


# Characteristic bisimulation for higher-order session processes

Dimitrios Kouzapas<sup>1</sup> · Jorge A. Pérez<sup>2</sup>  · Nobuko Yoshida<sup>3</sup>

Received: 27 November 2015 / Accepted: 28 November 2016 / Published online: 24 December 2016  
© The Author(s) 2017. This article is published with open access at Springerlink.com

**Abstract** For higher-order (process) languages, characterising contextual equivalence is a long-standing issue. In the setting of a higher-order  $\pi$ -calculus with *session types*, we develop *characteristic bisimilarity*, a typed bisimilarity which fully characterises contextual equivalence. To our knowledge, ours is the first characterisation of its kind. Using simple values inhabiting (session) types, our approach distinguishes from untyped methods for characterising contextual equivalence in higher-order processes: we show that observing as inputs only a precise finite set of higher-order values suffices to reason about higher-order session processes. We demonstrate how characteristic bisimilarity can be used to justify optimisations in session protocols with mobile code communication.

## 1 Introduction

*Context* In *higher-order process calculi* communicated values may contain processes. Higher-order concurrency has received significant attention from untyped and typed perspectives; see, e.g., [13, 15, 20, 26, 30, 33]. In this work, we consider  $\text{HO}\pi$ , a higher-order process calculus with *session communication*: it combines functional constructs (abstractions/applications, as in the call-by-value  $\lambda$ -calculus) and concurrent primitives (synchronisation on shared names, communication on linear names, recursion). By amalgamating functional and concurrent constructs,  $\text{HO}\pi$  may specify complex session protocols that include both first-order communication (name passing) and higher-order processes (process passing) and that can be type-checked using *session types* [9]. By enforcing *shared* and *linear* usage policies, session types ensure that each communication channel in a process specification conforms to its prescribed protocol. In session-based concurrency, distinguish-

---

✉ Jorge A. Pérez  
j.a.perez@rug.nl

<sup>1</sup> University of Glasgow, Glasgow, UK

<sup>2</sup> University of Groningen, Groningen and CWI, Amsterdam, The Netherlands

<sup>3</sup> Imperial College London, London, UK

ing between shared and linear names is important, for computation conceptually involves two distinct phases: the first one is non-deterministic and uses shared names, as it represents the interaction of processes seeking compatible protocol partners; the second phase proceeds deterministically along linear names, as it specifies the concurrent execution of the session protocols established in the first phase.

Although models of higher-order concurrency with session communication have been already developed (cf. works by Mostrous and Yoshida [25] and by Gay and Vasconcelos [5]), their *behavioural equivalences* remain little understood. Clarifying the status of these equivalences is essential to, e.g., justify non-trivial optimisations in protocols involving both name and process passing. An important aspect in the development of these typed equivalences is that typed semantics are usually *coarser* than untyped semantics. Indeed, since (session) types limit the contexts (environments) in which processes can interact, typed equivalences admit stronger properties than their untyped counterpart.

The form of contextual equivalence typically used in concurrency is *barbed congruence* [10,24]. A well-known behavioural equivalence for higher-order processes is *context bisimilarity* [31]. This is a characterisation of barbed congruence that offers an adequate distinguishing power at the price of heavy universal quantifications in output clauses. Obtaining alternative characterisations of context bisimilarity is thus a recurring, important problem for higher-order calculi—see, e.g., [13,15,21,30,31,34]. In particular, Sangiorgi [30,31] has given characterisations of context bisimilarity for higher-order processes; such characterisations, however, do not scale to calculi with *recursive types*, which are essential to express practical protocols in session-based concurrency. A characterisation that solves this limitation was developed by Jeffrey and Rathke [13]; their solution, however, does not consider *linearity* which, as explained above, is an important aspect in session-based concurrency.

*This work* Building upon [13,30,31], our discovery is that linearity as induced by session types plays a vital rôle in solving the open problem of characterising context bisimilarity for higher-order mobile processes with session communication. Our approach is to exploit the coarser semantics induced by session types to limit the behaviour of higher-order session processes. Indeed, the use of session typed contexts (i.e., environments disciplined by session types) leads to process semantics that admit stronger properties than untyped semantics. Formally, we enforce this limitation in behaviour by defining a *refined* labelled transition system (LTS) which effectively narrows down the spectrum of allowed process behaviours, exploiting elementary processes inhabiting session types. We then introduce *characteristic bisimilarity*: this new notion of typed bisimilarity is *more tractable* than context bisimilarity, in that it relies on the refined LTS for input actions and, more importantly, does not appeal to universal quantifications on output actions.

Our main result is that characteristic bisimilarity coincides with context bisimilarity. Besides confirming the value of characteristic bisimilarity as a useful reasoning technique for higher-order processes with sessions, this result is remarkable also from a technical perspective, for associated completeness proofs do not require operators for name matching, in contrast to Jeffrey and Rathke's technique for higher-order processes with recursive types [13].

*Outline* Next, we informally overview the key ideas of characteristic bisimilarity, our characterisation of contextual equivalence. Then, Sect. 3 presents the session calculus  $\text{HO}\pi$ . Section 4 gives the session type system for  $\text{HO}\pi$  and states type soundness. Section 5 develops *characteristic* bisimilarity and states our main result: characteristic bisimilarity

and contextual equivalence coincide for well-typed  $\text{HO}\pi$  processes (Theorem 2). Section 6 discusses related works, while Sect. 7 collects some concluding remarks.

This paper is a revised, extended version of the conference paper [16]. This presentation includes full technical details—definitions and proofs, collected in Appendices 1 and 2. In particular, we introduce *higher-order bisimilarity* (an auxiliary labelled bisimilarity) and highlight its rôle in the proof of Theorem 2. We also elaborate further on the use case scenario for characteristic bisimilarity given in [16] (the Hotel Booking scenario). Using an additional example, given in Sect. 6, we compare our approach with Jeffrey and Rathke’s [13]. Moreover, we offer extended discussions of related works.

## 2 Overview: characteristic bisimulations

We explain how we exploit session types to define characteristic bisimilarity. Key notions are *triggered* and *characteristic processes/values*. We first informally introduce some basic notation and terminology; formal definitions will be given in Sect. 3.

*Preliminaries* The syntax of  $\text{HO}\pi$  considered in this paper is given below. We write  $n$  to range over shared names  $a, b, \dots$  and  $s, s', \dots$  to range over session (linear) names. Also,  $u, w$  denotes a name or a name variable. Session names are sometimes called *endpoints*. We consider a notion of *duality* on names, particularly relevant for session names: we shall write  $\bar{s}$  to denote the dual endpoint of  $s$ .

Values $V, W$	$::= u$	names (shared and linear)
	$  \lambda x. P$	abstractions
Processes $P, Q$	$::= u!(V).P \mid u?(x).P$	output and input
	$  u \triangleleft l.P \mid u \triangleright \{l_i : P_i\}_{i \in I}$	labelled choice
	$  X \mid \mu X.P$	recursion
	$  V W$	value application
	$  P \mid Q \mid (v n)P \mid \mathbf{0}$	composition, restriction, inaction

Hence, the higher-order character of  $\text{HO}\pi$  comes from the fact that values exchanged in synchronisations include abstractions.

The semantics of  $\text{HO}\pi$  can be given in terms of a labelled transition system (LTS), denoted  $P \xrightarrow{\ell} P'$ , where  $\ell$  denotes a transition label or the internal action  $\tau$ . This way, for instance,  $P \xrightarrow{n?(V)} P'$  denotes an input transition (a value  $V$  received along  $n$ ) and  $P \xrightarrow{(v \tilde{m})n!(V)} P'$  denotes an output transition (a value  $V$  emitted along  $n$ , extruding names  $\tilde{m}$ ). Weak transitions, written  $P \xRightarrow{\ell} P'$ , abstract from internal actions in the usual way. Throughout the paper, we write  $\mathfrak{R}, \mathfrak{R}', \dots$  to denote binary relations on (typed) processes.

$\text{HO}\pi$  processes specify structured communications (protocols) as disciplined by *session types*, denoted  $S, S', \dots$ , which we informally describe next:

$S$	$::= !\langle U \rangle; S \mid ?(U); S$	output/input value of type $U$ , continue as $S$
	$  \oplus \{l_i : S_i\}_{i \in I} \mid \& \{l_i : S_i\}_{i \in I}$	internal/external labelled choice of an $S_i$
	$  \mu t. S \mid \mathbf{t}$	recursive protocol
	$  \text{end}$	completed protocol

As we will see, type  $U$  denotes first-order values (i.e., shared and session names) but also shared and linear functional types, denoted  $U \rightarrow \diamond$  and  $U \multimap \diamond$ , respectively, where  $\diamond$  is the type for processes.

*Issues of context bisimilarity* Context bisimilarity (denoted  $\approx$ , cf. Definition 12) is an overly demanding relation on higher-order processes. It is far from satisfactory due to two issues, associated to demanding clauses for output and input actions. A *first issue* is the universal quantification in the output clause of context bisimilarity. Suppose  $P \Re Q$ , for some context bisimulation  $\Re$ . We have the following clause:

$$(\star) \text{ Whenever } P \xrightarrow{(\nu \tilde{m}_1)n!(V)} P' \text{ there exist } Q', W \text{ such that } Q \xrightarrow{(\nu \tilde{m}_2)n!(W)} Q' \text{ and, for all } R \text{ with } \varepsilon_{\nu}(R) = \{x\}, (\nu \tilde{m}_1)(P' \mid R\{V/x\}) \Re (\nu \tilde{m}_2)(Q' \mid R\{W/x\}).$$

Intuitively, process  $R$  above stands for any possible *context* to which the emitted value ( $V$  and  $W$ ) is supposed to go. (As usual,  $R\{V/x\}$  denotes the capture-avoiding substitution of  $V$  for  $x$  in process  $R$ .) As explained in [31], considering all possible contexts  $R$  is key to achieve an adequate distinguishing power.

The *second issue* is due to inputs, and follows from the fact that we work with an *early* labelled transition system (LTS). Thus, an input prefix may observe infinitely many different values.

To alleviate these issues, in *characteristic bisimilarity* (denoted  $\approx^C$ , cf. Definition 18) we take two (related) steps:

- (a) We replace  $(\star)$  with a clause involving a context *more tractable* than  $R\{V/x\}$  (and  $R\{W/x\}$ ); and
- (b) We refine inputs to avoid observing infinitely many actions on the same input prefix.

*Trigger processes* To address (a), we exploit session types. We first observe that, for any  $V$ , process  $R\{V/x\}$  in  $(\star)$  is context bisimilar to the process

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

In fact, through a name application and a synchronisation on session endpoint  $s$  we do have  $P \approx R\{V/x\}$ :

$$\begin{aligned} P &\xrightarrow{\tau} (\nu s)(s?(x).R \mid \bar{s}!\langle V \rangle. \mathbf{0}) \\ &\xrightarrow{\tau} R\{V/x\} \mid \mathbf{0} \end{aligned}$$

where it is worth noticing that application and endpoint synchronisations are deterministic.

Now let us consider process  $T_V$  below, where  $t$  is a fresh name:

$$T_V = t?(x).(\nu s)(x s \mid \bar{s}!\langle V \rangle. \mathbf{0}) \tag{1}$$

If  $T_V$  inputs value  $\lambda z. z?(x).R$  then we have:

$$T_V \xrightarrow{t?(\lambda z. z?(x).R)} R\{V/x\} \approx P$$

Processes such as  $T_V$  offer a value at a fresh name; this class of *trigger processes* already suggests a tractable formulation of bisimilarity without the demanding output clause  $(\star)$ . Process  $T_V$  in (1) requires a higher-order communication along  $t$ . As we explain below, we can give an alternative trigger process; the key is using *elementary inhabitants* of session types.

*Characteristic processes and values* To address (b), we limit the possible input values (such as  $\lambda z. z?(x).R$  above) by exploiting session types. The key concept is that of **characteristic process/value** of a type, i.e., a simple process term that inhabits that type (Definition 13). To illustrate the key idea underlying characteristic processes, consider the session type

$$S =?(S_1 \rightarrow \diamond); !(S_2); \text{end},$$

which abstracts a protocol that first inputs an abstraction (i.e., a function from values  $S_1$  to processes), and then outputs a value of type  $S_2$ . Let  $P$  be the process  $u?(x).(u!(s_2).\mathbf{0} \mid x s_1)$ , where  $s_1, s_2$  are fresh names. It can be shown that  $P$  inhabits session type  $S$ ; for the purposes of the behavioural theory developed in this paper, process  $P$  will serve as a kind of characteristic (representative) process for  $S$  along name  $u$ .

Given a session type  $S$  and a name  $u$ , we write  $[S]^u$  for the characteristic process of  $S$  along  $u$ . Also, given a value type  $U$  (i.e., a type for channels or abstractions), we write  $[U]_c$  to denote its *characteristic value* (cf. Definition 13). As we explain next, we use  $[U]_c$  to refine input transitions.

*Refined input transitions* To refine input transitions, we need to observe an additional value,  $\lambda x. t?(y).(y x)$ , called the **trigger value** (cf. Definition 14). This is necessary: it turns out that a characteristic value alone as the observable input is not enough to define a sound bisimulation (cf. Example 5). Intuitively, the trigger value is used to observe/simulate application processes.

Based on the above discussion, we define an alternative LTS on typed processes, denoted  $\xrightarrow{\ell}$ . We use this refined LTS to define characteristic bisimulation (Definition 18), in which the demanding clause ( $\star$ ) is replaced with a more tractable output clause based on characteristic trigger processes (cf. (2) below). Key to this alternative LTS is the following (refined) transition rule for input actions (cf. Definition 15) which, roughly speaking, given some fresh  $t$ , only admits names  $m$ , trigger values  $\lambda x. t?(y).(y x)$ , and characteristic values  $[U]_c$ :

$$P \xrightarrow{n?(V)} P' \wedge (V = m \vee V \equiv \lambda x. t?(y).(y x) \vee V \equiv [U]_c) \Rightarrow P \xrightarrow{n?(V)} P'$$

Note the different notation for standard and refined transitions:  $\xrightarrow{n?(V)}$  vs.  $\xrightarrow{n?(V)}$ .

*Characteristic triggers* Following the same reasoning as (1), we can use an alternative trigger process, called **characteristic trigger process**, to replace clause ( $\star$ ). Given a fresh name  $t$  and a value  $V$  of with type  $U$ , we have:

$$t \leftarrow_c V : U \stackrel{\text{def}}{=} t?(x).(v s)(s?(y).[U]^y \mid \bar{s}!(V).\mathbf{0}) \tag{2}$$

This formulation is justified, because given  $T_V$  as in (1), we may show that

$$T_V \xrightarrow{t?([U]; \text{end}]_c} \approx t?(x).(v s)(s?(y).[U]^y \mid \bar{s}!(V).\mathbf{0})$$

Thus, unlike process (1), the characteristic trigger process in (2) does not involve a higher-order communication on  $t$ . In contrast to previous approaches [13,30] our characteristic trigger processes do *not* use recursion or replication. This is key to preserve linearity of session endpoints.

It is also noteworthy that  $\text{HO}\pi$  lacks name matching, which is crucial in [13] to prove completeness of bisimilarity. The lack of matching operators is compensated here with the use of (session) types. Matching gives the observer the ability to test the equality of received names. In contrast, in our theory a process trigger embeds a name into a characteristic process

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

(a) Syntax.

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

(b) Structural Congruence.

$$\begin{array}{ll}
 \text{[App]} & (\lambda x. P)V \longrightarrow P\{V/x\} & \text{[Pass]} & n!(V).P \mid \bar{n}?(x).Q \longrightarrow P \mid Q\{V/x\} \\
 \text{[Sel]} & \frac{j \in I}{n \triangleleft l_j.Q \mid \bar{n} \triangleright \{l_i : P_i\}_{i \in I} \longrightarrow Q \mid P_j} & \text{[Res]} & \frac{P \longrightarrow P'}{(\nu n)P \longrightarrow (\nu n)P'} \\
 \text{[Par]} & \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} & \text{[Cong]} & \frac{P \equiv Q \longrightarrow Q' \equiv P'}{P \longrightarrow P'}
 \end{array}$$

(c) Reduction Semantics.

Fig. 1 HOπ: syntax and semantics (structural congruence and reduction)

so as to observe its (typed) behaviour. Thus, equivalent processes deal with (possibly different) names that have the same (typed) behaviour.

### 3 A higher-order session π-calculus

We introduce the *higher-order session π-calculus* (HOπ) which, as hinted at above, includes both name and abstraction passing, shared and session communication, as well as recursion; it is essentially the language proposed in [25], where a behavioural theory is not developed.

#### 3.1 Syntax

The syntax of HOπ is defined in Fig. 1a. We use  $a, b, c, \dots$  to range over shared names and  $s, \bar{s}, \dots$  to range over session names. We use  $n, m, t, \dots$  for session or shared names. Intuitively, session names represent deterministic communication *endpoints*, while shared names represent non-deterministic points. We define the dual operation over names  $n$  as  $\bar{n}$  with  $\bar{\bar{s}} = s$  and  $\bar{\bar{a}} = a$ . This way, e.g., session names  $s$  and  $\bar{s}$  are two dual endpoints. Name variables are denoted with  $x, y, z, \dots$ , and recursive variables are denoted with  $X, Y, \dots$ . Values  $V, W$  include name identifiers  $u, v, \dots$  (first-order values) and abstractions  $\lambda x. P$  (higher-order values), where  $P$  is a process  $P$  and  $x$  is a name parameter.

Process terms include usual π-calculus constructs for sending and receiving values  $V$ : process  $u!(V).P$  denotes the output of  $V$  over name  $u$ , with continuation  $P$ , while process  $u?(x).P$  denotes the input prefix on name  $u$  of a value that will substitute variable  $x$  in the continuation  $P$ . Recursion is expressed by  $\mu X.P$ , which binds the recursive variable  $X$  in process  $P$ . Process  $VW$  represents the application of abstraction  $V$  to value  $W$ . Typing ensures that  $V$  is not a name. In the spirit of session-based π-calculi [9], we consider processes

$u \triangleright \{l_i : P_i\}_{i \in I}$  and  $u \triangleleft l.P$  to define labelled choice: given a finite index set  $I$ , process  $u \triangleright \{l_i : P_i\}_{i \in I}$  offers a choice among processes with pairwise distinct labels; process  $u \triangleleft l.P$  selects label  $l$  on name  $u$  and then behaves as  $P$ . Constructs for inaction  $\mathbf{0}$  and parallel composition  $P_1 \mid P_2$  are standard. Name restriction  $(\nu n)P$  is also as customary; we notice that restriction for session names  $(\nu s)P$  simultaneously binds endpoints  $s$  and  $\bar{s}$  in  $P$ .

We use  $\text{fv}(P)$  and  $\text{fn}(P)$  to denote the sets of free variables and names in  $P$ , respectively. In a statement, we will say that a name is *fresh* if it is not among the names of the objects (processes, actions, etc.) of the statement. We assume that  $V$  in  $u!\langle V \rangle.P$  does not include free recursive variables  $X$ . If  $\text{fv}(P) = \emptyset$ , we call  $P$  *closed*.

### 3.2 Semantics

Figure 1c defines the operational semantics of  $\text{HO}\pi$ , given as a reduction relation that relies on a *structural congruence* relation, denoted  $\equiv$  (Fig. 1b): it includes a congruence that ensures the consistent renaming of bound names, denoted  $\equiv_\alpha$ . We assume the expected extension of  $\equiv$  to values  $V$ . Reduction is denoted  $\longrightarrow$ ; some intuitions on the rules in Fig. 1 follow. Rule [App] defines value application. Rule [Pass] defines an interaction/synchronization at  $n$ ; it can be on a shared name (with  $\bar{n} = n$ ) or a session endpoint. Rule [Sel] is the standard rule for labelled choice/selection [9]: given a finite index set  $I$ , a process selects label  $l_j$  on name  $n$  over a pairwise distinct set of labels  $\{l_i\}_{i \in I}$  offered by a branching on the dual endpoint  $\bar{n}$ ; as a result, process  $P_j$  is selected, and the remaining alternatives are discarded. Other rules are standard. We write  $\longrightarrow^*$  for a multi-step reduction.

### 3.3 An example: the hotel booking scenario

To illustrate  $\text{HO}\pi$  and its expressive power, let us consider a usecase scenario that adapts the example given by Mostrous and Yoshida [25, 26]. 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.

We now present two  $\text{HO}\pi$  implementations of this scenario. For convenience, we write  $\text{if } e \text{ then } (P_1 ; P_2)$  to denote a conditional process that executes  $P_1$  or  $P_2$  depending on boolean expression  $e$  (encodable using labelled choice). The *first implementation* is as follows:

$$\begin{aligned} \text{Client}_1 &\stackrel{\text{def}}{=} (\nu h_1, h_2)(s_1!\langle \lambda x. P_{xy}\{h_1/y\} \rangle.s_2!\langle \lambda x. P_{xy}\{h_2/y\} \rangle.\mathbf{0} \mid \\ &\quad \bar{h}_1?(x).\bar{h}_2?(y).\text{if } x \leq y \text{ then} \\ &\quad (\bar{h}_1 \triangleleft \text{accept}.\bar{h}_2 \triangleleft \text{reject}.\mathbf{0} ; \bar{h}_1 \triangleleft \text{reject}.\bar{h}_2 \triangleleft \text{accept}.\mathbf{0})) \\ P_{xy} &\stackrel{\text{def}}{=} x!\langle \text{room} \rangle.x?(quote).y!\langle \text{quote} \rangle.y \triangleright \left\{ \begin{array}{l} \text{accept} : x \triangleleft \text{accept}.x!\langle \text{credit} \rangle.\mathbf{0} , \\ \text{reject} : x \triangleleft \text{reject}.\mathbf{0} \end{array} \right\} \end{aligned}$$

Process  $\text{Client}_1$  sends two abstractions with body  $P_{xy}$ , one to each hotel, using sessions  $s_1$  and  $s_2$ . That is,  $P_{xy}$  is the mobile code with free names  $x, y$ : while name  $x$  is meant to be instantiated by the hotel as the negotiating endpoint, name  $y$  is used to interact with  $\text{Client}_1$ . Intuitively, process  $P_{xy}$ :

- (i) sends the room requirements to the hotel;
- (ii) receives a quote from the hotel;
- (iii) sends the quote to  $\text{Client}_1$ ;

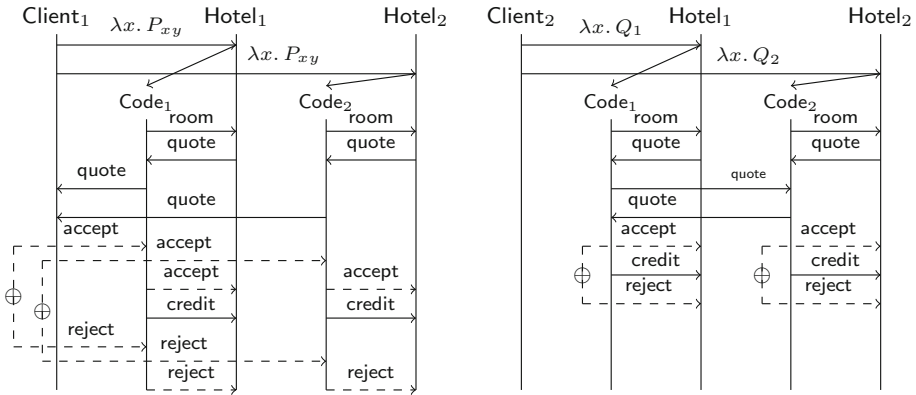


Fig. 2 Sequence diagrams for Client<sub>1</sub> and Client<sub>2</sub>, as in Sect. 3.3

- (iv) expects a choice from Client<sub>1</sub> 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<sub>1</sub> instantiates two copies of  $P_{xy}$  as abstractions on session  $x$ . It uses two fresh endpoints  $h_1, h_2$  to substitute channel  $y$  in  $P_{xy}$ . This enables communication with the mobile code(s). In fact, Client<sub>1</sub> uses the dual endpoints  $\bar{h}_1$  and  $\bar{h}_2$  to receive the negotiation result from the two remote instances of  $P$  and then inform the two processes for the final booking decision.

We present now a *second implementation* in which the two mobile processes reach an agreement by interacting with each other (rather than with the client):

$$\begin{aligned}
 \text{Client}_2 &\stackrel{\text{def}}{=} (v h)(s_1!\langle \lambda x. Q_1\{h/y\} \rangle. s_2!\langle \lambda x. Q_2\{\bar{h}/y\} \rangle. \mathbf{0}) \\
 Q_1 &\stackrel{\text{def}}{=} x!\langle \text{room} \rangle. x?(\text{quote}_1). y!\langle \text{quote}_1 \rangle. y?(\text{quote}_2). R_x \\
 Q_2 &\stackrel{\text{def}}{=} x!\langle \text{room} \rangle. x?(\text{quote}_1). y?(\text{quote}_2). y!\langle \text{quote}_1 \rangle. R_x \\
 R_x &\stackrel{\text{def}}{=} \text{if } \text{quote}_1 \leq \text{quote}_2 \text{ then } (x \triangleleft \text{accept}. x!\langle \text{credit} \rangle. \mathbf{0} ; x \triangleleft \text{reject}. \mathbf{0})
 \end{aligned}$$

Processes  $Q_1$  and  $Q_2$  negotiate a quote from the hotel in the same fashion as process  $P_{xy}$  in Client<sub>1</sub>. 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<sub>2</sub> 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.

The different protocols implemented by Client<sub>1</sub> and Client<sub>2</sub> can be represented by the sequence diagrams of Fig. 2. We will assign session types to these processes in Example 1. Later on, in Sect. 5.9 we will show that Client<sub>1</sub> and Client<sub>2</sub> are behaviourally equivalent using characteristic bisimilarity; see Proposition 3.

### 4 Types and typing

We define a session typing system for HO $\pi$  and state its main properties. As we explain below, our system distills the key features of [25,26].



### 4.1 Types

The syntax of types of  $\text{HO}\pi$  is given below:

$$\begin{array}{ll}
 \text{(value)} & U ::= C \mid L \\
 \text{(name)} & C ::= S \mid \langle S \rangle \mid \langle L \rangle \\
 \text{(abstractions)} & L ::= U \rightarrow \diamond \mid U \multimap \diamond \\
 \text{(session)} & S ::= !\langle U \rangle; S \mid ?\langle U \rangle; S \mid \oplus \{l_i : S_i\}_{i \in I} \mid \& \{l_i : S_i\}_{i \in I} \\
 & \mid \mu t. S \mid \mathbf{t} \mid \text{end}
 \end{array}$$

Value types  $U$  include the first-order types  $C$  and the higher-order types  $L$ . Session types are denoted with  $S$  and shared types with  $\langle S \rangle$  and  $\langle L \rangle$ . We write  $\diamond$  to denote the *process type*. The functional types  $U \rightarrow \diamond$  and  $U \multimap \diamond$  denote *shared* and *linear* higher-order types, respectively. Session types have the meaning already motivated in Sect. 2. The *output type*  $!\langle U \rangle; S$  first sends a value of type  $U$  and then follows the type described by  $S$ . Dually,  $?\langle U \rangle; S$  denotes an *input type*. The *selection type*  $\oplus \{l_i : S_i\}_{i \in I}$  and the *branching type*  $\& \{l_i : S_i\}_{i \in I}$  define labelled choice, implemented at the level of processes by internal and external choice mechanisms, respectively. Type  $\text{end}$  is the termination type. We assume the *recursive type*  $\mu t. S$  is guarded, i.e., the type variable  $t$  only appears under prefixes. This way, e.g., the type  $\mu t. t$  is not allowed. The sets of free/bound variables of a session type  $S$  are defined as usual; the sole binder is  $\mu t. S$ . Closed session types do not have free type variables.

Our type system is strictly included in that considered in [25, 26], which admits asynchronous communication and arbitrary nesting in functional types, i.e., their types are of the form  $U \multimap T$  (resp.  $U \rightarrow T$ ), where  $T$  ranges over  $U$  and the process type  $\diamond$ . In contrast, our functional types are of the form  $U \multimap \diamond$  (resp.  $U \rightarrow \diamond$ ).

We rely on notions of *duality* and *equivalence* for types. Let us write  $S_1 \sim S_2$  to denote that  $S_1$  and  $S_2$  are *type-equivalent* (see Definition 21 in the Appendix). This notion extends to value types as expected; in the following, we write  $U_1 \sim U_2$  to denote that  $U_1$  and  $U_2$  are type-equivalent. We write  $S_1$  *dual*  $S_2$  if  $S_1$  is the *dual* of  $S_2$ . Intuitively, duality converts  $!$  into  $?$  and  $\oplus$  into  $\&$  (and vice-versa). More formally, following [4], we have a co-inductive definition for type duality:

**Definition 1 (Duality)** Let  $\text{ST}$  be a set of closed session types. Two types  $S$  and  $S'$  are said to be *dual* if the pair  $(S, S')$  is in the largest fixed point of the monotone function  $F : \mathcal{P}(\text{ST} \times \text{ST}) \rightarrow \mathcal{P}(\text{ST} \times \text{ST})$  defined by:

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

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

### 4.2 Typing environments and judgements

Typing *environments* are defined below:

$$\begin{aligned} \Gamma &::= \emptyset \mid \Gamma \cdot x : U \rightarrow \diamond \mid \Gamma \cdot u : \langle S \rangle \mid \Gamma \cdot u : \langle L \rangle \mid \Gamma \cdot X : \Delta \\ \Lambda &::= \emptyset \mid \Lambda \cdot x : U \multimap \diamond \\ \Delta &::= \emptyset \mid \Delta \cdot u : S \end{aligned}$$

Typing environments  $\Gamma$ ,  $\Lambda$ , and  $\Delta$  satisfy different structural principles. Intuitively, the *exchange* principle indicates that the ordering of type assignments does not matter. *Weakening* says that type assignments need not be used. Finally, *contraction* says that type assignments may be duplicated.

The environment  $\Gamma$  maps variables and shared names to value types, and recursive variables to session environments; it admits weakening, contraction, and exchange principles. While  $\Lambda$  maps variables to linear higher-order types,  $\Delta$  maps session names to session types. Both  $\Lambda$  and  $\Delta$  are only subject to exchange. The domains of  $\Gamma$ ,  $\Lambda$  and  $\Delta$  are assumed pairwise distinct.

Given  $\Gamma$ , we write  $\Gamma \setminus x$  to denote the environment obtained from  $\Gamma$  by removing the assignment  $x : U \rightarrow \diamond$ , for some  $U$ . This notation applies similarly to  $\Delta$  and  $\Lambda$ ; we write  $\Delta \setminus \Delta'$  (and  $\Lambda \setminus \Lambda'$ ) with the expected meaning. Notation  $\Delta_1 \cdot \Delta_2$  means the disjoint union of  $\Delta_1$  and  $\Delta_2$ . We define *typing judgements* for values  $V$  and processes  $P$ :

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

While the judgement on the left says that under environments  $\Gamma$ ,  $\Lambda$ , and  $\Delta$  value  $V$  has type  $U$ ; the judgement on the right says that under environments  $\Gamma$ ,  $\Lambda$ , and  $\Delta$  process  $P$  has the process type  $\diamond$ . The type soundness result for  $\text{HO}\pi$  (Theorem 1) relies on two auxiliary notions on session environments:

**Definition 2** (*Session environments: balanced/reduction*) Let  $\Delta$  be a session environment.

- $\Delta$  is *balanced* if whenever  $s : S_1, \bar{s} : S_2 \in \Delta$  then  $S_1$  dual  $S_2$ .
- We define the reduction relation  $\longrightarrow$  on session environments as:

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

We rely on a typing system that is similar to the one developed in [25,26]. The typing system is defined in Fig. 3. Rules [SESS], [SH], [LVAR] are name and variable introduction rules. Rule [PROM] allows a value with a linear type  $U \multimap \diamond$  to be used as  $U \rightarrow \diamond$  if its linear environment is empty. Rule [EPROM] allows to freely use a shared type variable in a linear way.

Abstraction values are typed with Rule [ABS]. The key type for an abstraction is the type for the bound variable of the abstraction, i.e., for a bound variable with type  $C$  the corresponding abstraction has type  $C \multimap \diamond$ . The dual of abstraction typing is application typing, governed by Rule [APP]: we expect the type  $U$  of an application value  $W$  to match the type  $U \multimap \diamond$  or  $U \rightarrow \diamond$  of the application variable  $x$ .

