Acta Informatica

Characteristic Bisimulation for Higher-Order Session Processes

--Manuscript Draft--

Manuscript Number:	ACIN-D-15-00127R1
Full Title:	Characteristic Bisimulation for Higher-Order Session Processes
Article Type:	S.I. : CONCUR'2015
Funding Information:	Engineering and Physical Sciences Research Council (GB) (EP/K011715/1, EP/K034413/1,EP/L00058X/1) — Not applicable European Cooperation in Science and Technology (IC1201) — Not applicable
Abstract:	Characterising contextual equivalence is a long-standing issue for higher- order (process) languages. 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.
Corresponding Author:	Jorge A. Perez University of Groningen NETHERLANDS
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Groningen
Corresponding Author's Secondary Institution:	
First Author:	Dimitrios Kouzapas
First Author Secondary Information:	
Order of Authors:	Dimitrios Kouzapas
	Jorge A. Perez
	Nobuko Yoshida
Order of Authors Secondary Information:	
Author Comments:	
Response to Reviewers:	See attached letter (at the end of the PDF) for our detailed answers to the reviewers's comments.

Noname manuscript No.
(will be inserted by the editor)

Characteristic Bisimulation for Higher-Order Session Processes

Dimitrios Kouzapas[1] · **Jorge A. Pérez**[2] · **Nobuko Yoshida**[3]

the date of receipt and acceptance should be inserted later

Abstract Characterising contextual equivalence is a long-standing issue for higher-order (process) languages. In the setting of a higher-order π-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., [33,30,13,20,15,26]. In this work, we consider $\mathsf{HO}\pi$, a higher-order process calculus with *session communication*: it combines functional constructs (abstractions/applications, as in the call-by-value λ-calculus) and concurrent primitives (synchronisation on shared names, communication on linear names, recursion). By amalgamating functional and concurrent constructs, $\mathsf{HO}\pi$ may specify complex session protocols that include 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. Distinguishing between linear and shared channels/names is important, for session-based concurrency can be seen as involving 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.

1 University of Glasgow · 2 University of Groningen · 3 Imperial College London

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.

A well-known behavioural equivalence for higher-order processes is *context bisimilarity* [31]. This characterisation of barbed congruence 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., [30,31,13,15,34,21]. 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 in [13]; their solution, however, does not consider *linearity* which, as explained above, is an important aspect in session-based concurrency.

This Work. Building upon [30,31,13], 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. Formally, we enforce this limitation 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, § 3 presents the session calculus $\mathsf{HO}\pi$. § 4 gives the session type system for $\mathsf{HO}\pi$ and states type soundness. § 5 develops *characteristic* bisimilarity and states our main result: characteristic bisimilarity and contextual equivalence coincide for well-typed $\mathsf{HO}\pi$ processes (Theorem 2). § 6 discusses related works, while § 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 A and B. 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 § 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 § 3.

Preliminaries. The syntax of $\mathsf{HO}\pi$ considered in this paper is the following:

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

We write n to range over shared names $a, b, \dots$ and session (linear) names $s, \overline{s}, \dots$. 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 $\overline{s}$ to denote the dual endpoint of s. The higher-order character of $\mathsf{HO}\pi$ thus comes from the fact that V can be an abstraction. The semantics of $\mathsf{HO}\pi$ can be given in terms of a labelled transition system (LTS), denoted $P \xrightarrow{\ell} P'$, where ℓ denotes a transition label or the internal action τ. This way, e.g., $P \xrightarrow{n?\langle V\rangle} P'$ denotes an input transition along n and $P \xrightarrow{(\nu\,\widetilde{m})n!\langle V\rangle} P'$ denotes an output transition along n, sending value V, and extruding names $\widetilde{m}$. Weak transitions, written $P \overset{\ell}{\Longrightarrow} Q'$, abstract from internal actions in the usual way. Throughout the paper, we write $\Re, \Re', \dots$ to denote binary relations on (typed) processes.

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

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

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 ($\approx$, Definition 12) is an overly demanding relation on higher-order processes. There are 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:

($\star$) Whenever $P \xrightarrow{(\nu\, \widetilde{m_1})n!\langle V\rangle} P'$ there exist Q', W such that $Q \overset{(\nu\, \widetilde{m_2})n!\langle W\rangle}{\Longrightarrow} Q'$ and, ***for all*** R with $\mathtt{fv}(R) = \{x\}$, $(\nu\, \widetilde{m_1})(P' \mid R\{^{V}\!/x\}) \,\Re\, (\nu\, \widetilde{m_2})(Q' \mid R\{^{W}\!/x\})$.

Intuitively, in the above clause process R stands for any possible *context* to which the emitted value (V and W) is supposed to go. 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* ($\approx^{\mathtt{C}}$) 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

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

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

$$\begin{aligned} P &\longrightarrow (\nu\, s)(s?(x).R \mid \overline{s}!\langle V\rangle.\mathbf{0}) \\ &\longrightarrow R\{^{V}\!/x\} \mid \mathbf{0} \end{aligned}$$

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

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

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

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

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

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, a simple process term that inhabits that type (Definition 13). To illustrate the key idea underlying characteristic processes, consider $S = ?(S_1 \to \diamond); !\langle S_2 \rangle; \mathtt{end}$, i.e., the session type abstracting a protocol that first inputs an abstraction (a function from values S_1 to processes), and then outputs a value of type S_2. Let P be the process $u?(x).(u!\langle s_2 \rangle.\mathbf{0} \mid x\, s_1)$, 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, a given value type U (i.e., a type for channels or abstractions), then $(\!|U|\!)_{\mathsf{c}}$ denotes its *characteristic value* (cf. Definition 13). As we explain next, we use $(\!|U|\!)_{\mathsf{c}}$ 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 4, page 19). 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 $\overset{\ell}{\longmapsto}$. We use this refined LTS to define *characteristic bisimulation* ($\approx^{\mathtt{C}}$, 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 $(\!|U|\!)_{\mathsf{c}}$:

$$P \xrightarrow{n?\langle V \rangle} P' \ \wedge\ (V = m \vee V \equiv \lambda x.\, t?(y).(y\, x) \vee V \equiv (\!|U|\!)_{\mathsf{c}}) \quad \Rightarrow \quad P \overset{n?\langle V \rangle}{\longmapsto} P'$$

Note the different notation for standard and refined transitions: $\xrightarrow{n?\langle V \rangle}$ vs. $\overset{n?\langle V \rangle}{\longmapsto}$.

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:

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

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

$$T_V \overset{t?\langle (\!|?(U);\mathtt{end}|\!)_{\mathsf{c}} \rangle}{\longmapsto} \approx t?(x).(\nu\, s)(s?(y).(\!|U|\!)^y \mid \overline{s}!\langle V \rangle.\mathbf{0})$$

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

It is also noteworthy that $\mathsf{HO}\pi$ lacks name matching, which is crucial in [13] to prove completeness of bisimilarity. The lack of matching is compensated here with the use of (session) types. Matching gives the observer the ability to test the

$$n, m \;::=\; a, b \;\mid\; s, \overline{s} \qquad u, w \;::=\; n \;\mid\; x, y, z \qquad V, W \;::=\; u \;\mid\; \lambda x.\, P$$

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

(a) Syntax.

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

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

$$P \mid (\nu\, n)Q \equiv (\nu\, n)(P \mid Q)\ (n \notin \mathtt{fn}(P)) \qquad P \equiv Q \text{ if } P \equiv_\alpha Q$$

(b) Structural Congruence.

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

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

$$[\text{Par}] \quad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad [\text{Cong}] \quad \frac{P \equiv Q \longrightarrow Q' \equiv P'}{P \longrightarrow P'}$$

(c) Reduction Semantics.

Fig. 1: $\mathsf{HO}\pi$: Syntax and Semantics (Structural Congruence and Reduction).

equality of received names. In contrast, in our theory a process trigger embeds a name into a characteristic process 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* ($\mathsf{HO}\pi$) 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 $\mathsf{HO}\pi$ is defined in Figure 1a. We use $a, b, c, \ldots$ to range over shared names and $s, \overline{s}, \ldots$ to range over session names. We use $n, m, t, \ldots$ 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 $\overline{n}$ with $\overline{\overline{s}} = s$ and $\overline{a} = a$. This way, e.g., session names s and $\overline{s}$ are two dual endpoints. Name variables are denoted with $x, y, z, \ldots$, and recursive variables are denoted with $X, Y \ldots$. Values V, W include name identifiers $u, v, \ldots$ (first-order values) and abstractions $\lambda x.\, P$ (higher-order values), where P is a process P and x is a name parameter.

Terms include π-calculus usual constructs for sending and receiving values V. Process $u!\langle V\rangle.P$ denotes the output of V over name u, with continuation P; process $u?(x).P$ denotes the input prefix on name u of a value that will substitute variable x in continuation P. Recursion is expressed by $\mu X.P$, which binds the recursive variable X in process P. Process $V\,W$ is the application which substitutes values W on the abstraction V. Typing ensures that V is not a name. Processes $u \triangleright \{l_i : P_i\}_{i\in I}$ and $u \triangleleft l.P$ 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 $\mathbf{0}$, parallel composition $P_1 \mid P_2$, and name restriction $(\nu\, n)P$ are standard.

Session name restriction $(\nu\, s)P$ simultaneously binds endpoints s and $\overline{s}$ in P. We use $\mathtt{fv}(P)$ and $\mathtt{fn}(P)$ 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 $\mathtt{fv}(P) = \emptyset$, we call P *closed*.

3.2 Semantics

Figure 1c defines the operational semantics of $\mathsf{HO}\pi$, given as a reduction relation that relies on a *structural congruence* relation, denoted $\equiv$ (Figure 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 Figure 1 follow. Rule [App] defines value application; Rule [Pass] defines a shared interaction at n (with $\overline{n} = n$) or a session interaction; Rule [Sel] is the standard rule for labelled choice/selection: given an index set I, a process selects label l_j on name n over a set of labels $\{l_i\}_{i\in I}$ offered by a branching on the dual endpoint $\overline{n}$; and other rules are standard. We write $\longrightarrow^*$ for a multi-step reduction.

3.3 An Example: The Hotel Booking Scenario

To illustrate $\mathsf{HO}\pi$ 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 in not optimal, so the client sends mobile processes to both hotels to automatically negotiate and book a room.

We now present two $\mathsf{HO}\pi$ implementations of this scenario. For convenience, we write $\mathtt{if}\ e\ \mathtt{then}\ (P_1\ \mathtt{;}\ P_2)$ 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:

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

Process 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 Client_1. Intuitively, process P_{xy}:

(i) sends the room requirements to the hotel;
(ii) receives a quote from the hotel;
(iii) sends the quote to Client_1;
(iv) expects a choice from Client_1 whether to accept or reject the offer;
(v) if the choice is accept then it informs the hotel and performs the booking; otherwise, if the choice is reject then it informs the hotel and ends the session.

Client_1 instantiates two copies of P_{xy} as abstractions on session x. It uses 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_1 uses the dual endpoints $\overline{h_1}$ and $\overline{h_2}$ to receive the negotiation result from the two remote instances of P and then inform the two processes for the final booking decision.

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):

$$\begin{array}{rcl}\mathsf{Client}_2 & \stackrel{\mathtt{def}}{=} & (\nu\, h)(s_1!\langle \lambda x.\, Q_1\{^{h}\!/y\}\rangle.s_2!\langle \lambda x.\, Q_2\{^{\overline{h}}\!/y\}\rangle.\mathbf{0}) \\ Q_1 & \stackrel{\mathtt{def}}{=} & x!\langle \mathsf{room}\rangle.x?(\mathsf{quote}_1).y!\langle \mathsf{quote}_1\rangle.y?(\mathsf{quote}_2).R_x \\ Q_2 & \stackrel{\mathtt{def}}{=} & x!\langle \mathsf{room}\rangle.x?(\mathsf{quote}_1).y?(\mathsf{quote}_2).y!\langle \mathsf{quote}_1\rangle.R_x \\ R_x & \stackrel{\mathtt{def}}{=} & \mathtt{if}\ \ \mathsf{quote}_1 \leq \mathsf{quote}_2\ \mathtt{then}\ (x \triangleleft \mathsf{accept}.x!\langle \mathsf{credit}\rangle.\mathbf{0}\ ;\ x \triangleleft \mathsf{reject}.\mathbf{0})\end{array}$$

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

The different protocols implemented by Client_1 and Client_2 can be represented by the sequence diagrams of Figure 2. We will assign session types to these processes in Example 1. Later on, in § 5.9 we will show that Client_1 and Client_2 are behaviourally equivalent using characteristic bisimilarity; see Proposition 3 (page 27).

4 Types and Typing

We define a session typing system for $\mathsf{HO}\pi$ and state its main properties. Our system distils the key features of [25,26].

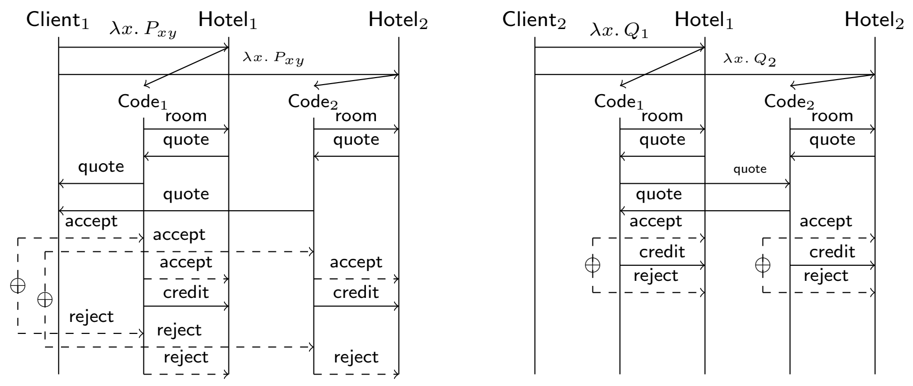

Fig. 2: Sequence diagrams for Client_1 and Client_2, as in § 3.3.

4.1 Types

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

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

Value type U includes 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. As for session types, the *output type* $!\langle U\rangle;S$ first sends a value of type U and then follows the type described by S. Dually, $?(U);S$ 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. We assume the *recursive type* $\mu\mathtt{t}.S$ is guarded, i.e., $\mu\mathtt{t}.\mathtt{t}$ is not allowed. Type $\mathtt{end}$ is the termination type.

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 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 $F : \mathcal{P}(\mathsf{ST} \times \mathsf{ST}) \to \mathcal{P}(\mathsf{ST} \times \mathsf{ST})$ defined by:

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

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

4.2 Typing Environments and Judgements

Typing *environments* are defined below:

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

Typing environments Γ, Λ, and Δ 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.

Γ maps variables and shared names to value types, and recursive variables to session environments; it admits weakening, contraction, and exchange principles. Given Γ, we write $\Gamma\backslash x$ to denote the environment obtained from Γ by removing the assignment $x : U{\to}\diamond$, for some U. While Λ maps variables to linear higher-order types, Δ maps session names to session types. Both Λ and Δ are only subject to exchange. The domains of Γ, Λ and Δ are assumed pairwise distinct. $\Delta_1 \cdot \Delta_2$ means the disjoint union of Δ_1 and Δ_2. We define *typing judgements* for values V and processes P:

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

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

Definition 2 (Session Environments: Balanced/Reduction) Let Δ be a session environment.

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

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

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

$$[\textsc{Prom}]\ \frac{\Gamma; \emptyset; \emptyset \vdash V \triangleright U \multimap \diamond}{\Gamma; \emptyset; \emptyset \vdash V \triangleright U \rightarrow \diamond} \qquad [\textsc{EProm}]\ \frac{\Gamma; \Lambda \cdot x : U \multimap \diamond; \Delta \vdash P \triangleright \diamond}{\Gamma \cdot x : U \rightarrow \diamond; \Lambda; \Delta \vdash P \triangleright \diamond} \qquad [\textsc{Abs}]\ \frac{\Gamma; \Lambda; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \emptyset; \Delta_2 \vdash x \triangleright U}{\Gamma \backslash x; \Lambda; \Delta_1 \backslash \Delta_2 \vdash \lambda x.\, P \triangleright U \multimap \diamond}$$

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

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

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

$$[\textsc{Req}]\ \frac{\Gamma; \emptyset; \emptyset \vdash u \triangleright \langle U \rangle \qquad \Gamma; \Lambda; \Delta_1 \vdash P \triangleright \diamond \qquad \Gamma; \emptyset; \Delta_2 \vdash V \triangleright U}{\Gamma; \Lambda; \Delta_1 \cdot \Delta_2 \vdash u!\langle V \rangle.P \triangleright \diamond}$$

$$[\textsc{Acc}]\ \frac{\Gamma; \emptyset; \emptyset \vdash u \triangleright \langle U \rangle \qquad \Gamma; \Lambda_1; \Delta_1 \vdash P \triangleright \diamond \qquad \Gamma; \Lambda_2; \Delta_2 \vdash x \triangleright U}{\Gamma \backslash x; \Lambda_1 \backslash \Lambda_2; \Delta_1 \backslash \Delta_2 \vdash u?(x).P \triangleright \diamond}$$

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

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

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

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

Fig. 3: Typing Rules for $\mathsf{HO}\pi$.

We rely on a typing system that is similar to the one developed in [25,26]. The typing system is defined in Figure 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, e.g., for 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 a send value V should appear as a prefix on the session type $!\langle U\rangle;S$ of u. Rule [Rcv] is its dual. We use a similar approach with session prefixes to type interaction between shared names as defined in Rules [Req] and [Acc], where the type of the sent/received object (S and L, respectively) should match the type of the sent/received subject ($\langle S\rangle$ and $\langle L\rangle$, respectively). Rules [Sel] and [Bra] for selection and branching are standard: 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 Γ as defined in Rule [Res]. Creation of a session name s creates and restricts two endpoints with dual types in Rule [ResS]. Rule [Par], combines the environments Λ and Δ of the parallel components of a parallel process. The disjointness of environments Λ and Δ is implied. Rule [End] adds a name with type `end` in Δ. The recursion requires that the body process matches the type of the recursive variable as in Rule [Rec]. The recursive variable is typed directly from the shared environment Γ as in Rule [RVar]. The inactive process $\mathbf{0}$ is typed with no linear environments as in Rule [Nil].

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

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

Proof Following standard lines. See Appendix A for details. □

Example 1 (Hotel Booking Revisited) We give types to the client processes of § 3.3. Assume

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

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 Client_1 is $\emptyset;\emptyset;s_1 :!\langle U{\multimap}\diamond\rangle;\mathsf{end}\cdot s_2 :!\langle U{\multimap}\diamond\rangle;\mathsf{end} \vdash \mathsf{Client}_1 \triangleright \diamond$.

The typings for Q_1 and Q_2 are $\emptyset;\emptyset;y :!\langle \mathsf{quote}\rangle;?(\mathsf{quote});\mathsf{end} \vdash \lambda x.\,Q_i \triangleright U{\multimap}\diamond$ $(i = 1,2)$ and the type for Client_2 is $\emptyset;\emptyset;s_1 :!\langle U{\multimap}\diamond\rangle;\mathsf{end}\cdot s_2 :!\langle U{\multimap}\diamond\rangle;\mathsf{end} \vdash \mathsf{Client}_2 \triangleright \diamond$.

5 Characteristic Bisimulation

We develop a theory for observational equivalence over session typed $\mathsf{HO}\pi$ processes that follows the principles laid in our previous works [19, 18]. We introduce *higher-order bisimulation* (Definition 17) and *characteristic bisimulation* (Definition 18), denoted $\approx^{\mathtt{H}}$ and $\approx^{\mathtt{C}}$, respectively. We prove that they coincide with reduction-closed, barbed congruence (Theorem 2, page 26).

We briefly summarise our strategy for obtaining Theorem 2. We begin by defining an (early) labelled transition system (LTS) on untyped processes (§ 5.1). Then, using the *environmental* transition semantics (§ 5.2), we define a typed LTS

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

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

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

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

Fig. 4: The Untyped LTS for $\mathsf{HO}\pi$ processes. We omit Rule $\langle\textsc{Par}_R\rangle$.

that formalises how a typed process interacts with a typed observer. Later, we define reduction-closed, barbed congruence and context bisimilarity, respectively (§ 5.3 and § 5.4). Subsequently, we define the refined LTS based on characteristic values (§ 5.5). Building upon this LTS, we define higher-order and characteristic bisimilarities (§ 5.6). Then, we develop an auxiliary proof technique based on deterministic transitions (§ 5.7). Our main result, the characterisation of barbed congruence in terms of $\approx^{\mathtt{H}}$ and $\approx^{\mathtt{C}}$, is stated in § 5.8. Finally, we revisit our two implementations for the Hotel Booking Scenario (§ 3.3), using Theorem 2 to show that they are behaviourally equivalent (§ 5.9).

5.1 Labelled Transition System for Processes

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

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

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

Dual actions occur on subjects that are dual between them and carry the same object; thus, output is dual to input and selection is dual to branching.

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

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

The (early) labelled transition system (LTS) over *untyped processes* is given in Figure 4. We write $P_1 \xrightarrow{\ell} P_2$ with the usual meaning. The rules are standard [19, 18]; we comment on some of them. A process with an output prefix can interact with the environment with an output action that carries a value V (Rule $\langle \text{Snd} \rangle$). Dually, in Rule $\langle \text{Rv} \rangle$ a receiver process can observe an input of an arbitrary value V. Select and branch processes observe the select and branch actions in Rules $\langle \text{Sel} \rangle$ and $\langle \text{Bra} \rangle$, resp. Rule $\langle \text{Res} \rangle$ 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 α-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. Such a relation, given in Figure 5, is defined on triples of environments by extending the LTSs in [19,18]; it is denoted

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

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

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

Output Actions are defined by Rules [SSnd] and [ShSnd]. Rule [SSnd] states the conditions for observing action $(\nu\, \widetilde{m})s!\langle V \rangle$ on a type tuple $(\Gamma, \Lambda, \Delta \cdot s : S)$. The session environment $\Delta \cdot s : S$ should include the session environment of the sent value V (denoted Δ' in the rule), *excluding* the session environments of names m_j in $\widetilde{m}$ which restrict the scope of value V (denoted Δ_j in the rule). Analogously, the linear variable environment Λ' of V should be included in Λ. The rule defines the scope extrusion of session names in $\widetilde{m}$; consequently, environments associated to their dual endpoints (denoted Δ'_j in the rule) appear in the resulting session environment. Similarly for shared names in $\widetilde{m}$ that are extruded. All free values used for typing V (denoted Λ' and Δ' in the rule) are subtracted from the resulting type tuple. The prefix of session s is consumed by the action. Rule [ShSnd] follows very 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]. We illustrate Rule [SSnd] by means of an example:

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

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

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

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

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

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

Fig. 5: Labelled Transition System for Typed Environments.

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

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

Then by Rule [SSnd], we can derive:

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

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

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

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

The typed LTS combines the LTSs in Figure 4 and Figure 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 $\Longrightarrow$ for the reflexive and transitive closure of $\rightarrow$, $\overset{\ell}{\Longrightarrow}$ for the transitions $\Longrightarrow\xrightarrow{\ell}\Longrightarrow$, and $\overset{\hat{\ell}}{\Longrightarrow}$ for $\overset{\ell}{\Longrightarrow}$ if $\ell \neq \tau$ otherwise $\Longrightarrow$.

A typed transition relation requires type judgements with an empty Λ. Notice that we are working with closed terms. Furthermore, we can always apply Rule [EProm] and obtain an empty Λ.

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 Δ. Recall that Δ 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 Δ_1 and Δ_2 are *confluent*, denoted $\Delta_1 \rightleftharpoons \Delta_2$, if there exists Δ 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).

Typed relations relate only closed terms 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. Δ_1 and Δ_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 $P \equiv (\nu\, \tilde{m})(n!\langle V \rangle.P_2 \mid P_3)$ or $P \equiv (\nu\, \tilde{m})(n \triangleleft l.P_2 \mid P_3)$, with $n \notin \tilde{m}$.
   (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 $\overline{n} \notin \Delta$.

(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 $\overline{n}$ is not present in the session environment Δ.

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 $n!\langle m \rangle.\mathit{succ}!\langle\rangle.\mathbf{0}$ (with fresh 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:

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

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

The first behavioural relation 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 Δ_1', Q', Δ_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 Δ_1'', Δ_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 $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{m_1}) n!\langle V_1 \rangle} \Delta_1' \vdash P_2$, there exist Q_2, V_2, Δ_2' such that $\Gamma; \Delta_2 \vdash Q_1 \overset{(\nu\, \widetilde{m_2}) n!\langle V_2 \rangle}{\Longrightarrow} \Delta_2' \vdash Q_2$ and for all R with $\mathtt{fv}(R) = \{x\}$:
$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_2 \mid R\{V_1/x\}) \;\Re\; \Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q_2 \mid R\{V_2/x\});$$
2) For all $\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_1' \vdash P_2$ such that ℓ is not an output, there exist Q_2, Δ_2' such that $\Gamma; \Delta_2 \vdash Q_1 \overset{\hat{\ell}}{\Longrightarrow} \Delta_2' \vdash Q_2$ and $\Gamma; \Delta_1' \vdash P_2 \;\Re\; \Delta_2' \vdash Q_2$; and
3) The symmetric cases of 1 and 2.

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

Fig. 6: Characteristic Processes (left) and Characteristic Values (right).

The largest such bisimulation is called *context bisimilarity* and is denoted by $\approx$.

As suggested in § 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 higher-order communication.

5.5 Characteristic Values and the Refined LTS

We formalise the ideas given in § 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 $\llbracket U \rrbracket^u$, and the *characteristic value* of U, denoted $\llbracket U \rrbracket_{\mathsf{c}}$, are defined in Figure 6.

