GLOBALLY GOVERNED SESSION SEMANTICS

DIMITRIOS KOUZAPAS AND NOBUKO YOSHIDA

Imperial College London
e-mail address: Dimitrios.Kouzapas@imperial.ac.uk

Imperial College London
e-mail address: n.yoshida@imperial.ac.uk

ABSTRACT. This paper proposes a bisimulation theory based on multiparty session types where a choreography specification governs the behaviour of session typed processes and their observer. The bisimulation is defined with the observer cooperating with the observed process in order to form complete global session scenarios and usable for proving correctness of optimisations for globally coordinating threads and processes. The induced bisimulation is strictly more fine-grained than the standard session bisimulation. The difference between the governed and standard bisimulations only appears when more than two interleaved multiparty sessions exist. This distinct feature enables to reason real scenarios in the large-scale distributed system where multiple choreographic sessions need to be interleaved. The compositionality of the governed bisimilarity is proved through the soundness and completeness with respect to the governed reduction-based congruence. Finally, its usage is demonstrated by a thread transformation governed under multiple sessions in a real usecase in the large-scale cyberinfrustracture.

1. INTRODUCTION

Backgrounds. Modern society increasingly depends on distributed software infrastructure such as the backend of popular Web portals, global E-science cyberinfrastructure, e-healthcare and e-governments. An application in such distributed environments is typically organised into many components that communicate through message passing. Thus an application is naturally designed as a collection of interaction scenarios, or *multiparty sessions*, each following an interaction pattern, or *choreographic protocol*. The theory for multiparty session types [24] captures these two natural abstraction units, representing the situation where two or more multiparty sessions (choreographies) can interleave for a single point application, with each message clearly identifiable as belonging to a specific session.

1998 ACM Subject Classification: D.3.1: Programming Languages (Formal Definitions and Theory); F.3.2: Semantics of Programming Languages (Process models).

Key words and phrases: the π-calculus, session types, bisimulation, barbed reduction-based congruence, multiparty session types, governance, soundness and completeness.

University of Glasgow, Dimitrios.Kouzapas@glasgow.ac.uk (On leave).

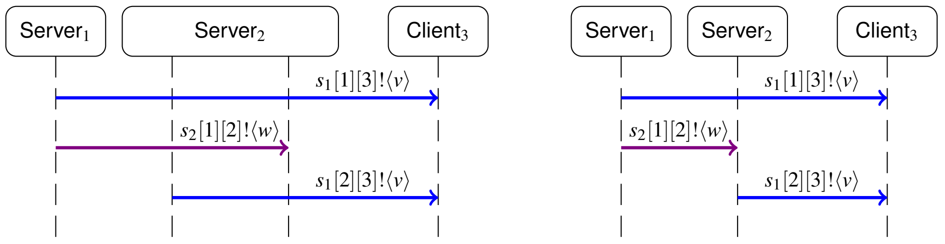
Server$_1$
Server$_2$
Client$_3$
$s_1[1][3]!\langle v\rangle$
$s_2[1][2]!\langle w\rangle$
$s_1[2][3]!\langle v\rangle$
Server$_1$
Server$_2$
Client$_3$
$s_1[1][3]!\langle v\rangle$
$s_2[1][2]!\langle w\rangle$
$s_1[2][3]!\langle v\rangle$

Figure 1: Resource Managment Example: (a) before optimisation; (b) after optimisation

This article introduces a behavioural theory which can reason about distributed processes that are controlled globally by multiple choreographic sessions. Typed behavioural theory has been one of the central topics of the study of the π-calculus throughout its history, for example, reasoning about various encodings into the typed π-calculi [42, 47]. Our theory treats the mutual effects of multiple choreographic sessions which are shared among distributed participants as their common knowledge or agreements, reflecting the origin of choreographic frameworks [9]. These features related to multiparty session type discipline make our theory distinct from any type-based bisimulations in the literature and also applicable to a real choreographic usecase from a large-scale distributed system. We say that our bisimulation is *globally governed*,since it uses global multiparty specifications to regulate the conversational behaviour of distributed processes.

Multiparty session types. To illustrate the idea for globally governed semantics, we first explain the mechanisms of multiparty session types [24]. Consider the simplest communication protocol where the participant implementing 1 sends a message of type bool to the participant implementing 2. A *global type* [24] is used to describe the protocol as:

$$G_1 \;=\; 1 \to 2 : \langle \mathtt{bool} \rangle.\mathtt{end}$$

where $\to$ denotes the flow of communication and end denotes protocol termination.

Global type G_1 is used as an agreement for the type-check specifications of both Server$_1$ and Server$_2$. We type-check implementations of both servers against the *projection* of G_1 into local session types. The following type

$$[2]!\langle \mathtt{bool} \rangle;\mathtt{end}$$

is the local session type from the point of view of participant 1, that describes the output of a bool-type value towards participant 2, while local type:

$$[1]?(\mathtt{bool});\mathtt{end}$$

is the local session type from the point of view of 2 that describes the input of a bool-type value from 1.

Resource management usecase. We will use a simplified usecase, UC.R2.13 “Acquire Data From Instrument”, c.f. § 6 of [1], to explain the main intuition of globally governed semantics and give insights on how our theory can reason about choreographic interactions.

Consider the scenario in Figure 1(a) where a single threaded Client$_3$ (participant 3) uses two services: from the single threaded Server$_1$ (participant 1) and from the dual threaded Server$_2$ (participant 2). In Figure 1, the vertical lines represent the threads in the participants. Additionally Server$_1$ sends an internal signal to Server$_2$. The communication patterns are described with the use of the following global protocols:

$$\begin{array}{rcl} G_a & = & 1 \to 3 : \langle ser \rangle.2 \to 3 : \langle ser \rangle.\mathtt{end} \\ G_b & = & 1 \to 2 : \langle sig \rangle.\mathtt{end} \end{array}$$

with G_a describing the communication between the two servers and the Client$_3$ and G_b describing the internal communication between the two servers. Participants $1,2$ and 3 are assigned to processes P_1, P_2 and P_3, respectively in order to implement a usecase as:

$$\begin{array}{rcl} P_1 & = & a[1](x).b[1](y).x[3]!\langle v \rangle;y[2]!\langle w \rangle;\mathbf{0} \\ P_2 & = & a[2](x).\overline{b}[2](y).(y[1]?(z);\mathbf{0} \mid x[3]!\langle v \rangle;\mathbf{0}) \\ P_3 & = & \overline{a}[3](x).x[1]?(z);x[2]?(y);\mathbf{0} \end{array}$$

Shared name a establishes the session corresponding to G_a, where Client$_3$ (P_3) uses prefix $\overline{a}[3](x)$ to initiate a session that involves three processes: Server$_1$ (P_1) and Server$_2$ (P_2) participate to the session with prefixes $a[1](x)$ and $a[2](x)$, respectively. In a similar way the session corresponding to G_b is established between Server$_1$ and Server$_2$.

The above scenario is subject to an optimisation due to the fact that the internal signal between Server$_1$ and Server$_2$ is invisible to clients because the communication link created after the session initiation is local. The optimisation is illustrated in Figure 1(b), where we require a single threaded service for Server$_2$ to avoid the overhead of an extra thread creation. The new implementation of participant 2 is:

$$R_2 = a[2](x).\overline{b}[2](y).y[1]?(z);x[3]!\langle v \rangle;\mathbf{0}$$

It is important to note that both P_2 and R_2 are typable under both G_a and G_b.

The motivation of this work is set when we compare the two server interfaces that are exposed towards Client$_3$, here implemented by process P_3. The two different interfaces are given by the two different implementations $P_1 \mid P_2$ and $P_1 \mid R_2$.

Untyped and linear bisimulations. It is obvious that in the untyped setting [45], $P_1 \mid P_2$ and $P_1 \mid R_2$ are *not* bisimilar since P_2 can exhibit the output action $s_a[2][3]!\langle v \rangle$ before the input action $s_b[2][1]?\langle w \rangle$ (assuming that variable x is substituted with session s_a and variable y is substituted with session s_b after the session initiation actions take place). More concretely we can analyse their transitions as follows where $a[\{1,2\}](s_a)$ is the label to start the session with Client$_3$ while $s_a[2][3]!\langle v \rangle$ is an output of value v from Server$_2$ to Client$_3$:

$$P_1 \mid P_2 \xrightarrow{a[\{1,2\}](s_a)} \xrightarrow{\tau} \xrightarrow{s_a[2][3]!\langle v \rangle}$$

$$P_1 \mid R_2 \xrightarrow{a[\{1,2\}](s_a)} \xrightarrow{\tau} \not\xrightarrow{s_a[2][3]!\langle v \rangle}$$

The same transitions are observed if we restrict the transition semantics to respect the traditional linearity principle based on session local types [30]. The full definition of the multiparty session bisimulation which follows the usual linearity property is found in § 4. We also give the detailed interaction patterns in Example 4.11 to explain why both untyped and linear bisimulations cannot equate $P_1 \mid P_2$ and $P_1 \mid R_2$.

Globally governed bisimulation. In the global setting, the behaviours of $P_1 \mid P_2$ and $P_1 \mid R_2$ are constrained by the specification of the global protocols, G_a and G_b. The service provided by Server$_2$ is available to Client$_3$ only after Server$_1$ sends a signal to Server$_2$. Protocol G_a dictates that action $s_a[2][3]!\langle v\rangle$ can only happen after action $s_b[2][1]?\langle w\rangle$ in P_2. This is because Client$_3$, which uses the service interface as a global observer, is also typed by the global protocol G_a and can only interact with action $s_a[2][3]!\langle v\rangle$ from Server$_1$ after it interacts with action $s_a[1][3]!\langle v\rangle$ from Server$_2$.

Hence in a globally typed setting, processes $P_1 \mid P_2$ and $P_1 \mid R_2$ are not distinguishable by Client$_3$ and thus the thread optimisation of R_2 is behaviourally justified.

Note that processes P_2 and R_2 (i.e. without the parallel composition P_1) are not observationaly equivalent under any set of session typed or untyped observational semantics. The governed bisimulation between $P_1 \mid P_2$ and $P_1 \mid R_2$ is achieved if we introduce an internal message of the session created on shared channel b between processes P_1 and P_2 and P_1 and R_2, respectively.

Changing a specification. A global protocol directly affects the behaviour of processes. We change global type G_a with global type:

$$G'_a = 2 \to 3 : \langle ser\rangle.1 \to 3 : \langle ser\rangle.\mathtt{end}$$

Processes $P_1 \mid P_2$ and $P_1 \mid R_2$ are also typable under protocol G'_a but now process R_2 can perform both the output to Client$_3$ and the input from Server$_1$ concurrently and according to the protocol G'_a that states that Client$_3$ can receive a message from Server$_2$ first. Hence $P_1 \mid P_2$ and $P_1 \mid R_2$ are no longer equivalent under global type G_a.

The above example gives an insight for our development of an equivalence theory that takes into account a global type as a specification. The interaction scenario between processes refines the behaviour of processes. To achieve such a theory of process equivalence we require to observe the labelled transitions together with the information provided by the global types. Global types define additional knowledge about how an observer (in the example above the observer is Client$_3$) will collaborate with the observed processes so that different global types (i.e. global witnesses) can induce the different behaviours.

Contributions and outline. This article introduces two classes of typed bisimulations based on multiparty session types. The first bisimulation definition is based on the typing information derived by local (endpoint) types, hence it resembles the standard linearity-based bisimulation for session types ([30]). The second bisimulation definition, which we call *globally governed session bisimilarity*, uses the information from global multiparty session types to derive the interaction pattern of the global observer. We prove that both bisimilarities coincide with a corresponding standard contextual equivalence [22] (see Theorems 4.10 and 5.15). The globally governed semantics give a more fine-grained bisimilarity equivalence comparing to the locally typed linear bisimulation relation. We identify the condition when the two semantic theories coincide (see Theorem 5.16). Interestingly our next theorem (Theorem 5.17) shows that both bisimilarity relations differ only when processes implement two or more interleaved global types. This feature makes the theory for govern multiparty bisimulation applicable to real situations where multiple choreographies are used to compose a single, large application. We demonstrate the use of governed bisimulation through the running example, which is applicable to a thread optimisation of a real usecase from a large scale distributed system [1].

This article is a full version of the extended abstract published in [29] and the first author's thesis [28]. Here we include the detailed definitions, expanded explanations, full reasoning of the usecases and complete proofs. The rest of the paper is organised as follows: Section 2 introduces a synchronous version of the calculus for the multiparty sessions. Section 3 defines a typing system

$$
\begin{array}{lllr|llr}
P & ::= & \overline{u}[\mathtt{p}](x).P & \text{Request} & \mid & \mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q & \text{Conditional} \\
 & \mid & u[\mathtt{p}](x).P & \text{Accept} & \mid & P \mid Q & \text{Parallel} \\
 & \mid & c[\mathtt{p}]!\langle e\rangle;P & \text{Sending} & \mid & \mathbf{0} & \text{Inaction} \\
 & \mid & c[\mathtt{p}]?(x);P & \text{Receiving} & \mid & (\nu\ n)P & \text{Hiding} \\
 & \mid & c[\mathtt{p}]\oplus l;P & \text{Selection} & \mid & \mu X.P & \text{Recursion} \\
 & \mid & c[\mathtt{p}]\&\{l_i : P_i\}_{i\in I} & \text{Branching} & \mid & X & \text{Variable}
\end{array}
$$

$$
\begin{array}{lllrlllr}
u & ::= & x \mid a & \text{Identifier} & c & ::= & s[\mathtt{p}] \mid x & \text{Session} \\
n & ::= & s \mid a & \text{Name} & v & ::= & a \mid \mathtt{tt} \mid \mathtt{ff} \mid s[\mathtt{p}] & \text{Value} \\
e & ::= & v \mid x \mid e = e' \mid \ldots & & & & & \text{Expression}
\end{array}
$$

Figure 2: Syntax for synchronous multiparty session calculus

and proves its subject reduction theorem. Section 4 presents the typed behavioural theory based on the local types for the synchronous multiparty sessions and proves that the typed bisimulation and reduction-based barbed congruence coincide. Section 5 introduces the semantics for globally governed behavioural theory and proves the three main theorems of this article. Section 6 presents a real-world usecase based on UC.R2.13 "Acquire Data From Instrument" from the Ocean Observatories Initiative (OOI) [1] and shows that the governed bisimulation can justify an optimisation of network services. Finally, Section 7 concludes with the related work. The appendix includes the full proofs.

Acknowledgement. The first author is funded by EPSRC Post Doctal Fellowship. The work has been partially sponsored by the Ocean Observatories Initiative, EPSRC EP/K011715/1, EP/K034413/1, EP/G015635/1 and EP/L00058X/1, and EU project FP7-612985 UpScale. We thank Rumyana Neykova for her suggestion of the usecase.

2. Synchronous Multiparty Sessions

This section defines a synchronous version of the multiparty session types. The syntax and typing follow the work in [7] without the definition for session endpoint queues (which are used for asynchronous communication). We choose to define our theory in the synchronous setting, since it allows the simplest formulations for demonstrating the essential concepts of bisimulations. An extension of the current theory to the asynchronous semantics is part of the work in [28], where session endpoint queues are used to provide asynchronous interaction patterns between session endpoints.

2.1. **Syntax.**

The syntax of the synchronous multiparty session calculus is defined in Figure 2. We assume two disjoint countable sets of names: one ranges over *shared names* $a, b, \ldots$ and another ranges over *session names* $s, s_1, \ldots$. Variables range over $x, y, \ldots$. *Roles* range over the natural numbers and are denoted as $\mathtt{p}, \mathtt{q}, \ldots$, labels range over $l, l_1, \ldots$ and constants range over $\mathtt{tt}, \mathtt{ff}, \ldots$. In general, when the sessions are interleaved, a participant may implement more than one roles. We often call $\mathtt{p}, \mathtt{q}, \ldots$ participants when there is no confusion. Symbol u is used to range over shared names or variables. Session endpoints are denoted as $s[\mathtt{p}], s[\mathtt{q}], \ldots$. The symbol c ranges over session endpoints or variables. Values range over shared names, constants (we can extend

$$P \equiv P \mid \mathbf{0} \qquad P \mid Q \equiv Q \mid P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R) \qquad P \equiv_\alpha Q \qquad \mu X.P \equiv P\{\mu X.P/X\}$$
$$(\nu\ n)(\nu\ n')P \equiv (\nu\ n')(\nu\ n)P \qquad (\nu\ n)\mathbf{0} \equiv \mathbf{0} \qquad (\nu\ n)(P) \mid Q \equiv (\nu\ n)(P \mid Q) \quad n \notin \mathtt{fn}(Q)$$

Figure 3: Structural Congruence for Synchronous Multiparty Session Calculus

$$\begin{array}{rcll}
\overline{a}[n](x).P_n \mid \Pi_{i=\{1,..,n-1\}} a[i](x).P_i & \longrightarrow & (\nu\ s)(P_n\{s[n]/x\} \mid \Pi_{i=\{1,..,n-1\}} P_i\{s[i]/x\}) & [\text{Link}] \\
s[\mathtt{p}][\mathtt{q}]!\langle e\rangle;P \mid s[\mathtt{q}][\mathtt{p}]?(x);Q & \longrightarrow & P \mid Q\{v/x\} \quad (e \downarrow v) & [\text{Comm}] \\
s[\mathtt{p}][\mathtt{q}] \oplus l_k;P \mid s[\mathtt{q}][\mathtt{p}]\&\{l_i : P_i\}_{i\in I} & \longrightarrow & P \mid P_k \quad (k \in I) & [\text{Label}]
\end{array}$$
$$\mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q \longrightarrow P \quad (e \downarrow \mathtt{tt}) \quad [\text{If-F}] \quad \mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q \longrightarrow Q \quad (e \downarrow \mathtt{ff}) \quad [\text{If-T}]$$
$$\frac{P \longrightarrow P'}{(\nu\ n)P \longrightarrow (\nu\ n)P'}\ [\text{Res}] \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}\ [\text{Par}] \qquad \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}\ [\text{Str}]$$

Figure 4: Operational semantics for synchronous multiparty session calculus

constants to include natural numbers $1,2,\dots$) and session endpoints $s[\mathtt{p}]$. Expressions $e, e', \dots$ are either values, logical operations on expressions or name matching operations $n = n'$.

The first two prefixes are the primitives for session initiation: $\overline{u}[\mathtt{p}](x).P$ initiates a new session through identifier u (which represents a shared interaction point) on the other multiple participants, each of the shape $u[\mathtt{q}](x).Q_\mathtt{q}$ where $1 \leq \mathtt{q} \leq \mathtt{p}-1$. The (bound) variable x will be substituted with the channel used for session communication. The basic session endpoint communication (i.e. the communication that takes place between two endpoints) is performed with the next two pairs of prefixes: the prefixes for sending ($c[\mathtt{p}]!\langle e\rangle;P$) and receiving ($c[\mathtt{p}]?(x);P$) a value and the prefixes for the selecting ($c[\mathtt{p}] \oplus l;P$) and branching ($c[\mathtt{p}]\&\{l_i : P_i\}_{i\in I}$) processes, where the former prefix chooses one of the branches offered by the latter prefix. Specifically, process $c[\mathtt{p}]!\langle e\rangle;P$ denotes the intention of sending a value to role $\mathtt{p}$; in a similar way, process $c[\mathtt{p}]?(x);P$ denotes the intention of receiving a value from role $\mathtt{p}$. A similar interaction pattern holds for the selection/branching pair of communication, with the difference that the intention is to send (respectively receive) a label that eventually determines the reduction of the branching process. Process $\mathbf{0}$ is the inactive process. The conditional process $\mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q$ offers processes P and Q according to the evaluation of expression e. Process $P \mid Q$ is the parallel composition, while $(\nu\ n)P$ restricts name n in the scope of P. We use the primitive recursor $\mu X.P$ with X as the recursive variable. We write $\mathtt{fn}(P)/\mathtt{bn}(P)$ and $\mathtt{fv}(P)/\mathtt{bv}(P)$ to denote the set of free/bound names and free/bound variables, respectively in process P. A process is closed if it is a term with no free variables.

2.2. **Operational semantics.** The operational semantics is defined in Figure 4. It uses the usual structure congruence relation (denoted by $\equiv$), which is defined as the least congruence relation that respects the rules in Figure 3. Structural congruence is an associative and commutative monoid over the parallel ($\mid$) operation with the inactive process ($\mathbf{0}$) being the neutral element. It respects alpha-renaming and recursion unfolding. The order of the name restriction operators has no effect with respect to structural congruence. Finally, the scope opening rule extends the scope of the restriction on a name from a process to a paralleled process, provided that the name does not occur free in the latter. We often write $(\nu\ n_1 n_2 \cdots n_m)(P)$ for $(\nu\ n_1)((\nu\ n_2)(\cdots(\nu\ n_m)(P)))$.

Global

$$\begin{array}{lcll}
G & ::= & \mathtt{p} \to \mathtt{q} : \langle U \rangle . G' & \text{exchange} \\
 & \mid & \mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i \in I} & \text{branching} \\
 & \mid & \mu \mathtt{t}. G & \text{recursion} \\
 & \mid & \mathtt{t} & \text{variable} \\
 & \mid & \mathtt{end} & \text{end}
\end{array}$$

Exchange

$$U \quad ::= \quad S \mid T$$

Sort

$$S \quad ::= \quad \mathtt{bool} \mid \langle G \rangle$$

Local

$$\begin{array}{lcll}
T & ::= & [\mathtt{p}]!\langle U \rangle ; T & \text{send} \\
 & \mid & [\mathtt{p}]?(U); T & \text{receive} \\
 & \mid & [\mathtt{p}] \oplus \{l_i : T_i\}_{i \in I} & \text{select} \\
 & \mid & [\mathtt{p}] \& \{l_i : T_i\}_{i \in I} & \text{branch} \\
 & \mid & \mu \mathtt{t}. T & \text{recursion} \\
 & \mid & \mathtt{t} & \text{variable} \\
 & \mid & \mathtt{end} & \text{end}
\end{array}$$

Figure 5: Global and local types

The reduction semantics of the calculus is defined in Figure 4. Rule [Link] defines synchronous session initiation and requires that all session endpoints must be present for a synchronous reduction, where each role p creates a session endpoint $s[\mathtt{p}]$ on a fresh session name s. The maximum participant with the maximum role $(\overline{a}[n](x).P)$ is responsible for requesting a session initiation. Rule [Comm] describes the semantics for sending a value to the corresponding receiving process. We assume an evaluation context $e \downarrow v$, where expression e evaluates to value v. Session endpoint $s[\mathtt{p}]$ sends a value v to session endpoint $s[\mathtt{q}]$, which in its turn waits to receive a value from role p. The interaction between selection and branching processes is defined by rule [Label], where we expect the selection part $s[\mathtt{p}]$ of the interaction to choose the continuation on the branching part $s[\mathtt{q}]$ of the interaction.

The rest of the rules follow the usual π-calculus rules. Rule [If] evaluates the boolean expression e. If the latter is true it proceeds with the first branch process defined; otherwise it proceeds with the second branch process defined. Rules [Res] and [Par] are inductive rules on the parallel and name restriction operators. Finally, rule [Str] closes the reduction relation under structural congruence closure. We write $\twoheadrightarrow$ for $(\longrightarrow \cup \equiv)^*$ [22,45].

3. Typing for Synchronous Multiparty Sessions

This section defines a typing system for the synchronous multiparty session calculus. The system is a synchronous version of one presented in [7]. We first define multiparty session types and then summarise the typing system for the synchronous multiparty session calculus. At the end of the section, we prove the subject reduction theorem (Theorem 3.9).

3.1. **Global and local types.**

We first give the definition of the global type and then define the local session type as a projection of the global type.

Global types, ranged over by $G, G', \ldots$ describe the whole conversation scenario of a multiparty session as a type signature. The grammar of the global types is given in Figure 5 (left).

The global type $\mathtt{p} \to \mathtt{q} : \langle U \rangle . G'$ describes the interaction where role p sends a message of type U to role q and then the interaction described in G' takes place. The *exchange type* $U, U', \ldots$ consists of *sort* types $S, S', \ldots$ for values (either base types or global types), and *local session* types $T, T', \ldots$ for channels (local types are defined in the next paragraph). Type $\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i \in I}$ describes the interaction where role p selects one of the labels l_i against role q. If l_j is selected, the interaction described in G_j takes place. We assume that $\mathtt{p} \neq \mathtt{q}$. Type $\mu\mathtt{t}.G$ is the recursive type. Type variables $(\mathtt{t}, \mathtt{t}', \dots)$ are guarded, i.e., type variables only appear under some prefix and do not appear free in the exchange types U. We take an *equi-recursive* view of recursive types, not distinguishing between $\mu\mathtt{t}.G$ and its unfolding $G\{\mu\mathtt{t}.G/\mathtt{t}\}$ [41, §21.8]. We assume that G in the grammar of sorts has no free type variables (i.e. type variables do not appear freely in carried types in exchanged global types). Type $\mathtt{end}$ represents the termination of the session.

Local types. Figure 5 (right) defines the syntax of local types. They correspond to the communication actions, representing sessions from the view-point of a single role. The *send type* $[\mathtt{p}]!\langle U\rangle;T$ expresses the sending to $\mathtt{p}$ of a value of type U, followed by the communications of T. Similarly, the *select type* $[\mathtt{p}] \oplus \{l_i : T_i\}_{i\in I}$ represents the transmission of label l_i to role $\mathtt{p}$. Label l_i is chosen in the set $\{l_i\}_{i\in I}$ and the selection prefix is followed by the communications described by T_i. The *receive* and *branch types* are described as dual types for the send and select types, respectively. In the receive type $[\mathtt{p}]?(U);T$ we expect that the typed process will receive a value of type U from role $\mathtt{p}$ while in the branch type $[\mathtt{p}]\&\{l_i : T_i\}_{i\in I}$ we expect a selection label $l_j \in \{l_i\}_{i\in I}$ from role $\mathtt{p}$. The rest of the local types are the same as global types. Recursion $\mu\mathtt{t}.T$ uses type variables $\mathtt{t}$ to perform a recursion via substitution $T\{\mu\mathtt{t}.T/\mathtt{t}\}$. The inactive type is written as $\mathtt{end}$.