In Rule [SEND], the type  $U$  of the sent value  $V$  should appear as a prefix on the session type  $!\langle U \rangle; S$  of  $u$ . Rule [RCV] is its dual. We use a similar approach with session prefixes to type interaction between shared names as defined in Rules [REQ] and [ACC], where the type of the sent/received object ( $S$  and  $L$ , respectively) should match the type of the sent/received subject ( $\langle S \rangle$  and  $\langle L \rangle$ , respectively). Rules [SEL] and [BRA] for selection and branching are standard:

$\frac{[\text{SESS}]}{\Gamma; \emptyset; \{u : S\} \vdash u \triangleright S}$	$\frac{[\text{SH}]}{\Gamma \cdot u : U; \emptyset; \emptyset \vdash u \triangleright U}$	$\frac{[\text{LVAR}]}{\Gamma; \{x : U \multimap \diamond\}; \emptyset \vdash x \triangleright U \multimap \diamond}$
$\frac{[\text{PROM}]}{\Gamma; \emptyset; \emptyset \vdash V \triangleright U \multimap \diamond}$	$\frac{[\text{EPROM}]}{\Gamma; \Lambda \cdot x : U \multimap \diamond; \Delta \vdash P \triangleright \diamond}$	$\frac{[\text{ABS}]}{\Gamma; \Lambda; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \emptyset; \Delta_2 \vdash x \triangleright U}$
$\frac{[\text{APP}]}{U = U' \multimap \diamond \vee U' \multimap \diamond \quad \Gamma; \Lambda; \Delta_1 \vdash V \triangleright U \quad \Gamma; \emptyset; \Delta_2 \vdash W \triangleright U'}$	$\frac{}{\Gamma; \Lambda; \Delta_1 \cdot \Delta_2 \vdash VW \triangleright \diamond}$	
$\frac{[\text{SEND}]}{\Gamma; \Lambda_1; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \Lambda_2; \Delta_2 \vdash V \triangleright U \quad u : S \in \Delta_1 \cdot \Delta_2}$		
$\frac{}{\Gamma; \Lambda_1 \cdot \Lambda_2; ((\Delta_1 \cdot \Delta_2) \setminus u) \cdot u : !\langle U \rangle; S \vdash u!(V).P \triangleright \diamond}$		
$\frac{[\text{RCV}]}{\Gamma; \Lambda_1; \Delta_1 \cdot u : S \vdash P \triangleright \diamond \quad \Gamma; \Lambda_2; \Delta_2 \vdash x \triangleright U}$		
$\frac{}{\Gamma \setminus x; \Lambda_1 \cdot \Lambda_2; \Delta_1 \setminus \Delta_2 \cdot u : ?\langle U \rangle; S \vdash u?(x).P \triangleright \diamond}$		
$\frac{[\text{REQ}]}{\Gamma; \emptyset; \emptyset \vdash u \triangleright \langle U \rangle \quad \Gamma; \Lambda; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \emptyset; \Delta_2 \vdash V \triangleright U}$		
$\frac{}{\Gamma; \Lambda; \Delta_1 \cdot \Delta_2 \vdash u!(V).P \triangleright \diamond}$		
$\frac{[\text{ACC}]}{\Gamma; \emptyset; \emptyset \vdash u \triangleright \langle U \rangle \quad \Gamma; \Lambda_1; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \Lambda_2; \Delta_2 \vdash x \triangleright U}$		
$\frac{}{\Gamma \setminus x; \Lambda_1 \setminus \Lambda_2; \Delta_1 \setminus \Delta_2 \vdash u?(x).P \triangleright \diamond}$		
$\frac{[\text{BRA}]}{\forall i \in I \quad \Gamma; \Lambda; \Delta \cdot u : S_i \vdash P_i \triangleright \diamond}$	$\frac{[\text{SEL}]}{\Gamma; \Lambda; \Delta \cdot u : S_j \vdash P \triangleright \diamond \quad j \in I}$	
$\frac{}{\Gamma; \Lambda; \Delta \cdot u : \&\{l_i : S_i\}_{i \in I} \vdash u \triangleright \{l_i : P_i\}_{i \in I} \triangleright \diamond}$		
$\frac{}{\Gamma; \Lambda; \Delta \cdot u : \oplus\{l_i : S_i\}_{i \in I} \vdash u \triangleleft l_j . P \triangleright \diamond}$		
$\frac{[\text{RESS}]}{\Gamma; \Lambda; \Delta \cdot s : S_1 \cdot \bar{s} : S_2 \vdash P \triangleright \diamond \quad S_1 \text{ dual } S_2}$	$\frac{[\text{RES}]}{\Gamma \cdot a : \langle S \rangle; \Lambda; \Delta \vdash P \triangleright \diamond}$	
$\frac{}{\Gamma; \Lambda; \Delta \vdash (\nu s)P \triangleright \diamond}$		
$\frac{}{\Gamma; \Lambda; \Delta \vdash (\nu a)P \triangleright \diamond}$		
$\frac{[\text{PAR}]}{\Gamma; \Lambda_i; \Delta_i \vdash P_i \triangleright \diamond \quad i = 1, 2}$	$\frac{[\text{END}]}{\Gamma; \Lambda; \Delta \vdash P \triangleright \diamond \quad u \notin \text{dom}(\Gamma, \Lambda, \Delta)}$	
$\frac{}{\Gamma; \Lambda_1 \cdot \Lambda_2; \Delta_1 \cdot \Delta_2 \vdash P_1 \mid P_2 \triangleright \diamond}$		
$\frac{}{\Gamma; \Lambda; \Delta \cdot u : \text{end} \vdash P \triangleright \diamond}$		
$\frac{[\text{REC}]}{\Gamma \cdot X : \Delta; \emptyset; \Delta \vdash P \triangleright \diamond}$	$\frac{[\text{RVAR}]}{\Gamma \cdot X : \Delta; \emptyset; \Delta \vdash X \triangleright \diamond}$	$\frac{[\text{NIL}]}{\Gamma; \emptyset; \emptyset \vdash \mathbf{0} \triangleright \diamond}$
$\frac{}{\Gamma; \emptyset; \Delta \vdash \mu X . P \triangleright \diamond}$		

**Fig. 3** Typing rules for  $\text{HO}\pi$

both rules prefix the session type with the selection type  $\oplus\{l_i : S_i\}_{i \in I}$  and  $\&\{l_i : S_i\}_{i \in I}$ , respectively.

A shared name creation  $a$  creates and restricts  $a$  in environment  $\Gamma$  as defined in Rule [RES]. Creation of a session name  $s$  creates and restricts two endpoints with dual types in Rule [RESS]. Rule [PAR] combines the environments  $\Lambda$  and  $\Delta$  of the parallel components of a parallel process. The disjointness of environments  $\Lambda$  and  $\Delta$  is implied. Rule [END] adds a name with type  $\text{end}$  in  $\Delta$ . The recursion requires that the body process matches the type of the recursive variable as in Rule [REC]. The recursive variable is typed directly from the

shared environment  $\Gamma$  as in Rule [RVAR], Rule [NIL] says that the inactive process  $\mathbf{0}$  is typed with empty linear environments  $\Lambda$  and  $\Delta$ .

We state the type soundness result for  $\text{HO}\pi$  processes.

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

*Proof* Following standard lines. See Appendix 1 for details. □

*Example 1* (The hotel booking example, revisited) We give types to the client processes of Sect. 3.3. Assume

$$S = !(\text{quote}); \&\{\text{accept} : \text{end}, \text{reject} : \text{end}\}$$

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

While the typing for  $\lambda x. P_{xy}$  is  $\emptyset; \emptyset; y : S \vdash \lambda x. P_{xy} \triangleright U \multimap \diamond$ , the typing for  $\text{Client}_1$  is  $\emptyset; \emptyset; s_1 : !\langle U \multimap \diamond \rangle; \text{end} \cdot s_2 : !\langle U \multimap \diamond \rangle; \text{end} \vdash \text{Client}_1 \triangleright \diamond$ .

The typings for  $Q_1$  and  $Q_2$  are  $\emptyset; \emptyset; y : !(\text{quote}); ?(\text{quote}); \text{end} \vdash \lambda x. Q_i \triangleright U \multimap \diamond$  ( $i = 1, 2$ ) and the type for  $\text{Client}_2$  is  $\emptyset; \emptyset; s_1 : !\langle U \multimap \diamond \rangle; \text{end} \cdot s_2 : !\langle U \multimap \diamond \rangle; \text{end} \vdash \text{Client}_2 \triangleright \diamond$ .

## 5 Characteristic bisimulation

We develop a theory for observational equivalence over session typed  $\text{HO}\pi$  processes that follows the principles laid in our previous works [18, 19]. We introduce *higher-order bisimulation* (Definition 17) and *characteristic bisimulation* (Definition 18), denoted  $\approx^H$  and  $\approx^C$ , respectively. We prove that they coincide with (reduction-closed) barbed congruence (denoted  $\cong$ , cf. Definition 11), the form of contextual equivalence used in concurrency. This characterisation result is given in Theorem 2.

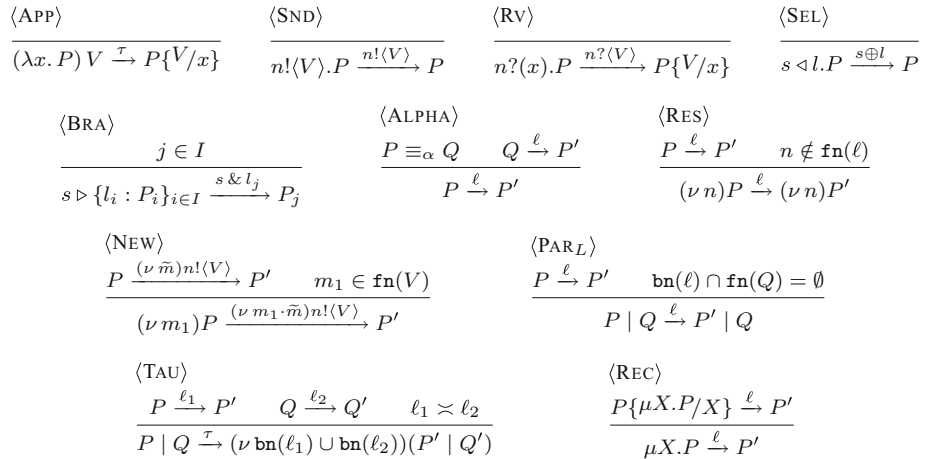
We briefly summarise our strategy for obtaining Theorem 2. We begin by defining an (early) labelled transition system (LTS) on untyped processes (Sect. 5.1). Then, using the *environmental* transition semantics (Sect. 5.2), we define a typed LTS that formalises how a typed process interacts with a typed observer. Later, we define reduction-closed, barbed congruence and context bisimilarity, respectively (Sects. 5.3 and 5.4). Subsequently, we define the refined LTS based on characteristic values (Sect. 5.5). Building upon this LTS, we define higher-order and characteristic bisimilarities (Sect. 5.6). Then, we develop an auxiliary proof technique based on deterministic transitions (Sect. 5.7). Our main result, the characterisation of barbed congruence in terms of  $\approx^H$  and  $\approx^C$ , is stated in Sect. 5.8. Finally, we revisit our two implementations for the Hotel Booking Scenario (Sect. 3.3), using Theorem 2 to show that they are behaviourally equivalent (Sect. 5.9).

### 5.1 Labelled transition system for processes

We define the interaction of processes with their environment using action labels  $\ell$ :

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

Label  $\tau$  defines internal actions. Action  $(v \tilde{m})n!\langle V \rangle$  denotes the sending of value  $V$  over channel  $n$  with a possible empty set of restricted names  $\tilde{m}$  (we may write  $n!\langle V \rangle$  when  $\tilde{m}$  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  $\text{fn}(\ell)$  and  $\text{bn}(\ell)$  to denote the sets



**Fig. 4** The untyped LTS for HOπ processes. We omit Rule  $\langle \text{PAR}_R \rangle$

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

*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 3** (*Dual actions*) We define duality on actions as the least symmetric relation  $\asymp$  on action labels that satisfies:

$$n \oplus l \asymp \bar{n} \& l \quad (\nu \tilde{m})n!\langle V \rangle \asymp \bar{n}?\langle V \rangle$$

The (early) labelled transition system (LTS) for *untyped* processes is given in Fig. 4. We write  $P_1 \xrightarrow{\ell} P_2$  with the usual meaning. The rules are standard [18, 19]; 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  $\langle \text{SND} \rangle$ ). Dually, in Rule  $\langle \text{RV} \rangle$  a receiver process can observe an input of an arbitrary value  $V$ . Select and branch processes observe the select and branch actions in Rules  $\langle \text{SEL} \rangle$  and  $\langle \text{BRA} \rangle$ , respectively. Rule  $\langle \text{RES} \rangle$  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  $\langle \text{NEW} \rangle$ ). Rule  $\langle \text{REC} \rangle$  handles recursion unfolding. Rule  $\langle \text{TAU} \rangle$  states that two parallel processes which perform dual actions can synchronise by an internal transition. Rules  $\langle \text{PAR}_L \rangle / \langle \text{PAR}_R \rangle$  and  $\langle \text{ALPHA} \rangle$  define standard treatments for actions under parallel composition and  $\alpha$ -renaming.

### 5.2 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. 5. Building upon the reduction relation for session environments in Definition 2, such a relation is defined on triples of environments by extending the LTSs in [18, 19]; it is denoted

$$\begin{array}{c}
\text{[SRV]} \\
\frac{\bar{s} \notin \text{dom}(\Delta) \quad \Gamma; \Lambda'; \Delta' \vdash V \triangleright U}{(\Gamma; \Lambda; \Delta \cdot s : ?(U); S) \xrightarrow{s?(V)} (\Gamma; \Lambda \cdot \Lambda'; \Delta \cdot \Delta' \cdot s : S)} \\
\\
\text{[SHRV]} \\
\frac{\Gamma; \emptyset; \emptyset \vdash a \triangleright \langle U \rangle \quad \Gamma; \Lambda'; \Delta' \vdash V \triangleright U}{(\Gamma; \Lambda; \Delta) \xrightarrow{a?(V)} (\Gamma; \Lambda \cdot \Lambda'; \Delta \cdot \Delta')} \\
\\
\text{[SSND]} \\
\frac{\Gamma \cdot \Gamma'; \Lambda'; \Delta' \vdash V \triangleright U \quad \Gamma'; \emptyset; \Delta_j \vdash m_j \triangleright U_j \quad \bar{s} \notin \text{dom}(\Delta) \quad \Delta' \setminus (\cup_j \Delta_j) \subseteq (\Delta \cdot s : S) \quad \Gamma'; \emptyset; \Delta'_j \vdash \bar{m}_j \triangleright U'_j \quad \Lambda' \subseteq \Lambda}{(\Gamma; \Lambda; \Delta \cdot s : !(U); S) \xrightarrow{(\nu \tilde{m})s!(V)} (\Gamma \cdot \Gamma'; \Lambda \setminus \Lambda'; (\Delta \cdot s : S \cdot \cup_j \Delta'_j) \setminus \Delta')} \\
\\
\text{[SHSND]} \\
\frac{\Gamma \cdot \Gamma'; \Lambda'; \Delta' \vdash V \triangleright U \quad \Gamma'; \emptyset; \Delta_j \vdash m_j \triangleright U_j \quad \Gamma; \emptyset; \emptyset \vdash a \triangleright \langle U \rangle \quad \Delta' \setminus (\cup_j \Delta_j) \subseteq \Delta \quad \Gamma'; \emptyset; \Delta'_j \vdash \bar{m}_j \triangleright U'_j \quad \Lambda' \subseteq \Lambda}{(\Gamma; \Lambda; \Delta) \xrightarrow{(\nu \tilde{m})a!(V)} (\Gamma \cdot \Gamma'; \Lambda \setminus \Lambda'; (\Delta \cdot \cup_j \Delta'_j) \setminus \Delta')} \\
\\
\text{[SEL]} \\
\frac{\bar{s} \notin \text{dom}(\Delta) \quad j \in I}{(\Gamma; \Lambda; \Delta \cdot s : \oplus\{l_i : S_i\}_{i \in I}) \xrightarrow{s \oplus l_j} (\Gamma; \Lambda; \Delta \cdot s : S_j)} \\
\\
\text{[BRA]} \qquad \qquad \qquad \text{[TAU]} \\
\frac{\bar{s} \notin \text{dom}(\Delta) \quad j \in I}{(\Gamma; \Lambda; \Delta \cdot s : \&\{l_i : T_i\}_{i \in I}) \xrightarrow{s \& l_j} (\Gamma; \Lambda; \Delta \cdot s : S_j)} \qquad \frac{\Delta_1 \longrightarrow \Delta_2 \vee \Delta_1 = \Delta_2}{(\Gamma; \Lambda; \Delta_1) \xrightarrow{\tau} (\Gamma; \Lambda; \Delta_2)}
\end{array}$$

**Fig. 5** Labelled transition system for typed environments

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

Recall that  $\Gamma$  admits weakening. Using this principle (not valid for  $\Lambda$  and  $\Delta$ ), we have  $(\Gamma', \Lambda_1, \Delta_1) \xrightarrow{\ell} (\Gamma', \Lambda_2, \Delta_2)$  whenever  $(\Gamma, \Lambda_1, \Delta_1) \xrightarrow{\ell} (\Gamma', \Lambda_2, \Delta_2)$ .

*Input actions* are defined by Rules [SRV] and [SHRV]. In Rule [SRV] the type of value  $V$  and the type of the object associated to the session type on  $s$  should coincide. The resulting type tuple must contain the environments associated to  $V$ . The dual endpoint  $\bar{s}$  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* are defined by Rules [SSND] and [SHSND]. Rule [SSND] states the conditions for observing action  $(\nu \tilde{m})s!(V)$  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  $\Delta'$  in the rule), *excluding* the session environments of names  $m_j$  in  $\tilde{m}$  which restrict the scope of value  $V$  (denoted  $\Delta_j$  in the rule). Analogously, the linear variable environment  $\Lambda'$  of  $V$  should

be included in  $\Lambda$ . The rule defines the scope extrusion of session names in  $\tilde{m}$ ; consequently, environments associated to their dual endpoints (denoted  $\Delta'_j$  in the rule) appear in the resulting session environment. Similarly for shared names in  $\tilde{m}$  that are extruded. All free values used for typing  $V$  (denoted  $\Lambda'$  and  $\Delta'$  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 2) or keeps it unchanged.

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

*Example 2* Consider environment tuple  $(\Gamma; \emptyset; s : !(\langle S \rangle; \text{end}) \multimap \diamond; \text{end} \cdot s' : S)$  and typed value  $V = \lambda x. x!(s').m?(z).\mathbf{0}$  with

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

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

$$(\Gamma; \emptyset; s : !(\langle S \rangle; \text{end}) \multimap \diamond; \text{end} \cdot s' : S) \xrightarrow{(vm)s!(V)} (\Gamma; \emptyset; s : \text{end} \cdot \bar{m} : !(\text{end}); \text{end})$$

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

**Notation 4** Given a value  $V$  of type  $U$ , we sometimes annotate the output action  $(v \tilde{m})n!(V)$  with the type of  $V$  as  $(v \tilde{m})n!(V : U)$ .

The typed LTS combines the LTSs in Figs. 4 and 5.

**Definition 5** (*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  $\implies$  for the reflexive and transitive closure of  $\rightarrow$ ,  $\xRightarrow{\ell}$  for the transitions  $\implies \xrightarrow{\ell} \implies$ , and  $\xRightarrow{\hat{\ell}}$  for  $\xRightarrow{\ell}$  if  $\ell \neq \tau$  otherwise  $\implies$ .

A typed transition relation requires type judgements with an empty  $\Lambda$ , 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. 3) and obtain an empty  $\Lambda$ . As it will be clear below (cf. Definition 7), we will be working with closed process terms, i.e., processes without free variables.

### 5.3 Reduction-closed, barbed congruence ( $\cong$ )

We now define *typed relations* and *contextual equivalence* (i.e., barbed congruence). To define typed relations, we first define *confluence* over session environments  $\Delta$ . Recall that  $\Delta$  captures session communication, which is deterministic. The notion of confluence allows us to abstract away from alternative computation paths that may arise due to non-interfering reductions of session names.

**Definition 6** (*Session environment confluence*) Two session environments  $\Delta_1$  and  $\Delta_2$  are *confluent*, denoted  $\Delta_1 \rightleftharpoons \Delta_2$ , if there exists a  $\Delta$  such that: i)  $\Delta_1 \longrightarrow^* \Delta$  and ii)  $\Delta_2 \longrightarrow^* \Delta$  (here we write  $\longrightarrow^*$  for the multi-step reduction in Definition 2).

We illustrate confluence by means of an example:

*Example 3* (Session environment confluence) Consider the (balanced) session environments:

$$\begin{aligned} \Delta_1 &= \{s_1 : T_1 \cdot s_2 : ?(U_2); \text{end} \cdot \bar{s}_2 : !(U_2); \text{end}\} \\ \Delta_2 &= \{s_1 : T_1 \cdot s_2 : !(U_1); ?(U_2); \text{end} \cdot \bar{s}_2 : ?(U_1); !(U_2); \text{end}\} \end{aligned}$$

Following Definition 2, we have that  $\Delta_1 \longrightarrow \{s_1 : T_1 \cdot s_2 : \text{end} \cdot \bar{s}_2 : \text{end}\}$  and  $\Delta_2 \longrightarrow \{s_1 : T_1 \cdot s_2 : \text{end} \cdot \bar{s}_2 : \text{end}\}$ . Therefore,  $\Delta_1$  and  $\Delta_2$  are confluent.  $\square$

Typed relations relate only closed processes whose session environments are balanced and confluent:

**Definition 7** (*Typed relation*) We say that a binary relation over typing judgements

$$\Gamma; \emptyset; \Delta_1 \vdash P_1 \triangleright \diamond \Re \Gamma; \emptyset; \Delta_2 \vdash P_2 \triangleright \diamond$$

is a *typed relation* whenever:

1.  $P_1$  and  $P_2$  are closed;
2.  $\Delta_1$  and  $\Delta_2$  are balanced (cf. Definition 2); and
3.  $\Delta_1 \rightleftharpoons \Delta_2$  (cf. Definition 6).

**Notation 8** (Typed relations) We write

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

to denote the typed relation  $\Gamma; \emptyset; \Delta_1 \vdash P_1 \triangleright \diamond \Re \Gamma; \emptyset; \Delta_2 \vdash P_2 \triangleright \diamond$ .

Next we define *barbs* [24] with respect to types.

**Definition 9** (*Barbs*) Let  $P$  be a closed process. We write

1. (a)  $P \downarrow_n$  if  $P \equiv (\nu \tilde{m})(n!(V).P_2 \mid P_3)$  or  $P \equiv (\nu \tilde{m})(n \triangleleft l.P_2 \mid P_3)$ , with  $n \notin \tilde{m}$ .  
 (b) We write  $P \Downarrow_n$  if  $P \longrightarrow^* \downarrow_n$ .
2. Similarly, we write  
 (a)  $\Gamma; \emptyset; \Delta \vdash P \downarrow_n$  if  $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$  with  $P \downarrow_n$  and  $\bar{n} \notin \Delta$ .  
 (b) We write  $\Gamma; \emptyset; \Delta \vdash P \Downarrow_n$  if  $P \longrightarrow^* P'$  and  $\Gamma; \emptyset; \Delta' \vdash P' \downarrow_n$ .

A barb  $\downarrow_n$  is an observable on an output (resp. select) prefix with subject  $n$ ; a weak barb  $\Downarrow_n$  is a barb after zero or more reduction steps. Typed barbs  $\downarrow_n$  (resp.  $\Downarrow_n$ ) are observed on typed processes  $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$ . When  $n$  is a session name we require that its dual endpoint  $\bar{n}$  is not present in the session environment  $\Delta$ .

Notice that observing output barbs is enough to (indirectly) observe input actions. For instance, the process  $P = n?(x).P'$  has an input barb on  $n$ ; by composing  $P$  with  $n!\langle m \rangle.succ!\langle \rangle.\mathbf{0}$  (with a fresh name *succ*) then one obtains a (weak) observation uniquely associated to the input along  $n$  in  $P$ .

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



**Definition 10** (*Context*) Context  $\mathbb{C}$  is defined over the syntax:

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

Notation  $\mathbb{C}[P]$  denotes the result of substituting the hole  $-$  in  $\mathbb{C}$  with process  $P$ .

The first behavioural relation that we define is reduction-closed, barbed congruence [10].

**Definition 11** (*Reduction-closed, barbed congruence*) Typed relation

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

is a *reduction-closed, barbed congruence* whenever:

- (1) (a) If  $P \longrightarrow P'$  then there exist  $\Delta'_1, Q', \Delta'_2$  such that  $Q \longrightarrow^* Q'$  and  $\Gamma; \Delta'_1 \vdash P' \Re \Delta'_2 \vdash Q'$ ;
- (b) and the symmetric case;
- (2) (a) If  $\Gamma; \Delta_1 \vdash P \Downarrow_n$  then  $\Gamma; \Delta_2 \vdash Q \Downarrow_n$ ;
- (b) and the symmetric case;
- (3) For all  $\mathbb{C}$ , there exist  $\Delta''_1, \Delta''_2$  such that  $\Gamma; \Delta''_1 \vdash \mathbb{C}[P] \Re \Delta''_2 \vdash \mathbb{C}[Q]$ .

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

### 5.4 Context bisimilarity ( $\approx$ )

Following Sangiorgi [31], we now define the standard (weak) context bisimilarity.

**Definition 12** (*Context bisimilarity*) A typed relation  $\Re$  is a *context bisimulation* if for all  $\Gamma; \Delta_1 \vdash P_1 \Re \Delta_2 \vdash Q_1$ ,

- (1) Whenever  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu \tilde{m}_1)n!(V_1)} \Delta'_1 \vdash P_2$ , there exist  $Q_2, V_2, \Delta'_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{(\nu \tilde{m}_2)n!(V_2)} \Delta'_2 \vdash Q_2$  and for all  $R$  with  $\text{fv}(R) = \{x\}$ :
 
$$\Gamma; \Delta'_1 \vdash (\nu \tilde{m}_1)(P_2 \mid R\{V_1/x\}) \Re \Delta'_2 \vdash (\nu \tilde{m}_2)(Q_2 \mid R\{V_2/x\});$$
- (2) For all  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta'_1 \vdash P_2$  such that  $\ell$  is not an output, there exist  $Q_2, \Delta'_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{\hat{\ell}} \Delta'_2 \vdash Q_2$  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 *context bisimilarity* and is denoted by  $\approx$ .

As suggested in Sect. 2, in the general case, context bisimilarity is an overly demanding relation on processes. Below we introduce *higher-order bisimulation* and *characteristic bisimulation*, which are meant to offer a *tractable* proof technique over session typed processes with first- and higher-order communication.

### 5.5 Characteristic values and the refined LTS

We formalise the ideas given in Sect. 2, concerning characteristic processes/values and the refined LTS. We first define characteristic processes/values:

**Definition 13** (*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  $[U]_{\mathbb{C}}$ , are defined in Fig. 6.

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

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

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

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

1. Let  $U$  be a channel type. Then, for some  $\Gamma, \Delta$ , we have  $\Gamma; \emptyset; \Delta \vdash \{U\}_c \triangleright U$ .
2. Let  $S$  be a session type. Then, for some  $\Gamma, \Delta$ , we have  $\Gamma; \emptyset; \Delta \cdot s : S \vdash \{S\}^s \triangleright \diamond$ .
3. Let  $U$  be a channel type. Then, for some  $\Gamma, \Delta$ , we have  $\Gamma \cdot a : U; \emptyset; \Delta \vdash \{U\}^a \triangleright \diamond$ .

*Proof* (Sketch) The proof is done by induction on the syntax of types. See Proposition 4 in the Appendix for details. □

We give an example of a characteristic process inhabiting a recursive type.

*Example 4* (Characteristic process for a recursive session type) Consider the type  $S = \mu t.!\langle U_1 \rangle; ?\langle U_2 \rangle; t$ . By Definition 13, we have that  $\{S\}^s = \{!\langle U_1 \rangle; ?\langle U_2 \rangle; \text{end}\}^s = s!\langle \{U_1\}_c \rangle.t!\langle s \rangle.\mathbf{0}$ . For this process, we can infer the following type derivations:

$$\frac{\Gamma; \emptyset; \Delta \triangleright \{U_1\}_c \triangleright U_2 \Gamma; \emptyset; t : !\langle ?\langle U_2 \rangle; \text{end} \rangle; \text{end} \cdot s : ?\langle U_2 \rangle; \text{end} \vdash t!\langle s \rangle.\mathbf{0} \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot t : !\langle ?\langle U_2 \rangle; \text{end} \rangle; \text{end} \cdot s : !\langle U_1 \rangle; ?\langle U_2 \rangle; \text{end} \vdash s!\langle \{U_1\}_c \rangle.t!\langle s \rangle.\mathbf{0} \triangleright \diamond}$$

and

$$\frac{\Gamma; \emptyset; \Delta \cdot t : !\langle ?\langle U_2 \rangle; \mu t.!\langle U_1 \rangle; ?\langle U_2 \rangle; t \rangle; \text{end} \cdot s : ?\langle U_2 \rangle; \mu t.!\langle U \rangle; t \vdash t!\langle s \rangle.\mathbf{0} \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot t : !\langle ?\langle U_2 \rangle; \mu t.!\langle U \rangle; t \rangle; \text{end} \cdot s : \mu t.!\langle U \rangle; t \vdash s!\langle \{U_1\}_c \rangle.t!\langle s \rangle.\mathbf{0} \triangleright \diamond}$$

The following example motivates the refined LTS explained in Sect. 2. We rely on the following definition.

**Definition 14** (*Trigger value*) Given a fresh name  $t$ , the *trigger value* on  $t$  is defined as the abstraction  $\lambda x. t?(y).(y x)$ .

*Example 5* (The need for the refined typed LTS) We illustrate the complementary rôle that characteristic values (cf. Fig. 6) and the trigger value (Definition 14) play in defining sound bisimilarities.

We first notice that observing characteristic values as inputs is not enough to define a sound bisimulation. Consider processes

$$P_1 = s?(x).(x s_1 \mid x s_2) \qquad P_2 = s?(x).(x s_1 \mid (\lambda z.\mathbf{0}) s_2) \tag{3}$$

such that

$$\Gamma; \emptyset; \Delta \cdot s : ?(\langle \text{end} \rangle \rightarrow \diamond); \text{end} \vdash P_i \triangleright \diamond \quad (i \in \{1, 2\})$$

with  $\Delta = s_1:\text{end} \cdot s_2:\text{end}$ . If  $P_1$  and  $P_2$  input along  $s$  a characteristic value of the form  $[(\text{end}) \rightarrow \diamond]_{\mathbf{c}} = \lambda z. \mathbf{0}$  (cf. Fig. 6), then both of them would evolve into:

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

therefore becoming context bisimilar. However, processes  $P_1$  and  $P_2$  in (3) are clearly *not* context bisimilar: many input actions may be used to distinguish them. For example, if  $P_1$  and  $P_2$  input  $\lambda x. (v s')(a!(s').\mathbf{0})$  with  $\Gamma; \emptyset; \emptyset \vdash a \triangleright \langle \text{end} \rangle$ , then their derivatives are not bisimilar:

$$\begin{aligned} \Gamma; \emptyset; \Delta \vdash P_1 &\xrightarrow{s?(\lambda x. (v s')(a!(s').\mathbf{0}))} \xrightarrow{\tau} \xrightarrow{\tau} \\ &\Delta \vdash (v s')(a!(s').\mathbf{0}) \mid (v s')(a!(s').\mathbf{0}) \\ \Gamma; \emptyset; \Delta \vdash P_2 &\xrightarrow{s?(\lambda x. (v s')(a!(s').\mathbf{0}))} \xrightarrow{\tau} \\ &\Delta \vdash (v s')(a!(s').\mathbf{0}) \mid (\lambda z. \mathbf{0}) s_2 \end{aligned}$$

Observing only the characteristic value results in an under-discriminating bisimulation. However, if a trigger value  $\lambda x. t?(y).(y x)$  (Definition 14) is received along  $s$ , we can distinguish  $P_1$  and  $P_2$  in (3):

$$\begin{aligned} \Gamma; \emptyset; \Delta \vdash P_1 &\xrightarrow{s?(\lambda x. t?(y).(y x))} \Delta \vdash t?(x).(x s_1) \mid t?(x).(x s_2) \quad \text{and} \\ \Gamma; \emptyset; \Delta \vdash P_2 &\xrightarrow{s?(\lambda x. t?(y).(y x))} \Delta \vdash t?(x).(x s_1) \mid (\lambda z. \mathbf{0}) s_2 \end{aligned}$$

In the light of this example, one natural question is whether the trigger value suffices to distinguish two processes (hence no need of characteristic values). This is not the case: the trigger value alone also results in an under-discriminating bisimulation relation. In fact, the trigger value can be observed on any input prefix of *any type*. For example, consider processes:

$$(v s)(n?(x).(x s) \mid \bar{s}!\langle \lambda x. R_1 \rangle.\mathbf{0}) \tag{4}$$

$$(v s)(n?(x).(x s) \mid \bar{s}!\langle \lambda x. R_2 \rangle.\mathbf{0}) \tag{5}$$

If processes in (4) and (5) input the trigger value, we obtain:

$$(v s)(t?(x).(x s) \mid \bar{s}!\langle \lambda x. R_1 \rangle.\mathbf{0})$$

$$(v s)(t?(x).(x s) \mid \bar{s}!\langle \lambda x. R_2 \rangle.\mathbf{0})$$

thus we can easily derive a bisimulation relation if we assume a definition of bisimulation that allows only trigger value input. But if processes in (4)/(5) input the characteristic value  $\lambda z. z?(x).(t!(z).\mathbf{0} \mid x m)$  (where  $m$  is a fresh name) then, under appropriate  $\Gamma$  and  $\Delta$ , they would become:

$$\Gamma; \emptyset; \Delta \vdash (v s)(s?(x).(t!(s).\mathbf{0} \mid x m) \mid \bar{s}!\langle \lambda x. R_i \rangle.\mathbf{0}) \approx \Delta \vdash R_i\{m/x\} \quad (i = 1, 2)$$

which are not bisimilar if  $R_1\{m/x\} \not\approx R_2\{m/x\}$ .

These examples illustrate the need for both trigger and characteristic values as an input observation in the refined transition relation. This will be the content of Definition 15 below. □

As explained in Sect. 2, 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 15** (*Refined typed labelled transition system*) The refined typed labelled transition relation on typing environments

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

is defined on top of the rules in Fig. 5 using the following rules:

$$\frac{[\text{TR}] \quad (\Gamma_1; \Lambda_1; \Delta_1) \xrightarrow{\ell} (\Gamma_2; \Lambda_2; \Delta_2) \quad \ell \neq n?(V)}{(\Gamma_1; \Lambda_1; \Delta_1) \xrightarrow{\ell} (\Gamma_2; \Lambda_2; \Delta_2)}$$

[RRcv]

$$\frac{(\Gamma_1; \Lambda_1; \Delta_1) \xrightarrow{n?(V)} (\Gamma_2; \Lambda_2; \Delta_2) \quad V = m \vee V \equiv [U]_c \vee V \equiv \lambda x. t?(y).(y x) \quad t \text{ fresh}}{(\Gamma_1; \Lambda_1; \Delta_1) \xrightarrow{n?(V)} (\Gamma_2; \Lambda_2; \Delta_2)}$$

Then, the refined typed labelled transition system

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

is given as in Definition 5, replacing the requirement  $(\Gamma, \emptyset, \Delta_1) \xrightarrow{\ell} (\Gamma, \emptyset, \Delta_2)$  with  $(\Gamma_1; \Lambda_1; \Delta_1) \xrightarrow{\ell} (\Gamma_2; \Lambda_2; \Delta_2)$ , as just defined. Following Definition 5, we write  $\Longrightarrow$  for the reflexive and transitive closure of  $\xrightarrow{\tau}$ ,  $\Longrightarrow_{\ell}$  for the transitions  $\Longrightarrow \xrightarrow{\ell} \Longrightarrow$ , and  $\Longrightarrow_{\ell}$  for  $\Longrightarrow$  if  $\ell \neq \tau$  otherwise  $\Longrightarrow$ .

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

**Notation 16** We sometimes write  $\xrightarrow{(v \tilde{m})n!(V:U)}$  when the type of  $V$  is  $U$ .

### 5.6 Higher-order bisimilarity ( $\approx^H$ ) and characteristic bisimilarity ( $\approx^C$ )

Having introduced a refined LTS on  $\text{HO}\pi$  processes, we now define *higher-order bisimilarity* and *characteristic bisimilarity*, two tractable bisimilarity relations. As explained in Sect. 2, the two bisimulations use two different trigger processes [cf. (2)]:

$$t \leftarrow_H V \stackrel{\text{def}}{=} \begin{cases} t?(x).(v s)(s?(y).(x y) \mid \bar{s}!(V).\mathbf{0}) & \text{if } V \text{ is a first-order value} \\ t?(x).(v s)(s?(y).(y x) \mid \bar{s}!(V).\mathbf{0}) & \text{if } V \text{ is a higher-order value} \end{cases} \quad (6)$$

$$t \leftarrow_C V : U \stackrel{\text{def}}{=} t?(x).(v s)(s?(y).[U]^y \mid \bar{s}!(V).\mathbf{0}) \quad (7)$$

The process in (6) is called *higher-order trigger process*, while process in (7) is called *characteristic trigger process*. Notice that while in (6) there is a higher-order input on  $t$ , in (7) the variable  $x$  does not play any rôle.