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

Proof (Sketch) The proof is done by induction on the syntax of types. See Proposition 4 (page 36) in the Appendix for details. □

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

Example 3 Consider the type $S = \mu\mathtt{t}.!\langle U_1\rangle;?(U_2);\mathtt{t}$. By Definition 13, we have that $\llbracket S \rrbracket^s = \llbracket !\langle U_1\rangle;?(U_2);\mathtt{end} \rrbracket^s = s!\langle \llbracket U_1 \rrbracket_{\mathsf{c}}\rangle.t!\langle s\rangle.\mathbf{0}$. For this process, we can infer the following type derivations:

$$
\frac{\Gamma;\emptyset;\Delta \triangleright \llbracket U_1 \rrbracket_{\mathsf{c}} \triangleright U_2 \Gamma;\emptyset;t :!\langle ?(U_2);\mathtt{end}\rangle;\mathtt{end} \cdot s :?(U_2);\mathtt{end} \vdash t!\langle s\rangle.\mathbf{0} \triangleright \diamond}{\Gamma;\emptyset;\Delta \cdot t :!\langle ?(U_2);\mathtt{end}\rangle;\mathtt{end} \cdot s :!\langle U_1\rangle;?(U_2);\mathtt{end} \vdash s!\langle \llbracket U_1 \rrbracket_{\mathsf{c}}\rangle.t!\langle s\rangle.\mathbf{0} \triangleright \diamond}
$$

and

$$
\frac{\Gamma;\emptyset;\Delta \cdot t :!\langle ?(U_2);\mu\mathtt{t}.!\langle U_1\rangle;?(U_2);\mathtt{t}\rangle;\mathtt{end} \cdot s :?(U_2);\mu\mathtt{t}.!\langle U\rangle;\mathtt{t} \vdash t!\langle s\rangle.\mathbf{0} \triangleright \diamond}{\Gamma;\emptyset;\Delta \cdot t :!\langle ?(U_2);\mu\mathtt{t}.!\langle U\rangle;\mathtt{t}\rangle;\mathtt{end} \cdot s : \mu\mathtt{t}.!\langle U\rangle;\mathtt{t} \vdash s!\langle \llbracket U_1 \rrbracket_{\mathsf{c}}\rangle.t!\langle s\rangle.\mathbf{0} \triangleright \diamond}
$$

The following example motivates the refined LTS explained in § 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 4 (The Need for the Refined Typed LTS) We illustrate the complementary rôle that characteristic values (cf. Figure 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

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

such that

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

with $\Delta = s_1{:}\mathtt{end} \cdot s_2{:}\mathtt{end}$. If P_1 and P_2 input along s a characteristic value of the form $\llbracket(\mathtt{end}){\rightarrow}\diamond\rrbracket_{\mathsf{c}} = \lambda z.\,\mathbf{0}$ (cf. Figure 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 $\lambda x.\,(\nu\, s')(a!\langle s'\rangle.\mathbf{0})$ with $\Gamma; \emptyset; \emptyset \vdash a \triangleright \langle \mathtt{end}\rangle$, then their derivatives are not bisimilar:

$$\begin{aligned}
\Gamma; \emptyset; \Delta \vdash P_1 &\xrightarrow{s?\langle \lambda x.\,(\nu\, s')(a!\langle s'\rangle.\mathbf{0})\rangle}\longrightarrow\longrightarrow \\
&\Delta \vdash (\nu\, s')(a!\langle s'\rangle.\mathbf{0}) \mid (\nu\, s')(a!\langle s'\rangle.\mathbf{0}) \\[1em]
\Gamma; \emptyset; \Delta \vdash P_2 &\xrightarrow{s?\langle \lambda x.\,(\nu\, s')(a!\langle s'\rangle.\mathbf{0})\rangle}\longrightarrow \\
&\Delta \vdash (\nu\, s')(a!\langle s'\rangle.\mathbf{0}) \mid (\lambda z.\,\mathbf{0})\,s_2
\end{aligned}$$

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):

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

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:

$$(\nu\, s)(n?(x).(x\,s) \mid \overline{s}!\langle \lambda x.\, R_1\rangle.\mathbf{0}) \tag{4}$$
$$(\nu\, s)(n?(x).(x\,s) \mid \overline{s}!\langle \lambda x.\, R_2\rangle.\mathbf{0}) \tag{5}$$

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

$$(\nu\, s)(t?(x).(x\, s) \mid \overline{s}!\langle \lambda x.\, R_1\rangle.\mathbf{0})$$
$$(\nu\, s)(t?(x).(x\, s) \mid \overline{s}!\langle \lambda x.\, R_2\rangle.\mathbf{0})$$

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 $\lambda z.\, z?(x).(t!\langle z\rangle.\mathbf{0} \mid x\, m)$, where m is a fresh name, then they would become, under appropriate Γ and Δ:

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

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. $\square$

As explained in § 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

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

is defined by on top of the rules in Figure 5 using the following rules:

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

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

Then, the refined typed labelled transition system

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

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

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

Notation 16 *Below we sometimes write* $\overset{(\nu\, \tilde{m})n!\langle V:U\rangle}{\longmapsto}$ *when the type of* V *is* U*.*

5.6 Higher-Order Bisimilarity ($\approx^{\mathtt{H}}$) and Characteristic Bisimilarity ($\approx^{\mathtt{C}}$)

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

$$t \hookleftarrow_{\mathtt{H}} V \overset{\mathtt{def}}{=} \begin{cases} t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle V\rangle.\mathbf{0}) & \text{if } V \text{ is a first-order value} \\ t?(x).(\nu\, s)(s?(y).(y\, x) \mid \overline{s}!\langle V\rangle.\mathbf{0}) & \text{if } V \text{ is a higher-order value} \end{cases} \tag{6}$$

$$t \Leftarrow_{\mathtt{C}} V : U \overset{\mathtt{def}}{=} t?(x).(\nu\, s)(s?(y).[\![U]\!]^{y} \mid \overline{s}!\langle V\rangle.\mathbf{0}) \tag{7}$$

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) 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 $\Re$ is a *higher-order bisimulation* if for all $\Gamma; \Delta_1 \vdash P_1 \;\Re\; \Delta_2 \vdash Q_1$

1) Whenever $\Gamma; \Delta_1 \vdash P_1 \overset{(\nu\, \widetilde{m_1})n!\langle V_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$, there exist Q_2, V_2, Δ_2' such that $\Gamma; \Delta_2 \vdash Q_1 \overset{(\nu\, \widetilde{m_2})n!\langle V_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$ and, for fresh t,
$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_2 \mid t \hookleftarrow_{\mathtt{H}} V_1) \;\Re\; \Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q_2 \mid t \hookleftarrow_{\mathtt{H}} V_2)$$
2) For all $\Gamma; \Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$ such that ℓ is not an output, there exist Q_2, Δ_2' such that $\Gamma; \Delta_2 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$ and $\Gamma; \Delta_1' \vdash P_2 \;\Re\; \Delta_2' \vdash Q_2$; and
3) The symmetric cases of 1 and 2.

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

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 $\Gamma; \Delta_1 \vdash P_1 \overset{(\nu\, \widetilde{m_1})n!\langle V_1:U_1\rangle}{\longmapsto} \Delta_1' \vdash P_2$ then there exist Q_2, V_2, Δ_2' such that $\Gamma; \Delta_2 \vdash Q_1 \overset{(\nu\, \widetilde{m_2})n!\langle V_2:U_2\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$ and, for fresh t,
$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_2 \mid t \Leftarrow_{\mathtt{C}} V_1 : U_1) \;\Re\; \Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q_2 \mid t \Leftarrow_{\mathtt{C}} V_2 : U_2)$$
2) For all $\Gamma; \Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2$ such that ℓ is not an output, there exist Q_2, Δ_2' such that $\Gamma; \Delta_2 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$ and $\Gamma; \Delta_1' \vdash P_2 \;\Re\; \Delta_2' \vdash Q_2$; and
3) The symmetric cases of 1 and 2.

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

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

Remark 1 (Differences between $\approx^{\mathtt{H}}$ and $\approx^{\mathtt{C}}$) Although $\approx^{\mathtt{H}}$ and $\approx^{\mathtt{C}}$ are conceptually similar, they differ in the kind of trigger process considered. Because of the application in $t \hookleftarrow_{\mathtt{H}} V$ (cf. (6)), $\approx^{\mathtt{H}}$ cannot be used to reason about first-order session processes (i.e., processes without higher-order features). In contrast, $\approx^{\mathtt{C}}$ 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 β-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 Δ. Transition $\Gamma; \Delta \vdash P \xmapsto{\tau} \Delta' \vdash P'$ is called:

- a *session-transition* whenever transition $P \xrightarrow{\tau} P'$ is derived using Rule $\langle \text{Tau} \rangle$ (where $\mathtt{subj}(\ell_1)$ and $\mathtt{subj}(\ell_2)$ in the premise are dual endpoints), possibly followed by uses of Rules $\langle \text{Alpha} \rangle$, $\langle \text{Res} \rangle$, $\langle \text{Rec} \rangle$, or $\langle \text{Par}_L \rangle / \langle \text{Par}_R \rangle$ (cf. Figure 4).
- a *β-transition* whenever transition $P \xrightarrow{\tau} P'$ is derived using Rule $\langle \text{App} \rangle$, possibly followed by uses of Rules $\langle \text{Alpha} \rangle$, $\langle \text{Res} \rangle$, $\langle \text{Rec} \rangle$, or $\langle \text{Par}_L \rangle / \langle \text{Par}_R \rangle$ (cf. Figure 4).

Notation 20 *We use the following notations:*

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

Deterministic transitions imply the τ-inertness property [7], which ensures behavioural invariance on deterministic transitions.

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

1. $\Gamma; \Delta \vdash P \xmapsto{\tau_{\mathsf{d}}} \Delta' \vdash P'$ *implies* $\Gamma; \Delta \vdash P \approx^{\mathtt{H}} \Delta' \vdash P'$.
2. $\Gamma; \Delta \vdash P \overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta' \vdash P'$ *implies* $\Gamma; \Delta \vdash P \approx^{\mathtt{H}} \Delta' \vdash P'$.

Proof (Sketch) The proof for Part 1 requires a case analysis on the structure of β-transition and session-transition. The proof for Part 2 is direct from Part 1. See Appendix B.2 (page 38) for the details of the proof. □

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

Lemma 1 (Up-to Deterministic Transition) *Let $\Gamma; \Delta_1 \vdash P_1 \ \Re\ \Delta_2 \vdash Q_1$ such that if whenever:*

1. $\forall (\nu\, \widetilde{m_1}) n!\langle V_1 \rangle$ *such that* $\Gamma; \Delta_1 \vdash P_1 \xmapsto{(\nu\, \widetilde{m_1}) n!\langle V_1 \rangle} \Delta_3 \vdash P_3$ *implies that* $\exists Q_2, V_2$ *such that* $\Gamma; \Delta_2 \vdash Q_1 \overset{(\nu\, \widetilde{m_2}) n!\langle V_2 \rangle}{\Longmapsto} \Delta_2' \vdash Q_2$ *and* $\Gamma; \Delta_3 \vdash P_3 \overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta_1' \vdash P_2$ *and for fresh t:*
$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_2 \mid t \hookleftarrow_{\mathtt{H}} V_1) \ \Re\ \Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q_2 \mid t \hookleftarrow_{\mathtt{H}} V_2)$$
2. $\forall \ell \neq (\nu\, \widetilde{m}) n!\langle V \rangle$ *such that* $\Gamma; \Delta_1 \vdash P_1 \xmapsto{\ell} \Delta_3 \vdash P_3$ *implies that* $\exists Q_2$ *such that* $\Gamma; \Delta_1 \vdash Q_1 \overset{\hat{\ell}}{\Longmapsto} \Delta_2' \vdash Q_2$ *and* $\Gamma; \Delta_3 \vdash P_3 \overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta_1' \vdash P_2$ *and* $\Gamma; \Delta_1' \vdash P_2 \ \Re\ \Delta_2' \vdash Q_2$.
3. *The symmetric cases of 1 and 2.*

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

Proof (Sketch) The proof proceeds by considering the relation

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

We may verify that $\Re^{\overset{\tau_{\mathsf{d}}}{\Longmapsto}}$ is a bisimulation with the use of Proposition 2. □

5.8 Characterisation of Higher-order and Characteristic Bisimilarities

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

Lemma 2 $\approx^{\mathtt{C}} = \approx^{\mathtt{H}}$.

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

$$(\nu\, \tilde{m}_1)(P_1 \mid t \Leftarrow_{\mathtt{C}} V : U) \approx^{\mathtt{H}} (\nu\, \tilde{m}_2)(P_2 \mid t \Leftarrow_{\mathtt{C}} V_2 : U)$$

given that

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

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

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

given that

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

Intuitively, the above equalities follow from the fact that characteristic trigger processes $t \Leftarrow_{\mathtt{C}} V : U$ and higher-order trigger processes $t \hookleftarrow_{\mathtt{H}} V$ exhibit similar behaviour, see Lemma 13 in the Appendix for more details. Indeed, while for the former we have

$$t?(x).(\nu\, s)(s?(y).\llparenthesis U \rrparenthesis^{y} \mid \overline{s}!\langle V\rangle.\mathbf{0}) \overset{t?\langle s'\rangle}{\longmapsto} (\nu\, s)(s?(y).\llparenthesis U \rrparenthesis^{y} \mid \overline{s}!\langle V\rangle.\mathbf{0})$$

for the latter we have:

$$t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle V\rangle.\mathbf{0}) \overset{t?\langle \llparenthesis U \rrparenthesis_{\mathtt{c}}\rangle}{\longmapsto} (\nu\, s)(s?(y).\llparenthesis U \rrparenthesis^{y} \mid \overline{s}!\langle V\rangle.\mathbf{0})$$

Using the above information we can show that typed relations induced by $\approx^{\mathtt{H}}$ and $\approx^{\mathtt{C}}$ coincide. The full proof is found in in Appendix B.3, Lemma 14 (page 46). □

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*

$$\Gamma; \Delta_1' \vdash P\{\lambda x.\, t?(y).(y\, x)/z\} \approx^{\mathtt{H}} \Delta_2' \vdash Q\{\lambda x.\, t?(y).(y\, x)/z\}$$

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

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

The full proof of Lemma 3 can be found in Appendix B.3, Lemma 17 (page 54); 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 $I = \{1, \ldots, n\}$, we shall write $\prod_{i \in I} P_i$ to stand for $P_1 \mid P_2 \mid \cdots \mid P_n$.

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

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, m_i)$$

then for all $\lambda\widetilde{x}.\, R$, there exist Δ_1', Δ_2' such that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda\widetilde{x}.\, R)\, \widetilde{n}) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda\widetilde{x}.\, R)\, \widetilde{m}).$$

Proof (Sketch) The proof follows the definition of the characteristic process; see Lemma 16, page 51, in the Appendix for details. Let us consider a particular case; we construct a typed relation $\Re$:

$$\begin{aligned}\Re = \{&\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda x.\, R)\, n_1)\ ,\ \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda x.\, R)\, n_2)\ \mid \\ &\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda x.\, t?(y).(y\, x))\, n_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda x.\, t?(y).(y\, x))\, n_2)\}\end{aligned}$$

$\Re$ 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 $\Re$ is a higher-order bisimulation. Assume that

$$\Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid R\{n_1/x\}) \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{\ell_1}{\longmapsto} \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P' \mid R'\{n_1/x\})$$

for some Δ_1''. Then from the definition of $\Re$ we have:

$$\Gamma; \emptyset; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda x.\, t?(y).(y\, x))\, n_1) \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{t?\langle (\!| U |\!)_{\mathsf{c}}\rangle}{\longmapsto}\overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1}'')(P \mid (\!| U |\!)^{n_1})$$

for some Δ_3. Characteristic processes have the following property, for any $U \neq \mathtt{end}$:

$$(\!| U |\!)^{n} \xrightarrow{\ell} t!\langle n\rangle.\mathbf{0}$$

By the last property we can always observe, for some Δ_1' (note that below ℓ_1 may be an action τ, thus denoting the interaction of P and $(\!| U |\!)^{n_1}$):

$$\Gamma; \emptyset; \Delta_3 \vdash (\nu\, \widetilde{m_1}'')(P \mid (\!| U |\!)^{n_1}) \overset{\ell_1}{\longmapsto} (\nu\, \widetilde{m_1}''')(P' \mid t'!\langle n_1\rangle.\mathbf{0}) \overset{t'!\langle n_1\rangle}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}''')P'$$

which implies, from the requirements of higher-order bisimulation, that there exist $(\nu\, \widetilde{m_2}'')(Q' \mid (\!| U |\!)^{x}\{n_2/x\})$ and Δ_4 such that

$$\Gamma; \emptyset; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda x.\, t?(y).(y\, x))\, n_2) \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{t?\langle (\!| U |\!)_{\mathsf{c}}\rangle}{\Longmapsto}\overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta_4 \vdash (\nu\, \widetilde{m_2}'')(Q' \mid (\!| U |\!)^{n_2})$$

By the shape of the characteristic process we can always observe for $\ell_2, \mathtt{subj}(\ell_2) = \mathtt{subj}(\ell_1)$ if ℓ_1 is output, and $\ell_2 = \ell_1$ otherwise, that:

$$\begin{aligned}\Gamma; \emptyset; \Delta_4 \vdash (\nu\, \widetilde{m_2}'')(Q' \mid (\!| U |\!)^{x}\{n_2/x\}) &\overset{\ell_2}{\Longmapsto} (\nu\, \widetilde{m_2}''')(Q'' \mid t'!\langle n_2\rangle.\mathbf{0}) \\ &\overset{t'!\langle n_2\rangle}{\longmapsto} \Delta_4' \vdash (\nu\, \widetilde{m_2}''')Q''\end{aligned} \qquad (8)$$

for some Δ_4' and

$$\Gamma; \Delta_3'' \vdash (\nu\, \widetilde{m_1}''')(P' \mid t'' \hookleftarrow_{\mathtt{H}} n_1) \approx^{\mathtt{H}} \Delta_4'' \vdash (\nu\, \widetilde{m_2}''')(Q'' \mid t'' \hookleftarrow_{\mathtt{H}} n_2) \qquad (9)$$

for some Δ_4''. From (8) we get

$$\Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid R\{^{n_2}/_x\}) \overset{\ell_2}{\Longrightarrow} \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q'' \mid R'\{^{n_2}/_x\})$$

for some Δ_2'' and from (9) we get

$$\Gamma; \Delta_3'' \vdash (\nu\, \widetilde{m_1}''')(P' \mid (\lambda x.\, t''?(y).(y\, x))\, n_1) \approx^{\mathtt{H}} \Delta_4'' \vdash (\nu\, \widetilde{m_2}''')(Q'' \mid (\lambda x.\, t''?(y).(y\, x))\, n_2)$$

which implies from the definition of $\Re$ that for R' we get

$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P' \mid R'\{^{n_1}/_x\})\ \Re\ \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q'' \mid R'\{^{n_2}/_x\})$$

as required. $\square$

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

Lemma 5 $\approx^{\mathtt{H}} \subseteq \approx$.

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 $\approx^{\mathtt{H}}$ (cf. Definition 17):

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P_2 \mid t \hookleftarrow_{\mathtt{H}} V_1)\ \Re\ \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q_2 \mid t \hookleftarrow_{\mathtt{H}} V_2)$$

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

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P_2 \mid R\{^{V_1}/_x\})\ \Re\ \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q_2 \mid R\{^{V_2}/_x\}).$$

The full proof is found in Appendix B.3, Lemma 18 (page 56). $\square$

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:

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

We show that $\mathcal{S}$ is a context bisimulation by a case analysis on the transitions of the pairs in $\mathcal{S}$. The full proof is found in Appendix B.3, Lemma 19 (page 58). $\square$

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

Lemma 7 $\cong\ \subseteq\ \approx^{\mathtt{H}}$.

Proof (Sketch) The proof exploits the *definability* technique developed in [8, §6.7] and refined for session types in [19,18]. 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 ℓ, 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:

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

See Definition 25 (page 62) 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 Δ_3, Δ_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 $T\langle \ell, succ\rangle$:

$$\Gamma; \Delta_3' \vdash P' \mid succ!\langle \overline{m}\rangle.\mathbf{0} \;\cong\; \Delta_4' \vdash Q' \mid succ!\langle \overline{m}\rangle.\mathbf{0} \tag{10}$$

which in turn means that whenever $\Gamma; \Delta_1 \vdash P \triangleright \diamond$ can perform an action $\overset{\ell}{\longmapsto}$ then we can derive that $\Gamma; \Delta_2 \vdash Q \triangleright \diamond$ can also perform action $\overset{\ell}{\Longmapsto}$ 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 $\approx$:

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

The full details can be found in Appendix B.3, Lemma 22 (page 64). $\square$

We can finally state our main result:

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

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

$$\cong \subseteq \approx^{\mathtt{H}} = \approx^{\mathtt{C}} \subseteq \approx \subseteq \cong$$

$\square$

5.9 Revisiting the Hotel Booking Scenario (Section 3.3)

Now we prove that Client_1 and Client_2 in § 3.3 are behaviourally equivalent.

Proposition 3 *Let* $S = !\langle \mathsf{room}\rangle; ?(\mathsf{quote}); \oplus\{\mathsf{accept} : !\langle \mathsf{credit}\rangle; \mathtt{end}, \mathsf{reject} : \mathtt{end}\}$ *and* $\Delta = s_1 : !\langle S \multimap \diamond\rangle; \mathtt{end} \cdot s_2 : !\langle S \multimap \diamond\rangle; \mathtt{end}$. *Then* $\emptyset; \Delta \vdash \mathsf{Client}_1 \approx^{\mathsf{C}} \Delta \vdash \mathsf{Client}_2$, *where* Client_1 *and* Client_2 *are as in* § *3.3.*

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

$$\llparenthesis ?(S \multimap \diamond); \mathtt{end} \rrparenthesis^{s} = s?(x).(t!\langle s\rangle.\mathbf{0} \mid \llparenthesis S \multimap \diamond \rrparenthesis^{x})$$

For convenience, we recall the definition of Client_1:

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

where

$$\begin{aligned} P_{xy} &\overset{\mathsf{def}}{=} x!\langle \mathsf{room}\rangle.x?(\mathsf{quote}).y!\langle \mathsf{quote}\rangle.y \triangleright \left\{ \begin{array}{l} \mathsf{accept} : x \triangleleft \mathsf{accept}.x!\langle \mathsf{credit}\rangle.\mathbf{0}\ , \\ \mathsf{reject} : x \triangleleft \mathsf{reject}.\mathbf{0} \end{array} \right\} \\ R' &\equiv \mathtt{if}\ \ x \leq y\ \mathtt{then}\ (\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0}\ \ ;\ \ \overline{h_1} \triangleleft \mathsf{reject}.\overline{h_2} \triangleleft \mathsf{accept}.\mathbf{0}) \end{aligned}$$

Also, the definition of Client_2 is as follows:

$$\begin{aligned} \mathsf{Client}_2 &\overset{\mathsf{def}}{=} (\nu\, h)(s_1!\langle \lambda x.\, Q_1\{^{h}\!/y\}\rangle.s_2!\langle \lambda x.\, Q_2\{^{\overline{h}}\!/y\}\rangle.\mathbf{0}) \\ Q_1 &\overset{\mathsf{def}}{=} x!\langle \mathsf{room}\rangle.x?(\mathsf{quote}_1).y!\langle \mathsf{quote}_1\rangle.y?(\mathsf{quote}_2).R_x \\ Q_2 &\overset{\mathsf{def}}{=} x!\langle \mathsf{room}\rangle.x?(\mathsf{quote}_1).y?(\mathsf{quote}_2).y!\langle \mathsf{quote}_1\rangle.R_x \\ R_x &\overset{\mathsf{def}}{=} \mathtt{if}\ \ \mathsf{quote}_1 \leq \mathsf{quote}_2\ \mathtt{then}\ (x \triangleleft \mathsf{accept}.x!\langle \mathsf{credit}\rangle.\mathbf{0}\ ;\ x \triangleleft \mathsf{reject}.\mathbf{0}) \end{aligned}$$

A detailed account of the observable behaviour of Client_1 is given in Figure 7, where we use the following shorthand notation:

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

Similarly, Figure 8 illustrates the actions possible from Client_2, which are the same as for Client_1. ☐

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 λ-calculus by Milner—this was not possible in the untyped polyadic π-calculus [23]. After [28], many works on typed π-calculi have investigated correctness of encodings of known concurrent and sequential calculi in order to examine semantic effects of proposed typing systems.

$$
\begin{array}{ll}
 & \emptyset; \emptyset; \Delta \vdash \mathsf{Client}_1 \\[4pt]
\xmapsto{s_1!\langle \lambda x.\, P_{xy}\{h_1/y\}\rangle} & \emptyset; \emptyset; s_2 :!\langle S{-\!\circ}\diamond\rangle;\mathsf{end} \cdot k_1 : S \vdash \\
 & (\nu\, h_1, h_2)(s_2!\langle \lambda x.\, P_{xy}\{h_2/y\}\rangle.\mathbf{0} \mid \overline{h_1}?(x).\overline{h_2}?(y).R' \\
 & \mid t_1 \Leftarrow_{\mathtt{C}} P_{xy}\{h_1/y\} : S{-\!\circ}\diamond) \\[4pt]
\xmapsto{s_2!\langle \lambda x.\, P_{xy}\{h_2/y\}\rangle} & \emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu\, h_1, h_2)(\overline{h_1}?(x).\overline{h_2}?(y).R' \\
 & t_1 \Leftarrow_{\mathtt{C}} P_{xy}\{h_1/y\} : S{-\!\circ}\diamond \mid t_2 \Leftarrow_{\mathtt{C}} P_{xy}\{h_2/y\} : S{-\!\circ}\diamond) \\[4pt]
\xmapsto{t_1?\langle b\rangle}\xmapsto{t_2?\langle b\rangle}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu\, h_1, h_2)(\overline{h_1}?(x).\overline{h_2}?(y).R' \\
 & \mid (\nu\, s_1', s_2')(P_{xy}\{h_1/y\}\{k_1/x\} \mid P_{xy}\{h_1/y\}\{k_2/x\}) \\
 & \mid t_3!\langle s_1'\rangle.\mathbf{0} \mid t_4!\langle s_2'\rangle.\mathbf{0}) \\[4pt]
\xmapsto{t_3!\langle s_1'\rangle}\xmapsto{t_4!\langle s_2'\rangle} & \emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu\, h_1, h_2)(\overline{h_1}?(x).\overline{h_2}?(y).R' \\
 & \mid P_{xy}\{h_1/y\}\{k_1/x\} \mid P_{xy}\{h_1/y\}\{k_2/x\}) \\
 & \mid (\nu\, s_1', s_2')(t_3' \Leftarrow_{\mathtt{C}} s_1' : \mathsf{end} \mid t_4 \Leftarrow_{\mathtt{C}} s_2' : \mathsf{end}) \\[4pt]
\xmapsto{t_3'!\langle c\rangle}\xmapsto{t_4'!\langle c\rangle} & \\
\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset; \emptyset; k_1 : S \cdot k_2 : S \vdash (\nu\, h_1, h_2)(\overline{h_1}?(x).\overline{h_2}?(y).R' \\
 & \mid P_{xy}\{h_1/y\}\{k_1/x\} \mid P_{xy}\{h_1/y\}\{k_2/x\}) \\[4pt]
\xmapsto{k_1!\langle \mathsf{room}\rangle}\xmapsto{k_2!\langle \mathsf{room}\rangle} & \\
\xmapsto{k_1?\langle \mathsf{quote}\rangle}\xmapsto{k_2?\langle \mathsf{quote}\rangle} & \emptyset; \emptyset; k_1 : S' \cdot k_2 : S' \vdash (\nu\, h_1, h_2)(\overline{h_1}?(x).\overline{h_2}?(y).R' \\
 & \mid h_1!\langle \mathsf{quote}\rangle.Q\{h_1/z\} \mid h_2!\langle \mathsf{quote}\rangle.Q\{h_2/z\}) \\[4pt]
\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset; \emptyset; k_1 : S' \cdot k_2 : S' \vdash \\
 & (\nu\, h_1, h_2)(\overline{h_1} \triangleleft \mathsf{accept}.\overline{h_2} \triangleleft \mathsf{reject}.\mathbf{0} \mid Q\{h_1/z\} \mid Q\{h_2/z\}) \\[4pt]
\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset; \emptyset; k_1 : S' \cdot k_2 : S' \vdash k_1 \triangleleft \mathsf{accept}.k_1!\langle \mathsf{credit}\rangle.\mathbf{0} \mid k_2 \triangleleft \mathsf{reject}.\mathbf{0} \\[4pt]
\xmapsto{k_1 \oplus \mathsf{accept}}\xmapsto{k_2 \oplus \mathsf{reject}}\xmapsto{k_1 \oplus \mathsf{credit}} & \emptyset; \emptyset; \emptyset \vdash \mathbf{0}
\end{array}
$$

Fig. 7: Observable actions from Client_1 (cf. § 5.9).

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 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 π-calculus into the monadic synchronous π-calculus. Later typed equivalences of a family of linear and affine calculi [2, 36, 3] were used to encode PCF [29, 22], the simply typed λ-calculus with sums and products, and System F [6] fully abstractly (a fully abstract encoding of the λ-calculi was an open problem in [23]). Yoshida, Honda, and Berger [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 [12, 1].