We define the roles occurring in a global type and the roles occurring in a local type.

Definition 3.1 (Roles)**.**

- We define $\mathtt{roles}(G)$ as the set of roles in protocol G:

$$\mathtt{roles}(\mathtt{end}) = \emptyset \qquad \mathtt{roles}(\mathtt{t}) = \emptyset \qquad \mathtt{roles}(\mu\mathtt{t}.G) = \mathtt{roles}(G)$$
$$\mathtt{roles}(\mathtt{p} \to \mathtt{q} : \langle U\rangle.G) = \{\mathtt{p},\mathtt{q}\} \cup \mathtt{roles}(G)$$
$$\mathtt{roles}(\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I}) = \{\mathtt{p},\mathtt{q}\} \cup \{\mathtt{roles}(G_i) \mid i \in I\}$$

- We define $\mathtt{roles}(T)$ on local types as:

$$\mathtt{roles}(\mathtt{end}) = \emptyset \qquad \mathtt{roles}(\mathtt{t}) = \emptyset \qquad \mathtt{roles}(\mu\mathtt{t}.T) = \mathtt{roles}(T)$$
$$\mathtt{roles}([\mathtt{p}]!\langle U\rangle;T) = \{\mathtt{p}\} \cup \mathtt{roles}(T) \qquad \mathtt{roles}([\mathtt{p}]?(U);T) = \{\mathtt{p}\} \cup \mathtt{roles}(T)$$
$$\mathtt{roles}([\mathtt{p}] \oplus \{l_i : T_i\}_{i\in I}) = \{\mathtt{p}\} \cup \{\mathtt{roles}(T_i) \mid i \in I\}$$
$$\mathtt{roles}([\mathtt{p}]\&\{l_i : T_i\}_{i\in I}) = \{\mathtt{p}\} \cup \{\mathtt{roles}(T_i) \mid i \in I\}$$

Global and local projections. The relation between global and local types is formalised by the usual projection function [7, 24], where the projection of a global type G over a role $\mathtt{p}$ results in a local type T.[1]

[1] For a simplicity of the presentation, we take the projection function from [7, 24], which does not use the mergeability operator [15]. The extension does not affect to the whole theory.

Definition 3.2 (Global projection and projection set)**.** The projection of a global type G onto a role $\mathtt{p}$ results in a local type and it is defined by induction on G:

$$\mathtt{p}' \to \mathtt{q} : \langle U \rangle.G \lceil \mathtt{p} \quad = \quad \begin{cases} [\mathtt{q}]!\langle U \rangle; G\lceil \mathtt{p} & \mathtt{p} = \mathtt{p}' \\ [\mathtt{p}']?(U); G\lceil \mathtt{p} & \mathtt{p} = \mathtt{q} \\ G\lceil \mathtt{p} & \text{otherwise} \end{cases}$$

$$\mathtt{p}' \to \mathtt{q} : \{l_i : G_i\}_{i\in I} \lceil \mathtt{p} \quad = \quad \begin{cases} [\mathtt{q}] \oplus \{l_i : G_i \lceil \mathtt{p}\}_{i\in I} & \mathtt{p} = \mathtt{p}' \\ [\mathtt{p}'] \& \{l_i : G_i \lceil \mathtt{p}\}_{i\in I} & \mathtt{p} = \mathtt{q} \\ G_1 \lceil \mathtt{p} & \text{if } \forall j \in I.\ G_1 \lceil \mathtt{p} = G_j \lceil \mathtt{p} \end{cases}$$

$$(\mu \mathtt{t}.G)\lceil \mathtt{p} \quad = \quad \begin{cases} \mu \mathtt{t}.(G\lceil \mathtt{p}) & G\lceil \mathtt{p} \neq \mathtt{t} \\ \mathtt{end} & \text{otherwise} \end{cases}$$

$$\mathtt{t}\lceil \mathtt{p} = \mathtt{t} \qquad \mathtt{end}\lceil \mathtt{p} = \mathtt{end}$$

A *projection* of G is defined as $\mathtt{proj}(G) = \{G\lceil \mathtt{p} \mid \mathtt{p} \in \mathtt{roles}(G)\}$.

Inactive $\mathtt{end}$ and recursive variable $\mathtt{t}$ types are projected to their respective local types. Note that we assume global types have roles starting from 1 up to some n, without skipping numbers in between.

We project $\mathtt{p}' \to \mathtt{q} : \langle U \rangle.G$ to party $\mathtt{p}$ as a sending local type if $\mathtt{p} = \mathtt{p}'$ and as a receiving local type if $\mathtt{p} = \mathtt{q}$. In any case the continuation of the projection is $G\lceil \mathtt{p}$.

For $\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I}$, the projection is the select local type for $\mathtt{p} = \mathtt{p}'$ and the branch local type $\mathtt{p} = \mathtt{q}$. Otherwise we use the projection of one of the global types $\{G_i \mid i \in I\}$ (all types G_i should have the same projection with respect to $\mathtt{p}$).

The first side condition of the recursive type $\mu\mathtt{t}.G$ ensures that it does not project to an invalid local type $\mu\mathtt{t}.\mathtt{t}$.

We use the local projection function to project a local type T onto a role $\mathtt{p}$ to produce binary session types. We use the local projection to extract technical results (well-formed linear environment) later in this section.

We use the usual binary session types, [21, 50], for the definition of local projection:

Definition 3.3 (Binary session types)**.**

$$B \ ::= \ !\langle U \rangle; B \ \mid \ ?(U); B \ \mid \ \oplus\{l_i : B_i\} \ \mid \ \&\{l_i : B_i\} \ \mid \ \mu\mathtt{t}.B \ \mid \ \mathtt{t} \ \mid \ \mathtt{end}$$

Definition 3.4 (Local projection)**.** The projection of a local type T onto a role $\mathtt{p}$ is defined by induction on T:

$$[\mathtt{p}]!\langle U\rangle;T\lceil\mathtt{q} = \begin{cases} !\langle U\rangle;T\lceil\mathtt{q} & \mathtt{q}=\mathtt{p} \\ T\lceil\mathtt{q} & \text{otherwise}\end{cases}$$

$$[\mathtt{p}]?(U);T\lceil\mathtt{q} = \begin{cases} ?(U);T\lceil\mathtt{q} & \mathtt{q}=\mathtt{p} \\ T\lceil\mathtt{q} & \text{otherwise}\end{cases}$$

$$[\mathtt{p}]\oplus\{l_i:T_i\}_{i\in I}\lceil\mathtt{q} = \begin{cases} \oplus\{l_i:T_i\lceil\mathtt{q}\}_{i\in I} & \mathtt{q}=\mathtt{p} \\ T_1\lceil\mathtt{q} & \text{if } \forall i\in I.T_i\lceil\mathtt{q}=T_1\lceil\mathtt{q}\end{cases}$$

$$[\mathtt{p}]\&\{l_i:T_i\}_{i\in I}\lceil\mathtt{q} = \begin{cases} \&\{l_i:T_i\lceil\mathtt{q}\}_{i\in I} & \mathtt{q}=\mathtt{p} \\ T_1\lceil\mathtt{q} & \text{if } \forall i\in I.T_i\lceil\mathtt{q}=T_1\lceil\mathtt{q}\end{cases}$$

$$(\mu\mathtt{t}.T)\lceil\mathtt{p} = \begin{cases} \mu\mathtt{t}.(T\lceil\mathtt{p}) & T\lceil\mathtt{p}\neq\mathtt{t} \\ \mathtt{end} & \text{otherwise}\end{cases}$$

$$\mathtt{t}\lceil\mathtt{p}=\mathtt{t} \qquad \mathtt{end}\lceil\mathtt{p}=\mathtt{end}$$

An inactive local type, a recursive variable and a recursive type are projected to their corresponding binary session type syntax. Types $[\mathtt{p}]!\langle U\rangle;T$ and $[\mathtt{p}]?(U);T$ are projected with respect to $\mathtt{q}$ to binary send and receive session types, respectively under the condition $\mathtt{p}=\mathtt{q}$, and continue with the projection of T on $\mathtt{q}$. If $\mathtt{p}\neq\mathtt{q}$, the local projection continues with the projection of T. A similar argument is applied for $[\mathtt{p}]\oplus\{l_i:T_i\}_{i\in I}$ and $[\mathtt{p}]\&\{l_i:T_i\}_{i\in I}$ if $\mathtt{p}=\mathtt{q}$. For the case $\mathtt{p}\neq\mathtt{q}$, we project one of the continuations in $\{T_i\}_{i\in I}$ since we expect all the projections of $\{T_i\}_{i\in I}$ to be the same [7,24].

We inductively define the notion of duality as a relation over the projected local types:

Definition 3.5 (Duality)**.** We define the duality function over binary session types as:

$$\mathtt{end}=\overline{\mathtt{end}} \quad \mathtt{t}=\bar{\mathtt{t}} \quad \overline{\mu\mathtt{t}.B}=\mu\mathtt{t}.\overline{B} \quad \overline{!\langle U\rangle;B}=?(U);\overline{B} \quad \overline{?(U);B}=!\langle U\rangle;\overline{B}$$
$$\overline{\oplus\{l_i:B_i\}_{i\in I}}=\&\{l_i:\overline{B_i}\}_{i\in I} \quad \overline{\&\{l_i:B_i\}_{i\in I}}=\oplus\{l_i:\overline{B_i}\}_{i\in I}$$

We assume only session types with tail recursion as in [7,24] (note that the inductive duality on non-tail recursive session types, i.e. session prefixes that carry a recursive variable as an object) is shown to be unsound [6]).

The result of the following proposition is used on the well-formedness criteria of a linear environment.

Proposition 3.6. *If* $\mathtt{p},\mathtt{q}\in\mathtt{roles}(G)$ *with* $\mathtt{p}\neq\mathtt{q}$ *then* $(G\lceil\mathtt{p})\lceil\mathtt{q}=\overline{(G\lceil\mathtt{q})\lceil\mathtt{p}}$.

Proof. The proof is done by induction on the structure of global types. □

3.2. **Typing system.** We define the typing system for the synchronous multiparty session calculus. The typing judgements for expressions and processes are of the shapes:

$$\Gamma\vdash e:S \quad \text{and} \quad \Gamma\vdash P\triangleright\Delta$$

where Γ is the shared environment which associates variables to sort types (i.e. base types or global types), shared names to global types and process variables to session environments; and Δ is the

$$\Gamma\cdot u : S \vdash u : S \quad [\text{Name}] \qquad\qquad \Gamma \vdash \mathtt{tt}, \mathtt{ff} : \mathtt{bool} \quad [\text{Bool}]$$

$$\frac{\Gamma \vdash e_i : \mathtt{bool}}{\Gamma \vdash e_1 \ \mathtt{and}\ e_2 : \mathtt{bool}}\ [\text{And}] \qquad\qquad \frac{\Gamma \vdash n_1 : U \quad \Gamma \vdash n_2 : U}{\Gamma \vdash n_1 = n_2 : \mathtt{bool}}\ [\text{Match}]$$

$$\frac{\begin{array}{c}\Gamma \vdash a : \langle G\rangle \quad \Gamma \vdash P \triangleright \Delta\cdot x : G\lceil \mathtt{p}\\ \max(\mathtt{roles}(G)) = \mathtt{p}\end{array}}{\Gamma \vdash \overline{a}[\mathtt{p}](x).P \triangleright \Delta}\ [\text{MReq}] \qquad \frac{\begin{array}{c}\Gamma \vdash a : \langle G\rangle \quad \Gamma \vdash P \triangleright \Delta\cdot x : G\lceil \mathtt{p}\\ 1 \le \mathtt{p} < \max(\mathtt{roles}(G))\end{array}}{\Gamma \vdash a[\mathtt{p}](x).P \triangleright \Delta}\ [\text{MAcc}]$$

$$\frac{\Gamma \vdash e : S \quad \Gamma \vdash P \triangleright \Delta\cdot c : T}{\Gamma \vdash c[\mathtt{q}]!\langle e\rangle;P \triangleright \Delta\cdot c : [\mathtt{q}]!\langle S\rangle;T}\ [\text{Send}] \qquad \frac{\Gamma\cdot x : S \vdash P \triangleright \Delta\cdot c : T}{\Gamma \vdash c[\mathtt{q}]?(x);P \triangleright \Delta\cdot c : [\mathtt{q}]?(S);T}\ [\text{Recv}]$$

$$\frac{\Gamma \vdash P \triangleright \Delta\cdot c : T}{\Gamma \vdash c[\mathtt{q}]!\langle c'\rangle;P \triangleright \Delta\cdot c : [\mathtt{q}]!\langle T'\rangle;T\cdot c' : T'}\ [\text{Deleg}] \qquad \frac{\Gamma \vdash P \triangleright \Delta\cdot c : T\cdot x : T'}{\Gamma \vdash c[\mathtt{q}]?(x);P \triangleright \Delta\cdot c : [\mathtt{q}]?(T');T}\ [\text{SRecv}]$$

$$\frac{\Gamma \vdash P \triangleright \Delta\cdot c : T}{\Gamma \vdash c[\mathtt{q}]\oplus l;P \triangleright \Delta\cdot c : [\mathtt{q}]\oplus\{l : T\}}\ [\text{Sel}] \qquad \frac{\Gamma \vdash P_i \triangleright \Delta\cdot c : T_i \quad \forall\, i\in I}{\Gamma \vdash c[\mathtt{q}]\&\{l_i : P_i\}_{i\in I} \triangleright \Delta\cdot c : [\mathtt{q}]\&\{l_i : T_i\}_{i\in I}}\ [\text{Bra}]$$

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1 \quad \Gamma \vdash P_2 \triangleright \Delta_2 \quad \mathtt{dom}(\Delta_1)\cap\mathtt{dom}(\Delta_2) = \emptyset}{\Gamma \vdash P_1 \mid P_2 \triangleright \Delta_1\cdot\Delta_2}\ [\text{Conc}] \qquad \frac{\Gamma \vdash e : \mathtt{bool} \quad \Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash Q \triangleright \Delta}{\Gamma \vdash \mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q \triangleright \Delta}\ [\text{If}]$$

$$\Gamma \vdash \mathbf{0} \triangleright \emptyset \quad [\text{Inact}] \qquad\qquad \frac{\Gamma \vdash \mathbf{0} \triangleright \Delta}{\Gamma \vdash \mathbf{0} \triangleright \Delta\cdot c : \mathtt{end}}\ [\text{Complete}]$$

$$\frac{\Gamma\cdot a : \langle G\rangle \vdash P \triangleright \Delta}{\Gamma \vdash (\nu\ a)P \triangleright \Delta}\ [\text{NRes}] \qquad \frac{\begin{array}{c}\mathtt{fco}(\{s[1] : T_1 \ldots s[n] : T_n\})\\ \Gamma \vdash P \triangleright \Delta\cdot s[1] : T_1 \ldots s[n] : T_n\end{array}}{\Gamma \vdash (\nu\ s)P \triangleright \Delta}\ [\text{SRes}]$$

$$\Gamma\cdot X : \Delta \vdash X \triangleright \Delta \quad [\text{Var}] \qquad\qquad \frac{\Gamma\cdot X : \Delta \vdash P \triangleright \Delta}{\Gamma \vdash \mu X.P \triangleright \Delta}\ [\text{Rec}]$$

Figure 6: Typing system for synchronous multiparty session calculus

session environment (or linear environment) which associates channels to session types. Formally we define:

$$\Gamma \ ::=\ \emptyset \ \mid\ \Gamma\cdot u : S \ \mid\ \Gamma\cdot X : \Delta \quad \text{and} \quad \Delta \ ::=\ \emptyset \ \mid\ \Delta\cdot c : T$$

assuming we can write $\Gamma\cdot u : S$ if $u \notin \mathtt{dom}(\Gamma)$. We extend this to a concatenation for typing environments as $\Delta\cdot\Delta' = \Delta\cup\Delta'$. We use the following definition to declare the coherence of session environments.

Definition 3.7 (Coherency)**.** Typing Δ is *coherent with respect to session s* (notation $\mathtt{co}(\Delta(s))$) if for all $s[\mathtt{p}] : T_{\mathtt{p}} \in \Delta$ there exists $s[\mathtt{q}] : T_{\mathtt{q}} \in \Delta$ such that $T_{\mathtt{p}}\lceil \mathtt{q} = \overline{T_{\mathtt{q}}\lceil \mathtt{p}}$. A typing Δ is *coherent* (notation $\mathtt{co}(\Delta)$) if it is coherent with respect to all s in its domain. We say a typing Δ is *fully coherent* (notation $\mathtt{fco}(\Delta)$) if it is coherent and if $s[\mathtt{p}] : T_{\mathtt{p}} \in \Delta$ then for all $\mathtt{q} \in \mathtt{roles}(T_{\mathtt{p}})$, $s[\mathtt{q}] : T_{\mathtt{q}} \in \Delta$.

Figure 6 defines the typing system. The typing rules presented here are essentially identical to the communication typing system for programs in [7], due to the fact that our calculus is synchronous (i.e. we do not use session endpoint queues).

Rule [Name] types a shared name or a shared variable in environment Γ. Rule [Bool] assigns the type $\mathtt{bool}$ to constants $\mathtt{tt}, \mathtt{ff}$. Logical expressions are also typed with the $\mathtt{bool}$ type via rule [And], etc. Rule [Match] ensures that the name matching operator has the boolean type.

Rules [MReq] and [MAcc] type the request and accept processes, respectively. Both rules check that the type of session variable x agrees with the global type of the session initiation name a that is projected on the corresponding role $\mathtt{p}$. Furthermore, in rule [MReq] we require that the initiating role $\mathtt{p}$ is the maximum among the roles of the global type G of a while in rule [MAcc] we require that role $\mathtt{p}$ is less than the maximum role of global type G. Note that these session initiation rules allow processes to contain some but not all of the roles in a session. Rules [MReq] and [MAcc] ensure that a parallel composition of processes that implement all the roles in a session can proceed with a sound session initiation.

Rules [Send] and [Recv] are used to type the send, $c[\mathtt{p}]!\langle v\rangle;P$, and receive, $c[\mathtt{p}]?(x);P$, session prefixes. Both rules prefix the local type of c in the linear environment the send type $[\mathtt{p}]!\langle U\rangle;T$, and receive type $[\mathtt{p}]?(U);T$, respectively. The typing is completed with the check of the object types v and x, respectively, in the shared environment Γ. The delegation of a session endpoint is typed under rules [Deleg] and [Srecv]. Both rules prefix the local type of c in the linear environment with the send and receive prefixes, respectively, in a similar way with rules [Send] and [Recv]. They check the type for the delegating object in the linear environment Δ, and a delegation respects the linearity of the delegating endpoint (in this case c' and x, respectively), i.e. when an endpoint is sent ([Deleg]) it should not be present in the linear environment of the continuation P. Similarly, when an endpoint is received ([Srecv]) the receiving endpoint should not be present before the reception.

Rules [Sel] and [Bra] type selecting and branching processes, respectively. A selection prefix is typed on a select local type, while a branching prefix is typed on a branch local type. A selection prefix with label l for role c uses label l to select the continuation on name c in the select local type. A branching process with labels l_i branches the local types of c in the corresponding P_i in the branch local type. Furthermore, all P_i processes should have the same linear type on names other than c.

Rule [Conc] types a parallel composition of processes. The disjointness condition on typing environments Δ_1 and Δ_2 ensures the linearity of the environment $\Delta_1 \cdot \Delta_2$. Rule [If] types conditional process, where we require that the expression e to be of `bool` type and that the branching processes have the same linear environment. Rule [Inact] types the empty process with the empty linear environment. Rule [Complete] does an explicit weakening on the linear typing of an inaction process to achieve a complete linear environment. A complete linear environment is defined as the linear typing where every session endpoint is mapped to the inactive local type end. Rule [Nres] defines the typing for shared name restriction. A restricted shared name should be present in the shared environment Γ before restriction and should not appear in Γ after the restriction. Rule [Sres] uses the full coherency property to restrict a session name. Full coherency requires that all session endpoints of a session are present in the linear environment before restriction and furthermore, it requires that their local projections are mutually dual. A restricted session name does not appear in the domain of the linear environment.

Rule [Var] returns the linear environment Δ that is assigned to process variable X in environment Γ. Rule [Rec] checks that the process P under the recursion has the same linear environment Δ as the recursion variable X.

Further examples of typing and typable processes can be found in [11].

Finally, we call the typing judgement $\Gamma \vdash P \triangleright \Delta$ *coherent* if $\mathtt{co}(\Delta)$.

3.3. **Type soundness.**

We proceed with the proof of a subject reduction theorem to show the soundness of the typing system.

Before we state the subject reduction theorem we define the reduction semantics for local types extended to include session environments. The reduction on a session environment of a process shows the change on the session environment after a possible reduction on the process. We use the approach from [7, 24] to define session environment reduction.

Definition 3.8 (Session environment reduction)**.**

(1) $\{s[\mathrm{p}] : [\mathrm{q}]!\langle U\rangle; T \cdot s[\mathrm{q}] : [\mathrm{p}]?(U); T'\} \longrightarrow \{s[\mathrm{p}] : T \cdot s[\mathrm{q}] : T'\}$.
(2) $\{s[\mathrm{p}] : [\mathrm{q}] \oplus \{l_i : T_i\}_{i\in I} \cdot s[\mathrm{q}] : [\mathrm{p}]\&\{l_j : T'_j\}_{j\in J}\} \longrightarrow \{s[\mathrm{p}] : T_k \cdot s[\mathrm{q}] : T'_k\} \quad I \subseteq J, k \in I$.
(3) $\Delta \cup \Delta' \longrightarrow \Delta \cup \Delta''$ if $\Delta' \longrightarrow \Delta''$.

Note that the second rule of the session environment reduction makes the reduction $\Delta \longrightarrow^* \Delta'$ non-deterministic (i.e. not always confluent). The typing system satisfies the subject reduction theorem [7]:

Theorem 3.9 (Subject reduction)**.** *If* $\Gamma \vdash P \triangleright \Delta$ *is coherent and* $P \twoheadrightarrow P'$ *then* $\Gamma \vdash P' \triangleright \Delta'$ *is coherent with* $\Delta \longrightarrow^* \Delta'$.

Proof. See Appendix A. □

4. Synchronous Multiparty Session Semantics

This section presents the session typed behavioural theory for synchronous multiparty sessions. The typed bisimulation uses a labelled transition system (LTS) on environment tuples (Γ, Δ) to control the behaviour of untyped processes. The LTS on environments introduces a constrain that captures accurately multiparty session interactions and lies at the heart of the session typed semantics. The bisimulation theory presented in this section is extended in the next section to define a bisimulation theory that uses a more fine-grained LTS, defined using the additional typing information of the global observer.

4.1. **Labelled transition system.**

We use the following labels $(\ell, \ell', \dots)$ to define the labelled transition system:

$$\begin{array}{rcl}\ell & ::= & \overline{a}[A](s) \;\mid\; a[A](s) \;\mid\; s[\mathrm{p}][\mathrm{q}]!\langle v\rangle \;\mid\; s[\mathrm{p}][\mathrm{q}]!(a) \\ & \mid & s[\mathrm{p}][\mathrm{q}]!(s'[\mathrm{p}']) \;\mid\; s[\mathrm{p}][\mathrm{q}]?\langle v\rangle \;\mid\; s[\mathrm{p}][\mathrm{q}] \oplus l \;\mid\; s[\mathrm{p}][\mathrm{q}]\&l \;\mid\; \tau\end{array}$$

Note that label $s[\mathrm{p}][\mathrm{q}]!\langle s'[\mathrm{p}']\rangle$ is subsumed in label $s[\mathrm{p}][\mathrm{q}]!\langle v\rangle$. Symbol A denotes a *role set*, which is a set of roles. Labels $\overline{a}[A](s)$ and $a[A](s)$ define the accept and request of a fresh session s by roles in set A, respectively. Actions on session channels are denoted with the labels $s[\mathrm{p}][\mathrm{q}]!\langle v\rangle$ and $s[\mathrm{p}][\mathrm{q}]?\langle v\rangle$ for output and input of value v from p to q on session s. Bound output values can be shared channels or session endpoints (delegation). $s[\mathrm{p}][\mathrm{q}] \oplus l$ and $s[\mathrm{p}][\mathrm{q}]\&l$ define the select and branch, respectively. Label τ is the unobserved transition.

Dual label definition is used to define the parallel rule in the label transition system:

Definition 4.1 (Dual Labels)**.** We define a duality relation $\asymp$ between two labels which is generated by the following axioms and synmetric ones:

$$s[\mathrm{p}][\mathrm{q}]!\langle v\rangle \asymp s[\mathrm{q}][\mathrm{p}]?\langle v\rangle \qquad s[\mathrm{p}][\mathrm{q}]!(v) \asymp s[\mathrm{q}][\mathrm{p}]?\langle v\rangle \qquad s[\mathrm{p}][\mathrm{q}] \oplus l \asymp s[\mathrm{q}][\mathrm{p}]\&l$$

Dual labels are input and output (respectively select and branch) on the same session channel and on complementary roles. For example, in $s[\mathrm{p}][\mathrm{q}]!\langle v\rangle$ and $s[\mathrm{q}][\mathrm{p}]?\langle v\rangle$, role p sends to q and role q receives from p.