We use higher-order trigger processes to define *higher-order bisimilarity*:

**Definition 17** (*Higher-order bisimilarity*) A typed relation  $\mathfrak{R}$  is a *higher-order bisimulation* if for all  $\Gamma; \Delta_1 \vdash P_1 \mathfrak{R} \Delta_2 \vdash Q_1$

- (1) Whenever  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(v \tilde{m}_1)n!(V_1)} \Delta'_1 \vdash P_2$ , there exist  $Q_2, V_2, \Delta'_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{(v \tilde{m}_2)n!(V_2)} \Delta'_2 \vdash Q_2$  and, for a fresh  $t$ ,

$$\Gamma; \Delta'_1 \vdash (v \tilde{m}_1)(P_2 \mid t \leftarrow_H V_1) \mathfrak{R} \Delta'_2 \vdash (v \tilde{m}_2)(Q_2 \mid t \leftarrow_H V_2)$$

- (2) For all  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta'_1 \vdash P_2$  such that  $\ell$  is not an output, there exist  $Q_2, \Delta'_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{\hat{\ell}} \Delta'_2 \vdash Q_2$  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  $\approx^H$ .

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

**Definition 18** (*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  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(v \tilde{m}_1)n!(V_1:U_1)} \Delta'_1 \vdash P_2$  then there exist  $Q_2, V_2, \Delta'_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{(v \tilde{m}_2)n!(V_2:U_2)} \Delta'_2 \vdash Q_2$  and, for a fresh  $t$ ,

$$\Gamma; \Delta''_1 \vdash (v \tilde{m}_1)(P_2 \mid t \leftarrow_c V_1 : U_1) \Re \Delta''_2 \vdash (v \tilde{m}_2)(Q_2 \mid t \leftarrow_c V_2 : U_2)$$

- (2) For all  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta'_1 \vdash P_2$  such that  $\ell$  is not an output, there exist  $Q_2, \Delta'_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{\hat{\ell}} \Delta'_2 \vdash Q_2$  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  $\approx^C$ .

Observe how we have used Notation 16 to explicitly refer to the type of the emitted value in output actions.

*Remark 1* (Differences between  $\approx^H$  and  $\approx^C$ ) Although  $\approx^H$  and  $\approx^C$  are conceptually similar, they differ in the kind of trigger process considered. Because of the application in  $t \leftarrow_H V$  (cf. (6)),  $\approx^H$  cannot be used to reason about first-order session processes (i.e., processes without higher-order features). In contrast,  $\approx^C$  is more general: it can uniformly input characteristic, first- or higher-order values.

### 5.7 Deterministic transitions and up-to techniques

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

**Definition 19** (*Deterministic transitions*) Suppose  $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$  with balanced  $\Delta$ . Transition  $\Gamma; \Delta \vdash P \xrightarrow{\tau} \Delta' \vdash P'$  is called:

- session-transition whenever transition  $P \xrightarrow{\tau} P'$  is derived using Rule  $\langle \text{TAU} \rangle$  (where  $\text{subj}(\ell_1)$  and  $\text{subj}(\ell_2)$  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. 4).
- a  $\beta$ -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. 4).

**Notation 20** We use the following notations:

- $\Gamma; \Delta \vdash P \xrightarrow{\tau_s} \Delta' \vdash P'$  denotes a session-transition.
- $\Gamma; \Delta \vdash P \xrightarrow{\tau_\beta} \Delta' \vdash P'$  denotes a  $\beta$ -transition.
- $\Gamma; \Delta \vdash P \xrightarrow{\tau_d} \Delta' \vdash P'$  denotes either a session-transition or a  $\beta$ -transition.

– We write  $\xRightarrow{\tau_d}$  to denote a (possibly empty) sequence of deterministic steps  $\xrightarrow{\tau_d}$ .

Deterministic transitions imply the  $\tau$ -inertness property [7], which ensures behavioural invariance on deterministic transitions:

**Proposition 2** ( $\tau$ -inertness) *Suppose  $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$  with balanced  $\Delta$ . Then*

1.  $\Gamma; \Delta \vdash P \xRightarrow{\tau_d} \Delta' \vdash P'$  implies  $\Gamma; \Delta \vdash P \approx^H \Delta' \vdash P'$ .
2.  $\Gamma; \Delta \vdash P \xRightarrow{\tau_d} \Delta' \vdash P'$  implies  $\Gamma; \Delta \vdash P \approx^H \Delta' \vdash P'$ .

*Proof* (Sketch) The proof of Part 1 requires to show that relation (we omit type information)

$$\mathfrak{R} = \{(P, P') \mid \Gamma; \Delta \vdash P \xRightarrow{\tau_d} \Delta' \vdash P'\}$$

is a higher-order bisimulation. The proof for Part 2 is direct from Part 1. See “Deterministic transitions” section of Appendix 2 for the details.  $\square$

Using the above properties, we can state the following up-to technique.

**Lemma 1** (Up-to deterministic transition) *Let  $\Gamma; \Delta_1 \vdash P_1 \mathfrak{R} \Delta_2 \vdash Q_1$  such that if whenever:*

1.  $\forall (v \widetilde{m}_1)n!(V_1)$  such that  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(v \widetilde{m}_1)n!(V_1)} \Delta_3 \vdash P_3$  implies that  $\exists Q_2, V_2$  such that  $\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{(v \widetilde{m}_2)n!(V_2)} \Delta'_2 \vdash Q_2$  and  $\Gamma; \Delta_3 \vdash P_3 \xRightarrow{\tau_d} \Delta'_1 \vdash P_2$  and for a fresh name  $t$  and  $\Delta'_1, \Delta'_2$ :

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P_2 \mid t \leftarrow_H V_1) \mathfrak{R} \Delta'_2 \vdash (v \widetilde{m}_2)(Q_2 \mid t \leftarrow_H V_2)$$

2.  $\forall \ell \neq (v \widetilde{m})n!(V)$  such that  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_3 \vdash P_3$  implies that  $\exists Q_2$  such that  $\Gamma; \Delta_1 \vdash Q_1 \xrightarrow{\ell} \Delta'_2 \vdash Q_2$  and  $\Gamma; \Delta_3 \vdash P_3 \xRightarrow{\tau_d} \Delta'_1 \vdash P_2$  and  $\Gamma; \Delta'_1 \vdash P_2 \mathfrak{R} \Delta'_2 \vdash Q_2$ .
3. The symmetric cases of 1 and 2.

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

*Proof* (Sketch) The proof proceeds by considering the relation

$$\mathfrak{R} \xRightarrow{\tau_d} = \{\Gamma; \Delta'_1 \vdash P_2, \Delta'_2 \vdash Q_1 \mid \Gamma; \Delta_1 \vdash P_1 \mathfrak{R} \Delta'_2 \vdash Q_1, \Gamma; \Delta_1 \vdash P_1 \xRightarrow{\tau_d} \Delta'_1 \vdash P_2\}$$

We may verify that  $\mathfrak{R} \xRightarrow{\tau_d}$  is a higher-order bisimulation by using Proposition 2.  $\square$

### 5.8 Characterisation of higher-order and characteristic bisimilarities

This section proves the main result; it allows us to use  $\approx^C$  and  $\approx^H$  as tractable reasoning techniques for  $\text{HO}\pi$  processes.

**Lemma 2**  $\approx^C = \approx^H$ .

*Proof* (Sketch) The main difference between  $\approx^H$  and  $\approx^C$  is the trigger process (higher-order triggers  $t \leftarrow_H V$  in  $\approx^H$  and characteristic triggers  $t \leftarrow_C V : U$  in  $\approx^C$ ). Thus, the most interesting case in the proof is when we observe an output from a process. When showing that  $\approx^C \subseteq \approx^H$ , the key after the output is to show that

$$(v \widetilde{m}_1)(P_1 \mid t \leftarrow_C V : U) \approx^H (v \widetilde{m}_2)(P_2 \mid t \leftarrow_C V_2 : U)$$

given that

$$(\nu \tilde{m}_1)(P_1 \mid t \leftarrow_H V) \approx^H (\nu \tilde{m}_2)(P_2 \mid t \leftarrow_H V_2).$$

Similarly, in the proof of  $\approx^H \subseteq \approx^C$ , the key step is showing that

$$(\nu \tilde{m}_1)(P_1 \mid t \leftarrow_H V) \approx^C (\nu \tilde{m}_2)(P_2 \mid t \leftarrow_H V_2)$$

given that

$$(\nu \tilde{m}_1)(P_1 \mid t \leftarrow_C V : U) \approx^C (\nu \tilde{m}_2)(P_2 \mid t \leftarrow_C V_2 : U).$$

The proof for the above equalities is coinductive, exploiting the freshness of the trigger name in each case; see Lemma 13 in the Appendix. While the proof of the first equality (i.e., higher-order triggers imply characteristic triggers) follows expected lines, the proof of the second equality (i.e., characteristic triggers imply higher-order triggers) is a bit more involved. Indeed, while higher-order trigger processes can input trigger values, characteristic triggers cannot. However, we prove that this does not represent a difference in behaviour; see case 2(c) in Lemma 13. To this end, we exploit an alternative trigger process, denoted  $t \leftarrow_A V$ , simpler than the higher-order trigger  $t \leftarrow_H V$  in (6):

$$t \leftarrow_A V = t?(x).(v s)(x s \mid \bar{s}!\langle V \rangle.0)$$

In the proofs for these coincidence results, we exploit some auxiliary results for trigger processes, including a two-way connection between  $t \leftarrow_H V$  and  $t \leftarrow_A V$  (cf. Lemma 12 (3) in the Appendix). We thus infer that characteristic trigger processes  $t \leftarrow_C V : U$  and higher-order trigger processes  $t \leftarrow_H V$  exhibit a similar behaviour.

In turn, using the above results we can show that typed relations induced by  $\approx^H$  and  $\approx^C$  coincide. The full proof is in “Proof of Theorem 2” section in Appendix 2, Lemma 14.  $\square$

The next lemma is crucial for the characterisation of higher-order and characteristic bisimilarities. It states that if two processes are equivalent under the trigger value then they are equivalent under any higher-order substitution.

**Lemma 3** (Process substitution) *Let  $P$  and  $Q$  be two processes and some fresh  $t$ . If*

$$\Gamma; \Delta'_1 \vdash P\{\lambda x.t?(y).(y x)/z\} \approx^H \Delta'_2 \vdash Q\{\lambda x.t?(y).(y x)/z\}$$

*then for all  $R$  such that  $\text{fv}(R) = \{x\}$ , we have*

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

The full proof of Lemma 3 can be found in “Proof of Theorem 2” section in Appendix 2, Lemma 17; it is obtained by (i) constructing a typed relation on the substitution properties stated by the lemma and (ii) proving that it is a higher-order bisimulation, using the auxiliary result given next. In the following, given a finite index set  $I = \{1, \dots, n\}$ , we shall write  $\prod_{i \in I} P_i$  to stand for  $P_1 \mid P_2 \mid \dots \mid P_n$ .

**Lemma 4** (Trigger substitution) *Let  $P$  and  $Q$  be processes. Also, let  $t$  be a fresh name. If*

$$\begin{aligned} & \Gamma; \Delta_1 \vdash (\nu \tilde{m}_1) \left( P \mid \prod_{i \in I} (\lambda x.t_i?(y).(y x)) n_i \right) \\ & \approx^H \vdash \Delta_2 (\nu \tilde{m}_2) \left( Q \mid \prod_{i \in I} (\lambda x.t_i?(y).(y x)) m_i \right) \end{aligned}$$

then for all  $\lambda\tilde{x}. R$ , there exist  $\Delta'_1, \Delta'_2$  such that

$$\Gamma; \Delta'_1 \vdash (v \tilde{m}_1)(P \mid (\lambda\tilde{x}. R) \tilde{n}) \approx^H \Delta'_2 \vdash (v \tilde{m}_2)(Q \mid (\lambda\tilde{x}. R) \tilde{m}).$$

*Proof* (Sketch) The proof follows the definition of the characteristic process; see Lemma 16, in the Appendix for details. Let us consider the particular case in which  $I$  is a singleton; we then construct a typed relation  $\mathfrak{R}$ :

$$\mathfrak{R} = \{ \Gamma; \Delta'_1 \vdash (v \tilde{m}_1)(P \mid (\lambda x. R) n_1), \Delta'_2 \vdash (v \tilde{m}_2)(Q \mid (\lambda x. R) n_2) \mid \Gamma; \Delta_1 \vdash (v \tilde{m}_1)(P \mid (\lambda x. t?(y).(y x)) n_1) \approx^H \Delta_2 \vdash (v \tilde{m}_2)(Q \mid (\lambda x. t?(y).(y x)) n_2) \}$$

The typed relation  $\mathfrak{R}$  can be shown to be a higher-order bisimulation by taking advantage of the shape of the characteristic process; each time that a characteristic process does a transition, an output  $t!\langle n \rangle.\mathbf{0}$  (on a fresh name  $t$ ) is observed, where  $n$  is either a shared or a session name. To better illustrate this, let us sketch the demanding case of the proof that  $\mathfrak{R}$  is a higher-order bisimulation. Assume that

$$\Gamma; \emptyset; \Delta'_1 \vdash (v \tilde{m}_1)(P \mid R\{n_1/x\}) \xrightarrow{\tau_d} \xrightarrow{\ell_1} \Delta''_1 \vdash (v \tilde{m}'_1)(P' \mid R'\{n_1/x\})$$

for some  $\Delta''_1$ . Then, from the definition of  $\mathfrak{R}$ , we have:

$$\Gamma; \emptyset; \Delta_1 \vdash (v \tilde{m}_1)(P \mid (\lambda x. t?(y).(y x)) n_1) \xrightarrow{\tau_d} \xrightarrow{t?\{[U]_c\}} \xrightarrow{\tau_d} \Delta_3 \vdash (v \tilde{m}''_1)(P \mid [U]^{n_1})$$

for some  $\Delta_3$ . Characteristic processes have the following property, for any  $U \neq \text{end}$ :

$$[U]^n \xrightarrow{\ell} t!\langle n \rangle.\mathbf{0}$$

By the last property we can always observe, for some  $\Delta''_3$  (note that below  $\ell_1$  may be an action  $\tau$ , thus denoting the interaction of  $P$  and  $[U]^{n_1}$ ):

$$\Gamma; \emptyset; \Delta_3 \vdash (v \tilde{m}''_1)(P \mid [U]^{n_1}) \xrightarrow{\ell_1} (v \tilde{m}'''_1)(P' \mid t!\langle n_1 \rangle.\mathbf{0}) \xrightarrow{t!\langle n_1 \rangle} \Delta''_3 \vdash (v \tilde{m}'''_1)P'$$

which implies, from the requirements of higher-order bisimulation, that there exist  $(v \tilde{m}''_2)(Q' \mid [U]^x\{n_2/x\})$  and  $\Delta_4$  such that

$$\Gamma; \emptyset; \Delta_2 \vdash (v \tilde{m}_2)(Q \mid (\lambda x. t?(y).(y x)) n_2) \xrightarrow{\tau_d} \xrightarrow{t?\{[U]_c\}} \xrightarrow{\tau_d} \Delta_4 \vdash (v \tilde{m}''_2)(Q' \mid [U]^{n_2})$$

By the shape of the characteristic process we can always observe for  $\ell_2, \text{subj}(\ell_2) = \text{subj}(\ell_1)$  if  $\ell_1$  is output, and  $\ell_2 = \ell_1$  otherwise, that:

$$\Gamma; \emptyset; \Delta_4 \vdash (v \tilde{m}''_2)(Q' \mid [U]^x\{n_2/x\}) \xrightarrow{\ell_2} (v \tilde{m}'''_2)(Q'' \mid t!\langle n_2 \rangle.\mathbf{0}) \xrightarrow{t!\langle n_2 \rangle} \Delta'_4 \vdash (v \tilde{m}'''_2)Q'' \tag{8}$$

for some  $\Delta'_4$  and

$$\Gamma; \Delta''_3 \vdash (v \tilde{m}'''_1)(P' \mid t'' \leftarrow_H n_1) \approx^H \Delta''_4 \vdash (v \tilde{m}'''_2)(Q'' \mid t'' \leftarrow_H n_2) \tag{9}$$

for some  $\Delta''_4$ . From (8) we get

$$\Gamma; \emptyset; \Delta'_2 \vdash (v \tilde{m}_2)(Q \mid R\{n_2/x\}) \xrightarrow{\ell_2} \Delta''_2 \vdash (v \tilde{m}'_2)(Q'' \mid R'\{n_2/x\})$$



for some  $\Delta_2''$  and from (9) we get

$$\Gamma; \Delta_3'' \vdash (\nu \widetilde{m}_1''')(P' \mid (\lambda x. t''?(y).(y x)) n_1) \approx^H \Delta_4'' \vdash (\nu \widetilde{m}_2''')(Q'' \mid (\lambda x. t''?(y).(y x)) n_2)$$

which implies from the definition of  $\mathfrak{R}$  that for  $R'$  we get

$$\Gamma; \Delta_1'' \vdash (\nu \widetilde{m}_1')(P' \mid R'\{n_1/x\}) \mathfrak{R} \Delta_2'' \vdash (\nu \widetilde{m}_2')(Q'' \mid R'\{n_2/x\})$$

as required. □

We now show that higher-order bisimilarity is sound with respect to context bisimilarity. To show soundness we use the crucial result of Lemma 3:

**Lemma 5**  $\approx^H \subseteq \approx$ .

*Proof (Sketch)* The proof relies on Lemma 3 to establish that:

1. Whenever two processes are higher-order bisimilar under the input of a characteristic value and a trigger value then they are higher-order bisimilar under the input of any value  $\lambda x. R$ , which is the requirement for  $\approx$  (cf. Definition 12).
2. The input requirement is then further used to prove that the output clause requirement for  $\approx^H$  (cf. Definition 17):

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P_2 \mid t \leftrightarrow_H V_1) \mathfrak{R} \Delta_2 \vdash (\nu \widetilde{m}_2)(Q_2 \mid t \leftrightarrow_H V_2)$$

implies the output clause requirement for  $\approx$ , that is, for all  $R$  with  $\text{fv}(R) = \{x\}$ :

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P_2 \mid R\{V_1/x\}) \mathfrak{R} \Delta_2 \vdash (\nu \widetilde{m}_2)(Q_2 \mid R\{V_2/x\}).$$

The full proof is found in “Proof of Theorem 2” section in Appendix 2, Lemma 18. □

Context bisimilarity is included in barbed congruence:

**Lemma 6**  $\approx \subseteq \cong$ .

*Proof (Sketch)* We show that  $\approx$  satisfies the defining properties of  $\cong$ . It is easy to show that  $\approx$  is reduction-closed and barb preserving (cf. Definition 6 and Definition 9). The most challenging part is to show that  $\approx$  is a congruence, in particular a congruence with respect to parallel composition. To this end, we construct the following relation:

$$\begin{aligned} S = \{ & (\Gamma; \emptyset; \Delta_1 \cdot \Delta_3 \vdash (\nu \widetilde{n}_1)(P_1 \mid R), \Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash (\nu \widetilde{n}_2)(P_2 \mid R)) \mid \\ & \Gamma; \Delta_1 \vdash P_1 \approx \Delta_2 \vdash P_2 \text{ and } \forall R \text{ such that } \Gamma; \emptyset; \Delta_3 \vdash R \triangleright \diamond \} \end{aligned}$$

We show that  $S$  is a context bisimulation by a case analysis on the transitions of the pairs in  $S$ . The full proof is found in “Proof of Theorem 2” section in Appendix 2, Lemma 19. □

The last ingredient required for our main result is the following inclusion.

**Lemma 7**  $\cong \subseteq \approx^H$ .

*Proof (Sketch)* The proof exploits the *definability* technique developed in [8, §6.7] and refined for session types in [18, 19]. Intuitively, this technique exploits small test processes that reveal the presence of a visible action by reducing with a given pair of processes and exhibiting a barb on a fresh name.

Intuitively, for each visible action  $\ell$ , we use a fresh name  $succ$  to we define a (typed) test process  $\Gamma; \emptyset; \Delta_2 \vdash T\langle \ell, succ \rangle \triangleright \diamond$  with the following property:

$$\Gamma; \Delta_1 \vdash P \mid T\langle \ell, succ \rangle \longrightarrow \Delta_2 \vdash P' \mid succ!(\overline{m}).\mathbf{0} \downarrow_{succ} \quad \text{iff} \quad \Gamma; \Delta \vdash P \xrightarrow{\ell} \Gamma; \Delta' \vdash P'$$

See Definition 25 for the formal definition. The test processes can therefore be used to check the typed labelled transition interactions of two processes that are related by reduction-closed, barbed congruence. Indeed, we have that

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

implies from congruence of  $\cong$ , that if there exist  $\Delta_3, \Delta_4$  such that:

$$\Gamma; \Delta_3 \vdash P \mid T\langle \ell, succ \rangle \cong \Delta_4 \vdash Q \mid T\langle \ell, succ \rangle$$

then it implies from reduction-closeness of  $\cong$  and the definition of  $T\langle \ell, succ \rangle$ :

$$\Gamma; \Delta'_3 \vdash P' \mid succ!(\overline{m}).\mathbf{0} \cong \Delta'_4 \vdash Q' \mid succ!(\overline{m}).\mathbf{0} \tag{10}$$

which in turn means that whenever  $\Gamma; \Delta_1 \vdash P \triangleright \diamond$  can perform an action  $\xrightarrow{\ell}$  then we can derive that  $\Gamma; \Delta_2 \vdash Q \triangleright \diamond$  can also perform action  $\xrightarrow{\ell}$  because of the result in (10). By applying Lemma 21 on (10) we can deduce that  $\Gamma; \Delta'_1 \vdash P' \cong \Delta'_2 \vdash Q'$ . This concludes the requirements of  $\approx$ :

$$\Gamma; \Delta \vdash P \approx^H \Delta' \vdash Q$$

The full details can be found in ‘‘Proof of Theorem 2’’ section in Appendix 2, Lemma 22. □

We can finally state our main result:

**Theorem 2** (Coincidence)  $\cong, \approx, \approx^H$  and  $\approx^C$  coincide in  $HO\pi$ .

*Proof* The proof is a direct consequence from our previous results: Lemma 2 (which proves  $\approx^H = \approx^C$ ), Lemma 5 (which proves  $\approx^H \subseteq \approx$ ), Lemma 6 (which proves  $\approx \subseteq \cong$ ), and Lemma 7 (which proves  $\cong \subseteq \approx^H$ ). Indeed, we may conclude

$$\cong \subseteq \approx^H = \approx^C \subseteq \approx \subseteq \cong$$

□

### 5.9 Revisiting the hotel booking scenario (Sect. 3.3)

Now we revisit our running example to prove that  $Client_1$  and  $Client_2$  in Sect. 3.3 are behaviourally equivalent.

**Proposition 3** Let  $S = !(room); ?(quote); \oplus\{accept : !(credit); end, reject : end\}$  and  $\Delta = s_1 : !(S \multimap \diamond); end \cdot s_2 : !(S \multimap \diamond); end$ . Then  $\emptyset; \Delta \vdash Client_1 \approx^C \Delta \vdash Client_2$ , where  $Client_1$  and  $Client_2$  are as in Sect. 3.3.

*Proof* We show a case where each typed process simulates the other, according to the definition of  $\approx^C$  (cf. Definition 18). In order to show the bisimulation game consider the definition of the characteristic process for type  $?(S \multimap \diamond); end$ . For fresh sessions  $s, k$ , we have

$$[?(S \multimap \diamond); end]^s = s?(x).(t!(s).\mathbf{0} \mid [S \multimap \diamond]^x)$$

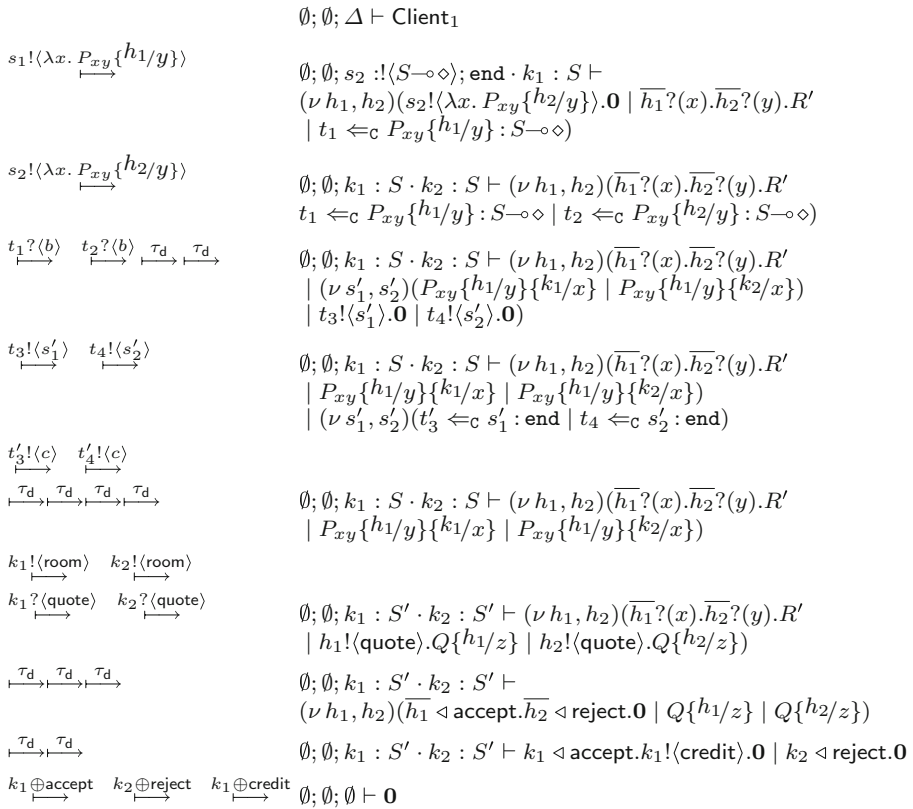


Fig. 7 Observable actions from Client<sub>1</sub> (cf. Sect. 5.9)

For convenience, we recall the definition of Client<sub>1</sub>:

$$\text{Client}_1 \stackrel{\text{def}}{=} (\nu h_1, h_2)(s_1! \langle \lambda x. P_{xy} \{h_1/y\} \rangle. s_2! \langle \lambda x. P_{xy} \{h_2/y\} \rangle. \mathbf{0} \mid \overline{h_1}?(x).\overline{h_2}?(y).R')$$

where

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

$$R' \equiv \text{if } x \leq y \text{ then } \overline{h_1} \triangleleft \text{accept}.\overline{h_2} \triangleleft \text{reject}.\mathbf{0} ; \overline{h_1} \triangleleft \text{reject}.\overline{h_2} \triangleleft \text{accept}.\mathbf{0}$$

Also, the definition of Client<sub>2</sub> is as follows:

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

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

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

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

A detailed account of the observable behaviour of Client<sub>1</sub> is given in Fig. 7, where we use the following shorthand notation:

	$\emptyset; \emptyset; \Delta \vdash \text{Client}_2$
$s_1! \langle \lambda x. Q_1 \{h/y\} \rangle$	$\emptyset; \emptyset; s_2 :!(S \multimap \diamond); \text{end} \cdot k_1 : S \vdash (\nu h)(s_2! \langle \lambda x. Q_2 \{\bar{h}/y\} \rangle. \mathbf{0} \mid t_1 \leftarrow c Q_1 \{h/y\} : S \multimap \diamond)$
$s_2! \langle \lambda x. Q_2 \{\bar{h}/y\} \rangle$	$\emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu h)(t_1 \leftarrow c Q_1 \{h/y\} : S \multimap \diamond \mid t_2 \leftarrow c Q_2 \{\bar{h}/y\} : S \multimap \diamond)$
$t_1? \langle b \rangle \quad t_2? \langle \bar{b} \rangle \quad \xrightarrow{\tau_d} \xrightarrow{\tau_d}$	$\emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu h)(P\{h/y\}\{k_1/x\} \mid P_{xy}\{\bar{h}/y\}\{k_2/x\} \mid (\nu s'_1, s'_2)(t_3! \langle s'_1 \rangle. \mathbf{0} \mid t_4! \langle s'_2 \rangle. \mathbf{0}))$
$t_3! \langle s'_1 \rangle \quad t_4! \langle s'_2 \rangle$	$\emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu h)(P\{h/y\}\{k_1/x\} \mid P_{xy}\{\bar{h}/y\}\{k_2/x\} \mid (\nu s'_1, s'_2)(t'_3 \leftarrow c s'_1 : \text{end} \mid t'_4 \leftarrow c s'_2 : \text{end}))$
$t'_3! \langle c \rangle \quad t'_4! \langle c \rangle$	
$\xrightarrow{\tau_d} \xrightarrow{\tau_d} \xrightarrow{\tau_d} \xrightarrow{\tau_d}$	$\emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu h)(P\{h/y\}\{k_1/x\} \mid P_{xy}\{\bar{h}/y\}\{k_2/x\})$
$k_1! \langle \text{room} \rangle \quad k_2! \langle \text{room} \rangle$	
$k_1? \langle \text{quote} \rangle \quad k_2? \langle \text{quote} \rangle$	$\emptyset; \emptyset; k_1 : S' \cdot k_2 : S' \vdash (\nu h)(h! \langle \text{quote}_1 \rangle. h? \langle \text{quote}_2 \rangle. R\{k_1/x\} \mid \bar{h}? \langle \text{quote}_2 \rangle. \bar{h}! \langle \text{quote}_1 \rangle. R\{k_2/x\})$
$\xrightarrow{\tau_d} \xrightarrow{\tau_d}$	$\emptyset; \emptyset; k_1 : S' \cdot k_2 : S' \vdash R\{k_1/x\} \mid R\{k_2/x\}$
$\xrightarrow{\tau_d} \xrightarrow{\tau_d}$	$\emptyset; \emptyset; k_1 : S' \cdot k_2 : S' \vdash k_1 \triangleleft \text{accept}. k_1! \langle \text{credit} \rangle. \mathbf{0} \mid k_2 \triangleleft \text{reject}. \mathbf{0}$
$k_1 \oplus \text{accept} \quad k_2 \oplus \text{reject} \quad k_1 \oplus \text{credit}$	$\emptyset; \emptyset; \emptyset \vdash \mathbf{0}$

**Fig. 8** Observable actions from  $\text{Client}_2$  (cf. Sect. 5.9)

$$\mathcal{Q} \equiv z \triangleright \{ \text{accept} : k_2 \triangleleft \text{accept}. k_2! \langle \text{credit} \rangle. \mathbf{0}, \text{reject} : k_2 \triangleleft \text{reject}. \mathbf{0} \}$$

Similarly, Fig. 8 illustrates the actions possible from  $\text{Client}_2$ , which are the same as for  $\text{Client}_1$ .  $\square$

## 6 Related work

Since types can limit contexts (environments) where processes can interact, typed equivalences usually offer *coarser* semantics than untyped equivalences. Pierce and Sangiorgi [28] demonstrated that IO-subtyping can justify the optimal encoding of the  $\lambda$ -calculus by Milner—this was not possible in the untyped polyadic  $\pi$ -calculus [23]. After [28], several works on typed  $\pi$ -calculi have investigated correctness of encodings of known concurrent and sequential calculi in order to examine semantic effects of proposed typing systems.

A type discipline closely related to session types is a family of linear typing systems. Kobayashi, Pierce, and Turner [14] first proposed a linearly typed reduction-closed, barbed congruence and used it to reason about a tail-call optimisation of higher-order functions encoded as processes. Yoshida [35] used a bisimulation of graph-based types to prove the full abstraction of encodings of the polyadic synchronous  $\pi$ -calculus into the monadic synchronous  $\pi$ -calculus. Later, typed equivalences of a family of linear and affine calculi [2, 3, 36] were used to encode PCF [22, 29], the simply typed  $\lambda$ -calculus with sums and products, and System F [6] fully abstractly (a fully abstract encoding of the  $\lambda$ -calculi was an open problem in [23]). Yoshida et al. [37] proposed a new bisimilarity method associated with a linear

type structure and strong normalisation; it presented applications to reason about secrecy in programming languages. A subsequent work [11] adapted these results to a practical direction, proposing new typing systems for secure higher-order and multi-threaded programming languages. In these works, typed properties, linearity and liveness, play a fundamental rôle in the analysis. In general, linear types are suitable to encode “sequentiality” in the sense of [1, 12].

Our work follows the behavioural semantics in [18, 19, 27] where a bisimulation is defined on an LTS that assumes a session typed observer. Our theory for higher-order sessions differentiates from the work in [19] and [18], which considers (first-order) binary and multiparty session types, respectively. Pérez et al [27] studied typed equivalences for a theory of binary sessions based on linear logic, without shared names.

Our approach to typed equivalences builds upon techniques developed by Sangiorgi [30, 31] and Jeffrey and Rathke [13]. As we have discussed, although context bisimilarity has a satisfactory discriminative power, its use is hindered by the universal quantification on output. To deal with this, Sangiorgi proposes *normal bisimilarity*, a tractable equivalence without universal quantification. To prove that context and normal bisimilarities coincide, the approach in [30] uses triggered processes. Triggered bisimulation is also defined on first-order labels where context bisimulation is restricted to arbitrary trigger substitution. This characterisation of context bisimilarity was refined in [13] for calculi with recursive types, not addressed in [30, 31] and quite relevant in session-based concurrency. The bisimulation in [13] is based on an LTS extended with trigger meta-notation. As in [30, 31], the LTS in [13] observes first-order triggered values instead of higher-order values, offering a more direct characterisation of contextual equivalence and lifting the restriction to finite types. *Environmental bisimulations* [32] use a higher-order LTS to define a bisimulation that stores the observer’s knowledge; hence, observed actions are based on this knowledge at any given time. This approach is enhanced in [15] with a mapping from constants to higher-order values. This allows to observe first-order values instead of higher-order values. It differs from [13, 31] in that the mapping between higher- and first-order values is no longer implicit.

*Comparison with respect to [13]* We briefly contrast the approach by Jeffrey and Rathke [13] and our approach based on characteristic bisimilarity ( $\approx^c$ ):

- The LTS in [13] is enriched with extra labels for triggers; an output action transition emits a trigger and introduces a parallel replicated trigger. Our approach retains usual labels/transitions; in case of output,  $\approx^c$  introduces a parallel *non-replicated* trigger.
- Higher-order input in [13] involves the input of a trigger which reduces after substitution. Rather than a trigger name,  $\approx^c$  decrees the input of a trigger value  $\lambda z. t?(x).(x z)$ .
- Unlike [13],  $\approx^c$  treats first- and higher-order values uniformly. As the typed LTS distinguishes linear and shared values, replicated closures are used only for shared values.
- In [13] name matching is crucial to prove completeness of bisimilarity. In our case,  $\text{HO}\pi$  lacks name matching and we use session types: a characteristic value inhabiting a type enables the simplest form of interactions with the environment.

