 5.5, 6.2, 6.5). We showed that HO suffices to encode the first-order session calculus [12], here denoted π. To our knowledge, this is a new result; its significance is stressed by the demanding encodability criteria considered, in particular full abstraction up to typed bisimilarities ($\approx^{\mathrm{H}}/\approx^{\mathrm{C}}$, cf. Propositions 5.3 and 5.6). This encoding is relevant in a broader setting, as known

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

encodings of name-passing into higher-order calculi [42,2,24,47,49] require limitations in source/target languages, do not consider types, and/or fail to satisfy strong encodability criteria (see below). We also showed that HO can encode HOπ and its extension with higher-order applications (HOπ^+). Thus, all these calculi are equally expressive with fully abstract encodings (up to $\approx^{\mathrm{H}}/\approx^{\mathrm{C}}$). To our knowledge, these are the first 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 [10,7,46,34,36]. Our formulation of precise encoding (Definition 4.6) builds upon existing proposals (e.g., [31,10,20]) to account for the session types associated to HOπ.

Early expressiveness studies for higher-order calculi are [44,39]; recent works include [2,20,21,47,48]. Due to the close relationship between higher-order process calculi and functional calculi, encodings of (variants of) the λ-calculus into the π-calculus (see, e.g., [38,8,50,3,43]) are also related. Sangiorgi's encoding of the higher-order π-calculus into the π-calculus [39] is fully abstract with respect to reduction-closed, barbed congruence. We have shown in § 5.2 that the analogue of Sangiorgi's encoding for the session-typed setting also enjoys full abstraction (up to $\approx^{\mathrm{H}}/\approx^{\mathrm{C}}$, cf. Proposition 5.5 (Page 20)). A basic form of input/output types is used in [41], where the encoding in [39] is cast in the asynchronous setting, with output and applications coalesced in a single construct. Building upon [41], a simply typed encoding for synchronous processes is given in [42]; the reverse encoding (i.e., first-order communication into higher-order processes) is also studied for an asynchronous, localised π-calculus (where only the output capability of names can be sent around). The work [40] studies hierarchies for calculi with *internal* first-order mobility and with higher-order mobility without name-passing (similarly as in HO); these hierarchies are defined according to the order of types needed in typing. Via fully abstract encodings, it is shown that name- and process-passing calculi with equal order of types have the same expressiveness.

Other related works are [2,24,47,21]. The paper [2] gives a fully abstract encoding of the π-calculus into Homer, a higher-order calculus with explicit locations, local names, and nested locations. The paper [24] presents a *reflective* calculus with a "quoting" operator: names are quoted processes and represent the code of a process; name-passing is then a way of passing the code of a process. This reflective calculus can encode both first- and higher-order π-calculus. Building upon [45], the work [47] 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. The paper [49] defines an encoding of the (untyped) π-calculus without relabelling. This encoding is quite different from the one in § 5.1: in [49] names are encoded using polyadic name abstractions (called *pipes*); guarded replication enables infinite behaviours. While our encoding satisfies full abstraction, the encoding in [49] 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 [49] describe this result as "weak".

A core higher-order calculus is studied in [21]: it lacks restriction, name passing, output prefix, and replication/recursion. Still, this untyped subcalculus of HO is Turing equivalent. The work [20] extends this core calculus with restriction, output prefix, and polyadicity; it shows that synchronous communication can encode asynchronous communication, and that process passing polyadicity induces an expressiveness hierarchy. The paper [48] complements [20] by studying the expressivity of second-order process abstractions. Polyadicity is shown to induce an expressiveness hierarchy; also, by adapting the encoding in [39], process abstractions are encoded into name abstractions. In contrast, here we give a fully abstract encoding of HO$\widetilde{\pi}^+$ into HO that preserves session types; this improves [20,48] by enforcing linearity disciplines on process behaviour. The focus of [20,47–49] is on untyped, higher-order processes; they do not address communication disciplined by (session) type systems.

Within session types, the works [6,5] encode binary sessions into a linearly typed π-calculus. While [6] gives an encoding of π into a linear calculus (an extension of [3]), the work [5] gives operational correspondence (without full abstraction) for the first- and higher-order π-calculi into [14]. By the result of [6], HOπ^+ is encodable into the linearly typed π-calculi. The syntax of HOπ is a subset of that in [27,28]. The work [27] develops a higher-order session calculus with process abstractions and applications; it admits the type $U = U_1 \to U_2 \dots U_n \to \diamond$ and its linear type U^1 which corresponds to $\tilde{U} \to \diamond$ and $\tilde{U} \multimap \diamond$ in a super-calculus of HOπ^+ and HO$\widetilde{\pi}$. Our results show that the calculus in [27] is not only expressed but also reasoned in HO via precise encodings (with a limited form of arrow types: $C \to \diamond$ and $C \multimap \diamond$). The work [30] studies two encodings: from PCF with an effect system into a session-typed π-calculus, and its reverse. The reverse encoding is used to implement session channel passing in Concurrent Haskell. In future work we plan to use the core calculi studied in this paper to implement higher-order communication efficiently into Concurrent Haskell without losing its expressiveness.

8. Concluding remarks

We have thoroughly studied the expressivity of the higher-order π-calculus with sessions, here denoted HOπ. To this end, we developed a new abstract notion of (precise) encoding that accounts for (session) types in both source and target calculi. Indeed, unlike most previous works, we have carried out our expressiveness study in the setting of *session types*. Types not only delineate and enable encodings; they inform the techniques required to reason about their correctness properties.

Our results cover a wide spectrum of features intrinsic to higher-order concurrency: pure process-passing (first- and higher-order abstractions), name-passing, polyadicity, linear/shared communication (cf. 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

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

strong typed bisimilarities. Indeed, although our encodings may be used in an untyped setting, session type information is critical to establish key properties for preciseness, in particular full abstraction.

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgments

We are grateful to Alen Arslanagić and to the anonymous reviewers for their useful remarks and suggestions.

This work has been partially sponsored by the Doctoral Prize Fellowship, EPSRC EP/K011715/1, EPSRC EP/K034413/1, EPSRC EP/L00058X/1, EPSRC EP/N027833/1, EPSRC EP/N028201/1, and EU COST Actions IC1201 (BETTY), IC1402 (ARVI), IC1405 (Reversible Computation), and CA15123 (EUTypes).

Kouzapas was partially funded by the European Union via the Horizon 2020: Future Emerging Topics call (FETOPEN), grant EU736876, project VISORSURF (http://www.visorsurf.eu).

Pérez has been partially supported by the Netherlands Organization for Scientific Research (NWO) under the VIDI Project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software). He is also affiliated to the NOVA Laboratory for Computer Science and Informatics (NOVA LINCS), Universidade Nova de Lisboa, Portugal.

Appendix A. Behavioural semantics

We report auxiliary definitions and results from [15,17], which were informally introduced in § 3.3.

A.1. Labelled transition system for processes

We define the interaction of processes with their environment using action labels ℓ:

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

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 $n\&l$, respectively. We write $\mathtt{fn}(\ell)$ and $\mathtt{bn}(\ell)$ to denote the sets of free/bound names in ℓ, respectively. Given $\ell \neq \tau$, we say ℓ is a *visible action*; we write $\mathtt{subj}(\ell)$ to denote its *subject*. This way, we have: $\mathtt{subj}((\nu\, \widetilde{m})n!\langle V\rangle) = \mathtt{subj}(n?\langle V\rangle) = \mathtt{subj}(n \oplus l) = \mathtt{subj}(n\&l) = n$.

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.

Definition Appendix A.1 *(Dual actions).* We define duality on actions as the least symmetric relation $\asymp$ on action labels that satisfies:

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

The (early) labelled transition system (LTS) fpr *untyped* processes is given in Fig. A.14. We write $P_1 \xrightarrow{\ell} P_2$ with the usual meaning. The rules are standard [19,18]; we comment on some of them. A process with an output prefix can interact with the environment with an output action that carries a value V (Rule $\langle$Snd$\rangle$). Dually, in Rule $\langle$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 $\langle$Sel$\rangle$ and $\langle$Bra$\rangle$, respectively. Rule $\langle$Res$\rangle$ enables an observable action from a process with an outermost restriction, provided that the restricted name does not occur free in the action. If a restricted name occurs free in the carried value of an output action, the process performs scope opening (Rule $\langle$New$\rangle$). Rule $\langle$Rec$\rangle$ handles recursion unfolding. Rule $\langle$Tau$\rangle$ states that two parallel processes which perform dual actions can synchronise by an internal transition. Rules $\langle \mathrm{Par}_L\rangle$/$\langle \mathrm{Par}_R\rangle$ and $\langle$Alpha$\rangle$ define standard treatments for actions under parallel composition and α-conversion.

A.2. Environmental labelled transition system

Our typed LTS is obtained by coupling the untyped LTS given before with a labelled transition relation on typing environments, given in Fig. A.15. Building upon the reduction relation for session environments in Definition 3.4, such a relation is defined on triples of environments by extending the LTSs in [19,18]; it is denoted

$$(\Gamma_1, \Lambda_1, \Delta_1) \xrightarrow{\ell} (\Gamma_2, \Lambda_2, \Delta_2)$$

Recall that Γ admits weakening. Using this principle (not valid for Λ and Δ), we have $(\Gamma', \Lambda_1, \Delta_1) \overset{\ell}{\mapsto} (\Gamma', \Lambda_2, \Delta_2)$ whenever $(\Gamma, \Lambda_1, \Delta_1) \overset{\ell}{\mapsto} (\Gamma', \Lambda_2, \Delta_2)$.

$$\langle \textsc{App} \rangle \quad \frac{}{(\lambda x.\, P)\, V \xrightarrow{\tau} P\{V/x\}} \qquad \langle \textsc{Snd} \rangle \quad \frac{}{n!\langle V \rangle.P \xrightarrow{n!\langle V \rangle} P} \qquad \langle \textsc{Rv} \rangle \quad \frac{}{n?(x).P \xrightarrow{n?\langle V \rangle} P\{V/x\}} \qquad \langle \textsc{Sel} \rangle \quad \frac{}{s \triangleleft l.P \xrightarrow{s \oplus l} P}$$

$$\langle \textsc{Bra} \rangle \quad \frac{j \in I}{s \triangleright \{l_i : P_i\}_{i \in I} \xrightarrow{s \& l_j} P_j} \qquad \langle \textsc{Alpha} \rangle \quad \frac{P \equiv_\alpha Q \qquad Q \xrightarrow{\ell} P'}{P \xrightarrow{\ell} P'} \qquad \langle \textsc{Res} \rangle \quad \frac{P \xrightarrow{\ell} P' \qquad n \notin \mathtt{fn}(\ell)}{(\nu\, n)P \xrightarrow{\ell} (\nu\, n)P'}$$

$$\langle \textsc{New} \rangle \quad \frac{P \xrightarrow{(\nu\, \widetilde{m})n!\langle V \rangle} P' \qquad m_1 \in \mathtt{fn}(V)}{(\nu\, m_1)P \xrightarrow{(\nu\, m_1 \cdot \widetilde{m})n!\langle V \rangle} P'} \qquad \langle \textsc{Par}_L \rangle \quad \frac{P \xrightarrow{\ell} P' \qquad \mathtt{bn}(\ell) \cap \mathtt{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\ell} P' \mid Q}$$

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

Fig. A.14. The untyped LTS for HOπ processes. We omit Rule $\langle \mathrm{Par}_R \rangle$.

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

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

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

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

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

Fig. A.15. Labelled transition system for typed environments.

Input actions These 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]). Following similar principles, Rule [ShRv] defines input actions for shared names.

Output actions These 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 $\Delta \cdot s{:}S$ should include the session environment of the sent value V (denoted Δ' in the rule), *excluding* the session environments of names m_j in $\widetilde{m}$ which restrict the scope of value V (denoted Δ_j in the rule). Analogously, the linear variable environment Λ' of V should be included in Λ. The rule defines the scope extrusion of session names in $\widetilde{m}$; consequently, environments associated to their dual endpoints (denoted Δ'_j in the rule) appear in the resulting session environment. Similarly for shared names in $\widetilde{m}$ that are extruded. All free values used for typing V (denoted Λ' and Δ' in the rule) are subtracted from the resulting type tuple. The prefix of session s is consumed by the action. Rule [ShSnd] follows similar ideas for output actions on shared names: the name must be typed with $\langle U \rangle$; conditions on value 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 reduces the session environment (cf. Definition 3.4) or keeps it unchanged.

We illustrate Rule [SSnd] by means of an example:

Example Appendix A.1. Consider environment tuple $(\Gamma;\emptyset;s:!\langle(!\langle S\rangle;\mathtt{end})\multimap\diamond\rangle;\mathtt{end}\cdot s':S)$ and typed value $V=\lambda x.x!\langle s'\rangle.m?(z).\mathbf{0}$ with

$$\Gamma;\emptyset;s':S\cdot m:?(\mathtt{end});\mathtt{end}\vdash V\triangleright(!\langle S\rangle;\mathtt{end})\multimap\diamond$$

Then, by Rule [SSnd], we can derive:

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

Observe how the protocol along s is partially consumed; also, the resulting session environment is extended with $\overline{m}$, the dual endpoint of the extruded name m.

Recall that we sometimes annotate the output action $(\nu\,\widetilde{m})n!\langle V\rangle$ with the type of V; this is written as $(\nu\,\widetilde{m})n!\langle V:U\rangle$ (cf. Remark 3.1).

The typed LTS combines the LTSs in Fig. A.14 and Fig. A.15.

Definition Appendix A.2 *(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 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 $\Longrightarrow$.

A typed transition relation requires type judgements with an empty Λ, i.e., an empty environment for linear higher-order types. Notice that for open process terms (i.e., with free variables), we can always apply Rule (EProm) (cf. Fig. 5) and obtain an empty Λ. We will be working with closed process terms, i.e., processes without free variables.

A.3. Characteristic values and the refined LTS

We first define characteristic processes/values:

Definition Appendix A.3 *(Characteristic process and values).* Let u and U be a name and a type, respectively. The *characteristic process* of U (along u), denoted $(\!|U|\!)^u$, and the *characteristic value* of U, denoted $(\!|U|\!)_{\mathsf{c}}$, are defined in Fig. 6.

We can verify that characteristic processes/values do inhabit their associated type.

Proposition Appendix A.1 *(Characteristic processes/values inhabit their types).*

1. *Let U be a channel type. Then, for some Γ,Δ, we have $\Gamma;\emptyset;\Delta\vdash(\!|U|\!)_{\mathsf{c}}\triangleright U$.*
2. *Let S be a session type. Then, for some Γ,Δ, we have $\Gamma;\emptyset;\Delta\cdot s:S\vdash(\!|S|\!)^s\triangleright\diamond$.*
3. *Let U be a channel type. Then, for some Γ,Δ, we have $\Gamma\cdot a:U;\emptyset;\Delta\vdash(\!|U|\!)^a\triangleright\diamond$.*

Definition Appendix A.4 *(Trigger value).* Given a fresh name t, the *trigger value* on t is defined as the abstraction $\lambda x.t?(y).(y\,x)$.