Our work follows the behavioural semantics in [19, 18, 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

$$
\begin{array}{ll}
 & \emptyset;\emptyset;\Delta \vdash \mathsf{Client}_2 \\[4pt]
\xmapsto{s_1!\langle \lambda x.\, Q_1\{{}^{h}\!/y\}\rangle} & \emptyset;\emptyset; s_2 :!\langle S{\multimap}\diamond\rangle;\mathsf{end} \cdot k_1 : S \vdash (\nu\, h)(s_2!\langle \lambda x.\, Q_2\{{}^{\overline{h}}\!/y\}\rangle.\mathbf{0} \\
 & \mid t_1 \Leftarrow_{\mathtt{C}} Q_1\{{}^{h}\!/y\} : S{\multimap}\diamond) \\[4pt]
\xmapsto{s_2!\langle \lambda x.\, Q_2\{{}^{\overline{h}}\!/y\}\rangle} & \emptyset;\emptyset; k_1 : S \cdot k_2 : S \vdash \\
 & (\nu\, h)(t_1 \Leftarrow_{\mathtt{C}} Q_1\{{}^{h}\!/y\} : S{\multimap}\diamond \mid t_2 \Leftarrow_{\mathtt{C}} Q_2\{{}^{\overline{h}}\!/y\} : S{\multimap}\diamond) \\[4pt]
\xmapsto{t_1?\langle b\rangle}\xmapsto{t_2?\langle b\rangle}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset;\emptyset; k_1 : S \cdot k_2 : S \vdash (\nu\, h)(P\{{}^{h}\!/y\}\{{}^{k_1}\!/x\} \mid P_{xy}\{{}^{\overline{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})) \\[4pt]
\xmapsto{t_3!\langle s_1'\rangle}\xmapsto{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}\{{}^{\overline{h}}\!/y\}\{{}^{k_2}\!/x\} \\
 & \mid (\nu\, s_1', s_2')(t_3' \Leftarrow_{\mathtt{C}} s_1' : \mathsf{end} \mid t_4' \Leftarrow_{\mathtt{C}} s_2' : \mathsf{end})) \\[4pt]
\xmapsto{t_3'!\langle c\rangle}\xmapsto{t_4'!\langle c\rangle} & \\
\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset;\emptyset; k_1 : S \cdot k_2 : S \vdash (\nu\, h)(P\{{}^{h}\!/y\}\{{}^{k_1}\!/x\} \mid P_{xy}\{{}^{\overline{h}}\!/y\}\{{}^{k_2}\!/x\}) \\[4pt]
\xmapsto{k_1!\langle \mathsf{room}\rangle}\xmapsto{k_2!\langle \mathsf{room}\rangle} & \\
\xmapsto{k_1?\langle \mathsf{quote}\rangle}\xmapsto{k_2?\langle \mathsf{quote}\rangle} & \emptyset;\emptyset; k_1 : S' \cdot k_2 : S' \vdash (\nu\, h)(h!\langle \mathsf{quote}_1\rangle.h?(\mathsf{quote}_2).R\{{}^{k_1}\!/x\} \\
 & \mid \overline{h}?(\mathsf{quote}_2).\overline{h}!\langle \mathsf{quote}_1\rangle.R\{{}^{k_2}\!/x\}) \\[4pt]
\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset;\emptyset; k_1 : S' \cdot k_2 : S' \vdash R\{{}^{k_1}\!/x\} \mid R\{{}^{k_2}\!/x\} \\[4pt]
\xmapsto{\tau_{\mathsf{d}}}\xmapsto{\tau_{\mathsf{d}}} & \emptyset;\emptyset; k_1 : S' \cdot k_2 : S' \vdash k_1 \triangleleft \mathsf{accept}.k_1!\langle \mathsf{credit}\rangle.\mathbf{0} \mid k_2 \triangleleft \mathsf{reject}.\mathbf{0} \\[4pt]
\xmapsto{k_1 \oplus \mathsf{accept}}\xmapsto{k_2 \oplus \mathsf{reject}}\xmapsto{k_1 \oplus \mathsf{credit}} & \emptyset;\emptyset;\emptyset \vdash \mathbf{0}
\end{array}
$$

Fig. 8: Observable actions from Client_2 (cf. § 5.9).

(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 the 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 [31,30] and quite relevant in session-based concurrency. The bisimulation in [13] is based on an LTS extended with trigger meta-notation. As in [31,30], 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 [31,13] in that the mapping between higher- and first-order values is no longer implicit.

Comparison with respect to [13]. We briefly contrast the approach in [13] and ours based on characteristic bisimilarity ($\approx^{\mathsf{C}}$):

- 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, $\approx^{\mathsf{C}}$ 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, $\approx^{\mathsf{C}}$ decrees the input of a trigger value $\lambda z.\, t?(x).(x\, z)$.
- Unlike [13], $\approx^{\mathsf{C}}$ 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, $\mathsf{HO}\pi$ lacks name matching and we use session types: a characteristic value inhabiting a type enables the simplest form of interactions with the environment.

We compare our approach to that in [13] using a representative example.

Example 5 Let $V = \lambda x.\, x\, (\lambda y.\, y!\langle m\rangle.\mathbf{0})$ be a value. Consider a process such that

$$\Gamma; \emptyset; \Delta \cdot n :!\langle U\rangle; \mathtt{end} \vdash n!\langle V\rangle.\mathbf{0} \triangleright \diamond$$

with $U = (((!\langle S\rangle; \mathtt{end}) \rightarrow \diamond) \rightarrow \diamond) \rightarrow \diamond$. We compare our approach to that in [13] by contrasting the transitions from P. In our framework, first we have a typed transition $\Gamma; \emptyset; \Delta \cdot n :!\langle U\rangle; \mathtt{end} \vdash P \xrightarrow{n!\langle V\rangle} \Gamma; \emptyset; \Delta \vdash \mathbf{0}$. In the framework of [13] a similar (but untyped) output transition takes place. Figure 9 presents a complete comparison of the labelled transitions in our approach (Figure 9a) and in [13] (Figure 9b). In our approach, we let

$$\llparenthesis U \rrparenthesis^{x} = x\, (\lambda y.\, (y\, a)) \qquad \text{for some fresh } a$$

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 Figure 9b) and at the end, two replicated processes remain. This is how linearity information in session types allows us to have simpler bisimulations. Note that τ_l and τ_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.

$$\begin{array}{lcl}
 & & t \Leftarrow_{\mathsf{C}} V : U \\
 & = & \Gamma; \emptyset; \Delta \vdash t?(z).(\nu\, s)(s?(x).\{\![U]\!\}^{x} \mid \overline{s}!\langle \lambda x.\, x\,(\lambda y.\, y!\langle m\rangle.\mathbf{0})\rangle.\mathbf{0}) \\
 & = & \Gamma; \emptyset; \Delta \vdash t?(z).(\nu\, s)(s?(x).x\,(\lambda y.\,(y\,a)) \mid \overline{s}!\langle \lambda x.\, x\,(\lambda y.\, y!\langle m\rangle.\mathbf{0})\rangle.\mathbf{0}) \\
(1) & \xrightarrow{t?\langle b\rangle} & \Gamma; \emptyset; \Delta \vdash (\nu\, s)(s?(x).x\,(\lambda y.\,(y\,a)) \mid \overline{s}!\langle \lambda x.\, x\,(\lambda y.\, y!\langle m\rangle.\mathbf{0})\rangle.\mathbf{0}) \\
(2) & \xrightarrow{\tau_{\mathsf{d}}} & \Gamma; \emptyset; \Delta \vdash \lambda x.\, x\,(\lambda y.\, y!\langle m\rangle.\mathbf{0})\,(\lambda y.\,(y\,a)) \\
(3) & \xrightarrow{\tau_{\mathsf{d}}} & \Gamma; \emptyset; \Delta \vdash (\lambda y.\,(y\,a))\,(\lambda y.\, y!\langle m\rangle.\mathbf{0}) \\
(4) & \xrightarrow{\tau_{\mathsf{d}}} & \Gamma; \emptyset; \Delta \vdash (\lambda y.\, y!\langle m\rangle.\mathbf{0})\,a \\
(5) & \xrightarrow{\tau_{\mathsf{d}}} & \Gamma; \emptyset; \Delta \vdash a!\langle m\rangle.\mathbf{0}
\end{array}$$

(a) Our approach.

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

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

Fig. 9: Comparing labelled transitions associated to the process in Example 5.

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 $\approx^{\mathtt{H}}$, 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 $\mathsf{HO}\pi$. 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 insightful remarks, which greatly helped us to improve the quality of this document.

This work has been partially sponsored by the Doctoral Prize Fellowship, EPSRC EP/K011715/1, EPSRC EP/K034413/1, and EPSRC EP/L00058X/1, EU project FP7-612985 UpScale, and EU COST Action IC1201 BETTY. 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.

References

1. S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.
2. M. Berger, K. Honda, and N. Yoshida. Sequentiality and the π-calculus. In *Proc. TLCA'01*, volume 2044 of *LNCS*, pages 29–45. Springer, 2001.
3. M. Berger, K. Honda, and N. Yoshida. Genericity and the pi-calculus. *Acta Inf.*, 42(2-3):83–141, 2005.
4. G. Bernardi, O. Dardha, S. J. Gay, and D. Kouzapas. On duality relations for session types. In M. Maffei and E. Tuosto, editors, *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*, pages 51–66. Springer, 2014.
5. S. J. Gay and V. T. Vasconcelos. Linear type theory for asynchronous session types. *J. Funct. Program.*, 20(1):19–50, 2010.
6. J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. CUP, 1989.
7. J. F. Groote and M. P. A. Sellink. Confluence for process verification. *Theor. Comput. Sci.*, 170(1-2):47–81, 1996.
8. M. Hennessy. *A Distributed Pi-Calculus*. CUP, 2007.
9. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In C. Hankin, editor, *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*, volume 1381 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 1998.
10. K. Honda and N. Yoshida. On reduction-based process semantics. *Theor. Comput. Sci.*, 151(2):437–486, 1995.
11. K. Honda and N. Yoshida. A uniform type structure for secure information flow. *ACM Trans. Program. Lang. Syst.*, 29(6), 2007.
12. J. M. E. Hyland and C. L. Ong. On full abstraction for PCF: i, ii, and III. *Inf. Comput.*, 163(2):285–408, 2000.
13. A. Jeffrey and J. Rathke. Contextual equivalence for higher-order pi-calculus revisited. *Logical Methods in Computer Science*, 1(1), 2005.
14. N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the pi-calculus. *ACM Trans. Program. Lang. Syst.*, 21(5):914–947, 1999.
15. V. Koutavas and M. Hennessy. First-order reasoning for higher-order concurrency. *Computer Languages, Systems & Structures*, 38(3):242–277, 2012.
16. D. Kouzapas, J. A. Pérez, and N. Yoshida. Characteristic Bisimulation for Higher-Order Session Processes. In L. Aceto and D. de Frutos Escrig, editors, *26th International Conference on Concurrency Theory (CONCUR 2015)*, volume 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 398–411, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

17. D. Kouzapas, J. A. Pérez, and N. Yoshida. On the relative expressiveness of higher-order session processes. In P. Thiemann, editor, *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*, volume 9632 of *Lecture Notes in Computer Science*, pages 446–475. Springer, 2016.
18. D. Kouzapas and N. Yoshida. Globally governed session semantics. *Logical Methods in Computer Science*, 10(4), 2014.
19. D. Kouzapas, N. Yoshida, R. Hu, and K. Honda. On asynchronous eventful session semantics. *MSCS*, 2015.
20. I. Lanese, J. A. Pérez, D. Sangiorgi, and A. Schmitt. On the expressiveness and decidability of higher-order process calculi. *Inf. Comput.*, 209(2):198–226, 2011.
21. S. Lenglet and A. Schmitt. Howe's Method for Contextual Semantics. In L. Aceto and D. de Frutos Escrig, editors, *26th International Conference on Concurrency Theory (CONCUR 2015)*, volume 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 212–225, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
22. R. Milner. Fully abstract models of typed *lambda*-calculi. *Theor. Comput. Sci.*, 4(1):1–22, 1977.
23. R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992.
24. R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th ICALP*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.
25. D. Mostrous and N. Yoshida. Two session typing systems for higher-order mobile processes. In S. R. D. Rocca, editor, *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4583 of *Lecture Notes in Computer Science*, pages 321–335. Springer, 2007.
26. D. Mostrous and N. Yoshida. Session typing and asynchronous subtyping for the higher-order π-calculus. *Inf. Comput.*, 241:227–263, 2015.
27. J. A. Pérez, L. Caires, F. Pfenning, and B. Toninho. Linear logical relations and observational equivalences for session-based concurrency. *Inf. Comput.*, 239:254–302, 2014.
28. B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *MSCS*, 6(5):409–454, 1996.
29. G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223 – 255, 1977.
30. D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher Order Paradigms.* PhD thesis, University of Edinburgh, 1992.
31. D. Sangiorgi. Bisimulation for Higher-Order Process Calculi. *Inf. & Comp.*, 131(2):141–178, 1996.
32. D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 293–302. IEEE Computer Society, 2007.
33. B. Thomsen. Plain CHOCS: A Second Generation Calculus for Higher Order Processes. *Acta Informatica*, 30(1):1–59, 1993.
34. X. Xu. On context bisimulation for parameterized higher-order processes. In *Proc. of ICE 2013*, volume 131 of *EPTCS*, pages 37–51, 2013.
35. N. Yoshida. Graph types for monadic mobile processes. In *FSTTCS*, volume 1180 of *LNCS*, pages 371–386. Springer, 1996.
36. N. Yoshida, M. Berger, and K. Honda. Strong normalisation in the pi -calculus. *Inf. Comput.*, 191(2):145–202, 2004.
37. N. Yoshida, K. Honda, and M. Berger. Linearity and bisimulation. In M. Nielsen and U. Engberg, editors, *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*, pages 417–434. Springer, 2002.

A The Typing System of HOπ

We first formally define *type equivalence*. Then we give details of the proof of Theorem 1 (page 12).

A.1 Type Equivalence

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

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

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

A.2 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 $x \notin \mathtt{dom}(\Gamma, \Lambda, \Delta)$ then $\Gamma \cdot x : U {\to} \diamond; \Lambda; \Delta \vdash P \triangleright \diamond$

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

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

Below, shared value means that there are no free linear names, thus Λ, Δ are empty (cf. Rule [Prom]).

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

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

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

We now state the instance of type soundness that we can derive from [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 typing associated to queues and other runtime elements. Since we are working with synchronous communication such an ordering can be omitted. The second part of the following statement corresponds to Theorem 1 (Page 12):

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 Δ. Then $P \longrightarrow P'$ implies $\Gamma; \emptyset; \Delta' \vdash P' \triangleright \diamond$ and $\Delta = \Delta'$ or $\Delta \longrightarrow \Delta'$.

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

1. Case [App]: Then we have

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

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

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

Then, by combining premise $\Gamma;\, \emptyset;\, \Delta \cdot \{x : S\} \vdash Q \triangleright \diamond$ with the substitution lemma (Lemma 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:

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

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

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

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

   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:

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

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

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

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

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

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

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

$$\dfrac{\Gamma;\, \emptyset;\, \emptyset \vdash n \triangleright \langle L\rangle \quad \Gamma;\, \emptyset;\, \Delta_1 \vdash Q_1 \triangleright \diamond \quad \Gamma;\, \emptyset;\, \emptyset \vdash V \triangleright L}{\Gamma;\, \emptyset;\, \Delta_1 \vdash n!\langle V\rangle.Q_1 \triangleright \diamond} \tag{13}$$

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

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

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

and the case is completed by using Rule [Par] with this 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).

B Proofs for Section 5

B.1 Typability of Characteristic Processes

We state and prove a more detailed form of Proposition 1 (Page 18). 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 $\mathsf{unfold}(S)$ is defined as:

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

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

Proof A straightforward induction on the syntax of S. $\square$

We state session type prefix deletion relation:

Definition 23 (Session Type Prefix Deletion) Given a session type S, the set $\mathsf{del}(S)$ is defined inductively as follows:

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

We may now finally state and prove the following proposition:

Proposition 4 (Characteristic Processes/Values Inhabit Their Types)

1. *Let U and $\llparenthesis U \rrparenthesis_{\mathsf{c}}$ be a type and its characteristic value, respectively.*
   (a) If $U = S$ then, for some s, we have $\emptyset; \emptyset; s : S \vdash \llparenthesis S \rrparenthesis_{\mathsf{c}} \triangleright S$.
   (b) If $U = \langle S\rangle$ then, for some a, we have $a : \langle S\rangle; \emptyset; \emptyset \vdash \llparenthesis \langle S\rangle \rrparenthesis_{\mathsf{c}} \triangleright \langle S\rangle$.
   (c) If $U = \langle L\rangle$ then, for some a, we have $a : \langle L\rangle; \emptyset; \emptyset \vdash \llparenthesis \langle L\rangle \rrparenthesis_{\mathsf{c}} \triangleright \langle L\rangle$.
   (d) If $U = U' \rightarrow \diamond$ and $\Gamma; \emptyset; \Delta \vdash \llparenthesis U' \rrparenthesis^x \triangleright \diamond$ then we have $\Gamma \backslash x; \emptyset; \Delta \backslash x \vdash \llparenthesis U' \rightarrow \diamond \rrparenthesis_{\mathsf{c}} \triangleright U' \rightarrow \diamond$.
   (e) If $U = U' \multimap \diamond$ and $\Gamma; \emptyset; \Delta \vdash \llparenthesis U' \rrparenthesis^x \triangleright \diamond$ then we have $\Gamma \backslash x; \emptyset; \Delta \backslash x \vdash \llparenthesis U' \multimap \diamond \rrparenthesis_{\mathsf{c}} \triangleright U' \multimap \diamond$.
2. *Let S and $\llparenthesis S \rrparenthesis^s$ be a session type and its characteristic process, respectively.*
   (a) If $S = \mathsf{end}$ then $\emptyset; \emptyset; \emptyset; \vdash \llparenthesis \mathsf{end} \rrparenthesis^s \triangleright \diamond$.
   (b) If $S = !\langle U\rangle; S'$ and $\Gamma; \emptyset; \Delta \vdash \llparenthesis U \rrparenthesis_{\mathsf{c}} \triangleright U$ then $\Gamma; \emptyset; \Delta \cdot t :!\langle S'\rangle; \mathsf{end} \cdot s :!\langle U\rangle; S' \vdash \llparenthesis !\langle U\rangle; S' \rrparenthesis^s \triangleright \diamond$.
   (c) If $S = ?(U); S'$ and $\Gamma; \emptyset; \Delta \vdash \llparenthesis U \rrparenthesis^x \triangleright \diamond$ then $\Gamma \backslash x; \emptyset; (\Delta \backslash x) \cdot t :?(S'); \mathsf{end} \cdot s :!\langle U\rangle; S' \vdash \llparenthesis ?(U); S' \rrparenthesis^s \triangleright \diamond$.
   (d) If $S = \oplus\{l_i : S_i\}_{i\in I}$ then $\emptyset; \emptyset; \{t_i :!\langle S_i\rangle; \mathsf{end}\}_{i\in I} \cdot s : \oplus\{l_i : S_i\}_{i\in I} \vdash \llparenthesis \oplus\{l_i : S_i\}_{i\in I} \rrparenthesis^s \triangleright \diamond$.
   (e) If $S = \&\{l_i : S_i\}_{i\in I}$ then $\emptyset; \emptyset; \{t_i :!\langle S_i\rangle; \mathsf{end}\}_{i\in I} \cdot s : \&\{l_i : S_i\}_{i\in I} \vdash \llparenthesis \&\{l_i : S_i\}_{i\in I} \rrparenthesis^s \triangleright \diamond$.
   (f) If $S = \mu\mathsf{t}.S'$ then either

– $\emptyset; \emptyset; \emptyset \vdash \llparenthesis \mu \mathsf{t}.S' \rrparenthesis^{s} \triangleright \diamond$
– *for all* $S_i \in \mathsf{del}(S)$ *there exist* Γ, Δ, *and* S'_i *such that*

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

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

3. *Let* U *and* $\llparenthesis U \rrparenthesis^{a}$ *be a channel type and its characteristic process, respectively.*
   (a) *If* $U = \langle S \rangle$ *and* $\emptyset; \emptyset; \Delta \vdash \llparenthesis S \rrparenthesis_{\mathsf{c}} \triangleright S$ *then* $a : \langle S \rangle; \emptyset; \Delta \cdot t :!\langle \langle S \rangle \rangle; \mathsf{end} \vdash \llparenthesis \langle S \rangle \rrparenthesis^{a} \triangleright \diamond$.
   (b) *If* $U = \langle L \rangle$ *and* $\Gamma; \emptyset; \Delta \vdash \llparenthesis L \rrparenthesis_{\mathsf{c}} \triangleright L$ *then* $\Gamma \cdot a : \langle L \rangle; \emptyset; \Delta \cdot t :!\langle \langle L \rangle \rangle; \mathsf{end} \vdash \llparenthesis \langle L \rangle \rrparenthesis^{a} \triangleright \diamond$.
   (c) *If* $U = U' \rightarrow \diamond$ *and* $\Gamma; \emptyset; \Delta \vdash \llparenthesis U' \rrparenthesis_{\mathsf{c}} \triangleright U'$ *then* $\Gamma \cdot x : U' \rightarrow \diamond; \emptyset; \Delta \vdash \llparenthesis U' \rightarrow \diamond \rrparenthesis^{x} \triangleright \diamond$.
   (d) *If* $U = U' \multimap \diamond$ *and* $\Gamma; \emptyset; \Delta \vdash \llparenthesis U' \rrparenthesis_{\mathsf{c}} \triangleright U'$ *then* $\Gamma \cdot x : U' \rightarrow \diamond; \emptyset; \Delta \vdash \llparenthesis U' \multimap \diamond \rrparenthesis^{x} \triangleright \diamond$.

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. Figure 3).
   - Case (d) $U = U' \rightarrow \diamond$: By Parts 2 and 3 of this lemma we obtain $\Gamma; \emptyset; \Delta \vdash \llparenthesis U' \rrparenthesis^{x} \triangleright \diamond$, which implies $\Gamma \backslash x; \emptyset; \Delta \backslash x \vdash \llparenthesis U' \rightarrow \diamond \rrparenthesis_{\mathsf{c}} \triangleright U' \rightarrow \diamond$ by Rules [Abs] and [EProm] (cf. Figure 3).
   - Case (e) $U = U' \multimap \diamond$: Similar, using Rule [Abs] (cf. Figure 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 $\llparenthesis S \rrparenthesis^{s} = s!\langle \llparenthesis U \rrparenthesis_{\mathsf{c}} \rangle.t!\langle s \rangle.\mathbf{0}$ and we may obtain the following derivation:

$$\dfrac{\begin{array}{c}\Gamma; \emptyset; s : S' \cdot t :!\langle S' \rangle; \mathsf{end} \triangleright t!\langle s \rangle.\mathbf{0} \triangleright \diamond \qquad \text{(Induction)} \\ \Gamma; \emptyset; \Delta \vdash \llparenthesis U \rrparenthesis_{\mathsf{c}} \triangleright U\end{array}}{\Gamma; \emptyset; \Delta \cdot s :!\langle U \rangle; S' \cdot t :!\langle S \rangle; \mathsf{end} \triangleright s!\langle \llparenthesis U \rrparenthesis_{\mathsf{c}} \rangle.t!\langle s \rangle.\mathbf{0} \triangleright \diamond}$$

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

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

   (c) Case $S = \mu \mathsf{t}.S'$: Then, by Definition 13, $\llparenthesis S \rrparenthesis = \llparenthesis S'\{\mathsf{end}/\mathsf{t}\} \rrparenthesis^{u}$. 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 $\llparenthesis S'\{\mathsf{end}/\mathsf{t}\} \rrparenthesis^{s} = s \triangleright \{l_i : t_i!\langle s \rangle.\mathbf{0}\}_{i \in I}$ and $\mathsf{del}(S) = \{S_i\}_{i \in I}$:

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

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

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

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

$$\dfrac{\text{Induction}}{\Gamma; \emptyset; \Delta \cdot t_i : S_i\{\mathsf{end}/\mathsf{t}\}\{\mathsf{end}/\mathsf{t}'\} \cdot s : S''\{\mathsf{end}/\mathsf{t}\}\{\mathsf{end}/\mathsf{t}'\} \vdash \llparenthesis S''\{\mathsf{end}/\mathsf{t}\}\{\mathsf{end}/\mathsf{t}'\} \rrparenthesis^{s} \triangleright \diamond}$$

      We may then type $\llparenthesis S \rrparenthesis^{s}$:

$$\dfrac{\text{Induction}}{\Gamma; \emptyset; \Delta \cdot t_i : S_i \cdot s : S \vdash \llparenthesis \mu \mathsf{t}.\mu \mathsf{t}'.S'' \rrparenthesis^{s} \triangleright \diamond}$$

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 $\emptyset; \emptyset; \Delta \vdash (\!|S|\!)_{\mathsf{c}} \triangleright S$. By applying Rule [Req] (cf. Figure 3) we obtain:

$$\frac{a : \langle S \rangle; \emptyset; \Delta \vdash (\!|S|\!)_{\mathsf{c}} \triangleright S \qquad a : \langle S \rangle; \emptyset; t :!\langle\langle S \rangle\rangle; \mathtt{end} \vdash t!\langle a \rangle.\mathbf{0} \triangleright \diamond}{a : \langle S \rangle; \emptyset; \Delta \cdot t :!\langle\langle S \rangle\rangle; \mathtt{end} \vdash (\!|\langle S \rangle|\!)^{a} \triangleright \diamond}$$

   (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 $\Gamma; \emptyset; \Delta \vdash (\!|U'|\!)_{\mathsf{c}} \triangleright U'$. By applying Rules [App] and [EProm] (cf. Figure 3) we obtain:

$$\frac{\dfrac{\Gamma; \emptyset; \Delta \vdash (\!|U'|\!)_{\mathsf{c}} \triangleright U' \qquad \Gamma; x : U' \multimap \diamond; \emptyset \vdash x \triangleright U' \multimap \diamond}{\Gamma; x : U' \multimap \diamond; \Delta \vdash x\, (\!|U'|\!)_{\mathsf{c}} \triangleright \diamond}}{\Gamma \cdot x : U' \to \diamond; \emptyset; \Delta \vdash (\!|U' \multimap \diamond|\!)^{x} \triangleright \diamond}$$

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

□

B.2 Deterministic Transitions

The proofs for Theorem 2 (Page 26) require an auxiliary result on deterministic transitions (Lemma 1, Page 22). 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 $u!\langle\tilde{V}\rangle.P$ and $u?(\tilde{x}).P$, respectively) as shorthand notations.

We now prove Proposition 2, as stated in Page 22:

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

1. $\Gamma; \Delta \vdash P \overset{\tau_{\mathsf{d}}}{\longmapsto} \Delta' \vdash P'$ implies $\Gamma; \Delta \vdash P \approx^{\mathtt{H}} \Delta' \vdash P'$.
2. $\Gamma; \Delta \vdash P \overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta' \vdash P'$ implies $\Gamma; \Delta \vdash P \approx^{\mathtt{H}} \Delta' \vdash P'$.

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

$$\Re = \{(P, P') \mid \Gamma; \Delta \vdash P \overset{\tau_{\mathsf{d}}}{\longmapsto} \Delta' \vdash P'\}$$

is a higher-order bisimulation.

Suppose first that $\Gamma; \Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P''$, 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). 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. The second case can be divided into two sub-cases: first, if $\tau = \tau_{\mathsf{d}}$ then $P' = P''$ and the thesis trivially follows; second, if $\tau \neq \tau_{\mathsf{d}}$ (i.e., P has the possibility of performing both τ_{d} and some other τ) then either P' has the same τ or P' does not have it, because τ_{d} excluded the occurrence of τ. We 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 $\Gamma; \Delta \vdash P' \overset{\ell}{\longmapsto} \Delta' \vdash P''$, for some P''. This case follows immediately by noticing that P can always match action ℓ by performing the deterministic action τ_{d} first, i.e., we can always have $\Gamma; \Delta \vdash P \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{\ell}{\longmapsto} \Delta' \vdash P''$. This concludes the proof. □

B.3 Proof of Theorem 2

We split the proof of Theorem 2 (Page 26) into several lemmas:

- Lemma 14 establishes $\approx^{\mathtt{H}} = \approx^{\mathtt{C}}$.
- Lemma 16 establishes a trigger substitution lemma (Lemma 4 in the main text), using Lemma 15 (Page 47).
- Lemma 18 exploits the process substitution result given by Lemma 17 (Lemma 3 in the main text) to prove that $\approx^{\mathtt{H}} \subseteq \approx$.
- Lemma 19 shows that $\approx$ is a congruence which implies $\approx \subseteq \cong$.
- Lemma 22 shows that $\cong \subseteq \approx^{\mathtt{H}}$ 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 ℓ be an action label (cf. § 5.1). We define the action $\breve{\ell}$ as

$$\breve{\ell} = \begin{cases} (\nu\, \widetilde{m_2})(n!\langle V_2\rangle) & \text{if } \ell = (\nu\, \widetilde{m_1})(n!\langle V_1\rangle), \text{ for some } V_2, \widetilde{m_2} \\ \ell & \text{otherwise} \end{cases}$$

Thus, given ℓ, its corresponding action $\breve{\ell}$ is either identical to ℓ, 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

$$t \leftharpoondown V = t?(x).(\nu\, s)(x\, s \mid \overline{s}!\langle V\rangle.\mathbf{0}) \tag{15}$$

The first auxiliary lemma states the equivalence (up to $\approx^{\mathtt{H}}$) between higher-order trigger processes $t \hookleftarrow_{\mathtt{H}} V$ (cf. (6)) and $t \leftharpoondown V$.