We define the notion of complete role set, used for defining session initiation transition semantics later:

$$\langle\text{Req}\rangle \quad \overline{a}[\mathtt{p}](x).P \xrightarrow{\overline{a}[\{\mathtt{p}\}](s)} P\{s[\mathtt{p}]/x\} \qquad \langle\text{Acc}\rangle \quad a[\mathtt{p}](x).P \xrightarrow{a[\{\mathtt{p}\}](s)} P\{s[\mathtt{p}]/x\}$$

$$\langle\text{Send}\rangle \quad s[\mathtt{p}][\mathtt{q}]!\langle e\rangle;P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle} P \quad (e\downarrow v) \qquad \langle\text{Rcv}\rangle \quad s[\mathtt{p}][\mathtt{q}]?(x);P \xrightarrow{s[\mathtt{p}][\mathtt{q}]?\langle v\rangle} P\{v/x\}$$

$$\langle\text{Sel}\rangle \quad s[\mathtt{p}][\mathtt{q}]\oplus l;P \xrightarrow{s[\mathtt{p}][\mathtt{q}]\oplus l} P \qquad \langle\text{Bra}\rangle \quad s[\mathtt{p}][\mathtt{q}]\&\{l_i : P_i\}_{i\in I} \xrightarrow{s[\mathtt{p}][\mathtt{q}]\& l_k} P_k$$

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

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

$$\langle\text{OpenS}\rangle \ \frac{P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle s'[\mathtt{p}']\rangle} P' \quad s \neq s'}{(\nu\ s')P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])} P'} \qquad \langle\text{OpenN}\rangle \ \frac{P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle a\rangle} P'}{(\nu\ a)P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!(a)} P'}$$

$$\langle\text{Alpha}\rangle \ \frac{P \equiv_\alpha P' \quad P' \xrightarrow{\ell} Q'}{P \xrightarrow{\ell} Q} \qquad \langle\text{AccPar}\rangle \ \frac{P_1 \xrightarrow{a[A](s)} P_1' \quad P_2 \xrightarrow{a[A'](s)} P_2' \quad A\cap A'=\emptyset}{P_1 \mid P_2 \xrightarrow{a[A\cup A'](s)} P_1' \mid P_2'}$$

$$\langle\text{ReqPar}\rangle \ \frac{P_1 \xrightarrow{a[A](s)} P_1' \quad P_2 \xrightarrow{\overline{a}[A'](s)} P_2' \quad A\cap A'=\emptyset,\ A\cup A' \text{ not complete w.r.t } \max(A')}{P_1 \mid P_2 \xrightarrow{\overline{a}[A\cup A'](s)} P_1' \mid P_2'}$$

$$\langle\text{TauS}\rangle \ \frac{P_1 \xrightarrow{a[A](s)} P_1' \quad P_2 \xrightarrow{\overline{a}[A'](s)} P_2' \quad A\cap A'=\emptyset,\ A\cup A' \text{ complete w.r.t } \max(A')}{P_1 \mid P_2 \xrightarrow{\tau} (\nu\ s)(P_1' \mid P_2')}$$

We omit the synmetric case of $\langle\text{Par}\rangle$ and conditionals.

Figure 7: Labelled transition system for processes

Definition 4.2 (Complete role set)**.** We say the role set A is *complete with respect to n* if $n = \max(A)$ and $A = \{1,2,\dots,n\}$.

A complete role set means that all global protocol participants are present in the set. For example, $\{1,3,4\}$ is not complete, but $\{1,2,3,4\}$ is with respect to 4 and not complete with number $n > 4$. We use $\mathtt{fn}(\ell)$ and $\mathtt{bn}(\ell)$ to denote a set of free and bound names in ℓ and set $\mathtt{n}(\ell) = \mathtt{bn}(\ell)\cup\mathtt{fn}(\ell)$.

Labelled transition system for processes. Figure 7 defines the untyped labelled transition system. Rules $\langle\text{Req}\rangle$ and $\langle\text{Acc}\rangle$ define that processes $\overline{a}[\mathtt{p}](x).P$ and $a[\mathtt{p}](x).P$ produce the accept and request labels, respectively for a fresh session s on role $\mathtt{p}$. Rules $\langle\text{Send}\rangle$ and $\langle\text{Rcv}\rangle$ predict that processes $s[\mathtt{p}][\mathtt{q}]!\langle v\rangle;P$ and $s[\mathtt{p}][\mathtt{q}]?(x);P$ produce the send and receive label, respectively for value v from role $\mathtt{p}$ to role $\mathtt{q}$ in session s. Similarly, rules $\langle\text{Sel}\rangle$ and $\langle\text{Bra}\rangle$ define that the select and branch labels are observed on processes $s[\mathtt{p}][\mathtt{q}]\oplus l;P$ and $s[\mathtt{p}][\mathtt{q}]\&\{l_i : P_i\}$ respectively.

The last three rules collect and synchronise the multiparty participants together. Rule $\langle\text{AccPar}\rangle$ accumulates the accept participants and records them into role set A. Rule $\langle\text{ReqPar}\rangle$ accumulates the accept participants and the request participant into role set A. Note that the request action role set always includes the maximum role number among the participants.

Finally, rule $\langle\text{TauS}\rangle$ checks that a role set is complete (i.e. all roles are present), thus a new session can be created under the τ-action (synchronisation). Other rules follow the usual inductive rules for the π-calculus labelled transition system.

Rule $\langle\text{Tau}\rangle$ synchronises two processes that exhibit dual labels, while rules $\langle\text{Par}\rangle$ and $\langle\text{Res}\rangle$ close the labelled transition system under the parallel composition and name restriction operators. Note that $\langle\text{Par}\rangle$ allows the output transition to the role $\mathtt{q}$ even the endpoint at $\mathtt{q}$ is in Q. This will be disallowed by the environment LTS defined in Figure 8 later.

Rules $\langle\text{OpenS}\rangle$ and $\langle\text{OpenN}\rangle$ are used for name extrusion. Finally, rule $\langle\text{Alpha}\rangle$ closes the LTS under structural congruence.

We write $\Longrightarrow$ for the reflexive and transitive closure of $\longrightarrow$, $\overset{\ell}{\Longrightarrow}$ for the transitions $\Longrightarrow\overset{\ell}{\longrightarrow}\Longrightarrow$ and $\overset{\hat{\ell}}{\Longrightarrow}$ for $\overset{\ell}{\Longrightarrow}$ if $\ell \neq \tau$ otherwise $\Longrightarrow$.

Typed labelled transition relation. We define the typed LTS on the basis of the untyped one. This is realised by defining an environment labelled transition system, defined in Figure 8, which uses the same labels defined for the untyped LTS. We write $(\Gamma,\Delta) \xrightarrow{\ell} (\Gamma',\Delta')$ as the notation where an environment (Γ,Δ) allows an action ℓ to take place, resulting in environment (Γ',Δ').

The intuition for this definition is that the observables on session channels occur when the corresponding endpoint is not present in the linear typing environment Δ, and the type of an action's object respects the environment (Γ,Δ). In the case when new names are created or received, the environment (Γ,Δ) is extended according to the new name.

Rule $\{\text{Req}\}$ says that a reception of a message via a is possible when a's type $\langle G\rangle$ is recorded into Γ and the resulting session environment records projected types from G $(\{s[i] : G\lceil i\}_{i\in A})$. Rule $\{\text{Acc}\}$ is for the send of a message via a and it is dual to the first rule. The next four rules are free value output $\{\text{Send}\}$, bound name output $\{\text{OpenN}\}$, free value input $\{\text{Recv}\}$ and name input $\{\text{RecvN}\}$. The rest of rules are free session output $\{\text{SendS}\}$, bound session output $\{\text{OpenS}\}$, and session input $\{\text{RecvS}\}$ as well as selection $\{\text{Sel}\}$ and branching $\{\text{Bra}\}$ rules. The bound session output $\{\text{OpenS}\}$ records a set of session types $s'[\mathtt{p}_i]$ at opened session s'. Rule $\{\text{Tau}\}$ follows the reduction rules for linear session environment defined in § 3.3 ($\Delta = \Delta'$ is the case for the reduction at hidden sessions). Note that if Δ already contains destination $(s[\mathtt{q}])$, the environment cannot perform the visible action, but only the final τ-action.

The typed LTS requires that a process can perform an untyped action ℓ and that its typing environment (Γ,Δ) can match the action ℓ.

Definition 4.3 (Typed transition)**.** A *typed transition relation* is a typed relation $\Gamma_1 \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} \Gamma_2 \vdash P_2 \triangleright \Delta_2$ if (1) $P_1 \xrightarrow{\ell} P_2$ and (2) $(\Gamma_1,\Delta_1) \xrightarrow{\ell} (\Gamma_2,\Delta_2)$ with $\Gamma_i \vdash P_i \triangleright \Delta_i$.

4.2. **Synchronous multiparty behavioural theory.**

We begin with the definition of the typed relation as the binary relation over closed, coherent and typed processes.

Definition 4.4 (Typed relation)**.** A relation $\mathscr{R}$ as *typed relation* if it relates two closed, coherent typed terms, written:

$$\Gamma \vdash P_1 \triangleright \Delta_1 \,\mathscr{R}\, \Gamma \vdash P_2 \triangleright \Delta_2$$

We often write $\Gamma \vdash P_1 \triangleright \Delta_1 \,\mathscr{R}\, P_2 \triangleright \Delta_2$.

Next we define the *barb* [3] with respect to the judgement:

Definition 4.5 (Barbs)**.** We write:

$$\frac{\Gamma(a) = \langle G \rangle \quad s \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta) \xrightarrow{a[A](s)} (\Gamma,\Delta\cdot\{s[i] : G\lceil i\}_{i\in A})}\{\mathrm{Req}\} \qquad \frac{\Gamma(a) = \langle G \rangle \quad s \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta) \xrightarrow{\overline{a}[A](s)} (\Gamma,\Delta\cdot\{s[i] : G\lceil i\}_{i\in A})}\{\mathrm{Acc}\}$$

$$\frac{\Gamma \vdash v : U \quad s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]!\langle U\rangle;T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]!\langle v\rangle} (\Gamma,\Delta\cdot s[\mathrm{p}] : T)}\{\mathrm{Send}\}$$

$$\frac{s[\mathrm{q}] \notin \mathtt{dom}(\Delta) \quad a \notin \mathtt{dom}(\Gamma)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]!\langle U\rangle;T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]!(a)} (\Gamma\cdot a : U,\Delta\cdot s[\mathrm{p}] : T)}\{\mathrm{OpenN}\}$$

$$\frac{\Gamma \vdash v : U \quad s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]?(U);T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]?\langle v\rangle} (\Gamma,\Delta\cdot s[\mathrm{p}] : T)}\{\mathrm{Recv}\}$$

$$\frac{a \notin \mathtt{dom}(\Gamma) \quad s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]?(U);T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]?\langle a\rangle} (\Gamma\cdot a : U,\Delta\cdot s[\mathrm{p}] : T)}\{\mathrm{RecvN}\}$$

$$\frac{s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s'[\mathrm{p}'] : T'\cdot s[\mathrm{p}] : [\mathrm{q}]!\langle T'\rangle;T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]!\langle s'[\mathrm{p}']\rangle} (\Gamma,\Delta\cdot s[\mathrm{p}] : T)}\{\mathrm{SendS}\}$$

$$\frac{s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]!\langle T'\rangle;T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]!(s'[\mathrm{p}'])} (\Gamma,\Delta\cdot s[\mathrm{p}] : T\cdot\{s'[\mathrm{p}_i] : T_i\})}\{\mathrm{OpenS}\}$$

$$\frac{s[\mathrm{q}], s'[\mathrm{p}'] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]?(T');T) \xrightarrow{s[\mathrm{p}][\mathrm{q}]?\langle s'[\mathrm{p}']\rangle} (\Gamma,\Delta\cdot s'[\mathrm{p}'] : T'\cdot s[\mathrm{p}] : T)}\{\mathrm{RecvS}\}$$

$$\frac{s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]\oplus\{l_i : T_i\}_{i\in I}) \xrightarrow{s[\mathrm{p}][\mathrm{q}]\oplus l_k} (\Gamma,\Delta\cdot s[\mathrm{p}] : T_k)}\{\mathrm{Sel}\}$$

$$\frac{s[\mathrm{q}] \notin \mathtt{dom}(\Delta)}{(\Gamma,\Delta\cdot s[\mathrm{p}] : [\mathrm{q}]\&\{l_i : T_i\}_{i\in I}) \xrightarrow{s[\mathrm{p}][\mathrm{q}]\& l_k} (\Gamma,\Delta\cdot s[\mathrm{p}] : T_k)}\{\mathrm{Bra}\} \qquad \frac{\Delta \longrightarrow \Delta' \quad \text{or} \quad \Delta = \Delta'}{(\Gamma,\Delta) \xrightarrow{\tau} (\Gamma,\Delta')}\{\mathrm{Tau}\}$$

Figure 8: Labelled transition system for environments

- $\Gamma \vdash P \triangleright \Delta \downarrow_{s[\mathrm{p}][\mathrm{q}]}$ if $P \equiv (\nu\ \tilde{a}\tilde{s})(s[\mathrm{p}][\mathrm{q}]!\langle v\rangle;R \mid Q)$ with $s \notin \tilde{s}$ and $s[\mathrm{q}] \notin \mathtt{dom}(\Delta)$.
- $\Gamma \vdash P \triangleright \Delta \downarrow_a$ if $P \equiv (\nu\ \tilde{a}\tilde{s})(\overline{a}[n](x).R \mid Q)$ with $a \notin \tilde{a}$. Then we write m for either a or $s[\mathrm{p}][\mathrm{q}]$.

We define $\Gamma \vdash P \triangleright \Delta \Downarrow_m$ if there exists Q such that $P \longrightarrow\!\!\!\!\!\rightarrow Q$ and $\Gamma \vdash Q \triangleright \Delta' \downarrow_m$.

The set of contexts is defined as follows:

$$\begin{array}{rcl} C & ::= & - \mid C \mid P \mid P \mid C \mid (\nu\ n)C \mid \mathtt{if}\ e\ \mathtt{then}\ C\ \mathtt{else}\ C' \mid \mu X.C \mid \\ & & s!\langle v\rangle;C \mid s?(x);C \mid s\oplus l;C \mid s\&\{l_i : C_i\}_{i\in I} \mid \overline{a}(x).C \mid a(x).C \end{array}$$

$C[P]$ substitutes process P for each hole $(-)$ in context C. We say $C[P]$ is *closed* if $\mathtt{fv}(C[P]) = \emptyset$.

Definition 4.6 (Linear environment convergence)**.** We write $\Delta_1 \rightleftharpoons \Delta_2$ if there exists Δ such that $\Delta_1 \longrightarrow^* \Delta$ and $\Delta_2 \longrightarrow^* \Delta$.

We expect processes with the *same* behaviour to have linear environments that converge since they should follow the same session behaviour.

Note that the convergence condition is not related with the fact that reduction on a linear environment is non-deterministic (see Definition 3.8) and two linear environments Δ_1 and Δ_2 which are non-deterministic may converge.

We define the reduction-closed congruence based on the definition of barb and [22].

Definition 4.7 (Reduction-closed congruence)**.** A typed relation $\mathscr{R}$ is a *reduction-reduction congruence* if it satisfies the following conditions for each $\Gamma \vdash P_1 \triangleright \Delta_1 \,\mathscr{R}\, P_2 \triangleright \Delta_2$:

(1) $\Delta_1 \rightleftharpoons \Delta_2$
(2) $\Gamma \vdash P_1 \triangleright \Delta_1 \Downarrow_m$ iff $\Gamma \vdash P_2 \triangleright \Delta_2 \Downarrow_m$.
(3) • $P_1 \twoheadrightarrow P_1'$ implies that there exists P_2' such that $P_2 \twoheadrightarrow P_2'$ and $\Gamma \vdash P_1' \triangleright \Delta_1' \,\mathscr{R}\, P_2' \triangleright \Delta_2'$ with $\Delta_1' \rightleftharpoons \Delta_2'$.
• the symmetric case.
(4) For all closed context C and for all Δ_1' and Δ_2', such that $\Gamma \vdash C[P_1] \triangleright \Delta_1'$ and $\Gamma \vdash C[P_2] \triangleright \Delta_2'$ then $\Delta_1' \rightleftharpoons \Delta_2'$ and $\Gamma \vdash C[P_1] \triangleright \Delta_1' \,\mathscr{R}\, \Gamma \vdash C[P_2] \triangleright \Delta_2'$.

The union of all reduction-closed congruence relations is denoted as $\cong^s$.

We now define the synchronous multiparty session bisimilarity as the greatest fixed point on the weak labelled transition relation for the pairs of co-directed processes.

Definition 4.8 (Synchronous multiparty session bisimulation)**.** A typed relation $\mathscr{R}$ over closed processes is a (weak) *synchronous multiparty session bisimulation* or often a *synchronous bisimulation* if, whenever $\Gamma \vdash P_1 \triangleright \Delta_1 \,\mathscr{R}\, P_2 \triangleright \Delta_2$, it holds:

(1) $\Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} \Gamma' \vdash P_1' \triangleright \Delta_1'$ implies $\Gamma \vdash P_2 \triangleright \Delta_2 \overset{\hat{\ell}}{\Longrightarrow} \Gamma' \vdash P_2' \triangleright \Delta_2'$ such that $\Gamma' \vdash P_1' \triangleright \Delta_1' \,\mathscr{R}\, P_2' \triangleright \Delta_2'$.
(2) The symmetric case.

The maximum bisimulation exists which we call *synchronous bisimilarity*, denoted by $\approx^s$. We sometimes leave environments implicit, writing e.g. $P \approx^s Q$.

Lemma 4.9. *If* $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$ *then* $\Delta_1 \rightleftharpoons \Delta_2$.

Proof. The proof uses the co-induction method and can be found in Appendix B.1. □

Theorem 4.10 (Soundness and completeness)**.** $\cong^s = \approx^s$.

Proof. The proof is a simplification of the proof of Theorem 5.15 in Appendix B.5. □

Example 4.11 (Synchronous multiparty bisimulation)**.** We use the running example from the introduction, § 1, for a demonstration of the bisimulation semantics developed in this section. In the introduction we considered transition under the untyped setting [45]. If we follow the typed labelled transition system developed in this section we obtain similar interaction patterns.

Recall the definition of processes P_1, P_2, P_3 and R_2 from § 1. The linear types for these processes are empty since they have no free session names.

$$\Gamma \vdash P_1 \triangleright \emptyset, \quad \Gamma \vdash P_2 \triangleright \emptyset, \quad \Gamma \vdash P_3 \triangleright \emptyset, \quad \text{and} \quad \Gamma \vdash R_2 \triangleright \emptyset$$

where $\Gamma = a : G_a \cdot b : G_b$ with

$$\begin{aligned} G_a &= 1 \to 3 : \langle U \rangle . 2 \to 3 : \langle U \rangle . \mathtt{end} \\ G_b &= 1 \to 2 : \langle U \rangle . \mathtt{end} \end{aligned}$$

We follow the untyped LTS from Figure 7 to obtain the following processes by $\langle\text{Acc}\rangle$ and $\langle\text{Req}\rangle$:

$$\begin{array}{rcl}
P_1 & \xrightarrow{a[\{1\}](s_a)} & P_1' = b[1](y).s_a[1][3]!\langle v\rangle;y[2]!\langle w\rangle;\mathbf{0} \\
P_2 & \xrightarrow{a[\{2\}](s_a)} & P_2' = \overline{b}[2](y).(y[1]?(z);\mathbf{0} \mid s_a[2][3]!\langle v\rangle;\mathbf{0}) \\
P_3 & \xrightarrow{\overline{a}[\{3\}](s_a)} & P_3' = s_a[3][1]?(z);s_a[3][2]?(y);\mathbf{0} \\
R_2 & \xrightarrow{a[\{2\}](s_a)} & R_2' = \overline{b}[2](y).(y[1]?(z);s_a[2][3]!\langle v\rangle;\mathbf{0})
\end{array}$$

The corresponding environment transitions are defined as:

$$\begin{array}{rcl}
(\Gamma,\emptyset) & \xrightarrow{a[\{1\}](s_a)} & (\Gamma,\ s_a[1]:[3]!\langle U\rangle;\mathtt{end}) \\
(\Gamma,\emptyset) & \xrightarrow{a[\{2\}](s_a)} & (\Gamma,\ s_a[2]:[3]!\langle U\rangle;\mathtt{end}) \\
(\Gamma,\emptyset) & \xrightarrow{a[\{3\}](s_a)} & (\Gamma,\ s_a[3]:[1]?(U);[2]?(U);\mathtt{end})
\end{array}$$

We can now observe the typed transitions as follows:

$$\begin{array}{rcl}
\Gamma\vdash P_1\triangleright\emptyset & \xrightarrow{a[\{1\}](s_a)} & \Gamma\vdash P_1'\triangleright s_a[1]:[3]!\langle U\rangle;\mathtt{end} \\
\Gamma\vdash P_2\triangleright\emptyset & \xrightarrow{a[\{2\}](s_a)} & \Gamma\vdash P_2'\triangleright s_a[2]:[3]!\langle U\rangle;\mathtt{end} \\
\Gamma\vdash P_3\triangleright\emptyset & \xrightarrow{a[\{3\}](s_a)} & \Gamma\vdash P_3'\triangleright s_a[3]:[1]?(U);[2]?(U);\mathtt{end}
\end{array}$$

By $\langle\text{AccPar}\rangle$, we have

$$P_1 \mid P_2 \xrightarrow{a[\{1,2\}](s_a)} P_1' \mid P_2'$$

By $\langle\text{ReqPar}\rangle$, another process combination would invoke

$$P_1 \mid P_3 \xrightarrow{\overline{a}[\{1,3\}](s_a)} P_1' \mid P_3'$$

If we compose the missing process in either of both processes, the role set $\{1,2,3\}$ is now complete with respect to 3, so that by synchronisation $\langle\text{TauS}\rangle$ we may observe:

$$P_1 \mid P_2 \mid P_3 \xrightarrow{\tau} (\nu\ s_a)(P_1' \mid P_2' \mid P_3')$$

Furthermore, we can also observe the corresponding typed transition, since (Γ,Δ) can always perform a τ action:

$$\Gamma\vdash P_1 \mid P_2 \mid P_3\triangleright\emptyset \xrightarrow{\tau} \Gamma\vdash(\nu\ s_a)(P_1' \mid P_2' \mid P_3')\triangleright\emptyset$$

Now we demonstrate the intuition given in § 1, i.e. the bisimulation developed in this section cannot equate $P_1 \mid P_2$ and $P_1 \mid R_2$. We have the following transitions:

$$\begin{array}{l}
\Gamma\vdash P_1' \mid P_2'\triangleright\Delta_0 \xrightarrow{\tau} \\
\qquad \Gamma\vdash(\nu\ s_b)(s_a[1][3]!\langle v\rangle;s_b[1][2]!\langle w\rangle;\mathbf{0} \mid s_b[2][1]?(z);\mathbf{0} \mid s_a[2][3]!\langle v\rangle;\mathbf{0}) = Q_1\triangleright\Delta_0 \\
\Gamma\vdash P_1' \mid R_2'\triangleright\Delta_0 \xrightarrow{\tau} \\
\qquad \Gamma\vdash(\nu\ s_b)(s_a[1][3]!\langle v\rangle;s_b[1][2]!\langle w\rangle;\mathbf{0} \mid s_b[2][1]?(z);s_a[2][3]!\langle v\rangle;\mathbf{0}) = Q_2\triangleright\Delta_0
\end{array}$$

with $\Delta_0 = s_a[1]:[3]!\langle U\rangle;\mathtt{end}\cdot s_a[2]:[3]!\langle U\rangle;\mathtt{end}$.

From this point, we can check:

$$\Gamma\vdash Q_1\triangleright\Delta_0 \not\approx^s \Gamma\vdash Q_2\triangleright\Delta_0$$

due to the fact that $(\Gamma,\Delta_0) \overset{s_a[2][3]!\langle v\rangle}{\longrightarrow}$ and

$$\Gamma \vdash Q_1 \triangleright \Delta_0 \overset{s_a[2][3]!\langle v\rangle}{\longrightarrow}$$
$$\Gamma \vdash Q_2 \triangleright \Delta_0 \overset{s_a[2][3]!\langle v\rangle}{\not\longrightarrow}$$

The next result distinguishes the semantics of the typed equivalence semantics developed in this section from the untyped equivalence semantics [45].

$$\Gamma \vdash Q_1 \mid P_3' \triangleright \Delta_0 \cdot s_a[3] : [1]?(U);[2]?(U);\mathtt{end} \approx^s \mathbf{0} \triangleright s_a[a] : \mathtt{end} \cdot s_a[2] : \mathtt{end} \cdot s_a[3] : \mathtt{end}$$

since

$$(\Gamma,\Delta) \overset{\ell}{\not\longrightarrow}$$

for any $\ell \neq \tau$ with $\Delta = \Delta_0 \cdot s_a[3] : [1]?(U);[2]?(U);\mathtt{end}$ (by the condition of $\{\mathsf{Send}\}$ in Figure 8).

However the untyped labelled transition semantics do not equate the two processes $Q_1 \mid P_3' \not\approx \mathbf{0}$ since $Q_1 \mid P_3' \overset{s_a[1][3]!\langle v\rangle}{\longrightarrow}$.