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 Appendix A.5 *(Refined typed labelled transition system).* The refined typed labelled transition relation on typing environments

$$(\Gamma_1;\Lambda_1;\Delta_1)\overset{\ell}{\mapsto}(\Gamma_2;\Lambda_2;\Delta_2)$$

is defined on top of the rules in Fig. A.15 using the following rules:

[Tr]
$$\frac{(\Gamma_1;\Lambda_1;\Delta_1)\xrightarrow{\ell}(\Gamma_2;\Lambda_2;\Delta_2)\qquad\ell\neq n?\langle V\rangle}{(\Gamma_1;\Lambda_1;\Delta_1)\overset{\ell}{\mapsto}(\Gamma_2;\Lambda_2;\Delta_2)}$$

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

[RRcv]

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

Then, the refined typed labelled transition system

$$\Gamma;\Delta_1\vdash P_1\overset{\ell}{\mapsto}\Delta_2\vdash P_2$$

is given as in Definition Appendix A.2, replacing the requirement

$$(\Gamma,\emptyset,\Delta_1)\xrightarrow{\ell}(\Gamma,\emptyset,\Delta_2)$$

with $(\Gamma_1;\Lambda_1;\Delta_1)\overset{\ell}{\mapsto}(\Gamma_2;\Lambda_2;\Delta_2)$, as just defined. Following Definition Appendix A.2, we write $\mapsto\!\!\!\!\Rightarrow$ for the reflexive and transitive closure of $\overset{\tau}{\mapsto}$, $\overset{\ell}{\mapsto\!\!\!\!\Rightarrow}$ for the transitions $\mapsto\!\!\!\!\Rightarrow\overset{\ell}{\mapsto}\mapsto\!\!\!\!\Rightarrow$, and $\overset{\hat{\ell}}{\mapsto\!\!\!\!\Rightarrow}$ for $\overset{\ell}{\mapsto\!\!\!\!\Rightarrow}$ if $\ell\neq\tau$ otherwise $\mapsto\!\!\!\!\Rightarrow$.

Notice that the (refined) transition $\Gamma;\Delta_1\vdash P_1\overset{\ell}{\mapsto}\Delta_2\vdash P_2$ implies the (ordinary) transition $\Gamma;\Delta_1\vdash P_1\xrightarrow{\ell}\Delta_2\vdash P_2$.

A.4. More on deterministic transitions and up-to techniques

As hinted at earlier, internal transitions associated to session interactions or β-reductions are deterministic. To define an auxiliary proof technique that exploits determinacy we require some auxiliary definitions.

Definition Appendix A.6 *(Deterministic transitions).* Suppose $\Gamma;\emptyset;\Delta\vdash P\triangleright\diamond$ with balanced Δ. Transition $\Gamma;\Delta\vdash P\overset{\tau}{\mapsto}\Delta'\vdash P'$ is called:

- a *session-transition* whenever transition $P\xrightarrow{\tau}P'$ is derived using Rule $\langle\text{Tau}\rangle$ (where $\mathtt{subj}(\ell_1)$ and $\mathtt{subj}(\ell_2)$ in the premise are dual endpoints), possibly followed by uses of Rules $\langle\text{Alpha}\rangle$, $\langle\text{Res}\rangle$, $\langle\text{Rec}\rangle$, or $\langle\text{Par}_L\rangle/\langle\text{Par}_R\rangle$ (cf. Fig. A.14).
- a *β-transition* whenever transition $P\xrightarrow{\tau}P'$ is derived using Rule $\langle\text{App}\rangle$, possibly followed by uses of Rules $\langle\text{Alpha}\rangle$, $\langle\text{Res}\rangle$, $\langle\text{Rec}\rangle$, or $\langle\text{Par}_L\rangle/\langle\text{Par}_R\rangle$ (cf. Fig. A.14).

Notation 3. *We use the following notations:*

- $\Gamma;\Delta\vdash P\overset{\tau_{\mathrm{s}}}{\mapsto}\Delta'\vdash P'$ *denotes a session-transition.*
- $\Gamma;\Delta\vdash P\overset{\tau_\beta}{\mapsto}\Delta'\vdash P'$ *denotes a β-transition.*
- $\Gamma;\Delta\vdash P\overset{\tau_{\mathrm{d}}}{\mapsto}\Delta'\vdash P'$ *denotes either a session-transition or a β-transition.*
- *We write* $\overset{\tau_{\mathrm{d}}}{\Longmapsto}$ *to denote a (possibly empty) sequence of deterministic steps* $\overset{\tau_{\mathrm{d}}}{\mapsto}$.

Using the above properties, we can state the following up-to technique. Recall that the higher-order trigger $t\leftarrow\!\!\!\!\rightarrow_{\mathrm{H}}V$ has been defined in (2) (Page 9).

Lemma Appendix A.1 *(Up-to deterministic transition). Let $\Gamma;\Delta_1\vdash P_1\ \Re\ \Delta_2\vdash Q_1$ such that if whenever:*

1. *$\forall(\nu\,\widetilde{m_1})n!\langle V_1\rangle$ such that $\Gamma;\Delta_1\vdash P_1\overset{(\nu\,\widetilde{m_1})n!\langle V_1\rangle}{\longmapsto}\Delta_3\vdash P_3$ implies that $\exists Q_2, V_2$ such that $\Gamma;\Delta_2\vdash Q_1\overset{(\nu\,\widetilde{m_2})n!\langle V_2\rangle}{\Longmapsto}\Delta_2'\vdash Q_2$ and $\Gamma;\Delta_3\vdash P_3\overset{\tau_{\mathrm{d}}}{\Longmapsto}\Delta_1'\vdash P_2$ and for a fresh name t and $\Delta_1'',\Delta_2'':$*

$$\Gamma;\Delta_1''\vdash(\nu\,\widetilde{m_1})(P_2\mid t\leftarrow\!\!\!\!\rightarrow_{\mathrm{H}}V_1)\ \Re\ \Delta_2''\vdash(\nu\,\widetilde{m_2})(Q_2\mid t\leftarrow\!\!\!\!\rightarrow_{\mathrm{H}}V_2)$$

2. *$\forall\ell\neq(\nu\,\widetilde{m})n!\langle V\rangle$ such that: $\Gamma;\Delta_1\vdash P_1\overset{\ell}{\mapsto}\Delta_3\vdash P_3$ implies that $\exists Q_2$ such that $\Gamma;\Delta_1\vdash Q_1\overset{\hat{\ell}}{\mapsto\!\!\!\!\Rightarrow}\Delta_2'\vdash Q_2$ and $\Gamma;\Delta_3\vdash P_3\overset{\tau_{\mathrm{d}}}{\Longmapsto}\Delta_1'\vdash P_2$ and $\Gamma;\Delta_1'\vdash P_2\ \Re\ \Delta_2'\vdash Q_2$.*
3. *The symmetric cases of 1 and 2.*

Then $\Re\subseteq\approx^{\mathrm{H}}$.

Proof (Sketch). The proof proceeds by showing that the relation

$$\Re^{\overset{\tau_{\mathrm{d}}}{\Longmapsto}}=\{(P_2,Q_1)\mid\Gamma;\Delta_1\vdash P_1\ \Re\ \Delta_2'\vdash Q_1,\ \ \Gamma;\Delta_1\vdash P_1\overset{\tau_{\mathrm{d}}}{\Longmapsto}\Delta_1'\vdash P_2\}$$

is a higher-order bisimulation, which requires the use of Proposition 3.1 (Page 10). □

Appendix B. Expressiveness results

In this section we give the proofs for the expressiveness results stated in §5 and §6. Proving precise encodings entails proving type preservation, operational correspondence, and full abstraction (cf. Definition 4.6). For operational correspondence, recall that we prove a stronger statement than Definition 4.5(3), as we consider both visible and internal actions. For full abstraction, we rely on a notational convention:

Notation 4 *(Typed relations). For the sake of readability, when describing typed relations we shall omit typing information for pairs of processes, which is usually clear from the context. This way, e.g., in the proof of Proposition Appendix B.3 (Page 40) we write*

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

instead of

$$\begin{aligned}\Re = \{(P_1, Q_1) \mid\ & \Gamma; \emptyset; \Delta_1 \vdash P_1 \triangleright \diamond \ \wedge\ \Gamma; \emptyset; \Delta_2 \vdash Q_1 \triangleright \diamond \\ & \wedge\ (\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \approx^{\mathrm{H}} (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]^1_f\}\end{aligned}$$

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

In this section we prove Theorem 5.1 (Page 18) which states that the encoding $[\![\cdot]\!]^1_f$ of $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_{\mathsf{HO}}$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated as Proposition 5.1 (Page 16) and proven here as Proposition Appendix B.1 (Page 35).
- Operational Correspondence, stated as Proposition 5.2 (Page 17) and proven here as Proposition Appendix B.2 (Page 37).
- Full Abstraction, stated as Proposition 5.3 (Page 18) and proven here as Proposition Appendix B.3 (Page 40).

Proposition Appendix B.1 *(Type preservation, $\mathsf{HO}\pi$ into HO). Let P be an $\mathsf{HO}\pi$ 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 $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. We consider four interesting cases:

1. Case $P = k!\langle n\rangle.P'$. Then there are several sub-cases, depending on whether k and n are linear or not. We content ourselves by checking the case in which k is a session (linear) name. There are two sub-cases, depending on whether n is a linear or a shared name.

   (a) 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:

   $$(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \vdash [\![P']\!]^1_f \triangleright \diamond$$

   Let us write U_1 to stand for $?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \multimap \diamond$. The corresponding typing in the target language is as follows:

   $$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{(\!|\Gamma|\!)^1; \{x : (\!|S_1|\!)^1 \multimap \diamond\}; \emptyset \vdash x \triangleright (\!|S_1|\!)^1 \multimap \diamond \qquad (\!|\Gamma|\!)^1; \emptyset; \{k' : (\!|S_1|\!)^1\} \vdash k' \triangleright (\!|S_1|\!)^1}{(\!|\Gamma|\!)^1; \{x : (\!|S_1|\!)^1 \multimap \diamond\}; k' : (\!|S_1|\!)^1 \vdash x\,k' \triangleright \diamond}}{(\!|\Gamma|\!)^1; \{x : (\!|S_1|\!)^1 \multimap \diamond\}; k' : (\!|S_1|\!)^1 \cdot z : \mathtt{end} \vdash x\,k' \triangleright \diamond}}{(\!|\Gamma|\!)^1; \emptyset; k' : (\!|S_1|\!)^1 \cdot z : ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \vdash z?(x).(x\,k') \triangleright \diamond}}{(\!|\Gamma|\!)^1; \emptyset; k' : (\!|S_1|\!)^1 \vdash \lambda z.\, z?(x).(x\,k') \triangleright U_1} \qquad \text{(B.1)}$$

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

   (b) In the second sub-case, we have $n = a$ (output of a shared name). Then we have the following typing in the source language:

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

   The typing in the target language is derived similarly as in the first sub-case.

2. Case $P = k?(x).Q$. Again, there are several sub-cases, depending on whether k and x have linear types. We content ourselves by checking the case in which k is a session (linear) name. We have two sub-cases, depending on the type of x (linear or shared name).

   (a) In the first case, x stands for a linear name. 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:

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

   Let us write U_1 to stand for $(?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end}) \multimap \diamond$. The corresponding typing in the target language is as follows; we have three auxiliary derivations:

   $$\frac{(\!|\Gamma|\!)^1; \{x : U_1\}; \emptyset \vdash x \triangleright U_1 \qquad (\!|\Gamma|\!)^1; \emptyset; s : ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \vdash s \triangleright ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end}}{(\!|\Gamma|\!)^1; \{x : U_1\}; s : ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \vdash xs \triangleright \diamond} \tag{B.2}$$

   $$\frac{(\!|\Gamma|\!)^1; \emptyset; \emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!|\Gamma|\!)^1; \emptyset; \overline{s} : \mathtt{end} \vdash \mathbf{0} \triangleright \diamond} \quad \frac{(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \cdot x : (\!|S_1|\!)^1 \vdash [\![Q]\!]^1_f \triangleright \diamond}{(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \vdash \lambda x.\, [\![Q]\!]^1_f \triangleright (\!|S_1|\!)^1 \multimap \diamond}$$
   $$\frac{}{(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \cdot \overline{s} : !\langle (\!|S_1|\!)^1 \multimap \diamond \rangle; \mathtt{end} \vdash \overline{s}!\langle \lambda x.\, [\![Q]\!]^1_f \rangle.\mathbf{0} \triangleright \diamond} \tag{B.3}$$

   $$\frac{\begin{array}{l}(\!|\Gamma|\!)^1; \{x : U_1\}; s : ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \vdash xs \triangleright \diamond \quad \text{(B.2)} \\ (\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \cdot \overline{s} : !\langle (\!|S_1|\!)^1 \multimap \diamond \rangle; \mathtt{end} \vdash \overline{s}!\langle \lambda x.\, [\![Q]\!]^1_f \rangle.\mathbf{0} \triangleright \diamond \quad \text{(B.3)}\end{array}}{\begin{array}{r}(\!|\Gamma|\!)^1; \{x : U_1\}; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \cdot s : ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \cdot \overline{s} : !\langle (\!|S_1|\!)^1 \multimap \diamond \rangle; \mathtt{end} \\ \vdash xs \mid \overline{s}!\langle \lambda x.\, [\![Q]\!]^1_f \rangle.\mathbf{0} \triangleright \diamond\end{array}} \tag{B.4}$$

   Finally we have:

   $$\frac{\frac{\begin{array}{r}(\!|\Gamma|\!)^1; \{x : U_1\}; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \cdot s : ?((\!|S_1|\!)^1 \multimap \diamond); \mathtt{end} \cdot \overline{s} : !\langle (\!|S_1|\!)^1 \multimap \diamond \rangle; \mathtt{end} \\ \vdash xs \mid \overline{s}!\langle \lambda x.\, [\![Q]\!]^1_f \rangle.\mathbf{0} \triangleright \diamond\end{array} \text{(B.4)}}{(\!|\Gamma|\!)^1; \{x : U_1\}; (\!|\Delta|\!)^1 \cdot k : (\!|S|\!)^1 \vdash (\nu\, s)(xs \mid \overline{s}!\langle \lambda x.\, [\![Q]\!]^1_f \rangle.\mathbf{0}) \triangleright \diamond}}{(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot k : ?(U_1); (\!|S|\!)^1 \vdash k?(x).(\nu\, s)(xs \mid \overline{s}!\langle \lambda x.\, [\![Q]\!]^1_f \rangle.\mathbf{0}) \triangleright \diamond}$$

   (b) In the second sub-case, x is a shared name, and we have the following typing in the source language:

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

   The typing in the target language is derived similarly as in the first sub-case.