Lemma 12 (Alternative Trigger Process) *Let P and Q be processes.*

1. *Let t be a fresh name, $\Delta_1 = \Delta_3 \cdot t :!\langle \mathtt{end}\rangle;\mathtt{end}$, and $\Delta_2 = \Delta_4 \cdot t :!\langle \mathtt{end}\rangle;\mathtt{end}$. Then:*

$$\Gamma;\Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0}))$$

   if and only if $\Gamma;\Delta_3 \vdash (\nu\, \widetilde{m_1})P \approx^{\mathtt{H}} \Delta_4 \vdash (\nu\, \widetilde{m_2})Q$ for some Δ_3, Δ_4.
2. *Let t a fresh name, $\Delta_1 = \Delta_3 \cdot t :!\langle \mathtt{end}\rangle;\mathtt{end}$ and $\Delta_2 = \Delta_4 \cdot t :!\langle \mathtt{end}\rangle;\mathtt{end}$. Then:*

$$\Gamma;\Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{C}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0}))$$

   if and only if $\Gamma;\Delta_3 \vdash (\nu\, \widetilde{m_1})P \approx^{\mathtt{C}} \Delta_4 \vdash (\nu\, \widetilde{m_2})Q$ for some Δ_3, Δ_4.
3. *Let t be a fresh name. Then*

$$\Gamma;\Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \leftharpoondown V_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \leftharpoondown V_2)$$

   if and only if, for some Δ_3, Δ_4,

$$\Gamma;\Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \approx^{\mathtt{H}} \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2)$$

Proof We analyze each of the three parts:

- Part 1. 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):

$$\begin{aligned}\Re = \{((\nu\, \widetilde{m_1})P\ ,\ (\nu\, \widetilde{m_2})Q) \ \mid \\ \Gamma;\Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0}))\}\end{aligned}$$

  We check the requirements of higher-order bisimulation for $\Re$. Suppose that

$$\Gamma;\Delta_3 \vdash (\nu\, \widetilde{m_1})P \overset{\ell}{\longmapsto} \Delta_3' \vdash (\nu\, \widetilde{m_1}')P'$$

  then we need to show a matching action from $(\nu\, \widetilde{m_2})Q$. We can derive that

$$\Gamma;\Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0})) \overset{\ell}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid (\nu\, s)(t!\langle s\rangle.\mathbf{0}))$$

for some Δ_1' which, from the freshness of t, implies that there exist Q' and Δ_2' such that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \overset{\breve{\ell}}{\Longrightarrow} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \qquad (16)$$

and

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid C_1 \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid C_2 \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0}))$$

with $C_1 = t' \hookleftarrow_{\mathtt{H}} V_1$ and $C_2 = t' \hookleftarrow_{\mathtt{H}} V_1$ (if ℓ and $\breve{\ell}$ are output actions with objects V_1 and V_2, respectively) and $C_1 = C_2 = \mathbf{0}$ (otherwise). The latter equation implies from the definition of $\Re$

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid C_1 \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \;\Re\; \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid C_2 \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0}))$$

and (16) implies

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})Q \overset{\breve{\ell}}{\Longrightarrow} \Delta_2' \vdash (\nu\, \widetilde{m_2}')Q'$$

to complete the proof of the case.

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

$$\Re = \{((\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0}))\ ,\ (\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})))\ \mid \\ \Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P) \approx^{\mathtt{H}} \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q)\}$$

We check the requirements of higher-order bisimulation for $\Re$.
Suppose that $(\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0}))$ moves; we need to infer an appropriate matching action from $(\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0}))$. We analyse three cases:

i) Process P moves autonomously, i.e., for some Δ_1' we have:

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \overset{\ell}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0}))$$

Then the proof is similar to the previous case.

ii) An action on the fresh name t, , i.e., for some Δ_1' we have:

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \overset{t!\langle s \rangle}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1})P$$

First notice that the typing derivation reveals that $\Delta_1(t) = \Delta_2(t) = !\langle \mathtt{end} \rangle; \mathtt{end}$. 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 $\mathtt{end}$. We can then observe that, for some Δ_2' we have:

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\nu\, s)(t!\langle s \rangle.\mathbf{0})) \overset{t!\langle s \rangle}{\Longrightarrow} \Delta_2' \vdash (\nu\, \widetilde{m_2})Q'$$

We need to show that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t' \hookleftarrow_{\mathtt{H}} s) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q' \mid t' \hookleftarrow_{\mathtt{H}} s)$$

The proof is easy if we consider that both processes can perform the up-to deterministic transitions $\overset{t'?\langle \lambda z.\, \mathbf{0} \rangle}{\longmapsto} \overset{\tau_{\mathsf{d}}}{\Longrightarrow}$:

$$\begin{aligned} & \Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t' \hookleftarrow_{\mathtt{H}} s) \\ \overset{t'?\langle \lambda z.\, \mathbf{0} \rangle}{\longmapsto}\ & \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s')(s'?(y).((\lambda z.\, \mathbf{0})\, y) \mid \overline{s'}!\langle s \rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\longmapsto}\ & \Delta_1' \vdash (\nu\, \widetilde{m_1})P \end{aligned}$$

and

$$\begin{aligned} & \Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid t' \hookleftarrow_{\mathtt{H}} s) \\ \overset{t'?\langle \lambda z.\, \mathbf{0} \rangle}{\longmapsto}\ & \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q' \mid (\nu\, s')(s'?(y).((\lambda z.\, \mathbf{0})\, y) \mid \overline{s'}!\langle s \rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\longmapsto}\ & \Delta_2' \vdash (\nu\, \widetilde{m_2})Q' \end{aligned}$$

The result is then immediate from the definition of $\Re$ that requires

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})P \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})Q'$$

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 $\Re$ be the typed relation (we omit the type information):

$$\begin{aligned}\Re = \{((\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1)\ ,\ (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2))\ \mid \\ \Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \leftharpoondown V_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \leftharpoondown V_2)\}\end{aligned}$$

We show that $\Re \subseteq \approx^{\mathtt{H}}$, with a case analysis on the defining requirements of higher-order bisimulation. Suppose that $(\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1)$ moves; we need to show an appropriate matching action from $(\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2)$. We analyze three possibilities:

i) P moves on its own, i.e., for some Δ_1' we have:

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \overset{\ell}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid t \hookleftarrow_{\mathtt{H}} V_2)$$

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

ii) An input action of the form $t?\langle n\rangle$ along fresh name t. In this case, we have that there exist Δ_1' and U such that $(\!|U|\!)_{\mathsf{c}} = n$:

$$\begin{aligned}\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) &\overset{t?\langle n\rangle}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)(s?(y).(y\, n) \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ &\overset{\tau_{\mathsf{d}}}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n))\end{aligned}$$

Furthermore, we can see that, for some Δ_2', we have

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2) \overset{t?\langle n\rangle}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n))$$

We therefore need to show that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n)) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n))$$

This is done by considering the requirements of $\Re$. Since $(\!|U|\!)_{\mathsf{c}} = n$, we have that $(\!|?(U); \mathtt{end}|\!)_{\mathsf{c}} = \lambda z.\, z?(y).(t'!\langle z\rangle.\mathbf{0} \mid (y\, n))$:

$$\begin{aligned}&\Gamma; \emptyset;\ \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \leftharpoondown V_1) \\ \overset{t?\langle (\!|?(U);\mathtt{end}|\!)_{\mathsf{c}}\rangle}{\longmapsto}\ &\Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)(\lambda z.\, z?(y).(t'!\langle z\rangle.\mathbf{0} \mid (y\, n))\, s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\end{aligned}$$

Furthermore, we can see that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \leftharpoondown V_2) \overset{t?\langle (\!|?(U);\mathtt{end}|\!)_{\mathsf{c}}\rangle}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}))$$

with

$$\begin{aligned}&\Gamma; \emptyset;\ \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)(\lambda z.\, z?(y).(t'!\langle z\rangle.\mathbf{0} \mid (y\, n))\, s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{\tau_{\mathsf{d}}}{\longmapsto}\ &\Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}))\end{aligned}$$

and

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}))$$

which implies, by Part 1 of this lemma,

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n)) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n))$$

as required.

iii) An action of the form $t?\langle \lambda z.\,\llbracket U' \rrbracket^{z} \rangle$ along fresh name t. This means that there exist U and Δ_1' such that $\llbracket U \rrbracket_{\mathsf{c}} = \lambda z.\,\llbracket U' \rrbracket^{z}$ and

$$\begin{array}{rl} & \Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \\ \overset{t?\langle \lambda z.\,\llbracket U' \rrbracket^{z} \rangle}{\longmapsto} & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)(s?(y).((\lambda z.\,\llbracket U' \rrbracket^{z})\, y) \mid \overline{s}!\langle V_1 \rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\longmapsto} & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_1) \end{array}$$

Furthermore, we have the following, for some Δ_2':

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2) \overset{t?\langle \lambda z.\,\llbracket U' \rrbracket^{z} \rangle}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_2)$$

We need to show that, for some Δ_3', Δ_4', we have

$$\Gamma; \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_1) \approx^{\mathtt{H}} \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_2)$$

This is done by considering the requirements of $\Re$. From the fact that $\llbracket U \rrbracket_{\mathsf{c}} = \lambda z.\,\llbracket U' \rrbracket^{z}$ we obtain that $\llbracket ?(U); \mathtt{end} \rrbracket_{\mathsf{c}} = \lambda w.\, w?(y).(t'!\langle w \rangle.\mathbf{0} \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, y)$ and

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \leftharpoonup V_1) \\ \overset{t?\langle \llbracket ?(U);\mathtt{end} \rrbracket_{\mathsf{c}} \rangle}{\longmapsto} & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)((\lambda w.\, w?(y).(t'!\langle w \rangle.\mathbf{0} \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, y))\, s \mid \overline{s}!\langle V_1 \rangle.\mathbf{0})) \end{array}$$

Furthermore we can see that for some Δ_2'

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \leftharpoonup V_2) \overset{t?\langle \llbracket ?(U);\mathtt{end} \rrbracket_{\mathsf{c}} \rangle}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_2 \mid (\nu\, s)(t'!\langle s \rangle.\mathbf{0}))$$

with

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)((\lambda w.\, w?(y).(t'!\langle w \rangle.\mathbf{0} \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, y))\, s \mid \overline{s}!\langle V_1 \rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{\tau_{\mathsf{d}}}{\longmapsto} & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_1 \mid (\nu\, s)(t'!\langle s \rangle.\mathbf{0})) \end{array}$$

and

$$\begin{array}{c} \Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_1 \mid (\nu\, s)(t'!\langle s \rangle.\mathbf{0})) \approx^{\mathtt{H}} \\ \vdash \Delta_2'(\nu\, \widetilde{m_2}')(Q' \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_2 \mid (\nu\, s)(t'!\langle s \rangle.\mathbf{0})) \end{array}$$

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

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_1) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\lambda z.\,\llbracket U' \rrbracket^{z})\, V_2)$$

(b) Second direction. Let $\Re$ be the typed relation (we omit the type information):

$$\begin{array}{rl} \Re = \{ & ((\nu\, \widetilde{m_1})(P \mid t \leftharpoonup V_1)\ ,\ (\nu\, \widetilde{m_2})(Q \mid t \leftharpoonup V_2)) \ \mid \\ & \qquad \Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \approx^{\mathtt{H}} \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2)\} \end{array}$$

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

i. Value V_1 is a higher-order value: This implies that there exist U and Δ_1' such that $\llbracket ?(U); \mathtt{end} \rrbracket_{\mathsf{c}} = \lambda z.\, z?(y).(t'!\langle z \rangle.\mathbf{0} \mid y\, n)$ and

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \leftharpoonup V_1) \\ \overset{t?\langle \llbracket ?(U);\mathtt{end} \rrbracket_{\mathsf{c}} \rangle}{\longmapsto} & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)((\lambda z.\, z?(y).(t'!\langle z \rangle.\mathbf{0} \mid (y\, n)))\, s \mid \overline{s}!\langle V_1 \rangle.\mathbf{0})) \end{array}$$

and

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)((\lambda z.\, z?(y).(t'!\langle z \rangle.\mathbf{0} \mid (y\, n)))\, s \mid \overline{s}!\langle V_1 \rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\longmapsto}\overset{\tau_{\mathsf{d}}}{\longmapsto} & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n) \mid (\nu\, s)(t'!\langle s \rangle.\mathbf{0})) \end{array}$$

Furthermore we can see that there exists Δ_2' such that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \leftharpoonup V_2) \overset{t?\langle \llbracket ?(U);\mathtt{end} \rrbracket_{\mathsf{c}} \rangle}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n) \mid (\nu\, s)(t'!\langle s \rangle.\mathbf{0}))$$

We need to show that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q \mid (V_2\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}))$$

This is done by considering the requirements of $\Re$. We know that $\llparenthesis U \rrparenthesis_{\mathsf{c}} = n$:

$$\begin{aligned}&\Gamma; \emptyset;\ \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1)\\ &\overset{t?\langle n\rangle}{\longmapsto}\ \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)(s?(y).(y\, n) \mid \overline{s}!\langle V_1\rangle.\mathbf{0}) \overset{\tau_{\mathsf{d}}}{\longmapsto} \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n)))\end{aligned}$$

for some Δ_3'. Furthermore we can see that for some Δ_4'

$$\Gamma; \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2) \overset{t?\langle n\rangle}{\Longmapsto} \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n))$$

and

$$\Gamma; \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n)) \approx^{\mathtt{H}} \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n))$$

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

$$\Gamma; \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid (V_1\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{H}} \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (V_2\, n) \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}))$$

ii. Value V_1 is a first-order value: This implies that there exist U and Δ_3' such that $\llparenthesis ?(U); \mathtt{end} \rrparenthesis_{\mathsf{c}} = \lambda w.\, w?(y).(t'!\langle w\rangle.\mathbf{0} \mid \lambda z.\, \llparenthesis U' \rrparenthesis^{z}\, y)$ and

$$\begin{aligned}&\Gamma; \emptyset;\ \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1)\\ &\overset{\llparenthesis ?(U);\mathtt{end}\rrparenthesis_{\mathsf{c}}}{\longmapsto}\ \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid (\nu\, s)(\lambda w.\, w?(y).(t'!\langle w\rangle.\mathbf{0} \mid \lambda z.\, \llparenthesis U' \rrparenthesis^{z}\, y)\, s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}))\end{aligned}$$

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

This concludes the proof of Part 3.

□

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

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2)$$

then there exist Δ_1', Δ_2' such that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U).$$

2) If

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \approx^{\mathtt{C}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U)$$

then there exist Δ_1', Δ_2' such that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \approx^{\mathtt{C}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2).$$

Proof We analyse both parts separately:

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

$$\begin{aligned}\Re = \{((\nu\, \widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U), (\nu\, \widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U)) \ \mid\\ \Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2)\}\end{aligned}$$

We show that $\Re \subseteq \approx^{\mathtt{H}}$. Suppose that $(\nu\, \widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U)$ moves; we need to find a matching move from $(\nu\, \widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U)$. We distinguish three cases, depending on the source/kind of visible action:

(a) P moves autonomously, i.e., for some Δ_3 we have:

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t \Lleftarrow_{\mathtt{C}} V_1 : U) \overset{\ell}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1}')(P' \mid t \Lleftarrow_{\mathtt{C}} V_1 : U)$$

We follow the requirements of $\Re$ and the freshness of t to conclude that there exists a Δ_1'' such that

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \overset{\ell}{\longmapsto} \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P' \mid t \hookleftarrow_{\mathtt{H}} V_1)$$

which implies, from the higher-order bisimilarity requirement of $\Re$ and the freshness of t, that there exist Q' and Δ_2'' such that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2) \overset{\check{\ell}}{\Longmapsto} \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q' \mid t \hookleftarrow_{\mathtt{H}} V_2) \qquad (17)$$

and, for some Δ_1''' and Δ_2''', that

$$\Gamma; \Delta_1''' \vdash (\nu\, \widetilde{m_1}'')(P' \mid t \hookleftarrow_{\mathtt{H}} V_1 \mid C_1) \approx^{\mathtt{H}} \Delta_2''' \vdash (\nu\, \widetilde{m_2}'')(Q' \mid t \hookleftarrow_{\mathtt{H}} V_2 \mid C_2) \qquad (18)$$

with C_1 (resp., C_2) being the higher-order trigger process in the cases where $\ell = (\nu\, \widetilde{m})n!\langle V_1'\rangle$ (resp., $\check{\ell} = (\nu\, \widetilde{m}')n!\langle V_2'\rangle$) and $C_1 = C_2 = \mathbf{0}$ otherwise. From (17) and the definition of $\Re$ we can conclude that there exists a Δ_4 such that

$$\Gamma; \Delta_2' \vdash (\nu\, \widetilde{m_1})(Q \mid t \Lleftarrow_{\mathtt{C}} V_2 : U) \overset{\check{\ell}}{\Longmapsto} \Delta_4 \vdash (\nu\, \widetilde{m_2}')(Q' \mid t \Lleftarrow_{\mathtt{C}} V_2 : U)$$

Equation (18) then allows us to infer the required conclusion, for some Δ_3', Δ_4':

$$\Gamma; \Delta_3' \vdash (\nu\, \widetilde{m_1}''')(P' \mid t \Lleftarrow_{\mathtt{C}} V_1 : U \mid C_1)\ \Re\ \Delta_4' \vdash (\nu\, \widetilde{m_2}''')(Q' \mid t \Lleftarrow_{\mathtt{C}} V_2 : U \mid C_2)$$

(b) $t \Lleftarrow_{\mathtt{C}} V_1 : U$ moves autonomously, i.e., for some Δ_3 we have:

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t \Lleftarrow_{\mathtt{C}} V_1 : U) \\ \overset{t?\langle m\rangle}{\longmapsto} & \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(s?(y).\{\!\!\{U\}\!\!\}^y \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \end{array}$$

Following requirements of $\Re$ and the freshness of t we can infer that there exists a Δ_1'' such that

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1) \\ \overset{t?\langle \{\!\!\{U\}\!\!\}_{\mathsf{c}}\rangle}{\longmapsto} & \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(s?(y).\{\!\!\{U\}\!\!\}^y \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \end{array}$$

which implies, from the higher-order bisimilarity requirement of $\Re$ and the freshness of t, that there exist Q' and Δ_2'' such that

$$\begin{array}{rl} & \Gamma; \emptyset; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2) \\ \Longmapsto & (\nu\, \widetilde{m_2})(Q_2 \mid t \hookleftarrow_{\mathtt{H}} V_2) \\ \overset{t?\langle \{\!\!\{U\}\!\!\}_{\mathsf{c}}\rangle}{\longmapsto} & (\nu\, \widetilde{m_2})(Q_2 \mid (\nu\, s)(s?(y).\{\!\!\{U\}\!\!\}^y \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \\ \Longmapsto & \Delta_2'' \vdash Q' \end{array} \qquad (19)$$

and

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(s?(y).\{\!\!\{U\}\!\!\}^y \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \\ \approx^{\mathtt{H}} & \Delta_2'' \vdash (\nu\, \widetilde{m_2})Q' \end{array} \qquad (20)$$

The freshness of t allows us to mimic the transitions in (19); for some Δ_4 we obtain:

$$\begin{array}{rl} & \Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid t \Lleftarrow_{\mathtt{C}} V_2 : U) \\ \Longmapsto & (\nu\, \widetilde{m_2})(Q_2 \mid t \Lleftarrow_{\mathtt{C}} V_2 : U) \\ \overset{t?\langle m\rangle}{\longmapsto} & (\nu\, \widetilde{m_2})(Q_2 \mid (\nu\, s)(s?(y).\{\!\!\{U\}\!\!\}^y \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \\ \Longmapsto & \Delta_4 \vdash Q' \end{array}$$

The conclusion is immediate from (20).

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

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

$$\begin{aligned}\mathfrak{R}' = \{((\nu\,\widetilde{m_1})(P \mid t \hookleftarrow_{\mathtt{H}} V_1), (\nu\,\widetilde{m_2})(Q \mid t \hookleftarrow_{\mathtt{H}} V_2)) \ \mid \\ \Gamma; \Delta_1' \vdash (\nu\,\widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \approx^{\mathtt{C}} \Delta_2' \vdash (\nu\,\widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \ \}\end{aligned}$$

To prove that $\mathfrak{R}' \subseteq \approx^{\mathtt{C}}$ we first consider relation $\mathfrak{R}$ (for readability, we omit type information):

$$\begin{aligned}\mathfrak{R} = \{((\nu\,\widetilde{m_1})(P \mid t \leftharpoondown V_1), (\nu\,\widetilde{m_2})(Q \mid t \leftharpoondown V_2)) \ \mid \\ \Gamma; \Delta_1' \vdash (\nu\,\widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \approx^{\mathtt{C}} \Delta_2' \vdash (\nu\,\widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \ \}\end{aligned}$$

By proving that $\mathfrak{R} \subseteq \approx^{\mathtt{C}}$ we can apply Lemma 12 (Part 3), to obtain that $\mathfrak{R}' \subseteq \approx^{\mathtt{C}}$. Suppose that $(\nu\,\widetilde{m_1})(P \mid t \leftharpoondown V_1)$ moves; we must exhibit a matching move from $(\nu\,\widetilde{m_2})(Q \mid t \leftharpoondown V_2)$. We distinguish three cases, depending on the source/kind of visible action:

   (a) P moves autonomously, i.e., for some Δ_3 we have:

$$\Gamma; \Delta_1' \vdash (\nu\,\widetilde{m_1})(P \mid t \leftharpoondown V_1) \overset{\ell}{\longmapsto} \Delta_3 \vdash (\nu\,\widetilde{m_1}')(P' \mid t \leftharpoondown V_1)$$

   Then, following the requirements of $\mathfrak{R}$ and the freshness of t, we infer that there exists some Δ_1'' such that

$$\Gamma; \Delta_1 \vdash (\nu\,\widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \overset{\ell}{\longmapsto} \Delta_1'' \vdash (\nu\,\widetilde{m_1}')(P' \mid t \Leftarrow_{\mathtt{C}} V_1 : U)$$

   which implies, from the characteristic bisimilarity requirement of $\mathfrak{R}$ and the freshness of t, that there exist Q' and Δ_2'' such that

$$\Gamma; \Delta_2 \vdash (\nu\,\widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \overset{\breve{\ell}}{\Longmapsto} \Delta_2'' \vdash (\nu\,\widetilde{m_2}')(Q' \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \tag{21}$$

   and

$$\begin{aligned}\Gamma; \emptyset;\ \Delta_1''' \vdash (\nu\,\widetilde{m_1}'')(P' \mid t \Leftarrow_{\mathtt{C}} V_1 : U \mid C_1) \\ \approx^{\mathtt{C}}\ \Delta_2''' \vdash (\nu\,\widetilde{m_2}'')(Q' \mid t \Leftarrow_{\mathtt{C}} V_2 : U \mid C_2)\end{aligned} \tag{22}$$

   with C_1 (resp., C_2) being the characteristic trigger process in the cases where $\ell = (\nu\,\widetilde{m})n!\langle V_1' \rangle$ (resp., $\breve{\ell} = (\nu\,\widetilde{m}')n!\langle V_2' \rangle$) and $C_1 = C_2 = \mathbf{0}$ otherwise. From (21) we can infer that there exists Δ_4 such that

$$\Gamma; \Delta_2' \vdash (\nu\,\widetilde{m_1})(Q \mid t \leftharpoondown V_2) \overset{\breve{\ell}}{\Longmapsto} \Delta_4 \vdash (\nu\,\widetilde{m_2}')(Q' \mid t \leftharpoondown V_2)$$

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

$$\Gamma; \Delta_3' \vdash (\nu\,\widetilde{m_1}''')(P' \mid t \leftharpoondown V_1 \mid C_1) \ \mathfrak{R}\ \Delta_4' \vdash (\nu\,\widetilde{m_2}''')(Q' \mid t \leftharpoondown V_2 \mid C_2)$$

   (b) $t \leftharpoondown V_1$ moves autonomously, i.e., for some Δ_3 we have:

$$\Gamma; \Delta_1' \vdash (\nu\,\widetilde{m_1})(P \mid t \leftharpoondown V_1) \overset{t?\langle (\!\{?(U);\mathtt{end}\}\!)_{\mathtt{c}} \rangle}{\longmapsto} \Delta_3 \vdash (\nu\,\widetilde{m_1})(P \mid (\nu\,s)((\!\{?(U);\mathtt{end}\}\!)_{\mathtt{c}}\, s \mid \overline{s}!\langle V_1 \rangle.\mathbf{0}))$$

   Following requirements of $\mathfrak{R}$ and the freshness of t we infer that there is a Δ_1'' such that

$$\begin{aligned}&\Gamma; \emptyset; \Delta_1 \vdash (\nu\,\widetilde{m_1})(P \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \\ \overset{t?\langle m \rangle}{\longmapsto}\ &\Delta_1'' \vdash (\nu\,\widetilde{m_1})(P \mid (\nu\,s)(s?(y).(\!\{U\}\!)^{y} \mid \overline{s}!\langle V_1 \rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\Longmapsto}\ &\Delta_1'' \vdash (\nu\,\widetilde{m_1})P'\end{aligned}$$

   which implies from the characteristic bisimulation requirement of $\mathfrak{R}$ and the freshness of t that there exist Q' and Δ_2'' such that

$$\begin{aligned}&\Gamma; \emptyset; \Delta_2 \vdash (\nu\,\widetilde{m_2})(Q \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \\ \Longmapsto\ &(\nu\,\widetilde{m_2})(Q_2 \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \\ \overset{t?\langle m \rangle}{\longmapsto}\ &(\nu\,\widetilde{m_2})(Q_2 \mid (\nu\,s)(s?(y).(\!\{U\}\!)^{y} \mid \overline{s}!\langle V_2 \rangle.\mathbf{0})) \\ \Longmapsto\ &\Delta_2'' \vdash Q'\end{aligned} \tag{23}$$

and

$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})P' \approx^{\mathtt{C}} \Delta_2'' \vdash (\nu\, \widetilde{m_2})Q'$$

which implies from Lemma 12 (Part 2) that for fresh t'

$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P' \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0})) \approx^{\mathtt{C}} \Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q') \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}) \qquad (24)$$

The freshness of t allows us to mimic the transitions in (23) to infer that, for some Δ_4, we have

$$\begin{array}{rl} & \Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid t \leftharpoondown V_2) \\ \Longmapsto & (\nu\, \widetilde{m_2})(Q_2 \mid t \leftharpoondown V_2) \\ \overset{t?\langle \{\!\!\{ ?(U);\mathtt{end} \}\!\!\}_{\mathsf{c}}\rangle}{\longmapsto} & (\nu\, \widetilde{m_2})(Q_2 \mid (\nu\, s)(\{\!\!\{ ?(U);\mathtt{end} \}\!\!\}_{\mathsf{c}}\, s \mid \overline{s}!\langle V_2\rangle.\mathbf{0})) \\ \Longmapsto & \Delta_4 \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0})) \end{array}$$

and

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)(\{\!\!\{ ?(U);\mathtt{end} \}\!\!\}_{\mathsf{c}}\, s \mid \overline{s}!\langle V_1\rangle.\mathbf{0})) \overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1}')(P' \mid (\nu\, s)(t'!\langle s\rangle.\mathbf{0}))$$

The conclusion is immediate from (24).

(c) $t \leftharpoondown V_1$ moves autonomously, i.e., for some Δ_3 we have:

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t \leftharpoondown V_1) \\ \overset{t?\langle \lambda x.\, t'?(y).(y\, x)\rangle}{\longmapsto} & \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)((\lambda x.\, t'?(y).(y\, x))s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}\,)) \end{array}$$

We show that there exist Δ_4 and $(\nu\, \widetilde{m_1})(Q \mid (\nu\, s)((\lambda x.\, t'?(y).(y\, x))s \mid \overline{s}!\langle V_2\rangle.\mathbf{0}\,))$ such that

$$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_2' \vdash (\nu\, \widetilde{m_1})(Q \mid t \leftharpoondown V_2) \\ \overset{t?\langle \lambda x.\, t'?(y).(y\, x)\rangle}{\Longmapsto} & \Delta_4 \vdash (\nu\, \widetilde{m_1})(Q \mid t' \leftharpoondown V_2) \end{array}$$

and

$$\begin{array}{l} \Gamma; \emptyset;\ \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid (\nu\, s)((\lambda x.\, t'?(y).y\, x)s \mid \overline{s}!\langle V_1\rangle.\mathbf{0}\,)) \\ \overset{\tau_{\mathsf{d}}}{\Longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid t' \leftharpoondown V_1) \end{array}$$

The result

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid t' \leftharpoondown V_1)\ \Re\ \Delta_4 \vdash (\nu\, \widetilde{m_1})(Q \mid t' \leftharpoondown V_2)$$

is immediate from the definition of $\Re$.

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

$\square$

Lemma 14 $\approx^{\mathtt{H}} = \approx^{\mathtt{C}}$.

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

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

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

We show that $\Re$ is a characteristic bisimulation. Suppose $\Gamma; \Delta_1 \vdash P \overset{\ell}{\longmapsto} \Delta_1' \vdash P'$. We need to show that $\Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond$ can match ℓ. The proof proceeds by a case analysis on the transition label $\ell = (\nu\, \widetilde{m_1})n!\langle V_1\rangle$, which is the only non-trivial case.