To further compare our approach to that in [13], we use a representative example.

*Example 6* Let  $V = \lambda x. x (\lambda y. y!(m).\mathbf{0})$  be a value. Consider a process such that

$$\Gamma; \emptyset; \Delta \cdot n :!(U); \text{end} \vdash n!(V).\mathbf{0} \triangleright \diamond$$

with  $U = (((!(S); \text{end}) \rightarrow \diamond) \rightarrow \diamond) \rightarrow \diamond$ . We contrast the transitions from  $P$ . In our framework, we have a typed transition  $\Gamma; \emptyset; \Delta \cdot n :!(U); \text{end} \vdash P \xrightarrow{n!(V)} \Gamma; \emptyset; \Delta \vdash \mathbf{0}$ . In the framework

$$\begin{aligned}
 & t \leftarrow_c V : U \\
 = & \Gamma; \emptyset; \Delta \vdash t?(z).(\nu s)(s?(x).[U]^x \mid \bar{s}!\langle \lambda x. x (\lambda y. y!\langle m \rangle. \mathbf{0}) \rangle. \mathbf{0}) \\
 = & \Gamma; \emptyset; \Delta \vdash t?(z).(\nu s)(s?(x).x (\lambda y. (y a)) \mid \bar{s}!\langle \lambda x. x (\lambda y. y!\langle m \rangle. \mathbf{0}) \rangle. \mathbf{0}) \\
 (1) \quad & \xrightarrow{t?(b)} \Gamma; \emptyset; \Delta \vdash (\nu s)(s?(x).x (\lambda y. (y a)) \mid \bar{s}!\langle \lambda x. x (\lambda y. y!\langle m \rangle. \mathbf{0}) \rangle. \mathbf{0}) \\
 (2) \quad & \xrightarrow{\tau_d} \Gamma; \emptyset; \Delta \vdash \lambda x. x (\lambda y. y!\langle m \rangle. \mathbf{0}) (\lambda y. (y a)) \\
 (3) \quad & \xrightarrow{\tau_d} \Gamma; \emptyset; \Delta \vdash (\lambda y. (y a)) (\lambda y. y!\langle m \rangle. \mathbf{0}) \\
 (4) \quad & \xrightarrow{\tau_d} \Gamma; \emptyset; \Delta \vdash (\lambda y. y!\langle m \rangle. \mathbf{0}) a \\
 (5) \quad & \xrightarrow{\tau_d} \Gamma; \emptyset; \Delta \vdash a!\langle m \rangle. \mathbf{0}
 \end{aligned}$$

(a) Our approach.

$$\begin{aligned}
 & \Gamma; \emptyset; \Delta \vdash *t?(x).x (\lambda y. y!\langle m \rangle. \mathbf{0}) \\
 (1) \quad & \xrightarrow{t?( \tau_l )} \Gamma; \emptyset; \Delta \vdash *t?(x).x (\lambda y. y!\langle m \rangle. \mathbf{0}) \mid (\lambda x. x (\lambda y. y!\langle m \rangle. \mathbf{0})) \tau_l \\
 (2) \quad & \xrightarrow{\tau_d} \Gamma; \emptyset; \Delta \vdash *t?(x).x (\lambda y. y!\langle m \rangle. \mathbf{0}) \mid \tau_l (\lambda y. y!\langle m \rangle. \mathbf{0}) \\
 (3) \quad & \xrightarrow{(\nu k)l!( \tau_k )} \Gamma; \emptyset; \Delta \vdash *t?(x).x (\lambda y. y!\langle m \rangle. \mathbf{0}) \mid *k?(y).y!\langle m \rangle. \mathbf{0} \\
 (4) \quad & \xrightarrow{k?(a)} \Gamma; \emptyset; \Delta \vdash *t?(x).x (\lambda y. y!\langle m \rangle. \mathbf{0}) \mid *k?(y).y!\langle m \rangle. \mathbf{0} \mid (\lambda y. y!\langle m \rangle. \mathbf{0}) a \\
 (5) \quad & \xrightarrow{\tau_d} \Gamma; \emptyset; \Delta \vdash *t?(x).x (\lambda y. y!\langle m \rangle. \mathbf{0}) \mid *k?(y).y!\langle m \rangle. \mathbf{0} \mid a!\langle m \rangle. \mathbf{0}
 \end{aligned}$$

(b) Jeffrey and Rathke’s approach [13].

Fig. 9 Comparing labelled transitions associated to the process in Example 6

of [13] a similar (but untyped) output transition takes place. Figure 9 presents a complete comparison of the labelled transitions in our approach (Fig. 9a) and in [13] (Fig. 9b). In our approach, we let

$$[U]^x = x (\lambda y. (y a)) \quad \text{for some fresh } a$$

Then we have one input transition (Line (1)), followed by four deterministic internal transitions; no replicated processes are needed. The approach of [13] also uses five transitions, but more visible transitions are required (three, see Lines (1), (2), and (3) in Fig. 9b) and at the end, two replicated processes remain (on  $t$  and  $k$ ). This is how linearity information in session types enables simpler bisimulations. Note that  $\tau_l$  and  $\tau_k$  in Lines (1) and (3) denote triggered processes on names  $l$  and  $k$ .

The previous comparison shows how our approach requires less visible transitions and replicated processes. Therefore, linearity information does simplify analyses, as it enables simpler witnesses in coinductive proofs.

### 7 Concluding remarks

Obtaining tractable characterisations of contextual equivalence is a long-standing issue for higher-order languages. In this paper, we have addressed this challenge for a higher-order language which integrates functional constructs and features from concurrent processes (name and process passing), and whose interactions are governed by *session types*, a behavioural type discipline for structured communications. The main result of our study is the development of *characteristic bisimilarity*, a relation on session typed processes which fully characterises contextual equivalence.

Compared to the well-known context bisimilarity, our notion of characteristic bisimilarity enables more tractable analyses without sacrificing distinguishing power. Our approach to simplified analysis rests upon two simple mechanisms. First, using *trigger processes* we lighten the requirements involved in output clauses. In particular, we can lift the need for heavy universal quantifications. Second, using *characteristic processes and values* we refine the requirements for input clauses. Formally supported by a refined LTS, the use of characteristic processes and values effectively narrows down input actions. Session type information (which includes linearity requirements on reciprocal communications), naturally available in scenarios of interacting processes, is crucial to define these two new mechanisms, and therefore to enable technical simplifications in our developments. As already discussed, our coincidence result is insightful also in the light of previous works on labelled equivalences for higher-order processes, in particular with respect to characterisations by Sangiorgi [30,31] and by Jeffrey and Rathke [13]. Our main result combines several technical innovations, including, e.g., up-to techniques for deterministic behaviours (cf. Lemma 1) and an alternative behavioural equivalence, called *higher-order bisimilarity* (denoted  $\approx^H$ , cf. Definition 17), which uses simpler trigger processes and is applicable to processes without first-order passing.

In addition to their intrinsic significance, our study has important consequences and applications in other aspects of the theory of higher-order processes. In particular, we have recently explored the *relative expressivity* of higher-order sessions [17]. Both characteristic and higher-order bisimilarities play an important rôle in establishing tight correctness properties (e.g., operational correspondence and full abstraction) for encodability results connecting different variants of  $HO\pi$ . Such variants cover features such as pure process passing (with first- and higher-order abstractions), pure name passing, polyadicity, linear/shared communication.

**Acknowledgements** We are grateful to the CONCUR’15 reviewers and attendees for their valuable feedback. We are also most grateful to the Acta Informatica reviewers for their many detailed and insightful remarks, which greatly helped us to improve this document. This work has been partially sponsored by the Doctoral Prize Fellowship, EPSRC EP/K011715/1, EPSRC EP/K034413/1, EPSRC EP/N027833/1 and EPSRC EP/L00058X/1, EU project FP7-612985 UpScale, and EU COST Actions IC1201 (BETTY), IC1402 (ARVI), and IC1405 (Reversible Computation). Pérez is also affiliated to the NOVA Laboratory for Computer Science and Informatics (NOVA LINCS, Ref. PESt/UID/CEC/04516/2013), Universidade Nova de Lisboa, Portugal.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## Appendix 1: The typing system of $HO\pi$

We first formally define *type equivalence*. Then we give details of the proof of Theorem 1.

### Type equivalence

**Definition 21** (*Type equivalence*) Let  $ST$  be 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  $F : \mathcal{P}(ST \times ST) \rightarrow \mathcal{P}(ST \times ST)$  defined by:

$$F(\mathfrak{R}) = \{(\text{end}, \text{end})\} \cup \{(!\langle U_1 \rangle; S_1, !\langle U_2 \rangle; S_2) \mid (S_1, S_2), (U_1, U_2) \in \mathfrak{R}\}$$

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

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

### Proof of Theorem 1 (type soundness)

As our type system is closely related to that considered by Mostrous and Yoshida [26], the proof of type soundness requires notions and properties which are instances of those already shown in [26]. We first state weakening and strengthening lemmas, which have standard proofs.

**Lemma 8** (Weakening—Lemma C.2 in [26])

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

**Lemma 9** (Strengthening—Lemmas C.3 and C.4 in [26]) *We have:*

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

Below, shared value means that there are no free linear names, thus  $\Lambda, \Delta$  are empty (cf. Rule [Prom] in Fig. 3).

**Lemma 10** (Substitution Lemma—Lemma C.10 in [26]) *We have:*

1. Suppose  $\Gamma; \Lambda; \Delta \cdot x : S \vdash P \triangleright \diamond$  and  $s \notin \text{dom}(\Gamma, \Lambda, \Delta)$ .  
Then  $\Gamma; \Lambda; \Delta \cdot s : S \vdash P\{s/x\} \triangleright \diamond$ .
2. Suppose  $\Gamma \cdot x : \langle U \rangle; \Lambda; \Delta \vdash P \triangleright \diamond$  and  $a \notin \text{dom}(\Gamma, \Lambda, \Delta)$ .  
Then  $\Gamma \cdot a : \langle U \rangle; \Lambda; \Delta \vdash P\{a/x\} \triangleright \diamond$ .
3. Suppose  $\Gamma; \Lambda_1 \cdot x : U \rightarrow \diamond; \Delta_1 \vdash P \triangleright \diamond$  and  $\Gamma; \Lambda_2; \Delta_2 \vdash V \triangleright U \rightarrow \diamond$  with  $\Lambda_1, \Lambda_2$  and  $\Delta_1, \Delta_2$  defined. Then  $\Gamma; \Lambda_1 \cdot \Lambda_2; \Delta_1 \cdot \Delta_2 \vdash P\{V/x\} \triangleright \diamond$ .
4. Suppose  $\Gamma \cdot x : U \rightarrow \diamond; \Lambda; \Delta \vdash P \triangleright \diamond$  and shared value  $V$  such that  $\Gamma; \emptyset; \emptyset \vdash V \triangleright U \rightarrow \diamond$ . Then  $\Gamma; \Lambda; \Delta \vdash P\{V/x\} \triangleright \diamond$ .

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

We now state the instance of type soundness that we can derive from [26]. It is worth noticing the definition of structural congruence in [26] is richer than ours. Also, their statement for subject reduction relies on an ordering on typings, associated to queues and other runtime elements. Since we are working with synchronous communication this ordering can be omitted. The second part of the following statement corresponds to Theorem 1:

**Theorem 3** (Type soundness) *We have:*

1. (Subject congruence) Suppose  $\Gamma; \Lambda; \Delta \vdash P \triangleright \diamond$ . Then  $P \equiv P'$  implies  $\Gamma; \Lambda; \Delta \vdash P' \triangleright \diamond$ .
2. (Subject reduction) Suppose  $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$  with balanced  $\Delta$ . Then  $P \longrightarrow P'$  implies  $\Gamma; \emptyset; \Delta' \vdash P' \triangleright \diamond$  and  $\Delta = \Delta' \text{ or } \Delta \longrightarrow \Delta'$ .



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

1. Case [App]: Then we have

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

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

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

Then, by combining premise  $\Gamma; \emptyset; \Delta \cdot \{x : S\} \vdash Q \triangleright \diamond$  with the substitution lemma (Lemma 10(1)), we obtain  $\Gamma; \emptyset; \Delta \cdot u : S \vdash Q\{u/x\} \triangleright \diamond$ , as desired.

2. Case [Pass]: There are several sub-cases, depending on the type of the communication subject  $n$  (which could be a shared or a linear name) and the type of the object  $V$  (which could be an abstraction or a shared/linear name). We analyse two representative sub-cases:

- (a)  $n$  is a shared name and  $V$  is a name  $v$ . Then we have the following reduction:

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

By assumption, we have the following typing derivation:

$$\frac{(11) \quad (12)}{\Gamma; \emptyset; \Delta_1 \cdot \{v : S\} \cdot \Delta_3 \vdash n!\langle v \rangle. Q_1 \mid n?(x). Q_2 \triangleright \diamond}$$

where (11) and (12) are as follows:

$$\frac{\Gamma' \cdot n : \langle S \rangle; \emptyset; \emptyset \vdash n \triangleright \langle S \rangle \quad \Gamma; \emptyset; \Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma; \emptyset; \{v : S\} \vdash v \triangleright S}{\Gamma; \emptyset; \Delta_1 \cdot \{v : S\} \vdash n!\langle v \rangle. Q_1 \triangleright \diamond} \quad (11)$$

$$\frac{\Gamma' \cdot n : \langle S \rangle; \emptyset; \emptyset \vdash n \triangleright \langle S \rangle \quad \Gamma; \emptyset; \Delta_3 \cdot x : S \vdash Q_2 \triangleright \diamond}{\Gamma; \emptyset; \Delta_3 \vdash n?(x). Q_2 \triangleright \diamond} \quad (12)$$

Now, by applying Lemma 10(1) on  $\Gamma; \emptyset; \Delta_3 \cdot x : S \vdash Q_2 \triangleright \diamond$  we obtain

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

and the case is completed by using Rule [Par] with this judgement:

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

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

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

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

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

$$\frac{(13) \quad (14)}{\Gamma; \emptyset; \Delta_1 \cdot \Delta_3 \vdash n!\langle v \rangle. Q_1 \mid n?(x). Q_2 \triangleright \diamond}$$

where (13) and (14) are as follows:

$$\frac{\Gamma; \emptyset; \emptyset \vdash n \triangleright \langle L \rangle \quad \Gamma; \emptyset; \Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma; \emptyset; \emptyset \vdash V \triangleright L}{\Gamma; \emptyset; \Delta_1 \vdash n!(V).Q_1 \triangleright \diamond} \quad (13)$$

$$\frac{\Gamma'; \emptyset; \emptyset \vdash n \triangleright \langle L \rangle \quad \Gamma'; \emptyset; \Delta_3 \vdash Q_2 \triangleright \diamond \quad \Gamma'; \emptyset; \emptyset \vdash x \triangleright L}{\Gamma; \emptyset; \Delta_3 \vdash n?(x).Q_2 \triangleright \diamond} \quad (14)$$

Now, by applying Lemma 10(4) on  $\Gamma' \setminus x; \emptyset; \Delta_3 \vdash Q_2 \triangleright \diamond$  and  $\Gamma; \emptyset; \emptyset \vdash V \triangleright L$  we obtain

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

and the case is completed by using Rule [Par] with this judgement:

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

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

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

## Appendix 2: Proofs for Sect. 5

### Typability of characteristic processes

We state and prove a more detailed form of Proposition 1. The case of recursive session types requires the following two auxiliary definitions for session type unfolding and prefix deletion.

**Definition 22** (*Session type unfolding*) Given a session type  $S$ , the function  $\text{unfold}(S)$  is defined as:

$$\begin{aligned} \text{unfold}(!\langle U \rangle; S) &= !\langle U \rangle; S & \text{unfold}?(U); S &= ?(U); S \\ \text{unfold}(\oplus\{l_i : S_i\}_{i \in I}) &= \oplus\{l_i : S_i\}_{i \in I} & \text{unfold}(\&\{l_i : S_i\}_{i \in I}) &= \&\{l_i : S_i\}_{i \in I} \\ \text{unfold}(\mu t.S) &= \text{unfold}(S\{\mu t.S/t\}) & \text{unfold}(\text{end}) &= \text{end} \end{aligned}$$

**Lemma 11** *Let  $S$  be a session type. Then  $\text{unfold}(S) = S'$  and  $S' \neq \mu t.S''$ .*

*Proof* A straightforward induction on the syntax of  $S$ . □

We define a relation for session type prefix deletion:

**Definition 23** (*Session type prefix deletion*) Given a session type  $S$ , the set  $\text{del}(S)$  is defined inductively as follows:

$$\begin{aligned} \text{del}(!\langle U \rangle; S) &= \{S\} & \text{del}?(U); S &= \{S\} \\ \text{del}(\oplus\{l_i : S_i\}_{i \in I}) &= \{S_i\}_{i \in I} & \text{del}(\&\{l_i : S_i\}_{i \in I}) &= \{S_i\}_{i \in I} \\ \text{del}(\mu t.S) &= \text{del}(\text{unfold}(\mu t.S)) & \text{del}(\text{end}) &= \{\text{end}\} \end{aligned}$$

We may now finally state and prove the following proposition:

**Proposition 4** (Characteristic processes/values inhabit their types)

1. Let  $U$  and  $[U]_c$  be a type and its characteristic value, respectively.

- (a) If  $U = S$  then, for some  $s$ , we have  $\emptyset; \emptyset; s : S \vdash [S]_c \triangleright S$ .
- (b) If  $U = \langle S \rangle$  then, for some  $a$ , we have  $a : \langle S \rangle; \emptyset; \emptyset \vdash [\langle S \rangle]_c \triangleright \langle S \rangle$ .

- (c) If  $U = \langle L \rangle$  then, for some  $a$ , we have  $a : \langle L \rangle; \emptyset; \emptyset \vdash [\langle L \rangle]_{\mathbf{c}} \triangleright \langle L \rangle$ .
- (d) If  $U = U' \rightarrow \diamond$  and  $\Gamma; \emptyset; \Delta \vdash [U']^x \triangleright \diamond$  then we have  $\Gamma \setminus x; \emptyset; \Delta \setminus x \vdash [U' \rightarrow \diamond]_{\mathbf{c}} \triangleright U' \rightarrow \diamond$ .
- (e) If  $U = U' \multimap \diamond$  and  $\Gamma; \emptyset; \Delta \vdash [U']^x \triangleright \diamond$  then we have  $\Gamma \setminus x; \emptyset; \Delta \setminus x \vdash [U' \multimap \diamond]_{\mathbf{c}} \triangleright U' \multimap \diamond$ .

2. Let  $S$  and  $[S]^s$  be a session type and its characteristic process, respectively.

- (a) If  $S = \text{end}$  then  $\emptyset; \emptyset; \emptyset \vdash [\text{end}]^s \triangleright \diamond$ .
- (b) If  $S = !\langle U \rangle; S'$  and  $\Gamma; \emptyset; \Delta \vdash [U]_{\mathbf{c}} \triangleright U$  then  $\Gamma; \emptyset; \Delta \cdot t : !\langle S' \rangle; \text{end} \cdot s : !\langle U \rangle; S' \vdash [!\langle U \rangle; S']^s \triangleright \diamond$ .
- (c) If  $S = ?\langle U \rangle; S'$  and  $\Gamma; \emptyset; \Delta \vdash [U]^x \triangleright \diamond$  then  $\Gamma \setminus x; \emptyset; (\Delta \setminus x) \cdot t : ?\langle S' \rangle; \text{end} \cdot s : !\langle U \rangle; S' \vdash [?\langle U \rangle; S']^s \triangleright \diamond$ .
- (d) If  $S = \oplus \{l_i : S_i\}_{i \in I}$  then  $\emptyset; \emptyset; \{t_i : !\langle S_i \rangle; \text{end}\}_{i \in I} \cdot s : \oplus \{l_i : S_i\}_{i \in I} \vdash [\oplus \{l_i : S_i\}_{i \in I}]^s \triangleright \diamond$ .
- (e) If  $S = \& \{l_i : S_i\}_{i \in I}$  then  $\emptyset; \emptyset; \{t_i : !\langle S_i \rangle; \text{end}\}_{i \in I} \cdot s : \& \{l_i : S_i\}_{i \in I} \vdash [\& \{l_i : S_i\}_{i \in I}]^s \triangleright \diamond$ .
- (f) If  $S = \mu t. S'$  then either
- $\emptyset; \emptyset; \emptyset \vdash [\mu t. S']^s \triangleright \diamond$
  - for all  $S_i \in \text{del}(S)$  there exist  $\Gamma, \Delta$ , and  $S'_i$  such that

$$\Gamma; \emptyset; \Delta \cdot \{t_i : S'_i\}_{i \in I} \cdot s : S' \{ \text{end}/t \} \vdash [S' \{ \text{end}/t \}]^s \triangleright \diamond$$

$$\text{and } \Gamma; \emptyset; \Delta \cdot \{t_i : !\langle S_i \rangle; \text{end}\}_{i \in I} \cdot s : \mu t. S' \vdash [\mu t. S']^s \triangleright \diamond.$$

3. Let  $U$  and  $[U]^a$  be a channel type and its characteristic process, respectively.

- (a) If  $U = \langle S \rangle$  and  $\emptyset; \emptyset; \Delta \vdash [S]_{\mathbf{c}} \triangleright S$  then  $a : \langle S \rangle; \emptyset; \Delta \cdot t : !\langle S \rangle; \text{end} \vdash [\langle S \rangle]^a \triangleright \diamond$ .
- (b) If  $U = \langle L \rangle$  and  $\Gamma; \emptyset; \Delta \vdash [L]_{\mathbf{c}} \triangleright L$  then  $\Gamma \cdot a : \langle L \rangle; \emptyset; \Delta \cdot t : !\langle L \rangle; \text{end} \vdash [\langle L \rangle]^a \triangleright \diamond$ .
- (c) If  $U = U' \rightarrow \diamond$  and  $\Gamma; \emptyset; \Delta \vdash [U']_{\mathbf{c}} \triangleright U'$  then  $\Gamma \cdot x : U' \rightarrow \diamond; \emptyset; \Delta \vdash [U' \rightarrow \diamond]^x \triangleright \diamond$ .
- (d) If  $U = U' \multimap \diamond$  and  $\Gamma; \emptyset; \Delta \vdash [U']_{\mathbf{c}} \triangleright U'$  then  $\Gamma \cdot x : U' \multimap \diamond; \emptyset; \Delta \vdash [U' \multimap \diamond]^x \triangleright \diamond$ .

*Proof* The proof proceeds by mutual induction on the syntax of types. We analyze the three parts separately:

1. We use the results from Parts 2 and 3 in a case analysis on the syntax of  $U$ .

- Cases (a)  $U = S$ , (b)  $U = \langle S \rangle$ , and (c)  $U = \langle L \rangle$ : The proof is straightforward from Rules [Sess] and [Sh] (cf. Fig. 3).
- Case (d)  $U = U' \rightarrow \diamond$ : By Parts 2 and 3 of this lemma we obtain  $\Gamma; \emptyset; \Delta \vdash [U']^x \triangleright \diamond$ , which implies  $\Gamma \setminus x; \emptyset; \Delta \setminus x \vdash [U' \rightarrow \diamond]_{\mathbf{c}} \triangleright U' \rightarrow \diamond$  by Rules [Abs] and [EProm] (cf. Fig. 3).
- Case (e)  $U = U' \multimap \diamond$ : Similar, using Rule [Abs] (cf. Fig. 3).

2. The proof is by induction on the syntax of  $S$ . We detail some notable cases:

- (a) Case  $S = !\langle U \rangle$ ;  $S'$ : Then, by Definition 13, we have  $[S]^s = s!\langle [U]_c \rangle.t!\langle s \rangle.\mathbf{0}$  and we may obtain the following derivation:

$$\frac{\begin{array}{c} \Gamma; \emptyset; s : S' \cdot t : !\langle S' \rangle; \text{end} \triangleright t!\langle s \rangle.\mathbf{0} \triangleright \diamond \quad (\text{Induction}) \\ \Gamma; \emptyset; \Delta \vdash [U]_c \triangleright U \end{array}}{\Gamma; \emptyset; \Delta \cdot s : !\langle U \rangle; S' \cdot t : !\langle S \rangle; \text{end} \triangleright s!\langle [U]_c \rangle.t!\langle s \rangle.\mathbf{0} \triangleright \diamond}$$

- (b) Case  $S = ?(S_1)$ ;  $S_2$ : Then, by Definition 13, we have  $[S]^s = s?(x).(t!\langle s \rangle.\mathbf{0} \mid [S_1]^x)$ . and we may obtain the following derivation:

$$\frac{\begin{array}{c} \Gamma; \emptyset; \Delta \cdot x : S_1 \vdash [S_1]^x \triangleright \diamond \quad (\text{Induction}) \\ \Gamma; \emptyset; t : !\langle S_2 \rangle; \text{end} \cdot s : S_2 \vdash t!\langle s \rangle.\mathbf{0} \triangleright \diamond \end{array}}{\frac{\Gamma; \emptyset; \Delta \cdot x : S_1 \cdot t : !\langle S_2 \rangle; \text{end} \cdot s : S_2 \vdash t!\langle s \rangle.\mathbf{0} \mid [S_1]^x \triangleright \diamond}{\Gamma; \emptyset; \Delta \cdot t : !\langle S_2 \rangle; \text{end} \cdot s : ?(U); S_2 \vdash s?(x).(t!\langle s \rangle.\mathbf{0} \mid [S_1]^x) \triangleright \diamond}}$$

- (c) Case  $S = \mu t.S'$ : Then, by Definition 13,  $[S] = [S'\{\text{end}/t\}]^\mu$ . The proof is done by induction on the shape of  $S'$ . We detail two sub-cases; the rest is similar or simpler.

- (i) Sub-case  $S' = \&\{l_i : S_i\}_{i \in I}$ : Then  $[S'\{\text{end}/t\}]^s = s \triangleright \{l_i : t_i!\langle s \rangle.\mathbf{0}\}_{i \in I}$  and  $\text{del}(S) = \{S_i\}_{i \in I}$ :

$$\frac{\forall i \in I, \emptyset; \emptyset; t_i : S_i \{\text{end}/t\} \vdash t_i!\langle s \rangle.\mathbf{0} \triangleright \diamond}{\emptyset; \emptyset; t_i : S_i \{\text{end}/t\} \cdot s : S' \{\text{end}/t\} \vdash s \triangleright \{l_i : t_i!\langle s \rangle.\mathbf{0}\}_{i \in I} \triangleright \diamond}$$

We may then type  $[\mu t.\&\{l_i : S_i\}_{i \in I}]^s$ :

$$\frac{\forall i \in I, \emptyset; \emptyset; t_i : S_i \vdash t_i!\langle s \rangle.\mathbf{0} \triangleright \diamond}{\emptyset; \emptyset; t_i : S_i \cdot s : \mu t.\&\{l_i : S_i\}_{i \in I} \vdash s \triangleright \{l_i : t_i!\langle s \rangle.\mathbf{0}\}_{i \in I} \triangleright \diamond}$$

- (ii) Sub-case  $S' = \mu t'.S''$ : Then  $[\mu t'.S''\{\text{end}/t\}]^s = [S''\{\text{end}/t\}\{\text{end}/t'\}]^s$ . If  $[S''\{\text{end}/t\}\{\text{end}/t'\}]^s = \mathbf{0}$  then the proof is straightforward. If  $\text{del}(S) = \{S_i\}_{i \in I}$  then by induction

$$\frac{(\text{Induction})}{\Gamma; \emptyset; \Delta \cdot t_i : S_i \{\text{end}/t\}\{\text{end}/t'\} \cdot s : S'' \{\text{end}/t\}\{\text{end}/t'\} \vdash [S'' \{\text{end}/t\}\{\text{end}/t'\}]^s \triangleright \diamond}$$

We may then type  $[S]^s$ :

$$\frac{(\text{Induction})}{\Gamma; \emptyset; \Delta \cdot t_i : S_i \cdot s : S \vdash [\mu t.\mu t'.S'']^s \triangleright \diamond}$$

3. The proof uses the result of Part 1. We do a case analysis on the structure of  $U$ .

- (a) Case  $U = \langle S \rangle$ : From Part 1 we have that  $\emptyset; \emptyset; \Delta \vdash [S]_c \triangleright S$ . By applying Rule [Req] (cf. Fig. 3) we obtain:

$$\frac{a : \langle S \rangle; \emptyset; \Delta \vdash [S]_c \triangleright S \quad a : \langle S \rangle; \emptyset; t : !\langle \langle S \rangle \rangle; \text{end} \vdash t!\langle a \rangle.\mathbf{0} \triangleright \diamond}{a : \langle S \rangle; \emptyset; \Delta \cdot t : !\langle \langle S \rangle \rangle; \text{end} \vdash [ \langle S \rangle ]^a \triangleright \diamond}$$

- (b) Case  $U = \langle S \rangle$ : Similar argumentation as in the previous case.

- (c) Case  $U = U' \multimap \diamond$ : From Part 1 we know that  $\Gamma; \emptyset; \Delta \vdash [U']_c \triangleright U'$ . By applying Rules [App] and [EProm] (cf. Fig. 3) we obtain:

$$\frac{\begin{array}{c} \Gamma; \emptyset; \Delta \vdash [U']_c \triangleright U' \quad \Gamma; x : U' \multimap \diamond; \emptyset \vdash x \triangleright U' \multimap \diamond \\ \Gamma; x : U' \multimap \diamond; \Delta \vdash x [U']_c \triangleright \diamond \end{array}}{\Gamma \cdot x : U' \multimap \diamond; \emptyset; \Delta \vdash [U' \multimap \diamond]^x \triangleright \diamond}$$

- (d) Case  $U = U' \rightarrow \diamond$ : Similar argumentation as in the previous case without applying Rule [EProm] (cf. Fig. 3). □

### Deterministic transitions

The proofs for Theorem 2 require an auxiliary result on deterministic transitions (Lemma 1). Some notions needed to prove this auxiliary result are presented next.

In the following we sometimes use polyadic abstractions (denoted  $\lambda\tilde{x}. P$ ) and polyadic name passing (denoted  $u!(\tilde{V}).P$  and  $u?(\tilde{x}).P$ , respectively) as shorthand notations.

We now prove Proposition 2:

**Proposition 5** ( $\tau$ -inertness) *Suppose  $\Gamma; \emptyset; \Delta \vdash P \triangleright \diamond$  with balanced  $\Delta$ . Then*

1.  $\Gamma; \Delta \vdash P \xrightarrow{\tau_d} \Delta' \vdash P'$  implies  $\Gamma; \Delta \vdash P \approx^H \Delta' \vdash P'$ .
2.  $\Gamma; \Delta \vdash P \xRightarrow{\tau_d} \Delta' \vdash P'$  implies  $\Gamma; \Delta \vdash P \approx^H \Delta' \vdash P'$ .

*Proof* We only prove Part 1; the proof for Part 2 follows straightforwardly. The proof proceeds by showing that the relation

$$\mathfrak{R} = \{(P, P') \mid \Gamma; \Delta \vdash P \xrightarrow{\tau_d} \Delta' \vdash P'\}$$

is a higher-order bisimulation.

Suppose first that  $\Gamma; \Delta \vdash P \xrightarrow{\ell} \Delta' \vdash P''$ , for some  $P''$ ; we have to show that  $P'$  can produce an appropriate matching action. There are two main cases:  $\ell \neq \tau$  (a visible action) and  $\ell = \tau$  (an unobservable, possibly deterministic action).

1. The first case follows easily by typing conditions and type soundness, which ensure that  $P'$  has the same potential as  $P$  for performing visible actions.
2. The second case can be divided into two sub-cases: first, if  $\tau = \tau_d$  then  $P' = P''$  and the thesis trivially follows; second, if  $\tau \neq \tau_d$  (i.e.,  $P$  has the possibility of performing both  $\tau_d$  and some other  $\tau$ ) then either  $P'$  has the same  $\tau$  or  $P'$  does not have it, because  $\tau_d$  excluded the occurrence of  $\tau$ . The thesis follows by noticing that, in the first case,  $P'$  can match the move from  $P$ ; the second case cannot occur because of typing and the definition of deterministic transitions.

Suppose now that  $\Gamma; \Delta \vdash P' \xrightarrow{\ell} \Delta' \vdash P''$ , for some  $P''$ . This case follows immediately by noticing that  $P$  can always match action  $\ell$  by performing the deterministic action  $\tau_d$  first, i.e., we can always have  $\Gamma; \Delta \vdash P \xrightarrow{\tau_d} \xrightarrow{\ell} \Delta' \vdash P''$ . This concludes the proof.  $\square$

### Proof of Theorem 2

We split the proof of Theorem 2 into several lemmas:

- Lemma 12 establishes useful properties of characteristic and higher-order processes, including a two-way connection between higher-order trigger processes and an alternative trigger process (denoted  $t \leftarrow_A V$ , defined below).
- Lemma 13 establishes the equivalence between characteristic and higher-order trigger processes.
- Lemma 14 establishes  $\approx^H = \approx^C$ .
- Lemma 16 establishes a trigger substitution lemma (Lemma 4 in the main text), using Lemma 15.
- Lemma 18 exploits the process substitution result given by Lemma 17 (Lemma 3 in the main text) to prove that  $\approx^H \subseteq \approx$ .
- Lemma 19 shows that  $\approx$  is a congruence which implies  $\approx \subseteq \cong$ .

– Lemma 22 shows that  $\cong \subseteq \approx^H$  using Lemma 20 (definability) and Lemma 21 (extrusion).

We introduce a useful notation for action labels, which will be used in the following to represent matching actions.

**Definition 24** Let  $\ell$  be an action label (cf. Sect. 5.1). We define the action  $\check{\ell}$  as

$$\check{\ell} = \begin{cases} (v \widetilde{m}_2)(n!(V_2)) & \text{if } \ell = (v \widetilde{m}_1)(n!(V_1)), \text{ for some } V_2, \widetilde{m}_2 \\ \ell & \text{otherwise} \end{cases}$$

Thus, given  $\ell$ , its corresponding action  $\check{\ell}$  is either identical to  $\ell$ , or an output on the same name, possibly with different object and extruded names.