3. Case $P_0 = X$. Then we have the following typing in the source language:

   $$\underbrace{\Gamma' \cdot X : \Delta}_{\Gamma}; \emptyset; \Delta \vdash X \triangleright \diamond$$

   Let $\Delta = n_1 : S_1, \ldots, n_m : S_m$, with $\mathtt{dom}(\Delta) = \tilde{n}$. By Definition 5.2, we have that

   $$(\!|\Gamma|\!)^1 = (\!|\Gamma' \cdot X : \{n_i : S_i\}_{1 \le i \le m}|\!)^1 = (\!|\Gamma'|\!)^1 \cdot z_X : (\underbrace{(\!|S_1|\!)^1, \ldots, (\!|S_m|\!)^1}_{\widetilde{T}}, S^*) \to \diamond$$

   where $S^* = \mu \mathrm{t}.?((\widetilde{T}, \mathrm{t}) \to \diamond); \mathtt{end}$, which is equivalent to $?((\widetilde{T}, S^*) \to \diamond); \mathtt{end}$. By Fig. 9,

   $$[\![X]\!]^1_f = (\nu\, s)(z_X\, (\tilde{n}, s) \mid \overline{s}!\langle z_X \rangle.\mathbf{0})$$

   with $\tilde{n} = f(X)$. We shall show that

   $$(\!|\Gamma'|\!)^1 \cdot z_X : (\widetilde{T}, S^*) \to \diamond; \emptyset; (\!|\Delta|\!)^1 \vdash [\![X]\!]^1_f \triangleright \diamond$$

   We first have two auxiliary derivations:

   $$\frac{\frac{}{(\!|\Gamma|\!)^1; \emptyset; \emptyset \vdash z_X \triangleright (\widetilde{T}, S^*) \to \diamond} \quad \frac{(\!|\Gamma|\!)^1; \emptyset; \{n_i : (\!|S_i|\!)^1\} \vdash n_i \triangleright (\!|S_i|\!)^1 \qquad (\!|\Gamma|\!)^1; \emptyset; \{s : S^*\} \vdash s \triangleright ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end}}{}}{(\!|\Gamma|\!)^1; \emptyset; (\!|\Delta|\!)^1 \cdot s : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end} \vdash z_X\, (\tilde{n}, s) \triangleright \diamond} \tag{B.5}$$

and

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1;\ \emptyset;\ \emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!|\Gamma|\!)^1;\ \emptyset;\ \overline{s} : \mathtt{end} \vdash \mathbf{0} \triangleright \diamond} \quad (\!|\Gamma|\!)^1;\ \emptyset;\ \emptyset \vdash z_X \triangleright (\widetilde{T}, S^*) \to \diamond}{(\!|\Gamma|\!)^1;\ \emptyset;\ \overline{s} : !\langle (\widetilde{T}, S^*) \to \diamond \rangle; \mathtt{end} \vdash \overline{s}!\langle z_X \rangle.\mathbf{0} \triangleright \diamond} \tag{B.6}$$

We may now derive:

$$\dfrac{\begin{array}{c}(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \cdot s : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end} \vdash z_X\,(\tilde{n}, s) \triangleright \diamond \quad \text{(B.5)} \\ (\!|\Gamma|\!)^1;\ \emptyset;\ \overline{s} : !\langle (\widetilde{T}, S^*) \to \diamond \rangle; \mathtt{end} \vdash \overline{s}!\langle z_X \rangle.\mathbf{0} \triangleright \diamond \quad \text{(B.6)}\end{array}}{(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \cdot s : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end}, \overline{s} : !\langle (\widetilde{T}, S^*) \to \diamond \rangle; \mathtt{end} \vdash z_X\,(\tilde{n}, s) \mid \overline{s}!\langle z_X \rangle.\mathbf{0} \triangleright \diamond}$$
$$(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \vdash (\nu\, s)(z_X\,(\tilde{n}, s) \mid \overline{s}!\langle z_X \rangle.\mathbf{0}) \triangleright \diamond$$

4. Case $P_0 = \mu X.P$. Then we have the following typing in the source language:

$$\dfrac{\Gamma \cdot X : \Delta;\ \emptyset;\ \Delta \vdash P \triangleright \diamond}{\Gamma;\ \emptyset;\ \Delta \vdash \mu X.P \triangleright \diamond}$$

By Fig. 9, we have:

$$[\![\mu X.P]\!]^1_f = (\nu\, s)(\overline{s}!\langle \lambda(\|\tilde{n}\|, y).\ y?(z_X).\llbracket [\![P]\!]^1_{f,\{X \to \tilde{n}\}} \rrbracket_{\emptyset} \rangle.\mathbf{0} \mid s?(z_X).[\![P]\!]^1_{f,\{X \to \tilde{n}\}})$$

We shall show that

$$(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \vdash [\![\mu X.P]\!]^1_f \triangleright \diamond$$

Below we write R to stand for $[\![P]\!]^1_{f,\{X \to \tilde{n}\}}$ and $\tilde{x} = \|\mathtt{fn}(P)\|$ (cf. Definition 5.1). Moreover, we write $\Delta_{\tilde{x}}$ to denote Δ after a renaming with names $\tilde{x}$.

We have two auxiliary derivations:

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1;\ \emptyset;\ \emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!|\Gamma|\!)^1;\ \emptyset;\ \overline{s} : \mathtt{end} \vdash \mathbf{0} \triangleright \diamond} \quad \dfrac{\dfrac{\dfrac{(\!|\Gamma|\!)^1 \cdot z_X : (\widetilde{T}, S^*) \to \diamond;\ \emptyset;\ (\!|\Delta_{\tilde{x}}|\!)^1 \vdash \llbracket R \rrbracket_{\emptyset} \triangleright \diamond}{(\!|\Gamma|\!)^1 \cdot z_X : (\widetilde{T}, S^*) \to \diamond;\ \emptyset;\ (\!|\Delta_{\tilde{x}}|\!)^1 \cdot y : \mathtt{end} \vdash \llbracket R \rrbracket_{\emptyset} \triangleright \diamond}}{(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta_{\tilde{x}}|\!)^1 \cdot y : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end} \vdash y?(z_X).\llbracket R \rrbracket_{\emptyset} \triangleright \diamond}}{(\!|\Gamma|\!)^1;\ \emptyset;\ \emptyset \vdash \lambda(\tilde{x}, y).\ y?(z_X).\llbracket R \rrbracket_{\emptyset} \triangleright (\widetilde{T}, S^*) \to \diamond}}{(\!|\Gamma|\!)^1;\ \emptyset;\ \overline{s} : !\langle (\widetilde{T}, S^*) \to \diamond \rangle; \mathtt{end} \vdash \overline{s}!\langle \lambda(\tilde{x}, y).\ y?(z_X).\llbracket R \rrbracket_{\emptyset} \rangle.\mathbf{0} \triangleright \diamond} \tag{B.7}$$

and

$$\dfrac{\dfrac{(\!|\Gamma|\!)^1 \cdot z_X : (\widetilde{T}, S^*) \to \diamond;\ \emptyset;\ (\!|\Delta_{\tilde{n}}|\!)^1 \vdash R \triangleright \diamond}{(\!|\Gamma|\!)^1 \cdot z_X : (\widetilde{T}, S^*) \to \diamond;\ \emptyset;\ (\!|\Delta_{\tilde{n}}|\!)^1 \cdot s : \mathtt{end} \vdash R \triangleright \diamond}}{(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \cdot s : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end} \vdash s?(z_X).R \triangleright \diamond} \tag{B.8}$$

We then have:

$$\dfrac{\begin{array}{c}(\!|\Gamma|\!)^1;\ \emptyset;\ \overline{s} : !\langle (\widetilde{T}, S^*) \to \diamond \rangle; \mathtt{end} \vdash \overline{s}!\langle \lambda(\tilde{x}, y).\ y?(z_X).\llbracket R \rrbracket_{\emptyset} \rangle.\mathbf{0} \triangleright \diamond \quad \text{(B.7)} \\ (\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \cdot s : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end} \vdash s?(z_X).R \triangleright \diamond \quad \text{(B.8)}\end{array}}{\begin{array}{c}(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \cdot s : ?((\widetilde{T}, S^*) \to \diamond); \mathtt{end}, \overline{s} : !\langle (\widetilde{T}, S^*) \to \diamond \rangle; \mathtt{end} \vdash \\ \overline{s}!\langle \lambda(\tilde{x}, y).\ y?(z_X).\llbracket R \rrbracket_{\emptyset} \rangle.\mathbf{0} \mid s?(z_X).R \triangleright \diamond\end{array}}$$
$$(\!|\Gamma|\!)^1;\ \emptyset;\ (\!|\Delta|\!)^1 \vdash (\nu\, s)(s?(z_X).R \mid \overline{s}!\langle \lambda(\tilde{x}, y).\ y?(z_X).\llbracket R \rrbracket_{\emptyset} \rangle.\mathbf{0}) \triangleright \diamond \qquad \square$$

We repeat the statement in Page 17. We use the mapping on actions $\{\!|\cdot|\!\}^1$ given in Definition 5.3.

Proposition Appendix B.2 *(Operational 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 *such that* $(\!|\Gamma|\!)^1; (\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\ell_2}{\longmapsto} (\!|\Delta'|\!)^1 \vdash R$, *with* $\ell_2 = \{\!|\ell_1|\!\}^1$, *and* $(\!|\Gamma|\!)^1; (\!|\Delta'|\!)^1 \vdash R \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta'|\!)^1 \vdash [\![P_0]\!]^1_f\{^m/_x\}$.

d) *If* $\ell_1 = \tau$ *and* $P \equiv (\nu\,\tilde{m})(n!\langle m\rangle.P_1 \mid n?(x).P_2)$ *and* $P' = (\nu\,\tilde{m})(P_1 \mid P_2\{^m/_x\})$ *then* $\exists R$ *such that* $(\!|\Gamma|\!)^1; (\!|\Delta|\!)^1 \vdash [\![P]\!]^1_f \overset{\tau}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R)$, *and* $(\!|\Gamma|\!)^1; (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid R) \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta|\!)^1 \vdash (\nu\,\tilde{m})([\![P_1]\!]^1_f \mid [\![P_2]\!]^1_f\{^m/_x\})$.

e) *If* $\ell_1 = \tau$ *and* $P \equiv (\nu\,\tilde{m})(n!\langle\lambda y.\, Q\rangle.P_1 \mid n?(x).P_2)$ *and* $P' = (\nu\,\tilde{m})(P_1 \mid P_2\{^{\lambda y.\, Q}/_x\})$ *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* $P \equiv (\nu\,\tilde{m})((\lambda x.\, P_1)\, V)$ *and* $P' = (\nu\,\tilde{m})(P_1\{^V/_x\})$ *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* $\ell_2 \in \{(\nu\,\tilde{m})n!\langle\lambda z.\, z?(x).(xm)\rangle,\ (\nu\,\tilde{m})n!\langle\lambda x.\, R\rangle,\ s \oplus l,\ s\&l\}$ *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* $(\!|\Gamma|\!)^1; (\!|\Delta'|\!)^1 \vdash Q \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta''|\!)^1 \vdash [\![P'\{^m/_z\}]\!]^1_f$

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* $(\!|\Gamma|\!)^1; (\!|\Delta|\!)^1 \vdash Q \overset{\tau_\beta}{\longmapsto}\overset{\tau_s}{\longmapsto}\overset{\tau_\beta}{\longmapsto} (\!|\Delta|\!)^1 \vdash [\![P_1]\!]^1_f \mid [\![P_2\{^m/_x\}]\!]^1_f$

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 1(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 judgement for P is as follows:

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

which, using Definition 5.2, can be expressed as:

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

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

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

from which the thesis follows easily.

2. Subcase 1(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:

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

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

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

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

$$\begin{aligned}&(\!|\Gamma|\!)^1;(\!|\Delta_0|\!)^1\cdot n:?(?((\!|S|\!)^1\multimap\diamond);\mathtt{end}\multimap\diamond);(\!|S_1|\!)^1\vdash n?(x).(\nu\, s)((xs)\mid\overline{s}!\langle\lambda x.\,[\![P']\!]^1_f\rangle.\mathbf{0})\triangleright\diamond\\ \xrightarrow{\{\ell_1\}^1}\ &(\!|\Gamma|\!)^1;(\!|\Delta_0|\!)^1\cdot n:(\!|S_1|\!)^1\cdot m:(\!|S|\!)^1\vdash R\triangleright\diamond\end{aligned}$$

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

$$\begin{aligned}R&\xrightarrow{\tau_\beta}(\nu\, s)(s?(x).(xm)\mid\overline{s}!\langle\lambda x.\,[\![P']\!]^1_f\rangle.\mathbf{0})\\&\xrightarrow{\tau_s}(\lambda x.\,[\![P']\!]^1_f)\,m\mid\mathbf{0}\\&\xrightarrow{\tau_\beta}[\![P']\!]^1_f\{^m/_x\}\end{aligned}$$

and so the thesis follows.

Part (2) - Soundness. We consider two representative cases, the rest is similar or simpler:

1. Subcase 2(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:

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

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

$$\begin{aligned}&(\!|\Gamma|\!)^1;\ (\!|\Delta_0|\!)^1\cdot n:!\langle?((\!|S|\!)^1\multimap\diamond);\mathtt{end}\multimap\diamond\rangle;(\!|S_1|\!)^1\vdash n!\langle\lambda z.\ z?(x).(xm)\rangle.[\![P']\!]^1_f\\ \xrightarrow{\ell_2}\ &(\!|\Gamma|\!)^1;\ (\!|\Delta_0|\!)^1\cdot n:(\!|S_1|\!)^1\vdash[\![P']\!]^1_f\end{aligned}$$

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

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

and thus the thesis follows easily by noticing that $\{\!|n!\langle m\rangle|\!\}^1 = n!\langle\lambda z.\ z?(x).(xm)\rangle$.

2. Subcase 2(b): $P = n?(x).P'$ and $\ell_2 = n?\langle\lambda y.\ y?(x).(xm)\rangle$. Then we have:

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

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

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

Now, in the source term P we can infer the following transition, from which the thesis follows:

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

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

We now present the proof of the full abstraction result (Proposition 5.3 (Page 18)). In the proof, we rely heavily on the (detailed) labelled correspondence given above to define typed bisimulation relations up-to determinacy (Appendix A.1). Proving that these relations indeed satisfy the requirements is immediate for most cases, where we just follow the requirements of the labelled correspondence transitions. The most interesting cases are the output cases, where the analyses should be done up-to the characteristic process.

Proposition Appendix B.3 *(Full abstraction,* HOπ *into* HO*).* $\Gamma; \Delta_1 \vdash P_1 \approx^{\mathrm{H}} \Delta_2 \vdash Q_1$ *if and only if* $(\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \approx^{\mathrm{H}} (\!|\Delta_2|\!)^1 \vdash [\![Q_1]\!]^1_f$.