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

$$\Gamma; \Delta_1 \vdash P \overset{(\nu\, \widetilde{m_1})n!\langle V_1\rangle}{\longmapsto} \Delta_1'' \vdash P' \qquad (25)$$

then there exist Δ_2'', Q, and V_2 such that:

$$\Gamma; \Delta_2 \vdash Q \overset{(\nu\, \widetilde{m_2})n!\langle V_2\rangle}{\Longmapsto} \Delta_2'' \vdash Q' \qquad (26)$$

and for fresh t:

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P' \mid t \hookleftarrow_{\mathtt{H}} V_1) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q' \mid t \hookleftarrow_{\mathtt{H}} V_2) \tag{27}$$

To show that $\Re$ is a characteristic bisimulation after the fact that transition (25) implies transition (26), we need to show that for fresh t and for some Δ_3, Δ_4:

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P' \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \;\Re\; \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q' \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \tag{28}$$

which follows from (27), Lemma 13(1), and the definition of $\Re$.

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

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

We show that $\Re$ is a higher-order bisimulation. Suppose $\Gamma; \Delta_1 \vdash P \overset{\ell}{\longmapsto} \Delta_1' \vdash P'$. We need to show that $\Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond$ can match action $\ell = (\nu\, \widetilde{m_1})n!\langle V_1\rangle$.

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

$$\Gamma; \Delta_1 \vdash P \overset{(\nu\, \widetilde{m_1})n!\langle V_1\rangle}{\longmapsto} \Delta_1'' \vdash P' \tag{29}$$

then there exist Δ_2'', Q, and V_2 such that:

$$\Gamma; \Delta_2 \vdash Q \overset{(\nu\, \widetilde{m_2})n!\langle V_2\rangle}{\Longmapsto} \Delta_2'' \vdash Q' \tag{30}$$

and for fresh t and some Δ_2':

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P' \mid t \Leftarrow_{\mathtt{C}} V_1 : U) \approx^{\mathtt{C}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q' \mid t \Leftarrow_{\mathtt{C}} V_2 : U) \tag{31}$$

To show that $\Re$ is a higher-order bisimulation after the fact that transition (29) implies transition (30), we need to show that for fresh t and some Δ_3, Δ_4:

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P' \mid t \hookleftarrow_{\mathtt{H}} V_1) \;\Re\; \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q' \mid t \hookleftarrow_{\mathtt{H}} V_2) \tag{32}$$

which follows from (31), Lemma 13(2), and the definition of $\Re$.

$\square$

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.*

1. *If $n_1 \neq n_2$ with $\Gamma; \emptyset; \Delta \vdash n_i \triangleright U$ with $U \neq \mathtt{end}$ and*

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda x.\, t?(y).(y\, x))\, n_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda x.\, t?(y).(y\, x))\, n_2)$$

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

2. *If $\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\!|U|\!)_{\mathtt{c}}\, n_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\!|U|\!)_{\mathtt{c}}\, n_2)$ then whenever*

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\!|U|\!)_{\mathtt{c}}\, n_1) \overset{\ell}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid (\lambda x.\, t?(y).(y\, x))\, n_1)$$

implies that there exist Δ_2', $(\nu\, \widetilde{m_2}')(Q' \mid (\lambda x.\, t?(y).(y\, x))\, n_2)$ such that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_1})(Q \mid (\!|U|\!)_{\mathtt{c}}\, n_2) \overset{\hat{\ell_2}}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\lambda x.\, t?(y).(y\, x))\, n_2)$$

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

3. *If $\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t!\langle n_1\rangle.\mathbf{0}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t!\langle n_2\rangle.\mathbf{0})$ then*

$$\Gamma; \Delta_1 \vdash (\nu\, m_1)(P \mid t?(x).(x\, n_1)) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, m_2)(Q \mid t?(x).(x\, n_2))$$

4. If n fresh and

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{{}^{n}\!/x\} \mid t!\langle n_1\rangle.\mathbf{0}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{{}^{n}\!/x\} \mid t!\langle m_1\rangle.\mathbf{0})$$

then

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{{}^{n_1}\!/x\}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{{}^{m_1}\!/x\})$$

Proof We analyse each part separately:

1. The proof for Part 1 is done by contradiction. Assume that $\overline{n_1} \notin \mathtt{fn}(P)$ and $\overline{n_2} \notin \mathtt{fn}(Q)$. Then the bisimulation requirement allows us to observe for some $U \neq \mathtt{end}$:

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda x.\, t?(y).(y\, x))\, n_1) \xmapsto{\tau_{\mathsf{d}}} \xmapsto{t?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle} \xmapsto{\ell} \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid t'!\langle n_1\rangle.\mathbf{0})$$

with $\mathtt{subj}(\ell) = n_1$, because of the characteristic process interaction, then from the freshness of t:

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda x.\, t?(y).(y\, x))\, n_2) \overset{t?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle}{\Longmapsto} \overset{\breve{\ell}}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q' \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2)$$

with $\mathtt{subj}(\breve{\ell}) = n_1$. But then in the former result we can observe an action on t' and on the latter we cannot, thus leading to a contradiction with respect to the bisimilarity assumption.

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

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_1})(Q \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2) \overset{\hat{\breve{\ell}}}{\not\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid (\lambda x.\, t?(y).(y\, x))\, n_2)$$

From the bisimilarity requirement we can observe

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_1})(Q \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2) \overset{\hat{\breve{\ell}}}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2)$$

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

$$\Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P' \mid (\lambda x.\, t?(y).(y\, x))\, n_1) \triangleright \diamond$$

that cannot be observed by process $\Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2)$—a contradiction.

3. For the proof of Part 3 we do a case analysis on the transitions for checking the bisimulation requirements. The interesting case is when, for some Δ_1'':

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t?(x).(x\, n_1)) \xmapsto{t?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle} \Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}'')(P \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_1)$$

From the freshness of t we can derive that, for some Δ_2''

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t?(x).(x\, n_2)) \overset{t?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle}{\Longmapsto} \Gamma; \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q'' \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2)$$

From the bisimulation requirement of the hypothesis we have that

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid t!\langle n_1\rangle.\mathbf{0}) \xmapsto{t!\langle n_1\rangle} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P)$$

implies

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid t!\langle n_2\rangle.\mathbf{0}) \overset{t!\langle n_2\rangle}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q')$$

for some Δ_1', Δ_2' and

$$\begin{aligned} \Gamma; \emptyset;\ & \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle n_1\rangle.\mathbf{0})) \\ \approx^{\mathtt{H}}\ & \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q' \mid t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle n_2\rangle.\mathbf{0})) \end{aligned}$$

Whenever

$$\begin{aligned} & \Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P \mid t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle n_1\rangle.\mathbf{0})) \\ \xmapsto{t?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle}\ & \Delta_1'' \vdash (\nu\, \widetilde{m_1}'')(P \mid (\nu\, s)(s?(y).(\llbracket U \rrbracket_{\mathsf{c}}\, y) \mid \overline{s}!\langle n_1\rangle.\mathbf{0})) \\ \xmapsto{\tau_{\mathsf{d}}}\ & \Delta_1'' \vdash (\nu\, \widetilde{m_1}'')(P \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_1) \end{aligned}$$

then

$$\begin{aligned} & \Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_2}')(Q' \mid t?(x).(\nu\, s)(s?(y).(x\, y) \mid \overline{s}!\langle n_2\rangle.\mathbf{0})) \\ \overset{t?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle}{\Longmapsto}\ & \Delta_2'' \vdash (\nu\, \widetilde{m_2}'')(Q''' \mid (\nu\, s)(s?(y).(\llbracket U \rrbracket_{\mathsf{c}}\, y) \mid \overline{s}!\langle n_2\rangle.\mathbf{0})) \\ \overset{\tau_{\mathsf{d}}}{\Longmapsto}\ & \Delta_2'' \vdash (\nu\, \widetilde{m_2}'')(Q'' \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_2) \end{aligned}$$

which concludes the case.

4. Part 4. Let $\Re$ be the typed relation

$$\begin{aligned}\Re = \{&\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{^{n_1}/_x\}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{^{m_1}/_x\}) \ \mid \\ &\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{^{n}/_x\} \mid t_1!\langle n_1\rangle.\mathbf{0}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{^{n}/_x\} \mid t_1!\langle m_1\rangle.\mathbf{0})\}\end{aligned}$$

Suppose that $(\nu\, \widetilde{m_1})(P\{^{n_1}/_x\})$ moves:

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{^{n_1}/_x\}) \overset{\ell}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P'\{^{n_1}/_x\})$$

We need to show a matching action from $(\nu\, \widetilde{m_2})(Q\{^{m_1}/_x\})$; we proceed to show that $\Re$ is a higher-order bisimulation by a case analysis on the subject of action ℓ.

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

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{^{n}/_x\}) \mid t!\langle n_1\rangle.\mathbf{0} \overset{\ell}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P'\{^{n}/_x\}) \mid t!\langle n_1\rangle.\mathbf{0}$$

  implies

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2}')(Q\{^{n}/_x\}) \mid t!\langle n_2\rangle.\mathbf{0} \overset{\breve{\ell}}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q'\{^{n}/_x\}) \mid t!\langle n_2\rangle.\mathbf{0}$$

  for some Δ_2' and

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_1}')(P'\{^{n}/_x\}) \mid t!\langle n_2\rangle.\mathbf{0} \mid C_1 \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q'\{^{n}/_x\}) \mid t!\langle n_2\rangle.\mathbf{0} \mid C_2$$

  with $C_1 = t \hookleftarrow_{\mathtt{H}} n_1$ and $C_2 = t \hookleftarrow_{\mathtt{H}} n_2$ if ℓ and $\breve{\ell}$ are output actions, $C_1 = \mathbf{0}$ and $C_2 = \mathbf{0}$ otherwise. From here we can imply that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_1})(Q\{^{n_2}/_x\}) \overset{\breve{\ell}}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q'\{^{n_2}/_x\})$$

  Furthermore, we can easily see that

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_1}')(P\{^{n_1}/_x\}) \mid C_1 \ \Re\ \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q'\{^{n_2}/_x\}) \mid C_2$$

- $\mathtt{subj}(\ell) = n_1$. We distinguish two subcases
  - $n_1 = n_2$. The case is similar as the previous case.
  - $n_1 \neq n_2$. From the premise and Part 1 of this lemma we get that $\overline{n_1} \in \mathtt{fn}(P)$ and $\overline{n_2} \in \mathtt{fn}(Q)$. The latter implies that this case is not possible, since no external action ℓ would be observed, because of the typed transition requirement.
- $\ell = \tau$. This implies the untyped transitions

$$\begin{aligned}(\nu\, \widetilde{m_1})(P\{^{n_1}/_x\}) &\overset{\ell_{11}'}{\longmapsto} (\nu\, \widetilde{m_{11}})(P_1\{^{n_1}/_x\}) &(33)\\ (\nu\, \widetilde{m_1})(P\{^{n_1}/_x\}) &\overset{\ell_{12}'}{\longmapsto} (\nu\, \widetilde{m_{12}})(P_2\{^{n_1}/_x\}) &(34)\\ \ell_{11}' &\asymp \ell_{12}' &(35)\end{aligned}$$

  We distinguish two cases:
  - $\mathtt{subj}(\ell_1') \neq n_1$. This case is similar with case 1 of this proof.
  - $\mathtt{subj}(\ell_1') = n_1$. First observe that

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{^{n}/_x\} \mid t!\langle n_1\rangle.\mathbf{0}) \overset{\ell_{11}''}{\longmapsto} \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P_1\{^{n}/_x\} \mid t!\langle n_1\rangle.\mathbf{0})$$

    for some Δ_1' with $\ell_{11}''\{^{n_1}/_n\} = \ell_{11}'$, which implies

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{^{n}/_x\} \mid t!\langle n_2\rangle.\mathbf{0}) \overset{\ell_{21}''}{\Longmapsto} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q_1\{^{n}/_x\} \mid t!\langle n_2\rangle.\mathbf{0})$$

    with $\ell_{21}''\{^{n_2}/_n\} = \ell_{21}'$, which in turn implies

$$(\nu\, \widetilde{m_2})(Q\{^{n_2}/_x\}) \overset{\ell_{22}'}{\longmapsto} (\nu\, \widetilde{m_{21}})(Q_1\{^{n_2}/_x\}) \qquad (36)$$

Also observe that for $U = \Delta(n_1)$

$$\begin{aligned} &\Gamma; \emptyset;\ \Delta_1 \vdash (\nu\, \widetilde{m_1})(P\{n/x\} \mid t!\langle n_1\rangle.\mathbf{0}) \\ \overset{t!\langle n_1\rangle}{\longmapsto}\overset{t?\langle (\!|U|\!)_{\mathsf{c}}\rangle}{\longmapsto}\overset{\tau_{\mathsf{d}}}{\Longmapsto}\ &\Delta_1'' \vdash (\nu\, \widetilde{m_1}'')(P\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_1/x\}) \end{aligned}$$

for some Δ_1'' which implies

$$\begin{aligned} &\Gamma; \emptyset;\ \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{n/x\} \mid t!\langle n_2\rangle.\mathbf{0}) \\ \overset{t!\langle n_2\rangle}{\Longmapsto}\overset{t?\langle (\!|U|\!)_{\mathsf{c}}\rangle}{\Longmapsto}\overset{\tau_{\mathsf{d}}}{\Longmapsto}\ &\Delta_2'' \vdash (\nu\, \widetilde{m_2}'')(Q'\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_2/x\}) \end{aligned}$$

for some Δ_2'' with

$$\begin{aligned} \Gamma; \emptyset;\ \Delta_1'' &\vdash (\nu\, \widetilde{m_1}'')(P\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_1/x\}) \\ \approx^{\mathtt{H}}\ \Delta_2'' &\vdash (\nu\, \widetilde{m_2}'')(Q'\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_2/x\}) \end{aligned}$$

From (34) we can see that, for some Δ_1'''

$$\begin{aligned} \Gamma; \emptyset;\ \Delta_1'' &\vdash (\nu\, \widetilde{m_1}'')(P\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_1/x\}) \\ \overset{\tau}{\longmapsto}\ \Delta_1''' &\vdash (\nu\, \widetilde{m_1}'')(P_2\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0}) \end{aligned}$$

which implies from Part 2 of this lemma

$$\begin{aligned} \Gamma; \emptyset;\ \Delta_2'' &\vdash (\nu\, \widetilde{m_2}'')(Q'\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_2/x\}) \\ \overset{\tau}{\Longmapsto}\ \Delta_2''' &\vdash (\nu\, \widetilde{m_2}'')(Q_2\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0}) \end{aligned} \tag{37}$$

for some Δ_2''' and

$$\begin{aligned} \Gamma; \emptyset;\ \Delta_1''' &\vdash (\nu\, \widetilde{m_1}'')(P_2\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0}) \\ \approx^{\mathtt{H}}\ \Delta_2''' &\vdash (\nu\, \widetilde{m_2}'')(Q_2\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0}) \end{aligned} \tag{38}$$

where (37) implies the untyped transition

$$(\nu\, \widetilde{m_2}'')(Q'\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_2/x\}) \overset{\ell_{22}''}{\Longmapsto} (\nu\, \widetilde{m_2}'')(Q_2\{n/x\} \mid (\!|U|\!)_{\mathsf{c}}\{n_2/x\})$$

and furthermore,

$$(\nu\, \widetilde{m_2}'')(Q'\{n_2/x\}) \overset{\ell_{22}'}{\Longmapsto} (\nu\, \widetilde{m_2}'')(Q_2\{n_2/x\})$$

with $\ell_{22}''\{n_2/n\} = \ell_{22}'$. From the last result and (36) we get

$$\begin{aligned} &\Gamma; \emptyset; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q\{n_2/x\}) \\ \overset{\tau}{\Longmapsto}\ &\Delta_2' \vdash (\nu\, \widetilde{m_2}'')(Q''\{n_2/x\}) \end{aligned}$$

Furthermore, from (38) we can get that, for some Δ_3,

$$\Gamma; \Delta_1''' \vdash (\nu\, \widetilde{m_1}'')(P_2\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0}) \overset{\ell_{12}''}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1}''')(P'\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0})$$

which implies

$$\Gamma; \Delta_2''' \vdash (\nu\, \widetilde{m_2}'')(Q_2\{n/x\} \mid t'!\langle n_2\rangle.\mathbf{0}) \overset{\ell_{22}''}{\longmapsto} \Delta_4 \vdash (\nu\, \widetilde{m_2}''')(Q''\{n/x\} \mid t'!\langle n_2\rangle.\mathbf{0})$$

and

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1}''')(P'\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0}) \approx^{\mathtt{H}} \Delta_4 \vdash (\nu\, \widetilde{m_2}''')(Q''\{n/x\} \mid t'!\langle n_1\rangle.\mathbf{0})$$

which in turn implies the required conclusion:

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}''')(P'\{n_1/x\}) \,\Re\, \Delta_2' \vdash (\nu\, \widetilde{m_2}''')(Q''\{n_2/x\})$$

□

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. Also, let $t_i, i \in I$ be a fresh name. If*

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, m_i)$$

then for all $\lambda\widetilde{x}.\, R$ there exist Δ_1', Δ_2' such that

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid (\lambda\widetilde{x}.\, R)\, \widetilde{n}) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid (\lambda\widetilde{x}.\, R)\, \widetilde{m}).$$

Proof We prove the result up-to the deterministic application transition that substitutes names n_i and m_i in process R, respectively. Let $\Re$ be the relation

$$\begin{aligned}\Re = \{&(\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid R\{\widetilde{n}/\widetilde{x}\})\ ,\ \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid R\{\widetilde{m}/\widetilde{x}\}))\ \mid\\ &\forall \lambda\widetilde{x}.\, R, \exists \Delta_1', \Delta_2'.\\ &\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, m_i)\\ &\}\end{aligned}$$

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.

1. Suppose an action from P, for some Δ_1'':

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid R\{\widetilde{n}/\widetilde{x}\}) \overset{\ell}{\longmapsto} \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P' \mid R\{\widetilde{n}/\widetilde{x}\})$$

This transition implies, for some Δ_3', the following:

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})P \overset{\ell}{\longmapsto} \Delta_3' \vdash (\nu\, \widetilde{m_1}')P'$$

which in turn implies, for some Δ_5:

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i) \overset{\ell}{\longmapsto} \Delta_5 \vdash (\nu\, \widetilde{m_1}')(P' \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i)$$

The latter implies, from the definition of $\approx^{\mathtt{H}}$ and the freshness of $t_i, \forall i \in I$, the following, for some Δ_6:

$$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, m_i) \overset{\breve{\ell}}{\Longmapsto} \Delta_6 \vdash (\nu\, \widetilde{m_2}')(Q' \mid \prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i)$$

and

$$\begin{aligned}\Gamma; \emptyset;\ \Delta_5 &\vdash (\nu\, \widetilde{m_1}')(P' \mid \textstyle\prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, n_i \mid C_1)\\ \approx^{\mathtt{H}}\ \Delta_6 &\vdash (\nu\, \widetilde{m_2}')(Q' \mid \textstyle\prod_{i\in I}(\lambda x.\, t_i?(y).(y\, x))\, m_i \mid C_2)\end{aligned} \tag{39}$$

where C_1, C_2 are the higher-order trigger processes if $\ell, \breve{\ell}$ are output actions and $C_1 = C_2 = \mathbf{0}$ otherwise. The former transition implies, for some Δ_4':

$$\Gamma; \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q) \overset{\breve{\ell}}{\Longmapsto} \Delta_4' \vdash (\nu\, \widetilde{m_2}')Q'$$

which in turn implies, for some Δ_2'':

$$\Gamma; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid R\{\widetilde{m}/\widetilde{x}\}) \overset{\breve{\ell}}{\Longmapsto} \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q' \mid \{\widetilde{m}/\widetilde{x}\})$$

Equation (39) and the definition of $\Re$ imply the desired conclusion for the case:

$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P' \mid R\{\widetilde{n}/\widetilde{x}\} \mid C_1) \approx^{\mathtt{H}} \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q' \mid R\{\widetilde{m}/\widetilde{x}\} \mid C_2)$$

2. Suppose an action from R:

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid R\{\widetilde{n}/\widetilde{x}\}) \overset{\ell}{\longmapsto} \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P \mid R'\{\widetilde{n}/\widetilde{x}\})$$

for some Δ_1''. We identify three sub-cases:

i. $\mathtt{subj}(\ell) \neq n_i$. The case is similar as above.

ii. $\mathtt{subj}(\ell) = n_k$ and $n_k = m_k$. From the definition of $\Re$ we get that

$$\begin{array}{l} \Gamma; \emptyset;\ \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I} t_i?(x).(x\, n_i)) \\ \overset{t_k?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle}{\longmapsto}\ \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, n_i) \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_k) \end{array}$$

for some Δ_3. This transition implies

$$\begin{array}{l} \Gamma; \emptyset;\ \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid \prod_{i\in I} t_i?(x).(x\, m_i)) \\ \overset{t_k?\langle \llbracket U \rrbracket_{\mathsf{c}}\rangle}{\Longmapsto}\ \Delta_4 \vdash (\nu\, \widetilde{m_1})(Q' \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, m_i) \mid \llbracket U \rrbracket^{x}\{m_k/x\}) \end{array}$$

and

$$\begin{array}{l} \Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid \displaystyle\prod_{i\in I\setminus\{k\}} t_i?(x).(x\, n_i) \mid \llbracket U \rrbracket_{\mathsf{c}}\, n_k) \\ \overset{\tau_\beta}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid \displaystyle\prod_{i\in I\setminus\{k\}} t_i?(x).(x\, n_i) \mid \llbracket U \rrbracket^{x}\{n_k/x\}) \\ \approx^{\mathtt{H}}\ \Delta_4 \vdash (\nu\, \widetilde{m_1})(Q' \mid \displaystyle\prod_{i\in I\setminus\{k\}} t_i?(x).(x\, m_i) \mid \llbracket U \rrbracket^{x}\{m_k/x\}) \end{array}$$

for some Δ_4. The proof follows Part 2 of Lemma 15 where, for some Δ_3':

$$\begin{array}{l} \Gamma; \emptyset;\ \Delta_3 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, n_i) \mid \llbracket U \rrbracket^{x}\{n_k/x\}) \\ \overset{\ell}{\longmapsto} \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, n_i) \mid t'!\langle n_k\rangle.\mathbf{0}) \end{array}$$

implies, for some Δ_4':

$$\begin{array}{l} \Gamma; \emptyset;\ \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q' \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, m_i) \mid \llbracket U \rrbracket^{x}\{m_k/x\}) \\ \overset{\ell}{\Longmapsto} \dots\ \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q'' \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, m_i) \mid t'!\langle m_k\rangle.\mathbf{0}) \end{array}$$

and furthermore, from Part 3 of Lemma 15

$$\begin{array}{l} \Gamma; \emptyset;\ \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, n_i) \mid t'?(y).(y\, n_k)) \\ \approx^{\mathtt{H}}\ \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q'' \mid \prod_{i\in I\setminus\{k\}} t_i?(x).(x\, m_i) \mid t'?(y).(y\, m_k)) \end{array}$$

that implies from the definition of $\Re$ that for all R such that $\{\widetilde{x}\} \subseteq \mathtt{fv}(R)$

$$\Gamma; \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P \mid R\{\widetilde{n}/\widetilde{x}\})\ \Re\ \Delta_4' \vdash (\nu\, \widetilde{m_2}')(Q'' \mid R\{\widetilde{m}/\widetilde{x}\})$$

The case concludes when we verify that, for some Δ_2'', we have:

$$\Gamma; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid R\{\widetilde{m}/x\}) \overset{\ell}{\Longmapsto} \Delta_2'' \vdash (\nu\, \widetilde{m_1}')(Q'' \mid R'\{\widetilde{m}/x\})$$

iii. $\mathtt{subj}(\ell) = n_k$ and $n_k \neq m_k$. This case is not possible. Lemma 15 implies that n_k is a session and $\overline{n_k} \in \mathtt{fn}(P)$. From the definition of typed transition (Definition 5) we get that we cannot observe ℓ on $R\{\widetilde{n}/\widetilde{x}\}$, because $\overline{n_k} \in \mathtt{fn}(P)$ and $(\Gamma; \emptyset; \Delta) \overset{\ell}{\not\longmapsto}$.

3. Suppose the interaction of P and R, for some Δ_1':

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid R\{\widetilde{n}/\widetilde{x}\}) \longmapsto \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P' \mid R'\{\widetilde{n}/\widetilde{x}\})$$

From the typed reduction definition (Definition 5) we get that

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})P \overset{\ell_1}{\longmapsto} \Delta_R \vdash (\nu\, \widetilde{m_1})P \tag{40}$$

$$\Gamma; \Delta_1' \vdash R\{\widetilde{n}/\widetilde{x}\} \overset{\ell_2}{\longmapsto} \Delta_R' \vdash R'\{\widetilde{n}/\widetilde{x}\} \tag{41}$$

$$\ell_1 \asymp \ell_2$$

We distinguish several subcases:

i. $\ell_1 = n_k?\langle V\rangle$ and $\ell_2 = (\nu\,\widetilde{m})(\overline{n_k}!\langle V\rangle)$. From the requirement of $\Re$ we get that there exists U such that, for some Δ_3:

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta_1 \vdash (\nu\,\widetilde{m_1})(P \mid \textstyle\prod_{i\in I} t_i?(x).(x\,n_i))\\ \overset{t?\langle \llparenthesis U\rrparenthesis_{\mathsf{c}}\rangle}{\longmapsto}\overset{\tau_{\mathsf{d}}}{\Longrightarrow}\ &\Delta_3 \vdash (\nu\,\widetilde{m_1}')(P \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,n_i) \mid \llparenthesis U\rrparenthesis^x\{{}^{n_k}\!/x\})\end{aligned}$$

which in turn implies, for some Δ_4, that

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta_2 \vdash (\nu\,\widetilde{m_2})(Q \mid \textstyle\prod_{i\in I} t_i?(x).(x\,m_i))\\ \overset{t?\langle \llparenthesis U\rrparenthesis_{\mathsf{c}}\rangle}{\Longmapsto}\overset{\tau_{\mathsf{d}}}{\Longrightarrow}\ &\Delta_4 \vdash (\nu\,\widetilde{m_2}')(Q' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,m_i) \mid \llparenthesis U\rrparenthesis^x\{{}^{m_k}\!/x\})\end{aligned}$$

and

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta_3 \vdash (\nu\,\widetilde{m_1}')(P \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,n_i) \mid \llparenthesis U\rrparenthesis^x\{{}^{n_k}\!/x\})\\ \approx^{\mathtt{H}}\ &\Delta_4 \vdash (\nu\,\widetilde{m_2}')(Q' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,m_i) \mid \llparenthesis U\rrparenthesis^x\{{}^{m_k}\!/x\})\end{aligned}$$

From Part 2 of Lemma 15 we obtain that if, for some Δ_3',

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta_3 \vdash (\nu\,\widetilde{m_1}')(P \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,n_i) \mid \llparenthesis U\rrparenthesis^x\{{}^{n_k}\!/x\})\\ \overset{\tau}{\longmapsto}\ &\Delta_3' \vdash (\nu\,\widetilde{m_1}'')(P' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,n_i) \mid t_k'?(x).(x\,n_k))\end{aligned}$$

then

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta_4 \vdash (\nu\,\widetilde{m_2}')(Q' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,m_i) \mid \llparenthesis U\rrparenthesis^x\{{}^{m_k}\!/x\})\\ \overset{\tau}{\Longmapsto}\ &\Delta_4' \vdash (\nu\,\widetilde{m_2}'')(Q'' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,m_i) \mid t_k'?(x).(x\,m_k))\end{aligned}$$

and

$$\begin{aligned}&\Gamma;\emptyset;\ \Delta_3' \vdash (\nu\,\widetilde{m_1}'')(P' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,n_i) \mid t_k'?(x).(x\,n_k))\\ \approx^{\mathtt{H}}\ &\Delta_4' \vdash (\nu\,\widetilde{m_2}'')(Q'' \mid \textstyle\prod_{i\in I\backslash k} t_i?(x).(x\,m_i) \mid t_k'?(x).(x\,m_k))\end{aligned} \tag{42}$$

for some Δ_4'. This means that there exists a U' (cf. Definition 13)

$$(\nu\,\widetilde{m_1}')(P \mid \prod_{i\in I\backslash k} t_i?(x).(x\,n_i)) \overset{n_k?\langle \llparenthesis U'\rrparenthesis_{\mathsf{c}}\rangle}{\longmapsto} (\nu\,\widetilde{m_3})(P'\{{}^{\llparenthesis U'\rrparenthesis_{\mathsf{c}}}\!/x\} \mid \prod_{i\in I\backslash k} t_i?(x).(x\,n_i))$$

$$\llparenthesis U\rrparenthesis^x\{{}^{n_k}\!/x\} \overset{n_k!\langle \llparenthesis U'\rrparenthesis_{\mathsf{c}}\rangle}{\longmapsto} t_k'!\langle n_k\rangle.\mathbf{0}$$

$$(\nu\,\widetilde{m_2}'')(Q' \mid \prod_{i\in I\backslash k} t_i?(x).(x\,m_i)) \overset{m_k?\langle \llparenthesis U'\rrparenthesis_{\mathsf{c}}\rangle}{\Longmapsto} (\nu\,\widetilde{m_4})(Q''\{{}^{\llparenthesis U'\rrparenthesis_{\mathsf{c}}}\!/x\} \mid \prod_{i\in I\backslash k} t_i?(x).(x\,m_i))$$

$$\llparenthesis U\rrparenthesis^x\{{}^{m_k}\!/x\} \overset{m_k!\langle \llparenthesis U'\rrparenthesis_{\mathsf{c}}\rangle}{\longmapsto} t_k'!\langle m_k\rangle.\mathbf{0}$$