We now introduce an alternative trigger process that is used to simplify the proofs. Let

$$t \leftarrow_A V = t?(x).(v s)(x s \mid \bar{s}!\langle V \rangle.\mathbf{0}) \tag{15}$$

The simpler formulation of alternative trigger process (with respect to the higher-order trigger process, cf. (6)) is useful in proofs. However, the input of characteristic values on name  $t$  results in the creation of redundant parallel components:

$$\begin{array}{ccc} & & t?(x).(v s)(x s \mid \bar{s}!\langle V \rangle.\mathbf{0}) \\ t?(\lambda x. x?(y).(t'!\langle x \rangle.\mathbf{0} \mid [U']^y)) & & (v s)((\lambda x. x?(y).(t'!\langle x \rangle.\mathbf{0} \mid [U']^y)) s \mid \bar{s}!\langle V \rangle.\mathbf{0}) \\ \xrightarrow{\tau_d} \xrightarrow{\tau_d} & & (v s)([U]^y\{V/y\} \mid t'!\langle s \rangle.\mathbf{0}) \\ \equiv & & [U]^y\{V/y\} \mid (v s)(t'!\langle s \rangle.\mathbf{0}) \end{array}$$

Indeed, processes of the form  $(v s)t'!\langle s \rangle.\mathbf{0}$  are redundant because the restricted name  $s$  has no interactions. The following lemma shows that we can ignore these processes (up to  $\approx^H$  and  $\approx^C$ ). It also states the equivalence (up to  $\approx^H$ ) between higher-order trigger processes  $t \leftarrow_H V$  (cf. (6)) and  $t \leftarrow_A V$ .

**Lemma 12** (Auxiliary results for trigger processes) *Let  $P$  and  $Q$  be processes.*

1. *Let  $t$  be a fresh name,  $\Delta_1 = \Delta_3 \cdot t \text{ !}(\text{end})$ ;  $\text{end}$ , and  $\Delta_2 = \Delta_4 \cdot t \text{ !}(\text{end})$ ;  $\text{end}$ . Then*

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (v s)(t'!\langle s \rangle.\mathbf{0})) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid (v s)(t'!\langle s \rangle.\mathbf{0}))$$

*if and only if*

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)P \approx^H \Delta_4 \vdash (v \widetilde{m}_2)Q$$

2. *Let  $t$  a fresh name,  $\Delta_1 = \Delta_3 \cdot t \text{ !}(\text{end})$ ;  $\text{end}$  and  $\Delta_2 = \Delta_4 \cdot t \text{ !}(\text{end})$ ;  $\text{end}$ . Then*

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (v s)(t'!\langle s \rangle.\mathbf{0})) \approx^C \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid (v s)(t'!\langle s \rangle.\mathbf{0}))$$

*if and only if*

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)P \approx^C \Delta_4 \vdash (v \widetilde{m}_2)Q$$

3. *Let  $t$  be a fresh name. Then*

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_A V_1) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_A V_2)$$

*if and only if, for some  $\Delta_3, \Delta_4$ ,*

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \approx^H \Delta_4 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2)$$

*Proof* We analyze each of the three parts:

– Part I. We split the proof into the two directions of the if and only if requirements.

(a) First direction. Consider the typed relation (we omit the type information):

$$\mathfrak{R} = \{((v \widetilde{m}_1)P, (v \widetilde{m}_2)Q) \mid \Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (v s)(t!\langle s \rangle.\mathbf{0})) \approx^{\text{H}} \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid (v s)(t!\langle s \rangle.\mathbf{0}))\}$$

We check the requirements of higher-order bisimulation for  $\mathfrak{R}$ . Suppose that

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)P \xrightarrow{\ell} \Delta'_3 \vdash (v \widetilde{m}'_1)P'$$

then we need to show a matching action from  $(v \widetilde{m}_2)Q$ . We can derive that

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (v s)(t!\langle s \rangle.\mathbf{0})) \xrightarrow{\ell} \Delta'_1 \vdash (v \widetilde{m}'_1)(P' \mid (v s)(t!\langle s \rangle.\mathbf{0}))$$

for some  $\Delta'_1$  which, from the freshness of  $t$ , implies that there exist  $Q'$  and  $\Delta'_2$  such that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid (v s)(t!\langle s \rangle.\mathbf{0})) \xrightarrow{\check{\ell}} \Delta'_2 \vdash (v \widetilde{m}'_2)(Q' \mid (v s)(t!\langle s \rangle.\mathbf{0})) \quad (16)$$

and

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P' \mid C_1 \mid (v s)(t!\langle s \rangle.\mathbf{0})) \approx^{\text{H}} \Delta'_2 \vdash (v \widetilde{m}'_2)(Q' \mid C_2 \mid (v s)(t!\langle s \rangle.\mathbf{0}))$$

where the shape of  $C_1, C_2$  depends on  $\ell$  and  $\check{\ell}$ : if they are output actions with objects  $V_1$  and  $V_2$ , respectively, then  $C_1 = t' \leftarrow_{\text{H}} V_1$  and  $C_2 = t' \leftarrow_{\text{H}} V_2$ ; otherwise,  $C_1 = C_2 = \mathbf{0}$ . The latter equation implies from the definition of  $\mathfrak{R}$

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P' \mid C_1 \mid (v s)(t!\langle s \rangle.\mathbf{0})) \mathfrak{R} \Delta'_2 \vdash (v \widetilde{m}'_2)(Q' \mid C_2 \mid (v s)(t!\langle s \rangle.\mathbf{0}))$$

and (16) implies

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)Q \xrightarrow{\check{\ell}} \Delta'_2 \vdash (v \widetilde{m}'_2)Q'$$

to complete the proof of the case.

(b) Second direction. Consider the typed relation (we omit the type information):

$$\mathfrak{R} = \{((v \widetilde{m}_1)(P \mid (v s)(t!\langle s \rangle.\mathbf{0})), (v \widetilde{m}_2)(Q \mid (v s)(t!\langle s \rangle.\mathbf{0}))) \mid \Gamma; \Delta_3 \vdash (v \widetilde{m}_1)(P) \approx^{\text{H}} \Delta_4 \vdash (v \widetilde{m}_2)(Q)\}$$

We check the requirements of higher-order bisimulation for  $\mathfrak{R}$ . Suppose that  $(v \widetilde{m}_1)(P \mid (v s)(t!\langle s \rangle.\mathbf{0}))$  moves; we need to infer an appropriate matching action from  $(v \widetilde{m}_2)(Q \mid (v s)(t!\langle s \rangle.\mathbf{0}))$ . We analyse three cases:

(i) Process  $P$  moves autonomously, i.e., for some  $\Delta'_1$  we have:

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (v s)(t!\langle s \rangle.\mathbf{0})) \xrightarrow{\ell} \Delta'_1 \vdash (v \widetilde{m}'_1)(P' \mid (v s)(t!\langle s \rangle.\mathbf{0}))$$

Then the proof is similar to the previous case.

(ii) An action on the fresh name  $t$ , i.e., for some  $\Delta'_1$  we have:

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (v s)(t!\langle s \rangle.\mathbf{0})) \xrightarrow{t!\langle s \rangle} \Delta'_1 \vdash (v \widetilde{m}_1)P$$

First notice that the typing derivation reveals that  $\Delta_1(t) = \Delta_2(t) = !\langle \text{end} \rangle$ ;  $\text{end}$ . This is because the dual endpoint of the (restricted) session  $s$  does not appear in

$(\nu s)(t!\langle s \rangle.\mathbf{0})$  and thus it has the inactive type  $\text{end}$ . We can then observe that, for some  $\Delta'_2$ , we have:

$$\Gamma; \Delta_2 \vdash (\nu \widetilde{m}_2)(Q \mid (\nu s)(t!\langle s \rangle.\mathbf{0})) \xrightarrow{t!\langle s \rangle} \Delta'_2 \vdash (\nu \widetilde{m}_2)Q'$$

We need to show that

$$\Gamma; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid t' \leftarrow_{\text{H}} s) \approx^{\text{H}} \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q' \mid t' \leftarrow_{\text{H}} s)$$

The proof is easy if we consider that both processes can perform the up-to deterministic transitions  $\xrightarrow{t'?( \lambda z, \mathbf{0} )} \xrightarrow{\tau_{\text{d}}}$ :

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid t' \leftarrow_{\text{H}} s) \\ & \xrightarrow{t'?( \lambda z, \mathbf{0} )} \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid (\nu s')(s'?(y).((\lambda z. \mathbf{0}) y) \mid \overline{s'}!\langle s \rangle.\mathbf{0})) \\ & \xrightarrow{\tau_{\text{d}}} \Delta'_1 \vdash (\nu \widetilde{m}_1)P \end{aligned}$$

and

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid t' \leftarrow_{\text{H}} s) \\ & \xrightarrow{t'?( \lambda z, \mathbf{0} )} \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q' \mid (\nu s')(s'?(y).((\lambda z. \mathbf{0}) y) \mid \overline{s'}!\langle s \rangle.\mathbf{0})) \\ & \xrightarrow{\tau_{\text{d}}} \Delta'_2 \vdash (\nu \widetilde{m}_2)Q' \end{aligned}$$

The result is then immediate from the definition of  $\mathfrak{R}$  that requires

$$\Gamma; \Delta'_1 \vdash (\nu \widetilde{m}_1)P \approx^{\text{H}} \Delta'_2 \vdash (\nu \widetilde{m}_2)Q'$$

- (iii) A synchronization along name  $t$ : this is not possible due to the freshness of  $t$ .

This concludes the proof of Part 1.

- Part 2 follows same arguments and structure as the proof for Part 1.
- Part 3 relies on Part 1. We analyse the two directions of the if and only if requirement.

(a) First direction. Let  $\mathfrak{R}$  be the typed relation (we omit the type information):

$$\begin{aligned} \mathfrak{R} = \{ & ((\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{H}} V_1), (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{H}} V_2)) \mid \\ & \Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{A}} V_1) \approx^{\text{H}} \Delta_2 \vdash (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{A}} V_2) \} \end{aligned}$$

We show that  $\mathfrak{R} \subseteq \approx^{\text{H}}$ , with a case analysis on the defining requirements of higher-order bisimulation. Suppose that  $(\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{H}} V_1)$  moves; we need to show an appropriate matching action from  $(\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{H}} V_2)$ . We analyze three possibilities:

- (i)  $P$  moves on its own, i.e., for some  $\Delta'_1$  we have:

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{H}} V_1) \xrightarrow{\ell} \Delta'_1 \vdash (\nu \widetilde{m}_1')(P' \mid t \leftarrow_{\text{H}} V_2)$$

The proof is similar to case (a) of Part 1 of this lemma.

- (ii) An input action of the form  $t'?(n)$  along a fresh name  $t$ . Let  $U$  be such that  $[U]_{\text{C}} = n$  and let  $V_1$  be a higher-order value. There exists a  $\Delta'_1$  such that:

$$\begin{aligned} \Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{H}} V_1) & \xrightarrow{t'?(n)} \Delta'_1 \vdash (\nu \widetilde{m}_1')(P \mid (\nu s)(s?(y).(y n) \mid \overline{s'}!(V_1).\mathbf{0})) \\ & \xrightarrow{\tau_{\text{d}}} \Delta'_1 \vdash (\nu \widetilde{m}_1')(P \mid (V_1 n)) \end{aligned}$$



Furthermore, we can see that, for some  $\Delta'_2$ , we have

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_{\mathbb{H}} V_2) \xrightarrow{t^?(n)} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n))$$

We therefore need to show that

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (V_1 n)) \approx^{\mathbb{H}} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n))$$

This is done by considering the requirements of  $\mathfrak{R}$ .

Because of the definition of the alternative trigger, the input of the trigger value has no effect on the bisimulation relation:

$$\begin{array}{c} \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_{\mathbb{A}} V_1) \\ t^?(\lambda z. t^?(w).wz) \xrightarrow{\tau_d} \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid t' \leftarrow_{\mathbb{A}} V_1) \end{array}$$

Since  $[U]_{\mathbb{C}} = n$ , we have that  $[(?U); \text{end}] \rightarrow \diamond]_{\mathbb{C}} = \lambda z. z^?(y).(t'!\langle z \rangle. \mathbf{0} \mid (y n))$ :

$$\begin{array}{c} \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_{\mathbb{A}} V_1) \\ t^?([(?U); \text{end}] \rightarrow \diamond]_{\mathbb{C}}) \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (v s)(\lambda z. z^?(y).(t'!\langle z \rangle. \mathbf{0} \mid (y n)) s \mid \bar{s}!\langle V_1 \rangle. \mathbf{0})) \end{array}$$

Furthermore, we can see that

$$\begin{array}{c} \Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_{\mathbb{A}} V_2) \\ t^?([(?U); \text{end}] \rightarrow \diamond]_{\mathbb{C}}) \xrightarrow{\tau_d} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n) \mid (v s)(t'!\langle s \rangle. \mathbf{0})) \end{array}$$

We also have

$$\begin{array}{c} \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (v s)(\lambda z. z^?(y).(t'!\langle z \rangle. \mathbf{0} \mid (y n)) s \mid \bar{s}!\langle V_1 \rangle. \mathbf{0})) \\ \xrightarrow{\tau_d} \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (V_1 n) \mid (v s)(t'!\langle s \rangle. \mathbf{0})) \end{array}$$

and so we can infer from the up-to technique for deterministic transitions (Lemma 1) that

$$\begin{array}{c} \Gamma; \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (V_1 n) \mid (v s)(t'!\langle s \rangle. \mathbf{0})) \\ \approx^{\mathbb{H}} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n) \mid (v s)(t'!\langle s \rangle. \mathbf{0})) \end{array}$$

which implies, by Part 1 of this lemma, the desired conclusion:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (V_1 n)) \approx^{\mathbb{H}} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n))$$

- (iii) An action of the form  $t^?(\lambda z. [U']^z)$  along the fresh name  $t$ . Let  $U$  such that  $[U]_{\mathbb{C}} = \lambda z. [U']^z$ . There exist  $U$  and  $\Delta'_1$  such that

$$\begin{array}{c} \Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_{\mathbb{H}} V_1) \\ t^?(\lambda z. [U']^z) \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (v s)(s^?(y).(\lambda z. [U']^z y) \mid \bar{s}!\langle V_1 \rangle. \mathbf{0})) \\ \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (\lambda z. [U']^z) V_1) \end{array}$$

Furthermore, we have the following transition, for some  $\Delta'_2$ :

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_{\mathbb{H}} V_2) \xrightarrow{t^?(\lambda z. [U']^z)} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (\lambda z. [U']^z) V_2)$$

We need to show that, for some  $\Delta'_3, \Delta'_4$ , the following holds:

$$\Gamma; \Delta'_3 \vdash (v \widetilde{m}'_1)(P \mid (\lambda z. [U']^z) V_1) \approx^H \Delta'_4 \vdash (v \widetilde{m}'_2)(Q' \mid (\lambda z. [U']^z) V_2)$$

This is done by considering the requirements of  $\mathfrak{R}$ .

Here again note that the input of the trigger value has no effect on the bisimulation relation.

$$\begin{aligned} & \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_A V_1) \\ & \xrightarrow{t?( \lambda z. t'?(w).wz)} \xrightarrow{\tau_d} \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid t' \leftarrow_A V_1) \end{aligned}$$

We now consider the input of the characteristic value on  $t$ . From the fact that  $[U]_c = \lambda z. [U']^z$  we obtain that  $[(?(U); \text{end}) \rightarrow \diamond]_c = \lambda w. w?(y).(t'!(w).\mathbf{0} \mid (\lambda z. [U']^z) y)$  and

$$\begin{aligned} & \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_A V_1) \xrightarrow{t?([?(U); \text{end}) \rightarrow \diamond]_c} \\ & \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid (v s)((\lambda w. w?(y).(t'!(w).\mathbf{0} \mid (\lambda z. [U']^z) y)) s \mid \bar{s}!(V_1).\mathbf{0})) \end{aligned}$$

Furthermore, we have the following transition, for some  $\Delta'_2$ :

$$\begin{aligned} & \Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_A V_2) \xrightarrow{t?([?(U); \text{end}]_c) \rightarrow \diamond} \\ & \Delta'_2 \vdash (v \widetilde{m}'_2)(Q' \mid (\lambda z. [U']^z) V_2 \mid (v s)(t'!(s).\mathbf{0})) \end{aligned}$$

We also have

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid (v s)((\lambda w. w?(y).(t'!(w).\mathbf{0} \mid (\lambda z. [U']^z) y)) s \mid \bar{s}!(V_1).\mathbf{0})) \\ & \xrightarrow{\tau_d} \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid (\lambda z. [U']^z) V_1 \mid (v s)(t'!(s).\mathbf{0})) \end{aligned}$$

and so we can infer from the up-to technique for deterministic transitions (Lemma 1) that

$$\begin{aligned} & \Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid (\lambda z. [U']^z) V_1 \mid (v s)(t'!(s).\mathbf{0})) \approx^H \\ & \vdash \Delta'_2(v \widetilde{m}'_2)(Q' \mid (\lambda z. [U']^z) V_2 \mid (v s)(t'!(s).\mathbf{0})) \end{aligned}$$

which implies, by Part 1 of this lemma, the desired conclusion:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid (\lambda z. [U']^z) V_1) \approx^H \Delta'_2 \vdash (v \widetilde{m}'_2)(Q' \mid (\lambda z. [U']^z) V_2)$$

(b) Second direction. Let  $\mathfrak{R}$  be the typed relation (we omit the type information):

$$\begin{aligned} \mathfrak{R} = \{ & ((v \widetilde{m}_1)(P \mid t \leftarrow_A V_1), (v \widetilde{m}_2)(Q \mid t \leftarrow_A V_2)) \mid \\ & \Gamma; \Delta_3 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \approx^H \Delta_4 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2) \} \end{aligned}$$

We show that  $\mathfrak{R} \subseteq \approx^H$ , with a case analysis on the defining requirements of higher-order bisimulation. We focus on the cases related to an input action on the fresh name  $t$ ; other cases are similar.

- i. Value  $V_1$  is a higher-order value: This implies that there exist  $U$  and  $\Delta'_1$  such that  $[(?(U); \text{end}]_c = \lambda z. z?(y).(t'!(z).\mathbf{0} \mid y n)$  and

$$\begin{aligned} & \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_A V_1) \\ & \xrightarrow{t?([?(U); \text{end}]_c)} \Delta'_1 \vdash (v \widetilde{m}'_1)(P \mid (v s)((\lambda z. z?(y).(t'!(z).\mathbf{0} \mid (y n))) s \mid \bar{s}!(V_1).\mathbf{0})) \end{aligned}$$

and

$$\Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (v s)((\lambda z. z?(y).(t'!\langle z \rangle.\mathbf{0} \mid (y n))) s \mid \bar{s}!(V_1).\mathbf{0})) \\ \xrightarrow{\tau_d} \xrightarrow{\tau_d} \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (V_1 n) \mid (v s)(t'!\langle s \rangle.\mathbf{0}))$$

Furthermore we can see that there exists  $\Delta'_2$  such that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_A V_2) \stackrel{t?(U); \text{end}]_c}{\Longrightarrow} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n) \mid (v s)(t'!\langle s \rangle.\mathbf{0}))$$

We need to show that

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1')(P \mid (V_1 n) \mid (v s)(t'!\langle s \rangle.\mathbf{0})) \\ \approx^H \Delta'_2 \vdash (v \widetilde{m}_2')(Q \mid (V_2 n) \mid (v s)(t'!\langle s \rangle.\mathbf{0}))$$

This is done by considering the requirements of  $\mathfrak{R}$ . We know that  $[U]_c = n$ :

$$\Gamma; \emptyset; \Delta_3 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \\ \xrightarrow{t?(n)} \Delta'_3 \vdash (v \widetilde{m}_1')(P \mid (v s)(s?(y).(y n) \mid \bar{s}!(V_1).\mathbf{0})) \xrightarrow{\tau_d} \Delta'_3 \vdash (v \widetilde{m}_1')(P \mid (V_1 n))$$

for some  $\Delta'_3$ . Furthermore we can see that for some  $\Delta'_4$

$$\Gamma; \Delta_4 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2) \stackrel{t?(n)}{\Longrightarrow} \Delta'_4 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n))$$

and

$$\Gamma; \Delta'_3 \vdash (v \widetilde{m}_1')(P \mid (V_1 n)) \approx^H \Delta'_4 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n))$$

which imply, by Part 1 of this lemma, the desired conclusion:

$$\Gamma; \Delta'_3 \vdash (v \widetilde{m}_1')(P \mid (V_1 n) \mid (v s)(t'!\langle s \rangle.\mathbf{0})) \\ \approx^H \Delta'_4 \vdash (v \widetilde{m}_2')(Q' \mid (V_2 n) \mid (v s)(t'!\langle s \rangle.\mathbf{0}))$$

- ii Value  $V_1$  is a first-order value: This implies that there exist  $U$  and  $\Delta'_3$  such that  $[?(U); \text{end}]_c = \lambda w. w?(y).(t'!\langle w \rangle.\mathbf{0} \mid \lambda z. [U']^z y)$  and

$$\Gamma; \emptyset; \Delta_3 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \\ \stackrel{[?(U); \text{end}]_c}{\Longrightarrow} \Delta'_3 \vdash (v \widetilde{m}_1')(P \mid (v s)(\lambda w. w?(y).(t'!\langle w \rangle.\mathbf{0} \mid \lambda z. [U']^z y) s \mid \bar{s}!(V_1).\mathbf{0}))$$

This case follows a similar proof structure as the previous case.

This concludes the proof of Part 3.  $\square$

The next lemma states the equivalence between the characteristic and higher-order trigger processes (cf. (6) and (7)).

**Lemma 13** (Trigger process equivalence) *Let  $P$  and  $Q$  be processes,  $t$  be a fresh name, and let  $\Gamma; \emptyset; \Delta \vdash V_i \triangleright U, i \in \{1, 2\}$ .*

(1) *If*

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2)$$

*then there exist  $\Delta'_1, \Delta'_2$  such that*

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U) \approx^H \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_C V_2 : U).$$

(2) If

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U) \approx^C \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_C V_2 : U)$$

then there exist  $\Delta'_1, \Delta'_2$  such that

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \approx^C \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2).$$

*Proof* We analyse both parts separately:

1. Consider the typed relation (for readability, we omit type information):

$$\mathfrak{R} = \{((v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U), (v \widetilde{m}_2)(Q \mid t \leftarrow_C V_2 : U)) \mid \Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \approx^H \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2)\}$$

We show that  $\mathfrak{R} \subseteq \approx^H$ . Suppose that  $(v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U)$  moves; we need to find a matching move from  $(v \widetilde{m}_2)(Q \mid t \leftarrow_C V_2 : U)$ . We distinguish three cases, depending on the source/kind of visible action:

(a)  $P$  moves autonomously, i.e., for some  $\Delta_3$  we have:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U) \xrightarrow{\ell} \Delta_3 \vdash (v \widetilde{m}'_1)(P' \mid t \leftarrow_C V_1 : U)$$

We follow the requirements of  $\mathfrak{R}$  and the freshness of  $t$  to conclude that there exists a  $\Delta''_1$  such that

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_H V_1) \xrightarrow{\ell} \Delta''_1 \vdash (v \widetilde{m}'_1)(P' \mid t \leftarrow_H V_1)$$

which implies, from the higher-order bisimilarity requirement of  $\mathfrak{R}$  and the freshness of  $t$ , that there exist  $Q'$  and  $\Delta'_2$  such that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_H V_2) \xrightarrow{\check{\ell}} \Delta'_2 \vdash (v \widetilde{m}'_2)(Q' \mid t \leftarrow_H V_2) \tag{17}$$

and, for some  $\Delta'''_1$  and  $\Delta'''_2$ , that

$$\Gamma; \Delta'''_1 \vdash (v \widetilde{m}''_1)(P' \mid t \leftarrow_H V_1 \mid C_1) \approx^H \Delta'''_2 \vdash (v \widetilde{m}''_2)(Q' \mid t \leftarrow_H V_2 \mid C_2) \tag{18}$$

where the shape of  $C_1, C_2$  depends on  $\ell$  and  $\check{\ell}$ : if they are output actions with objects  $V'_1$  and  $V'_2$ , respectively, then  $C_1 = t' \leftarrow_H V'_1$  and  $C_2 = t' \leftarrow_H V'_2$ ; otherwise,  $C_1 = C_2 = \mathbf{0}$ .

From (17) and the definition of  $\mathfrak{R}$  we can conclude that there exists a  $\Delta_4$  such that

$$\Gamma; \Delta'_2 \vdash (v \widetilde{m}_1)(Q \mid t \leftarrow_C V_2 : U) \xrightarrow{\check{\ell}} \Delta_4 \vdash (v \widetilde{m}'_2)(Q' \mid t \leftarrow_C V_2 : U)$$

Equation (18) then allows us to infer the required conclusion, for some  $\Delta'_3, \Delta'_4$ :

$$\Gamma; \Delta'_3 \vdash (v \widetilde{m}_1''')(P' \mid t \leftarrow_C V_1 : U \mid C_1) \mathfrak{R} \Delta'_4 \vdash (v \widetilde{m}_2''')(Q' \mid t \leftarrow_C V_2 : U \mid C_2)$$

(b)  $t \leftarrow_C V_1 : U$  moves autonomously, i.e., for some  $\Delta_3$  we have:

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U) \\ \xrightarrow{t?(m)} \Delta_3 \vdash (v \widetilde{m}_1)(P \mid (v s)(s?(y).[U]^y \mid \bar{s}!(V_1).\mathbf{0})) \end{aligned}$$

Following requirements of  $\mathfrak{R}$  and the freshness of  $t$  we can infer that there exists a  $\Delta'_1$  such that

$$\begin{aligned} & \Gamma; \emptyset; \Delta_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{H}} V_1) \\ & \xrightarrow{t?(U)_c} \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid (\nu s)(s?(y).\llbracket U \rrbracket^y \mid \overline{s!}\langle V_1 \rangle.\mathbf{0})) \end{aligned}$$

which implies, from the higher-order bisimilarity requirement of  $\mathfrak{R}$  and the freshness of  $t$ , that there exist  $Q'$  and  $\Delta'_2$  such that

$$\begin{aligned} & \Gamma; \emptyset; \Delta_2 \vdash (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{H}} V_2) \\ & \xRightarrow{\text{H}} (\nu \widetilde{m}_2)(Q_2 \mid t \leftarrow_{\text{H}} V_2) \\ & \xrightarrow{t?(U)_c} (\nu \widetilde{m}_2)(Q_2 \mid (\nu s)(s?(y).\llbracket U \rrbracket^y \mid \overline{s!}\langle V_2 \rangle.\mathbf{0})) \\ & \xRightarrow{\text{H}} \Delta'_2 \vdash Q' \end{aligned} \tag{19}$$

and

$$\Gamma \Delta'_1 (\nu \widetilde{m}_1)(P \mid (\nu s)(s?(y).\llbracket U \rrbracket^y \mid \overline{s!}\langle V_1 \rangle.\mathbf{0})) \approx^{\text{H}} \Delta'_2 (\nu \widetilde{m}_2) Q' \tag{20}$$

The freshness of  $t$  allows us to mimic the transitions in (19); for some  $\Delta_4$  we obtain:

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{C}} V_2 : U) \\ & \xRightarrow{\text{H}} (\nu \widetilde{m}_2)(Q_2 \mid t \leftarrow_{\text{C}} V_2 : U) \\ & \xrightarrow{t?(m)} (\nu \widetilde{m}_2)(Q_2 \mid (\nu s)(s?(y).\llbracket U \rrbracket^y \mid \overline{s!}\langle V_2 \rangle.\mathbf{0})) \\ & \xRightarrow{\text{H}} \Delta_4 \vdash Q' \end{aligned}$$

The conclusion is immediate from (20).

- (c) The action comes from the interaction of  $P$  and  $t \leftarrow_{\text{H}} V_1$ : This case is not possible, due to the freshness of  $t$ .

2. Consider the typed relation (for readability, we omit type information):

$$\mathfrak{R}' = \{((\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{H}} V_1), (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{H}} V_2)) \mid \Gamma; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{C}} V_1 : U) \approx^{\text{C}} \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{C}} V_2 : U)\}$$

To prove that  $\mathfrak{R}' \subseteq \approx^{\text{C}}$  we first consider relation  $\mathfrak{R}$  which uses the alternative trigger in (15) (for readability, we omit type information):

$$\mathfrak{R} = \{((\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{A}} V_1), (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{A}} V_2)) \mid \Gamma; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{C}} V_1 : U) \approx^{\text{C}} \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{C}} V_2 : U)\}$$

By proving that  $\mathfrak{R} \subseteq \approx^{\text{C}}$  we can apply Lemma 12 (Part 3), to obtain that  $\mathfrak{R}' \subseteq \approx^{\text{C}}$ . Suppose that  $(\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{A}} V_1)$  moves; we must exhibit a matching move from  $(\nu \widetilde{m}_2)(Q \mid t \leftarrow_{\text{A}} V_2)$ . We distinguish four cases, depending on the source/kind of visible action:

- (a)  $P$  moves autonomously, i.e., for some  $\Delta_3$  we have:

$$\Gamma; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{A}} V_1) \xrightarrow{\ell} \Delta_3 \vdash (\nu \widetilde{m}'_1)(P' \mid t \leftarrow_{\text{A}} V_1)$$

Then, following the requirements of  $\mathfrak{R}$  and the freshness of  $t$ , we infer that there exists a  $\Delta''_1$  such that

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_{\text{C}} V_1 : U) \xrightarrow{\ell} \Delta''_1 \vdash (\nu \widetilde{m}'_1)(P' \mid t \leftarrow_{\text{C}} V_1 : U)$$

which implies, from the characteristic bisimilarity requirement of  $\mathfrak{R}$  and the freshness of  $t$ , that there exist  $Q'$  and  $\Delta_2''$  such that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_C V_2 : U) \xrightarrow{\check{\ell}} \Delta_2'' \vdash (v \widetilde{m}_2')(Q' \mid t \leftarrow_C V_2 : U) \quad (21)$$

and

$$\Gamma \Delta_1''' (v \widetilde{m}_1'')(P' \mid t \leftarrow_C V_1 : U \mid C_1) \approx^C \Delta_2''' (v \widetilde{m}_2''')(Q' \mid t \leftarrow_C V_2 : U \mid C_2) \quad (22)$$

with  $C_1$  (resp.,  $C_2$ ) being the characteristic trigger process in the cases where  $\ell = (v \widetilde{m})n!(V_1')$  (resp.,  $\check{\ell} = (v \widetilde{m}')n!(V_2')$ ), and  $C_1 = C_2 = \mathbf{0}$  otherwise. From (21) we can infer that there exists  $\Delta_4$  such that

$$\Gamma; \Delta_2' \vdash (v \widetilde{m}_1)(Q \mid t \leftarrow_A V_2) \xrightarrow{\check{\ell}} \Delta_4 \vdash (v \widetilde{m}_2')(Q' \mid t \leftarrow_A V_2)$$

Equation (22) then allows us to obtain the desired conclusion:

$$\Gamma; \Delta_3' \vdash (v \widetilde{m}_1''')(P' \mid t \leftarrow_A V_1 \mid C_1) \mathfrak{R} \Delta_4' \vdash (v \widetilde{m}_2''')(Q' \mid t \leftarrow_A V_2 \mid C_2)$$

- (b)  $t \leftarrow_A V_1$  moves autonomously due to the input of characteristic value, i.e., for some  $\Delta_3$  we have:

$$\begin{aligned} \Gamma; \Delta_1' \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_A V_1) &\xrightarrow{t?(U); \text{end} \rightarrow \diamond]_c} \\ \Delta_3 \vdash (v \widetilde{m}_1)(P \mid (v s)([?(U); \text{end} \rightarrow \diamond]_c s \mid \bar{s}!(V_1).\mathbf{0})) & \end{aligned}$$

Following requirements of  $\mathfrak{R}$  and the freshness of  $t$ , we infer that there is a  $\Delta_1''$  such that

$$\begin{aligned} \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t \leftarrow_C V_1 : U) \\ \xrightarrow{t?(m)} \Delta_1'' \vdash (v \widetilde{m}_1)(P \mid (v s)(s?(y).[U]^y \mid \bar{s}!(V_1).\mathbf{0})) \\ \xrightarrow{\tau_d} \Delta_1' \vdash (v \widetilde{m}_1)P' \end{aligned}$$

which implies, from the characteristic bisimulation requirement of  $\mathfrak{R}$  and the freshness of  $t$ , that there exist  $Q'$  and  $\Delta_2''$  such that

$$\begin{aligned} \Gamma; \emptyset; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t \leftarrow_C V_2 : U) \\ \xrightarrow{\quad} (v \widetilde{m}_2)(Q_2 \mid t \leftarrow_C V_2 : U) \\ \xrightarrow{t?(m)} (v \widetilde{m}_2)(Q_2 \mid (v s)(s?(y).[U]^y \mid \bar{s}!(V_2).\mathbf{0})) \\ \xrightarrow{\quad} \Delta_2'' \vdash Q' \end{aligned} \quad (23)$$

and

$$\Gamma; \Delta_1'' \vdash (v \widetilde{m}_1)P' \approx^C \Delta_2'' \vdash (v \widetilde{m}_2)Q'$$

which in turn implies from Lemma 12 (Part 2) the following, for a fresh  $t'$ :

$$\Gamma; \Delta_1' \vdash (v \widetilde{m}_1)(P' \mid (v s)(t'!(s).\mathbf{0})) \approx^C \Delta_2' \vdash (v \widetilde{m}_2)(Q' \mid (v s)(t'!(s).\mathbf{0})) \quad (24)$$

The freshness of  $t$  allows us to mimic the transitions in (23) to infer that, for some  $\Delta_4$ , we have

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid t \leftarrow_A V_2) \\ \implies & (\nu \widetilde{m}_2)(Q_2 \mid t \leftarrow_A V_2) \\ t^?([?(U); \text{end}]_c) & \xrightarrow{\quad} (\nu \widetilde{m}_2)(Q_2 \mid (\nu s)([?(U); \text{end}]_c s \mid \bar{s}!(V_2).\mathbf{0})) \\ \implies & \Delta_4 \vdash (\nu \widetilde{m}'_2)(Q' \mid (\nu s)(t'!(s).\mathbf{0})) \end{aligned}$$

and

$$\begin{aligned} \Gamma; \Delta_3 \vdash (\nu \widetilde{m}_1)(P \mid (\nu s)([?(U); \text{end}]_c s \mid \bar{s}!(V_1).\mathbf{0})) & \xrightarrow{\tau_d} \\ \Delta_3 \vdash (\nu \widetilde{m}'_1)(P' \mid (\nu s)(t'!(s).\mathbf{0})) & \end{aligned}$$

The conclusion is immediate from (24).