5. Globally Governed Behavioural Theory

We introduce the semantics for globally governed behavioural theory. In the previous section, the local typing (Δ) is used to constrain the untyped LTS and give rise to a local typed LTS. In a multiparty distributed environment, communication follows the global protocol, which controls both an observed process and its observer. The local typing is not sufficient to maintain the consistency of transitions of a process with respect to a global protocol. In this section we refine the environment LTS with a *global environment* E to give a more fine-grained control over the LTS of the processes. We then show a bisimulation-based reasoning technique which equates the two processes $P_1 \mid P_2$ and $P_1 \mid R_2$ in § 1 by the governed bisimulation, which cannot be equated by the standard synchronous typed bisimulation $\approx^s$ studied in the previous section.

5.1. Global environments and configurations.

We define a *global environment* $(E, E', \ldots)$ as a mapping from session names to global types.

$$E \ ::= \ E \cdot s : G \ \mid \ \emptyset$$

We extend the projection definition on global environments E as follows:

$$\mathtt{proj}(E) = \bigcup_{s:G \in E} \mathtt{proj}(s:G)$$

where $\mathtt{proj}(s:G)$ associates the projection of type G with session name s as follows: $\mathtt{proj}(s:G) = \{s[\mathtt{p}] : G\lceil\mathtt{p} \mid \mathtt{p} \in \mathtt{roles}(G)\}$. Note that E is a mapping from a session channel to a global type, while Γ is a mapping from a shared channel to a global type.

We define a labelled reduction relation over global environments which corresponds to $\Delta_0 \longrightarrow \Delta_0'$ defined in § 3.3. We use the labels:

$$\lambda \ ::= \ s:\mathtt{p}\to\mathtt{q}:U \ \mid \ s:\mathtt{p}\to\mathtt{q}:l$$

to annotate reductions over global environments. We define $\mathtt{out}(\lambda)$ and $\mathtt{inp}(\lambda)$ as:

$$\begin{array}{lllll} \mathtt{out}(s:\mathtt{p}\to\mathtt{q}:U) & = & \mathtt{out}(s:\mathtt{p}\to\mathtt{q}:l) & = & \mathtt{p} \\ \mathtt{inp}(s:\mathtt{p}\to\mathtt{q}:U) & = & \mathtt{inp}(s:\mathtt{p}\to\mathtt{q}:l) & = & \mathtt{q} \end{array}$$

and write $\mathtt{p} \in \ell$ if $\mathtt{p} \in \{\mathtt{out}(\ell)\} \cup \{\mathtt{inp}(\ell)\}$.

Definition 5.1 (Global environment reduction)**.** We define the relation $E \xrightarrow{\lambda} E'$ as the smallest relation generated by the following rules:

$$\{s : \mathtt{p} \to \mathtt{q} : \langle U \rangle.G\} \overset{s:\mathtt{p}\to\mathtt{q}:U}{\longrightarrow} \{s : G\} \quad (\text{Inter}) \qquad \{s : \mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I}\} \overset{s:\mathtt{p}\to\mathtt{q}:l_k}{\longrightarrow} \{s : G_k\} \quad (\text{SelBra})$$

$$\frac{\{s : G\} \xrightarrow{\lambda} \{s : G'\} \quad \mathtt{p},\mathtt{q} \notin \lambda}{\{s : \mathtt{p} \to \mathtt{q} : \langle U \rangle.G\} \xrightarrow{\lambda} \{s : \mathtt{p} \to \mathtt{q} : \langle U \rangle.G'\}} \ (\text{IPerm})$$

$$\frac{\forall i \in I \quad \{s : G_i\} \xrightarrow{\lambda} \{s : G_i'\} \quad \mathtt{p},\mathtt{q} \notin \lambda}{\{s : \mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I}\} \xrightarrow{\lambda} \{s : \mathtt{p} \to \mathtt{q} : \{l_i : G_i'\}_{i\in I}\}} \ (\text{SBPerm}) \qquad \frac{E \xrightarrow{\lambda} E'}{E \cdot E_0 \xrightarrow{\lambda} E' \cdot E_0} \ (\text{GEnv})$$

We often omit the label λ by writing $\longrightarrow$ for $\xrightarrow{\lambda}$ and $\longrightarrow^*$ for $(\xrightarrow{\lambda})^*$. Rule (Inter) is the axiom for the input and output interaction between two parties; rule (SelBra) reduces on the select/branch choice; Rules (IPerm) and (SBPerm) define the case where we can permute action λ to be performed under $\mathtt{p} \to \mathtt{q}$ if $\mathtt{p}$ and $\mathtt{q}$ are not related to the participants in λ. Note that in our synchronous semantics, we can permute two actions with no relevance in the participating roles without changing the interaction semantics of the entire global protocol. Finaly rule (GEnv) is a congruence rule over global environments.

As a simple example of the above LTS, consider the global type:

$$s : \mathtt{p} \to \mathtt{q} : \langle U_1 \rangle.\mathtt{p}' \to \mathtt{q}' : \{l_1 : \mathtt{end}, l_2 : \mathtt{p}' \to \mathtt{q}' : \langle U_2 \rangle.\mathtt{end}\}$$

Since $\mathtt{p},\mathtt{q},\mathtt{p}',\mathtt{q}'$ are pairwise distinct, we can apply the second and third rules to obtain:

$$s : \mathtt{p} \to \mathtt{q} : \langle U_1 \rangle.\mathtt{p}' \to \mathtt{q}' : \{l_1 : \mathtt{end}, l_2 : \mathtt{p}' \to \mathtt{q}' : \langle U_2 \rangle.\mathtt{end}\} \overset{s:\mathtt{p}'\to\mathtt{q}':l_1}{\longrightarrow} s : \mathtt{p} \to \mathtt{q} : \langle U_1 \rangle.\mathtt{end}$$

Next we introduce the *governance judgement* which controls the behaviour of processes by the global environment.

Definition 5.2 (Governance judgement)**.** Let $\Gamma \vdash P \triangleright \Delta$ be coherent. We write $E, \Gamma \vdash P \triangleright \Delta$ if $\exists E'$ such that $E \longrightarrow^* E'$ and $\Delta \subseteq \mathtt{proj}(E')$.

The global environment E records the knowledge of both the environment (Δ) of the observed process P and the environment of its *observer*. The side conditions ensure that E is coherent with Δ: there exist E' reduced from E whose projection should cover the environment of P since E should include the observer's information together with the observed process information recorded into Δ. The reason that E is allowed to have a few reduction steps behind the local environment Δ is that the observer has more informative global knowledge (in the form of a global type) before the moment the session was actually reduced to Δ which coincides with the projection of E'.

Next we define the LTS for well-formed environment configurations.

Definition 5.3 (Environment configuration)**.** We write (E, Γ, Δ) if $\exists E'$ such that $E \longrightarrow^* E'$ and $\Delta \subseteq \mathtt{proj}(E')$.

The up-to reduction requirement on E allows a global environment E to configure linear environments Δ that also differ up-to reduction. Specificaly a global environment E configures pairs of linear environments that type equivalent processes.

We refined the reduction relation on Δ in § 3.3 as a labelled reduction relation on Δ, which is used for defining a labelled transition system over environment configurations:

Definition 5.4 (Linear typing labelled reduction)**.**

$$\frac{s \notin \mathtt{dom}(E) \quad (\Gamma,\Delta_1) \overset{a[A](s)}{\longrightarrow} (\Gamma,\Delta_2)}{(E,\Gamma,\Delta_1) \overset{a[A](s)}{\longrightarrow} (E \cdot s : G,\Gamma,\Delta_2)} \text{[Acc]} \qquad \frac{s \notin \mathtt{dom}(E) \quad (\Gamma,\Delta_1) \overset{\overline{a}[A](s)}{\longrightarrow} (\Gamma,\Delta_2)}{(E,\Gamma,\Delta_1) \overset{\overline{a}[A](s)}{\longrightarrow} (E \cdot s : G,\Gamma,\Delta_2)} \text{[Req]}$$

$$\frac{(\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle}{\longrightarrow} (\Gamma,\Delta_2) \quad E_1 \overset{s:\mathtt{p}\to\mathtt{q}:U}{\longrightarrow} E_2}{(E_1,\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle}{\longrightarrow} (E_2,\Gamma,\Delta_2)} \text{[Out]}$$

$$\frac{(\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]?\langle v\rangle}{\longrightarrow} (\Gamma \cdot v : U,\Delta_2) \quad E_1 \overset{s:\mathtt{q}\to\mathtt{p}:U}{\longrightarrow} E_2}{(E_1,\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]?\langle v\rangle}{\longrightarrow} (E_2,\Gamma \cdot v : U,\Delta_2)} \text{[In]}$$

$$\frac{(\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]!(a)}{\longrightarrow} (\Gamma \cdot a : \langle G\rangle,\Delta_2) \quad E_1 \overset{s:\mathtt{q}\to\mathtt{p}:\langle G\rangle}{\longrightarrow} E_2}{(E_1,\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]!(a)}{\longrightarrow} (E_2,\Gamma \cdot a : \langle G\rangle,\Delta_2)} \text{[ResN]}$$

$$\frac{(\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])}{\longrightarrow} (\Gamma,\Delta_2 \cdot \{s'[\mathtt{p}_i] : T_i\}_{i\in I}) \quad \forall i \in I. G\lceil \mathtt{p}_i = T_i \quad s' \notin \mathtt{dom}(E_1) \quad E_1 \overset{s:\mathtt{q}\to\mathtt{p}:T_{\mathtt{p}'}}{\longrightarrow} E_2}{(E_1,\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])}{\longrightarrow} (E_2 \cdot s' : G,\Gamma,\Delta_2 \cdot \{s'[\mathtt{p}_i] : T_i\}_{i\in I})} \text{[ResS]}$$

$$\frac{(\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]\oplus l}{\longrightarrow} (\Gamma,\Delta_2) \quad E_1 \overset{s:\mathtt{p}\to\mathtt{q}:l}{\longrightarrow} E_2}{(E_1,\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]\oplus l}{\longrightarrow} (E_2,\Gamma,\Delta_2)} \text{[Sel]} \qquad \frac{(\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]\& l}{\longrightarrow} (\Gamma,\Delta_2) \quad E_1 \overset{s:\mathtt{q}\to\mathtt{p}:l}{\longrightarrow} E_2}{(E_1,\Gamma,\Delta_1) \overset{s[\mathtt{p}][\mathtt{q}]\& l}{\longrightarrow} (E_2,\Gamma,\Delta_2)} \text{[Bra]}$$

$$\frac{(\Delta_1 = \Delta_2, E_1 = E_2) \vee (\Delta_1 \longrightarrow \Delta_2, E_1 \xrightarrow{\lambda} E_2)}{(E_1,\Gamma,\Delta_1) \xrightarrow{\tau} (E_2,\Gamma,\Delta_2)} \text{[Tau]}$$

$$\frac{E_1 \longrightarrow^* E_1' \quad (E_1',\Gamma_1,\Delta_1) \xrightarrow{\ell} (E_2,\Gamma_2,\Delta_2)}{(E_1,\Gamma_1,\Delta_1) \xrightarrow{\ell} (E_2,\Gamma_2,\Delta_2)} \text{[Inv]}$$

Figure 9: Labelled transition system for environment configuations

(1) $\{s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle; T \cdot s[\mathtt{q}] : [\mathtt{p}]?(U); T'\} \overset{s:\mathtt{p}\to\mathtt{q}:U}{\longrightarrow} \{s[\mathtt{p}] : T \cdot s[\mathtt{q}] : T'\}$.

(2) $\{s[\mathtt{p}] : [\mathtt{q}] \oplus \{l_i : T_i\}_{i\in I} \cdot s[\mathtt{q}] : [\mathtt{p}]\&\{l_j : T_j'\}_{j\in J}\} \overset{s:\mathtt{p}\to\mathtt{q}:l_k}{\longrightarrow} \{s[\mathtt{p}] : T_k \cdot s[\mathtt{q}] : T_k'\}\ I \subseteq J, k \in I$.

(3) $\Delta \cup \Delta' \xrightarrow{\lambda} \Delta \cup \Delta''$ if $\Delta' \xrightarrow{\lambda} \Delta''$.

Figure 9 defines an LTS over environment configurations that refines the LTS over environments (i.e $(\Gamma,\Delta) \xrightarrow{\ell} (\Gamma',\Delta')$) in § 4.1.

Each rule requires a corresponding environment transition (Figure 8 in § 4.1) and a corresponding labelled global environment transition in order to control a transition following the global protocol. Rule [Acc] defines the acceptance of a session initialisation by creating a new mapping $s : G$ which matches Γ in a governed environment E. Rule [Req] defines the request for a new session and it is dual to [Acc].

The next six rules are the transition relations on session channels and we assume the condition $\mathtt{proj}(E_1) \supseteq \Delta_1$ to ensure the base action of the environment matches one in a global environment. [Out] is a rule for the output where the type of the value and the action of (Γ,Δ) meets those in

E. [In] is a rule for the input and dual to [Out]. [ResN] is a scope opening rule for a name so that the environment can perform the corresponding type $\langle G\rangle$ of a. [ResS] is a scope opening rule for a session channel which creates a set of mappings for the opened session channel s' corresponding to the LTS of the environment. [Sel] and [Bra] are the rules for select and branch, which are similar to [Out] and [In]. Rule [Tau] defines the silent action for environment configurations, where we require that reduction on global environments matches reduction on the linear typing. At the same time we allow a silent action with no effect on the environment configuration. Rule [Inv] closes the labelled transition system with respect to the global environment. Global environment E_1 reduces to E_1' to perform the observer's actions, hence the observed process can perform the action w.r.t. E_1'.

Hereafter we write $\longrightarrow$ for $\xrightarrow{\tau}$.

Example 5.5 (LTS for environment configuration)**.**
Let:

$$\begin{array}{rcl} E & = & s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.\mathtt{p}\to\mathtt{q}:\langle U\rangle.G \\ \Gamma & = & v:U \\ \Delta & = & s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T_{\mathtt{p}} \end{array}$$

with $G\lceil\mathtt{p}=T_{\mathtt{p}}$, $G\lceil\mathtt{q}=T_{\mathtt{q}}$ and $\mathtt{roles}(G)=\{\mathtt{p},\mathtt{q}\}$.

Tuple (E,Γ,Δ) is an environment configuration since there exists E' such that:

$$E\longrightarrow E' \text{ implies } \mathtt{proj}(E')\supset\Delta$$

Recall that we can write $E\longrightarrow E'$ for $E\xrightarrow{\lambda}E'$. Indeed we can see that:

$$\begin{array}{rcl} E & \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} & s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G \\ \mathtt{proj}(s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G) & = & s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T_{\mathtt{p}}\cdot s[\mathtt{q}]:[\mathtt{p}]?(U);T_{\mathtt{q}} \\ \mathtt{proj}(s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G) & \supset & \Delta \end{array}$$

An environment configuration transition takes a place on environment configuration (E,Γ,Δ) if we apply the condition of rule [Out] to obtain:

$$\begin{array}{rcl} s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G & \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} & s:G \\ (\Gamma,s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T_{\mathtt{p}}) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle} & (\Gamma,s[\mathtt{p}]:T_{\mathtt{p}}) \end{array}$$

thus we can obtain:

$$(s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G,\Gamma,\Delta)\xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle}(s:G,\Gamma,s[\mathtt{p}]:T_{\mathtt{p}})$$

By last result and the fact that:

$$E\longrightarrow s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G$$

we use rule [Inv], to obtain:

$$(E,\Gamma,\Delta)\xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle}(s:G,\Gamma,s[\mathtt{p}]:T_{\mathtt{p}})$$

as required.

Governed reduction-closed congruence. To define the reduction-closed congruence, we first refine the barb, which is controlled by the global witness where observables of a configuration are defined with the global environment of the observer.

Definition 5.6 (Governed barb)**.**

$$\frac{s[\mathtt{q}] \notin \mathtt{dom}(\Delta) \quad \exists E' \text{ such that } E \longrightarrow^* E' \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} \quad \Delta\cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T \subseteq \mathtt{proj}(E')}{(E,\Gamma,\Delta\cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T) \downarrow_{s[\mathtt{p}][\mathtt{q}]}}$$

$$\frac{s[\mathtt{q}] \notin \mathtt{dom}(\Delta) \quad \exists E' \text{ such that } E \longrightarrow^* E' \xrightarrow{s:\mathtt{p}\to\mathtt{q}:l_k}, \quad \Delta\cdot s[\mathtt{p}] : [\mathtt{q}] \oplus \{l_i : T_i\}_{i\in I} \subseteq \mathtt{proj}(E') \quad k \in I}{(E,\Gamma,\Delta\cdot s[\mathtt{p}] : [\mathtt{q}] \oplus \{l_i : T_i\}_{i\in I}) \downarrow_{s[\mathtt{p}][\mathtt{q}]}}$$

$$\frac{a \in \mathtt{dom}(\Gamma)}{(E,\Gamma,\Delta) \downarrow_a}$$

We write $(E,\Gamma,\Delta) \Downarrow_m$ if $(E,\Gamma,\Delta) \longrightarrow^* (\Gamma,\Delta',E')$ and $(\Gamma,\Delta',E') \downarrow_m$.

We define the binary operator $\sqcup$ over global environments based on the inclusion of the syntax tree for global types. The operation is used to define the typed relation with respect to a global witness and the governed bisimulation. The operator $\sqcup$ is used to relate two different, but compatible observers, E_1 and E_2.

Definition 5.7. Let T_1 and T_2 denote local types as defined in § 3. We write $T_1 \sqsubseteq T_2$ if the syntax tree of T_2 includes one of T_1 as a leaf. We extend to $G_1 \sqsubseteq G_2$ by defining $\forall s[\mathtt{p}] : T_1 \in \mathtt{proj}(s : G_1), \exists s[\mathtt{p}] : T_2 \in \mathtt{proj}(s : G_2)$ and $T_1 \sqsubseteq T_2$. We define: $E_1 \sqcup E_2 = \{s : E_i(s) \mid E_j(s) \sqsubseteq E_i(s), i,j \in \{1,2\}, i \neq j\} \cup \{s : E_1(s), s' : E_2(s') \mid s \notin \mathtt{dom}(E_2), s' \notin \mathtt{dom}(E_1)\}$.

As an example for global types inclusion consider that:

$$[\mathtt{q}]?(U');T \sqsubseteq [\mathtt{p}]!\langle U\rangle;[\mathtt{q}]?(U');T$$

As an example of $E_1 \sqcup E_2$, let us define:

$$\begin{aligned} E_1 &= s_1 : \mathtt{p} \to \mathtt{q} : \langle U_1\rangle.\mathtt{p}' \to \mathtt{q}' : \langle U_2\rangle.\mathtt{p} \to \mathtt{q} : \langle U_3\rangle.\mathtt{end}\cdot s_2 : \mathtt{p} \to \mathtt{q} : \langle W_2\rangle.\mathtt{end} \\ E_2 &= s_1 : \mathtt{p} \to \mathtt{q} : \langle U_3\rangle.\mathtt{end}\cdot s_2 : \mathtt{p}' \to \mathtt{q}' : \langle W_1\rangle.\mathtt{p} \to \mathtt{q} : \langle W_2\rangle.\mathtt{end} \end{aligned}$$

Then

$$E_1 \sqcup E_2 = s_1 : \mathtt{p} \to \mathtt{q} : \langle U_1\rangle.\mathtt{p}' \to \mathtt{q}' : \langle U_2\rangle.\mathtt{p} \to \mathtt{q} : \langle U_3\rangle.\mathtt{end}\cdot s_2 : \mathtt{p}' \to \mathtt{q}' : \langle W_1\rangle.\mathtt{p} \to \mathtt{q} : \langle W_2\rangle.\mathtt{end}$$

The behavioural relation w.r.t. a global witness is defined below.

Definition 5.8 (Configuration relation)**.** The relation $\mathscr{R}$ is a *configuration relation* between two configurations $E_1,\Gamma \vdash P_1 \triangleright \Delta_1$ and $E_2,\Gamma \vdash P_2 \triangleright \Delta_2$, written

$$E_1 \sqcup E_2, \Gamma \vdash P \triangleright \Delta_1 \;\mathscr{R}\; P_2 \triangleright \Delta_2$$

if $E_1 \sqcup E_2$ is defined.

Proposition 5.9 (Decidability)**.**

(1) *Given E_1 and E_2, a problem whether $E_1 \sqcup E_2$ is defined or not is decidable and if it is defined, the calculation of $E_1 \sqcup E_2$ terminates.*

(2) *Given E, a set $\{E' \mid E \longrightarrow^* E'\}$ is finite.*

Proof. (1) since $T_1 \sqsubseteq T_2$ is a syntactic tree inclusion, it is reducible to a problem to check the isomorphism between two types. This problem is decidable [50].

(2) the global LTS has one-to-one correspondence with the LTS of global automata in [14] whose reachability set is finite. □

Definition 5.10 (Global configuration transition)**.** We write $E_1, \Gamma_1 \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} E_2, \Gamma_2 \vdash P_2 \triangleright \Delta_2$ if $E_1, \Gamma_1 \vdash P_1 \triangleright \Delta_1$, $\Gamma_1 \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} \Gamma_2 \vdash P_2 \triangleright \Delta_2$ and $(E_1, \Gamma_1, \Delta_1) \xrightarrow{\ell} (E_2, \Gamma_2, \Delta_2)$.

Note that $\Gamma_2 \vdash P_2 \triangleright \Delta_2$ immediately holds by Definition 4.3.

The proposition below states that the configuration LTS preserves the well-formedness.

Proposition 5.11 (Invariants)**.**

(1) $(E_1, \Gamma, \Delta_1) \xrightarrow{\ell} (E_2, \Gamma_2, \Delta_2)$ *implies that* $(E_2, \Gamma_2, \Delta_2)$ *is an environment configuration.*
(2) *If* $\Gamma \vdash P \triangleright \Delta$ *and* $P \longrightarrow P'$ *with* $\mathtt{co}(\Delta)$*, then* $E, \Gamma \vdash P \triangleright \Delta \longrightarrow E, \Gamma \vdash P' \triangleright \Delta'$ *and* $\mathtt{co}(\Delta')$*.*
(3) *If* $E_1, \Gamma_1 \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} E_2, \Gamma_2 \vdash P_2 \triangleright \Delta_2$ *then* $E_2, \Gamma_2 \vdash P_2 \triangleright \Delta_2$ *is a governance judgement.*

Proof. The proof for Part 1 and Part 3 can be found in Appendix B.3. Part 2 is verified by simple transitions using [Tau] in Figure 9. $\mathtt{co}(\Delta')$ is derived by Theorem 3.9. □

The definition of the reduction-closed congruence for governance follows. Below we define $E, \Gamma \vdash P \triangleright \Delta \Downarrow_m$ if $P \Downarrow_m$ and $(E, \Gamma, \Delta) \Downarrow_m$.

Definition 5.12 (Governed reduction-closed congruence)**.** A configuration relation $\mathscr{R}$ is *governed reduction-closed congruence* if $E, \Gamma \vdash P_1 \triangleright \Delta_1 \;\mathscr{R}\; P_2 \triangleright \Delta_2$ then

(1) $E, \Gamma \vdash P_1 \triangleright \Delta_1 \Downarrow_n$ if and only if $E, \Gamma \vdash P_2 \triangleright \Delta_2 \Downarrow_n$
(2)
- $P_1 \twoheadrightarrow P_1'$ if there exists P_2' such that $P_2 \twoheadrightarrow P_2'$ and $E, \Gamma \vdash P_1' \triangleright \Delta_1' \;\mathscr{R}\; P_2' \triangleright \Delta_2'$.
- the symmetric case.

(3) For all closed context C, such that $E, \Gamma \vdash C[P_1] \triangleright \Delta_1'$ and $E, \Gamma \vdash C[P_2] \triangleright \Delta_2'$ then $E, \Gamma \vdash C[P_1] \triangleright \Delta_1' \;\mathscr{R}\; C[P_2] \triangleright \Delta_2'$.

The union of all governed reduction-closed congruence relations is denoted as $\cong_g^s$.

5.2. **Globally governed bisimulation and its properties.**

This subsection introduces the globally governed bisimulation relation definition and studies its main properties.

Definition 5.13 (Globally governed bisimulation)**.** A configuration relation $\mathscr{R}$ is a *globally governed weak bisimulation* (or governed bisimulation) if whenever $E, \Gamma \vdash P_1 \triangleright \Delta_1 \;\mathscr{R}\; P_2 \triangleright \Delta_2$, it holds:

(1) $E, \Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} E_1', \Gamma' \vdash P_1' \triangleright \Delta_1'$ implies $E, \Gamma \vdash P_2 \triangleright \Delta_2 \overset{\hat{\ell}}{\Longrightarrow} E_2', \Gamma' \vdash P_2' \triangleright \Delta_2'$ such that $E_1' \sqcup E_2', \Gamma' \vdash P_1' \triangleright \Delta_1' \;\mathscr{R}\; P_2' \triangleright \Delta_2'$.
(2) The symmetric case.

The maximum bisimulation exists which we call *governed bisimilarity*, denoted by $\approx_g^s$. We sometimes leave environments implicit, writing e.g. $P \approx_g^s Q$.

Lemma 5.14.

(1) $\approx_g^s$ *is congruent.*
(2) $\cong_g^s \subseteq \approx_g^s$*.*

Proof. The proof of (1) is by a case analysis on the context structure. The interesting case is the parallel composition, which uses Proposition 5.11. See Appendix B.4.

The proof uses the technique from [19] (the external actions can be always tested). The proof can be found in Appendix B.5. □

Theorem 5.15 (Soundness and completeness)**.** $\approx_g^s = \cong_g^s$.