But then we can generalise the above result to match actions in (40) and (41):

$$(\nu\,\widetilde{m_2}'')(Q' \mid \prod_{i\in I\backslash k} t_i?(x).(x\,m_i)) \overset{m_k?\langle V\rangle}{\Longmapsto} (\nu\,\widetilde{m_4})(Q''\{{}^{V}\!/x\} \mid \prod_{i\in I\backslash k} t_i?(x).(x\,m_i))$$

$$R\{{}^{\widetilde{m}}\!/\widetilde{x}\} \overset{m_k!\langle V\rangle}{\longmapsto} R'\{{}^{\widetilde{n}}\!/\widetilde{x}\} \qquad m_k \in \widetilde{m}$$

to obtain, for some Δ_2'', that

$$\Gamma; \Delta_2' \vdash (\nu\,\widetilde{m_2})(Q \mid R\{{}^{\widetilde{n}}\!/\widetilde{x}\}) \Longmapsto \Delta_2'' \vdash (\nu\,\widetilde{m_2}')(Q''\{{}^{V}\!/x\} \mid R'\{{}^{\widetilde{m}}\!/\widetilde{x}\})$$

Furthermore the definition of $\Re$ and (42) allow us to conclude the case with the implication that

$$\Gamma; \Delta_1'' \vdash (\nu\,\widetilde{m_1}')(P'\{{}^{V}\!/x\} \mid R'\{{}^{\widetilde{n}}\!/\widetilde{x}\})\ \Re\ \Delta_2'' \vdash (\nu\,\widetilde{m_2}')(Q''\{{}^{V}\!/x\} \mid R'\{{}^{\widetilde{m}}\!/\widetilde{x}\})$$

ii. An important sub-case is when $\ell_1 = n?\langle n_k\rangle$ and $\ell_2 = n!\langle n_k\rangle$. From the definition of $\Re$ we have that

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P \mid \prod_{i\in I} t_1?(x).(x\, n_i)) \overset{n?\langle m\rangle}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1})(P'\{m/x\} \mid \prod_{i\in I} t_1?(x).(x\, n_i))$$

for some Δ_3. This transition implies, for some Δ_4, that

$$\begin{array}{l}\Gamma; \emptyset; \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid \prod_{i\in I} t_1?(x).(x\, n_i)) \\ \overset{n?\langle m\rangle}{\Longmapsto} \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q'\{m/x\} \mid \prod_{i\in I} t_1?(x).(x\, n_i))\end{array} \tag{43}$$

and

$$\Gamma; \Delta_3 \vdash (\nu\, \widetilde{m_1})(P'\{m/x\} \mid \prod_{i\in I} t_1?(x).(x\, n_i)) \approx^{\mathtt{H}} \Delta_4 \vdash (\nu\, \widetilde{m_2})(Q'\{m/x\} \mid \prod_{i\in I} t_1?(x).(x\, n_i))$$

We infer from Part 4 of Lemma 15 that

$$\begin{array}{l}\Gamma; \emptyset; \Delta'_3 \vdash (\nu\, \widetilde{m_1})(P'\{n_k/x\} \mid \prod_{i\in I\setminus\{k\}} t_1?(x).(x\, n_i)) \\ \approx^{\mathtt{H}} \Delta'_4 \vdash (\nu\, \widetilde{m_2})(Q'\{m_k/x\} \mid \prod_{i\in I\setminus\{k\}} t_1?(x).(x\, n_i))\end{array}$$

which implies from the definition of $\Re$ that

$$\begin{array}{l}\Gamma; \emptyset; \Delta'_1 \vdash (\nu\, \widetilde{m_1})(P'\{n_k/x\} \mid R\{\widetilde{m}/\widetilde{x}\}) \\ \approx^{\mathtt{H}} \Delta'_2 \vdash (\nu\, \widetilde{m_2})(Q'\{m_k/x\} \mid R'\{\widetilde{m}/\widetilde{x}\})\end{array}$$

From (43) and (41) we obtain, for some Δ''_2, the following

$$\Gamma; \Delta'_2 \vdash (\nu\, \widetilde{m_2})(Q \mid R\{\widetilde{m}/\widetilde{x}\}) \Longmapsto \Delta''_2 \vdash (\nu\, \widetilde{m_2})(Q'\{m_k/x\} \mid R'\{\widetilde{m}/\widetilde{x}\})$$

which concludes the case.

iii. The most demanding sub-case is the case where $\ell_1 = n_k?\langle n_l\rangle$ and $\ell_2 = n_k!\langle n_l\rangle$. The proof is a consequence of the previews two sub-cases.

iv. The rest of the sub-cases are similar (or easier) to the above cases.

$\square$

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 $z \in \mathtt{fv}(P_1)$ and $z \in \mathtt{fv}(P_2)$. Also, let t be a fresh name. If*

$$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P_1\{\lambda x.\, t?(y).(y\, x)/z\}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(P_2\{\lambda x.\, t?(y).(y\, x)/z\})$$

then for all $\lambda x.\, R$ there exist Δ'_1 and Δ'_2 such that

$$\Gamma; \Delta'_1 \vdash (\nu\, \widetilde{m_1})(P_1\{\lambda x.\, R/z\}) \approx^{\mathtt{H}} \Delta'_2 \vdash (\nu\, \widetilde{m_2})(P_2\{\lambda x.\, R/z\})$$

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

$$\begin{array}{rl}\Re = \{((\nu\, \widetilde{m_1})(P_1\{\lambda x.\, R/z\}), (\nu\, \widetilde{m_2})(P_2\{\lambda x.\, R/z\})) & \mid \\ \quad \Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P_1\{\lambda x.\, t?(y).(y\, x)/z\}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2})(P_2\{\lambda x.\, t?(y).(y\, x)/z\}) & \}\end{array}$$

We show that $\Re$ is a higher-order bisimulation. Suppose that

$$\Gamma; \Delta'_1 \vdash (\nu\, \widetilde{m_1})(P_1\{\lambda x.\, R/z\}) \overset{\ell}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1})(P'_1\{\lambda x.\, R/z\}) \tag{44}$$

for some Δ_3. We should exhibit an appropriate matching action from $(\nu\, \widetilde{m_2})(P_2\{\lambda x.\, R/z\})$. Our analysis distinguishes two cases, depending on whether the substitution $\{\lambda x.\, R/z\}$ has an effect on the action denoted by ℓ:

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 $\mathtt{subj}(\ell) \neq t$. Furthermore, from the requirements of $\Re$ we get that there exist Δ_1'' and P_1' such that

   $$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1})(P_1\{\lambda x.\, t?(y).(y\, x)/z\}) \overset{\ell}{\longmapsto} \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_1'\{\lambda x.\, t?(y).(y\, x)/z\})$$

   which, in turn, implies that there exist Δ_2'' and P_2' such that

   $$\Gamma; \Delta_2 \vdash (\nu\, \widetilde{m_2}')(P_2\{\lambda x.\, t?(y).(y\, x)/z\}) \overset{\breve{\ell}}{\Longmapsto} \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(P_2'\{\lambda x.\, t?(y).(y\, x)/z\}) \quad (45)$$

   and

   $$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1}'')(P_1'\{\lambda x.\, t?(y).(y\, x)/z\} \mid C_1) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2}'')(P_2'\{\lambda x.\, t?(y).(y\, x)/z\} \mid C_2)$$

   with C_1 (resp., C_2) being the higher-order trigger process in the cases where $\ell = (\nu\, \widetilde{m})n!\langle V_1\rangle$ (resp., $\breve{\ell} = (\nu\, \widetilde{m}')n!\langle V_2\rangle$) and $C_1 = C_2 = \mathbf{0}$ otherwise. Because C_1 and C_2 are closed terms we can rewrite the substitution as:

   $$\Gamma; \Delta_1 \vdash (\nu\, \widetilde{m_1}'')((P_1' \mid C_1)\{\lambda x.\, t?(y).(y\, x)/z\}) \approx^{\mathtt{H}} \Delta_2 \vdash (\nu\, \widetilde{m_2}'')((P_2' \mid C_2)\{\lambda x.\, t?(y).(y\, x)/z\})$$

   Since $\ell, \breve{\ell}$ do not act on the substitution, we can consider any $\lambda x.\, R$ instead of $\lambda x.\, t?(y).(y\, x)$. Thus we further imply that

   $$\Gamma; \Delta_3' \vdash (\nu\, \widetilde{m_1}'')((P_1' \mid C_1)\{\lambda x.\, R/z\}) \;\Re\; \Delta_4' \vdash (\nu\, \widetilde{m_2}'')((P_2' \mid C_2)\{\lambda x.\, R/z\}) \quad (46)$$

   From (45) we can derive the transition

   $$\Gamma; \Delta_2' \vdash (\nu\, \widetilde{m_2})(P_2\{\lambda x.\, R/z\}) \overset{\breve{\ell}}{\Longmapsto} \Delta_4 \vdash (\nu\, \widetilde{m_2}')(P_2'\{\lambda x.\, R/z\})$$

   Equation (46) concludes the case.
2. Case $P_1 \equiv P \mid \prod_{i\in I} z\, n_i \mid z\, n_1$ such that $P \neq P' \mid z\, n'$. This is the case where action ℓ 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 ℓ:
   - Consider the following transition, for some Δ_3:

     $$\begin{array}{l}\Gamma; \emptyset;\; \Delta_1' \vdash (\nu\, \widetilde{m_1})((P \mid \prod_{i\in I} z\, n_i \mid z\, n_1)\{\lambda x.\, R/z\}) \\ \overset{\ell}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1})((P' \mid \prod_{i\in I} z\, n_i \mid z\, n_1)\{\lambda x.\, R/z\})\end{array}$$

     This sub-case is similar as the previous case.
   - Consider the following transition, for some Δ_3, and assuming that $Q = P \mid \prod_{i\in I} z\, n_i$:

     $$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})((Q \mid z\, n_1)\{\lambda x.\, R/z\}) \overset{\tau}{\longmapsto} \Delta_3 \vdash (\nu\, \widetilde{m_1})(Q\{\lambda z.\, R/z\} \mid R\{n_1/z\}) \quad (47)$$

     which is the application of name n_1 on abstraction $\lambda x.\, R$. From the requirements of $\Re$ we imply that

     $$\begin{array}{l}\Gamma; \emptyset;\; \Delta_1 \vdash (\nu\, \widetilde{m_1})((Q \mid z\, n_1)\{\lambda x.\, t?(y).(y\, x)/z\}) \\ \overset{\tau}{\longmapsto} \Delta_1'' \vdash (\nu\, \widetilde{m_1})(Q\{\lambda x.\, t?(y).(y\, x)/z\} \mid t?(y).(y\, n_1))\end{array}$$

     for some Δ_1''. Thus implies that there exist P_2' and Δ_2'' such that

     $$\begin{array}{l}\Gamma; \emptyset;\; \Delta_2 \vdash (\nu\, \widetilde{m_2})(P_2\{\lambda x.\, t?(y).(y\, x)/z\}) \\ \Longmapsto \Delta_2'' \vdash (\nu\, \widetilde{m_2})(P_2'\{\lambda x.\, t?(y).(y\, x)/z\})\end{array} \quad (48)$$

     and

     $$\begin{array}{l}\Gamma; \emptyset;\; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(Q\{\lambda x.\, t?(y).(y\, x)/z\} \mid t?(y).(y\, n_1)) \\ \approx^{\mathtt{H}} \Delta_2'' \vdash (\nu\, \widetilde{m_2})(P_2'\{\lambda x.\, t?(y).(y\, x)/z\})\end{array}$$

From the last pair we can see that for fresh t' if

$$\begin{aligned} &\Gamma; \emptyset;\ \Delta_1'' \vdash (\nu\, \widetilde{m_1})(Q\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid t?(y).(y\, n_1)) \\ \overset{t?\langle \lambda x.\, t'?(y).(y\, x)\rangle}{\longmapsto}\ &\Delta_1''' \vdash (\nu\, \widetilde{m_2})(Q\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, t'?(y).(y\, x))\, n_1) \end{aligned}$$

this implies that there exist P_2'', Δ_2''' such that

$$\begin{aligned} &\Gamma; \emptyset; \Delta_2'' \vdash (\nu\, \widetilde{m_2})(P_2'\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\}) \\ \Longmapsto\ & (\nu\, m_2)((P_3 \mid z\, n_2)\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\}) \\ \overset{t?\langle \lambda x.\, t'?(y).(y\, x)\rangle}{\longmapsto}\ & (\nu\, m_2)(P_3\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, t'?(y).(y\, x))\, n_2) \\ \Longmapsto\ & \Delta_2''' \vdash (\nu\, \widetilde{m_2})(P_2''\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, t'?(y).(y\, x))\, n_2) \end{aligned} \tag{49}$$

and

$$\begin{aligned} &\Gamma; \emptyset;\ \Delta_1''' \vdash (\nu\, \widetilde{m_1})(Q\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, t'?(y).(y\, x))\, n_1) \\ \approx^{\mathtt{H}}\ &\Delta_2''' \vdash (\nu\, \widetilde{m_2})(P_2''\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, t'?(y).(y\, x))\, n_2) \end{aligned}$$

From Lemma 16 we can deduce that, for all $\lambda x.\, R$, if there exist Δ_5 and Δ_6 then

$$\begin{aligned} &\Gamma; \emptyset;\ \Delta_5 \vdash (\nu\, \widetilde{m_1})(Q\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, R)\, n_1) \\ \approx^{\mathtt{H}}\ &\Delta_6 \vdash (\nu\, \widetilde{m_2})(P_2''\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\} \mid (\lambda x.\, R)\, n_2) \end{aligned}$$

from the definition of $\Re$ we have that for all $\lambda x.\, R$, if there exist Δ_3 and Δ_4

$$\begin{aligned} &\Gamma; \emptyset;\ \Delta_3 \vdash (\nu\, \widetilde{m_1})(Q\{{}^{(\lambda x.\, R)}/_{z}\} \mid (\lambda x.\, R)\, n_1) \\ \Re\ \ &\Delta_4 \vdash (\nu\, \widetilde{m_2})(P_2''\{{}^{(\lambda x.\, R)}/_{z}\} \mid (\lambda x.\, R)\, n_2) \end{aligned} \tag{50}$$

We show that we can mimic first the transition in (48) and then the silent part of transitions (49) to get:

$$\begin{aligned} &\Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2})(P_2\{{}^{(\lambda x.\, R)}/_{z}\}) \\ \Longmapsto\ &\Delta_2' \vdash (\nu\, \widetilde{m_2})(P_2'\{{}^{\lambda x.\, t?(y).(y\, x)}/_{z}\}) \\ \Longmapsto\ &\Delta_4 \vdash (\nu\, m_2)(P_2''\{{}^{(\lambda x.\, R)}/_{z}\} \mid (\lambda x.\, R)\, n_2) \end{aligned} \tag{51}$$

We showed that if (45) then (51) and (50) as required to show that $\Re$ is a higher-order bisimulation. $\square$

Lemma 18 $\approx^{\mathtt{H}} \subseteq \approx$.

Proof Let $\Re$ be the typed relation (for readability, we omit typing information):

$$\Re = \{(P_1, Q_1) \ \mid\ \Gamma; \Delta_1 \vdash P_1 \approx^{\mathtt{H}} \Delta_2 \vdash Q_1\}$$

We show that $\Re$ is a context bisimulation (cf. Definition 12 (Page 17)). Suppose that

$$\Gamma; \Delta_1 \vdash P_1 \overset{\ell}{\longmapsto} \Delta_1' \vdash P_2 \tag{52}$$

We need to infer an appropriate matching transition from Q_1. The proof proceeds by a case analysis on ℓ. We distinguish four cases: ℓ is not an output or a higher-order input action; ℓ is a higher-order input action; ℓ is an higher-order output; ℓ is a first-order output.

1. Case $\ell \notin \{(\nu\, \widetilde{m_1})n!\langle \lambda \widetilde{x}.\, P\rangle, (\nu\, \widetilde{m_1}')n!\langle \widetilde{m_1}\rangle, n?\langle \lambda \widetilde{x}.\, P\rangle\}$: We first notice that in this case the definition of $\approx$ and $\approx^{\mathtt{H}}$ coincide, so we have to show the existence of Q_2 and Δ_2' such that:

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

   and

$$\Gamma; \Delta_1' \vdash P_2 \,\Re\, \Delta_2' \vdash Q_2.$$

   This is immediate from transition (52) and the definition of $\approx^{\mathtt{H}}$ (cf. Definition 17 (Page 21)).

2. Case $\ell = n?\langle \lambda \widetilde{x}.\, P\rangle$: In this case, the transition (52) can be written as

   $$\Gamma; \Delta_1 \vdash P_1 \overset{n?\langle \lambda \widetilde{z}.\, t?(y).(y\, \widetilde{z})\rangle}{\longmapsto} \Delta_1'' \vdash P_2\{^{\lambda \widetilde{z}.\, t?(y).(y\, \widetilde{z})}/_{x}\}$$

   for some Δ_1''. In turn, the above transition and $\Re$ imply the existence of Q_2 and Δ_2'' such that:

   $$\Gamma; \Delta_2 \vdash Q_1 \overset{n?\langle \lambda \widetilde{z}.\, t?(y).(y\, \widetilde{z})\rangle}{\Longmapsto} \Delta_2'' \vdash Q_2\{^{\lambda \widetilde{z}.\, t?(y).(y\, \widetilde{z})}/_{x}\}$$

   and

   $$\Gamma; \Delta_1'' \vdash P_2\{^{\lambda \widetilde{z}.\, t?(y).(y\, \widetilde{z})}/_{x}\} \approx^{\mathtt{H}} \Delta_2'' \vdash Q_2\{^{\lambda \widetilde{z}.\, t?(y).(y\, \widetilde{z})}/_{x}\}.$$

   Then, by using the previous equality and Lemma 17 (Page 54), we may conclude that

   $$\Gamma; \Delta_1' \vdash P_2\{^{\lambda \widetilde{x}.\, P}/_{x}\} \approx^{\mathtt{H}} \Delta_2' \vdash Q_2\{^{\lambda \widetilde{x}.\, P}/_{x}\}$$

   for some Δ_1', Δ_2', for all P with $\mathtt{fv}(P) = \{\widetilde{x}\}$, as required.
3. Case $\ell = (\nu\, \widetilde{m_1}')n!\langle \widetilde{m_1}\rangle$: In this case, transition (52) and $\Re$ imply the existence of Δ_2', a process Q_2, and name m_2 such that

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

   and

   $$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid t \hookleftarrow_{\mathtt{H}} m_1) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid t \hookleftarrow_{\mathtt{H}} m_2)$$

   for some fresh t. From Case 2 (higher-order input) of this proof we can conclude that for all R with $\mathtt{fv}(R) = \{x\}$ and for some Δ_1'', Δ_2'':

   $$\begin{array}{rcl} \Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid t \hookleftarrow_{\mathtt{H}} m_1) & \xrightarrow{t?\langle \lambda z.\, z?(x).R\rangle} & (\nu\, \widetilde{m_1}')(P_2 \mid (\nu\, s)(s?(x).R \mid \overline{s}!\langle m_1\rangle.\mathbf{0})) \\ & \xrightarrow{\tau_{\mathsf{d}}} & \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid R\{^{m_1}/_{x}\}) \end{array}$$

   and

   $$\begin{array}{rcl} \Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid t \hookleftarrow_{\mathtt{H}} m_2) & \xrightarrow{t?\langle \lambda z.\, z?(x).R\rangle} & (\nu\, \widetilde{m_2}')(Q_2 \mid (\nu\, s)(s?(x).R \mid \overline{s}!\langle m_2\rangle.\mathbf{0})) \\ & \xrightarrow{\tau_{\mathsf{d}}} & \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid R\{^{m_2}/_{x}\}) \end{array}$$

   where, due to the deterministic internal transitions (cf. Definition 19), it is easy to see that

   $$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid R\{^{m_1}/_{x}\}) \approx^{\mathtt{H}} \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid R\{^{m_2}/_{x}\})$$

   for all R with $\mathtt{fv}(R) = \{x\}$, as required by the definition of $\approx$ (Definition 12 (Page 17)).
4. Case $\ell = (\nu\, \widetilde{m_1}')n!\langle \lambda \widetilde{x}.\, P\rangle$: This case is similar to the previous case but makes use of the alternative trigger, $t \leftharpoondown V$ (cf. (15)). The definition of $\Re$ and transition (52) allow us to infer the existence of some Δ_2', Q, and Q_2 such that

   $$\Gamma; \Delta_2 \vdash Q_1 \overset{(\nu\, \widetilde{m_2}')n!\langle \lambda \widetilde{x}.\, Q\rangle}{\Longmapsto} \Delta_2' \vdash Q_2$$

   and

   $$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid t \hookleftarrow_{\mathtt{H}} \lambda \widetilde{x}.\, P) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid t \hookleftarrow_{\mathtt{H}} \lambda \widetilde{x}.\, Q)$$

   for some fresh t. Using Lemma 12 (Page 39), we above equality implies that

   $$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid t \leftharpoondown \lambda \widetilde{x}.\, P) \approx^{\mathtt{H}} \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid t \leftharpoondown \lambda \widetilde{x}.\, Q)$$

   which in turn implies

   $$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_1' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid t \leftharpoondown \lambda \widetilde{x}.\, P) \\ \overset{t?\langle \lambda y.\, t'?(x).(x\, y)\rangle}{\longmapsto} & \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid (\nu\, s)((\lambda y.\, t'?(x).(x\, y))s \mid \overline{s}!\langle \lambda \widetilde{x}.\, P\rangle.\mathbf{0})) \end{array}$$

   for some Δ_1' and for some Δ_2'':

   $$\begin{array}{rl} & \Gamma; \emptyset;\ \Delta_2' \vdash (\nu\, \widetilde{m_2}')(Q_2 \mid t \leftharpoondown \lambda \widetilde{x}.\, Q) \\ \overset{t?\langle \lambda y.\, t'?(x).(x\, y)\rangle}{\Longmapsto} & \Delta_2'' \vdash (\nu\, \widetilde{m_1}')(Q_2' \mid (\nu\, s)((\lambda y.\, t'?(x).(x\, y))s \mid \overline{s}!\langle \lambda \widetilde{x}.\, Q\rangle.\mathbf{0})) \end{array}$$

and

$$\begin{array}{l}\Gamma;\emptyset;\ \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid (\nu\, s)((\lambda y.\, t'?(x).(x\, y))s \ \mid \overline{s}!\langle \lambda\widetilde{x}.\, P\rangle.\mathbf{0})) \\ \quad \approx^{\mathtt{H}} \Delta_2'' \vdash (\nu\, \widetilde{m_1}')(Q_2' \mid (\nu\, s)((\lambda y.\, t'?(x).(x\, y))s \ \mid \overline{s}!\langle \lambda\widetilde{x}.\, Q\rangle.\mathbf{0}))\end{array}.$$

From the Case 2 of this proof (higher-order input), we have

$$\begin{array}{l}\Gamma;\emptyset;\ \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_2 \mid (\nu\, s)((\lambda y.\, y?(x).R)\, s \mid \overline{s}!\langle \lambda\widetilde{x}.\, P\rangle.\mathbf{0})) \\ \quad \approx^{\mathtt{H}} \Delta_2'' \vdash (\nu\, \widetilde{m_1}')(Q_2' \mid (\nu\, s)((\lambda y.\, y?(x).R\,)s \mid \overline{s}!\langle \lambda\widetilde{x}.\, Q\rangle.\mathbf{0}))\end{array}$$

for all R with $\mathtt{fv}(R) = \{x\}$. Now, using deterministic transitions (cf. Definition 19) is easy to see that

$$\Gamma;\Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_2 \mid R\{^{\lambda\widetilde{x}.\, P}/y\}) \approx^{\mathtt{H}} \Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q_2 \mid R\{^{\lambda\widetilde{x}.\, Q}/y\})$$

for all R with $\mathtt{fv}(R) = \{x\}$, as required by the definition of $\approx$ (cf. Definition 12).

5. Case $\ell = (\nu\, \widetilde{m_1}')n!\langle\widetilde{m_1}\rangle$: This case is similar to the previous one.
 □

Lemma 19 $\approx\subseteq\cong$.

Proof We prove that $\approx$ (cf. Definition 12 (Page 17)) satisfies the three defining properties of $\cong$: reduction closure, barb preservation, congruence (cf. Definition 11 (Page 17)).

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 Δ_2' and P_2' such that

$$\Gamma;\Delta_2 \vdash P_2 \Longrightarrow \Delta_2' \vdash P_2' \qquad \text{and} \qquad \Gamma;\Delta_1 \vdash P_1' \approx \Delta_2' \vdash P_2'$$

The same arguments hold for the symmetric case, thus $\approx$ is reduction-closed.

II. **Barb Preservation:** Following Definition 9 (Page 16), we have that $\Gamma;\emptyset;\Delta_1 \vdash P_1 \downarrow_n$ implies

$$P \cong (\nu\, \widetilde{m})(n!\langle V_1\rangle.P_3 \mid P_4)$$

with $\overline{n} \notin \Delta_1$. From the definition of $\approx$ we infer that

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

implies the existence of Δ_2', V_2, and P_2' such that

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

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; we construct an typed relation defined as

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

We show that $\mathcal{S}$ is a context bisimulation. Suppose that

$$\Gamma;\Delta_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_1})(P_1 \mid R) \xrightarrow{\ell} \Delta_1' \cdot \Delta_3 \vdash P'$$

for some Δ_1'. We must show an appropriate matching action from $(\nu\, \widetilde{n_2})(P_2 \mid R)$. We proceed by a case analysis on the “source” of action ℓ (i.e., P_1, R, an interaction between P_1 and R).