Proof. For the right-to-left direction we show that the following relation $\Re$:

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

is a higher-order bisimulation (Definition 3.11). Suppose $\Gamma; \Delta_1 \vdash P_1 \overset{\ell}{\mapsto} \Delta_1' \vdash P_2$; we perform a case analysis on the shape of ℓ, using the soundness direction of operational correspondence (cf. Proposition Appendix B.2 (Page 37)). The most interesting case is when $\ell = (\nu\, \tilde{m_1}')n!\langle m_1\rangle$; the other cases are similar or easier.

Given $\Gamma; \Delta_1 \vdash P_1 \overset{(\nu\, \tilde{m_1}')n!\langle m_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$, we have that Proposition Appendix B.2 (Page 37) implies:

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

Now, combining this transition with the definition of $\Re$ we obtain both:

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

and

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

Based on the encoding $[\![\cdot]\!]^1_f$ (cf. Fig. 9), we may rewrite the above equality as follows:

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

We may then observe that:

$$\begin{array}{rlcl} & \Gamma; \Delta_1' & \vdash & (\nu\, \tilde{m_1}')(P_2 \mid t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle m_1\rangle.\mathbf{0})) \\ \Re & \Delta_2' & \vdash & (\nu\, \tilde{m_2}')(Q_2 \mid t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle m_2\rangle.\mathbf{0})) \end{array}$$

which can be rewritten to coincide with the output clause of higher-order bisimilarity (Definition 3.11), as required:

$$\Gamma; \Delta_1' \vdash (\nu\, \tilde{m_1}')(P_2 \mid t \leftrightarrow_{\mathrm{H}} m_1)\ \Re\ \Delta_2' \vdash (\nu\, \tilde{m_2}')(Q_2 \mid t \leftrightarrow_{\mathrm{H}} m_2)$$

This concludes the proof.

For the left-to-right direction, we consider the following relation:

$$\Re = \{([\![P_1]\!]^1_f, [\![Q_1]\!]^1_f) \mid \Gamma; \Delta_1 \vdash P_1 \approx^{\mathrm{H}} \Delta_2 \vdash Q_1\}$$

We show that $\Re \subset \approx^{\mathrm{H}}$. Suppose $(\!|\Gamma|\!)^1; (\!|\Delta_1|\!)^1 \vdash [\![P_1]\!]^1_f \overset{\ell}{\mapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]^1_f$; we perform a case analysis on the shape of ℓ, using the soundness direction of operational correspondence (cf. Proposition Appendix B.2 (Page 37)). We consider three cases:

1. Case: $\ell \notin \{(\nu\, \tilde{m})n!\langle \lambda x.\, P\rangle, n?\langle \lambda x.\, P\rangle\}$. Then, we have that Proposition Appendix B.2 (Page 37) implies $\Gamma; \Delta_1 \vdash P_1 \overset{\ell}{\mapsto} \Delta_1' \vdash P_2$. From this transition and the definition of $\Re$ we infer both:

$$\Gamma; \Delta_2 \vdash Q_1 \overset{\ell}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{B.9}$$

$$\Gamma; \Delta_1' \vdash P_2 \approx^{\mathrm{H}} \Delta_2' \vdash Q_2 \tag{B.10}$$

From (B.9) and Proposition Appendix B.2 (Page 37) we obtain:

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

Furthermore, from (B.10) and the definition of $\Re$ we obtain, as required:

$$(\!|\Gamma|\!)^1; (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]^1_f \;\Re\; (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]^1_f$$

2. Case: $\ell = (\nu\,\tilde{m})n!\langle\lambda x.\, P\rangle$. We distinguish two sub-cases, depending on whether $\lambda x.\, P$ corresponds to the encoding of a name.
   - If $\lambda x.\, P$ does not correspond to the encoding of a name, then by Proposition Appendix B.2 (Page 37) we infer that

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

     implies

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

     and the rest of the argument proceeds as in the previous case.
   - If $\lambda x.\, P$ does correspond to the encoding of a name, then by Proposition Appendix B.2 (Page 37) we infer that

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

     implies

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

     for some m_1. From the latter transition and the definition of $\Re$ we infer both:

     $$\Gamma; \Delta_2 \vdash Q_1 \overset{(\nu\,\tilde{m_2}')n!\langle m_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{B.11}$$

     and

     $$\begin{aligned}&\Gamma; \Delta_1' \vdash (\nu\,\tilde{m_1}')(P_2 \mid t?(x).(\nu\,s)(s?(y).(x\,y) \mid \overline{s}!\langle m_1\rangle.\mathbf{0}))\\ \approx^{\mathrm{H}}\;\; &\Delta_2' \vdash (\nu\,\tilde{m_2}')(Q_2 \mid t?(x).(\nu\,s)(s?(y).(x\,y) \mid \overline{s}!\langle m_2\rangle.\mathbf{0}))\end{aligned} \tag{B.12}$$

     for some m_2. From (B.11) and Proposition Appendix B.2 (Page 37), we obtain:

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

     Furthermore, from (B.12) and the definition of $\Re$ we obtain the following:

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

     which coincides with the output clause of higher-order bisimilarity, as required.
3. Case: $\ell = n?\langle\lambda x.\, P\rangle$. Also here we distinguish whether the received abstraction corresponds to the encoding of a name:
   - If $\lambda x.\, P$ does not correspond to the encoding of a name, then the proof proceeds as in previous cases.
   - If $\lambda x.\, P$ does correspond to the encoding of a name, then by Proposition Appendix B.2 (Page 37) we infer that

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

     implies

     $$\Gamma; \Delta_1 \vdash P_1 \xmapsto{n?\langle m_1\rangle} \Delta_1' \vdash P_2 \tag{B.13}$$

     $$(\!|\Gamma|\!)^1; (\!|\Delta_1''|\!)^1 \vdash R \overset{\tau_\beta}{\longmapsto} \overset{\tau_{\mathrm{s}}}{\longmapsto} (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]^1_f \tag{B.14}$$

     From (B.13) and the definition of $\Re$ we infer:

     $$\Gamma; \Delta_2 \vdash Q_1 \overset{n?\langle m_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2 \tag{B.15}$$

     $$\Gamma; \Delta_1' \vdash P_2 \approx^{\mathrm{H}} \Delta_2' \vdash Q_2 \tag{B.16}$$

     for some m_2. From (B.15) and Proposition Appendix B.2 (Page 37) we obtain:

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

Furthermore, from (B.16) and the definition of $\Re$ we obtain:

$$(\!|\Gamma|\!)^1; (\!|\Delta_1'|\!)^1 \vdash [\![P_2]\!]^1_f \;\Re\; (\!|\Delta_2'|\!)^1 \vdash [\![Q_2]\!]^1_f$$

If we consider result (B.14) we obtain:

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

and then we may show that $\Re$ is a bisimulation up-to $\overset{\tau_s}{\Longmapsto}$, following Lemma Appendix A.1. □

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

In this section we prove Theorem 5.2 (Page 21), which states that the encoding $[\![\cdot]\!]^2$ of $\mathcal{L}_{\mathsf{HO}\pi}$ into $\mathcal{L}_\pi$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated as Proposition 5.4 (Page 19) and proven here as Proposition Appendix B.4 (Page 42).
- Operational Correspondence, stated as Proposition 5.5 (Page 20) and proven here as Proposition Appendix B.5 (Page 44).
- Full Abstraction, stated as Proposition 5.6 (Page 21) and proven here as Proposition Appendix B.6 (Page 46).

Proposition Appendix B.4 *(Type preservation, $\mathsf{HO}\pi$ into π). Let P be an $\mathsf{HO}\pi$ 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 $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$. We consider three representative cases:

1. Case $P = k!\langle\lambda x.\, Q\rangle.P$. Then there are several sub-cases, depending on whether k and x have linear types. We content ourselves by checking the case in which k is a session (linear) name. We then have two possibilities, depending on the typing for $\lambda x.\, Q$.

   (a) The first sub-case concerns a linear typing, and so we have in the source language:

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

   Following Fig. 11, we have $[\![k!\langle\lambda x.\, Q\rangle.P]\!]^2 = (\nu\, a)(u!\langle a\rangle.([\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2))$. By IH we have:

   $$(\!|\Gamma|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \cdot x : (\!|S_1|\!)^2 \vdash [\![Q]\!]^2 \triangleright \diamond$$
   $$(\!|\Gamma|\!)^2; \emptyset; (\!|\Delta_1|\!)^2 \cdot k : (\!|S|\!)^2 \vdash [\![P]\!]^2 \triangleright \diamond$$

   Let $U_1 = ?((\!|S_1|\!)^2); \mathtt{end}$. Also, we write $(\!|\Gamma'|\!)^2$ to stand for $(\!|\Gamma|\!)^2 \cdot a : \langle U_1\rangle$. We first have:

   $$\frac{(\!|\Gamma'|\!)^2; \emptyset; \emptyset \vdash a \triangleright \langle U_1\rangle \qquad \dfrac{\dfrac{\dfrac{}{(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \cdot x : (\!|S_1|\!)^2 \vdash [\![Q]\!]^2 \triangleright \diamond}}{(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \cdot y : \mathtt{end} \cdot x : (\!|S_1|\!)^2 \vdash [\![Q]\!]^2 \triangleright \diamond}}{(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \cdot y : U_1 \vdash y?(x).[\![Q]\!]^2 \triangleright \diamond}}{(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \vdash a?(y).y?(x).[\![Q]\!]^2 \triangleright \diamond} \tag{B.17}$$

   We then have:

   $$\frac{(\!|\Gamma'|\!)^2; \emptyset; \emptyset \vdash a \triangleright \langle U_1\rangle \qquad \dfrac{\begin{array}{r}(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_1|\!)^2 \cdot k : (\!|S|\!)^2 \;\vdash\; [\![P]\!]^2 \triangleright \diamond \\ (\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_2|\!)^2 \;\vdash\; a?(y).y?(x).[\![Q]\!]^2 \triangleright \diamond \quad \text{(B.17)}\end{array}}{\begin{array}{r}(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_1|\!)^2 \cdot (\!|\Delta_2|\!)^2 \cdot k : (\!|S|\!)^2 \vdash \\ [\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2 \triangleright \diamond\end{array}}}{\dfrac{(\!|\Gamma'|\!)^2; \emptyset; (\!|\Delta_1|\!)^2 \cdot (\!|\Delta_2|\!)^2 \cdot k : !\langle\langle U_1\rangle\rangle; (\!|S|\!)^2 \vdash k!\langle a\rangle.([\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2) \triangleright \diamond}{\begin{array}{r}(\!|\Gamma|\!)^2; \emptyset; (\!|\Delta_1|\!)^2 \cdot (\!|\Delta_2|\!)^2 \cdot k : !\langle\langle U_1\rangle\rangle; (\!|S|\!)^2 \vdash \\ (\nu\, a)(k!\langle a\rangle.([\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2)) \triangleright \diamond\end{array}}} \tag{B.18}$$

   which concludes the proof for this sub-case.

   (b) In the second sub-case, $\lambda x.\, Q$ has a shared type, and so $\mathtt{fs}(Q) = \emptyset$. We have the following typing in the source language:

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

Following Fig. 11, we have $[\![k!\langle\lambda x.\,Q\rangle.P]\!]^2=(\nu\,a)(u!\langle a\rangle.([\![P]\!]^2\mid *a?(y).y?(x).[\![Q]\!]^2))$. Recall that by Not. 1, $*P$ is a shorthand notation for $\mu X.(P\mid X)$. By IH we have:

$$(\!|\Gamma|\!)^2;\emptyset;x:(\!|S_1|\!)^2\vdash[\![Q]\!]^2\triangleright\diamond$$
$$(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta|\!)^2\cdot k:(\!|S|\!)^2\vdash[\![P]\!]^2\triangleright\diamond$$

Let $U_1=?((\!|S_1|\!)^2);\mathtt{end}$. We also have:

$$(\!|\Gamma_1|\!)^2=(\!|\Gamma|\!)^2\cdot a:\langle U_1\rangle$$
$$(\!|\Gamma_2|\!)^2=(\!|\Gamma_1|\!)^2\cdot X:\emptyset$$

Also, let $(*)$ and $(**)$ stand for $(\!|\Gamma_2|\!)^2;\emptyset;\emptyset\vdash a\triangleright\langle U_1\rangle$ and $(\!|\Gamma_2|\!)^2;\emptyset;\emptyset\vdash X\triangleright\diamond$, respectively. We first have two auxiliary derivations:

$$\frac{\dfrac{\dfrac{\dfrac{\overline{(\!|\Gamma_2|\!)^2;\emptyset;x:(\!|S_1|\!)^2\vdash[\![Q]\!]^2\triangleright\diamond}}{(\!|\Gamma_2|\!)^2;\emptyset;y:\mathtt{end}\cdot x:(\!|S_1|\!)^2\vdash[\![Q]\!]^2\triangleright\diamond}\quad\overline{(*)}}{(\!|\Gamma_2|\!)^2;\emptyset;y:?((\!|S_1|\!)^2);\mathtt{end}\vdash y?(x).[\![Q]\!]^2\triangleright\diamond}\quad\overline{(**)}}{(\!|\Gamma_2|\!)^2;\emptyset;\emptyset\vdash a?(y).y?(x).[\![Q]\!]^2\triangleright\diamond}}{(\!|\Gamma_2|\!)^2;\emptyset;\emptyset\vdash a?(y).y?(x).[\![Q]\!]^2\mid X\triangleright\diamond}}{(\!|\Gamma_1|\!)^2;\emptyset;\emptyset\vdash\mu X.(a?(y).y?(x).[\![Q]\!]^2\mid X)\triangleright\diamond}\tag{B.19}$$

and

$$\frac{(\!|\Gamma_1|\!)^2;\emptyset;(\!|\Delta|\!)^2\cdot k:(\!|S|\!)^2\vdash[\![P]\!]^2\triangleright\diamond\quad (\!|\Gamma_1|\!)^2;\emptyset;\emptyset\vdash\mu X.(a?(y).y?(x).[\![Q]\!]^2\mid X)\triangleright\diamond\ \text{(B.19)}}{(\!|\Gamma_1|\!)^2;\emptyset;(\!|\Delta|\!)^2\cdot k:(\!|S|\!)^2\vdash[\![P]\!]^2\mid\mu X.(a?(y).y?(x).[\![Q]\!]^2\mid X)\triangleright\diamond}\tag{B.20}$$

We now finally have:

$$\frac{\dfrac{(\!|\Gamma_1|\!)^2;\emptyset;\emptyset\vdash a\triangleright\langle U_1\rangle\quad (\!|\Gamma_1|\!)^2;\emptyset;(\!|\Delta|\!)^2\cdot k:(\!|S|\!)^2\vdash[\![P]\!]^2\mid\mu X.(a?(y).y?(x).[\![Q]\!]^2\mid X)\triangleright\diamond\ \text{(B.20)}}{(\!|\Gamma_1|\!)^2;\emptyset;(\!|\Delta|\!)^2\cdot k:!\langle\langle U_1\rangle\rangle;(\!|S|\!)^2\vdash k!\langle a\rangle.([\![P]\!]^2\mid\mu X.(a?(y).y?(x).[\![Q]\!]^2\mid X))\triangleright\diamond}}{(\!|\Gamma|\!)^2;\emptyset;(\!|\Delta|\!)^2\cdot k:!\langle\langle U_1\rangle\rangle;(\!|S|\!)^2\vdash(\nu\,a)(k!\langle a\rangle.([\![P]\!]^2\mid\mu X.(a?(y).y?(x).[\![Q]\!]^2\mid X)))\triangleright\diamond}$$

This completes the proof for this case.