Proof. The fact that $\approx_g^s \subseteq \cong_g^s$ comes directly from the first part of Lemma 5.14. The proof is completed using the second part of Lemma 5.14. □

The next theorem clarifies the relation between the locally controlled bisimilarity $\approx^s$ and globally governed bisimilarity $\approx^s_g$.

Theorem 5.16. *If for all E such that $E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ then $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$. Also if $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$, then for all E, $E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$.*

Proof. The proof is based on the properties that exist between semantics of the environment tuples (Γ,Δ) and the semantics of the environment configurations (E,Γ,Δ). The full proof can be found in Appendix B.6. □

To clarify the above theorem, consider the following processes:

$$\begin{array}{rcl} P_1 & = & s_1[1][3]!\langle v\rangle;s_2[1][2]!\langle w\rangle;\mathbf{0} \mid s_1[2][3]!\langle v\rangle;s_2[2][1]?(x);s_2[2][3]!\langle x\rangle;\mathbf{0} \\ P_2 & = & s_1[1][3]!\langle v\rangle;\mathbf{0} \mid s_2[1][2]!\langle w\rangle;\mathbf{0} \mid s_1[2][3]!\langle v\rangle;s_2[2][1]?(x);s_2[2][3]!\langle x\rangle;\mathbf{0} \end{array}$$

We can show that $P_1 \approx^s P_2$. By Theorem 5.16, we expect that for all E, we have $E,\Gamma \vdash P_1 \triangleright \Delta_1$ and $E,\Gamma \vdash P_2 \triangleright \Delta_2$ then $E \vdash P_1 \approx^s_g P_2$. This is in fact true because the possible E that can type P_1 and P_2 are:

$$\begin{array}{rcl} E_1 & = & s_1 : 1 \to 3 : \langle U\rangle.2 \to 3 : \langle U\rangle.\mathtt{end} \cdot s_2 : 1 \to 2 : \langle W\rangle.2 \to 3 : \langle W\rangle.\mathtt{end} \\ E_2 & = & s_1 : 2 \to 3 : \langle U\rangle.1 \to 3 : \langle U\rangle.\mathtt{end} \cdot s_2 : 1 \to 2 : \langle W\rangle.2 \to 3 : \langle W\rangle.\mathtt{end} \end{array}$$

and all the up-to weakening instances E (see Lemma B.1) of E_1 and E_2.

To clarify the difference between $\approx^s$ and $\approx^s_g$, we introduce the notion of a *simple multiparty process* defined in [24]. A simple process contains only a single session so that it satisfies the progress property as proved in [24]. Formally a process P is *simple* when it is typable with a type derivation where the session typing in the premise and the conclusion of each prefix rule is restricted to at most a single session (i.e. any $\Gamma \vdash P \triangleright \Delta$ which appears in a derivation, Δ contains at most one session channel in its domain, see [24]). Since there is no interleaving of sessions in simple processes, the difference between $\approx^s$ and $\approx^s_g$ disappears.

Theorem 5.17 (Coincidence)**.** *Assume P_1 and P_2 are simple. If there exists E such that $E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$, then $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$.*

Proof. The proof follows the fact that if P is simple and $\Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} P' \triangleright \Delta'$ then $\exists E$ such that $E,\Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} P' \triangleright \Delta'$ to continue that if P_1 and P_2 are simple and there exists E such that $E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ then $\forall E, E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$. The result then comes by applying Theorem 5.16. The details of the proof are in the Appendix B.7. □

To clarify the above theorem, consider:

$$\begin{array}{rcl} P_1 & = & s[1][2]?(x);s[1][3]!\langle x\rangle;\mathbf{0} \mid s[2][1]!\langle v\rangle;\mathbf{0} \\ P_2 & = & s[1][3]!\langle v\rangle;\mathbf{0} \end{array}$$

It holds that for

$$E = s : 2 \to 1 : \langle U\rangle.1 \to 3 : \langle U\rangle.\mathtt{end}$$

We can easily reason that $E \vdash P_1 \approx^s_g P_2$ hence $P_1 \approx^s P_2$.

Example 5.18 (Governed bisimulation)**.** Recall the example from § 1 and Example 4.11. Q_1 is the process corresponding to a sequential thread (this corresponds to $P_1 \mid P_2$ in § 1), while Q_2 has a parallel thread instead of the sequential composition (this corresponds to $P_1 \mid R_2$ in § 1).

$$\begin{array}{rcl} Q_1 & = & (\nu\ s_b)(s_a[1][3]!\langle v\rangle;s_b[1][2]!\langle w\rangle;\mathbf{0} \mid s_b[2][1]?(x);\mathbf{0} \mid s_a[2][3]!\langle v\rangle;\mathbf{0}) \\ Q_2 & = & (\nu\ s_b)(s_a[1][3]!\langle v\rangle;s_b[1][2]!\langle w\rangle;\mathbf{0} \mid s_b[2][1]?(x);s_a[2][3]!\langle v\rangle;\mathbf{0}) \end{array}$$

Assume:

$$\begin{array}{rcl}\Gamma & = & a : G_a \cdot b : G_b \\ \Delta_0 & = & s_a[1] : [3]!\langle S\rangle;\mathtt{end} \cdot s_a[2] : [3]!\langle S\rangle;\mathtt{end}\end{array}$$

Then we have $\Gamma \vdash Q_1 \triangleright \Delta_0$ and $\Gamma \vdash Q_2 \triangleright \Delta_0$. Now assume the two global witnesses as:

$$\begin{array}{rcl}E_1 & = & s_a : 1 \to 3 : \langle S\rangle.2 \to 3 : \langle S\rangle.\mathtt{end} \\ E_2 & = & s_a : 2 \to 3 : \langle S\rangle.1 \to 3 : \langle S\rangle.\mathtt{end}\end{array}$$

Then the projection of E_1 and E_2 is given as:

$$\begin{array}{rcl}\mathtt{proj}(E_1) & = & s_a[1] : [3]!\langle S\rangle;\mathtt{end} \cdot s_a[2] : [3]!\langle S\rangle;\mathtt{end} \cdot s_a[3] : [1]?(S);[2]?(S);\mathtt{end} \\ \mathtt{proj}(E_2) & = & s_a[1] : [3]!\langle S\rangle;\mathtt{end} \cdot s_a[2] : [3]!\langle S\rangle;\mathtt{end} \cdot s_a[3] : [2]?(S);[1]?(S);\mathtt{end}\end{array}$$

with $\Delta_0 \subset \mathtt{proj}(E_1)$ and $\Delta_0 \subset \mathtt{proj}(E_2)$. The reader should note that the difference between E_1 and E_2 is the type of the participant 3 at s_a (the third mapping in E_1 and E_2).

By definition of the global environment configuration, we can write:

$$E_i, \Gamma \vdash Q_1 \triangleright \Delta_0 \text{ and } E_i, \Gamma \vdash Q_2 \triangleright \Delta_0 \text{ for } i = 1,2.$$

Both processes are well-formed global configurations under both witnesses. Now we can observe

$$\Gamma \vdash Q_1 \triangleright \Delta_0 \xrightarrow{s_a[2][3]!\langle v\rangle} \Gamma \vdash Q_1' \triangleright \Delta_0'$$

but

$$\Gamma \vdash Q_2 \triangleright \Delta_0 \overset{s_a[2][3]!\langle v\rangle}{\not\longrightarrow}$$

Hence $\Gamma \vdash Q_1 \triangleright \Delta_0 \not\approx^s Q_2 \triangleright \Delta_0$ as detailed in Example 4.11.

Similarly, we have:

$$E_2, \Gamma \vdash Q_1 \triangleright \Delta_0 \not\approx^s_g Q_2 \triangleright \Delta_0$$

because E_2 allows to output action $s_a[2][3]!\langle v\rangle$ by [Out] in Figure 9 (since $E_2 \xrightarrow{s_a:2\to 3:S} E_2'$).

On the other hand, since E_1 *forces* to wait for $s_a[2][3]!\langle v\rangle$,

$$E_1, \Gamma \vdash Q_1 \triangleright \Delta_0 \overset{s_a[2][3]!\langle v\rangle}{\not\longrightarrow}$$

because we cannot apply [Out] in Figure 9. E_1 does not allow to output action $s_a[2][3]!\langle v\rangle$ (since $E_1 \overset{s_a:2\to 3:S}{\not\longrightarrow}$). Hence Q_1 and Q_2 are bisimilar under E_1, i.e. $E_1, \Gamma \vdash Q_1 \triangleright \Delta_0 \approx^s_g Q_2 \triangleright \Delta_0$. This concludes the optimisation illustrated in § 1 is correct.

6. Usecase: UC.R2.13 "Acquire Data From Instrument" from the Ocean Observatories Initiative (OOI) [1]

The running example for the thread transformation in the previous sections is the minimum to demonstrate a difference between $\approx^s_g$ and $\approx^s$. This discipline can be applied to general situations where multiple agents need to interact following a global specification. Our governance bisimulation can be useful in other large applications, for example, it can be applied to the optimisation and verification of distributed systems, and the correctness of service communication. In this section, we present a reasoning example based on the real world usecase, UC.R2.13 "Acquire Data From Instrument", from the Ocean Observatories Initiative (OOI) [1], and show the optimisation and verification of network services.

In this usecase, we assume a user program (`U`) which is connected to the Integrated Observatory Network (ION). The ION provides the interface between users and remote sensing instruments. The user requests, via the ION agent services (`A`), the acquisition of processed data from an instrument (`I`). More specifically the user requests from the ION two different formats of the instrument data.

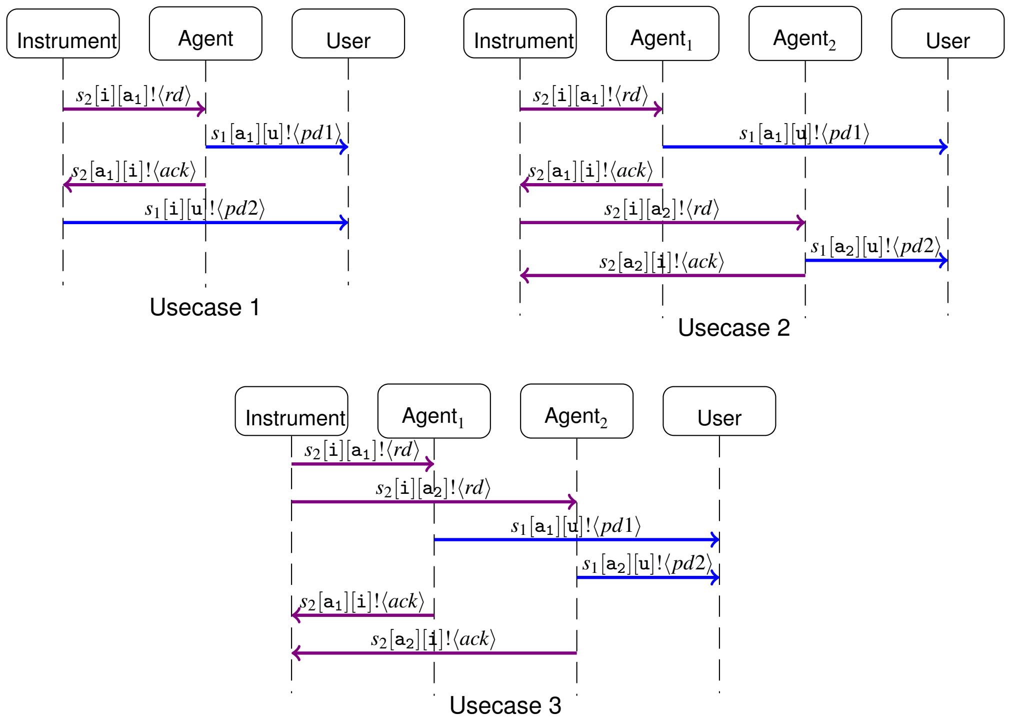
Instrument
Agent
User
$s_2[\mathtt{i}][\mathtt{a}_1]!\langle rd\rangle$
$s_1[\mathtt{a}_1][\mathtt{u}]!\langle pd1\rangle$
$s_2[\mathtt{a}_1][\mathtt{i}]!\langle ack\rangle$
$s_1[\mathtt{i}][\mathtt{u}]!\langle pd2\rangle$
Usecase 1
Instrument
Agent$_1$
Agent$_2$
User
$s_2[\mathtt{i}][\mathtt{a}_1]!\langle rd\rangle$
$s_1[\mathtt{a}_1][\mathtt{u}]!\langle pd1\rangle$
$s_2[\mathtt{a}_1][\mathtt{i}]!\langle ack\rangle$
$s_2[\mathtt{i}][\mathtt{a}_2]!\langle rd\rangle$
$s_1[\mathtt{a}_2][\mathtt{u}]!\langle pd2\rangle$
$s_2[\mathtt{a}_2][\mathtt{i}]!\langle ack\rangle$
Usecase 2
Instrument
Agent$_1$
Agent$_2$
User
$s_2[\mathtt{i}][\mathtt{a}_1]!\langle rd\rangle$
$s_2[\mathtt{i}][\mathtt{a}_2]!\langle rd\rangle$
$s_1[\mathtt{a}_1][\mathtt{u}]!\langle pd1\rangle$
$s_1[\mathtt{a}_2][\mathtt{u}]!\langle pd2\rangle$
$s_2[\mathtt{a}_1][\mathtt{i}]!\langle ack\rangle$
$s_2[\mathtt{a}_2][\mathtt{i}]!\langle ack\rangle$
Usecase 3

Figure 10: Three usecases from UC.R2.13 “Acquire Data From Instrument” in [1]

In the above usecase we distinguish two points of communication coordination: i) an internal ION multiparty communication and ii) an external communication between ION instruments and agents and the user. In other words it is natural to require the initiation of two multiparty session types to coordinate the services and clients involved in the usecase. The behaviour of the multiparty session connection between the User (U) and ION is dependent on the implementation and the synchronisation of the internal ION session.

Below we present three possible implementation scenarios and compare their behaviour with respect to the user program. Depending on the ION requirements we can choose the best implementation with the correct behaviour.

6.1. Usecase Scenario 1.

In the first scenario (depicted in Usecase 1 in Figure 10) the user program (U) wants to acquire the first format of data from the instrument (I) and at the same time acquire the second format of the data from an agent service (A). The communication between the agent (A) and the instrument happens internally in the ION on a separate private session.

(1) A new session connection s_1 is established between (U), (I) and (A).
(2) A new session connection s_2 is established between (A) and (I).
(3) (I) sends raw data through s_2 to (A).
(4) (A) sends processed data (format 1) through s_1 to (U).
(5) (A) sends the acknowledgement through s_2 to (I).
(6) (I) sends processed data (format 2) through s_1 to (U).

The above scenario is implemented as follows:

$$I_0 \mid A \mid U$$

where

$$\begin{array}{rcl} I_0 & = & a[\mathtt{i0}](s_1).\overline{b}[\mathtt{i0}](s_2).s_2[\mathtt{i0}][\mathtt{a_1}]!\langle\mathtt{rd}\rangle;s_2[\mathtt{i0}][\mathtt{a_1}]?(x);s_1[\mathtt{i0}][\mathtt{u}]!\langle\mathtt{pd}\rangle;\mathbf{0} \\ A & = & a[\mathtt{a_1}](s_1).b[\mathtt{a_1}](s_2).s_2[\mathtt{a_1}][\mathtt{i0}]?(x);s_1[\mathtt{a_1}][\mathtt{u}]!\langle\mathtt{pd}\rangle;s_2[\mathtt{a_1}][\mathtt{i0}]!\langle\mathtt{ack}\rangle;\mathbf{0} \\ U & = & \overline{a}[\mathtt{u}](s_1).s_1[\mathtt{u}][\mathtt{a_1}]?(x);s_1[\mathtt{u}][\mathtt{i0}]?(y);\mathbf{0} \end{array}$$

and `i0` is the instrument role, $\mathtt{a_1}$ is the agent role and `u` is the user role.

6.2. **Usecase scenario 2.** Use case scenario 1 implementation requires from the instrument program to process raw data in a particular format (format 2) before sending them to the user program. In a more modular and fine-grain implementation, the instrument program should only send raw data to the ION interface for processing and forwarding to the user. A separate session between the instrument and the ION interface and a separate session between the ION interface and the user make a distinction into different logical and processing levels.

To capture the above implementation we assume a scenario (depicted in Usecase 2 in Figure 10) with the user program (U), the instrument (I) and agents ($\mathtt{A}_1$) and ($\mathtt{A}_2$):

(1) A new session connection s_1 is established between (U), ($\mathtt{A}_1$) and ($\mathtt{A}_2$).
(2) A new session connection s_2 is established between ($\mathtt{A}_1$), ($\mathtt{A}_2$) and (I).
(3) (I) sends raw data through s_2 to ($\mathtt{A}_1$).
(4) ($\mathtt{A}_1$) sends processed data (format 1) through s_1 to (U).
(5) ($\mathtt{A}_1$) sends the acknowledgement through s_2 to (I).
(6) (I) sends raw data through s_2 to ($\mathtt{A}_2$).
(7) ($\mathtt{A}_2$) sends processed data (format 2) through s_1 to (U).
(8) ($\mathtt{A}_2$) sends the acknowledgement through s_2 to (I).

The above scenario is implemented as follows:

$$I_1 \mid A_1 \mid A_2 \mid U$$

where

$$\begin{array}{rcl} I_1 & = & \overline{b}[\mathtt{i}](s_2).s_2[\mathtt{i}][\mathtt{a_1}]!\langle\mathtt{rd}\rangle;s_2[\mathtt{i}][\mathtt{a_1}]?(x);s_2[\mathtt{i}][\mathtt{a_2}]!\langle\mathtt{rd}\rangle;s_2[\mathtt{i}][\mathtt{a_1}]?(x);\mathbf{0} \\ A_1 & = & a[\mathtt{a_1}](s_1).b[\mathtt{a_1}](s_2).s_2[\mathtt{a_1}][\mathtt{i}]?(x);s_1[\mathtt{a_1}][\mathtt{u}]!\langle\mathtt{pd}\rangle;s_2[\mathtt{a_1}][\mathtt{i}]!\langle\mathtt{ack}\rangle;\mathbf{0} \\ A_2 & = & a[\mathtt{a_2}](s_1).b[\mathtt{a_2}](s_2).s_2[\mathtt{a_2}][\mathtt{i}]?(x);s_1[\mathtt{a_2}][\mathtt{u}]!\langle\mathtt{pd}\rangle;s_2[\mathtt{a_2}][\mathtt{i}]!\langle\mathtt{ack}\rangle;\mathbf{0} \\ U & = & \overline{a}[\mathtt{u}](s_1).s_1[\mathtt{u}][\mathtt{a_1}]?(x);s_1[\mathtt{u}][\mathtt{a_2}]?(y);\mathbf{0} \end{array}$$

and `i` is the instrument role, $\mathtt{a_1}$ and $\mathtt{a_2}$ are the agent roles and `u` is the user role. Furthermore, for session s_1 we let role `i0` (from scenario 1) as $\mathtt{a_2}$, since we maintain the session s_1 as it is defined in the scenario 1.

6.3. **Usecase scenario 3.** A step further is to enhance the performance of usecase scenario 2 if the instrument (I) code in usecase scenario 2 can have a different implementation, where raw data is sent to both agents ($\mathtt{A}_1$, $\mathtt{A}_2$) before any acknowledgement is received. ION agents can process data in parallel resulting in an optimised implementation. This scenario is depicted in Usecase 3 in Figure 10.

(1) A new session connection s_1 is established between (U), ($\mathtt{A}_1$) and ($\mathtt{A}_2$).
(2) A new session connection s_2 is established between ($\mathtt{A}_1$), ($\mathtt{A}_2$) and (I).
(3) (I) sends raw data through s_2 to ($\mathtt{A}_1$).
(4) (I) sends raw data through s_2 to ($\mathtt{A}_2$).
(5) ($\mathtt{A}_1$) sends processed data (format 1) through s_1 to (U).

(6) $(\mathtt{A}_1)$ sends acknowledgement through s_2 to $(\mathtt{I})$.
(7) $(\mathtt{A}_2)$ sends processed data (format 2) through s_1 to $(\mathtt{U})$.
(8) $(\mathtt{A}_2)$ sends acknowledgement through s_2 to $(\mathtt{I})$.

The process is now refined as

$$I_2 \mid A_1 \mid A_2 \mid U$$

where

$$I_2 \quad = \quad \overline{b}[\mathtt{i}](s_2).s_2[\mathtt{i}][\mathtt{a}_1]!\langle\mathtt{rd}\rangle;s_2[\mathtt{i}][\mathtt{a}_2]!\langle\mathtt{rd}\rangle;s_2[\mathtt{i}][\mathtt{a}_1]?(x);s_2[\mathtt{i}][\mathtt{a}_1]?(x);\mathbf{0}$$

and $\mathtt{i}$ implements the instrument role, $\mathtt{a}_1$ and $\mathtt{a}_2$ are the agent roles and $\mathtt{u}$ is the user role.

6.4. **Bisimulations.** The main concern of the three scenarios is to implement the Integrated Ocean Network interface respecting the multiparty communication protocols.

Having the user process as the observer we can see that typed processes:

$$\Gamma \vdash I_0 \mid A \triangleright \Delta_0 \quad \text{and} \quad \Gamma \vdash I_1 \mid A_1 \mid A_2 \triangleright \Delta_1$$

are bisimilar (using $\approx^s$) since in both process we observe the following transition relations (recall that $\mathtt{i0} = \mathtt{a}_2$) :

$$\Gamma \vdash I_0 \mid A \triangleright \Delta_0 \overset{a[s](\mathtt{a}_1,\mathtt{i0})}{\longrightarrow} \overset{\tau}{\longrightarrow} \overset{s_1[\mathtt{a}_1][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow} \overset{s_1[\mathtt{i0}][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow}$$

and

$$\Gamma \vdash I_1 \mid A_1 \mid A_2 \triangleright \Delta_1 \overset{a[s](\mathtt{a}_1,\mathtt{a}_2)}{\longrightarrow} \overset{\tau}{\longrightarrow} \overset{s_1[\mathtt{a}_1][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow} \overset{s_1[\mathtt{a}_2][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow}$$

Next we give the bisimulation closure. Let:

$$\begin{array}{lclcl}
\Gamma \vdash I_0 \mid A \triangleright \Delta_0 & \overset{a[s](\mathtt{a}_1,\mathtt{a}_2)}{\longrightarrow} & \Gamma \vdash P_1 \triangleright \Delta_{01} & \overset{\tau}{\longrightarrow} & \Gamma \vdash P_2 \triangleright \Delta_{02} \\
& \overset{\tau}{\longrightarrow} & \Gamma \vdash P_3 \triangleright \Delta_{03} & \overset{s_1[\mathtt{a}_1][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow} & \Gamma \vdash P_4 \triangleright \Delta_{04} \\
& \overset{\tau}{\longrightarrow} & \Gamma \vdash P_5 \triangleright \Delta_{05} & \overset{s_1[\mathtt{i0}][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow} & \Gamma \vdash P_6 \triangleright \Delta_{06}
\end{array}$$

$$\begin{array}{lclcl}
\Gamma \vdash I_1 \mid A_1 \mid A_2 \triangleright \Delta_1 & \overset{a[s](\mathtt{a}_1,\mathtt{a}_2)}{\longrightarrow} & \Gamma \vdash Q_1 \triangleright \Delta_{11} & \overset{\tau}{\longrightarrow} & \Gamma \vdash Q_2 \triangleright \Delta_{12} \\
& \overset{\tau}{\longrightarrow} & \Gamma \vdash Q_3 \triangleright \Delta_{13} & \overset{s_1[\mathtt{a}_1][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow} & \Gamma \vdash Q_4 \triangleright \Delta_{14} \\
& \overset{\tau}{\longrightarrow} & \Gamma \vdash Q_5 \triangleright \Delta_{15} & \overset{\tau}{\longrightarrow} & \Gamma \vdash Q_6 \triangleright \Delta_{16} \\
& \overset{s_1[\mathtt{a}_2][\mathtt{u}]!\langle\mathtt{pd}\rangle}{\longrightarrow} & \Gamma \vdash P_7 \triangleright \Delta_{17} & \overset{\tau}{\longrightarrow} & \Gamma \vdash Q_8 \triangleright \Delta_{18}
\end{array}$$

The bisimulation closure is:

$$\begin{array}{rcl}
\mathscr{R} & = & \{(\Gamma \vdash I_0 \mid A \triangleright \Delta_0, \Gamma \vdash I_1 \mid A_1 \mid A_2 \triangleright \Delta_1), (\Gamma \vdash P_1 \triangleright \Delta_{01}, \Gamma \vdash Q_1 \triangleright \Delta_{11}) \\
& & (\Gamma \vdash P_2 \triangleright \Delta_{02}, \Gamma \vdash Q_2 \triangleright \Delta_{12}), (\Gamma \vdash P_3 \triangleright \Delta_{03}, \Gamma \vdash Q_3 \triangleright \Delta_{13}) \\
& & (\Gamma \vdash P_4 \triangleright \Delta_{04}, \Gamma \vdash Q_4 \triangleright \Delta_{14}), (\Gamma \vdash P_5 \triangleright \Delta_{05}, \Gamma \vdash Q_5 \triangleright \Delta_{15}) \\
& & (\Gamma \vdash P_5 \triangleright \Delta_{05}, \Gamma \vdash Q_6 \triangleright \Delta_{16}), (\Gamma \vdash P_6 \triangleright \Delta_{06}, \Gamma \vdash Q_7 \triangleright \Delta_{17}) \\
& & (\Gamma \vdash P_6 \triangleright \Delta_{06}, \Gamma \vdash Q_8 \triangleright \Delta_{18})\}
\end{array}$$

The two implementations (scenario 1 and scenario 2) are completely interchangeable with respect to $\approx^s$.