1. Suppose that ℓ originates in P_1:

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_1})(P_1 \mid R) \xrightarrow{\ell} \Delta'_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{n'_1})(P'_1 \mid R)$$

The case is divided into three sub-cases, depending on the shape of ℓ:

i. Sub-case $\ell \notin \{(\nu\, \widetilde{m})n!\langle\lambda\widetilde{x}.\, Q\rangle, (\nu\, \widetilde{mm_1})n!\langle\widetilde{m_1}\rangle\}$: 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 Δ'_2 such that

$$\Gamma; \Delta_1 \vdash P_2 \overset{\ell}{\Longrightarrow} \Delta'_2 \vdash P'_2 \qquad (53)$$

$$\Gamma; \Delta'_1 \vdash P'_1 \approx \Delta'_2 \vdash P'_2. \qquad (54)$$

From transition (53) we may infer that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2})(P_2 \mid R) \overset{\ell}{\Longrightarrow} \Delta'_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2}')(P'_2 \mid R)$$

Furthermore, from (54) and the definition of $\mathcal{S}$ we infer that

$$\Gamma; \Delta'_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_1}')(P'_1 \mid R) \;\mathcal{S}\; \Delta'_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2}')(P'_2 \mid R)$$

as desired.

ii. Sub-case $\ell = (\nu\, \widetilde{m_1})n!\langle\lambda\widetilde{x}.\, Q_1\rangle$: Then we infer the typed transition

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{m_1})n!\langle\lambda\widetilde{x}.\, Q_1\rangle} \Delta'_1 \vdash P'_1$$

which implies the existence of P'_2, Δ'_2, Δ''_1, and Δ''_2 such that

$$\Gamma; \Delta_1 \vdash P_2 \overset{(\nu\, \widetilde{m_2})n!\langle\lambda\widetilde{x}.\, Q_2\rangle}{\Longrightarrow} \Delta'_2 \vdash P'_2 \qquad (55)$$

and

$$\Gamma; \Delta''_1 \vdash (\nu\, \widetilde{n_1}'')(P'_1 \mid Q\{\lambda\widetilde{x}.\, Q_1/x\}) \;\approx\; \Delta''_2 \vdash (\nu\, \widetilde{n_2}'')(P'_2 \mid Q\{\lambda\widetilde{x}.\, Q_2/x\}) \qquad (56)$$

for all Q with $x \in \mathtt{fv}(Q)$. From transition (55), we infer that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2})(P_2 \mid R) \overset{(\nu\, \widetilde{m_2})n!\langle\lambda\widetilde{x}.\, Q_2\rangle}{\Longrightarrow} \Delta'_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2}')(P'_2 \mid R)$$

Furthermore, from (56) we conclude that

$$\Gamma; \Delta''_1{\cdot}\Delta_3 \vdash (\nu\, \widetilde{n_1}'')(P'_1 \mid Q\{(\widetilde{x})Q_1/x\} \mid R) \;\mathcal{S}\; \Delta''_2{\cdot}\Delta_3 \vdash (\nu\, \widetilde{n_2}'')(P'_2 \mid Q\{\lambda\widetilde{x}.\, Q_2/x\} \mid R)$$

for all Q, with $x \in \mathtt{fv}(Q)$, as desired.

iii. Sub-case $\ell = (\nu\, \widetilde{mm_1})n!\langle\widetilde{m_1}\rangle$: From the definition of typed transition we infer that

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{mm_1})n!\langle\widetilde{m_1}\rangle} \Delta'_1 \vdash P'_1$$

which, in turn, implies that there exist Δ'_2, P'_2, and m_2 such that

$$\Gamma; \Delta_1 \vdash P_2 \overset{(\nu\, \widetilde{mm_2})n!\langle\widetilde{m_2}\rangle}{\Longrightarrow} \Delta'_2 \vdash P'_2 \qquad (57)$$

and

$$\Gamma; \Delta''_1 \vdash (\nu\, \widetilde{n_1})(P'_1 \mid Q\{\widetilde{m_1}/\widetilde{x}\}) \;\approx\; \Delta''_2 \vdash (\nu\, \widetilde{n_2})(P'_2 \mid Q\{\widetilde{m_2}/\widetilde{x}\}) \qquad (58)$$

for some Δ''_1 and Δ''_2, for all Q with $\{x\} = \mathtt{fv}(Q)$. From transition (57) we infer that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2}')(P_2 \mid R) \overset{(\nu\, \widetilde{mm_2})n!\langle\widetilde{m_2}\rangle}{\Longrightarrow} \Delta'_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2}''')(P'_2 \mid R)$$

Furthermore, from (58) we conclude that

$$\Gamma; \Delta''_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_1}'')(P'_1 \mid Q\{\widetilde{m_1}/\widetilde{x}\} \mid R) \;\mathcal{S}\; \Delta''_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{n_2}'')(P'_2 \mid Q\{\widetilde{m_2}/\widetilde{x}\} \mid R)$$

for all Q with $x \in \mathtt{fv}(Q)$, as desired.

2. Suppose that ℓ originates in R:

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_1})(P_1 \mid R) \xrightarrow{\ell} \Delta_1 \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P_1 \mid R')$$

This case is also divided into three sub-cases:

i. Sub-case $\ell \notin \{(\nu\, \widetilde{m})n!\langle \lambda \widetilde{x}.\, Q\rangle, (\nu\, \widetilde{mm_1})n!\langle \widetilde{m_1}\rangle\}$: From the LTS we infer that

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

for some Δ_3', which in turn implies

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_2})(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2}')(P_2 \mid R')$$

Now, from the definition of $\mathcal{S}$ we may obtain the desired conclusion:

$$\Gamma; \Delta_1 \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P_1 \mid R')\ \mathcal{S}\ \Delta_2 \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2}')(P_2 \mid R')$$

ii. Sub-case $\ell = (\nu\, \widetilde{m_1})n!\langle \lambda \widetilde{x}.\, Q\rangle$: From the LTS we infer that:

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell} \Delta_3' \vdash R' \tag{59}$$

for some Δ_3'. We then have that

$$\Gamma; \emptyset; \Delta_3'' \vdash (\nu\, \widetilde{m}')(R' \mid R_1\{{}^{\lambda \widetilde{x}.\, Q}/_{x}\}) \triangleright \diamond \tag{60}$$

for some Δ_3'' and for all R_1 with $\{x\} = \mathtt{fv}(R_1)$. Now, from (59) we obtain that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_2}')(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2})(P_2 \mid R')$$

Then, from (60) and the definition of $\mathcal{S}$ we obtain that

$$\begin{array}{ll} \Gamma; \emptyset;\ \Delta_1 \cdot \Delta_3'' \vdash (\nu\, \widetilde{m_1})(P_1 \mid (\nu\, \widetilde{m}')(R' \mid R_1\{{}^{\lambda \widetilde{x}.\, Q}/_{x}\})) \\ \quad \mathcal{S}\ \ \Delta_2 \cdot \Delta_3'' \vdash (\nu\, \widetilde{m_2})(P_2 \mid (\nu\, \widetilde{m}')(R' \mid R_1\{{}^{\lambda \widetilde{x}.\, Q}/_{x}\})) \end{array}$$

for all R_1 with $x \in \mathtt{fv}(R_1)$, as desired.

iii. Sub-case $\ell = (\nu\, \widetilde{mm})n!\langle \widetilde{m}\rangle$: Similarly as above, from the typed LTS we infer that:

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell} \Delta_3' \vdash R' \tag{61}$$

for some Δ_3'. We then have that

$$\Gamma; \emptyset; \Delta_3'' \vdash (\nu\, \widetilde{m}')(R' \mid Q\{{}^{\widetilde{m}}/_{\widetilde{x}}\}) \triangleright \diamond \tag{62}$$

for all Q with $\{\widetilde{x}\} = \mathtt{fv}(Q)$, for some Δ_3''. Now, from (61), we obtain that

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_2})(P_2 \mid R) \xrightarrow{\ell} \Delta_2 \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2})(P_2 \mid R')$$

Then, from (62) and the definition of $\mathcal{S}$ we obtain

$$\begin{array}{ll} \Gamma; \emptyset;\ \Delta_1 \cdot \Delta_3'' \vdash (\nu\, \widetilde{m_1})(P_1 \mid (\nu\, \widetilde{m})(R' \mid Q\{{}^{\widetilde{m}'}/_{\widetilde{x}}\})) \\ \quad \mathcal{S}\ \ \Delta_2 \cdot \Delta_3'' \vdash (\nu\, \widetilde{m_2})(P_2 \mid (\nu\, \widetilde{m}')(R' \mid Q\{{}^{\widetilde{m}}/_{\widetilde{x}}\})) \end{array}$$

for all Q with $\{\widetilde{x}\} = \mathtt{fv}(Q)$, as required.

3. We finally suppose that ℓ originates from the interaction between P_1 and R:

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_1})(P_1 \mid R) \xrightarrow{\tau} \Delta_1' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P_1' \mid R')$$

for some Δ_1', Δ_3'. We then have that

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{\ell_1} \Delta_1' \vdash P_1'$$

and

$$\Gamma; \Delta_3 \vdash R \xrightarrow{\ell_2} \Delta_3 \vdash R' \tag{63}$$

with $\ell_1 \asymp \ell_2$ (cf. Definition 3 (Page 13)).
This case is divided into three sub-cases:

i. $\ell_1 \notin \{(\nu\, \widetilde{m})n!\langle \lambda \widetilde{x}.\, Q\rangle, (\nu\, \widetilde{mm_1})n!\langle \widetilde{m_1}\rangle\}$: Then the transition from P_1 implies

$$\Gamma; \Delta_2 \vdash P_2 \overset{\hat{\ell_1}}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{64}$$
$$\Gamma; \Delta_1' \vdash P_1' \approx \Delta_2' \vdash P_2' \tag{65}$$

for some Δ_2'. From (63) and (64) we obtain

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_2})(P_2 \mid R) \Longrightarrow \Delta_2' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2}')(P_2' \mid R')$$

Then, from (65) and the definition of $\mathcal{S}$ we obtain the desired conclusion:

$$\Gamma; \Delta_1' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1}')(P_1' \mid R')\ \mathcal{S}\ \Delta_2' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2}')(P_2' \mid R')$$

ii. $\ell_1 = (\nu\, \widetilde{m_1})n!\langle \lambda \widetilde{x}.\, Q_1\rangle$: Then we have the transition

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{m_1})n!\langle \lambda \widetilde{x}.\, Q_1\rangle} \Delta_1' \vdash P_1'$$

for some Δ_1', which implies

$$\Gamma; \Delta_3 \vdash R \xrightarrow{n?\langle \lambda \widetilde{x}.\, Q_1\rangle} \Delta_3' \vdash R'\{\lambda \widetilde{x}.\, Q_1/x\} \tag{66}$$
$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_1})(P_1 \mid R) \xrightarrow{\tau} \Delta_1' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1}'')(P_1' \mid R'\{\lambda \widetilde{x}.\, Q_1/x\}) \tag{67}$$

for some Δ_1' and Δ_3'. In turn, the output transition from P_1 implies the existence of Δ_2', Q_2, P_2' such that

$$\Gamma; \Delta_2 \vdash P_2 \overset{(\nu\, \widetilde{m_2})n!\langle \lambda \widetilde{x}.\, Q_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{68}$$
$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_1' \mid Q\{\lambda \widetilde{x}.\, Q_1/x\}) \approx \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(P_2' \mid Q\{\lambda \widetilde{x}.\, Q_2/x\}) \tag{69}$$

for all Q with $\{x\} = \mathtt{fv}(Q)$, and for some Δ_1'' and Δ_2''. From (66) we obtain

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

for some Δ_3'', which may be combined with (68) to obtain

$$\Gamma; \Delta_2 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_2})(P_2 \mid R) \Longrightarrow \Delta_2' \cdot \Delta_3'' \vdash (\nu\, \widetilde{m_2}'')(P_2' \mid R'\{\lambda \widetilde{x}.\, Q_2/x\})$$

We conclude by setting Q as R' in (69) so as to obtain:

$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_1' \mid R'\{\lambda \widetilde{x}.\, Q_1/x\})\ \mathcal{S}\ \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(P_2' \mid R'\{\lambda \widetilde{x}.\, Q_2/x\}).$$

iii. $\ell_1 = (\nu\, \widetilde{m m_1}) n!\langle \widetilde{m_1}\rangle$. Then we have the transition

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{m m_1}) n!\langle \widetilde{m_1}\rangle} \Delta_1' \vdash P_1'$$

for some Δ_1', which implies

$$\Gamma; \Delta_3 \vdash R \xrightarrow{n?\langle \widetilde{m_1}\rangle} \Delta_3' \vdash R'\{\widetilde{m_1}/\widetilde{x}\} \qquad (70)$$

$$\Gamma; \Delta_1 \cdot \Delta_3 \vdash (\nu\, \widetilde{m_1})(P_1 \mid R) \xrightarrow{\tau} \Delta_1' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1}'')(P_1' \mid R'\{m_1/x\})$$

for some Δ_3'. In turn, the output transition from P_1 implies the existence of Δ_2', P_2, and $\widetilde{m_2}$ such that

$$\Gamma; \Delta_2 \vdash P_2 \overset{(\nu\, \widetilde{m m_2}) n!\langle \widetilde{m_2}\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \qquad (71)$$

$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1}')(P_1' \mid Q\{\widetilde{m_1}/\widetilde{x}\}) \;\approx\; \Delta_2'' \vdash (\nu\, \widetilde{m_2}')(P_2' \mid Q\{\widetilde{m_2}/\widetilde{x}\}) \qquad (72)$$

for all Q with $\{x\} = \mathtt{fv}(Q)$, and for some Δ_1'' and Δ_2''. From (70) and the Substitution Lemma (Lemma 10 (Page 34)) we obtain

$$\Gamma; \Delta_3 \vdash R \xrightarrow{n?\langle \widetilde{m_2}\rangle} \Delta_3'' \vdash R'\{\widetilde{m_2}/\widetilde{x}\}$$

for some Δ_3'', which may be combined with (71) to obtain

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

We conclude by setting Q as R' in (72):

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

□

In order to prove Lemma 7 (Page 25) (i.e., $\cong \subseteq \approx^{\mathtt{H}}$), below we follow the technique developed in [8] and refined for session types in [19, 18].

Definition 25 (Definability) Let $\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond$. A visible action ℓ is *definable* whenever, given a fresh name $succ$, there exists a (testing) process

$$\Gamma; \emptyset; \Delta_2 \vdash T\langle \ell, succ\rangle \triangleright \diamond$$

such that:

1. If $\Gamma; \Delta_1 \vdash P \overset{\ell}{\longmapsto} \Delta_1' \vdash P'$ then, for some Δ_2', either
   a) $\ell \neq (\nu\, \widetilde{m}) n!\langle V\rangle$ and $P \mid T\langle \ell, succ\rangle \longrightarrow P' \mid succ!\langle \overline{n}\rangle.\mathbf{0}$ and $\Gamma; \emptyset; \Delta_1' \cdot \Delta_2' \vdash P' \mid succ!\langle \overline{n}\rangle.\mathbf{0} \triangleright \diamond$
   b) $\ell = (\nu\, \widetilde{m}) n!\langle V\rangle$ and $P \mid T\langle \ell, succ\rangle \longrightarrow (\nu\, \widetilde{m})(P' \mid t \hookleftarrow_{\mathtt{H}} V \mid succ!\langle \overline{n}, V\rangle.\mathbf{0})$ and $\Gamma; \emptyset; \Delta_1' \cdot \Delta_2' \vdash (\nu\, \widetilde{m})(P' \mid t \hookleftarrow_{\mathtt{H}} V \mid succ!\langle \overline{n}, V\rangle.\mathbf{0}) \triangleright \diamond$, for some fresh t.
2. If $P \mid T\langle \ell, succ\rangle \longrightarrow Q$ with $\Gamma; \emptyset; \Delta \vdash Q \downarrow_{succ}$ then there exists a P' such that $\Gamma; \Delta_1 \vdash P \overset{\ell}{\Longmapsto} \Delta_1' \vdash P'$ and one of the following holds:
   a) $\ell \neq (\nu\, \widetilde{m}) n!\langle V\rangle$ and $Q \equiv P' \mid succ!\langle \overline{n}\rangle.\mathbf{0}$.
   b) $\ell = (\nu\, \widetilde{m}) n!\langle V\rangle$ and $Q \equiv (\nu\, \widetilde{m})(P' \mid t \hookleftarrow_{\mathtt{H}} V \mid succ!\langle \overline{n}, V\rangle.\mathbf{0})$, for some fresh t.

We first show that every visible action ℓ is definable.

Lemma 20 (Definability) *Every visible action ℓ is definable.*

Proof Let $succ$ be a fresh name. We define $T\langle \ell, succ\rangle$ as follows:

$$\begin{aligned}
T\langle n?\langle V\rangle, succ\rangle &= \overline{n}!\langle V\rangle.succ!\langle \overline{n}\rangle.\mathbf{0} \\
T\langle n \,\&\, l, succ\rangle &= \overline{n} \triangleleft l.succ!\langle \overline{n}\rangle.\mathbf{0} \\
T\langle (\nu\, \widetilde{m}) n!\langle V\rangle, succ\rangle &= \overline{n}?(y).(t \hookleftarrow_{\mathtt{H}} y \mid succ!\langle \overline{n}, y\rangle.\mathbf{0}) \\
T\langle n \oplus l, succ\rangle &= \overline{n} \triangleright \{l : succ!\langle \overline{n}\rangle.\mathbf{0}), l_i : (\nu\, a)(a?(y).succ!\langle \overline{n}\rangle.\mathbf{0})\}_{i \in I}
\end{aligned}$$

Let process

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

It is straightforward to do a case analysis on all actions ℓ such that

$$\Gamma; \emptyset; \Delta \vdash P \overset{\ell}{\longmapsto} \Delta' \vdash P'$$

to show that ℓ is definable. $\square$

We require the following auxiliary result:

Lemma 21 (Extrusion) Let P and Q be processes, and let *succ* be a fresh name. If

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \cong \Delta_2 \vdash (\nu\, \widetilde{m_2})(Q \mid succ!\langle \overline{n}, V_2\rangle.\mathbf{0})$$

with $\{\widetilde{m_1}\} = \mathtt{fn}(V_1)$ and $\{\widetilde{m_2}\} = \mathtt{fn}(V_2)$ then there exist Δ_1 and Δ_2 such that

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

Proof Let $\mathcal{S}$ be a relation defined as:

$$\begin{aligned}\mathcal{S} = \{(&\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond\ ,\ \Gamma; \emptyset; \Delta_2 \vdash Q \triangleright \diamond) \mid \\ &\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \cong \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid succ!\langle \overline{n}, V_2\rangle.\mathbf{0}),\\ &\wedge m_1 \in \mathtt{fn}(V_1) \wedge m_2 \in \mathtt{fn}(V_2)\ \}\end{aligned}$$

We show that $\mathcal{S}$ is a reduction-closed, barbed congruence.

I. **Reduction-closed:** The reduction $P \longrightarrow P'$ implies

$$(\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \longrightarrow (\nu\, \widetilde{m_1})(P' \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0})$$

which, due to freshness of *succ*, in turn implies

$$(\nu\, \widetilde{m_1})(Q \mid succ!\langle \overline{n}, V_2\rangle.\mathbf{0}) \longrightarrow^* (\nu\, \widetilde{m_1})(Q' \mid succ!\langle \overline{n}, V_2\rangle.\mathbf{0}).$$

Therefore, $Q \longrightarrow^* Q'$, as required.

II. **Barb Preserving:** Suppose $\Gamma; \emptyset; \Delta_1 \vdash P \downarrow_m$. We analyse three cases, depending on the nature of m:

i) Case $m \neq s$ (m is not a session name): Then from $\Gamma; \emptyset; \Delta_1 \vdash P \downarrow_m$ we infer

$$\Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \downarrow_m$$

for some Δ_1', which implies

$$\Gamma; \emptyset; \Delta_2' \vdash (\nu\, \widetilde{m_2})(Q \mid succ!\langle \overline{n}, V_2\rangle.\mathbf{0}) \Downarrow_m .$$

for some Δ_2'. Then, from the freshness of *succ*, we obtain $\Gamma; \emptyset; \Delta_2 \vdash Q \Downarrow_m$, as required.

ii) Case: $m = s$ (m is a session name) and $m \neq n$. The proof follows a similar reasoning as in the previous case.

iii) 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 implies that $n, \overline{n} \in \mathtt{dom}(\Delta_1')$. Therefore, from the definition of barbs (Definition 9 (Page 16)) we can infer that

$$\Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \not\downarrow_n$$

because both endpoints of session n are present in Δ_1'.

To observe the desired barb we exploit an additional test process, with an extra fresh name $succ'$. We compose $\Gamma; \emptyset; \Delta_1 \vdash P \triangleright \diamond$ with $\overline{succ}?(x, y).T\langle \ell, succ'\rangle$ where $\mathtt{subj}(\ell) = x$. We then have

$$\Gamma; \emptyset; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \mid \overline{succ}?(x, y).T\langle \ell, succ'\rangle \triangleright \diamond$$

The definition of definability and the fact that $\Gamma; \emptyset; \Delta_1 \vdash P \downarrow_n$ imply that

$$(\nu\, \widetilde{m_1})(P \mid succ!\langle \overline{n}, V_1\rangle.\mathbf{0}) \mid \overline{succ}?(x, \widetilde{y}).T\langle \ell, succ'\rangle \longrightarrow^* (\nu\, \widetilde{m_1})(P' \mid succ'!\langle \overline{n}, V_1'\rangle.\mathbf{0})$$

and furthermore

$$(\nu\, \widetilde{m_2})(Q \mid succ!\langle \overline{n}, V_2\rangle.\mathbf{0}) \mid \overline{succ}?(x, \widetilde{y}).T\langle \ell, succ'\rangle \longrightarrow^* (\nu\, \widetilde{m_2})(Q' \mid succ'!\langle \overline{n}, V_2'\rangle.\mathbf{0})$$

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

$$\begin{aligned}\mathcal{C} = \{(&\Gamma;\emptyset;\Delta_1\cdot\Delta_3 \vdash P \mid R \triangleright \diamond, \Gamma;\emptyset;\Delta_2\cdot\Delta_3 \vdash Q \mid R \triangleright \diamond) \ \mid \\ &\forall R \text{ such that } \exists \Delta_3, \Gamma;\emptyset;\Delta_3 \vdash R \triangleright \diamond \wedge \\ &\Gamma;\Delta_1' \vdash (\nu\,\widetilde{m_1})(P \mid \mathit{succ}!\langle \overline{n}, V_1\rangle.\mathbf{0}) \cong \Delta_2' \vdash (\nu\,\widetilde{m_2})(Q \mid \mathit{succ}!\langle \overline{n}, V_2\rangle.\mathbf{0})\}\end{aligned}$$

We show that $\mathcal{C}$ is a congruence with respect to parallel composition. We distinguish two cases:

i) Case $(\overline{n} \cup \mathtt{fn}(V_1) \cup \mathtt{fn}(V_2)) \cap \mathtt{fn}(R) = \emptyset$: Then from the contextual definition of $\cong$ we can deduce that for all $\Gamma;\emptyset;\Delta_3 \vdash R \triangleright \diamond$:

$$\Gamma;\Delta_1'\cdot\Delta_3 \vdash (\nu\,\widetilde{m_1})(P \mid \mathit{succ}!\langle \overline{n}, V_1\rangle.\mathbf{0}) \mid R \cong \Delta_2'\cdot\Delta_3 \vdash (\nu\,\widetilde{m_2})(Q \mid \mathit{succ}!\langle \overline{n}, V_2\rangle.\mathbf{0}) \mid R$$

Because of the requirement $(\overline{n} \cup \mathtt{fn}(V_1) \cup \mathtt{fn}(V_2)) \cap \mathtt{fn}(R) = \emptyset$ the above is structurally congruent to

$$\Gamma;\Delta_1'\cdot\Delta_3 \vdash (\nu\,\widetilde{m_1})(P \mid \mathit{succ}!\langle \overline{n}, V_1\rangle.\mathbf{0} \mid R) \cong \Delta_2'\cdot\Delta_3 \vdash (\nu\,\widetilde{m_2})(Q \mid \mathit{succ}!\langle \overline{n}, V_2\rangle.\mathbf{0} \mid R)$$

The desired conclusion is then immediate from the definition of $\mathcal{C}$.

ii) Case $\widetilde{s} = \{\overline{n}, \widetilde{m_1}\} \cap \{\overline{n}, \widetilde{m_2}\} \subseteq \mathtt{fn}(R)$: Let $R^{\widetilde{y}}$ 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 $\Gamma;\emptyset;\Delta_3' \vdash \overline{\mathit{succ}}?(\widetilde{y}).(R^{\widetilde{y}} \mid \mathit{succ}'!\langle\widetilde{y}\rangle.\mathbf{0}) \triangleright \diamond$:

$$\begin{aligned}\Gamma;\emptyset;\Delta_1'' &\vdash (\nu\,\widetilde{m_1})(P \mid \mathit{succ}!\langle \overline{n}, V_1\rangle.\mathbf{0}) \mid \overline{\mathit{succ}}?(\widetilde{y}).(R^{\widetilde{y}} \mid \mathit{succ}'!\langle\widetilde{y}\rangle.\mathbf{0}) \\ \cong \Delta_2'' &\vdash (\nu\,\widetilde{m_2})(Q \mid \mathit{succ}!\langle \overline{n}, V_2\rangle.\mathbf{0}) \mid \overline{\mathit{succ}}?(\widetilde{y}).(R^{\widetilde{y}} \mid \mathit{succ}'!\langle\widetilde{y}\rangle.\mathbf{0})\end{aligned}$$

for some Δ_1'', Δ_2''. Applying reduction closeness to the above pair we infer:

$$\Gamma;\Delta_1'' \vdash (\nu\,\widetilde{m_1})(P \mid R \mid \mathit{succ}'!\langle \overline{n}, V_1\rangle.\mathbf{0}) \cong \Delta_2'' \vdash (\nu\,\widetilde{m_2})(Q \mid R \mid \mathit{succ}'!\langle \overline{n}, V_2\rangle.\mathbf{0})$$

The conclusion then follows from the definition of $\mathcal{C}$.

□

We can finally prove Lemma 7 (Page 25):

Lemma 22 $\cong \subseteq \approx^{\mathtt{H}}$.

Proof Let $\Re$ be the typed relation (we omit the typing information in the definition):

$$\Re = \{(P_1, P_2) \ \mid \ \Gamma;\Delta_1 \vdash P_1 \cong \Delta_2 \vdash P_2\}$$

We prove that $\Re$ 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

$$\Gamma;\Delta_2 \vdash P_2 \overset{\tau}{\Longrightarrow} \Delta_2' \vdash P_2'$$

for some Δ_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 $T\langle \ell, \mathit{succ}\rangle$ to obtain

$$\Gamma;\Delta_1\cdot\Delta_3 \vdash P_1 \mid T\langle \ell, \mathit{succ}\rangle \cong \Delta_2\cdot\Delta_3 \vdash P_2 \mid T\langle \ell, \mathit{succ}\rangle \tag{73}$$

for some Δ_3. From this point on we distinguish two sub-cases:

i. Sub-case $\ell \in \{n?\langle V_1\rangle, n \oplus l, n \,\&\, l\}$: We then obtain

$$P_1 \mid T\langle \ell, succ\rangle \longrightarrow P_1' \mid succ!\langle \widetilde{n}\rangle.\mathbf{0}$$
$$\Gamma; \emptyset; \Delta_1' \cdot \Delta_3' \vdash P_1' \mid succ!\langle \widetilde{n}\rangle.\mathbf{0} \downarrow_{succ}$$

for some Δ_3'. From (73) we may now infer:

$$\Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash P_2 \mid T\langle \ell, succ\rangle \Downarrow_{succ}$$

which, using Lemma 20 (Page 62), implies

$$\Gamma; \Delta_2 \vdash P_2 \overset{\ell}{\Longrightarrow} \Delta_2' \vdash P_2'$$
$$P_2 \mid T\langle \ell, succ\rangle \longrightarrow^* P_2' \mid succ!\langle \widetilde{n}\rangle.\mathbf{0}$$

and

$$\Gamma; \Delta_1' \cdot \Delta_3' \vdash P_1' \mid succ!\langle \widetilde{n}\rangle.\mathbf{0} \cong \Delta_2' \cdot \Delta_3' \vdash P_2' \mid succ!\langle \widetilde{n}\rangle.\mathbf{0}$$

We then apply Lemma 21 (Page 63) 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 $\ell = (\nu\, \widetilde{m_1})n!\langle V_1\rangle$: Note that $T\langle (\nu\, \widetilde{m_1})n!\langle V_1\rangle, succ\rangle = T\langle (\nu\, \widetilde{m_2})n!\langle V_2\rangle, succ\rangle$. The transition from P_1 can be then written as

$$\Gamma; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{m_1})n!\langle V_1\rangle} \Delta_1' \vdash P_1' \tag{74}$$

for some Δ_1'. If we use the test process $T\langle (\nu\, \widetilde{m_1})n!\langle V_1\rangle, succ\rangle$, then we may obtain:

$$P_1 \mid T\langle (\nu\, \widetilde{m_1})n!\langle V_1\rangle, succ\rangle \longrightarrow (\nu\, \widetilde{m_1})(P_1' \mid t \hookleftarrow_{\mathtt{H}} V_1) \mid succ!\langle \widetilde{n}, V_1\rangle.\mathbf{0}$$
$$\Gamma; \emptyset; \Delta_1' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1})(P_1' \mid t \hookleftarrow_{\mathtt{H}} V_1) \mid succ!\langle \widetilde{n}, V_1\rangle.\mathbf{0} \downarrow_{succ}$$

for some Δ_3'. Using (73) we may then infer

$$\Gamma; \emptyset; \Delta_2 \cdot \Delta_3 \vdash P_2 \mid T\langle (\nu\, \widetilde{m_2})n!\langle V_2\rangle, succ\rangle \Downarrow_{succ}$$

which, using Lemma 20 (Page 62), implies

$$\Gamma; \Delta_2 \vdash P_2 \overset{(\nu\, \widetilde{m_2})n!\langle V_2\rangle}{\Longrightarrow} \Delta_2' \vdash P_2' \tag{75}$$
$$P_2 \mid T\langle \ell, succ\rangle \longrightarrow^* (\nu\, \widetilde{m_2})(P_2' \mid t \hookleftarrow_{\mathtt{H}} V_2) \mid succ!\langle \widetilde{n}, V_2\rangle.\mathbf{0}$$

for some Δ_2', and

$$\begin{aligned}\Gamma; \emptyset;\ & \Delta_1' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_1})(P_1' \mid t \hookleftarrow_{\mathtt{H}} \lambda\widetilde{x}.\, Q_1) \mid succ!\langle \widetilde{n}, V_1\rangle.\mathbf{0} \\ \cong\ & \Delta_2' \cdot \Delta_3' \vdash (\nu\, \widetilde{m_2})(P_2' \mid t \hookleftarrow_{\mathtt{H}} \lambda\widetilde{x}.\, Q_2) \mid succ!\langle \widetilde{n}, V_2\rangle.\mathbf{0}\end{aligned}$$

We then apply Lemma 21 (Page 63) to obtain:

$$\Gamma; \Delta_1' \vdash (\nu\, \widetilde{m_1})(P_1' \mid t \hookleftarrow_{\mathtt{H}} V_1) \cong \Delta_2' \vdash (\nu\, \widetilde{m_2})(P_2' \mid t \hookleftarrow_{\mathtt{H}} V_2)$$

From the above result and the definition of $\Re$ we finally obtain:

$$\begin{aligned}\Gamma; \emptyset;\ & \Delta_1' \vdash (\nu\, \widetilde{m_1})(P_1' \mid t \hookleftarrow_{\mathtt{H}} V_1) \\ \Re\ \ & \Delta_2' \vdash (\nu\, \widetilde{m_2})(P_2' \mid t \hookleftarrow_{\mathtt{H}} V_2)\end{aligned}$$

as required.