2. Case $P=k?(x).P$. Here again there are several sub-cases, depending on whether k and x have linear types. We content ourselves by checking the case in which k is a session (linear) name. Then there are two sub-cases: $x:S_1\to\diamond$ and $x:S_1\multimap\diamond$.

(a) In the first case, we have the following typing in the source language:

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

Following Fig. 11, the corresponding typing in the target language is as follows:

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

(b) In the second case, we have the following typing in the source language:

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

The corresponding typing in the target language is as follows:

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

3. Case $P = xk$. Also here we have two sub-cases, depending on whether x has linear or shared type.

(a) In the first sub-case, x is linear and so we have the following source typing:

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

Notice that by Rule (EPROM) we have:

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

Following Fig. 11, we have that $[\![xk]\!]^2 = (\nu\, s)(x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0})$. Let us write $(\!|\Gamma_1|\!)^2$ to stand for $(\!|\Gamma|\!)^2 \cdot x : \langle ?((\!|S_1|\!)^2); \mathtt{end}\rangle$. To derive the corresponding typing in the target language we first need an auxiliary derivation:

$$\frac{\dfrac{(\!|\Gamma_1|\!)^2;\ \emptyset;\ \emptyset \vdash \mathbf{0} \triangleright \diamond}{(\!|\Gamma_1|\!)^2;\ \emptyset;\ \overline{s} : \mathtt{end} \vdash \mathbf{0} \triangleright \diamond} \quad (\!|\Gamma_1|\!)^2;\ \emptyset;\ \{k : (\!|S_1|\!)^2\} \vdash k \triangleright (\!|S_1|\!)^2}{(\!|\Gamma_1|\!)^2;\ \emptyset;\ k : (\!|S_1|\!)^2 \cdot \overline{s} : !\langle (\!|S_1|\!)^2\rangle; \mathtt{end} \vdash \overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond} \tag{B.21}$$

We then have:

$$\dfrac{\dfrac{\begin{array}{c}(\!|\Gamma_1|\!)^2;\ \emptyset;\ \emptyset \vdash x \triangleright \langle ?((\!|S_1|\!)^2); \mathtt{end}\rangle \\ (\!|\Gamma_1|\!)^2;\ \emptyset; k : (\!|S_1|\!)^2 \cdot \overline{s} : !\langle (\!|S_1|\!)^2\rangle; \mathtt{end} \vdash \overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond \quad \text{(B.21)} \\ (\!|\Gamma_1|\!)^2;\ \emptyset;\ \{s : ?((\!|S_1|\!)^2); \mathtt{end}\} \vdash s \triangleright ?((\!|S_1|\!)^2); \mathtt{end}\end{array}}{(\!|\Gamma_1|\!)^2;\ \emptyset;\ k : (\!|S_1|\!)^2 \cdot s : ?((\!|S_1|\!)^2); \mathtt{end} \cdot \overline{s} : !\langle (\!|S_1|\!)^2\rangle; \mathtt{end} \vdash x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0} \triangleright \diamond}}{(\!|\Gamma_1|\!)^2;\ \emptyset;\ k : (\!|S_1|\!)^2 \vdash (\nu\, s)(x!\langle s\rangle.\overline{s}!\langle k\rangle.\mathbf{0}) \triangleright \diamond}$$

which completes the proof for this sub-case.

(b) In the second sub-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. □

We repeat the statement in Page 20. Recall that we use the mapping on actions $\{\!|\cdot|\!\}^2$ given in Definition 5.5.

Proposition Appendix B.5 *(Operational correspondence, HOπ into π). Let P be an HOπ process such that $\Gamma;\ \emptyset;\ \Delta \vdash P \triangleright \diamond$.*

1. *Suppose $\Gamma;\ \Delta \vdash P \overset{\ell_1}{\longmapsto} \Delta' \vdash P'$. Then we have:*

   a) *If $\ell_1 = (\nu\, \tilde{m})n!\langle \lambda x.\ Q\rangle$, then $\exists \Gamma',\ \Delta''$ where either:*

   - $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\ell_1\}^2}{\longmapsto} \Gamma' \cdot (\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \mid *a?(y).y?(x).[\![Q]\!]^2$ *(if $\mathtt{fs}(Q) = \emptyset$)*
   - $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\ell_1\}^2}{\longmapsto} (\!|\Gamma|\!)^2;\ \Delta'' \vdash [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$ *(otherwise)*

   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* $(\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta''|\!)^2 \vdash (\nu\, a)(R \mid *a?(y).y?(x).[\![Q]\!]^2)$ *(if $\mathtt{fs}(Q) = \emptyset$)*
   - $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\{\ell_1\}^2}{\longmapsto} (\!|\Gamma|\!)^2;\ (\!|\Delta''|\!)^2 \vdash R$, *and* $(\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta''|\!)^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$ *(otherwise)*

   c) *If $\ell_1 = \tau$, with $\tau \neq \tau_\beta$ 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 (with $\mathtt{fs}(Q) = \emptyset$);*
   - $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Delta'|\!)^2 \vdash (\nu\, \tilde{m})([\![P_1]\!]^2 \mid (\nu\, s)([\![P_2]\!]^2\{^{\overline{s}}/_x\} \mid s?(y).y?(x).[\![Q]\!]^2))$, *for some P_1, P_2, Q (with $\mathtt{fs}(Q) \neq \emptyset$);*
   - $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \overset{\tau}{\longmapsto} (\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$

d) If $\ell_1 = \tau_\beta$ then $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \stackrel{\tau_{\mathtt{s}}}{\longmapsto} (\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$

e) If $\ell_1 \in \{n \oplus l, n\&l\}$ then

$\exists \ell_2 = \{\!|\ell_1|\!\}^2$ such that $(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \stackrel{\ell_2}{\longmapsto} (\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2$.

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

a) If $\ell_2 = (\nu\, m)n!\langle m\rangle$ then one of the following holds:

- $\exists P'$ such that $P \stackrel{(\nu\, m)n!\langle m\rangle}{\longmapsto} P'$ and $R = [\![P']\!]^2$;
- $\exists Q, P'$ such that $P \stackrel{n!\langle \lambda x.\, Q\rangle}{\longmapsto} P'$ and $R = [\![P']\!]^2 \mid *a?(y).y?(x).[\![Q]\!]^2$ and $\mathtt{fs}(Q) = \emptyset$;
- $\exists Q, P'$ such that $P \stackrel{n!\langle \lambda x.\, Q\rangle}{\longmapsto} P'$ and $R = [\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2$ and $\mathtt{fs}(Q) \neq \emptyset$;

b) If $\ell_2 = n?\langle m\rangle$ then one of the following holds:

- $\exists P'$ such that $P \stackrel{n?\langle m\rangle}{\longmapsto} P'$ and $R = [\![P']\!]^2$;
- $\exists Q, P'$ such that $P \stackrel{n?\langle \lambda x.\, Q\rangle}{\longmapsto} P'$
  and $(\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta'|\!)^2 \vdash (\nu\, a)(R \mid *a?(y).y?(x).[\![Q]\!]^2)$ and $\mathtt{fs}(Q) = \emptyset$;
- $\exists Q, P'$ such that $P \stackrel{n?\langle \lambda x.\, Q\rangle}{\longmapsto} P'$
  and $(\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta'|\!)^2 \vdash (\nu\, s)(R \mid s?(y).y?(x).[\![Q]\!]^2)$ and $\mathtt{fs}(Q) \neq \emptyset$.

c) If $\ell_2 = \tau$ then $\exists P'$ such that $P \stackrel{\tau}{\longmapsto} P'$ and $(\!|\Gamma|\!)^2;\ (\!|\Delta'|\!)^2 \vdash [\![P']\!]^2 \approx^{\mathrm{C}} (\!|\Delta'|\!)^2 \vdash R$.

d) If $\ell_2 \notin \{n!\langle m\rangle, n \oplus l, n\&l\}$ then $\exists \ell_1$ such that $\ell_1 = \{\!|\ell_2|\!\}^2$ and

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

Proof. The proof proceeds by transition induction. We only give details for the proof of Part 1, as Part 2 proceeds straightforwardly. We consider four representative sub-cases:

1. Case 1(a), with $\mathtt{fs}(Q) = \emptyset$. Then $\Gamma; \emptyset; \Delta \vdash P \stackrel{n!\langle \lambda x.\, Q\rangle}{\longmapsto} \Delta' \vdash P'$, and so we infer

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

and from Definition 5.4 we have $\{\!|n!\langle \lambda x.\, Q\rangle|\!\}^{=} (\nu\, a)n!\langle a\rangle$, as required.

2. Case 1(a), with $\mathtt{fs}(Q) \neq \emptyset$. Then we have $P = n!\langle \lambda x.\, Q\rangle.P'$ and

$$[\![P]\!]^2 = (\nu\, s)(n!\langle \overline{s}\rangle.[\![P']\!]^2 \mid s?(y).y?(x).[\![Q]\!]^2)$$

and the argument proceeds as in the previous case.

3. Case 1(b), with $\mathtt{fs}(Q) = \emptyset$. Then $\Gamma; \emptyset; \Delta \vdash P \stackrel{n?\langle \lambda x.\, Q\rangle}{\longmapsto} \Delta' \vdash P'\{^{\lambda x.\, Q}/_x\}$ and so we infer that

$$(\!|\Gamma|\!)^2;\ (\!|\Delta|\!)^2 \vdash [\![P]\!]^2 \stackrel{n?\langle a\rangle}{\longrightarrow} (\!|\Delta''|\!)^2 \vdash R\{^a/_x\}$$

with $\{\!|n?\langle \lambda x.\, Q\rangle|\!\}^2 = n?\langle a\rangle$. It remains to show that

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

which can be proven by structural induction on P'. The most interesting case is when $P' = x\,m$. We then have:

$$[\![x\,m\{^{\lambda x.\, Q}/_x\}]\!]^2 = [\![Q\{^m/_x\}]\!]^2$$

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

The right-hand side process can evolve as follows:

$$\begin{array}{lrcl} & (\!|\Gamma|\!)^2;\ (\!|\Delta''|\!)^2 & \vdash & (\nu\, a)((\nu\, s)(x!\langle s\rangle.\overline{s}!\langle m\rangle.\mathbf{0})\{^a/_x\} \mid *a?(y).y?(x).[\![Q]\!]^2) \\ \stackrel{\tau}{\longmapsto}\stackrel{\tau_{\mathtt{s}}}{\longmapsto} & (\!|\Delta''|\!)^2 & \vdash & (\nu\, a)([\![Q\{^m/_x\}]\!]^2 \mid *a?(y).y?(x).[\![Q]\!]^2) \end{array}$$

which is bisimilar with $[\![Q\{^m/_x\}]\!]^2$ because a is fresh.

An interesting inductive step case is parallel composition, i.e., $P' = P_1 \mid P_2$. We need to show:

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

We know that

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

and so we conclude immediately exploiting the fact that $\approx^{\mathrm{C}}$ is a congruence.

4. Case 1(b), with $\mathtt{fs}(Q) \neq \emptyset$. This case is similar to the previous one. □

Proposition Appendix B.6 *(Full abstraction, from HOπ to π). Let P_1, Q_1 be HOπ processes. $\Gamma; \Delta_1 \vdash P_1 \approx^{\mathrm{C}} \Delta_2 \vdash Q_1$ if and only if $(\!|\Gamma|\!)^2; (\!|\Delta_1|\!)^2 \vdash [\![P_1]\!]^2 \approx^{\mathrm{C}} (\!|\Delta_2|\!)^2 \vdash [\![Q_1]\!]^2$.*

Proof. The proof follows directly from operational correspondence (Proposition Appendix B.5 (Page 44)). The different cases of the proposition are used to define bisimulation relation to prove the right-to-left direction, and a bisimulation up-to determinate transition (Lemma Appendix A.1) to prove the left-to-right direction.

For the right-to-left direction, we show that the following relation:

$$\Re = \{(P, Q) \mid (\!|\Gamma|\!)^2; (\!|\Delta_1|\!)^2 \vdash [\![P]\!]^2 \approx^{\mathrm{C}} (\!|\Delta_2|\!)^2 \vdash [\![Q]\!]^2\}$$

is a characteristic bisimilarity (Definition 3.12). Suppose $\Gamma; \Delta_1 \vdash P_1 \overset{\ell}{\mapsto} \Delta_1' \vdash P_2$; we perform a case analysis on the shape of ℓ, using the soundness direction of operational correspondence (cf. Proposition Appendix B.5 (Page 44)). The most interesting case is when $\ell = n!\langle\lambda x.\, R_1\rangle$; the other cases follow the bisimulation game that is implied by Proposition Appendix B.5 (Page 44).

Given $\Gamma_1; \Delta_1 \vdash P \overset{n!\langle\lambda x.\, R_1\rangle}{\longmapsto} \Delta_1' \vdash P'$, by Proposition Appendix B.5 (Page 44) (Part 1), we infer that:

$$(\!|\Gamma|\!)^2; (\!|\Delta_1|\!)^2 \vdash [\![P]\!]^2 \overset{(\nu\, a)n!\langle a_1 : U\rangle}{\longmapsto} (\!|\Delta_1'|\!)^2 \vdash [\![P']\!]^2 \mid *a_1!\langle y\rangle.y?(x).[\![R_1]\!]^2$$

which implies, from the requirements of $\approx^{\mathrm{C}}$, both

$$(\!|\Gamma|\!)^2; (\!|\Delta_2|\!)^2 \vdash [\![Q]\!]^2 \overset{(\nu\, a)n!\langle a_2 : U\rangle}{\Longmapsto} (\!|\Delta_2'|\!)^2 \vdash [\![Q']\!]^2 \mid *a_2!\langle y\rangle.y?(x).[\![R_2]\!]^2 \tag{B.22}$$

and

$$\begin{array}{rrcl} & (\!|\Gamma|\!)^2; (\!|\Delta_1'|\!)^2 & \vdash & (\nu\, a_1)([\![P']\!]^2 \mid *a_1?(y).y?(x).[\![R_1]\!]^2 \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle a_1\rangle.\mathbf{0})) \\ \approx^{\mathrm{C}} & (\!|\Delta_2'|\!)^2 & \vdash & (\nu\, a_2)([\![Q']\!]^2 \mid *a_2?(y).y?(x).[\![R_2]\!]^2 \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle a_2\rangle.\mathbf{0})) \end{array}$$

Now, from (B.22) and Proposition Appendix B.5 (Page 44) (Part 2), we infer that there exist Q', R_2 such that:

$$\Gamma_2; \Delta_2 \vdash Q \overset{n!\langle\lambda x.\, R_2\rangle}{\Longmapsto} \Delta_2' \vdash Q'$$

By following the (deterministic) transitions from the latter pair of processes we obtain that:

$$\begin{array}{rrcl} & \Gamma; \Delta_1' & \vdash & P' \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle R_1\rangle.\mathbf{0}) \\ \Re & \Delta_2' & \vdash & Q' \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle R_2\rangle.\mathbf{0}) \end{array}$$

This suffices to conclude, because from the definition of $[\![\cdot]\!]^2$ (cf. Fig. 11) we have:

$$[\![P' \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle R_1\rangle.\mathbf{0})]\!]^2 = [\![P']\!]^2 \mid *a_2?(y).y?(x).[\![R_2]\!]^2 \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle a_2\rangle.)\mathbf{0}$$

(and similarly for $Q' \mid t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid s!\langle R_2\rangle.\mathbf{0})$).