If we proceed with the case of the scenario 3 we can see that typed process $\Gamma \vdash I_2 \mid A_1 \mid A_2 \triangleright \Delta_2$ cannot be simulated (using $\approx^s$) by scenarios 1 and 2, since we can observe the execution:

$$\Gamma \vdash I_1 \mid A_1 \mid A_2 \triangleright \Delta_1 \xrightarrow{\tau} \xrightarrow{a[s](\mathtt{a}_1,\mathtt{a}_2)} \xrightarrow{s_1[\mathtt{a}_2][\mathtt{u}]!\langle \mathtt{pd}\rangle}$$

By changing the communication ordering in the ION private session s_2 we change the communication behaviour on the external session channel s_1. Nevertheless, the communication behaviour remains the same if we take into account the global multiparty protocol of s_1 and the way it governs the behaviour of the three usecase scenarios.

Hence we use $\approx^s_g$. The definition of the global environment is as follows:

$$E \;=\; s_1 : \mathtt{a}_1 \to \mathtt{u} : \langle \mathtt{PD}\rangle.\mathtt{a}_2 \to \mathtt{u} : \langle \mathtt{PD}\rangle.$$

The global protocol governs processes $I_1 \mid A_1 \mid A_2$ (similarly, $I_0 \mid A$) and $I_2 \mid A_1 \mid A_2$ to always observe action $\xrightarrow{s_1[\mathtt{a}_2][\mathtt{u}]!\langle \mathtt{pd}\rangle}$ after action $\xrightarrow{s_1[\mathtt{a}_1][\mathtt{u}]!\langle \mathtt{pd}\rangle}$ for both processes.

Also note that the global protocol for s_2 is not present in the global environment, because s_2 is restricted. The specification and implementation of session s_2 are abstracted from the behaviour of session s_1.

7. Related and Future Work

Session types [21, 46] have been studied over the last decade for a wide range of process calculi and programming languages, as a typed foundation for structured communication programming. Recently several works developed multiparty session types and their extensions. While typed behavioural equivalences are one of the central topics of the π-calculus, surprisingly the typed behavioural semantics based on session types have been less explored and focusing only on binary (two-party) sessions.

In this section we first compare our work in a broader context in relation with the previous work on typed behavioural theories in the π-calculus. We then discuss and compare our work with more specific results: behavioural theories in the binary session types and bisimulations defined with environments.

Typed behavioural theories in the π-calculus. An effect of types to behaviours of processes was first studied with the IO-subtyping in [42]. Since types can limit contexts (environments) where processes can interact, typed equivalences usually offer *coarse* semantics than untyped semantics. After [42], many works on typed π-calculi have investigated correctness of encodings of known concurrent and sequential calculi in order to examine semantic effects of proposed typing systems.

The type discipline closely related to session types is a family of linear typing systems. The work [26] first proposed a linearly typed barbed congruence and reasoned a tail-call optimisation of higher-order functions which are encoded as processes. The work [47] had used a bisimulation of graph-based types to prove the full abstraction of encodings of the polyadic synchronous π-calculus into the monadic synchronous π-calculus. Later typed equivalences of a family of linear and affine calculi [4, 5, 48] were used to encode PCF [33, 43], the simply typed λ-calculi with sums and products, and system F [18] fully abstractly (a fully abstract encoding of the λ-calculi was an open problem in [34]). The work [49] proposed a new bisimilarity method associated with linear type structure and strong normalisation. It presented applications to reason secrecy in programming languages. A subsequent work [23] adapted these results to a practical direction. It proposes new typing systems for secure higher-order and multi-threaded programming languages. In these works, typed properties, linearity and liveness, play a fundamental role in the analysis. In general, linear types are suitable to encode "sequentiality" in the sense of [2, 25].

Our first bisimulation $\approx^s$ is classified as one of linear bisimulations, capturing a mixture between shared behaviours (interactions at shared names) and linear behaviours (interactions at session names). Hence it is coarser than the untyped semantics (see Example 4.11). Contrast to these linear bisimulations, the governance bisimulation offers more *fine-grained* equivalences since the same typable processes are observed in different ways depending on a witness (global types). See the last paragraph for a relationship with environment bisimulations.

Behavioural theories in the binary session types. Our work in [30, 31] develops an *asynchronous binary* session typed behavioural theory with event operations. A labelled transition system is defined on session type process judgements and ensures properties, such as linearity in the presence of asynchronous queues. We then apply the theory to validate a transformation between threaded and event servers based on the Lauer-Needham duality [32]. For reasoning this transformation, we use a confluence technique developed in [40]. We have established several up-to techniques using confluence and determinacy properties on reductions on typed session names. These useful up-to techniques are still applicable to our standard and governed bisimulations since the up-to bisimulation obtained in [30, 31] is only concerned on the local τ-actions on session names. It is an interesting future work to investigate the up-to techniques or useful axioms which are specific to the governed bisimulation.

The work [39] proves that the proof conversions induced by a Linear Logic interpretation of session types coincide with an observational equivalence over a strict subset of the binary synchronous session processes. The approach is extended to the binary asynchronous and binary synchronous polymorphic session processes in [16] and [8], respectively.

The main focus of our paper is *multiparty* session types and governed bisimulation, whose definitions and properties crucially depend on information of global types. In the first author's PhD thesis [28], we studied how governed bisimulations can be systematically developed under various semantics including three kinds of asynchronous semantics by modularly changing the LTS for processes, environments and global types. For governed bisimulations, we can reuse all of the definitions among four semantics by only changing the conditions of the LTS of global types to suit each semantics.

Another recent work [13] gives a fully abstract encoding of a *binary* synchronous session typed calculus into a linearly typed π-calculus [4].[2] We believe the same encoding method is smoothly applicable to $\approx^s$ since it is defined solely based on the projected types (i.e. local types). However a governed bisimulation requires a global witness, hence the additional global information would be required for full abstraction.

Behavioural semantics defined with environments. The constructions of our work are hinted by [20] which studies typed behavioural semantics for the π-calculus with IO-subtyping where an LTS for pairs of typing environments and processes is used for defining typed testing equivalences and barbed congruence. On the other hand, in [20], the type environment indexing the observational equivalence resembles more a dictator where the refinement can be obtained by the fact that the observer has only partial knowledge on the typings, than a coordinator like our approach. Several papers have developed bisimulations for the higher-order π-calculus or its variants using the information of the environments. In [44] the authors take a general approach for developing a behavioural theory for higher order processes, both in the λ-calculus and the π-calculus. The bisimulation relations are developed in the presence of an environment knowledge for higher order communication.

[2]The work [12] also uses linear types to encode binary session types for the first and higher-order π-calculi [35], but it does not study full abstraction results with respect to a behavioural equivalence or bisimulation.

Congruence and compositionality of processes are restricted with respect to the environment. A recent paper [27] uses a pair of a process and an observer knowledge set for the LTS. The knowledge set contains a mapping from first order values to the higher-order processes, which allows a tractable higher-order behavioural theory using the first-order LTS.

We record a choreographic type as the witness in the environment to obtain fine-grained bisimulations of multiparty processes. The highlight of our bisimulation construction is an effective use of the semantics of global types for LTSs of processes (cf. [Inv] in Figure 9 and Definition 5.10). Global types can guide the coordination among parallel threads giving explicit protocols, hence it is applicable to a semantic-preserving optimisation (cf. Example 5.18 and § 6).

Future work. While it is known that it is undecidable to check $P \approx Q$ in the full π-calculus, it is an interesting future topic to investigate automated bisimulation-checking techniques or finite axiomatisations for the governed bisimulations for some subset of multiparty session processes.

More practical future direction is incorporating with, not only well-known subtyping of session types [13, 17] but also advanced refinements for communication optimisation (such as asynchronous subtyping [36, 37] and asynchronous distributed states [10]) to seek practical applications of governed bisimulations to, e.g. parallel algorithms [38] and distributed computing [1].

References

[1] Ocean Observatories Initiative (OOI). `http://www.oceanobservatories.org/`.
[2] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *TCS*, 163:409–470, 2000.
[3] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi-calculus. *TCS*, 195(2):291–324, 1998.
[4] M. Berger, K. Honda, and N. Yoshida. Sequentiality and the π-calculus. In *Proc. TLCA'01*, volume 2044 of *LNCS*, pages 29–45, 2001.
[5] M. Berger, K. Honda, and N. Yoshida. Genericity and the pi-calculus. *Acta Inf.*, 42(2-3):83–141, 2005.
[6] G. Bernardi and M. Hennessy. Using higher-order contracts to model session types. *CoRR*, abs/1310.6176, 2013.
[7] L. Bettini et al. Global progress in dynamically interleaved multiparty sessions. In *CONCUR*, volume 5201 of *LNCS*, pages 418–433. Springer, 2008.
[8] L. Caires, J. A. Pérez, F. Pfenning, and B. Toninho. Behavioral polymorphism and parametricity in session-based communication. In *ESOP*, volume 7792 of *LNCS*, pages 330–349. Springer, 2013.
[9] W3C Web Services Choreography. `http://www.w3.org/2002/ws/chor/`.
[10] T.-C. Chen and K. Honda. Specifying stateful asynchronous properties for distributed programs. In *CONCUR*, volume 7454 of *LNCS*, pages 209–224. Springer, 2012.
[11] M. Coppo, M. Dezani-Ciancaglini, N. Yoshida, and L. Padvani. Global progress in dynamically interleaved multiparty sessions. *Mathematical Structures in Computer Science*. To appear. Available from `http://mrg.doc.ic.ac.uk/publications.html`.
[12] O. Dardha, E. Giachino, and D. Sangiorgi. Session types revisited. In *Proceedings of the 14th symposium on Principles and practice of declarative programming*, PPDP '12, pages 139–150, New York, NY, USA, 2012. ACM.
[13] R. Demangeon and K. Honda. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR*, volume 6901 of *LNCS*, pages 280–296. Springer, 2011.
[14] P.-M. Deniélou and N. Yoshida. Multiparty session types meet communicating automata. In *ESOP*, volume 7211 of *LNCS*, pages 194–213. Springer, 2012.
[15] P.-M. Deniélou, N. Yoshida, A. Bejleri, and R. Hu. Parameterised multiparty session types. *Logical Methods in Computer Science*, 8(4), 2012.
[16] H. DeYoung, L. Caires, F. Pfenning, and B. Toninho. Cut Reduction in Linear Logic as Asynchronous Session-Typed Communication. In P. Cégielski and A. Durand, editors, *CSL*, volume 16 of *LIPIcs*, pages 228–242. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
[17] S. Gay and M. Hole. Subtyping for Session Types in the Pi-Calculus. *Acta Informatica*, 42(2/3):191–225, 2005.
[18] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. CUP, 1989.

[19] M. Hennessy. *A Distributed Pi-Calculus*. CUP, 2007.
[20] M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Mathematical Structures in Computer Science*, 14(5):651–684, 2004.
[21] K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP'98*, volume 1381 of *LNCS*, pages 22–138. Springer, 1998.
[22] K. Honda and N. Yoshida. On reduction-based process semantics. *TCS*, 151(2):437–486, 1995.
[23] K. Honda and N. Yoshida. A uniform type structure for secure information flow. *TOPLAS*, 29(6), 2007.
[24] K. Honda, N. Yoshida, and M. Carbone. Multiparty Asynchronous Session Types. In *POPL'08*, pages 273–284. ACM, 2008.
[25] J. M. E. Hyland and C. H. L. Ong. On full abstraction for PCF. *Inf. & Comp.*, 163:285–408, 2000.
[26] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the Pi-Calculus. *TOPLAS*, 21(5):914–947, Sept. 1999.
[27] V. Koutavas and M. Hennessy. A testing theory for a higher-order cryptographic language. In *ESOP*, volume 6602 of *LNCS*, pages 358–377, 2011.
[28] D. Kouzapas. *A study of bisimulation theory for session types*. PhD thesis, Department of Computing, Imperial College London, June 2013.
[29] D. Kouzapas and N. Yoshida. Globally governed session semantics. In P. R. D'Argenio and H. C. Melgratti, editors, *CONCUR*, volume 8052 of *LNCS*, pages 395–409. Springer, 2013.
[30] D. Kouzapas, N. Yoshida, and K. Honda. On asynchronous session semantics. In *FMOODS/FORTE*, volume 6722 of *Lecture Notes in Computer Science*, pages 228–243, 2011.
[31] D. Kouzapas, N. Yoshida, R. Hu, and K. Honda. On asynchronous eventful session semantics. *MSCS*. To appear.
[32] H. C. Lauer and R. M. Needham. On the duality of operating system structures. *SIGOPS Oper. Syst. Rev.*, 13(2):3–19, 1979.
[33] R. Milner. Fully abstract models of typed lambda-calculi. *TCS*, 4(1):1 – 22, 1977.
[34] R. Milner. Functions as processes. *MSCS*, 2(2):119–141, 1992.
[35] D. Mostrous and N. Yoshida. Two session typing systems for higher-order mobile processes. In *TLCA'07*, volume 4583 of *LNCS*, pages 321–335. Springer, 2007.
[36] D. Mostrous and N. Yoshida. Session-based communication optimisation for higher-order mobile processes. In *TLCA'09*, volume 5608 of *LNCS*, pages 203–218. Springer, 2009.
[37] D. Mostrous, N. Yoshida, and K. Honda. Global principal typing in partially commutative asynchronous sessions. In *ESOP'09*, number 5502 in LNCS. Springer, 2009.
[38] N. Ng, N. Yoshida, and K. Honda. Multiparty Session C: Safe parallel programming with message optimisation. In *TOOLS*, volume 7304 of *LNCS*, pages 202–218. Springer, 2012.
[39] J. A. Pérez, L. Caires, F. Pfenning, and B. Toninho. Linear Logical Relations for Session-Based Concurrency. In *ESOP*, volume 7211 of *LNCS*, pages 539–558. Springer, 2012.
[40] A. Philippou and D. Walker. On confluence in the pi-calculus. In *ICALP'97*, volume 1256 of *Lecture Notes in Computer Science*, pages 314–324. Springer, 1997.
[41] B. Pierce. *Types and Programming Languages*. MIT Press, 2002.
[42] B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *MSCS*, 6(5):409–454, 1996.
[43] G. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223 – 255, 1977.
[44] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *LICS*, pages 293–302. IEEE Computer Society, 2007.
[45] D. Sangiorgi and D. Walker. *The π-Calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
[46] K. Takeuchi, K. Honda, and M. Kubo. An Interaction-based Language and its Typing System. In *PARLE'94*, volume 817 of *LNCS*, pages 398–413, 1994.
[47] N. Yoshida. Graph types for monadic mobile processes. In *FSTTCS*, volume 1180 of *LNCS*, pages 371–386. Springer, 1996.
[48] N. Yoshida, M. Berger, and K. Honda. Strong Normalisation in the π-Calculus. *Information and Computation*, 191(2004):145–202, 2004.
[49] N. Yoshida, K. Honda, and M. Berger. Linearity and bisimulation. In *FoSSaCs02*, volume 2303 of *LNCS*, pages 417–433. Springer, 2002.
[50] N. Yoshida and V. T. Vasconcelos. Language primitives and type discipline for structured communication-based programming revisited: Two systems for higher-order session communication. *Electr. Notes Theor. Comput. Sci.*, 171(4):73–93, 2007.

APPENDIX A. PROOF FOR THEOREM 3.9

We first state a substitution lemma, used in the proof of the subject reduction theorem (Theorem 3.9). The statement is from the substitution lemma in [7].

Lemma A.1 (Substitution)**.**

- *If* $\Gamma \cdot x : S \vdash P \triangleright \Delta$ *and* $\Gamma \vdash v : S$ *then* $\Gamma \vdash P\{v/x\} \triangleright \Delta$.
- *If* $\Gamma \vdash P \triangleright \Delta \cdot y : T$ *then* $\Gamma \vdash P\{s[\mathtt{p}]/y\} \triangleright \Delta \cdot s[\mathtt{p}] : T$

Proof. The proof is a standard induction on the typing derivation. □

We state Subject congruence.

Lemma A.2 (Subject Congruence)**.** *Let* $\Gamma \vdash P_1 \triangleright \Delta$ *and* $P_1 \equiv P_2$. *Then* $\Gamma \vdash P_2 \triangleright \Delta$.

Proof. The proof is standard and straightforward by induction. □

A.1. **Proof for Theorem 3.9.**

Proof. For the subject reduction proof we apply induction on the structure of the reduction relation. We present the two key cases: the rest is similar with the subject reduction theorem for the communication typing system in [7].

Case: [Link]
Let:

$$P = a[\mathtt{p}](x_1).P_1 \mid \dots \mid \overline{a}[n](x_n).P_n$$

We apply the typing rules [MAcc], [MReq] and [Conc] to obtain $\Gamma \vdash P \triangleright \Delta$ with $\mathtt{co}(\Delta)$. Next assume that:

$$P \longrightarrow P' = (\nu\ s)(P_1\{s[1]/x_1\} \mid \dots \mid P_n\{s[n]/x_n\})$$

From rule [Conc] and Lemma A.1, we obtain:

$$\Gamma \vdash P_1\{s[1]/x_1\} \mid \dots \mid P_n\{s[n]/x_n\} \triangleright \Delta \cdot s[1] : T_1 \dots s[n] : T_n$$

with $\mathtt{fco}(\{s[1] : T_1 \dots s[n] : T_n\})$ (since each T_i is a projection of $\Gamma(a)$). We apply rule [SRes] to obtain $\Gamma \vdash P' \triangleright \Delta$, as required.

Case: [Comm]
Let:

$$P = s[\mathtt{p}][\mathtt{q}]!\langle v\rangle;P_1 \mid s[\mathtt{q}][\mathtt{p}]?(x);P_2$$

and

$$P \longrightarrow P_1 \mid P_2\{v/x\}$$

We apply typing rules [Send], [Rcv] and [Conc] to obtain: $\Gamma \vdash P \triangleright \Delta$ with:

$$\Delta = \Delta_1 \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T_{\mathtt{p}} \cdot s[\mathtt{q}] : [\mathtt{p}]?(U);T_{\mathtt{q}}$$

and using Lemma A.1 we obtain that $\Gamma \vdash P_1 \mid P_2\{v/x\} \triangleright \Delta_1$.
From the induction hypothesis we know that $\mathtt{co}(\Delta)$. From the coherency of Δ and from Proposition 3.6 we obtain:

$$\begin{array}{c} \mathtt{co}(\Delta_1) \\ ([\mathtt{q}]!\langle U\rangle;T_{\mathtt{p}})\lceil \mathtt{q} \quad = \quad \overline{([\mathtt{p}]?(U);T_{\mathtt{q}})\lceil \mathtt{p}} \end{array}$$

The latter result implies that $!\langle U\rangle;(T_{\mathtt{p}}\lceil\mathtt{q}) = \overline{?(U);(T_{\mathtt{q}}\lceil\mathtt{p})}$, which in turn implies that $T_{\mathtt{p}}\lceil\mathtt{q} = \overline{T_{\mathtt{q}}\lceil\mathtt{p}}$. From the last result and the coherency of Δ_1 we get:

$$\mathtt{co}(\Delta_1 \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot s[\mathtt{q}] : T_{\mathtt{q}})$$

Hence $\Gamma \vdash P_1 \mid P_2\{v/x\} \triangleright \Delta'$ with $\Delta' = \Delta_1 \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot s[\mathtt{q}] : T_{\mathtt{q}}$, and Δ' is coherent. □

APPENDIX B. PROOFS FOR BISIMULATION PROPERTIES

B.1. **Proof for Lemma 4.9.**

Proof. We use the coinductive method based on the bisimilarity definition. Assume that for $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$, we have $\Delta_1 \rightleftharpoons \Delta_2$. Then by the definition of $\rightleftharpoons$, there exists Δ such that:

$$\Delta_1 \longrightarrow^* \Delta \text{ and } \Delta_2 \longrightarrow^* \Delta \tag{B.1}$$

Now assume that $\Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} P_1' \triangleright \Delta_1'$ then, $\Gamma \vdash P_2 \triangleright \Delta_2 \xRightarrow{\ell} P_2' \triangleright \Delta_2'$ and by the typed transition definition we obtain $(\Gamma, \Delta_1) \xrightarrow{\ell} (\Gamma, \Delta_1')$ and $(\Gamma, \Delta_2) \xRightarrow{\ell} (\Gamma, \Delta_2')$. We need to show that $\Delta_1' \rightleftharpoons \Delta_2'$.

We prove by a case analysis on the transition $\xrightarrow{\ell}$ on (Γ, Δ_1) and (Γ, Δ_2).

Case $\ell = \tau$:
We use the fact that $\xrightarrow{\tau}$ with $\equiv$ coincides with $\longrightarrow$. By Theorem 3.9, we obtain that if $\Gamma \vdash P_1 \triangleright \Delta_1$ and $P_1 \longrightarrow P_1'$ then $\Gamma \vdash P_1' \triangleright \Delta_1'$ and $\Delta_1 \longrightarrow \Delta_1'$ or $\Delta_1 = \Delta_1'$.

For environment Δ_2 we obtain that if $\Gamma \vdash P_2 \triangleright \Delta_2$ and $P_2 \twoheadrightarrow P_2'$ then $\Gamma \vdash P_2' \triangleright \Delta_2'$ and $\Delta_2 \longrightarrow^* \Delta_2'$. From the coinductive hypothesis in (B.1), we obtain that there exists Δ such that:

$$\begin{array}{c} \Delta_1 \longrightarrow \Delta_1' \longrightarrow^* \Delta \\ \Delta_2 \longrightarrow^* \Delta_2' \longrightarrow^* \Delta \end{array}$$

or

$$\begin{array}{c} \Delta_1 = \Delta_1' \longrightarrow^* \Delta \\ \Delta_2 \longrightarrow^* \Delta_2' \longrightarrow^* \Delta \end{array}$$

as required.

Case $\ell = a[\mathtt{p}](s)$ or $\ell = \overline{a}[\mathtt{p}](s)$:
The environment tuple transition on ℓ is:

$$\begin{array}{rcl} (\Gamma, \Delta_1) & \xrightarrow{\ell} & (\Gamma, \Delta_1 \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot \ldots \cdot s[\mathtt{q}] : T_{\mathtt{q}}) \\ (\Gamma, \Delta_2) & \Longrightarrow\xrightarrow{\ell}\Longrightarrow & (\Gamma, \Delta_2'' \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot \ldots \cdot s[\mathtt{q}] : T_{\mathtt{q}}) \end{array}$$

We set

$$\Delta' = \Delta \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot \ldots \cdot s[\mathtt{q}] : T_{\mathtt{q}}$$

to obtain:

$$\begin{array}{rcl} \Delta_1 \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot \ldots \cdot s[\mathtt{q}] : T_{\mathtt{q}} & \longrightarrow^* & \Delta' \\ \Delta_2'' \cdot s[\mathtt{p}] : T_{\mathtt{p}} \cdot \ldots \cdot s[\mathtt{q}] : T_{\mathtt{q}} & \longrightarrow^* & \Delta' \end{array}$$

by the coinductive hypothesis (B.1).

Case $\ell = s[\mathtt{p}][\mathtt{q}]!\langle v\rangle$:

We know from the definition of environment transition, $s[\mathtt{q}] \notin \mathtt{dom}(\Delta_1)$ and $s[\mathtt{q}] \notin \mathtt{dom}(\Delta_2)$ and thus $s[\mathtt{q}] \notin \mathtt{dom}(\Delta)$.

From the typed transition we know that Δ_1 and Δ_2 have the form:

$$\begin{array}{rcl}\Delta_1 & = & s[\mathtt{p}] : [\mathtt{q}]!\langle v\rangle;T \cdot \Delta_1''\\ \Delta_2 & = & s[\mathtt{p}] : [\mathtt{q}]!\langle v\rangle;T \cdot \Delta_2''\end{array}$$

and from the coinductive hypothesis (B.1), there exists $\Delta = s[\mathtt{p}] : [\mathtt{q}]!\langle v\rangle;T \cdot \Delta''$ such that:

$$\begin{array}{rcl}\Delta_1 & \longrightarrow^* & \Delta\\ \Delta_2 & \longrightarrow^* & \Delta\end{array}$$

Note that there is no reduction on $s[\mathtt{p}]$ because $s[\mathtt{q}] \notin \mathtt{dom}(\Delta)$.
From the environment transition relation we obtain that:

$$\begin{array}{rcl}\Delta_1' & = & s[\mathtt{p}] : T \cdot \Delta_1''\\ \Delta_2' & = & s[\mathtt{p}] : T \cdot \Delta_2''\end{array}$$

The last step is to set $\Delta' = s[\mathtt{p}] : T \cdot \Delta''$ to obtain $\Delta_1' \longrightarrow^* \Delta'$ and $\Delta_2' \longrightarrow^* \Delta'$ as required.

Case $\ell = s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])$:
This case follows a similar argumentation with the case $\ell = s[\mathtt{p}][\mathtt{q}]!\langle v\rangle$.
We know from the definition of environment transition, $s[\mathtt{q}] \notin \mathtt{dom}(\Delta_1)$ and $s[\mathtt{q}] \notin \mathtt{dom}(\Delta_2)$, thus $s[\mathtt{q}] \notin \mathtt{dom}(\Delta)$.