□

Reply to Referees

Characteristic Bisimulation for Higher-Order Session Processes

Dimitrios Kouzapas, Jorge A. Pérez, and Nobuko Yoshida

First of all, we thank the reviewers for their precise comments and thorough remarks. We have done our best effort to take them into account when revising our paper. Next, we give detailed responses to all of their comments, both major and minor, explaining how they were considered.

Following the feedback from the reviewers (in particular, comments A35 - A44 by Reviewer #1), we have streamlined the technical development. In this revision we use a simpler notion of characteristic processes (Figure 6), which enables us to give a more direct proof of process substitution lemmas.

In this revision we have also expanded on the role of session types (in particular, linearity of sessions) in our developments. We have emphasized on these points in Sections 1 and 2.

Reviewer #1 (47 comments)

Thank you very much for your detailed comments. We answer each point below.

A1. Section 2: I am not sure this section should exist, as it relies on notions that are not been introduced, and the reader (or at least, most of them) is not familiar with the calculus yet. I would suggest to move some of the discussion in the introduction (eg, the part about context bisimilarity) or in Section 5

- We believe this section is useful to highlight the technical novelties in our approach. We have added material to make this section more clear and self-contained.

A2. p4, l21: P' \mapsto P', I guess it should P \mapsto P'

✔ Fixed.

A3. p4, l39/40: "name matching is crucial to prove completeness", If I recall correctly, Sangiorgi does not need name matching in his seminal work on HOpi

- The language used in Sangiorgi's PhD thesis includes matching. In Sangiorgi's "Bisimulation for Higher-Order Process Calculi" (Information and Computation), the used language does not include matching. We have rephrased this sentence.

A4. p5, fig1: the presentation could be improved (eg, one column for the reduction rules)

✔ Fixed: we revised the presentation of Figure 1.

A5. syntax: it's weird that you do not have a and \overline a for shared names, why this choice?

- Indeed, we use \overline only for session names, not for shared names. This is to emphasise the fact that each session name has two endpoints. Notice that the overline for session channels is important technically [GH05] because it helps to distinguish two session endpoints. Shared names can have an arbitrary number of endpoints.

A6. Why having lambda-like abstractions in the syntax?

- The use of lambda-abstractions as values is what makes our language `higher-order'. Indeed, one may see higher-order process calculi such as HOπ as a bridge between the pi-calculus and the lambda-calculus. We have elaborated on this relation in the Introduction.

A7. p7 l29: "our system distills the key features of .." what are the differences between the two systems?

- Our system is strictly included in that considered by Mostrous and Yoshida (Information and Computation, 2015): they admit asynchronous communication and arbitrary nesting in functional types (their types are of the form U -> T, where T ranges over U and the process type <>). In contrast, our functional types are of the form U -> <>.

A8. In general, provide more intuitions on the type system, and explain in particular why linearity is important.

✔ Fixed: Now we clarify the distinction between linear and shared computation in session right at the beginning: see for instance the opening paragraph in the introduction.

A9. p7 l49, definition 19: make a reference to the Appendix (or simply move this definition in the main text)

✔ Fixed: we have revised this text, mentioning the appendix and clarifying the notation for type equivalence.

A10. p8, l47: the font used here for rule names is not the same as in Fig 3

✔ Fixed: we have checked the fonts (but also the description of the rules).

A11. p8, l48-50: "where we require ... shared type." There is something wrong with this sentence, please rephrase

✔ Fixed: we have rephrased this sentence.

A12. fig 3: the rule [Sess] is wrong, and there is a P \triangleright T in rule [End]

✔ Fixed: we have included the correct formulation for Rule [Sess] and amended Rule [End].

A13. p10 th 1: is this a new result? I would expect this to have been proved when the type system has been introduced in the previous work. If so, the proof (in Appendix) could be omitted, unless the differences between the two type systems are major.

- We understand the reviewer's suggestion, but we prefer to keep the type system and its results in order to make the paper self-contained. As mentioned above (cf. A7), the system in this paper is a particular case of the system of Mostrous and Yoshida, where typing is more general (it considers asynchronous communication and arbitrary functional types). We think it is convenient to present the properties of the type system considering the features that we use in this presentation.

A14. p10, l31: \lambda x.P -> P_{xy}?

✔ Fixed.

A15. p11, l39: defining subj on the two remaining labels does not take that much space

✔ Fixed: we have added the definition for the remaining labels.

A16. p12, Notation 3: "given a V of value type U" -> given a value V of type U

✔ Fixed.

A17. p13, def 4: I don't understand why it is required that the \Lambda are empty
- Because the transition relation is defined between closed processes (i.e. processes without higher-order variables).

A18. p13, l50: ii)
✔ Fixed.

A19. p13, def 5: what is the intuition behind confluence?
- Since Delta captures session communication, which is deterministic, the notion of confluence allows us to abstract away from alternative computation paths due to non-interfering reductions of session names. We have clarified this in the text (see Section 5.3).

A20. p14, barbs: why observing only the outputs and not the inputs?
- Observing the outputs is enough because input actions can be observed indirectly by output actions on fresh names. For instance, the process P = n(x).P' has an input barb; by composing P with n!<m>. succ!<>. 0 (with fresh succ) then one obtains a (weak) observation uniquely associated to the input in P. We have clarified this in the text (see Section 5.3).

A21. p15, l10: a definition is missing in Fig 6
✔ Fixed.

A22. 16, l33: a closing parenthesis is missing
- We could not find the corresponding errors.

A23. p17, l8: I assume m is fresh?
✔ Fixed: Yes, m is fresh. We have added this in the text.

A24. p18, def 16: there is a mismatch between U, U_1, U_2
✔ Fixed.

A25. p20, Lemma 5 and other henceforth: improve the spacing (it is not a composition of relations)
✔ Fixed. Now it's Lemma 6.

A26. p22, prop 3: the proof in Appendix C is short enough that it can be put in the main text. Also, remind the definitions of Client 1 and Client 2, for convenience
✔ Fixed: We have moved the proof in the appendix to the main text: see Section 5.9 and Figures 7 and 8.

A27. It would be interesting to see the same proof with characteristic bisimilarity to compare the two (there is no example with that bisimilarity in the paper). Also, (at least) an additional example would be nice.
✔ Fixed: We have incorporated this suggestion by commenting on the differences between characteristic and higher-order similarity - see the new Remark 1 before Section 5.7.

A28. p22, l49: \lambda-calculi => calculus
✔ Fixed.

A29. p24, fig 7: spacing issue between (5) and (6)
✔ Fixed. It was not a spacing issue. The figure compares two different approaches according to the example. We have changed the layout of the figure, so to better distinguish the two parts of the comparison; see Figure 9 now.

A30. p28, lemma 9: I don't understand why V must be typed with empty Lambda and Delta
- Typing value V with an empty Δ and Λ means that V has no linear values and session channels, thus it is shared and can be substituted in environment Γ. Also see typing rule [Prom].

A31. p28, th 3: the first item is not in th 1
✔ Fixed.

A32. p29, l15: "representative subcases", say which other subcases are similar to these two
✔ Fixed: We have added more explanations on the other sub-cases.

A33. p30, l34: "Actions s? and s! are forbidden by the LTS", I don't understand how and why
✔ Fixed.

A34. p30, l42 and after: "\simeq_H is a congruence" where is it proved?
✔ Fixed.

A35. p31, proof of lemma 10: you prove that H is in C, which looks like the "easy" implication to me (the testing processes in C can do less stuff than those in H), so do the other implication as well (I assume there is no limitation on space)
✔ Fixed. Lemma 10 now becomes Lemma 14. We added much more detailed proofs and we consider the missing implication.

A36. p32, lemma 11: "for some U", should be quantified elsewhere (it does not encompass item 1)
✔ Fixed. Lemma 11 no longer is used as in the previous version: we have rewritten the proof, which is now much clearer because it is based on the new definition of characteristic processes. See Lemma 17 for process substitution; it relies on Lemmas 16 and 15.

A37. p33, eq (18): how do we know that Q1|Q2 matches with the same Q'1, Q'2 as in (16)? The two H relations involving P1, P2, Q1, Q2 seem independent
p33, case2: don't you have to check also inputs on t with process of the form {U}_x?
p33, l45: an arrow should be a \mapsto

- As explained above, we no longer use Lemma 11, and so this comment no longer applies.

A38. p34, lemma 12: we can write P=\nu \emptyset (P | 0), I suppose you want some properties on P1, P2, that are not written there
p34, l12: "the normal form" -> a normal form, I doubt this normal form is unique
Lemma 12 is not needed since we changed the proofs.

A39. P35, l1: how do we know that we can apply Lemma 11? (in particular, the conditions on P1 H Q1 | Q3)
p35, l5: "from (22)" probably from another (unnumbered) equation
p35, l36: change P2 into something else, it is not the same P2 as in (24)
p35, l49: replace R with P, I guess
p36, case 3: unless I am mistaken, using case 2 tells you that the transition of line 11 is matched by \nu m2 (Q2 | ...) ==?..==> Q'2, but it does not tell you anything on the shape of Q'2 (in particular, \lambda z Q may have been applied in R, so it cannot be written as R{\lambda z Q/x})
p36, l46: the "m" in the transition should be n

✔ Fixed. Lemma 13 now becomes Lemma 17. We have changed the statement of the lemma and now it is simpler; we remove the requirement for substituting the characteristic process. This requirement is no longer needed because of the new characteristic process definition. We have also rewritten the proof, to accommodate changes in the definitions. Other minor comments don't apply.

A40. p39, l42: the substitution lemma does not give you that directly (it's just making another input with R)
✔ Fixed. No need to call upon the Substitution Lemma.

A41. p38, l50: the \vdash at the end
✔ Fixed.

A42. p40, l4: the s1 should be something else
✔ Fixed. It should have been m_1

A43. p40, l39: the \emph is not needed
✔ Fixed.

A44. p41, lemma 17: I have a lot of issues with this lemma: as it stands, it is not true : \nu m (m.0 | succ..) \congr O | succ... but m.0 not \congr to 0, so there are hypothesis missing. Or when it is applied in the following proof, it seems that you don't need to remove the restrictions.
✔ Fixed. Old Lemma 17 is now Lemma 21. The extra hypotheses have been added.

A45. In the proof of this lemma,
- barb preserving: I don't understand why you need to distinguish two cases.
l26: I don't understand how the fact that P has a barb on n plays a role in the following transitions
- congruence: I don't follow what you are proving there, in particular why you need two cases again
✔ Fixed. We now have an updated Definability Lemma (see Lemma 20). The barb preserving case of the proof now contains a clear distinction of three sub-cases that includes sub-cases for shared and session names. The extra hypothesis on the statement of the lemma makes the distinction of the two sub-cases of the congruence case clear.

A46. p43, l2: (x)Q => \lambda x Q
✔ Fixed.

A47. there is a lot of \tilda missing in the rest of the proof
✔ Fixed.

Reviewer #2 (89 comments)

Thank you very much for your detailed comments. We answer each point below.

-- 2

B1. Explain why session types lead to a "coarser semantics".
✔ Fixed: We have rephrased this sentence: we mention that typed contexts (environments disciplined by types) usually result in typed semantics that admit stronger properties than untyped semantics.

B2. "this new notion of typed bisimilarity is tractable": it is rather more tractable, I think work needs to be done to make it really tractable.
✔ Fixed: We have written 'more tractable than context bisimilarity'.

B3. "as an useful"
✔ Fixed.

B4. "context bisimilarities" referring to contextual equivalence is more appropriate here
✔ Fixed.

B5. "discusses with related works"
✔ Fixed.

B6. "an empirical comparison": "empirical" is misleading here
✔ Fixed: We have rewritten this.

-- 3

B7. "binary relations" the typed relations aren't really binary, since they do not involve only a pair of processes.
- We disagree: the relations in this paper contain pairs of typed processes. Even if the definition of these relations includes type information, they remain essentially binary relations.

B8. "on the output clause" -> "in the output clause"
✔ Fixed.

B9. "closure R{V/x}": this is a concern about terminology which I have for the whole paper. Why call this a "closure"? I believe this is non-standard terminology, and, more importantly, I find it misleading. I moreover have the impression that "closure" is used for processes (as here) and for relations. I strongly recommend to avoid using closure this way.
- Closure is not standard terminology. We use it mostly to refer to typed relations that we show to be bisimilarities. We have revised this terminology consistently throughout the paper.

--4

B10. "This is necessary:" it seems that you prove that this is sufficient, but not that it is necessary. Actually, it would be interesting to try and understand how necessary the definitions you propose are. For instance, in which setting would charactestic values (resp. trigger values) be enough?

- We understand the reviewer's suggestion but it seems unlikely that characteristic values or trigger values in isolation would lead to interesting behavioral equivalences. Indeed, each notion is sufficiently simple on its own (so it is not clear how to make it more elementary) and our results indicate that their combination lead to an appropriate distinguishing power.

B11. "admits only names...": rephrase

✔ Fixed.

B12. The notations involving characteristic values are used without being introduced (there is no need for a formal definition at this stage, but you should comment on them).

✔ Fixed: We have rephrased/reordered the text under 'Refined Input Transitions' in order to make the different notations explicit (and hopefully clearer).

B13. The last sentence of Section 2 should be rephrased: the relation with matching is important, especially in view of previous work. A more detailed explanation should be given.

✔ Fixed. We have added some related explanations before Section 3.

--5

B14. The first paragraph of 3.1 should be re-read (e.g., there are inconsistencies in the usage of metavariables).

✔ Fixed: we have rewritten this paragraph.

B15. "a set of" -> "the sets of"

✔ Fixed.

B16. "Rule [App] is a value application": rephrase

✔ Fixed.

--6

B17. "Notice that the above.. Client_1." I found this comment usueless.

✔ Fixed: We have removed this comment.

--7

B18. I could not find Definition 19 (and it should be presented here anyway, instead of pointing to a later part of the paper).

✔ Fixed. We have clarified that Definition 19 is in the Appendix (now Definition 21).

--8

B19. Def. 1: "be a set" / "if the pair"

✔ Fixed.

B20. we lack the definition of \sim

✔ Fixed: it has been introduced before, via the pointer to the Appendix.

--9

B21. Rule [Sess] is not the right one.

✔ Fixed. We have added the correct rule. Please see also comment A12.

B22. Figure 3: I find the treatment of the received name in the rules involving reception rather puzzling. Instead of mentioning \Gamma\setminus x in the conclusion, why not adopting the standard presentation, where the typing context is extended in the premise? The rules for restriction are actually presented using the latter style.

- The two treatments are equivalent and we choose Gamma\x; we have defined it in the text.

B23. In rules [Req] and [Acc], you could improve readability by writing directly U1 = <U2>.

✔ Fixed.

--10

B24. Example 2: use "typing" rather than "type".

✔ Fixed.

--11

B25. Figure 4:

. rule <Bra>, "j\in I" should appear next to the rule, not in conclusion

. you should introduce \equiv_{\alpha} in the main text

✔ Fixed both points.

B26. Section 5.1: It is unfortunate that you rely on the distinction between l (for labels) and \ell (for actions). A different choice of meta-variables would make things easier to read.

- We understand the reviewer's observation, but we think that the current notation for labels and actions doesn't lead to ambiguities: the font used is different, and which "l" is intended in each case is clear from the context.

--12

B27. "closes the LTS under restriction": rephrase (similarly for "under parallel composition and alpha-renaming")

✔ Fixed: We have rephrased the sentences.

B28. "results by" -> "is obtained by"

✔ Fixed.

B29. "Notice ... shared environments.": you should state weakening, and observe that transitions can only extend the shared environment.

✔ Fixed: We have revised this sentence. Also, we have added intuitive explanations for exchange, contraction, and weakening (see Section 4.2).

B30. Paragraph about "Output Actions": when explaining how the usages of typing environments are constrained by the rules, use the metavariables appearing in the rules. This makes things easier for the reader (this is done in some parts, referring, e.g., to \Lambda', but it should be done systematically).

✔ Fixed: We have revised this paragraph, adding notation used in the output rules.

B31. I suggest to refer to Example 3 when discussing the corresponding rule, which is rather difficult to digest.

✔ Fixed: We have followed the reviewer's suggestion, moving the example closer to the explanation of the output rules.

--13

B32. Example 3: you should provide more explanations about the correspondence with the inference rule.

✔ Fixed: Now it is Example 2 (Section 5.2). We have added more explanations

B33. Definition 5: usually, in the theory of rewriting, "confluent" is rather used for a relation. "Joinable" seems a better terminology than "confluent" in the present situation.

- This suggestion is fairly reasonable, but we prefer to keep the terminology unchanged, so to be consistent with our previous works in which we give similar definitions.

--14

B34. Definition 8, items 1.a and 1.b: isn't it rather "P\downarrow n if either (a) or (b)"?

✔ Fixed. Now it's Definition 9.

B35. Definition 9: | is missing after (\nu n)C

✔ Fixed. Now it's Definition 10.

B36. Definition 10: where does \Delta'_1 come from? This kind of imprecision occurs in quite a lot of places in the paper, where new environments are generated along transitions. Quantifications over such environments should appear explicitly.

✔ Fixed. Now it's Definition 11.

--15

B37. Figure 6: a case is missing (!). For the case of t, you should explain the correspondence between t and X_t. You should also provide explanations as to why there is an output in the case of {<S>}^u.

- We changed the definition of Figure 6 to make a proof more accessible taking Reviewer #1's comments. However we added the more detailed proof for typability in Appendix.

B38. Definition 11: fv(R) = {x}.

✔ Fixed.

B39. "hinted at" -> "suggested"

✔ Fixed.

B40. "is hard to compute" is too colloquial: rephrase.

✔ Fixed: We have rephrased this.

--16

B41. A dot is missing at the end of item 3.

✔ Fixed.

B42. "to define a sound bisimulation closure": see above.

✔ Fixed.

B43. "and substitute over x": rephrase.
✔ Fixed: We have rephrased this.

B44. About Example 4. When reading the discussion involving P1 and P2, one has the impression that we are running into an infortunate situation, because, "by accident", we end up with twice the same process. This is reminiscent of phenomena of aliasing, where a name occurs twice "by accident". You should comment on that, and on why the difficulty you are facing cannot be solved by simply using fresh names when interacting with the observed processes. Given the important role played by the trigger value, I suggest to present its definition more formally, and not along the text (in the current version, it is presented in the introduction).
✔ Fixed: We have adopted the reviewer's suggestion of presenting the trigger value more formally.

B45. The last sentence of the example ("In conclusion, ..") should be rephrased: what we see here is just that having either only trigger or only characteristic values is not enough. It would be nice to have a more thorough discussion about what we obtain if we stick to just one of those. For instance, is there a subcalculus in which the equivalence we obtain using only one kind of process is meaningful?
- Please refer to our answer to comment B10.

B46. Finally, the ideas exposed in this example are crucial in the definition of characteristic bisimulation. Most of the example could be presented in the introduction, in order to give the general intuitions.
- Moving the example earlier to the paper is a reasonable option, but we prefer keeping the current structure. Notice that Section 2 already gives the intuitions for characteristic bisimulation.

B47. Definition 13. The presentation could be improved, by adopting a grammar for transmitted values (cf. the premise involving disjunctions), and by clarifying how the relation |-\ell-> is defined (in particular, why defining a derived LTS on top of the previous one, and not defining a new LTS, which shares some rules with the previous one?).
✔ Now it's Defnition 15. Fixed as suggested.

B48. "bisimularity"
✔ Fixed.

B49. [?(U);end]^s: there is a problem in the notation.
✔ Fixed.

--18

B50. You should explain why you introduce Higher-Order bisimilarity (Def.15), instead of sticking to Characteristic bisimilarity (Def.16). As it is, the reader has the impression that the former is just a variant of the latter, and does not bring much. This impression is confirmed by the fact that triggers exhibit similar behaviour, as stated in the proof sketch of Lemma 2. By reading these paragraphs, one wonders whether there is a point in considering both equivalences. If there is a point in introducing both, you should explain this. Otherwise, I think that the paper would make a stronger point by concentrating on only one equivalence (seen as "the answer" to the question you are studying). As said above, if HO bisimilarity is a useful tool to reason about Characteristic bisimilarity, which seems to be the case, this should be explained more clearly.

- As explained in our response to comment A27, we have added Remark 1 (before Section 5.7) to discuss the differences between (and (dis)advantages of) characteristic and higher-order bisimulation.

B51. The statement of Definition 17 could be simplified, by imposing that a derivation contains an occurrence of the tau rule (resp. an app rule).

- We prefer to keep the current definition (Definition 19).

B52. Subsection 5.7 contains really standard material, it could be made shorter.

- The content of Sub-section 5.7 is not really standard material for typed bisimilarities based on session types. Moreover, the results stated here are crucial in our proofs and we keep them as they are.

--19

B53. "it allows us to use ~~H and ~~C": I suggest to remove "~~H and", to stress the fact that the relation you put forward is ~~C.

- We do put forward characteristic bisimilarity, but as now clarified by Remark 1, higher-order bisimilarity offers certain advantages (and is key in our technical results).

--20

B54. "the closures of ~~H": weird terminology, see above.

- We have fixed this; see our reply to comment B9.

B55. The result given by Lemma 2 plays an important role in your contribution. I suggest to give a more detailed exposition than the intuitions that you present here. I moreover find that the explanations you give do not bring much, actually: the fact that the transition on top of page 20 can mimick the transition on the bottom of page 19 comes as no surprise, and does not tell why the Lemma holds.

✔ Fixed. We extended the explanations and added more proofs in Appendix.

B56. Use only one terminology between "contextual bisimilarity" and "context bisimilarity".

✔ Fixed.

B57. "implies the output clause of ≈ , i.e. the output clause requirement for ≈ H :" there is no need for this repetition.

✔ Fixed: We have revised this.

--21

B58. "The test process" -> "The test processes"

✔ Fixed.

B59. Proof of Lemma 6: explain where Delta_3 and Delta_4 come from (as remarked above, a similar remark can be made in several places in the paper and in the appendix).

✔ Fixed. Now it’s Lemma 7.

B60. "With further reasoning on (8)" is too vague: explain how the reasoning goes, if this is worth explaining.

✔ Fixed.

B61. Full proofs are not given in the main body of the text. Overall, the level of detail, from Lemma 2 on, is satisfactory. However, the reasoning steps you chose to describe are as expected, and nothing really surprising is presented. I suggest reworking the presentation in order to highlight the places in the proof where the design choices involving characteristic values and trigger values are at work. This could help the reader understanding, at a technical level, what I see as the main contribution of this paper (namely, the definition of characteristic bisimilarity relying on characteristic and trigger values).
✔ Fixed. We added the explanations as suggested. See Lemma 3 and Lemma 4 in the main text.

B62. Theorem 2 (and elsewhere): the numbering of Lemmas, Definitions etc. should be made more uniform.
- We are following the numbering implemented by the journal's Latex style.

--22
B63. "can equate... [23]." This is difficult to follow, you should rephrase.
✔ Fixed.

B64. "reasoned a tail-call optimisation"
✔ Fixed.

B65. "encoding of the lambda-calculi" -> "encoding of the lambda-calculus"
✔ Fixed.

--23
B66. "to reason secrecy"
✔ Fixed.

--24

B67. "We describe the transitions required to check the bisimilarity of this process with itself.": there is no point in proving that a process is equivalent to itself. You should rather say that you just compare transitions involved in an equivalence proof in the two settings.
✔ Fixed.

B68. "more economical closures": rephrase
✔ Fixed.

B69. "reveals" -> "illustrates", or "suggests"
✔ Fixed.

B70. About the comparison presented on Figure 7: it is helpful to compare your LTS and [14] (and this could be presented above, when introducing the LTS). However, counting the number of visible transitions does not really help. There is a problem of layout before line (6).

- This is Figure 9 now. Notice that the comparison involves counting the number of transitions, but also illustrates the process closures induced by the bisimilarities, and the savings in such closures that are possible thanks to linearity. We have revised the layout of the figure.

--25

B71. "inform and enable.. developments.": you should elaborate on that; the current sentence is too vague.

✔ Fixed: We have rephrased this.

B72. "innovations, including... (cf. Lemma 1)": this goes as expected, and it is not an innovation.

- As we explain in our answer to comment B52, in our view the up-to technique given by Lemma 1 is novel for session typed processes.

B73. "an intermediate behavioural equivalence, called higher-order bisimilarity...": this is mentioned here for the first time (if I am not mistaken). As mentioned above, it would really help to explain the relation between the two equivalences much earlier in the paper.

✔ Fixed: The new Remark 1 now clarifies this aspect in Section 5.

--26

B74. You should go through the bibliographical references, in order to make them more complete, and obtain a uniform treatment of these.

✔ Fixed: We have revised and updated these references.

--30

B75. "coincides the reduction"

✔ Fixed.

B76. "since," -> "since"

✔ Fixed.

B77. "the fact that bisimulation is a congruence": where did you prove this for ~~H? alternatively, why is it obvious?

✔ Fixed.

--31

B78. "state and proof"

✔ Fixed.

B79. Proof of Lemma 10: where do Delta''1, Delta''2 come from? The proof should be rewritten, I found it very difficult to follow. In particular, the sentence "To prove that R is a characteristic... (14)." is quite misleading. Later on, one has the impression that to establish that two processes are HO-bisimilar, one just has to inspect the input transition along t. You should explain how the translations emanating from P' and Q' are dealt with. This confusion between which hypotheses hold and needs to be proved occurs in several proofs in this appendix.

✔ Fixed. Now it's Lemma 14. Following also suggestions from Reviewer 1, the proof has been split into two lemmas and has been re-written in a more clear way. See also our answers to previous comments above.

--32

B80. Given the role played by Lemma 11, I suggest to provide more explanations about its proof in the main text.

✔ Fixed. This Lemma is no longer used. We have added explanations.

B81. "Exist"
"Case P2\neq xn for some n." is a bit misleading.
"We create a closure:"
"verify the closure R."
"The case concludes": rephrase
"or C1=C2=0" shoud rather be "and C1=C2=0"

This lemma changed so these comments no longer apply.

B82. The point where you rely on (16) and (18) to yield the expected result seems important in your development: I suggest to present this part of the reasoning in the main text.

✔ Fixed. This comment on old Lemma 11 no longer applies. See above for further explanations on the novelties in this version.

--40

B83. "Definibility" -> "Definability"

✔ Fixed.

B84. "Assuming.. definable.": rephrase, to make it more clear.

✔ Fixed.

--41

B85. "We show that S is a congruence.": you actually show more than this.

✔ Fixed.

B86. You use the freshness of succ in the proof of Lemma 17, this should appear in the statement of the Lemma.

✔ Fixed. Now it's Lemma 21.

B87. "We show that C is a congruence": with respect to parallel composition?

✔ Fixed.

--42

B88. "we to do"

✔ Fixed.

--43

B89. "which by definition of R we get"

✔ Fixed.