For the left-to-right direction, we show that the relation:

$$\Re = \{([\![P]\!]^2, [\![Q]\!]^2) \mid \Gamma; \Delta_1 \vdash P \approx^{\mathrm{C}} \Delta_2 \vdash Q\}$$

is a characteristic bisimulation. Suppose $(\!|\Gamma|\!)^2; (\!|\Delta_1|\!)^2 \vdash [\![P]\!]^2 \overset{\ell}{\mapsto} (\!|\Delta_1'|\!)^2 \vdash R$; we need to exhibit a corresponding move from $[\![Q]\!]^2$. To this end, we perform a case analysis on the shape of ℓ, using Proposition Appendix B.5 (Page 44) (Part 2).

One interesting case is when $\ell = (\nu\, a_1)n!\langle a_1\rangle$ and $P = n!\langle\lambda x.\, R_1\rangle.P'$ with $\mathtt{fs}(R_1) = \emptyset$, for some R_1, P'; the other cases are similar or simpler. Given these assumptions, and considering Fig. 11, the transition from $[\![P]\!]^2$ is as follows:

$$\begin{array}{rrcl} & (\!|\Gamma|\!)^2; (\!|\Delta_1|\!)^2 & \vdash & (\nu\, a_1)(n!\langle a_1\rangle.[\![P']\!]^2 \mid *a_1?(y).y?(x).[\![R_1]\!]^2) \\ \overset{(\nu\, a)n!\langle a_1\rangle}{\longmapsto} & (\!|\Delta_1'|\!)^2 & \vdash & [\![P']\!]^2 \mid *a_1?(y).y?(x).[\![R_1]\!]^2 \end{array}$$

Then, using Proposition Appendix B.5 (Page 44) (Part 2(a)), we may infer a transition from P:

$$\Gamma;\Delta_1 \vdash n!\langle\lambda x.\, R_1\rangle.P' \xmapsto{n!\langle\lambda x.\, R_1\rangle} \Delta_1' \vdash P'$$

In turn, this transition, together with the definition of $\mathfrak{R}$, enable us to infer both:

$$\Gamma;\Delta_2 \vdash Q \xmapsto{n!\langle\lambda x.\, R_2\rangle} \Delta_2' \vdash Q'$$

and

$$\begin{array}{rrcl} & \Gamma;\Delta_1' & \vdash & P' \mid t?(x).(\nu\, s)(s?(y).[\![U]\!]^y \mid s!\langle R_1\rangle.\mathbf{0}) \\ \approx^{\mathbb{C}} & \Delta_2' & \vdash & Q' \mid t?(x).(\nu\, s)(s?(y).[\![U]\!]^y \mid s!\langle R_2\rangle.\mathbf{0}) \end{array}$$

for some R_2. Now, using this transition from Q in combination with Proposition Appendix B.5 (Page 44) (Part 1(a)) we obtain:

$$(\!|\Gamma|\!)^2;(\!|\Delta_2|\!)^2 \vdash [\![Q]\!]^2 \xmapsto{(\nu\, a_2)n!\langle a_2\rangle} \Delta_2' \vdash [\![Q']\!]^2 \mid *a_2?(y).y?(x).[\![R_2]\!]^2$$

From the definition of $\mathfrak{R}$ (and the fact that the pair of mapped processes can observe only deterministic transitions) we may finally obtain:

$$\begin{array}{rrcl} & (\!|\Gamma|\!)^2;(\!|\Delta_1'|\!)^2 & \vdash & [\![P']\!]^2 \mid *a_1?(y).y?(x).[\![R_1]\!]^2 \mid t?(x).(\nu\, s)(s?(y).[\![U]\!]^y \mid s!\langle a_1\rangle.\mathbf{0}) \\ \mathfrak{R} & (\!|\Delta_2'|\!)^2 & \vdash & [\![Q']\!]^2 \mid *a_2?(y).y?(x).[\![R_2]\!]^2 \mid t?(x).(\nu\, s)(s?(y).[\![U]\!]^y \mid s!\langle a_2\rangle.\mathbf{0}) \end{array}$$

as required. This suffices, because

$$[\![P' \mid t?(x).(\nu\, s)(s?(y).[\![U]\!]^y \mid s!\langle R_1\rangle.\mathbf{0})]\!]^2 = [\![P']\!]^2 \mid *a_2?(y).y?(x).[\![R_2]\!]^2 \mid t?(x).(\nu\, s)(s?(y).[\![U]\!]^y \mid s!\langle a_2\rangle.\mathbf{0})$$

(and similarly for Q'.) □

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

In this section we prove Theorem 6.1 (Page 26), which states that the encoding $[\![\cdot]\!]^3$ of $\mathcal{L}_{\mathsf{HO}\pi^+}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated as Proposition 6.1 (Page 25) and proven here as Proposition Appendix B.7 (Page 47).
- Operational Correspondence, stated as Proposition 6.2 (Page 26) and proven here as Proposition Appendix B.8 (Page 48).
- Full Abstraction, stated as Proposition 6.3 (Page 26) and proven here as Proposition Appendix B.9 (Page 49).

Proposition Appendix B.7 *(Type preservation. From $\mathsf{HO}\pi^+$ to $\mathsf{HO}\pi$). Let P be an $\mathsf{HO}\pi^+$ 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 two representative cases:

1. Case $P = u!\langle\lambda x.\, Q\rangle.P'$, with u linear and $\lambda x.\, Q$ with linear type. Then we have the following typing in $\mathsf{HO}\pi^+$:

$$\dfrac{\dfrac{}{\Gamma;\Lambda_1;\Delta_1\cdot u : S \vdash P' \triangleright \diamond} \quad \dfrac{\dfrac{\overline{\Gamma\cdot x : L;\Lambda_2;\Delta_2 \vdash Q \triangleright \diamond} \quad \overline{\Gamma\cdot x : L;\emptyset;\emptyset \vdash x \triangleright L}}{\Gamma;\Lambda_2;\Delta_2 \vdash \lambda x : L.\, Q \triangleright L \multimap \diamond}}{\Gamma;\Lambda_1\cdot\Lambda_2;\Delta_1\cdot\Delta_2\cdot u :!\langle L\multimap\diamond\rangle;S \vdash u!\langle\lambda x.\, Q\rangle.P' \triangleright \diamond}$$

Thus, by IH we have:

$$(\!|\Gamma|\!)^3;(\!|\Lambda_1|\!)^3;(\!|\Delta_1|\!)^3\cdot u : (\!|S|\!)^3 \vdash [\![P']\!]^3 \triangleright \diamond \tag{B.23}$$

$$(\!|\Gamma|\!)^3\cdot x : (\!|L|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3 \vdash [\![Q]\!]^3 \triangleright \diamond \tag{B.24}$$

$$(\!|\Gamma|\!)^3\cdot x : (\!|L|\!)^3;\emptyset;\emptyset \vdash x \triangleright (\!|L|\!)^3 \tag{B.25}$$

Following Fig. 12, the corresponding encoding and typing in $\mathsf{HO}\pi$ is as follows. First an auxiliary derivation:

$$\dfrac{\dfrac{\overline{(\text{B.24})}}{(\!|\Gamma|\!)^3\cdot x : (\!|L|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3\cdot z : \mathtt{end} \vdash [\![Q]\!]^3 \triangleright \diamond} \quad \overline{(\text{B.25})}}{(\!|\Gamma|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3\cdot z :?((\!|L|\!)^3);\mathtt{end} \vdash z?(x).[\![Q]\!]^3 \triangleright \diamond} \tag{B.26}$$

Then we have:

$$\dfrac{\dfrac{}{(\!|\Gamma|\!)^3;\emptyset;z:?((\!|L|\!)^3);\mathtt{end}\vdash z\triangleright?((\!|L|\!)^3);\mathtt{end}}\ (\text{B.26})\quad \dfrac{}{(\!|\Gamma|\!)^3;(\!|\Lambda_2|\!)^3;(\!|\Delta_2|\!)^3\vdash \lambda z.\,z?(x).[\![Q]\!]^3\triangleright(?((\!|L|\!)^3);\mathtt{end})\multimap\diamond}\ (\text{B.23})}{(\!|\Gamma|\!)^3;(\!|\Lambda_1|\!)^3\cdot(\!|\Lambda_2|\!)^3;(\!|\Delta_1|\!)^3\cdot(\!|\Delta_2|\!)^3\cdot u:!\langle?((\!|L|\!)^3);\mathtt{end}\multimap\diamond\rangle;(\!|S|\!)^3\vdash u!\langle\lambda z.\,z?(x).[\![Q]\!]^3\rangle.[\![P']\!]^3\triangleright\diamond}$$

2. Case $P=(\lambda x.\,P)\,(\lambda y.\,Q)$. We may have different possibilities for the types of each abstraction. We consider only one of them, as the rest are similar:

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

Thus, by IH we have:

$$(\!|\Gamma|\!)^3\cdot x:(\!|C\to\diamond|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3\vdash[\![P]\!]^3\triangleright\diamond \tag{B.27}$$

$$(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_1|\!)^3\cdot y:(\!|C|\!)^3\vdash[\![Q]\!]^3\triangleright\diamond \tag{B.28}$$

Following Fig. 12, the corresponding typing in HOπ is as follows. First, we present an auxiliary derivation; recall that $(\!|C\multimap\diamond|\!)^3=(\!|C|\!)^3\multimap\diamond$.

$$\dfrac{\dfrac{}{(\!|\Gamma|\!)^3\cdot x:(\!|C\to\diamond|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3\cdot s:\mathtt{end}\vdash[\![P]\!]^3\triangleright\diamond}\ (\text{B.27})}{(\!|\Gamma|\!)^3;(\!|\Lambda|\!)^3;(\!|\Delta_1|\!)^3\cdot s:?((\!|C\multimap\diamond|\!)^3);\mathtt{end}\vdash s?(x).[\![P]\!]^3\triangleright\diamond} \tag{B.29}$$

We now have:

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3\cdot y:(\!|C|\!)^3\vdash[\![Q]\!]^3\triangleright\diamond}\ (\text{B.28})}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3\vdash\lambda y.[\![Q]\!]^3\triangleright(\!|C\multimap\diamond|\!)^3}}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3\cdot\overline{s}:\mathtt{end}\vdash\lambda y.[\![Q]\!]^3\triangleright(\!|C\multimap\diamond|\!)^3}}{(\!|\Gamma|\!)^3;\emptyset;(\!|\Delta_2|\!)^3\cdot\overline{s}:!\langle(\!|C\multimap\diamond|\!)^3\rangle;\mathtt{end}\vdash\overline{s}!\langle\lambda y.[\![Q]\!]^3\rangle.\mathbf{0}\triangleright\diamond}\ (\text{B.29})$$

□

We repeat the statement in Page 26. Recall that we use the mapping on actions $\{\!|\cdot|\!\}^3$ given in Definition 6.2.

Proposition Appendix B.8 *(Operational correspondence. From HOπ^+ to HOπ). Let $\Gamma;\emptyset;\Delta\vdash P$ be an HOπ^+ process.*

1. $\Gamma;\Delta\vdash P\overset{\ell}{\mapsto}\Delta'\vdash P'$ *implies*

   a) *If $\ell\in\{(\nu\,\tilde{m})n!\langle\lambda x.\,Q\rangle,n?\langle\lambda x.\,Q\rangle\}$ then $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3\vdash[\![P]\!]^3\overset{\ell'}{\mapsto}(\!|\Delta'|\!)^3\vdash[\![P']\!]^3$ 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 $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3\vdash[\![P]\!]^3\overset{\ell}{\mapsto}(\!|\Delta'|\!)^3\vdash[\![P']\!]^3$.*
   c) *If $\ell=\tau_\beta$ then $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3\vdash[\![P]\!]^3\overset{\tau}{\mapsto}\Delta''\vdash R$ 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 $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3\vdash[\![P]\!]^3\overset{\tau}{\mapsto}(\!|\Delta'|\!)^3\vdash[\![P']\!]^3$.*

2. $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3\vdash[\![P]\!]^3\overset{\ell}{\mapsto}(\!|\Delta''|\!)^3\vdash Q$ *implies*

   a) *If $\ell\in\{(\nu\,\tilde{m})n!\langle\lambda x.\,R\rangle,n?\langle\lambda x.\,R\rangle\}$ then $\Gamma;\Delta\vdash P\overset{\ell'}{\mapsto}\Delta'\vdash P'$ 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 $\Gamma;\Delta\vdash P\overset{\ell}{\mapsto}\Delta'\vdash P'$ and $Q\equiv[\![P']\!]^3$.*
   c) *If $\ell=\tau$ then either $\Gamma;\Delta\vdash P\overset{\tau}{\mapsto}\Delta'\vdash P'$ 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. We consider both parts separately, considering the mapping in Fig. 12.