From the typed transition we know that Δ_1 and Δ_2 have the form:

$$\begin{array}{rcl}\Delta_1 & = & s[\mathtt{p}] : [\mathtt{q}]!\langle T'\rangle;T \cdot \Delta_1''\\ \Delta_2 & = & s[\mathtt{p}] : [\mathtt{q}]!\langle T'\rangle;T \cdot \Delta_2''\end{array}$$

and from the coinductive hypothesis (B.1), there exists $\Delta = s[\mathtt{p}] : [\mathtt{q}]!\langle T'\rangle;T \cdot \Delta''$ such that:

$$\begin{array}{rcl}\Delta_1 & \longrightarrow^* & \Delta\\ \Delta_2 & \longrightarrow^* & \Delta\end{array}$$

From the environment transition relation we obtain that:

$$\begin{array}{rcl}\Delta_1' & = & s[\mathtt{p}] : T \cdot \Delta_1''\\ \Delta_2' & = & s[\mathtt{p}] : T \cdot \Delta_2''\end{array}$$

The last step is to set $\Delta' = s[\mathtt{p}] : T \cdot \Delta''$ to obtain $\Delta_1' \longrightarrow^* \Delta'$ and $\Delta_2' \longrightarrow^* \Delta'$, as required.

The remaining cases on session channel actions are similar. □

B.2. **Weakening and strengthening.** The following lemmas are essential for invariant properties.

Lemma B.1 (Weakening). (1) *If $E,\Gamma \vdash P \triangleright \Delta$ then*

- $E \cdot s : G,\Gamma \vdash P \triangleright \Delta$.
- *$E = E' \cdot s : G$ and $\exists G'$ such that $\{s : G'\} \longrightarrow^* \{s : G\}$ then $E' \cdot s : G',\Gamma \vdash P \triangleright \Delta$.*

(2) *If $(E,\Gamma,\Delta) \xrightarrow{\ell} (E,\Gamma',\Delta')$ then*

- $(E \cdot s : G,\Gamma,\Delta) \xrightarrow{\ell} (E \cdot s : G,\Gamma',\Delta')$
- *If $E = E' \cdot s : G$ and $\{s : G'\} \longrightarrow^* \{s : G\}$ then $(E' \cdot s : G',\Gamma,\Delta) \xrightarrow{\ell} (E' \cdot s : G',\Gamma',\Delta')$*

(3) *If* $E,\Gamma \vdash P_1 \triangleright \Delta_2 \approx^s_g P_2 \triangleright \Delta_2$
- $E \cdot s : G, \Gamma \vdash P_1 \triangleright \Delta_2 \approx^s_g P_2 \triangleright \Delta_2$
- *If* $E = E' \cdot s : G$ *and* $\{s : G'\} \longrightarrow^* \{s : G\}$ *then* $E' \cdot s : G', \Gamma \vdash P_1 \triangleright \Delta_2 \approx^s_g P_2 \triangleright \Delta_2$

Proof. We only show Part 1. Other parts are similar.

- From the governance judgement definition we have that $E \longrightarrow^* E'$ and $\mathtt{proj}(E') \supseteq \Delta$. Let $E \cdot s : G \longrightarrow E' \cdot s : G$. Then $\mathtt{proj}(E' \cdot s : G) = \mathtt{proj}(E') \cup \mathtt{proj}(s : G) \supseteq \mathtt{proj}(E') \supseteq \Delta$.
- From the governance judgement definition we have that there exist E_1 an G_1 such that $E' \cdot s : G \longrightarrow^* E_1 \cdot s : G_1$ and $\mathtt{proj}(E_1 \cdot s : G_1) \supseteq \Delta$. Let $E' \cdot s : G' \longrightarrow^* E' \cdot s : G \longrightarrow^* E_1 \cdot s : G_1$. Hence the result is immediate.

$\square$

Lemma B.2 (Strengthening)**.** (1) *If* $E \cdot s : G, \Gamma \vdash P \triangleright \Delta$ *and*
- *If* $s \notin \mathtt{fn}(P)$ *then* $E, \Gamma \vdash P \triangleright \Delta$
- *If* $\exists G', G_1$ *s.t.* $E \cdot s : G \longrightarrow^* E_2 \cdot s : G' \longrightarrow^* E_1 \cdot s : G_1$ *with* $\mathtt{proj}(E_1 \cdot s : G_1) \supseteq \Delta$*, then* $E \cdot s : G', \Gamma \vdash P \triangleright \Delta$

(2) *If* $(E \cdot s : G, \Gamma, \Delta) \xrightarrow{\ell} (E' \cdot s : G, \Gamma', \Delta')$ *then*
- $(E, \Gamma, \Delta) \xrightarrow{\ell} (E', \Gamma', \Delta')$
- *If* $\exists G'$ *s.t.* $E \cdot s : G \longrightarrow^* E_2 \cdot s : G' \longrightarrow^* E_1 \cdot s : G_1$ *with* $\mathtt{proj}(E_1 \cdot s : G_1) \supseteq \Delta$*,* $(E \cdot s : G', \Gamma, \Delta) \xrightarrow{\ell} (E' \cdot s : G', \Gamma', \Delta')$

(3) *If* $E \cdot s : G, \Gamma \vdash P_1 \triangleright \Delta_2 \approx^s_g P_2 \triangleright \Delta_2$
- *If* $s \notin \mathtt{fn}(P)$ *then* $E, \Gamma \vdash P_1 \triangleright \Delta_2 \approx^s_g P_2 \triangleright \Delta_2$
- *If* $\exists G'$ *s.t.* $E \cdot s : G \longrightarrow^* E_2 \cdot s : G' \longrightarrow^* E_1 \cdot s : G_1$ *with* $\mathtt{proj}(E_1 \cdot s : G_1) \supseteq \Delta$*,* $E \cdot s : G', \Gamma \vdash P_1 \triangleright \Delta_2 \approx^s_g P_2 \triangleright \Delta_2$

Proof. We prove for part 1. Other parts are similar.

- From the governance judgement definition we have that $E \cdot s : G \longrightarrow^* E_1 \cdot s : G_1$ and $\mathtt{proj}(E_1 \cdot s : G_1) = \mathtt{proj}(E_1) \cup \mathtt{proj}(s : G_1) \supseteq \Delta$. Since $s \notin \mathtt{fn}(P)$ then $s \notin \mathtt{dom}(\Delta)$, then $\mathtt{proj}(s : G_1) \cap \Delta = \emptyset$. So $\mathtt{proj}(E_1) \supseteq \Delta$ and $E \longrightarrow^* E_1$.
- The result is immediate from the definition of governance judgement.

$\square$

B.3. **Configuration Transition Properties.**

Lemma B.3.

- *If* $E \overset{s:\mathtt{p}\to\mathtt{q}:U}{\longrightarrow} E'$ *then* $\{s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle; T_\mathtt{p}, s[\mathtt{q}] : [\mathtt{p}]?(U); T_\mathtt{q}\} \subseteq \mathtt{proj}(E)$ *and* $\{s[\mathtt{p}] : T_\mathtt{p}, s[\mathtt{q}] : T_\mathtt{q}\} \subseteq \mathtt{proj}(E')$.
- *If* $E \overset{s:\mathtt{p}\to\mathtt{q}:l}{\longrightarrow} E'$ *then* $\{s[\mathtt{p}] : [\mathtt{q}] \oplus \{l_i : T_{i\mathtt{p}}\}, s[\mathtt{q}] : [\mathtt{p}]\&\{l_i : T_{i\mathtt{q}}\}\} \subseteq \mathtt{proj}(E)$ *and* $\{s[\mathtt{p}] : T_{k\mathtt{p}}, s[\mathtt{q}] : T_{k\mathtt{q}}\} \subseteq \mathtt{proj}(E')$

Proof. Part 1: We apply inductive hypothesis on the structure of the definition of $s : \mathtt{p} \to \mathtt{q} : U$. The base case

$$\{s : \mathtt{p} \to \mathtt{q} : \langle U\rangle.G\} \overset{s:\mathtt{p}\to\mathtt{q}:U}{\longrightarrow} \{s : G\}$$

is easy since

$$\{s[\mathtt{p}] : (\mathtt{p} \to \mathtt{q} : \langle U \rangle.G)\lceil \mathtt{p}, s[\mathtt{q}] : (\mathtt{p} \to \mathtt{q} : \langle U \rangle.G)\lceil \mathtt{q}\} =$$
$$\{s[\mathtt{p}] : [\mathtt{q}]!\langle U \rangle; T_{\mathtt{p}}, s[\mathtt{q}] : [\mathtt{p}]?(U); T_{\mathtt{q}}\} \subseteq \mathtt{proj}(s : \mathtt{p} \to \mathtt{q} : \langle U \rangle.G)$$

and

$$\{s[\mathtt{p}] : G\lceil \mathtt{p}, s[\mathtt{q}] : G\lceil \mathtt{q}\} = \{s[\mathtt{p}] : T_{\mathtt{p}}, s[\mathtt{q}] : T_{\mathtt{q}}\} \subseteq \mathtt{proj}(s : G)$$

The main induction rule concludes that:

$$\{s : \mathtt{p}' \to \mathtt{q}' : \langle U \rangle.G\} \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} \{s : G\}$$

if $\mathtt{p} \neq \mathtt{p}'$ and $\mathtt{q} \neq \mathtt{q}'$ and $\{s : G\} \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} \{s : G'\}$. From the induction hypothesis we know that:

$$\begin{aligned} \{s[\mathtt{p}] : [\mathtt{q}]!\langle U \rangle; T_{\mathtt{p}}, s[\mathtt{q}] : [\mathtt{p}]?(U); T_{\mathtt{q}}\} &\subseteq \mathtt{proj}(s : G) \\ \{s[\mathtt{p}] : T_{\mathtt{p}}, s[\mathtt{q}] : T_{\mathtt{q}}\} &\subseteq \mathtt{proj}(s : G') \end{aligned}$$

to conclude that:

$$\{s[\mathtt{p}] : (\mathtt{p}' \to \mathtt{q}' : \langle U \rangle.G)\lceil \mathtt{p}, s[\mathtt{q}] : (\mathtt{p}' \to \mathtt{q}' : \langle U \rangle.G)\lceil \mathtt{q}\} =$$
$$\{s[\mathtt{p}] : G\lceil \mathtt{p}, s[\mathtt{q}] : G\lceil \mathtt{q}\} =$$
$$\{s[\mathtt{p}] : [\mathtt{q}]!\langle U \rangle; T_{\mathtt{p}}, s[\mathtt{q}] : [\mathtt{p}]?(U); T_{\mathtt{q}}\} \subseteq \mathtt{proj}(s : G)$$

and

$$\{s[\mathtt{p}] : (\mathtt{p}' \to \mathtt{q}' : \langle U \rangle.G')\lceil \mathtt{p}, s[\mathtt{q}] : (\mathtt{p}' \to \mathtt{q}' : \langle U \rangle.G')\lceil \mathtt{q}\} =$$
$$\{s[\mathtt{p}] : G'\lceil \mathtt{p}, s[\mathtt{q}] : G'\lceil \mathtt{q}\} =$$
$$\{s[\mathtt{p}] : T_{\mathtt{p}}, s[\mathtt{q}] : T_{\mathtt{q}}\} \subseteq \mathtt{proj}(s : G')$$

as required.

Part 2: Similar.

Part 3: From the global configuration transition relation (Definition 5.10), we obtain that

$$\begin{aligned} (E_1, \Gamma_1, \Delta_1) &\xrightarrow{\ell} (E_2, \Gamma_2, \Delta_2) \\ \Gamma_1 \vdash P_1 \triangleright \Delta_1 &\xrightarrow{\ell} \Gamma_2 \vdash P_2 \triangleright \Delta_2 \end{aligned}$$

Then from the definition of governed environment, we can show that $E_2, \Gamma_2 \vdash P_2 \triangleright \Delta_2$ is a governed judgement: this is because $(E_2, \Gamma_2, \Delta_2)$ is an environment configuration, hence $\exists E_2',\ E_2 \longrightarrow E_2'$ and $\mathtt{proj}(E_2') \supseteq \Delta_2$. □

Proof for Proposition 5.11.

Proof. **(1)** We apply induction on the definition structure of $\xrightarrow{\ell}$.

Basic Step:

Case: $\ell = \overline{a}[s](A)$. From rule [Acc] we obtain

$$(E_1, \Gamma_1, \Delta_1) \xrightarrow{\ell} (E_1 \cdot s : G, \Gamma_1, \Delta_1 \cdot \{s[\mathtt{p}_i] : s\lceil \mathtt{p}_i\}_{\mathtt{p}_i \in A})$$

From the environment configuration definition we obtain that

$$\exists E_1' \text{ such that } E_1 \longrightarrow^* E_1', \quad \mathtt{proj}(E_1') \supseteq \Delta_1$$

We also obtain that $\mathtt{proj}(s : G) \supseteq \{s[\mathtt{p}_i] : G\lceil \mathtt{p}_i\}_{i\in A}$. Thus we can conclude that

$$E_1 \cdot s : G \longrightarrow^* E_1' \cdot s : G$$
$$\mathtt{proj}(E_1 \cdot s : G) \supseteq \Delta_1 \cdot \{s[\mathtt{p}_i] : G\lceil \mathtt{p}_i\}_{\mathtt{p}_i\in A}$$

Case: $\ell = \overline{a}[s](A)$. Similar as above.

Case: $\ell = s[\mathtt{p}][\mathtt{q}]!\langle v\rangle$. From rule [Out] we obtain

$$(E_1,\Gamma,\Delta \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T) \quad \xrightarrow{\ell} \quad (E_2,\Gamma,\Delta \cdot s[\mathtt{p}] : T) \tag{B.2}$$
$$\mathtt{proj}(E_1) \quad \supseteq \quad \Delta \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T \tag{B.3}$$
$$E_1 \quad \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} \quad E_2 \tag{B.4}$$

From (B.3), we obtain $\mathtt{proj}(E_1) \supseteq \Delta \cdot \{s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T \cdot s[\mathtt{q}] : [\mathtt{p}]?(U);T'\}$ and from (B.4) and Lemma B.3, we obtain that $\mathtt{proj}(E_2) \supseteq \Delta \cdot \{s[\mathtt{p}] : T \cdot s[\mathtt{q}] : T'\}$.

Case: $\ell = s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])$.

$$(E_1,\Gamma,\Delta \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle T\mathtt{p}'\rangle;T) \quad \xrightarrow{\ell} \quad (E_2 \cdot s : G,\Gamma,\Delta \cdot s[\mathtt{p}] : T \cdot \{s[\mathtt{p}_i] : G\lceil \mathtt{p}_i\}) \tag{B.5}$$
$$\mathtt{proj}(E_1) \quad \supseteq \quad \Delta \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle T_{\mathtt{p}}'\rangle;T \tag{B.6}$$
$$E_1 \quad \xrightarrow{s:\mathtt{p}\to\mathtt{q}:T_{\mathtt{p}}'} \quad E_2 \tag{B.7}$$
$$\mathtt{proj}(s : G) \quad \supseteq \quad \{s[\mathtt{p}_i] : G\lceil \mathtt{p}_i\} \tag{B.8}$$

From (B.6) we obtain $\mathtt{proj}(E_1) \supseteq \Delta \cdot \{s[\mathtt{p}] : [\mathtt{q}]!\langle U\rangle;T \cdot s[\mathtt{q}] : [\mathtt{p}]?(U);T'\}$ and from (B.7) and Lemma B.3 we obtain that $\mathtt{proj}(E_2) \supseteq \Delta \cdot \{s[\mathtt{p}] : T \cdot s[\mathtt{q}] : T'\} \supset \Delta \cdot s[\mathtt{p}] : T$. From (B.8) we obtain that $\mathtt{proj}(E_2 \cdot s : G) \supseteq \Delta \cdot s[\mathtt{p}] : T \cdot \{s[\mathtt{p}_i] : G\lceil \mathtt{p}_i\}$, as required.
The rest of the base cases are similar.

Inductive Step:
The inductive rule for environment configuration is [Inv]. Let $(E_1,\Gamma_1,\Delta_1) \xrightarrow{\ell} (E_2,\Gamma_2,\Delta_2)$. From rule [Inv] we obtain:

$$E_1 \quad \longrightarrow^* \quad E_1' \tag{B.9}$$
$$(E_1',\Gamma_1,\Delta_1) \quad \xrightarrow{\ell} \quad (E_2,\Gamma_2,\Delta_2) \tag{B.10}$$

From the inductive hypothesis we know that, from (B.10), there exists E_3 such that $E_2 \longrightarrow^* E_3$ and $\Delta_2 \subseteq \mathtt{proj}(E_3)$. Then the result is by (B.9). □

Lemma B.4.

(1) *If* $(E,\Gamma,\Delta_1) \xrightarrow{\ell} (E',\Gamma',\Delta_2)$ *then* $(\Gamma,\Delta_1) \xrightarrow{\ell} (\Gamma',\Delta_2)$
(2) *If* $(E,\Gamma,\Delta_1) \xrightarrow{\ell} (E',\Gamma',\Delta_1')$ *and* $\Delta_1 \rightleftharpoons \Delta_2$ *then* $(E,\Gamma,\Delta_2) \stackrel{\ell}{\Longrightarrow} (E',\Gamma',\Delta_2')$
(3) *If* $(\Gamma,\Delta_1) \xrightarrow{\ell} (\Gamma',\Delta_2)$ *then there exists* E *such that* $(E,\Gamma,\Delta_1) \xrightarrow{\ell} (E',\Gamma',\Delta_2)$
(4) *If* $(E,\Gamma,\Delta \cdot s[\mathtt{p}] : T_{\mathtt{p}}) \xrightarrow{\ell} (E',\Gamma,\Delta' \cdot s[\mathtt{p}] : T_{\mathtt{p}})$ *then* $(E,\Gamma,\Delta) \xrightarrow{\ell} (E',\Gamma,\Delta')$
(5) *If* $(E,\Gamma,\Delta_1) \xrightarrow{\ell} (E',\Gamma,\Delta_2)$ *then* $(E,\Gamma,\Delta_1 \cdot \Delta) \xrightarrow{\ell}_s (E',\Gamma,\Delta_2 \cdot \Delta)$ *provided that if* $(E,\Gamma,\Delta) \xrightarrow{\ell'} (E,\Gamma,\Delta')$ *then* $\ell \not\asymp \ell'$

Proof. Part 1:
The proof for part 1 is easy to be implied by a case analysis on the configuration transition definition with respect to environment transition definition.

Part 2:
By the case analysis on ℓ.
Case $\ell = \tau$**:** The result is trivial.
Case $\ell = \overline{a}[\mathtt{p}](s)$ **or** $\ell = a[\mathtt{p}](s)$**:** The result comes from a simple transition.
Case $\ell = s[\mathtt{p}][\mathtt{q}]!\langle v \rangle$**:** $\Delta_1 \rightleftharpoons \Delta_2$ implies $\Delta_1 \longrightarrow^* \Delta$ and $\Delta_2 \longrightarrow^* \Delta$ for some Δ and $\Delta = \Delta' \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U \rangle; T$.

Hence $(E, \Gamma, \Delta_2) \Longrightarrow (E, \Gamma, \Delta) \xrightarrow{\ell}$ as required.
Case $\ell = s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])$**:** $\Delta_1 \rightleftharpoons \Delta_2$ implies $\Delta_1 \longrightarrow^* \Delta$ and $\Delta_2 \longrightarrow^* \Delta$ for some Δ and $\Delta = \Delta' \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle T' \rangle; T$.
$(E, \Gamma, \Delta_2) \Longrightarrow (E, \Gamma, \Delta) \xrightarrow{\ell}$ as required. The remaining cases are similar.

Part 3:
We do a case analysis on ℓ.
Cases $\ell = \tau, \ell = \overline{a}[\mathtt{p}](s), \ell = a[\mathtt{p}](s)$**:** The result holds for any E.
Case $\ell = s[\mathtt{p}][\mathtt{q}]!\langle v \rangle : \Delta_1 = \Delta_1' \cdot \Delta_1''$ with $\Delta_1'' = s[\mathtt{p}] : [\mathtt{q}]!\langle U \rangle; T_\mathtt{p} \cdot \ldots \cdot s[\mathtt{r}] : T_\mathtt{r}$ Choose $E = E' \cdot s : G$ with $\Delta_1'' \subseteq \mathtt{proj}(s : G)$ and $s[\mathtt{q}] : [\mathtt{p}]?(U); T_\mathtt{q} \in \mathtt{proj}(s : G)$ and $\Delta_1 \subseteq \mathtt{proj}(E)$. By the definition of configuration transition relation, we obtain $(E, \Gamma, \Delta) \xrightarrow{\ell} (E, \Gamma', \Delta_2)$, as required.

Remaining cases are similar.

Part 4:
$(E, \Gamma, \Delta \cdot s[\mathtt{p}] : T_\mathtt{p}) \xrightarrow{\ell} (E', \Gamma, \Delta' \cdot s[\mathtt{p}] : T_\mathtt{p})$ implies that $s[\mathtt{p}] \notin \mathtt{subj}(\ell)$. The result then follows from the definition of configuration transition.

Part 5:
Case $\ell = \tau, \ell = \overline{a}[\mathtt{p}](s), \ell = a[\mathtt{p}](s)$**:** The result holds by definition of the configuration transition.
Case $\ell = s[\mathtt{p}][\mathtt{q}]!\langle U \rangle$**:** we have that $\Delta_1 = \Delta_1' \cdot s[\mathtt{p}] : [\mathtt{q}]!\langle U \rangle; T$ and $E \xrightarrow{s:\mathtt{p} \to \mathtt{q}:U} E'$. $s[\mathtt{q}] \in \Delta$, then by definition of weak configuration pair we have $\Delta = \Delta'' \cdot s[\mathtt{q}] : [\mathtt{p}]?(U); T$ and $(E, \Gamma, \Delta) \xrightarrow{s[\mathtt{q}][\mathtt{p}]?\langle U \rangle}$. But this contradicts with the assumption $\ell \not\asymp \ell'$, so $s[\mathtt{q}] \notin \Delta$. By the definition of configuration pair transition we obtain that $(E, \Gamma, \Delta_1 \cdot \Delta) \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle U \rangle} (E, \Gamma, \Delta_2 \cdot \Delta)$. Remaining cases are similar. □

B.4. **Proof for Lemma 5.14 (1).**

Proof. Since we are dealing with closed processes, the interesting case is parallel composition. We need to show that if $E, \Gamma \vdash P \triangleright \Delta_1 \approx_g^s E, \Gamma \vdash Q \triangleright \Delta_2$ then for all R such that $E, \Gamma \vdash P \mid R \triangleright \Delta_3, E, \Gamma \vdash Q \mid R \triangleright \Delta_4$ then $E, \Gamma \vdash P \mid R \triangleright \Delta_3 \approx_g^s Q \mid R \triangleright \Delta_4$.

We define the following configuration relation.

$$\begin{array}{rl} S = & \{(E, \Gamma \vdash P \mid R \triangleright \Delta_3,\ E, \Gamma \vdash Q \mid R \triangleright \Delta_4) \mid \\ & E, \Gamma \vdash P \triangleright \Delta_1 \approx_g^s Q \triangleright \Delta_2, \\ & \forall R \text{ such that } E, \Gamma \vdash P \mid R \triangleright \Delta_3, E, \Gamma \vdash Q \mid R \triangleright \Delta_4\} \end{array}$$

Before we proceed to a case analysis, we extract general results. Let $\Gamma \vdash P \triangleright \Delta_1, \Gamma \vdash Q \triangleright \Delta_2, \Gamma \vdash R \triangleright \Delta_5, \Gamma \vdash P \mid R \triangleright \Delta_3, \Gamma \vdash Q \mid R \triangleright \Delta_4$ then from typing rule [Conc] we obtain

$$\Delta_3 = \Delta_1 \cup \Delta_5 \tag{B.11}$$

$$\Delta_4 = \Delta_2 \cup \Delta_5 \tag{B.12}$$

$$\Delta_1 \cap \Delta_5 = \emptyset \tag{B.13}$$

$$\Delta_2 \cap \Delta_5 = \emptyset \tag{B.14}$$

We prove that S is a bisimulation. There are three cases:
Case: 1

$$E, \Gamma \vdash P \mid R \triangleright \Delta_3 \xrightarrow{\ell} E_1', \Gamma \vdash P' \mid R \triangleright \Delta_3'$$

with $\mathtt{bn}(\ell) \cap \mathtt{fn}(R) = \emptyset$.
From typed transition definition we have that:

$$P \mid R \xrightarrow{\ell} P' \mid R \tag{B.15}$$

$$(E, \Gamma, \Delta_3) \xrightarrow{\ell} (E_1', \Gamma, \Delta_3') \tag{B.16}$$

Transition (B.15) and rule $\langle \text{Par} \rangle$ (LTS in Figure 7) imply:

$$P \xrightarrow{\ell} P' \tag{B.17}$$

From (B.11), transition (B.16) can be written as $(E, \Gamma, \Delta_1 \cup \Delta_5) \xrightarrow{\ell} (E_1', \Gamma, \Delta_1' \cup \Delta_5)$, to conclude from part 4 of Lemma B.4, that:

$$(E, \Gamma, \Delta_1) \xrightarrow{\ell} (E_1', \Gamma, \Delta_1') \tag{B.18}$$

$$\mathtt{subj}(\ell) \notin \mathtt{dom}(\Delta_5) \tag{B.19}$$

Transitions (B.17) and (B.18) imply $E, \Gamma \vdash P \triangleright \Delta_1 \xrightarrow{\ell} E_1', \Gamma \vdash P' \triangleright \Delta_1'$. From the definition of set S we obtain $E, \Gamma \vdash Q \triangleright \Delta_2 \overset{\ell}{\Longrightarrow} E_2', \Gamma \vdash Q' \triangleright \Delta_2'$.
From the typed transition definition we have that:

$$Q \overset{\ell}{\Longrightarrow} Q' \tag{B.20}$$

$$(E, \Gamma, \Delta_2) \overset{\ell}{\Longrightarrow} (E_2', \Gamma, \Delta_2') \tag{B.21}$$

From (B.19) and part 5 of Lemma B.4 we can write: $(E, \Gamma, \Delta_2 \cup \Delta_5) \overset{\ell}{\Longrightarrow} (E_2', \Gamma, \Delta_2' \cup \Delta_5)$, which implies, from (B.20), $E, \Gamma \vdash Q \mid R \triangleright \Delta_4 \overset{\ell}{\Longrightarrow} E_2', \Gamma \vdash Q' \mid R \triangleright \Delta_4'$. Furthermore, we can see that:

$$(E_1' \sqcup E_2', \Gamma \vdash P' \mid R \triangleright \Delta_4' \ , \ E_1' \sqcup E_2', \Gamma \vdash Q' \mid R \triangleright \Delta_4') \in S$$

as required, noting $E_1' \sqcup E_2'$ is defined by the definition of S.