- (c)  $t \leftarrow_A V_1$  moves autonomously due to the input of a trigger process, i.e., for some  $\Delta_3$  we have:

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid t \leftarrow_A V_1) \\ t^?(\lambda x. t'?(y).(y x)) & \xrightarrow{\quad} \Delta_3 \vdash (\nu \widetilde{m}_1)(P \mid (\nu s)((\lambda x. t'?(y).(y x))s \mid \bar{s}!(V_1).\mathbf{0})) \end{aligned}$$

We show that there exist  $\Delta_4$  and  $(\nu \widetilde{m}_1)(Q \mid (\nu s)((\lambda x. t'?(y).(y x))s \mid \bar{s}!(V_2).\mathbf{0}))$  such that

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_2 \vdash (\nu \widetilde{m}_1)(Q \mid t \leftarrow_A V_2) \\ t^?(\lambda x. t'?(y).(y x)) & \xrightarrow{\quad} \Delta_4 \vdash (\nu \widetilde{m}_1)(Q \mid t' \leftarrow_A V_2) \end{aligned}$$

and

$$\begin{aligned} \Gamma; \emptyset; \Delta_3 \vdash (\nu \widetilde{m}_1)(P \mid (\nu s)((\lambda x. t'?(y).(y x))s \mid \bar{s}!(V_1).\mathbf{0})) & \\ \xrightarrow{\tau_d} \Delta_3 \vdash (\nu \widetilde{m}_1)(P \mid t' \leftarrow_A V_1) & \end{aligned}$$

The result

$$\Gamma; \Delta_3 \vdash (\nu \widetilde{m}_1)(P \mid t' \leftarrow_A V_1) \Re \Delta_4 \vdash (\nu \widetilde{m}_1)(Q \mid t' \leftarrow_A V_2)$$

is immediate from the definition of  $\Re$ .

- (d) The action comes from the interaction of  $P$  and  $t \leftarrow_A V_1$ : This case is not possible, due to the freshness of  $t$ . □

**Lemma 14**  $\approx^H = \approx^C$ .

*Proof* We split the proof into two parts: the direction  $\approx^H \subseteq \approx^C$  and the direction  $\approx^C \subseteq \approx^H$ . Since the two equivalences differ only in the output case, our analysis focuses on output actions.

1. Direction  $\approx^H \subseteq \approx^C$ . Consider the typed relation (for readability, we omit type information):

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

We show that  $\Re$  is a characteristic bisimulation. Suppose  $\Gamma; \Delta_1 \vdash P \xrightarrow{\ell} \Delta'_1 \vdash P'$ . We need to show that  $\Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond$  can match  $\ell$ . The proof proceeds by a case analysis on the transition label  $\ell = (\nu \widetilde{m}_1)n!(V_1)$ , which is the only non-trivial case.

From the definition of  $\Re$  we have that if:

$$\Gamma; \Delta_1 \vdash P \xrightarrow{(\nu \widetilde{m}_1)n!(V_1)} \Delta'_1 \vdash P' \tag{25}$$

then there exist  $\Delta_2'', Q$ , and  $V_2$  such that:

$$\Gamma; \Delta_2 \vdash Q \stackrel{(v \widetilde{m}_2)n!(V_2)}{\Longrightarrow} \Delta_2'' \vdash Q' \tag{26}$$

and for a fresh  $t$  and some  $\Delta_1'$  and  $\Delta_2'$ :

$$\Gamma; \Delta_1' \vdash (v \widetilde{m}_1)(P' \mid t \leftarrow_H V_1) \approx^H \Delta_2' \vdash (v \widetilde{m}_2)(Q' \mid t \leftarrow_H V_2) \tag{27}$$

To show that  $\mathfrak{R}$  is a characteristic bisimulation after the fact that transition (25) implies transition (26), we need to show that for a fresh  $t$  and for some  $\Delta_3, \Delta_4$ :

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)(P' \mid t \leftarrow_C V_1 : U) \mathfrak{R} \Delta_4 \vdash (v \widetilde{m}_2)(Q' \mid t \leftarrow_C V_2 : U) \tag{28}$$

which follows from (27), Lemma 13(1), and the definition of  $\mathfrak{R}$ .

- Direction  $\approx^C \subseteq \approx^H$ . Consider the typed relation (for readability, we omit type information):

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

We show that  $\mathfrak{R}$  is a higher-order bisimulation. Suppose  $\Gamma; \Delta_1 \vdash P \xrightarrow{\ell} \Delta_1' \vdash P'$  with  $\ell = (v \widetilde{m}_1)n!(V_1)$ . We need to show that  $\Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond$  can match  $\ell$ .

From the definition of  $\mathfrak{R}$  we have that if:

$$\Gamma; \Delta_1 \vdash P \stackrel{(v \widetilde{m}_1)n!(V_1)}{\xrightarrow{\ell}} \Delta_1' \vdash P' \tag{29}$$

then there exist  $\Delta_2'', Q$ , and  $V_2$  such that:

$$\Gamma; \Delta_2 \vdash Q \stackrel{(v \widetilde{m}_2)n!(V_2)}{\Longrightarrow} \Delta_2'' \vdash Q' \tag{30}$$

and for a fresh  $t$  and some  $\Delta_1', \Delta_2'$ :

$$\Gamma; \Delta_1' \vdash (v \widetilde{m}_1)(P' \mid t \leftarrow_C V_1 : U) \approx^C \Delta_2' \vdash (v \widetilde{m}_2)(Q' \mid t \leftarrow_C V_2 : U) \tag{31}$$

To show that  $\mathfrak{R}$  is a higher-order bisimulation after the fact that transition (29) implies transition (30), we need to show that for a fresh  $t$  and some  $\Delta_3, \Delta_4$ :

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)(P' \mid t \leftarrow_H V_1) \mathfrak{R} \Delta_4 \vdash (v \widetilde{m}_2)(Q' \mid t \leftarrow_H V_2) \tag{32}$$

which follows from (31), Lemma 13(2), and the definition of  $\mathfrak{R}$ . □

We state an auxiliary lemma that captures a property of trigger processes in terms of process equivalence.

**Lemma 15** (Trigger process application) *Let  $P$  and  $Q$  be processes. Also, let  $t$  be a fresh name.*

- If  $n_1 \neq n_2$  with  $\Gamma; \emptyset; \Delta \vdash n_i \triangleright U$  with  $U \neq \text{end}$  and

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (\lambda x. t?(y).(y x)) n_1) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid (\lambda x. t?(y).(y x)) n_2)$$

then  $n_1, n_2$  are session names and  $\overline{n}_1 \in \text{fn}(P)$  and  $\overline{n}_2 \in \text{fn}(Q)$ .

- If  $\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid [U]_C n_1) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid [U]_C n_2)$  then for all  $\ell$  whenever

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid [U]_C n_1) \xrightarrow{\ell} \Delta_1' \vdash (v \widetilde{m}_1')(P' \mid (\lambda x. t?(y).(y x)) n_1)$$

then there exist  $\Delta_2', (v \widetilde{m}_2')(Q' \mid (\lambda x. t?(y).(y x)) n_2)$  such that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_1)(Q \mid [U]_C n_2) \stackrel{\ell_2}{\Longrightarrow} \Delta_2' \vdash (v \widetilde{m}_2')(Q' \mid (\lambda x. t?(y).(y x)) n_2)$$

with  $\ell_2 = \check{\ell}$ .



3. If  $\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t!\langle n_1 \rangle. \mathbf{0}) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t!\langle n_2 \rangle. \mathbf{0})$  then

$$\Gamma; \Delta_1 \vdash (v m_1)(P \mid t?(x).(x n_1)) \approx^H \Delta_2 \vdash (v m_2)(Q \mid t?(x).(x n_2))$$

4. If  $n$  is fresh and

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P\{n/x\} \mid t!\langle n_1 \rangle. \mathbf{0}) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q\{n/x\} \mid t!\langle m_1 \rangle. \mathbf{0})$$

then

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P\{n_1/x\}) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(Q\{m_1/x\})$$

*Proof* We analyse each part separately:

1. The proof for Part 1 is by contradiction. Assume that  $\overline{n}_1 \notin \text{fn}(P)$  or  $\overline{n}_2 \notin \text{fn}(Q)$ . Then the bisimulation requirement allows us to observe the following transition, for some  $U \neq \text{end}$ . Note that the shape of  $[U]^{n_1}$  enables an observable action on  $n_1$ , which results in the process  $t'!\langle n_1 \rangle. \mathbf{0}$ :

$$\begin{array}{l} \Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid (\lambda x. t?(y).(y x)) n_1) \\ \xrightarrow{\tau_d} (v \widetilde{m}_1)(P \mid t?(y).y n_1) \\ \xrightarrow{t?([U \rightarrow \diamond]_c)} (v \widetilde{m}_1)(P \mid \lambda x. [U]^x n_1) \\ \xrightarrow{\tau_d} \xrightarrow{\ell} \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid C \mid t'!\langle n_1 \rangle. \mathbf{0}) \end{array}$$

where  $C = \mathbf{0}$  if  $\ell$  is not an input action, and  $C = [U]^m$  if  $\ell$  is an input action and  $\text{subj}(\ell) = n_1$ . Because of the characteristic process interaction, from the freshness of  $t$ , we have:

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid (\lambda x. t?(y).(y x)) n_2) \xrightarrow{t?([U \rightarrow \diamond]_c)} \xrightarrow{\check{\ell}} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid C \mid [U \rightarrow \diamond]_c n_2)$$

with  $\text{subj}(\check{\ell}) = n_2$ . But since  $(v \widetilde{m}_1)(P \mid t'!\langle n_1 \rangle. \mathbf{0})$  has an action on  $t'$  not present in  $(v \widetilde{m}_2')(Q' \mid C \mid [U \rightarrow \diamond]_c n_2)$ , we derive a contradiction with respect to the bisimilarity assumption.

2. The proof for Part 2 is also by contradiction. Assume that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_1)(Q \mid [U]_c n_2) \not\approx \hat{\ell} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid (\lambda x. t?(y).(y x)) n_2)$$

From the bisimilarity requirement we can observe

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_1)(Q \mid [U]_c n_2) \xrightarrow{\hat{\ell}} \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid [U]_c n_2)$$

But then we can observe an action on the fresh name  $t$  on process

$$\Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1')(P' \mid (\lambda x. t?(y).(y x)) n_1) \triangleright \diamond$$

that cannot be observed by process  $\Gamma; \emptyset; \Delta'_2 \vdash (v \widetilde{m}_2')(Q' \mid [U]_c n_2)$ —a contradiction.

3. For the proof of Part 3 we do a case analysis on the transitions for checking the bisimulation requirements. The most interesting case is when, for some  $\Delta'_1$ :

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P \mid t?(x).(x n_1)) \xrightarrow{t?([U]_c)} \Gamma; \Delta'_1 \vdash (v \widetilde{m}_1'')(P \mid [U]_c n_1)$$

From the freshness of  $t$  we can derive that, for some  $\Delta'_2$  and  $Q''$

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q \mid t?(x).(x n_2)) \xrightarrow{t?([U]_c)} \Gamma; \Delta'_2 \vdash (v \widetilde{m}_2'')(Q'' \mid [U]_c n_2)$$

From the bisimulation requirement of the hypothesis we have that

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P \mid t!\langle n_1 \rangle. \mathbf{0}) \xrightarrow{t!\langle n_1 \rangle} \Delta'_1 \vdash (\nu \widetilde{m}'_1)(P)$$

implies

$$\Gamma; \Delta_2 \vdash (\nu \widetilde{m}_2)(Q \mid t!\langle n_2 \rangle. \mathbf{0}) \xrightarrow{t!\langle n_2 \rangle} \Delta'_2 \vdash (\nu \widetilde{m}'_2)(Q')$$

for some  $\Delta'_1, \Delta'_2$  and

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \vdash (\nu \widetilde{m}'_1)(P \mid t?(x).( \nu s)(s?(y).(x y) \mid \bar{s}!\langle n_1 \rangle. \mathbf{0})) \\ \approx^H \Delta'_2 \vdash (\nu \widetilde{m}'_2)(Q' \mid t?(x).( \nu s)(s?(y).(x y) \mid \bar{s}!\langle n_2 \rangle. \mathbf{0})) \end{aligned}$$

Whenever

$$\begin{aligned} \Gamma; \Delta'_1 \vdash (\nu \widetilde{m}'_1)(P \mid t?(x).( \nu s)(s?(y).(x y) \mid \bar{s}!\langle n_1 \rangle. \mathbf{0})) \\ \xrightarrow{t?[U]_c} \Delta''_1 \vdash (\nu \widetilde{m}''_1)(P \mid (\nu s)(s?(y).([U]_c y) \mid \bar{s}!\langle n_1 \rangle. \mathbf{0})) \\ \xrightarrow{\tau_d} \Delta'_1 \vdash (\nu \widetilde{m}''_1)(P \mid [U]_c n_1) \end{aligned}$$

then for some  $Q''_2$

$$\begin{aligned} \Gamma; \Delta'_2 \vdash (\nu \widetilde{m}'_2)(Q' \mid t?(x).( \nu s)(s?(y).(x y) \mid \bar{s}!\langle n_2 \rangle. \mathbf{0})) \\ \xrightarrow{t?[U]_c} \Delta''_2 \vdash (\nu \widetilde{m}''_2)(Q'' \mid (\nu s)(s?(y).([U]_c y) \mid \bar{s}!\langle n_2 \rangle. \mathbf{0})) \\ \xrightarrow{\tau_d} \Delta'_2 \vdash (\nu \widetilde{m}''_2)(Q'' \mid [U]_c n_2) \end{aligned}$$

which concludes the case.

4. For the proof of Part 4, let  $\mathfrak{R}$  be the typed relation

$$\begin{aligned} \mathfrak{R} = \{ \Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P\{n_1/x\}) \approx^H \Delta_2 \vdash (\nu \widetilde{m}_2)(Q\{m_1/x\}) \mid \\ \Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P\{n/x\} \mid t_1!\langle n_1 \rangle. \mathbf{0}) \approx^H \Delta_2 \vdash (\nu \widetilde{m}_2)(Q\{n/x\} \mid t_1!\langle m_1 \rangle. \mathbf{0}) \} \end{aligned}$$

Suppose that  $(\nu \widetilde{m}_1)(P\{n_1/x\})$  moves:

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P\{n_1/x\}) \xrightarrow{\ell} \Delta'_1 \vdash (\nu \widetilde{m}'_1)(P'\{n_1/x\})$$

We need to show a matching action from  $(\nu \widetilde{m}_2)(Q\{m_1/x\})$ ; we proceed to show that  $\mathfrak{R}$  is a higher-order bisimulation by a case analysis on the subject/shape of action  $\ell$ . There are three cases:

(a) If  $\text{subj}(\ell) \neq n_1$  then the proof is straightforward from the premise of the proposition. First observe that

$$\Gamma; \Delta_1 \vdash (\nu \widetilde{m}_1)(P\{n/x\}) \mid t!\langle n_1 \rangle. \mathbf{0} \xrightarrow{\ell} \Delta'_1 \vdash (\nu \widetilde{m}'_1)(P'\{n/x\}) \mid t!\langle n_1 \rangle. \mathbf{0}$$

implies

$$\Gamma; \Delta_2 \vdash (\nu \widetilde{m}_2)(Q\{n/x\}) \mid t!\langle m_2 \rangle. \mathbf{0} \xrightarrow{\check{\ell}} \Delta'_2 \vdash (\nu \widetilde{m}'_2)(Q'\{n/x\}) \mid t!\langle m_2 \rangle. \mathbf{0}$$

for some  $\Delta'_2$  and

$$\Gamma; \Delta_2 \vdash (\nu \widetilde{m}'_1)(P'\{n/x\}) \mid t!\langle n_1 \rangle. \mathbf{0} \mid C_1 \approx^H \Delta'_2 \vdash (\nu \widetilde{m}'_2)(Q'\{n/x\}) \mid t!\langle m_1 \rangle. \mathbf{0} \mid C_2$$

with  $C_1 = t \leftarrow_H n_1$  and  $C_2 = t \leftarrow_H m_1$  if  $\ell$  and  $\check{\ell}$  are output actions,  $C_1 = \mathbf{0}$  and  $C_2 = \mathbf{0}$  otherwise. From here we can deduce that

$$\Gamma; \Delta_2 \vdash (\nu \widetilde{m}_1)(Q\{m_1/x\}) \xrightarrow{\check{\ell}} \Delta'_2 \vdash (\nu \widetilde{m}'_2)(Q'\{m_1/x\})$$

Furthermore, we can easily see that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_1')(P\{n_1/x\}) \mid C_1 \text{ \&R } \Delta_2' \vdash (v \widetilde{m}_2')(Q'\{m_1/x\}) \mid C_2$$

(b)  $\text{subj}(\ell) = n_1$ . We distinguish two sub-cases:

- $n_1 = m_1$ . The case is similar to the previous case.
- $n_1 \neq m_1$ . From the premise and Part 1 of this lemma we get that  $\overline{n}_1 \in \text{fn}(P)$  and  $\overline{m}_1 \in \text{fn}(Q)$ . The latter implies that this case is not possible, since no external action  $\ell$  would be observed, because of the typed transition requirement.

(c)  $\ell = \tau$ . This implies the untyped transitions:

$$(v \widetilde{m}_1)(P\{n_1/x\}) \xrightarrow{\ell'_{11}} (v \widetilde{m}_{11})(P_1\{n_1/x\}) \tag{33}$$

$$(v \widetilde{m}_1)(P\{n_1/x\}) \xrightarrow{\ell'_{12}} (v \widetilde{m}_{12})(P_2\{n_1/x\}) \tag{34}$$

$$\ell'_{11} \simeq \ell'_{12} \tag{35}$$

We distinguish two sub-cases:

- $\text{subj}(\ell'_{11}) \neq n_1$ . This case is similar to Case 1 of this proof.
- $\text{subj}(\ell'_{11}) = n_1$ . First observe that

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P\{n/x\} \mid t!\langle n_1 \rangle.\mathbf{0}) \xrightarrow{\ell''_{11}} \Delta'_1 \vdash (v \widetilde{m}_1')(P_1\{n/x\} \mid t!\langle n_1 \rangle.\mathbf{0})$$

for some  $\Delta'_1$  with  $\ell''_{11}\{n_1/n\} = \ell'_{11}$ , which implies

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}_2)(Q\{n/x\} \mid t!\langle m_1 \rangle.\mathbf{0}) \xrightarrow{\ell''_{21}} \Delta'_2 \vdash (v \widetilde{m}_2')(Q_1\{n/x\} \mid t!\langle m_1 \rangle.\mathbf{0})$$

with  $\ell''_{21}\{m_1/n\} = \ell'_{21}$ , which in turn implies

$$(v \widetilde{m}_2)(Q\{m_1/x\}) \xrightarrow{\ell'_{22}} (v \widetilde{m}_{21})(Q_1\{m_1/x\}) \tag{36}$$

Also observe that

$$\begin{aligned} \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)(P\{n/x\} \mid t!\langle n_1 \rangle.\mathbf{0}) \\ \xrightarrow{t!\langle n_1 \rangle} \Delta_1''' \vdash (v \widetilde{m}_1'')(P\{n/x\}) \end{aligned}$$

for some  $\Delta_1'''$  which implies

$$\begin{aligned} \Gamma; \emptyset; \Delta_2 \vdash (v \widetilde{m}_2)(Q\{n/x\} \mid t!\langle m_1 \rangle.\mathbf{0}) \\ \xrightarrow{t!\langle m_1 \rangle} \Delta_2'''' \vdash (v \widetilde{m}_2'')(Q'\{n/x\}) \end{aligned}$$

for some  $\Delta_2''''$  with

$$\begin{aligned} \Gamma; \emptyset; \Delta_1'''' \vdash (v \widetilde{m}_1''')(P\{n/x\} \mid t' \leftarrow_{\text{H}} n_1) \\ \approx^{\text{H}} \Delta_2'''' \vdash (v \widetilde{m}_2''')(Q'\{n/x\} \mid t' \leftarrow_{\text{H}} m_1) \end{aligned}$$

From here observe that for  $U = \Delta_1''''(n_1)$

$$\begin{aligned} \Gamma; \emptyset; \Delta_1'''' \vdash (v \widetilde{m}_1)(P\{n/x\} \mid t' \leftarrow_{\text{H}} n_1) \\ \xrightarrow{t' ? \langle [U]_{\text{c}} \rangle} \xrightarrow{\tau_{\text{d}}} \Delta_1'' \vdash (v \widetilde{m}_1'')(P\{n/x\} \mid [U]^x\{n_1/x\}) \end{aligned}$$

for some  $\Delta_1''$ , which implies

$$\Gamma; \emptyset; \Delta_2''' \vdash (v \widetilde{m}_2)(Q\{n/x\} \mid t' \leftrightarrow_{\text{H}} m_1)$$

$$\stackrel{t'?(U)c}{\vdash} \stackrel{\tau_d}{\vdash} \Delta_2' \vdash (v \widetilde{m}_2'')(Q'\{n/x\} \mid [U]^x\{m_1/x\})$$

for some  $\Delta_2''$  with

$$\Gamma; \emptyset; \Delta_1'' \vdash (v \widetilde{m}_1'')(P\{n/x\} \mid [U]^x\{n_1/x\})$$

$$\approx^{\text{H}} \Delta_2'' \vdash (v \widetilde{m}_2'')(Q'\{n/x\} \mid [U]^x\{m_1/x\})$$

From (34), i.e., the fact that the two parallel components of the process interact on name  $n_1$ , we can see that, for some  $\Delta_1'''$

$$\Gamma; \emptyset; \Delta_1'' \vdash (v \widetilde{m}_1'')(P\{n/x\} \mid [U]^x\{n_1/x\})$$

$$\stackrel{\tau}{\vdash} \Delta_1''' \vdash (v \widetilde{m}_1'')(P_2\{n/x\} \mid C_1 \mid t'!\langle n_1 \rangle.\mathbf{0})$$

where  $C_1 = \mathbf{0}$  if the action on  $[U]^x\{n_1/x\}$  is not an input action and  $C_1 = [U']^a$  otherwise. This in turn implies from Part 2 of this lemma

$$\Gamma; \emptyset; \Delta_2'' \vdash (v \widetilde{m}_2'')(Q'\{n/x\} \mid [U]^x\{m_1/x\})$$

$$\stackrel{\tau}{\vdash} \Delta_2''' \vdash (v \widetilde{m}_2'')(Q_2\{n/x\} \mid C_2 \mid t'!\langle m_1 \rangle.\mathbf{0}) \tag{37}$$

for some  $\Delta_2'''$  and  $C_2 = \mathbf{0}$  when the action on  $[U]^x\{m_1/x\}$  is not an input action and  $C_2 = [U']^b$  otherwise. It is then implied that

$$\Gamma; \emptyset; \Delta_1''' \vdash (v \widetilde{m}_1'')(P_2\{n/x\} \mid C_1 \mid t'!\langle n_1 \rangle.\mathbf{0})$$

$$\approx^{\text{H}} \Delta_2''' \vdash (v \widetilde{m}_2'')(Q_2\{n/x\} \mid C_2 \mid t'!\langle m_1 \rangle.\mathbf{0}) \tag{38}$$

where (37) implies the untyped transition

$$(v \widetilde{m}_2'')(Q'\{n/x\} \mid [U]^x\{m_1/x\}) \stackrel{\ell_{22}''}{\Longrightarrow} (v \widetilde{m}_2'')(Q_2\{n/x\} \mid [U]^x\{m_1/x\})$$

and furthermore,

$$(v \widetilde{m}_2'')(Q'\{m_1/x\}) \stackrel{\ell_{22}'''}{\Longrightarrow} (v \widetilde{m}_2'')(Q_2\{m_1/x\})$$

with  $\ell_{22}''\{m_1/n\} = \ell_{22}'''$ . From the last result and (36) we get

$$\Gamma; \emptyset; \Delta_2 \vdash (v \widetilde{m}_2)(Q\{m_1/x\})$$

$$\stackrel{\tau}{\vdash} \Delta_2' \vdash (v \widetilde{m}_2'')(Q''\{m_1/x\})$$

Furthermore, from (38) we can get that, for some  $\Delta_3$ ,

$$\Gamma; \Delta_1''' \vdash (v \widetilde{m}_1'')(P_2\{n/x\} \mid C_1 \mid t'!\langle n_1 \rangle.\mathbf{0}) \stackrel{\ell_{12}''}{\vdash}$$

$$\Delta_3 \vdash (v \widetilde{m}_1''')(P'\{n/x\} \mid C_1 \mid t'!\langle n_1 \rangle.\mathbf{0})$$

which implies

$$\Gamma; \Delta_2''' \vdash (v \widetilde{m}_2'')(Q_2\{n/x\} \mid C_2 \mid t'!\langle m_1 \rangle.\mathbf{0}) \stackrel{\ell_{22}'''}{\vdash}$$

$$\Delta_4 \vdash (v \widetilde{m}_2''')(Q''\{n/x\} \mid C_2 \mid t'!\langle m_1 \rangle.\mathbf{0})$$

and

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1''')(P'\{n/x\} \mid C_1 \mid t'!\langle n_1 \rangle.\mathbf{0}) \approx^{\text{H}}$$

$$\Delta_4 \vdash (v \widetilde{m}_2''')(Q''\{n/x\} \mid C_2 \mid t'!\{n_1\}.\mathbf{0})$$

which in turn implies the required conclusion:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1''')(P'\{m_1/x\} \mid C_1) \Re \Delta'_2 \vdash (v \widetilde{m}_2''')(Q''\{m_1/x\} \mid C_2)$$

□

A process substitution lemma is useful for showing the contextuality property for higher-order and characteristic bisimilarities. Before we state and prove a process substitution lemma, we give an intermediate result. (This is Lemma 4 in the main text.)

**Lemma 16** (Trigger substitution) *Let  $P$  and  $Q$  be processes. Suppose that all  $t_i, i \in I$  are fresh names. If*

$$\begin{aligned} \Gamma; \Delta_1 \vdash (v \widetilde{m}_1) \left( P \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) n_i \right) \\ \approx^H \Delta_2 \vdash (v \widetilde{m}_2) \left( Q \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) m_i \right) \end{aligned}$$

then for all  $\lambda \widetilde{x}. R$  there exist  $\Delta'_1, \Delta'_2$  such that

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid (\lambda \widetilde{x}. R) \widetilde{n}) \approx^H \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid (\lambda \widetilde{x}. R) \widetilde{m}).$$

*Proof* We prove the result up-to the application of names  $n_i$  and  $m_i$  to process  $R$ . Let  $\Re$  be the relation

$$\begin{aligned} \Re = \left\{ (\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid R\{\widetilde{n}/\widetilde{x}\}) \right. \\ \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid R\{\widetilde{m}/\widetilde{x}\}) \mid \forall \lambda \widetilde{x}. R, \exists \Delta'_1, \Delta'_2. \\ \Gamma; \Delta_1 \vdash (v \widetilde{m}_1) \left( P \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) n_i \right) \\ \left. \vdash \approx^H \Delta_2 (v \widetilde{m}_2) \left( Q \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) m_i \right) \right\} \end{aligned}$$

We show that  $\Re$  is a higher-order bisimulation. The proof is done by a case analysis on the actions that can be observed on the pairs of processes, so to check their higher-order bisimulation requirements. There are three cases:

1. Suppose an action from  $P$ , for some  $\Delta'_1$ :

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid R\{\widetilde{n}/\widetilde{x}\}) \xrightarrow{\ell} \Delta'_1 \vdash (v \widetilde{m}_1')(P' \mid R\{\widetilde{n}/\widetilde{x}\})$$

This transition implies, for some  $\Delta'_3$ , the following:

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1)P \xrightarrow{\ell} \Delta'_3 \vdash (v \widetilde{m}_1')P'$$

which in turn implies, for some  $\Delta_5$ :

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1) \left( P \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) n_i \right)$$

$$\xrightarrow{\ell} \Delta_5 \vdash (v \widetilde{m}_1') \left( P' \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) n_i \right)$$

The latter implies the following, from the definition of  $\approx^H$  and the freshness of  $t_i$ , for some  $\Delta_6$ :

$$\begin{aligned} & \Gamma; \Delta_2 \vdash (v \widetilde{m}_2') \left( Q \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) m_i \right) \\ & \xRightarrow{\check{\ell}} \Delta_6 \vdash (v \widetilde{m}_2') \left( Q' \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) n_i \right) \end{aligned}$$

and

$$\begin{aligned} & \Gamma; \emptyset; \Delta_5 \vdash (v \widetilde{m}_1') (P' \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) n_i \mid C_1) \\ & \approx^H \Delta_6 \vdash (v \widetilde{m}_2') (Q' \mid \prod_{i \in I} (\lambda x. t_i?(y).(y x)) m_i \mid C_2) \end{aligned} \tag{39}$$

where  $C_1, C_2$  are higher-order trigger processes if  $\ell, \check{\ell}$  are output actions, and  $C_1 = C_2 = \mathbf{0}$  otherwise. At this point we can infer, for some  $\Delta'_4$ :

$$\Gamma; \Delta_4 \vdash (v \widetilde{m}_2)(Q) \xRightarrow{\check{\ell}} \Delta'_4 \vdash (v \widetilde{m}_2') Q'$$

which in turn implies, for some  $\Delta''_2$ :

$$\Gamma; \Delta'_2 \vdash (v \widetilde{m}_2')(Q \mid R\{\widetilde{m}/\widetilde{x}\}) \xRightarrow{\check{\ell}} \Delta''_2 \vdash (v \widetilde{m}_2')(Q' \mid R\{\widetilde{m}/\widetilde{x}\})$$

Equation (39) and the definition of  $\mathfrak{R}$  imply the desired conclusion for the case:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1')(P' \mid R\{\widetilde{n}/\widetilde{x}\} \mid C_1) \mathfrak{R} \Delta''_2 \vdash (v \widetilde{m}_2')(Q' \mid R\{\widetilde{m}/\widetilde{x}\} \mid C_2)$$

2. Suppose an action from  $R$ :

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid R\{\widetilde{n}/\widetilde{x}\}) \xrightarrow{\ell} \Delta''_1 \vdash (v \widetilde{m}_1')(P \mid R\{\widetilde{n}/\widetilde{x}\})$$

for some  $\Delta''_1$ . We identify three sub-cases:

- i.  $\text{subj}(\ell) \neq n_i$ , i.e. the subject of  $\ell$  is not in  $\widetilde{n}$ . The case is similar as above.
- ii.  $\text{subj}(\ell) = n_k$  and  $n_k = m_k$ . From the definition of  $\mathfrak{R}$  we get that

$$\begin{aligned} & \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1) (P \mid \prod_{i \in I} t_i?(x).(x n_i)) \\ & \xrightarrow{t_k?(\llbracket U \rightarrow \diamond \rrbracket_{\mathbf{c}})} \Delta_3 \vdash (v \widetilde{m}_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid \llbracket U \rrbracket_{\mathbf{c}} n_k \right) \end{aligned}$$

for some  $\Delta_3$ . Recall that  $\llbracket U \rightarrow \diamond \rrbracket_{\mathbf{c}} = \lambda x. \llbracket U \rrbracket^x$  (cf. Fig. 6); this transition implies

$$\begin{aligned} & \Gamma; \emptyset; \Delta_2 \vdash (v \widetilde{m}_2) (Q \mid \prod_{i \in I} t_i?(x).(x m_i)) \\ & \xrightarrow{t_k?(\llbracket U \rightarrow \diamond \rrbracket_{\mathbf{c}})} \Delta_4 \vdash (v \widetilde{m}_1) \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid \llbracket U \rrbracket^x \{m_k/x\} \right) \end{aligned}$$

and from bisimilarity up-to deterministic transition (Lemma 1):

$$\Gamma; \Delta_3 \vdash (v \widetilde{m}_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid \llbracket U \rightarrow \diamond \rrbracket_{\mathbf{c}} n_k \right)$$

$$\begin{aligned} & \xrightarrow{\tau_\beta} \Delta_3 \vdash (\nu \widetilde{m}_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid [U]^x \{n_k/x\} \right) \\ & \approx^H \Delta_4 \vdash (\nu \widetilde{m}_1) \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid [U]^x \{m_k/x\} \right) \end{aligned}$$

for some  $\Delta_4$ . From the shape of  $[U]^x$  we can observe

$$\begin{aligned} \Gamma; \emptyset; \Delta_3 \vdash (\nu \widetilde{m}_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid [U]^x \{n_k/x\} \right) \\ \xrightarrow{\ell} \Delta'_3 \vdash (\nu \widetilde{m}'_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid t'!(n_k).\mathbf{0} \right) \end{aligned}$$

implies, for some  $\Delta'_4$ :

$$\begin{aligned} \Gamma; \emptyset; \Delta_4 \vdash (\nu \widetilde{m}_2) \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid [U]^x \{m_k/x\} \right) \\ \xRightarrow{\ell} \dots \Delta'_4 \vdash (\nu \widetilde{m}'_2) \left( Q'' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid t'!(m_k).\mathbf{0} \right) \end{aligned}$$

and furthermore, from Part 3 of Lemma 15

$$\begin{aligned} \Gamma; \emptyset; \Delta'_3 \vdash (\nu \widetilde{m}'_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid t'?(y).(y n_k) \right) \\ \approx^H \Delta'_4 \vdash (\nu \widetilde{m}'_2) \left( Q'' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid t'?(y).(y m_k) \right) \end{aligned}$$

that implies from the definition of  $\mathfrak{R}$  that for all  $R$  such that  $\{\widetilde{x}\} \subseteq \text{fv}(R)$

$$\Gamma; \Delta'_3 \vdash (\nu \widetilde{m}'_1)(P \mid R\{\widetilde{n}/\widetilde{x}\}) \mathfrak{R} \Delta'_4 \vdash (\nu \widetilde{m}'_2)(Q'' \mid R\{\widetilde{m}/\widetilde{x}\})$$

The case concludes when we verify that, for some  $\Delta''_2$ , we have:

$$\Gamma; \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid R\{\widetilde{m}/x\}) \xRightarrow{\ell} \Delta''_2 \vdash (\nu \widetilde{m}'_1)(Q'' \mid R'\{\widetilde{m}/x\})$$

- iii.  $\text{subj}(\ell) = n_k$  and  $n_k \neq m_k$ . This case is not possible. Lemma 15 implies that  $n_k$  is a session and  $\overline{n_k} \in \text{fn}(P)$ . From the definition of typed transition (Definition 5) we get that we cannot observe  $\ell$  on  $R\{\widetilde{n}/\widetilde{x}\}$ , because  $\overline{n_k} \in \text{fn}(P)$  and  $(\Gamma; \emptyset; \Delta) \not\vdash \ell$ .