1. The proof of Part 1 proceeds by transition induction. We content ourselves by showing two interesting cases; other cases are similar. Suppose $\Gamma;\Delta\vdash P\overset{\ell}{\mapsto}\Delta'\vdash P'$.

a) Case 1(a): Then $\Gamma;\Delta \vdash n!\langle \lambda x.\, Q\rangle.P \xrightarrow{n!\langle \lambda x.\, Q\rangle} \Delta \vdash P'$. By following the encoding in Fig. 12, we have that

$$\begin{aligned}\llbracket P\rrbracket^3 &= n!\langle \{\!|\lambda x.\, Q\,|\!\}^3\rangle.\llbracket P'\rrbracket^3\\ &= n!\langle \lambda z.\, z?(x).\llbracket Q\rrbracket^3\rangle.\llbracket P'\rrbracket^3\end{aligned}$$

and therefore $(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash n!\langle \lambda z.\, z?(x).\llbracket Q\rrbracket^3\rangle.\llbracket P\rrbracket^3 \xrightarrow{n!\langle \lambda z.\, z?(x).\llbracket Q\rrbracket^3\rangle} \Delta \vdash \llbracket P'\rrbracket^3$, as required.

b) Case 1(c): Then $\Gamma;\Delta \vdash (\lambda x.\, Q_1)\,\lambda y.\, Q_2 \xrightarrow{\tau_\beta} \Delta \vdash Q_1\{^{\lambda y.\, Q_2}/_x\}$. By following the encoding in Fig. 12, we have the following:

$$\begin{aligned}\llbracket P\rrbracket^3 &= (\nu\, s)(s?(x).\llbracket Q_1\rrbracket^3 \mid \overline{s}!\langle \{\!|\lambda y.\, Q_2|\!\}^3\rangle.\mathbf{0})\\ &= (\nu\, s)(s?(x).\llbracket Q_1\rrbracket^3 \mid \overline{s}!\langle \lambda z.\, z?(y).\llbracket Q_2\rrbracket^3\rangle.\mathbf{0})\end{aligned}$$

and therefore

$$(\!|\Gamma|\!)^3;(\!|\Delta|\!)^3 \vdash (\nu\, s)(s?(x).\llbracket Q_1\rrbracket^3 \mid \overline{s}!\langle \lambda z.\, z?(y).\llbracket Q_2\rrbracket^3\rangle.\mathbf{0}) \xrightarrow{\tau_s} (\!|\Delta'|\!)^3 \vdash \llbracket Q_1\rrbracket^3\{^{\lambda z.\, z?(y).\llbracket Q_2\rrbracket^3}/_x\}$$

We are left to show that $\llbracket Q_1\{^{\lambda y.\, Q_2}/_x\}\rrbracket^3$ and $\llbracket Q_1\rrbracket^3\{^{\lambda z.\, z?(y).\llbracket Q_2\rrbracket^3}/_x\}$ are related by $\approx^{\mathrm{H}}$. This follows easily from the structure of the encoding $\llbracket\cdot\rrbracket^3$, which mimics higher-order applications using deterministic transitions only.

2. The proof of Part 2 also proceeds by transition induction. All cases are easy: they are similar to those described for Part 1 or follow directly from the encoding in Fig. 12. □

Proposition Appendix B.9 *(Full abstraction. From* $\mathsf{HO}\pi^+$ *to* $\mathsf{HO}\pi$*). Let* P *and* 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* $(\!|\Gamma|\!)^3;(\!|\Delta_1|\!)^3 \vdash \llbracket P\rrbracket^3 \approx^{\mathrm{H}} (\!|\Delta_2|\!)^3 \vdash \llbracket Q\rrbracket^3$*.*

Proof (Sketch). The right-to-left direction is proven by showing that the relation

$$\mathfrak{R}_1 = \{(P, Q) \mid (\!|\Gamma|\!)^3;(\!|\Delta_1|\!)^3 \vdash \llbracket P\rrbracket^3 \approx^{\mathrm{H}} (\!|\Delta_2|\!)^3 \vdash \llbracket Q\rrbracket^3\}$$

is a higher-order bisimulation, following Part 2 of Proposition Appendix B.8 (Page 48) for subcases (a) and (b). In subcase (c) we use Proposition 3.1 (Page 10). Similarly, the left-to-right direction is proven by showing that the relation:

$$\mathfrak{R}_2 = \{(\llbracket P\rrbracket^3, \llbracket Q\rrbracket^3) \mid \Gamma;\Delta_1 \vdash P \approx^{\mathrm{H}} \Delta_2 \vdash Q\}$$

is a higher-order bisimulation up to deterministic transitions by following Part 1 of Proposition Appendix B.8 (Page 48). The proof is straightforward for subcases (a), (b), and (d). In subcase (c) we use Lemma Appendix A.1. □

B.4. Properties for encoding $\mathcal{L}_{\mathsf{HO}\tilde{\pi}}$ *into* $\mathcal{L}_{\mathsf{HO}\pi}$

In this section we prove Theorem 6.2 (Page 29), which states that the encoding $\llbracket\cdot\rrbracket^4$ of $\mathcal{L}_{\mathsf{HO}\tilde{\pi}}$ into $\mathcal{L}_{\mathsf{HO}\pi}$ is precise. A precise encoding requires to prove three independent results:

- Type preservation, stated as Proposition 6.4 (Page 27) and proven here as Proposition Appendix B.10 (Page 49).
- Operational Correspondence, stated as Proposition 6.5 (Page 28) and proven here as Proposition Appendix B.11 (Page 50).
- Full Abstraction, stated as Proposition 6.6 (Page 29) and proven here as Proposition Appendix B.12 (Page 52).

Proposition Appendix B.10 *(Type preservation. From* $\mathsf{HO}\tilde{\pi}$ *to* $\mathsf{HO}\pi$*). Let* P *be an* $\mathsf{HO}\tilde{\pi}$ *process. If* $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$ *then* $(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta|\!)^4 \vdash \llbracket P\rrbracket^4 \triangleright \diamond$*.*

Proof. By induction on the inference $\Gamma;\emptyset;\Delta \vdash P \triangleright \diamond$. We examine two representative cases, using dyadic 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 $\llbracket P\rrbracket^4$. By IH we have both:

$$(\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_1|\!)^4\cdot n:(\!|S|\!)^4 \vdash \llbracket P'\rrbracket^4 \triangleright \diamond \qquad (\!|\Gamma|\!)^4;\emptyset;(\!|\Delta_2|\!)^4\cdot x_1:(\!|C_1|\!)^4\cdot x_2:(\!|C_2|\!)^4 \vdash \llbracket Q\rrbracket^4 \triangleright \diamond$$

Let $L = (C_1, C_2) \multimap \diamond$. By Fig. 13 we have $(\!|L|\!)^4 = \big(?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathtt{end}\big) \multimap \diamond$ and $[\![P]\!]^4 = n!\langle \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4\rangle.[\![P']\!]^4$. We first infer the following auxiliary typing derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_2|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot x_2 : (\!|C_2|\!)^4 \vdash [\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_2|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot x_2 : (\!|C_2|\!)^4 \cdot z : \mathtt{end} \vdash [\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_2|\!)^4 \cdot x_1 : (\!|C_1|\!)^4 \cdot z : ?((\!|C_2|\!)^4); \mathtt{end} \vdash z?(x_2).[\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_2|\!)^4 \cdot z : ?((\!|C_1|\!)^4); ?((\!|C_2|\!)^4); \mathtt{end} \vdash z?(x_1).z?(x_2).[\![Q]\!]^4 \triangleright \diamond}}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_2|\!)^4 \vdash \lambda z.\, z?(x_1).z?(x_2).[\![Q]\!]^4 \triangleright ((\!|C_1|\!)^4, (\!|C_2|\!)^4) \multimap \diamond} \tag{B.30}$$

Now we have:

$$\dfrac{\dfrac{}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_1|\!)^4 \cdot k : (\!|S|\!)^4 \vdash [\![P']\!]^4 \triangleright \diamond}\ (\text{B.30})}{(\!|\Gamma|\!)^4; \emptyset; (\!|\Delta_1|\!)^4 \cdot (\!|\Delta_2|\!)^4 \cdot n :!\langle (\!|L|\!)^4\rangle; (\!|S|\!)^4 \vdash [\![P]\!]^4 \triangleright \diamond}$$

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 then 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. 13, 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:

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

We repeat the statement in Page 28. Recall that we use the mapping on actions $\{\!|\cdot|\!\}^4$ given in Definition 6.4.

Proposition Appendix B.11 *(Operational correspondence. From* $\mathsf{HO}\tilde{\pi}$ *to* $\mathsf{HO}\pi$*). Let* $\Gamma; \emptyset; \Delta \vdash P$ *be an* $\mathsf{HO}\tilde{\pi}$ *process.*

1. $\Gamma; \Delta \vdash P \overset{\ell}{\mapsto} \Delta' \vdash P'$ *implies*

   a) *If* $\ell = (\nu\, \tilde{m}')n!\langle \tilde{m}\rangle$ *then* $(\!|\Gamma|\!)^4; (\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell_1}{\mapsto} \ldots \overset{\ell_k}{\mapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ *with* $\{\!|\ell|\!\}^4 = \ell_1, \cdots, \ell_k$.

   b) *If* $\ell = n?\langle \tilde{m}\rangle$ *then* $(\!|\Gamma|\!)^4; (\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell_1}{\mapsto} \ldots \overset{\ell_k}{\mapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ *with* $\{\!|\ell|\!\}^4 = \ell_1, \cdots, \ell_k$.

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

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

   e) *If* $\ell = \tau_\beta$ *then* $(\!|\Gamma|\!)^4; (\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\tau_\beta}{\longmapsto} \overset{\tau_s}{\longmapsto} \ldots \overset{\tau_s}{\longmapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ *with* $\{\!|\ell|\!\}^4 = \tau_\beta, \underbrace{\tau_s, \cdots, \tau_s}_{k}$.

   f) *If* $\ell = \tau$ *then* $(\!|\Gamma|\!)^4; (\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\tau}{\mapsto} \ldots \overset{\tau}{\mapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ *with* $\{\!|\ell|\!\}^4 = \underbrace{\tau, \cdots, \tau}_{k}$.

2. $(\!|\Gamma|\!)^4; (\!|\Delta|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\mapsto} (\!|\Delta_1|\!)^4 \vdash P_1$ *implies*

   a) *If* $\ell \in \{n?\langle m\rangle, n!\langle m\rangle, (\nu\, m)n!\langle m\rangle\}$ *then* $\Gamma; \Delta \vdash P \overset{\ell'}{\mapsto} \Delta' \vdash P'$ *and* $(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash P_1 \overset{\ell_2}{\mapsto} \ldots \overset{\ell_k}{\mapsto} (\!|\Delta'|\!)^4 \vdash [\![P']\!]^4$ *with* $\{\!|\ell'|\!\}^4 = \ell_1, \cdots, \ell_k$ *and* $\ell = \ell_1$.

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

   c) *If* $\ell \in \{n \oplus l, n\&l\}$ *then* $\Gamma; \Delta \vdash P \overset{\ell}{\mapsto} \Delta' \vdash P'$ *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* $\{\!|\ell|\!\}^4 = \tau_\beta, \underbrace{\tau_s, \cdots, \tau_s}_{k}$.

   e) *If* $\ell = \tau$ *and* $\ell \neq \tau_\beta$ *then* $\Gamma; \Delta \vdash P \overset{\tau}{\mapsto} \Delta' \vdash P'$ *and* $(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash P_1 \overset{\tau}{\mapsto} \ldots \overset{\tau}{\mapsto} (\!|\Delta'|\!)^4 \vdash (\!|P'|\!)^4$ *with* $\{\!|\ell|\!\}^4 = \underbrace{\tau, \cdots, \tau}_{k}$.

Proof. The proof of both parts is by transition induction, following the mapping defined in Fig. 13. We consider four representative cases, using dyadic communication:

1. 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 judgement 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. 13, 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.

2. 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 $\Gamma, S, C_1, C_2, \Delta_0, \Delta_1$, 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 judgement 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. 13, can be equivalently expressed as:

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

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

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

which concludes the proof for this case.

3. 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:

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

Observe that 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:

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

which concludes the proof for this case.

4. 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:

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

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:

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

which concludes the proof for this case. □

Proposition Appendix B.12 *(Full abstraction. From* $\mathsf{HO}\tilde{\pi}$ *to* $\mathsf{HO}\pi$*). Let* P, Q *be* $\mathsf{HO}\pi^+$ *process with* $\Gamma;\, \emptyset;\, \Delta_1 \vdash P \triangleright \diamond$ *and* $\Gamma;\, \emptyset;\, \Delta_2 \vdash Q \triangleright \diamond$.

Then $\Gamma;\, \Delta_1 \vdash P \approx^{\mathrm{H}} \Delta_2 \vdash Q$ *if and only if* $(\!|\Gamma|\!)^4;\, (\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \approx^{\mathrm{H}} (\!|\Delta_2|\!)^4 \vdash [\![Q\,]\!]^4$.

Proof. The proof is coinductive, and follows as a consequence of Proposition Appendix B.11 (Page 50).

The right-to-left direction follows by showing that the relation

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

is a higher-order bisimulation, by following Part 2 of Proposition Appendix B.11 (Page 50). Suppose P makes a transition with label ℓ; we must exhibit a matching move from Q. We illustrate four representative cases:

1. If $\ell \in \{n?\langle m\rangle, n!\langle m\rangle, (\nu\, m)n!\langle m\rangle\}$ then $(\!|\Gamma|\!)^4;\, (\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\mapsto} (\!|\Delta_1'|\!)^4 \vdash P_1$ implies

$$(\!|\Gamma|\!)^4;\, (\!|\Delta_2|\!)^4 \vdash [\![Q\,]\!]^4 \overset{\ell}{\Longmapsto} (\!|\Delta_2'|\!)^4 \vdash Q_1$$

From Part 2(a) of Proposition Appendix B.11 (Page 50) we conclude that

$$\Gamma;\, \Delta_1 \vdash P \overset{\ell}{\mapsto} \Delta_1'' \vdash P'$$

and

$$(\!|\Gamma|\!)^4;\, (\!|\Delta_1'|\!)^4 \vdash P_1 \overset{\ell_2}{\longmapsto} \ldots \overset{\ell_n}{\longmapsto} (\!|\Delta_1''|\!)^4 \vdash [\![P']\!]^4$$

with $\{\!|\ell|\!\}^4 = \{\ell_1, \ldots, \ell_n\}$. Moreover, $\Gamma;\, \Delta_2 \vdash Q \overset{\ell}{\mapsto} \Delta_2'' \vdash Q'$ and

$$(\!|\Gamma|\!)^4;\, (\!|\Delta_2'|\!)^4 \vdash Q_1 \overset{\ell_2}{\Longmapsto} \ldots \overset{\ell_n}{\Longmapsto} (\!|\Delta_2''|\!)^4 \vdash [\![Q'\,]\!]^4$$

If we follow the bisimulation game we conclude that

$$(\!|\Gamma|\!)^4;\, (\!|\Delta_1''|\!)^4 \vdash [\![P']\!]^4 \approx^{\mathrm{H}} (\!|\Delta_2''|\!)^4 \vdash [\![Q'\,]\!]^4$$

and

$$\Gamma;\, \Delta_1'' \vdash P' \;\Re\; \Delta_2'' \vdash Q'$$

as required.