Case: 2

$$E, \Gamma \vdash P \mid R \triangleright \Delta_3 \xrightarrow{\tau} E' \vdash P' \mid R' \triangleright \Delta_3'$$

From the typed transition definition, we have that:

$$P \mid R \xrightarrow{\tau} P' \mid R' \tag{B.22}$$

$$(E, \Gamma, \Delta_3) \xrightarrow{\tau} (E', \Gamma, \Delta_3') \tag{B.23}$$

From (B.22) and rule $\langle\text{Tau}\rangle$, we obtain

$$P \xrightarrow{\ell} P' \qquad \text{(B.24)}$$

$$R \xrightarrow{\ell'} R' \qquad \text{(B.25)}$$

From (B.11), transition (B.23) can be written $(E,\Gamma,\Delta_1 \cup \Delta_5) \xrightarrow{\tau} (E',\Gamma,\Delta_1' \cup \Delta_5')$, to conclude that

$$(E,\Gamma,\Delta_1) \xrightarrow{\ell} (E',\Gamma,\Delta_1') \qquad \text{(B.26)}$$

$$(E,\Gamma,\Delta_5) \xrightarrow{\ell'} (E',\Gamma,\Delta_5') \qquad \text{(B.27)}$$

From (B.24) and (B.26), we conclude that $E,\Gamma \vdash P \triangleright \Delta_1 \xrightarrow{\ell} E',\Gamma \vdash P' \triangleright \Delta_1'$ and from (B.25) and (B.27), we have $E,\Gamma \vdash R \triangleright \Delta_5 \xrightarrow{\ell'} E',\Gamma \vdash R' \triangleright \Delta_5'$.
From the definition of set S, we obtain that $E,\Gamma \vdash Q \triangleright \Delta_2 \overset{\ell}{\Longrightarrow} E,\Gamma \vdash Q' \triangleright \Delta_2'$ implies

$$Q \overset{\ell}{\Longrightarrow} Q' \qquad \text{(B.28)}$$

$$(E,\Gamma,\Delta_2) \overset{\ell}{\Longrightarrow} (E',\Gamma,\Delta_2') \qquad \text{(B.29)}$$

From (B.25), we obtain that $Q \mid R \overset{\tau}{\Longrightarrow} Q' \mid R'$ and $(E,\Gamma,\Delta_2 \cup \Delta_5) \overset{\tau}{\Longrightarrow} (E',\Gamma,\Delta_2' \cup \Delta_5')$, implies:

$$E,\Gamma \vdash Q \mid R \triangleright \Delta_4 \overset{\tau}{\Longrightarrow} E' \vdash Q' \mid R' \triangleright \Delta_4'$$

with

$$(E',\Gamma \vdash P' \mid R' \triangleright \Delta_4' \;,\; E',\Gamma \vdash Q' \mid R' \triangleright \Delta_4') \in S$$

as required.

Case: 3

$$E,\Gamma \vdash P \mid R \triangleright \Delta_3 \xrightarrow{\ell} E',\Gamma \vdash P \mid R' \triangleright \Delta_3'$$

This case is similar with the above cases.

B.5. **Proof for Lemma 5.14 (2).** The proof for the completeness follows the technique which uses the testers in [19]. We need to adapt the testers to multiparty session types.

Definition B.5 (Definability)**.** Let $\mathtt{obj}(\ell)$ and $\mathtt{subj}(\ell)$ to denote a set of object and subject of ℓ, respectively. Let N be a finite set of shared names and session endpoints for testing the receiving objects defined as $N \ ::= \ \emptyset \ \mid \ N \cdot s[\mathtt{p}] \ \mid \ N \cdot a$. An external action ℓ is *definable* if for a set of names N, fresh session succ, n is the dual endpoint of $\mathtt{subj}(\ell)$ (i.e. the dual endpoint of $s[\mathtt{p}][\mathtt{q}]$ is $s[\mathtt{q}]$), there is a *testing process* $T\langle N, \text{succ}, \ell\rangle$ with the property that for every process P and $\mathtt{fn}(P) \subseteq N$,

- $E_1,\Gamma \vdash P \triangleright \Delta_1 \xrightarrow{\ell} E_1',\Gamma' \vdash P' \triangleright \Delta_1'$ implies that $E,\Gamma \vdash T\langle N, \text{succ}, \ell\rangle \mid P \triangleright \Delta \twoheadrightarrow$ $E',\Gamma \vdash (\nu\ \mathtt{bn}(\ell), b)(P' \mid \text{succ}[1][2]!\langle \mathtt{obj}(\ell), n\rangle; \mathbf{0}) \triangleright \Delta'$
- $E,\Gamma \vdash T\langle N, \text{succ}, \ell\rangle \mid P \triangleright \Delta \twoheadrightarrow E',\Gamma \vdash Q \triangleright \Delta'$ and $E',\Gamma \vdash Q \triangleright \Delta' \Downarrow \text{succ}$ implies for some $E_1,\Gamma \vdash P \triangleright \Delta_1 \xrightarrow{\ell} E_1',\Gamma' \vdash P' \triangleright \Delta_1'$, we have $Q \equiv (\nu\ \mathtt{bn}(\ell), b)(P' \mid \text{succ}[1][2]!\langle \mathtt{obj}(\ell), n\rangle; \mathbf{0})$.

Hereafter we omit the environments if they are obvious from the context.

Lemma B.6 (Definability)**.** *Every external action is definable.*

Proof. The cases of the input actions and the session initialisation (accept and request) are straightforward [19]:

(1) $T\langle N, \text{succ}, a[A](s)\rangle =$
$(\nu\, b)(\overline{a}[n](x).\text{succ}!\langle \mathtt{tt}\rangle; b[1](x).R \mid a[\mathtt{p}_1](x).b[1](x).R_1 \mid \cdots \mid a[\mathtt{p}_m](x).b[1](x).R_m)$
with $\mathtt{p}_1,\dots,\mathtt{p}_m \notin A$ and $\{\mathtt{p}_1,\dots,\mathtt{p}_m\}\cup A$ complete w.r.t. $n = \max(\{\mathtt{p}_1,\dots,\mathtt{p}_m\}\cup A)$.
(2) $T\langle N, \text{succ}, s[\mathtt{p}][\mathtt{q}]?\langle v\rangle\rangle = s[\mathtt{q}][\mathtt{p}]!\langle v\rangle; \text{succ}[1][2]!\langle s[\mathtt{q}]\rangle; \mathbf{0}$
(3) $T\langle N, \text{succ}, s[\mathtt{p}][\mathtt{q}]\&l\rangle = s[\mathtt{q}][\mathtt{p}]\oplus l; \text{succ}[1][2]!\langle s[\mathtt{q}]\rangle; \mathbf{0}$
(4) $T\langle N, \text{succ}, \overline{a}[A](s)\rangle = (\nu\, b)(a[\mathtt{p}_1](x).\text{succ}!\langle \mathtt{tt}\rangle; b[1](x).R_1 \mid \dots \mid a[\mathtt{p}_m](x).; b[1](x).R_m)$
with $\mathtt{p}_1,\dots,\mathtt{p}_m \notin A$ and $\{\mathtt{p}_1,\dots,\mathtt{p}_m\}\cup A$ complete w.r.t. $\max(\{\mathtt{p}_1,\dots,\mathtt{p}_m\}\cup A)$.

The requirements of Definition B.5 is verified straightforwardly.

For the output cases, we use the matching operator as [19, § 2.7].

(5) $T\langle N, \text{succ}, s[\mathtt{p}][\mathtt{q}]!\langle v\rangle\rangle =$
$s[\mathtt{q}][\mathtt{p}]?(x);$
$\mathtt{if}\ x = v\ \mathtt{then}\ \text{succ}[1][2]!\langle x, s[\mathtt{q}]\rangle; \mathbf{0}\ \mathtt{else}\ (\nu\, b)(b[1](x).\text{succ}[1][2]!\langle x, s[\mathtt{q}]\rangle; \mathbf{0})$
(6) $T\langle N, \text{succ}, s[\mathtt{p}][\mathtt{q}]!(v)\rangle =$
$s[\mathtt{q}][\mathtt{p}]?(x);$
$\mathtt{if}\ x \notin N\ \mathtt{then}\ \text{succ}[1][2]!\langle x, s[\mathtt{q}]\rangle; \mathbf{0}\ \mathtt{else}\ (\nu\, b)(b[1](x).\text{succ}[1][2]!\langle x, s[\mathtt{q}]\rangle; \mathbf{0})$
(7) $T\langle N, \text{succ}, s[\mathtt{p}][\mathtt{q}]\oplus l_k\rangle =$
$s[\mathtt{q}][\mathtt{p}]\&\{l_k : \text{succ}[1][2]!\langle s[\mathtt{q}]\rangle; \mathbf{0},\quad l_i : (\nu\, b)(b[1](x).\text{succ}[1][2]!\langle s[\mathtt{q}]\rangle; \mathbf{0})\}_{i\in I\setminus k}$

The requirements of Definition B.5 are straightforward to verify. Note that we need to have process $\text{succ}[1][2]!\langle s[\mathtt{q}]\rangle; \mathbf{0}$ on both conditions in the if-statement since succ is a session channel (see [If] in Figure 6 in § 3). ☐

The next lemma follows [19, Lemma 2.38].

Lemma B.7 (Extrusion). *Assume succ is fresh and $b \notin \{\vec{m}\}\cup \mathtt{fn}(P)\cup \mathtt{fn}(Q)$ and $\{\vec{m}\} \subseteq \mathtt{fn}(v) \subseteq \{\vec{n}\}$.*

$$E, \Gamma \vdash (\nu \vec{m}, b)(P \mid succ[1][2]!\langle \vec{n}, s[\mathtt{q}]\rangle; \mathbf{0} \mid \prod_i R_i) \triangleright \Delta_1 \tag{B.30}$$

$$\cong\ (\nu \vec{m}, b)(Q \mid succ[1][2]!\langle \vec{n}, s[\mathtt{q}]\rangle; \mathbf{0} \mid \prod_i R_i) \triangleright \Delta_2 \tag{B.31}$$

with $R_i = b[1](x).R'_i$ then

$$E', \Gamma' \vdash P \triangleright \Delta'_1 \cong Q \triangleright \Delta'_2 \tag{B.32}$$

Proof. Let relation

$$\begin{aligned}\mathscr{S} \quad = \quad & \{(E', \Gamma' \vdash P \triangleright \Delta'_1, E', \Gamma' \vdash Q \triangleright \Delta'_2) \mid \\ & E, \Gamma \vdash (\nu \vec{m}, b)(P \mid \text{succ}[1][2]!\langle \vec{n}, s[\mathtt{q}]\rangle; \mathbf{0} \mid \textstyle\prod_i R_i) \triangleright \Delta_1 \cong^s_g \\ & (\nu \vec{m}, b)(Q \mid \text{succ}[1][2]!\langle \vec{n}, s[\mathtt{q}]\rangle; \mathbf{0} \mid \textstyle\prod_i R_i) \triangleright \Delta_2\}\end{aligned}$$

where we assume succ is fresh and $b \notin \{\vec{m}\}\cup \mathtt{fn}(P)\cup \mathtt{fn}(Q)$. We will show that $\mathscr{S}$ is governed reduction-closed.

Typability. We show $\mathscr{S}$ is a typed relation. From the definition of $\mathscr{S}$, we have $\Delta'_1 \bowtie \Delta'_2$. By using typing rules [NRes], [SRes], [Conc], we obtain $(\Gamma' \vdash P \triangleright \Delta'_1, \Gamma' \vdash Q \triangleright \Delta'_2)$ is in the typed relation. Then we can set $E' = E \cup E_0 \cup \{\text{succ} : 1 \to 2 : U.\mathtt{end}\}$ where $E_0 = \{s : G_s\}$ if $s = \vec{m}$; otherwise $E_0 = \emptyset$ to make $\mathscr{S}$ a governed relation.

Reduction-closedness. Immediate by the assumption that succ is fresh and $\prod_i R_i \not\longrightarrow$.

Barb preserving. Suppose $E',\Gamma' \vdash P \triangleright \Delta'_1 \downarrow_n$ with $n \notin \vec{m}$. Then

$$E,\Gamma \vdash (\nu\vec{m},b)(P \mid \text{succ}[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i) \triangleright \Delta_1 \downarrow_n$$

by the freshness of succ. This implies $E,\Gamma \vdash (\nu\vec{m},b)(Q \mid \text{succ}[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i) \triangleright \Delta_2 \Downarrow_n$. Since $\text{succ}[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i$ does not reduce, we have $E',\Gamma' \vdash Q \triangleright \Delta'_2 \Downarrow_n$, as required.

Suppose $E',\Gamma' \vdash P \triangleright \Delta'_1 \downarrow_{s[\mathtt{p}]}$. Then we chose $T\langle N,\text{succ}',\ell\rangle$ such that

$E,\Gamma \vdash (\nu\vec{m},b)(P \mid \text{succ}[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i \mid \text{succ}[2][1]?(\vec{y},x);T\langle N,\text{succ}',\ell\rangle) \triangleright \Delta''_1 \twoheadrightarrow$
$P' \mid \prod_i R_i \mid \text{succ}'[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \triangleright \Delta'''_1$

which implies

$E,\Gamma \vdash (\nu\vec{m},b)(Q \mid \text{succ}[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i \mid \text{succ}[2][1]?(\vec{y},x);T\langle N,\text{succ}',\ell\rangle) \triangleright \Delta''_2 \twoheadrightarrow$
$Q' \mid \prod_i R_i \mid \text{succ}'[1][2]!\langle\vec{n},s[\mathtt{q}]\rangle;\mathbf{0} \triangleright \Delta'''_1$

which implies $E,\Gamma \vdash Q \triangleright \Delta'_2 \Downarrow_{s[\mathtt{p}]}$.

Contextual property. The only interesting case is if $E',\Gamma' \vdash P \triangleright \Delta'_1 \mathscr{S} E',\Gamma' \vdash Q \triangleright \Delta'_2$ then $E'',\Gamma' \vdash P \mid R \triangleright \Delta''_1 \mathscr{S} E'',\Gamma' \vdash Q \mid R \triangleright \Delta''_2$ for all R.

We compose with $O = \text{succ}[2][1]?(\vec{y},x);(R \mid \text{succ}'[1][2]!\langle\vec{z},x\rangle;\mathbf{0})$.

$$\begin{array}{lll}
(\nu\vec{m},b)(P \mid \text{succ}[1][2]!\langle v,s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i \mid O) & \cong^s_g & (\nu\vec{m},b)(Q \mid \text{succ}[1][2]!\langle v,s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i \mid O) \\
\text{implies} & & \\
(\nu\vec{m},b)(P \mid R \mid \text{succ}'[1][2]!\langle v,s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i) & \cong^s_g & (\nu\vec{m},b)(Q \mid R \mid \text{succ}'[1][2]!\langle v,s[\mathtt{q}]\rangle;\mathbf{0} \mid \prod_i R_i) \\
\text{implies} & & \\
P \mid R \mathscr{S} Q \mid R & &
\end{array}$$

□

We can now we prove the completness direction

We prove:

if $E,\Gamma \vdash P \triangleright \Delta_1 \cong^s_g E,\Gamma \vdash Q \triangleright \Delta_2$ and $E,\Gamma \vdash P \triangleright \Delta_1 \xrightarrow{\ell} E',\Gamma' \vdash P' \triangleright \Delta'_1$, then
$E,\Gamma \vdash Q \triangleright \Delta_2 \overset{\hat{\ell}}{\Longrightarrow} E',\Gamma' \vdash Q' \triangleright \Delta'_2$ such that $E',\Gamma' \vdash P' \triangleright \Delta'_1 \cong^s_g E',\Gamma' \vdash Q' \triangleright \Delta'_2$

The case $\ell = \tau$ is trivial by the definition of $\cong^s_g$.

For the case of $\ell \neq \tau$, we use Lemma B.6 and Lemma B.7. Since the governed witness and session environments do not change the proof, we omit environments. Suppose $P \cong^s_g Q$ and $P \xrightarrow{\ell} P'$. We must find a matching weak transition from Q. Choose N to contain all the free names in both P and Q and choose succ and b to be fresh for both P and Q. We denote the tester by T for convenience.

Because $\cong^s_g$ is contextual, we know $T \mid P \cong^s_g T \mid Q$. We also know

$$T \mid P \longrightarrow^* (\nu\, \mathtt{bn}(\ell),b)(P' \mid \text{succ}[1][2]!\langle\mathtt{obj}(\ell),n\rangle;\mathbf{0})$$

and therefore $T \mid Q \longrightarrow^* Q''$ for some Q'' such that

$$(\nu\, \mathtt{bn}(\ell),b)(P' \mid \text{succ}[1][2]!\langle\mathtt{obj}(\ell),n\rangle;\mathbf{0}) \cong^s_g Q''$$

and $Q'' \Downarrow \text{succ}$. Thus by Lemma B.6, we can set $Q'' \equiv (\nu\, \mathtt{bn}(\ell),b)(Q' \mid \text{succ}[1][2]!\langle\mathtt{obj}(\ell),n\rangle;\mathbf{0})$ where $Q \overset{\ell}{\Longrightarrow} Q'$. Since $\equiv$ is included in $\cong^s_g$, we have:

$$(\nu\, \mathtt{bn}(\ell),b)(P' \mid \text{succ}[1][2]!\langle\mathtt{obj}(\ell),n\rangle;\mathbf{0}) \cong^s_g (\nu\, \mathtt{bn}(\ell),b)(Q' \mid \text{succ}[1][2]!\langle\mathtt{obj}(\ell),n\rangle;\mathbf{0})$$

By Lemma B.7, we have $P' \cong^s_g Q'$, as required.

□

B.6. **Proof for Lemma 5.16.**

Proof. We prove direction if $\forall E, E, \Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ then $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s \Gamma \vdash P_2 \triangleright \Delta_2$.
If $\Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} P_1' \triangleright \Delta_1'$ then $P_1 \xrightarrow{\ell} P_1'$ and $(\Gamma, \Delta_1) \xrightarrow{\ell} (\Gamma', \Delta_1')$.

From part 3 of Lemma B.4 we choose E such that $(E, \Gamma, \Delta_1) \xrightarrow{\ell} (E', \Gamma', \Delta_1')$. Since $\forall E, E, \Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ it can now be implied that, $E, \Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} E', \Gamma \vdash P_1' \triangleright \Delta_1'$ implies $E, \Gamma \vdash P_2 \triangleright \Delta_2 \xRightarrow{\ell} E', \Gamma \vdash P_2' \triangleright \Delta_2'$ which implies $P_2 \xRightarrow{\ell} P_2'$ and $(E, \Gamma, \Delta_2) \xRightarrow{\ell} (E', \Gamma', \Delta_2')$.
From part 1 of Lemma B.4 we obtain $(\Gamma, \Delta_2) \xRightarrow{\ell} (\Gamma', \Delta_2')$ implies $\Gamma \vdash P_2 \triangleright \Delta_2 \xRightarrow{\ell} P_2' \triangleright \Delta_2'$ as required.

We prove direction if $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s \Gamma \vdash P_2 \triangleright \Delta_2$ then $\forall E, E, \Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$.
Let $E, \Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} P_1' \triangleright \Delta_1'$ then

$$P_1 \xrightarrow{\ell} P_1' \tag{B.33}$$

$$(E, \Gamma, \Delta_1) \xrightarrow{\ell} (E', \Gamma', \Delta_1') \tag{B.34}$$

If $\Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} P_1' \triangleright \Delta_1'$ then $P_1 \xrightarrow{\ell} P_1', (\Gamma, \Delta_1) \xrightarrow{\ell} (\Gamma', \Delta_1')$ and $\Gamma \vdash P_2 \triangleright \Delta_2 \xrightarrow{\ell} P_2' \triangleright \Delta_2'$.
From the last implication we obtain

$$P_2 \xRightarrow{\ell} P_2' \tag{B.35}$$

$$(\Gamma, \Delta_2) \xRightarrow{\ell} (\Gamma', \Delta_2') \tag{B.36}$$

$$\Delta_1 \rightleftharpoons \Delta_2 \tag{B.37}$$

We apply part 2 of Lemma B.4 to (B.34) and (B.37) to obtain $(E, \Gamma, \Delta_2) \xRightarrow{\ell} (E', \Gamma', \Delta_2')$. From the last result and (B.35), we obtain $E, \Gamma \vdash P_2 \triangleright \Delta_2 \xRightarrow{\ell} E', \Gamma \vdash P_2' \triangleright \Delta_2'$. □

B.7. **Proof for Theorem 5.17.**

Proof. $\Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} \Gamma' \vdash P' \triangleright \Delta'$ implies $(\Gamma, \Delta) \xrightarrow{\ell} (\Gamma', \Delta')$ by definition. This implies there exists E such that $(E, \Gamma, \Delta) \xrightarrow{\ell} (E', \Gamma', \Delta')$ by part 3 of Lemma B.4. Then by definition, if $\Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} \Gamma' \vdash P' \triangleright \Delta'$ then $\exists E$ such that $E, \Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} E', \Gamma' \vdash P' \triangleright \Delta'$. Similarly, we have if $\Gamma \vdash P \triangleright \Delta \xRightarrow{\hat{\ell}} \Gamma' \vdash P' \triangleright \Delta'$ then $\exists E$ such that $E, \Gamma \vdash P \triangleright \Delta \xRightarrow{\hat{\ell}} E', \Gamma' \vdash P' \triangleright \Delta'$.

Suppose P is simple. Then by definition, we can set $P \equiv (\nu\ \vec{a}\vec{s})(P_1 \mid P_2 \mid \cdots \mid P_n)$ where P_i contains either zero or a single session name s, which is not used in process P_j, $i \neq j$.

Suppose $\Gamma \vdash P \triangleright \Delta$. Then it is derived from $\Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i$ where Δ_i contains a zero or single session name and $\Delta \cdot \Delta_0 = \Delta_1 \cdots \Delta_n$ where Γ_0 and Δ_0 correspond to the environments of the restrictions $\vec{a}$ and $\vec{s}$, respectively. If $E_i, \Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i$ and $(\Gamma \cdot \Gamma_0, \Delta_i) \xrightarrow{\ell} (\Gamma' \cdot \Gamma_0', \Delta_i')$, then since Δ_i contains at most only a single session, the condition $E \xrightarrow{\lambda} E'$ in the premise in [Out, In, Sel, Bra, Tau] in Figure 9 is always true. Hence $(E_i, \Gamma \cdot \Gamma_0, \Delta_i) \xrightarrow{\ell} (E_i', \Gamma' \cdot \Gamma_0', \Delta_i')$. From here, we obtain for all E_i such that $E_i, \Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i$, if $\Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i \xrightarrow{\ell} \Gamma' \cdot \Gamma'_0 \vdash P'_i \triangleright \Delta'_i$, then we have $E_i, \Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i \xrightarrow{\ell} E'_i, \Gamma' \cdot \Gamma'_0 \vdash P'_i \triangleright \Delta'_i$. Similarly, for the case of $\Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i \overset{\ell}{\Longrightarrow} \Gamma' \cdot \Gamma'_0 \vdash P'_i \triangleright \Delta'_i$.

Now by applying the parallel composition, we can reason for all E such that $E \longrightarrow^* \sqcup_i E_i$, if $E, \Gamma \vdash P \triangleright \Delta$ and $\Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} \Gamma' \vdash P' \triangleright \Delta'$, there exits $\Gamma \cdot \Gamma_0 \vdash P_i \triangleright \Delta_i \xrightarrow{\ell} \Gamma' \cdot \Gamma'_0 \vdash P'_i \triangleright \Delta'_i$, which implies $E, \Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} E', \Gamma' \vdash P' \triangleright \Delta'$, with $E \longrightarrow^* \sqcup_i E_i \xrightarrow{\lambda} \sqcup_i E'_i = E'$.

By this, if P is simple and $\Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} \Gamma' \vdash P' \triangleright \Delta'$, for all E such that $E, \Gamma \vdash P \triangleright \Delta$, $E, \Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} E', \Gamma' \vdash P' \triangleright \Delta'$. Hence if there exits E such that $E, \Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} E', \Gamma' \vdash P' \triangleright \Delta'$, then for all E_0, $E_0, \Gamma \vdash P \triangleright \Delta \xrightarrow{\ell} E'_0, \Gamma' \vdash P' \triangleright \Delta'$. Similarly, for $E, \Gamma \vdash P \triangleright \Delta \overset{\ell}{\Longrightarrow} E', \Gamma' \vdash P' \triangleright \Delta'$.

Now suppose if P_1 and P_2 are simple and $\exists E$ such that $E, \Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$. From the above result and part 2 of Lemma B.4, if P_1 and P_2 are simple and $\exists E$ such that $E, \Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ then $\forall E, E, \Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$. By applying Lemma 5.16 we are done. □