3. Suppose the interaction of  $P$  and  $R$ , for some  $\Delta'_1$ :

$$\Gamma; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid R\{\widetilde{n}/\widetilde{x}\}) \xrightarrow{\tau} \Delta'_1 \vdash (\nu \widetilde{m}'_1)(P' \mid R'\{\widetilde{n}/\widetilde{x}\})$$

From the typed reduction definition (Definition 5) we get that

$$\Gamma; \Delta_3 \vdash (\nu \widetilde{m}_1)P \xrightarrow{\ell_1} \Delta_R \vdash (\nu \widetilde{m}_1)P' \tag{40}$$

$$\begin{aligned} \Gamma; \Delta'_1 \vdash R\{\widetilde{n}/\widetilde{x}\} \xrightarrow{\ell_2} \Delta'_R \vdash R'\{\widetilde{n}/\widetilde{x}\} \\ \ell_1 \circ \ell_2 \tag{41} \end{aligned}$$

We distinguish several subcases:

- i.  $\ell_1 = n_k?(V)$  and  $\ell_2 = (\nu \widetilde{m})(\overline{n_k}!(V))$ . From the requirement of  $\mathfrak{R}$  we get that there exists  $U \rightarrow \diamond$  such that, for some  $\Delta_3$ :

$$\begin{aligned} \Gamma; \emptyset; \Delta_1 \vdash (\nu \widetilde{m}_1) \left( P \mid \prod_{i \in I} t_i?(x).(x n_i) \right) \\ t_k?(U \rightarrow \diamond) \xrightarrow{\tau_d} \Delta_3 \vdash (\nu \widetilde{m}'_1) \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid [U]^x \{n_k/x\} \right) \end{aligned}$$

which in turn implies, for some  $\Delta_4$ , that

$$\Gamma; \emptyset; \Delta_2 \vdash (v \widetilde{m}_2) \left( Q \mid \prod_{i \in I} t_i?(x).(x m_i) \right) \\
 \xrightarrow{t_k? \{ [U \rightarrow \circ]_c \}} \xrightarrow{\tau_d} \Delta_4 \vdash (v \widetilde{m}_2') \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid [U]^x \{m_k/x\} \right)$$

and

$$\Gamma; \emptyset; \Delta_3 \vdash (v \widetilde{m}_1') \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid [U]^x \{n_k/x\} \right) \\
 \approx^H \Delta_4 \vdash (v \widetilde{m}_2') \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid [U]^x \{m_k/x\} \right)$$

From the shape of  $[U]^x$  we can observe the interaction between  $[U]^x$  and  $P$  to obtain that if, for some  $\Delta'_3$  and some  $[U']_c$  (cf. Definition 13), we have

$$\Gamma; \emptyset; \Delta_3 \vdash (v \widetilde{m}_1') \left( P \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid [U]^x \{n_k/x\} \right) \\
 \xrightarrow{\tau} \Delta'_3 \vdash (v \widetilde{m}_1'') \left( P' \{ [U']_c / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid t'_k!(n_k).\mathbf{0} \right)$$

then

$$\Gamma; \emptyset; \Delta_4 \vdash (v \widetilde{m}_2') \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid [U]^x \{m_k/x\} \right) \\
 \xrightarrow{\tau} \Delta'_4 \vdash (v \widetilde{m}_2'') \left( Q'' \{ [U']_c / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid t'_k!(m_k).\mathbf{0} \right)$$

and

$$\Gamma; \emptyset; \Delta'_3 \vdash (v \widetilde{m}_1'') \left( P' \{ [U']_c / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid t'_k!(n_k).\mathbf{0} \right) \\
 \approx^H \Delta'_4 \vdash (v \widetilde{m}_2'') \left( Q'' \{ [U']_c / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid t'_k!(m_k).\mathbf{0} \right) \tag{42}$$

for some  $\Delta'_4$ . From Lemma 15(3) we obtain

$$\Gamma; \emptyset; \Delta'_3 \vdash (v \widetilde{m}_1'') \left( P' \{ [U']_c / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \mid t'_k?(x).(x n_k) \right) \\
 \approx^H \Delta'_4 \vdash (v \widetilde{m}_2'') \left( Q'' \{ [U']_c / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \mid t'_k?(x).(x m_k) \right) \tag{43}$$

From the definition of  $\mathfrak{R}$  we get that

$$\Gamma; \emptyset; \Delta'_3 \vdash (v \widetilde{m}_1'') (P' \{ [U']_c / x \} \mid R' \{ \widetilde{n} / \widetilde{x} \}) \\
 \mathfrak{R} \Delta'_4 \vdash (v \widetilde{m}_2'') (Q'' \{ [U']_c / x \} \mid R' \{ \widetilde{m} / \widetilde{x} \})$$

From the above result we can match actions in (40) and (41):

$$(v \widetilde{m}_2'') \left( Q' \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \right) \xrightarrow{m_k?(V)} (v \widetilde{m}_4) \left( Q'' \{ V / x \} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \right) \\
 R \{ \widetilde{m} / \widetilde{x} \} \xrightarrow{m_k!(V)} R' \{ \widetilde{n} / \widetilde{x} \} \quad m_k \in \widetilde{m}$$

to obtain, for some  $\Delta''_2$ , that

$$\Gamma; \Delta'_2 \vdash (v \widetilde{m}_2) (Q \mid R \{ \widetilde{n} / \widetilde{x} \}) \iff \Delta''_2 \vdash (v \widetilde{m}_2') (Q'' \{ V / x \} \mid R' \{ \widetilde{m} / \widetilde{x} \})$$

Furthermore the definition of  $\mathfrak{R}$  and (42) allow us to conclude the case:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1') (P' \{ V / x \} \mid R' \{ \widetilde{n} / \widetilde{x} \}) \mathfrak{R} \Delta'_2 \vdash (v \widetilde{m}_2') (Q'' \{ V / x \} \mid R' \{ \widetilde{m} / \widetilde{x} \})$$



- ii. An important sub-case is when  $\ell_1 = n?(n_k)$  and  $\ell_2 = n!(n_k)$ . From the definition of  $\mathfrak{R}$  we have that

$$\begin{aligned} \Gamma; \Delta_1 \vdash (v \widetilde{m}_1) \left( P \mid \prod_{i \in I} t_i?(x).(x n_i) \right) \\ \xrightarrow{n?(m)} \Delta_3 \vdash (v \widetilde{m}_1) \left( P'\{m/x\} \mid \prod_{i \in I} t_i?(x).(x n_i) \right) \end{aligned}$$

for some  $\Delta_3$ . This transition implies, for some  $\Delta_4$ , that

$$\begin{aligned} \Gamma; \emptyset; \Delta_2 \vdash (v \widetilde{m}_2) (Q \mid \prod_{i \in I} t_i?(x).(x m_i)) \\ \xrightarrow{n?(m)} \Delta_4 \vdash (v \widetilde{m}_2) (Q'\{m/x\} \mid \prod_{i \in I} t_i?(x).(x m_i)) \end{aligned} \tag{44}$$

and

$$\begin{aligned} \Gamma; \Delta_3 \vdash (v \widetilde{m}_1) \left( P'\{m/x\} \mid \prod_{i \in I} t_i?(x).(x n_i) \right) \\ \approx^H \Delta_4 \vdash (v \widetilde{m}_2) \left( Q'\{m/x\} \mid \prod_{i \in I} t_i?(x).(x m_i) \right) \end{aligned}$$

We infer from Lemma 15(4) that

$$\begin{aligned} \Gamma; \emptyset; \Delta'_3 \vdash (v \widetilde{m}_1) \left( P'\{n_k/x\} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x n_i) \right) \\ \approx^H \Delta'_4 \vdash (v \widetilde{m}_2) \left( Q'\{m_k/x\} \mid \prod_{i \in I \setminus \{k\}} t_i?(x).(x m_i) \right) \end{aligned}$$

which implies from the definition of  $\mathfrak{R}$  that

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1) (P'\{n_k/x\} \mid R\{\widetilde{m}/\widetilde{x}\}) \\ \mathfrak{R} \Delta'_2 \vdash (v \widetilde{m}_2) (Q'\{m_k/x\} \mid R'\{\widetilde{m}/\widetilde{x}\}) \end{aligned}$$

From (44) and (41) we obtain, for some  $\Delta''_2$ , the following

$$\Gamma; \Delta'_2 \vdash (v \widetilde{m}_2) (Q \mid R\{\widetilde{m}/\widetilde{x}\}) \Longrightarrow \Delta''_2 \vdash (v \widetilde{m}_2) (Q'\{m_k/x\} \mid R'\{\widetilde{m}/\widetilde{x}\})$$

which concludes the case.

- iii. The sub-case  $\ell_1 = n_k?(n_l)$  and  $\ell_2 = n_k!(n_l)$ . The proof is a consequence of the previous two sub-cases.
- iv. The rest of the sub-cases are similar (or easier) to the above cases. □

We can now state a process substitution lemma (Lemma 3 in the main text). Given a higher-order bisimulation under a trigger value substitution, we can generalise for any value substitution.

**Lemma 17** (Process substitution) *Let  $P_1$  and  $P_2$  be processes, with  $z \in f\mathcal{V}(P_1)$  and  $z \in f\mathcal{V}(P_2)$ . Also, let  $t$  be a fresh name. If*

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1) (P_1\{\lambda x.t?(y).(y x)/z\}) \approx^H \Delta_2 \vdash (v \widetilde{m}_2) (P_2\{\lambda x.t?(y).(y x)/z\})$$

then for all  $\lambda x.R$  there exist  $\Delta'_1$  and  $\Delta'_2$  such that

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1) (P_1\{\lambda x.R/z\}) \approx^H \Delta'_2 \vdash (v \widetilde{m}_2) (P_2\{\lambda x.R/z\})$$

*Proof* Consider the typed relation (for readability, we omit type information):

$$\mathfrak{R} = \{(\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P_1\{\lambda x. R/z\}), \Delta'_2 \vdash (v \widetilde{m}_2)(P_2\{\lambda x. R/z\})) \mid \Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P_1\{\lambda x. t?(y).(y x)/z\}) \approx^H \Delta_2 \vdash (v \widetilde{m}_2)(P_2\{\lambda x. t?(y).(y x)/z\})\}$$

We show that  $\mathfrak{R}$  is a higher-order bisimulation. Suppose that

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P_1\{\lambda x. R/z\}) \xrightarrow{\ell} \Delta_3 \vdash (v \widetilde{m}_1)(P'_1\{\lambda x. R/z\}) \tag{45}$$

for some  $\Delta_3$ . We should exhibit an appropriate matching action from  $(v \widetilde{m}_2)(P_2\{\lambda x. R/z\})$ . Our analysis distinguishes two cases, depending on whether the substitution  $\{\lambda x. R/z\}$  has an effect on the action denoted by  $\ell$ :

1. Case  $P_1 \not\equiv Q \mid z n$ : That is, the substitution does not affect top-level processes. In other words, we can infer from the freshness of  $t$  that  $\text{subj}(\ell) \neq t$ . Furthermore, from the requirements of  $\mathfrak{R}$  we get that there exist  $\Delta'_1$  and  $P'_1$  such that

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}_1)(P_1\{\lambda x. t?(y).(y x)/z\}) \xrightarrow{\ell} \Delta''_1 \vdash (v \widetilde{m}_1)(P'_1\{\lambda x. t?(y).(y x)/z\})$$

which, in turn, implies that there exist  $\Delta'_2$  and  $P'_2$  such that

$$\Gamma; \Delta_2 \vdash (v \widetilde{m}'_2)(P_2\{\lambda x. t?(y).(y x)/z\}) \xrightarrow{\check{\ell}} \Delta''_2 \vdash (v \widetilde{m}'_2)(P'_2\{\lambda x. t?(y).(y x)/z\}) \tag{46}$$

and

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}''_1)(P'_1\{\lambda x. t?(y).(y x)/z\} \mid C_1) \approx^H \Delta_2 \vdash (v \widetilde{m}''_2)(P'_2\{\lambda x. t?(y).(y x)/z\} \mid C_2)$$

with  $C_1$  (resp.,  $C_2$ ) being the higher-order trigger process in the cases where  $\ell = (v \widetilde{m})n!(V_1)$  (resp.,  $\check{\ell} = (v \widetilde{m}')n!(V_2)$ ), and  $C_1 = C_2 = \mathbf{0}$  otherwise. Because  $C_1$  and  $C_2$  are closed terms we can rewrite the substitution as:

$$\Gamma; \Delta_1 \vdash (v \widetilde{m}''_1)((P'_1 \mid C_1)\{\lambda x. t?(y).(y x)/z\}) \approx^H \Delta_2 \vdash (v \widetilde{m}''_2)((P'_2 \mid C_2)\{\lambda x. t?(y).(y x)/z\})$$

Since  $\ell, \check{\ell}$  do not act on the substitution, we can consider the same transition with any  $\lambda x. R$  instead of  $\lambda x. t?(y).(y x)$ . Thus, from the definition of  $\mathfrak{R}$ , we further deduce that

$$\Gamma; \Delta'_3 \vdash (v \widetilde{m}''_1)((P'_1 \mid C_1)\{\lambda x. R/z\}) \mathfrak{R} \Delta'_4 \vdash (v \widetilde{m}''_2)((P'_2 \mid C_2)\{\lambda x. R/z\}) \tag{47}$$

Note that  $C_1$  and  $C_2$  are used to meet the bisimulation requirements for the output case. From (46) we can derive the transition

$$\Gamma; \Delta'_2 \vdash (v \widetilde{m}_2)(P_2\{\lambda x. R/z\}) \xrightarrow{\check{\ell}} \Delta_4 \vdash (v \widetilde{m}'_2)(P'_2\{\lambda x. R/z\})$$

Equation (47) concludes the case.

2. Case  $P_1 \equiv P \mid \prod_{i \in I} z n_i \mid z n_1$ , such that  $P \not\equiv P' \mid z n'$ . This is the case where action  $\ell$  might happen on the process that is being substituted (note that a substituted process needs to be applied first).

We identify two sub-cases, depending on the source of the action  $\ell$ :

- (a) Consider the following transition, for some  $\Delta_3$ :

$$\begin{aligned} &\Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1) \left( (P \mid \prod_{i \in I} z n_i \mid z n_1)\{\lambda x. R/z\} \right) \\ &\xrightarrow{\ell} \Delta_3 \vdash (v \widetilde{m}_1) \left( (P' \mid \prod_{i \in I} z n_i \mid z n_1)\{\lambda x. R/z\} \right) \end{aligned}$$

This sub-case is similar to the previous case.

- (b) Consider the following transition, for some  $\Delta_3$ , and assuming that  $Q = P \mid \prod_{i \in I} z n_i$ :

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)((Q \mid z n_1)\{\lambda x. R/z\}) \xrightarrow{\tau} \Delta_3 \vdash (v \widetilde{m}_1)(Q\{\lambda z. R/z\} \mid R\{n_1/z\}) \tag{48}$$

which is the application of name  $n_1$  on abstraction  $\lambda x. R$ .

From the requirements of  $\mathfrak{R}$  we infer that

$$\begin{aligned} \Gamma; \emptyset; \Delta_1 \vdash (v \widetilde{m}_1)((Q \mid z n_1)\{\lambda x. t?(y).(y x)/z\}) \\ \xrightarrow{\tau} \Delta''_1 \vdash (v \widetilde{m}_1)(Q\{\lambda x. t?(y).(y x)/z\} \mid t?(y).(y n_1)) \end{aligned}$$

for some  $\Delta''_1$ . This implies that there exist  $P'_2$  and  $\Delta''_2$  such that

$$\Gamma \Delta_2 (v \widetilde{m}_2)(P_2\{\lambda x. t?(y).(y x)/z\}) \Longrightarrow \Delta''_2 (v \widetilde{m}_2)(P'_2\{\lambda x. t?(y).(y x)/z\}) \tag{49}$$

and

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1)(Q\{\lambda x. t?(y).(y x)/z\} \mid t?(y).(y n_1)) \\ \approx^H \Delta''_2 \vdash (v \widetilde{m}_2)(P'_2\{\lambda x. t?(y).(y x)/z\}) \end{aligned}$$

From the last pair we can see that for a fresh  $t'$  if

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1)(Q\{\lambda x. t?(y).(y x)/z\} \mid t?(y).(y n_1)) \\ \xrightarrow{t?(\lambda x. t'(y).(y x))} \Delta'''_1 \vdash (v \widetilde{m}_1)(Q\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. t'(y).(y x)) n_1) \end{aligned}$$

then from the freshness of  $t$ , there exist  $P''_2, \Delta'''_2$  such that

$$\begin{aligned} \Gamma; \emptyset; \Delta''_2 \vdash (v \widetilde{m}_2)(P'_2\{\lambda x. t?(y).(y x)/z\}) \\ \xrightarrow{\tau_\beta} \xrightarrow{t?(\lambda x. t'(y).(y x))} \xrightarrow{\tau_\beta} \begin{aligned} & (v m_2)((P_3 \mid z n_2)\{\lambda x. t?(y).(y x)/z\}) \\ & (v m_2)(P_3\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. t'(y).(y x)) n_2) \\ & \Delta'''_2 \vdash (v \widetilde{m}_2)(P''_2\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. t'(y).(y x)) n_2) \end{aligned} \end{aligned} \tag{50}$$

and

$$\begin{aligned} \Gamma; \emptyset; \Delta'''_1 \vdash (v \widetilde{m}_1)(Q\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. t'(y).(y x)) n_1) \\ \approx^H \Delta'''_2 \vdash (v \widetilde{m}_2)(P''_2\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. t'(y).(y x)) n_2) \end{aligned}$$

From Lemma 16 we can deduce that, for all  $\lambda x. R$ , there exist  $\Delta_5$  and  $\Delta_6$  such that

$$\begin{aligned} \Gamma; \emptyset; \Delta_5 \vdash (v \widetilde{m}_1)(Q\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. R) n_1) \\ \approx^H \Delta_6 \vdash (v \widetilde{m}_2)(P''_2\{\lambda x. t?(y).(y x)/z\} \mid (\lambda x. R) n_2) \end{aligned}$$

from the definition of  $\mathfrak{R}$  we have that for all  $\lambda x. R$ , if there exist  $\Delta_3$  and  $\Delta_4$

$$\Gamma \Delta_3 (v \widetilde{m}_1)(Q\{\lambda x. R/z\} \mid (\lambda x. R) n_1) \mathfrak{R} \Delta_4 (v \widetilde{m}_2)(P''_2\{\lambda x. R/z\} \mid (\lambda x. R) n_2) \tag{51}$$

We show that we can mimic first the transition in (49) and then the silent part of transitions (50) to get:

$$\begin{aligned} \Gamma; \emptyset; \Delta'_2 \vdash (v \widetilde{m}_2)(P_2\{\lambda x. R/z\}) \\ \Longrightarrow \Delta'_2 \vdash (v \widetilde{m}_2)(P'_2\{\lambda x. R/z\}) \\ \Longrightarrow \Delta_4 \vdash (v m_2)(P''_2\{\lambda x. R/z\} \mid (\lambda x. R) n_2) \end{aligned} \tag{52}$$

We showed that if (48) then (52) and (51) as required to show that  $\mathfrak{R}$  is a higher-order bisimulation.  $\square$

**Lemma 18**  $\approx^H \subseteq \approx$ .

*Proof* Let  $\mathfrak{R}$  be the typed relation (for readability, we omit typing information):

$$\mathfrak{R} = \{(P_1, Q_1) \mid \Gamma; \Delta_1 \vdash P_1 \approx^H \Delta_2 \vdash Q_1\}$$

We show that  $\mathfrak{R}$  is a context bisimulation (cf. Definition 12). Suppose that

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta'_1 \vdash P_2 \tag{53}$$

We need to infer an appropriate matching transition from  $Q_1$ . The proof proceeds by a case analysis on  $\ell$ . We distinguish four cases:  $\ell$  is not an output or a higher-order input action;  $\ell$  is a higher-order input action;  $\ell$  is a higher-order output;  $\ell$  is a first-order output.

1. Case  $\ell \notin \{(v \widetilde{m}_1)n!(\lambda \widetilde{x}. P), (v \widetilde{m}'_1)n!(\widetilde{m}_1), n?(\lambda \widetilde{x}. P)\}$ : We first notice that in this case the definition of  $\approx$  and  $\approx^H$  coincide, so we have to show the existence of  $Q_2$  and  $\Delta'_2$  such that:

$$\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{\ell} \Delta'_2 \vdash Q_2$$

and

$$\Gamma; \Delta'_1 \vdash P_2 \mathfrak{R} \Delta'_2 \vdash Q_2.$$

This is immediate from transition (53) and the definition of  $\approx^H$  (cf. Definition 17).

2. Case  $\ell = n?(\lambda \widetilde{x}. P)$ : In this case, the transition (53) can be written as

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{n?(\lambda \widetilde{z}. t?(y).(y \widetilde{z}))} \Delta''_1 \vdash P_2\{\lambda \widetilde{z}. t?(y).(y \widetilde{z})/x\}$$

for some  $\Delta''_1$ . In turn, the above transition and  $\mathfrak{R}$  imply the existence of  $Q_2$  and  $\Delta''_2$  such that:

$$\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{n?(\lambda \widetilde{z}. t?(y).(y \widetilde{z}))} \Delta''_2 \vdash Q_2\{\lambda \widetilde{z}. t?(y).(y \widetilde{z})/x\}$$

and

$$\Gamma; \Delta''_1 \vdash P_2\{\lambda \widetilde{z}. t?(y).(y \widetilde{z})/x\} \approx^H \Delta''_2 \vdash Q_2\{\lambda \widetilde{z}. t?(y).(y \widetilde{z})/x\}.$$

Then, by using the previous equality and Lemma 17, we may conclude that

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

for some  $\Delta'_1, \Delta'_2$ , for all  $P$  with  $\varepsilon_V(P) = \{\widetilde{x}\}$ , as required.

3. Case  $\ell = (v \widetilde{m}'_1)n!(\widetilde{m}_1)$ : In this case, transition (53) and  $\mathfrak{R}$  imply the existence of  $\Delta'_2$ , a process  $Q_2$ , and name  $m_2$  such that

$$\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{(v \widetilde{m}'_2)n!(m_2)} \Delta'_2 \vdash Q_2$$

and

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P_2 \mid t \leftarrow_H m_1) \approx^H \Delta'_2 \vdash (v \widetilde{m}'_2)(Q_2 \mid t \leftarrow_H m_2)$$

for some fresh  $t$ . From Case 2 of this proof (higher-order input) we can conclude that for all  $R$  with  $\text{fv}(R) = \{x\}$  and for some  $\Delta_1'', \Delta_2''$ :

$$\Gamma; \emptyset; \Delta_1' \vdash (v \widetilde{m}_1')(P_2 \mid t \leftarrow_{\text{H}} m_1) \xrightarrow[t_d]{t?(\lambda z. z?(x).R)} (v \widetilde{m}_1')(P_2 \mid (v s)(s?(x).R \mid \bar{s}!(m_1).\mathbf{0}))$$

$$\xrightarrow[t_d]{} \Delta_1'' \vdash (v \widetilde{m}_1')(P_2 \mid R\{m_1/x\})$$

and

$$\Gamma; \emptyset; \Delta_2' \vdash (v \widetilde{m}_2')(Q_2 \mid t \leftarrow_{\text{H}} m_2) \xrightarrow[t_d]{t?(\lambda z. z?(x).R)} (v \widetilde{m}_2')(Q_2 \mid (v s)(s?(x).R \mid \bar{s}!(m_2).\mathbf{0}))$$

$$\xrightarrow[t_d]{} \Delta_2'' \vdash (v \widetilde{m}_2')(Q_2 \mid R\{m_2/x\})$$

where, due to the deterministic internal transitions (cf. Definition 19), it is easy to see that

$$\Gamma; \Delta_1'' \vdash (v \widetilde{m}_1')(P_2 \mid R\{m_1/x\}) \approx^{\text{H}} \Delta_2'' \vdash (v \widetilde{m}_2')(Q_2 \mid R\{m_2/x\})$$

for all  $R$  with  $\text{fv}(R) = \{x\}$ , as required by the definition of  $\approx$  (Definition 12).

4. Case  $\ell = (v \widetilde{m}_1')n!(\lambda \tilde{x}. P)$ : This case is similar to the previous case but makes use of the alternative trigger,  $t \leftarrow_{\text{A}} V$  (cf. (15)). The definition of  $\mathfrak{H}$  and transition (53) allow us to infer the existence of some  $\Delta_2', Q$ , and  $Q_2$  such that

$$\Gamma; \Delta_2 \vdash Q_1 \xrightarrow{(v \widetilde{m}_2')n!(\lambda \tilde{x}. Q)} \Delta_2' \vdash Q_2$$

and

$$\Gamma; \Delta_1' \vdash (v \widetilde{m}_1')(P_2 \mid t \leftarrow_{\text{H}} \lambda \tilde{x}. P) \approx^{\text{H}} \Delta_2' \vdash (v \widetilde{m}_2')(Q_2 \mid t \leftarrow_{\text{H}} \lambda \tilde{x}. Q)$$

for some fresh  $t$ . Using Lemma 12, we above equality implies that

$$\Gamma; \Delta_1' \vdash (v \widetilde{m}_1')(P_2 \mid t \leftarrow_{\text{A}} \lambda \tilde{x}. P) \approx^{\text{H}} \Delta_2' \vdash (v \widetilde{m}_2')(Q_2 \mid t \leftarrow_{\text{A}} \lambda \tilde{x}. Q)$$

which in turn implies

$$\Gamma; \emptyset; \Delta_1' \vdash (v \widetilde{m}_1')(P_2 \mid t \leftarrow_{\text{A}} \lambda \tilde{x}. P)$$

$$\xrightarrow[t?(\lambda y. t'?(x).(x y))]{t?(\lambda y. t'?(x).(x y))} \Delta_1'' \vdash (v \widetilde{m}_1')(P_2 \mid (v s)((\lambda y. t'?(x).(x y))s \mid \bar{s}!(\lambda \tilde{x}. P).\mathbf{0}))$$

for some  $\Delta_1''$  and

$$\Gamma; \emptyset; \Delta_2' \vdash (v \widetilde{m}_2')(Q_2 \mid t \leftarrow_{\text{A}} \lambda \tilde{x}. Q)$$

$$\xrightarrow[t?(\lambda y. t'?(x).(x y))]{t?(\lambda y. t'?(x).(x y))} \Delta_2'' \vdash (v \widetilde{m}_1')(Q_2' \mid (v s)((\lambda y. t'?(x).(x y))s \mid \bar{s}!(\lambda \tilde{x}. Q).\mathbf{0}))$$

for some  $\Delta_2''$ , and

$$\Gamma; \emptyset; \Delta_1'' \vdash (v \widetilde{m}_1')(P_2 \mid (v s)((\lambda y. t'?(x).(x y))s \mid \bar{s}!(\lambda \tilde{x}. P).\mathbf{0}))$$

$$\approx^{\text{H}} \Delta_2'' \vdash (v \widetilde{m}_1')(Q_2' \mid (v s)((\lambda y. t'?(x).(x y))s \mid \bar{s}!(\lambda \tilde{x}. Q).\mathbf{0}))$$

From the Case 2 of this proof (higher-order input), we have

$$\Gamma; \emptyset; \Delta_1'' \vdash (v \widetilde{m}_1')(P_2 \mid (v s)((\lambda y. y?(x).R) s \mid \bar{s}!(\lambda \tilde{x}. P).\mathbf{0}))$$

$$\approx^{\text{H}} \Delta_2'' \vdash (v \widetilde{m}_1')(Q_2' \mid (v s)((\lambda y. y?(x).R) s \mid \bar{s}!(\lambda \tilde{x}. Q).\mathbf{0}))$$

for all  $R$  with  $\text{fv}(R) = \{x\}$ . Now, using deterministic transitions (cf. Definition 19) is easy to see that

$$\Gamma; \Delta_1'' \vdash (v \widetilde{m}_1')(P_2 \mid R\{\lambda \tilde{x}. P/y\}) \approx^{\text{H}} \Delta_2'' \vdash (v \widetilde{m}_2')(Q_2 \mid R\{\lambda \tilde{x}. Q/y\})$$

for all  $R$  with  $\text{fv}(R) = \{x\}$ , as required by the definition of  $\approx$  (cf. Definition 12).  $\square$

**Lemma 19**  $\approx \subseteq \cong$ .

*Proof* We prove that  $\approx$  (cf. Definition 12) satisfies the three defining properties of  $\cong$ : reduction closure, barb preservation, and congruence (cf. Definition 11).

I. *Reduction-closed* Let  $\Gamma; \Delta_1 \vdash P_1 \approx \Delta_2 \vdash P_2$ . The reduction

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

implies that there exist  $\Delta'_2$  and  $P'_2$  such that

$$\Gamma; \Delta_2 \vdash P_2 \implies \Delta'_2 \vdash P'_2 \quad \text{and} \quad \Gamma; \Delta_1 \vdash P'_1 \approx \Delta'_2 \vdash P'_2$$

The same arguments hold for the symmetric case, thus  $\approx$  is reduction-closed.

II. *Barb preservation* Following Definition 9, we have that  $\Gamma; \emptyset; \Delta_1 \vdash P_1 \downarrow_n$  implies

$$P \cong (v \tilde{m})(n!(V_1).P_3 \mid P_4)$$

with  $\bar{n} \notin \Delta_1$ . From the definition of  $\approx$  we infer that

$$\Gamma; \Delta_1 \vdash (v \tilde{m})(n!(V_1).P_3 \mid P_4) \xrightarrow{(v s_1)n!(V_1)} \Delta'_1 \vdash (v \tilde{m}')(P_3 \mid P_4)$$

implies the existence of  $\Delta'_2$ ,  $V_2$ , and  $P'_2$  such that

$$\Gamma; \Delta_2 \vdash P_2 \xrightarrow{(v m_2)n!(V_2)} \Delta'_2 \vdash P'_2$$

Therefore, we infer that  $\Gamma; \emptyset; \Delta_2 \vdash P_2 \downarrow_n$ , as desired.

III. *Congruence* We have to show that  $\approx$  is preserved under any context. The most interesting context case is parallel composition. Input congruence, which is the case that generates substitution, is straightforward, since we are dealing with closed terms.

To show the congruence of the parallel composition we construct a typed relation defined as

$$\mathcal{S} = \{(\Gamma; \emptyset; \Delta_1 \cdot \Delta_3 \vdash (v \tilde{n}_1)(P_1 \mid R) \triangleright \diamond, \Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash (v \tilde{n}_2)(P_2 \mid R) \triangleright \diamond) \mid \Gamma; \Delta_1 \vdash P_1 \approx \Delta_2 \vdash P_2, \forall \Gamma; \emptyset; \Delta_3 \vdash R \triangleright \diamond\}$$

We show that  $\mathcal{S}$  is a context bisimulation. Suppose that

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (v \tilde{n}_1)(P_1 \mid R) \xrightarrow{\ell} \Delta'_1 \cdot \Delta_3 \vdash P'$$

for some  $\Delta'_1$ . We must show an appropriate matching action from  $(v \tilde{n}_2)(P_2 \mid R)$ . We proceed by a case analysis on the “source” of action  $\ell$  (i.e.,  $P_1$ ,  $R$ , an interaction between  $P_1$  and  $R$ ). There are three cases:

1. Suppose that  $\ell$  originates in  $P_1$ :

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (v \tilde{n}_1)(P_1 \mid R) \xrightarrow{\ell} \Delta'_1 \cdot \Delta_3 \vdash (v \tilde{n}'_1)(P'_1 \mid R)$$

The case is divided into three sub-cases, depending on the shape of  $\ell$ :

i. Sub-case  $\ell \notin \{(v \tilde{m})n!(\lambda \tilde{x}. Q), (v \tilde{m} \tilde{m}_1)n!(\tilde{m}_1)\}$ : Then from the definition of typed transition we infer:

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta'_1 \vdash P'_1$$

which implies the existence of  $P'_2$  and  $\Delta'_2$  such that

$$\Gamma; \Delta_1 \vdash P_2 \xrightarrow{\ell} \Delta'_2 \vdash P'_2 \tag{54}$$

$$\Gamma; \Delta'_1 \vdash P'_1 \approx \Delta'_2 \vdash P'_2. \tag{55}$$

From transition (54) we may infer that

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

Furthermore, from (55) and the definition of  $\mathcal{S}$  we infer the desired conclusion:

$$\Gamma; \Delta'_1 \cdot \Delta_3 \vdash (\nu \tilde{n}'_1)(P'_1 \mid R) \mathcal{S} \Delta'_2 \cdot \Delta_3 \vdash (\nu \tilde{n}'_2)(P'_2 \mid R)$$

ii. Sub-case  $\ell = (\nu \tilde{m}_1)n!(\lambda \tilde{x}. Q_1)$ : Then we infer the typed transition

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu \tilde{m}_1)n!(\lambda \tilde{x}. Q_1)} \Delta'_1 \vdash P'_1$$

which implies the existence of  $P'_2, \Delta'_2, \Delta''_1,$  and  $\Delta''_2$  such that

$$\Gamma; \Delta_1 \vdash P_2 \xrightarrow{(\nu \tilde{m}_2)n!(\lambda \tilde{x}. Q_2)} \Delta'_2 \vdash P'_2 \tag{56}$$

and

$$\Gamma; \Delta'_1 \vdash (\nu \tilde{n}''_1)(P'_1 \mid Q\{\lambda \tilde{x}. Q_1/x\}) \approx \Delta''_2 \vdash (\nu \tilde{n}''_2)(P'_2 \mid Q\{\lambda \tilde{x}. Q_2/x\}) \tag{57}$$

for all  $Q$  with  $x \in \text{fv}(Q)$ . From transition (56), we infer that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu \tilde{n}_2)(P_2 \mid R) \xrightarrow{(\nu \tilde{m}_2)n!(\lambda \tilde{x}. Q_2)} \Delta'_2 \cdot \Delta_3 \vdash (\nu \tilde{n}'_2)(P'_2 \mid R)$$

Furthermore, from (57) we conclude that

$$\begin{aligned} &\Gamma; \Delta'_1 \cdot \Delta_3 \vdash (\nu \tilde{n}''_1)(P'_1 \mid Q\{\lambda \tilde{x}. Q_1/x\} \mid R) \mathcal{S} \\ &\vdash \Delta''_2 \cdot \Delta_3 \vdash (\nu \tilde{n}''_2)(P'_2 \mid Q\{\lambda \tilde{x}. Q_2/x\} \mid R) \end{aligned}$$

for all  $Q$ , with  $x \in \text{fv}(Q)$ , as desired.

iii. Sub-case  $\ell = (\nu \tilde{m}\tilde{m}_1)n!(\tilde{m}_1)$ : From the definition of typed transition we infer that

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu \tilde{m}\tilde{m}_1)n!(\tilde{m}_1)} \Delta'_1 \vdash P'_1$$

which, in turn, implies that there exist  $\Delta'_2, P'_2,$  and  $m_2$  such that

$$\Gamma; \Delta_1 \vdash P_2 \xrightarrow{(\nu \tilde{m}\tilde{m}_2)n!(\tilde{m}_2)} \Delta'_2 \vdash P'_2 \tag{58}$$

and

$$\Gamma; \Delta'_1 \vdash (\nu \tilde{n}_1)(P'_1 \mid Q\{\tilde{m}_1/\tilde{x}\}) \approx \Delta'_2 \vdash (\nu \tilde{n}_2)(P'_2 \mid Q\{\tilde{m}_2/\tilde{x}\}) \tag{59}$$

for some  $\Delta''_1$  and  $\Delta''_2$ , for all  $Q$  with  $\{x\} = \text{fv}(Q)$ . From transition (58) we infer that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu \tilde{n}'_2)(P_2 \mid R) \xrightarrow{(\nu \tilde{m}\tilde{m}_2)n!(\tilde{m}_2)} \Delta'_2 \cdot \Delta_3 \vdash (\nu \tilde{n}''_2)(P'_2 \mid R)$$

Furthermore, from (59) we conclude that

$$\Gamma; \Delta'_1 \cdot \Delta_3 \vdash (\nu \tilde{n}''_1)(P'_1 \mid Q\{\tilde{m}_1/\tilde{x}\} \mid R) \mathcal{S} \Delta'_2 \cdot \Delta_3 \vdash (\nu \tilde{n}''_2)(P'_2 \mid Q\{\tilde{m}_2/\tilde{x}\} \mid R)$$

for all  $Q$  with  $x \in \text{fv}(Q)$ , as desired.