2. If $\ell \in \{(\nu\,\tilde{m})n!\langle\lambda x.\,R\rangle, n?\langle\lambda x.\,R\rangle\}$ then $(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\mapsto} (\!|\Delta_1'|\!)^4 \vdash P_1$ implies both

$$(\!|\Gamma|\!)^4; (\!|\Delta_2|\!)^4 \vdash [\![Q]\!]^4 \overset{\ell}{\Longmapsto} (\!|\Delta_2'|\!)^4 \vdash Q_1$$

and

$$(\!|\Gamma|\!)^4; (\!|\Delta_1'|\!)^4 \vdash P_1 \mid C \approx^{\mathrm{H}} (\!|\Delta_2'|\!)^4 \vdash Q_1 \mid C$$

with C corresponding to the characteristic process if ℓ is an output action and $C = \mathbf{0}$ otherwise. From Part 2(b) of Proposition Appendix B.11 (Page 50) we conclude that

$$\Gamma; \Delta_1 \vdash P \overset{\ell'}{\mapsto} \Delta_1' \vdash P'$$

with $\{\!|\ell'|\!\}^4 = \ell$ and $P_1 \equiv [\![P']\!]^4$ and $\Gamma; \Delta_2 \vdash Q \overset{\ell'}{\mapsto} \Delta_2' \vdash Q'$ and $P_1 \equiv [\![P']\!]^4$ and

$$(\!|\Gamma|\!)^4; (\!|\Delta_1'|\!)^4 \vdash [\![P' \mid C]\!]^4 \approx^{\mathrm{H}} (\!|\Delta_2'|\!)^4 \vdash [\![Q' \mid C]\!]^4$$

because the characteristic trigger in the case where $\ell = n!\langle\lambda x.\,R\rangle$ remains the same for $\{\!|\ell|\!\}^4$.

3. If $\ell \in \{n \oplus l, n\&l\}$ then $(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \overset{\ell}{\mapsto} (\!|\Delta_1'|\!)^4 \vdash P_1$ implies

$$(\!|\Gamma|\!)^4; (\!|\Delta_2|\!)^4 \vdash [\![Q]\!]^4 \overset{\ell}{\Longmapsto} (\!|\Delta_2'|\!)^4 \vdash Q_1$$

From Part 2(c) of Proposition Appendix B.11 (Page 50) we conclude that $\Gamma; \Delta_1 \vdash P \overset{\ell}{\mapsto} \Delta_1' \vdash P'$ with $P_1 \equiv [\![P']\!]^4$ and

$$\Gamma; \Delta_2 \vdash Q \overset{\ell}{\mapsto} \Delta_2' \vdash Q'$$

with $Q_1 \equiv [\![Q']\!]^4$, which concludes the case.

4. The cases for $\ell = \tau$ are similar and correspond to Parts 2(d), 2(e) of Proposition Appendix B.11 (Page 50).

The left-to-right direction follows by showing that the relation:

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

is a higher-order bisimulation up to deterministic transitions, by following Part 1 of Proposition Appendix B.11 (Page 50). Suppose $[\![P]\!]^4$ makes a transition with label ℓ; we should exhibit a matching move from $[\![Q]\!]^4$. We consider six cases:

1. If $\ell = (\nu\,\tilde{m}')n!\langle\tilde{m}\rangle$ then $\Gamma; \Delta_1 \vdash P \overset{\ell}{\mapsto} \Delta_1' \vdash P'$ implies $\Gamma; \Delta_2 \vdash Q \overset{\ell}{\Longmapsto} \Delta_2' \vdash Q'$ and

$$\Gamma; \Delta_1' \vdash P' \mid C \approx^{\mathrm{H}} \Delta_2' \vdash Q' \mid C$$

with C corresponding to the trigger process. Furthermore, from Part 1 (a) of Proposition Appendix B.11 (Page 50) we have that

$$(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \overset{\ell_1}{\longmapsto} \ldots \overset{\ell_n}{\longmapsto} (\!|\Delta_1'|\!)^4 \vdash [\![P']\!]^4$$

with $\{\!|\ell|\!\}^4 = \{\ell_1, \ldots, \ell_n\}$ and $(\!|\Gamma|\!)^4; (\!|\Delta_2|\!)^4 \vdash [\![Q]\!]^4 \overset{\ell_1}{\Longmapsto} \ldots \overset{\ell_n}{\Longmapsto} (\!|\Delta_2'|\!)^4 \vdash [\![Q']\!]^4$ and

$$(\!|\Gamma|\!)^4; (\!|\Delta_1'|\!)^4 \vdash [\![P' \mid C_1 \mid C_2]\!]^4 \approx^{\mathrm{H}} (\!|\Delta_2'|\!)^4 \vdash [\![Q' \mid C_1 \mid C_2]\!]^4$$

because the characteristic triggers remain the same for $\{\!|\ell|\!\}^4$.

2. If $\ell = n?\langle\tilde{m}\rangle$ then $\Gamma; \Delta_1 \vdash P \overset{\ell}{\mapsto} \Delta_1' \vdash P'$ implies $\Gamma; \Delta_2 \vdash Q \overset{\ell}{\Longmapsto} \Delta_2' \vdash Q'$ and

$$\Gamma; \Delta_1' \vdash P' \approx^{\mathrm{H}} \Delta_2' \vdash Q'$$

Furthermore, from Part 1 (b) of Proposition Appendix B.11 (Page 50) we have that

$$(\!|\Gamma|\!)^4; (\!|\Delta_1|\!)^4 \vdash [\![P]\!]^4 \overset{\ell_1}{\longmapsto} \ldots \overset{\ell_n}{\longmapsto} (\!|\Delta_1'|\!)^4 \vdash [\![P']\!]^4$$

with $\{\!|\ell|\!\}^4 = \{\ell_1, \ldots, \ell_n\}$ and $(\!|\Gamma|\!)^4; (\!|\Delta_2|\!)^4 \vdash [\![Q]\!]^4 \overset{\ell_1}{\Longmapsto} \ldots \overset{\ell_n}{\Longmapsto} (\!|\Delta_2'|\!)^4 \vdash [\![Q']\!]^4$, as required.

3. The case for $\ell = (\nu\,\tilde{m})n!\langle\lambda\tilde{x}.\,R\rangle$ is similar to the first case.
4. The case for $\ell = n?\langle\lambda\tilde{x}.\,R\rangle$ is similar to the second case.
5. The case for $\ell \in \{n \oplus l, n\&l\}$ is similar to the second case.
6. The case for $\ell = \tau$ is similar to the second case. □

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002

References

[1] Giovanni Bernardi, Ornela Dardha, Simon J. Gay, Dimitrios Kouzapas, On duality relations for session types, in: Prof. of TGC, in: LNCS, vol. 8902, Springer, 2014, pp. 51–66.
[2] Mikkel Bundgaard, Thomas T. Hildebrandt, Jens Chr. Godskesen, A cps encoding of name-passing in higher-order mobile embedded resources, Theor. Comput. Sci. 356 (3) (2006) 422–439.
[3] Martin Berger, Kohei Honda, Nobuko Yoshida, Sequentiality and the π-calculus, in: Proc. TLCA'01, in: LNCS, vol. 2044, 2001, pp. 29–45.
[4] Viviana Bono, Luca Padovani, Typing copyless message passing, LMCS 8 (1) (2012).
[5] Ornela Dardha, Elena Giachino, Davide Sangiorgi, Session types revisited, in: Proc. of PPDP'12, ACM, 2012, pp. 139–150.
[6] Romain Demangeon, Kohei Honda, Full abstraction in a subtyped pi-calculus with linear types, in: CONCUR, in: LNCS, vol. 6901, Springer, 2011, pp. 280–296.
[7] Yuxi Fu, Hao Lu, On the expressiveness of interaction, Theor. Comput. Sci. 411 (11–13) (2010) 1387–1451.
[8] Yuxi Fu, Variations on mobile processes, Theor. Comput. Sci. 221 (1–2) (1999) 327–368.
[9] Daniele Gorla, A taxonomy of process calculi for distribution and mobility, Distrib. Comput. 23 (4) (2010) 273–299.
[10] Daniele Gorla, Towards a unified approach to encodability and separation results for process calculi, Inf. Comput. 208 (9) (2010) 1031–1053.
[11] Simon J. Gay, Vasco Thudichum Vasconcelos, Linear type theory for asynchronous session types, J. Funct. Program. 20 (1) (2010) 19–50.
[12] Kohei Honda, Vasco T. Vasconcelos, Makoto Kubo, Language primitives and type disciplines for structured communication-based programming, in: ESOP'98, in: LNCS, vol. 1381, Springer, 1998, pp. 22–138.
[13] Kohei Honda, Nobuko Yoshida, On reduction-based process semantics, TCS 151 (2) (1995) 437–486.
[14] Naoki Kobayashi, Benjamin C. Pierce, David N. Turner, Linearity and the Pi-calculus, TOPLAS 21 (5) (September 1999) 914–947.
[15] Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida, Characteristic bisimulation for higher-order session processes, in: CONCUR 2015, in: LIPIcs, vol. 42, Dagstuhl, Germany, 2015, pp. 398–411.
[16] Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida, On the relative expressiveness of higher-order session processes, in: Peter Thiemann (Ed.), Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings, in: Lecture Notes in Computer Science, vol. 9632, Springer, 2016, pp. 446–475.
[17] Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida, Characteristic bisimulation for higher-order session processes, Acta Inform. 54 (3) (2017) 271–341.
[18] Dimitrios Kouzapas, Nobuko Yoshida, Globally governed session semantics, LMCS 10 (4) (2014).
[19] Dimitrios Kouzapas, Nobuko Yoshida, Raymond Hu, Kohei Honda, On asynchronous eventful session semantics, MSCS (2015).
[20] Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, Alan Schmitt, On the expressiveness of polyadic and synchronous communication in higher-order process calculi, in: ICALP, vol. 6199, 2010, pp. 442–453.
[21] Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, Alan Schmitt, On the expressiveness and decidability of higher-order process calculi, Inf. Comput. 209 (2) (2011) 198–226.
[22] Christopher League, Zhong Shao, Valery Trifonov, Type-preserving compilation of Featherweight Java, ACM Trans. Program. Lang. Syst. 24 (2) (2002) 112–152.
[23] Robin Milner, The Polyadic pi-Calculus: a Tutorial, Technical report, Technical Report ECS-LFCS-91-180, University of Edinburgh, 1991.
[24] L. Gregory Meredith, Matthias Radestock, A reflective higher-order calculus, Electron. Notes Theor. Comput. Sci. 141 (5) (2005) 49–67.
[25] Robin Milner, Davide Sangiorgi, Barbed bisimulation, in: W. Kuich (Ed.), 19th ICALP, in: LNCS, vol. 623, Springer, 1992, pp. 685–695.
[26] J. Gregory Morrisett, David Walker, Karl Crary, Neal Glew, From system F to typed assembly language, ACM Trans. Program. Lang. Syst. 21 (3) (1999) 527–568.
[27] Dimitris Mostrous, Nobuko Yoshida, Two session typing systems for higher-order mobile processes, in: TLCA, in: LNCS, vol. 4583, Springer, 2007, pp. 321–335.
[28] Dimitris Mostrous, Nobuko Yoshida, Session typing and asynchronous subtying for higher-order π-calculus, Inf. Comput. 241 (2015) 227–263.
[29] Uwe Nestmann, What is a "good" encoding of guarded choice?, Inf. Comput. 156 (1–2) (2000) 287–319.
[30] Dominic Orchard, Nobuko Yoshida, Effects as sessions, sessions as effects, in: POPL 2016, ACM, 2016.
[31] Palamidessi Catuscia, Comparing the expressive power of the synchronous and asynchronous pi-calculi, MSCS 13 (5) (2003) 685–719.
[32] Joachim Parrow, Expressiveness of process algebras, Electron. Notes Theor. Comput. Sci. 209 (2008) 173–186.
[33] Jorge A. Pérez, Higher-Order Concurrency: Expressiveness and Decidability Results, PhD thesis, University of Bologna, 2010.
[34] Kirstin Peters, Uwe Nestmann, Ursula Goltz, On distributability in process calculi, in: Proc. of ESOP 2013, in: LNCS, vol. 7792, Springer, 2013, pp. 310–329.
[35] Catuscia Palamidessi, Vijay A. Saraswat, Frank D. Valencia, Björn Victor, On the expressiveness of linearity vs persistence in the asychronous pi-calculus, in: Proc. of LICS 2006, 2006, pp. 59–68.
[36] Kirstin Peters, Rob J. van Glabbeek, Analysing and comparing encodability criteria, in: Proc. of EXPRESS/SOS 2015, in: EPTCS, vol. 190, 2015, pp. 46–60.
[37] Zhong Shao, Andrew W. Appel, A type-based compiler for standard ML, in: Proc. of PLDI'95, ACM, 1995, pp. 116–129.
[38] D. Sangiorgi, The lazy lambda calculus in a concurrency scenario, in: 7th LICS Conf., IEEE Computer Society Press, 1992, pp. 102–109.
[39] Davide Sangiorgi, Expressing Mobility in Process Algebras: First-Order and Higher Order Paradigms, PhD thesis, University of Edinburgh, 1992.
[40] D. Sangiorgi, π-calculus, internal mobility and agent-passing calculi, TCS 167 (2) (1996) 235–274.
[41] Davide Sangiorgi, Asynchronous process calculi: the first- and higher-order paradigms, Theor. Comput. Sci. 253 (2) (2001) 311–350.
[42] Davide Sangiorgi, David Walker, The π-Calculus: A Theory of Mobile Processes, Cambridge University Press, 2001.
[43] Davide Sangiorgi, Xian Xu, Trees From Functions as Processes, Proc. of CONCUR 2014, vol. 8704, Springer, 2014, pp. 78–92.
[44] Bent Thomsen, Calculi for Higher Order Communicating Systems, PhD thesis, Dept. of Comp. Sci., Imperial College, 1990.
[45] Bent Thomsen, Plain CHOCS: a second generation calculus for higher order processes, Acta Inform. 30 (1) (1993) 1–59.
[46] Rob J. van Glabbeek, Musings on encodings and expressiveness, in: Proc. of EXPRESS/SOS 2012, in: EPTCS, vol. 89, 2012, pp. 81–98.
[47] Xian Xu, Distinguishing and relating higher-order and first-order processes by expressiveness, Acta Inform. 49 (7–8) (2012) 445–484.
[48] Xian Xu, Qiang Yin, Huan Long, On the expressiveness of parameterization in process-passing, in: WS-FM, in: LNCS, vol. 8379, Springer, 2014, pp. 147–167.
[49] Xian Xu, Qiang Yin, Huan Long, On the computation power of name parameterization in higher-order processes, in: Proc. of ICE 2015, in: EPTCS, vol. 189, 2015, pp. 114–127.
[50] Nobuko Yoshida, Martin Berger, Kohei Honda, Strong normalisation in the pi-calculus, Inf. Comput. 191 (2) (2004) 145–202.
[51] Nobuko Yoshida, Graph types for monadic mobile processes, in: FSTTCS, in: LNCS, vol. 1180, Springer, 1996, pp. 371–386.

Please cite this article in press as: D. Kouzapas et al., On the relative expressiveness of higher-order session processes, Inf. Comput. (2019), https://doi.org/10.1016/j.ic.2019.06.002