2. Suppose that  $\ell$  originates in  $R$ :

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (v \widetilde{m}_1)(P_1 \mid R) \xrightarrow{\ell} \Delta_1 \cdot \Delta'_3 \vdash (v \widetilde{m}'_1)(P_1 \mid R')$$

This case is also divided into three sub-cases:

i. Sub-case  $\ell \notin \{(v \widetilde{m})n!(\lambda \widetilde{x}. Q), (v \widetilde{m}\widetilde{m}_1)n!(\widetilde{m}_1)\}$ : From the LTS we infer that

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell} \Delta'_3 \vdash R'$$

for some  $\Delta'_3$ , which in turn implies

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (v \widetilde{m}_2)(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta'_3 \vdash (v \widetilde{m}'_2)(P_2 \mid R')$$

Now, from the definition of  $\mathcal{S}$  we may obtain the desired conclusion:

$$\Gamma; \Delta_1 \cdot \Delta'_3 \vdash (v \widetilde{m}'_1)(P_1 \mid R') \mathcal{S} \Delta_2 \cdot \Delta'_3 \vdash (v \widetilde{m}'_2)(P_2 \mid R')$$

ii. Sub-case  $\ell = (v \widetilde{m}_1)n!(\lambda \widetilde{x}. Q)$ : From the LTS we infer that:

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell} \Delta'_3 \vdash R' \tag{60}$$

for some  $\Delta'_3$ . We then have that

$$\Gamma; \emptyset; \Delta''_3 \vdash (v \widetilde{m}')(R' \mid R_1\{\lambda \widetilde{x}. Q/x\}) \triangleright \diamond \tag{61}$$

for some  $\Delta''_3$  and for all  $R_1$  with  $\{x\} = \text{fv}(R_1)$ . Now, from (60) we obtain that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (v \widetilde{m}_2)(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta'_3 \vdash (v \widetilde{m}_2)(P_2 \mid R')$$

Then, from (61) and the definition of  $\mathcal{S}$  we obtain that

$$\begin{aligned} \Gamma; \emptyset; \Delta_1 \cdot \Delta'_3 \vdash (v \widetilde{m}_1)(P_1 \mid (v \widetilde{m}')(R' \mid R_1\{\lambda \widetilde{x}. Q/x\})) \\ \mathcal{S} \Delta_2 \cdot \Delta'_3 \vdash (v \widetilde{m}_2)(P_2 \mid (v \widetilde{m}')(R' \mid R_1\{\lambda \widetilde{x}. Q/x\})) \end{aligned}$$

for all  $R_1$  with  $x \in \text{fv}(R_1)$ , as desired.

iii. Sub-case  $\ell = (v \widetilde{m}\widetilde{m}_1)n!(\widetilde{m})$ : Similarly as above, from the typed LTS we infer that:

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell} \Delta'_3 \vdash R' \tag{62}$$

for some  $\Delta'_3$ . We then have that

$$\Gamma; \emptyset; \Delta''_3 \vdash (v \widetilde{m}')(R' \mid R_1\{\widetilde{m}/\widetilde{x}\}) \triangleright \diamond \tag{63}$$

for all  $R_1$  with  $\{\widetilde{x}\} = \text{fv}(R_1)$ , for some  $\Delta''_3$ . Now, from (62), we obtain that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (v \widetilde{m}_2)(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta'_3 \vdash (v \widetilde{m}_2)(P_2 \mid R')$$

Then, from (63) and the definition of  $\mathcal{S}$  we obtain the desired conclusion:

$$\begin{aligned} \Gamma; \emptyset; \Delta_1 \cdot \Delta'_3 \vdash (v \widetilde{m}_1)(P_1 \mid (v \widetilde{m}')(R' \mid R_1\{\widetilde{m}/\widetilde{x}\})) \\ \mathcal{S} \Delta_2 \cdot \Delta'_3 \vdash (v \widetilde{m}_2)(P_2 \mid (v \widetilde{m}')(R' \mid R_1\{\widetilde{m}/\widetilde{x}\})) \end{aligned}$$



3. We finally suppose that  $\ell$  originates from the interaction between  $P_1$  and  $R$ :

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (v \widetilde{m}_1)(P_1 \mid R) \xrightarrow{\tau} \Delta'_1 \cdot \Delta'_3 \vdash (v \widetilde{m}'_1)(P'_1 \mid R')$$

for some  $\Delta'_1, \Delta'_3$ . We then have that

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell_1} \Delta'_1 \vdash P'_1$$

and

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell_2} \Delta_3 \vdash R' \tag{64}$$

with  $\ell_1 \asymp \ell_2$  (cf. Definition 3). This case is divided into two sub-cases:

i.  $\ell_1 \notin \{(v \widetilde{m})n!(\lambda \widetilde{x}. Q), (v \widetilde{m} \widetilde{m}_1)n!(\widetilde{m}_1)\}$ : Then the transition from  $P_1$  implies

$$\Gamma; \Delta_2 \vdash P_2 \xrightarrow{\ell'_1} \Delta'_2 \vdash P'_2 \tag{65}$$

$$\Gamma; \Delta'_1 \vdash P'_1 \approx \Delta'_2 \vdash P'_2 \tag{66}$$

for some  $\Delta'_2$ . From (64) and (65) we obtain

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (v \widetilde{m}_2)(P_2 \mid R) \implies \Delta'_2 \cdot \Delta'_3 \vdash (v \widetilde{m}'_2)(P'_2 \mid R')$$

Then, from (66) and the definition of  $\mathcal{S}$  we obtain the desired conclusion:

$$\Gamma; \Delta'_1 \cdot \Delta'_3 \vdash (v \widetilde{m}'_1)(P'_1 \mid R') \mathcal{S} \Delta'_2 \cdot \Delta'_3 \vdash (v \widetilde{m}'_2)(P'_2 \mid R')$$

ii.  $\ell_1 = (v \widetilde{m}_1)n!(V_1)$ : Then we have the transition

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(v \widetilde{m}_1)n!(V_1)} \Delta'_1 \vdash P'_1$$

for some  $\Delta'_1$ , which implies

$$\Gamma; \Delta_3 \vdash R \xrightarrow{n?(V_1)} \Delta'_3 \vdash R'\{V_1/x\} \tag{67}$$

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (v \widetilde{m}_1)(P_1 \mid R) \xrightarrow{\tau} \Delta'_1 \cdot \Delta'_3 \vdash (v \widetilde{m}''_1)(P'_1 \mid R'\{V_1/x\}) \tag{68}$$

for some  $\Delta'_1$  and  $\Delta'_3$ . In turn, the output transition from  $P_1$  implies the existence of  $\Delta'_2, Q_2, P'_2$  such that

$$\Gamma; \Delta_2 \vdash P_2 \xrightarrow{(v \widetilde{m}_2)n!(V_2)} \Delta'_2 \vdash P'_2 \tag{69}$$

$$2\Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P'_1 \mid R'\{V_1/x\}) \approx \Delta'_2 \vdash (v \widetilde{m}'_2)(P'_2 \mid R'\{V_2/x\}) \tag{70}$$

for all  $R'$  with  $\{x\} = \text{fv}(Q)$ , and for some  $\Delta''_1$  and  $\Delta''_2$ . From (67) we obtain

$$\Gamma; \Delta_3 \vdash R \xrightarrow{n?(V_2)} \Delta'_3 \vdash R'\{V_2/x\}$$

for some  $\Delta''_3$ , which may be combined with (69) to obtain

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (v \widetilde{m}_2)(P_2 \mid R) \implies \Delta'_2 \cdot \Delta''_3 \vdash (v \widetilde{m}'_2)(P'_2 \mid R'\{V_2/x\})$$

From (70) and the definition of  $\mathcal{S}$  we can then get:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}'_1)(P'_1 \mid R'\{V_1/x\}) \mathcal{S} \Delta'_2 \cdot \Delta''_3 \vdash (v \widetilde{m}'_2)(P'_2 \mid R'\{V_2/x\}).$$

as required. □

In order to prove Lemma 7 (i.e.,  $\cong \subseteq \approx^H$ ), below we follow the technique developed in [8] and refined for session types in [18, 19].

**Definition 25** (Definability) Let  $\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond$ . A visible action  $\ell$  is *definable* whenever, given a fresh name *succ*, there exists a (testing) process  $\Gamma; \emptyset; \Delta_2 \vdash T(\ell, \text{succ}) \triangleright \diamond$  such that:

1. If  $\Gamma; \Delta_1 \vdash P \xrightarrow{\ell} \Delta'_1 \vdash P'$  then, for some  $\Delta'_2$ , either
  - (a)  $\ell \neq (v \tilde{m})n!(V)$  and  $P \mid T(\ell, \text{succ}) \longrightarrow P' \mid \text{succ}!(\bar{n}).\mathbf{0}$  and  $\Gamma; \emptyset; \Delta'_1 \cdot \Delta'_2 \vdash P' \mid \text{succ}!(\bar{n}).\mathbf{0} \triangleright \diamond$
  - (b)  $\ell = (v \tilde{m})n!(V)$  and  $P \mid T(\ell, \text{succ}) \longrightarrow (v \tilde{m})(P' \mid t \leftrightarrow_H V \mid \text{succ}!(\bar{n}, V).\mathbf{0})$  and  $\Gamma; \emptyset; \Delta'_1 \cdot \Delta'_2 \vdash (v \tilde{m})(P' \mid t \leftrightarrow_H V \mid \text{succ}!(\bar{n}, V).\mathbf{0}) \triangleright \diamond$ , for some fresh  $t$ .
2. If  $P \mid T(\ell, \text{succ}) \longrightarrow Q$  with  $\Gamma; \emptyset; \Delta \vdash Q \downarrow_{\text{succ}}$  then there exists a  $P'$  such that  $\Gamma; \Delta_1 \vdash P \xrightarrow{\ell} \Delta'_1 \vdash P'$  and one of the following holds:
  - (a)  $\ell \neq (v \tilde{m})n!(V)$  and  $Q \equiv P' \mid \text{succ}!(\bar{n}).\mathbf{0}$ .
  - (b)  $\ell = (v \tilde{m})n!(V)$  and  $Q \equiv (v \tilde{m})(P' \mid t \leftrightarrow_H V \mid \text{succ}!(\bar{n}, V).\mathbf{0})$ , for some fresh  $t$ .

We first show that every visible action  $\ell$  is definable.

**Lemma 20** (Definability) *Every visible action  $\ell$  is definable.*

*Proof* Let *succ* be a fresh name. We define:

$$T(\ell, \text{succ}) = \begin{cases} \bar{n}!(V).\text{succ}!(\bar{n}).\mathbf{0} & \text{if } \ell = n?(V) \\ \bar{n} \triangleleft l.\text{succ}!(\bar{n}).\mathbf{0} & \text{if } \ell = n \& l \\ \bar{n}?(y).(t \leftrightarrow_H y \mid \text{succ}!(\bar{n}, y).\mathbf{0}) & \text{if } \ell = (v \tilde{m})n!(V) \\ \bar{n} \triangleright \{l : \text{succ}!(\bar{n}).\mathbf{0}, l_i : (v a)(a?(y).\text{succ}!(\bar{n}).\mathbf{0})\}_{i \in I} & \text{if } \ell = n \oplus l \end{cases}$$

Consider the process

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

It is straightforward to do a case analysis on all actions  $\ell$  such that

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

to show that  $\ell$  is definable. □

We rely on the following auxiliary result:

**Lemma 21** (Extrusion) *Let  $P$  and  $Q$  be processes, and let *succ* be a fresh name. If*

$$\Gamma; \Delta'_1 \vdash (v \tilde{m}_1)(P \mid \text{succ}!(\bar{n}, V_1).\mathbf{0}) \cong \Delta_2 \vdash (v \tilde{m}_2)(Q \mid \text{succ}!(\bar{n}, V_2).\mathbf{0})$$

*with  $\{\tilde{m}_1\} = \text{fn}(V_1)$  and  $\{\tilde{m}_2\} = \text{fn}(V_2)$  then there exist  $\Delta_1$  and  $\Delta_2$  such that*

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

*Proof* Let  $\mathcal{S}$  be a relation defined as:

$$\begin{aligned} \mathcal{S} = \{ & (\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond, \Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond) \mid \\ & \Gamma; \Delta'_1 \vdash (v \tilde{m}_1)(P \mid \text{succ}!(\bar{n}, V_1).\mathbf{0}) \cong \Delta'_2 \vdash (v \tilde{m}_2)(Q \mid \text{succ}!(\bar{n}, V_2).\mathbf{0}), \\ & \wedge m_1 \in \text{fn}(V_1) \wedge m_2 \in \text{fn}(V_2) \} \end{aligned}$$

We show that  $\mathcal{S}$  is a reduction-closed, barbed congruence.

I. *Reduction-closed* The reduction  $P \longrightarrow P'$  implies

$$(\nu \widetilde{m}_1)(P \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0}) \longrightarrow (\nu \widetilde{m}_1)(P' \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0})$$

which, due to freshness of  $\text{succ}$ , in turn implies

$$(\nu \widetilde{m}_1)(Q \mid \text{succ}!\langle \bar{n}, V_2 \rangle.\mathbf{0}) \longrightarrow^* (\nu \widetilde{m}_1)(Q' \mid \text{succ}!\langle \bar{n}, V_2 \rangle.\mathbf{0})$$

Therefore,  $Q \longrightarrow^* Q'$ . Furthermore,

$$(\nu \widetilde{m}_1)(P' \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0}) \cong (\nu \widetilde{m}_1)(Q' \mid \text{succ}!\langle \bar{n}, V_2 \rangle.\mathbf{0})$$

that implies

$$\Gamma; \Delta'_1 \vdash P' \mathcal{S} \Delta'_2 \vdash Q'$$

as required.

II. *Barb preserving* Suppose  $\Gamma; \emptyset; \Delta_1 \vdash P \Downarrow_m$ . We analyse three cases, depending on the nature of  $m$ :

1. Case  $m \neq s$  ( $m$  is not a session name): Then from  $\Gamma; \emptyset; \Delta_1 \vdash P \Downarrow_m$  we infer

$$\Gamma; \emptyset; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0}) \Downarrow_m$$

for some  $\Delta'_1$ , which implies

$$\Gamma; \emptyset; \Delta'_2 \vdash (\nu \widetilde{m}_2)(Q \mid \text{succ}!\langle \bar{n}, V_2 \rangle.\mathbf{0}) \Downarrow_m.$$

for some  $\Delta'_2$ . Then, from the freshness of  $\text{succ}$ , we obtain  $\Gamma; \emptyset; \Delta_2 \vdash Q \Downarrow_m$ , as required.

2. Case:  $m = s$  ( $m$  is a session name) and  $m \neq n$ . The proof follows a similar reasoning as in the previous case.

3. Case:  $m = s$  ( $m$  is a session name) and  $m = n$  and  $\Gamma; \emptyset; \Delta_1 \vdash P \Downarrow_n$ . In this case, the fact that  $n$  is a session name implies that  $n, \bar{n} \in \text{dom}(\Delta'_1)$ . Therefore, from the definition of barbs (Definition 9) we can infer that

$$\Gamma; \emptyset; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0}) \not\Downarrow_n$$

because both endpoints of session  $n$  are present in  $\Delta'_1$ .

To observe the desired barb we exploit an additional test process, with an extra fresh name  $\text{succ}'$ . We compose  $\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond$  with  $\overline{\text{succ}}?(x, y).T\langle \ell, \text{succ}' \rangle$  where  $\text{subj}(\ell) = x$ . We then have

$$\Gamma; \emptyset; \Delta'_1 \vdash (\nu \widetilde{m}_1)(P \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0}) \mid \overline{\text{succ}}?(x, y).T\langle \ell, \text{succ}' \rangle \triangleright \diamond$$

The definition of definability and the fact that  $\Gamma; \emptyset; \Delta_1 \vdash P \Downarrow_n$  imply that

$$\begin{aligned} & (\nu \widetilde{m}_1)(P \mid \text{succ}!\langle \bar{n}, V_1 \rangle.\mathbf{0}) \mid \overline{\text{succ}}?(x, y).T\langle \ell, \text{succ}' \rangle \\ & \longrightarrow^* (\nu \widetilde{m}_1)(P' \mid \text{succ}'!\langle \bar{n}, V'_1 \rangle.\mathbf{0}) \end{aligned}$$

and furthermore

$$\begin{aligned} & (\nu \widetilde{m}_2)(Q \mid \text{succ}!\langle \bar{n}, V_2 \rangle.\mathbf{0}) \mid \overline{\text{succ}}?(x, y).T\langle \ell, \text{succ}' \rangle \\ & \longrightarrow^* (\nu \widetilde{m}_2)(Q' \mid \text{succ}'!\langle \bar{n}, V'_2 \rangle.\mathbf{0}) \end{aligned}$$

The last sequence of reductions implies that  $\Gamma; \emptyset; \Delta_2 \vdash Q \Downarrow_n$ , as required.

III. *Congruence* The key case is congruence with respect to parallel composition. The other cases are easier due to the fact that we are working with closed process terms (i.e. input congruence is straightforward on closed process terms). Let us define relation  $\mathcal{C}$  as

$$\begin{aligned} \mathcal{C} = \{ & (\Gamma; \emptyset; \Delta_1 \cdot \Delta_3 \vdash P \mid R \triangleright \diamond, \Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash Q \mid R \triangleright \diamond) \mid \\ & \forall R \text{ such that } \exists \Delta_3, \Gamma; \emptyset; \Delta_3 \vdash R \triangleright \diamond \wedge \\ & \Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid succ!(\bar{n}, V_1).\mathbf{0}) \cong \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid succ!(\bar{n}, V_2).\mathbf{0}) \} \end{aligned}$$

We want to show that  $\mathcal{C} \subseteq \mathcal{S}$ . To this end, we show that  $\mathcal{C}$  is a congruence with respect to parallel composition. We distinguish two cases:

- (i) Case  $(\bar{n} \cup \text{fn}(V_1) \cup \text{fn}(V_2)) \cap \text{fn}(R) = \emptyset$ : Then from the contextual definition of  $\cong$  we can deduce that for all  $\Gamma; \emptyset; \Delta_3 \vdash R \triangleright \diamond$ :

$$\begin{aligned} & \Gamma; \Delta'_1 \cdot \Delta_3 \vdash (v \widetilde{m}_1)(P \mid succ!(\bar{n}, V_1).\mathbf{0}) \mid R \cong \\ & \vdash \Delta'_2 \cdot \Delta_3 (v \widetilde{m}_2)(Q \mid succ!(\bar{n}, V_2).\mathbf{0}) \mid R \end{aligned}$$

Because of the requirement  $(\bar{n} \cup \text{fn}(V_1) \cup \text{fn}(V_2)) \cap \text{fn}(R) = \emptyset$  the above is structurally congruent to

$$\begin{aligned} & \Gamma; \Delta'_1 \cdot \Delta_3 \vdash (v \widetilde{m}_1)(P \mid succ!(\bar{n}, V_1).\mathbf{0} \mid R) \\ & \cong \Delta'_2 \cdot \Delta_3 \vdash (v \widetilde{m}_2)(Q \mid succ!(\bar{n}, V_2).\mathbf{0} \mid R) \end{aligned}$$

The desired conclusion is then immediate from the definition of  $\mathcal{C}$ .

- (ii) Case  $\widetilde{s} = \{\bar{n}, \widetilde{m}_1\} \cup \{\bar{n}, \widetilde{m}_2\} \cap \text{fn}(R)$ : Let  $R^{\widetilde{y}}$  be such that  $R = R^{\widetilde{y}}\{\widetilde{s}/\widetilde{y}\}$ . From the contextual definition of  $\cong$ , given a fresh name  $succ'$ , we can deduce that for all  $\Gamma; \emptyset; \Delta'_3 \vdash \overline{succ}^?(\widetilde{y}).(R^{\widetilde{y}} \mid succ'!(\widetilde{y}).\mathbf{0}) \triangleright \diamond$ :

$$\begin{aligned} & \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1)(P \mid succ!(\bar{n}, V_1).\mathbf{0}) \mid \overline{succ}^?(\widetilde{y}).(R^{\widetilde{y}} \mid succ'!(\widetilde{y}).\mathbf{0}) \\ & \cong \Delta'_2 \vdash (v \widetilde{m}_2)(Q \mid succ!(\bar{n}, V_2).\mathbf{0}) \mid \overline{succ}^?(\widetilde{y}).(R^{\widetilde{y}} \mid succ'!(\widetilde{y}).\mathbf{0}) \end{aligned}$$

for some  $\Delta'_1, \Delta'_2$ . Applying reduction closeness to the above pair we infer:

$$\Gamma; \Delta''_1 \vdash (v \widetilde{m}_1)(P \mid R \mid succ'!(\bar{n}, V_1).\mathbf{0}) \cong \Delta''_2 \vdash (v \widetilde{m}_2)(Q \mid R \mid succ'!(\bar{n}, V_2).\mathbf{0})$$

The conclusion then follows from the definition of  $\mathcal{C}$ . □

We can finally prove Lemma 7:

**Lemma 22**  $\cong \subseteq \approx^H$ .

*Proof* Let  $\mathfrak{R}$  be the typed relation (we omit the typing information in the definition):

$$\mathfrak{R} = \{(P_1, P_2) \mid \Gamma; \Delta_1 \vdash P_1 \cong \Delta_2 \vdash P_2\}$$

We prove that  $\mathfrak{R}$  is a higher-order bisimulation. Suppose that  $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta'_1 \vdash P'_1$ ; we must find a matching action from  $P_2$ . We distinguish two cases:

1. Suppose  $\ell = \tau$ . Then we have

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\tau} \Delta'_1 \vdash P'_1$$

The result follows the reduction closeness property of  $\cong$  since

$$\Gamma; \Delta_2 \vdash P_2 \xrightarrow{\tau} \Delta'_2 \vdash P'_2$$

for some  $\Delta'_2$ , and

$$\Gamma; \Delta'_1 \vdash P'_1 \cong \Delta'_2 \vdash P'_2 \text{ implies } \Gamma; \Delta'_1 \vdash P'_1 \Re \Delta'_2 \vdash P'_2.$$

2. Suppose  $\ell \neq \tau$ . Then we choose test  $T(\ell, succ)$  to obtain

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash P_1 \mid T(\ell, succ) \cong \Delta_2 \cdot \Delta_3 \vdash P_2 \mid T(\ell, succ) \tag{71}$$

for some  $\Delta_3$ . From this point on we distinguish two sub-cases:

i. Sub-case  $\ell \in \{n?(V_1), n \oplus l, n \& l\}$ : We then obtain

$$\begin{aligned} P_1 \mid T(\ell, succ) &\longrightarrow P'_1 \mid succ!(\bar{n}).\mathbf{0} \\ \Gamma; \emptyset; \Delta'_1 \cdot \Delta'_3 \vdash P'_1 \mid succ!(\bar{n}).\mathbf{0} &\downarrow_{succ} \end{aligned}$$

for some  $\Delta'_3$ . From (71) we may now infer:

$$\Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash P_2 \mid T(\ell, succ) \downarrow_{succ}$$

which, using Lemma 20, implies

$$\begin{aligned} \Gamma \Delta_2 P_2 &\xrightarrow{\ell} \Delta'_2 P'_2 \\ P_2 \mid T(\ell, succ) &\longrightarrow^* P'_2 \mid succ!(\bar{n}).\mathbf{0} \end{aligned}$$

and

$$\Gamma; \Delta'_1 \cdot \Delta'_3 \vdash P'_1 \mid succ!(\bar{n}).\mathbf{0} \cong \Delta'_2 \cdot \Delta'_3 \vdash P'_2 \mid succ!(\bar{n}).\mathbf{0}$$

We then apply Lemma 21 to obtain the required result:

$$\Gamma; \Delta'_1 \vdash P'_1 \cong \Delta'_2 \vdash P'_2 \text{ implies } \Gamma; \Delta'_1 \vdash P'_1 \Re \Delta'_2 \vdash P'_2.$$

ii. Sub-case  $\ell = (v \widetilde{m}_1)n!(V_1)$ : Note that  $T((v \widetilde{m}_1)n!(V_1), succ) = T((v \widetilde{m}_2)n!(V_2), succ)$ . The transition from  $P_1$  can be then written as

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(v \widetilde{m}_1)n!(V_1)} \Delta'_1 \vdash P'_1 \tag{72}$$

for some  $\Delta'_1$ . If we use the test process  $T((v \widetilde{m}_1)n!(V_1), succ)$ , then we may obtain:

$$\begin{aligned} P_1 \mid T((v \widetilde{m}_1)n!(V_1), succ) &\longrightarrow (v \widetilde{m}_1)(P'_1 \mid t \leftrightarrow_{\text{H}} V_1) \mid succ!(\bar{n}, V_1).\mathbf{0} \\ \Gamma; \emptyset; \Delta'_1 \cdot \Delta'_3 \vdash (v \widetilde{m}_1)(P'_1 \mid t \leftrightarrow_{\text{H}} V_1) &\mid succ!(\bar{n}, V_1).\mathbf{0} \downarrow_{succ} \end{aligned}$$

for some  $\Delta'_3$ . Using (71) we may then infer

$$\Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash P_2 \mid T((v \widetilde{m}_2)n!(V_2), succ) \downarrow_{succ}$$

which, using Lemma 20, implies

$$\begin{aligned} \Gamma; \Delta_2 \vdash P_2 &\xrightarrow{(v \widetilde{m}_2)n!(V_2)} \Delta'_2 \vdash P'_2 \\ P_2 \mid T(\ell, succ) &\longrightarrow^* (v \widetilde{m}_2)(P'_2 \mid t \leftrightarrow_{\text{H}} V_2) \mid succ!(\bar{n}, V_2).\mathbf{0} \end{aligned} \tag{73}$$

for some  $\Delta'_2$ , and

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \cdot \Delta'_3 \vdash (v \widetilde{m}_1)(P'_1 \mid t \leftrightarrow_{\text{H}} \lambda \widetilde{x}. Q_1) &\mid succ!(\bar{n}, V_1).\mathbf{0} \\ \cong \Delta'_2 \cdot \Delta'_3 \vdash (v \widetilde{m}_2)(P'_2 \mid t \leftrightarrow_{\text{H}} \lambda \widetilde{x}. Q_2) &\mid succ!(\bar{n}, V_2).\mathbf{0} \end{aligned}$$

We then apply Lemma 21 to obtain:

$$\Gamma; \Delta'_1 \vdash (v \widetilde{m}_1)(P'_1 \mid t \leftrightarrow_{\text{H}} V_1) \cong \Delta'_2 \vdash (v \widetilde{m}_2)(P'_2 \mid t \leftrightarrow_{\text{H}} V_2)$$

From the above result and the definition of  $\mathfrak{R}$  we finally obtain:

$$\begin{aligned} \Gamma; \emptyset; \Delta'_1 \vdash (v \widetilde{m}_1)(P'_1 \mid t \leftrightarrow_H V_1) \\ \mathfrak{R} \Delta'_2 \vdash (v \widetilde{m}_2)(P'_2 \mid t \leftrightarrow_H V_2) \end{aligned}$$

as required.  $\square$

## References

- Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. *Inf. Comput.* **163**(2), 409–470 (2000)
- Berger, M., Honda, K., Yoshida, N.: Sequentiality and the  $\pi$ -calculus. In: *Proceedings of TLCA'01*. Volume 2044 of LNCS, pp. 29–45. Springer, Berlin (2001)
- Berger, M., Honda, K., Yoshida, N.: Genericity and the pi-calculus. *Acta Inf.* **42**(2–3), 83–141 (2005)
- Bernardi, G., Dardha, O., Gay, S.J., Kouzapas, D.: On duality relations for session types. In: Maffei, M., Tuosto, E. (eds.) *Trustworthy Global Computing—9th International Symposium, TGC 2014, Rome, Italy, September 5–6, 2014. Revised Selected Papers*, Volume 8902 of *Lecture Notes in Computer Science*, pp. 51–66. Springer, Berlin (2014)
- Gay, S.J., Vasconcelos, V.T.: Linear type theory for asynchronous session types. *J. Funct. Program.* **20**(1), 19–50 (2010)
- Girard, J.-Y., Lafont, Y., Taylor, P.: *Proofs and Types*, Volume 7 of *Cambridge Tracts in Theoretical Computer Science*. CUP, Cambridge (1989)
- Groote, J.F., Sellink, M.P.A.: Confluence for process verification. *Theor. Comput. Sci.* **170**(1–2), 47–81 (1996)
- Hennessy, M.: *A Distributed Pi-Calculus*. CUP, Cambridge (2007)
- Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type disciplines for structured communication-based programming. In: Hankin, C. (ed.) *Programming Languages and Systems—ESOP'98, 7th European Symposium on Programming, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28–April 4, 1998, Proceedings*, *Lecture Notes in Computer Science*, vol. 1381, pp. 122–138. Springer, Berlin (1998)
- Honda, K., Yoshida, N.: On reduction-based process semantics. *Theor. Comput. Sci.* **151**(2), 437–486 (1995)
- Honda, K., Yoshida, N.: A uniform type structure for secure information flow. *ACM Trans. Program. Lang. Syst.* **29**(6) (2007)
- Hyland, J.M.E., Ong, C.L.: On full abstraction for PCF: I, II, and III. *Inf. Comput.* **163**(2), 285–408 (2000)
- Jeffrey, A., Rathke, J.: Contextual equivalence for higher-order pi-calculus revisited. *Log. Methods Comput. Sci.* **1**(1) 2005
- Kobayashi, N., Pierce, B.C., Turner, D.N.: Linearity and the pi-calculus. *ACM Trans. Program. Lang. Syst.* **21**(5), 914–947 (1999)
- Koutavas, V., Hennessy, M.: First-order reasoning for higher-order concurrency. *Comput. Lang. Syst. Struct.* **38**(3), 242–277 (2012)
- Kouzapas, D., Pérez, J.A., Yoshida, N.: Characteristic bisimulation for higher-order session processes. In: Aceto, L., de Frutos Escrig, D. (eds.) *26th International Conference on Concurrency Theory (CONCUR 2015). Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 42, pp. 398–411. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2015)
- Kouzapas, D., Pérez, J.A., Yoshida, N.: On the relative expressiveness of higher-order session processes. In: Thiemann, P. (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*, *Lecture Notes in Computer Science*, vol. 9632, pp. 446–475. Springer, Berlin (2016)
- Kouzapas, D., Yoshida, N.: Globally governed session semantics. *Log. Methods Comput. Sci.* **10**(4) (2014)
- Kouzapas, D., Yoshida, N., Hu, R., Honda, K.: On asynchronous eventful session semantics. *Math. Struct. Comput. Sci.* **26**(2), 303–364 (2016)
- Lanese, I., Pérez, J.A., Sangiorgi, D., Schmitt, A.: On the expressiveness and decidability of higher-order process calculi. *Inf. Comput.* **209**(2), 198–226 (2011)
- Lenglet, S., Schmitt, A.: Howe's method for contextual semantics. In: Aceto, L., de Frutos Escrig, D. (eds.) *26th International Conference on Concurrency Theory (CONCUR 2015). Leibniz International Proceed-*

- ings in Informatics (LIPIcs), vol. 42, pp. 212–225. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2015)
22. Milner, R.: Fully abstract models of typed lambda-calculi. *Theor. Comput. Sci.* **4**(1), 1–22 (1977)
  23. Milner, R.: Functions as processes. *Math. Struct. Comput. Sci.* **2**(2), 119–141 (1992)
  24. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: Kuich, W. (ed.) 19th ICALP, Volume 623 of LNCS, pp. 685–695. Springer, Berlin (1992)
  25. Mostrous, D., Yoshida, N.: Two session typing systems for higher-order mobile processes. In: Rocca, S.R.D. (ed.) *Typed Lambda Calculi and Applications*, 8th International Conference, TLCA 2007, Paris, France, June 26–28, 2007, Proceedings. *Lecture Notes in Computer Science*, vol. 4583, pp. 321–335. Springer, Berlin (2007)
  26. Mostrous, D., Yoshida, N.: Session typing and asynchronous subtyping for the higher-order  $\pi$ -calculus. *Inf. Comput.* **241**, 227–263 (2015)
  27. Pérez, J.A., Caires, L., Pfenning, F., Toninho, B.: Linear logical relations and observational equivalences for session-based concurrency. *Inf. Comput.* **239**, 254–302 (2014)
  28. Pierce, B., Sangiorgi, D.: Typing and subtyping for mobile processes. *MSCS* **6**(5), 409–454 (1996)
  29. Plotkin, G.: LCF considered as a programming language. *Theor. Comput. Sci.* **5**(3), 223–255 (1977)
  30. Sangiorgi, D.: Expressing mobility in process algebras: first-order and higher order paradigms. Ph.D. Thesis, University of Edinburgh (1992)
  31. Sangiorgi, D.: Bisimulation for higher-order process calculi. *Inf. Comput.* **131**(2), 141–178 (1996)
  32. Sangiorgi, D., Kobayashi, N., Sumii, E.: Environmental bisimulations for higher-order languages. In: 22nd IEEE Symposium on Logic in Computer Science (LICS 2007). 10–12 July 2007, Wroclaw, Poland, Proceedings, pp. 293–302. IEEE Computer Society, Washington (2007)
  33. Thomsen, B.: Plain CHOCS: a second generation calculus for higher order processes. *Acta Inf.* **30**(1), 1–59 (1993)
  34. Xu, X.: On context bisimulation for parameterized higher-order processes. In: Proceedings of ICE 2013, Volume 131 of EPTCS, pp. 37–51 (2013)
  35. Yoshida, N.: Graph types for monadic mobile processes. In: FSTTCS. Volume 1180 of LNCS, pp. 371–386. Springer, Berlin (1996)
  36. Yoshida, N., Berger, M., Honda, K.: Strong normalisation in the pi-calculus. *Inf. Comput.* **191**(2), 145–202 (2004)
  37. Yoshida, N., Honda, K., Berger, M.: Linearity and bisimulation. In: Nielsen, M., Engberg, U. (eds.) *Foundations of Software Science and Computation Structures*, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8–12, 2002, Proceedings, Volume 2303 of Lecture Notes in Computer Science, pp. 417–434. Springer, Berlin (2002)