knn

Kouzapas, D., and Yoshida, N. (2013) *Globally Governed Session Semantics.* In: CONCUR 2013:Concurrency Theory, 24th International Conference, 27-30 Aug 2013, Buenos Aires, Argentina.

Copyright © 2013 Springer Verlag

http://eprints.gla.ac.uk/101388/

Deposited on: 22 January 2015

Enlighten – Research publications by members of the University of Glasgow
http://eprints.gla.ac.uk

Globally Governed Session Semantics

Dimitrios Kouzapas and Nobuko Yoshida

Imperial College London

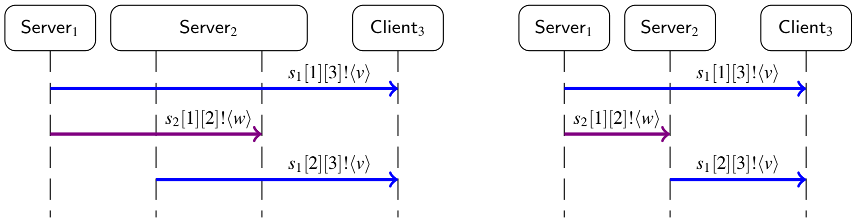
Abstract. This paper proposes a new bisimulation theory based on multiparty session types where a choreography specification governs the behaviour of session typed processes and their observer. The bisimulation is defined with the observer cooperating with the observed process in order to form complete 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. The compositionality of the governed bisimilarity is proved through the soundness and completeness with respect to the governed reduction-based congruence.

1 Introduction

Modern society increasingly depends on distributed software infrastructures such as the backend of popular Web portals, global E-science cyberinfrastructure, e-healthcare and e-governments. An application in these environments is typically organised into many components which 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 theories of multiparty session types [11] capture 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.

This paper introduces a new behavioural theory which can reason about distributed processes globally controlled 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, in order to reason about various encodings into the typed π-calculi [16, 18]. Our theory treats the mutual effects of multiple choreographic sessions which are shared among distributed participants as their common knowledges or agreements, reflecting the origin of choreographic frameworks [5]. These features make our theory distinct from any type-based bisimulations in the literature and the theory applicable to real choreographic usecase from a large-scale distributed system. Since our bisimulation is based on the regulation of conversational behaviours of distributed components by global specifications, we call our bisimulation *globally governed bisimulation*.

To illustrate the key idea, we first explain the mechanisms of multiparty session types [11]. Let us consider a simple protocol where participant 1 sends a message of type `bool` to participant 2. To develop the code for this protocol, we start by specifying the *global type* [11] as $G_1 = 1 \to 2 : \langle \texttt{bool} \rangle.\texttt{end}$ where $\to$ signifies the flow of communication and `end` denotes protocol termination. With agreement on G_1 as a specification for participant 1 and participant 2, each program can be implemented separately. Then for type-checking, G_1 is *projected* into local session types: one local session type from 1's point of view, $[2]!\langle \texttt{bool} \rangle$ (output to 2 with `bool`-type), and another from 2's point of view, $[1]?\langle \texttt{bool} \rangle$ (input from 1 with `bool`-type), against which both processes are checked to be correct.


Server₁
Server₂
Client₃
$s_1[1][3]!\langle v \rangle$
$s_2[1][2]!\langle w \rangle$
$s_1[2][3]!\langle v \rangle$
Server₁
Server₂
Client₃
$s_1[1][3]!\langle v \rangle$
$s_2[1][2]!\langle w \rangle$
$s_1[2][3]!\langle v \rangle$

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

Now we explain how our new theory can reason about an optimisation of choreography interactions (a simplified usecase (UC.R2.13 "Acquire Data From Instrument") from [1]). Consider the two global types between three participants $(1,2,3)$:

$$G_a = 1 \to 3 : \langle ser \rangle.2 \to 3 : \langle ser \rangle.\texttt{end}, \quad G_b = 1 \to 2 : \langle sig \rangle.\texttt{end}$$

and a scenario in the diagram (a) where Client3 (participant 3) uses two services, the first from Server1 (participant 1) and Server2 (participant 2), and Server1 sends an internal signal to Server2. The three parties belonging to these protocols are implemented as:

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

where session name a establishes the session corresponding to G_a. Client3 (P_3) initiates a session involving three processes as the third participant by $\overline{a}[3](x)$: Service1 (P_1) and Service2 (P_2) participate to the session $a[1](x)$ and $a[2](x)$, respectively. Similarly the session corresponding to G_b is established between Service1 and Service2.

Since from Client3, the internal signal is invisible, we optimise Server2 to a single thread to avoid an unnecessary thread creation as $R_2 = a[2](x).\overline{b}[2](y).y[1]?(z);x[3]!\langle v \rangle;\mathbf{0}$ in in the diagram (b). Note that both P_2 and R_2 are typable under G_a and G_b. Obviously, in the untyped setting, $P_1 \mid P_2$ and $P_1 \mid R_2$ are *not* bisimilar since in P_2, the output action $x[3]!\langle v \rangle$ can be observed before the input action $y[1]?(z)$ happens. However, with the global constraints given by G_a and G_b, a service provided by Server2 is only available to Client3 after Server1 sends a signal to Server2, i.e. action $x[3]!\langle v \rangle$ can only happen after action $y[1]?(z)$ in P_2. Hence $P_1 \mid P_2$ and $P_1 \mid R_2$ are not distinguishable by Client3 and the thread optimisation of R_2 is correct.

On the other hand, if we change the global type G_a as:

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

then R_2 can perform both the output to Client3 and the input from Server1 concurrently since G'_a states that Client3 can receive the message from Server2 first. Hence $P_1 \mid P_2$ and $P_1 \mid R_2$ are no longer equivalent.

The key point to make this difference possible is to observe the behaviour of processes together with the information provided by the global types. The global types can define additional knowledge about how the observer (the client in the above example) will collaborate with the observed processes so that different global types (i.e. global witnesses) can induce the different equivalences.

Contributions. This paper introduces two kinds of typed bisimulations based on multiparty session types. The first bisimulation is solely based on local (endpoint) types defined without global information, hence it resembles the standard linearity-based bisimulation. The second one is a *globally governed session bisimilarity* which uses multiparty session types as information for a global witness. We prove that each coincides with a corresponding standard contextual equivalence [10] (Theorems 3.1 and 4.1). The governed bisimulation gives more fine-grained equivalences than the locally typed bisimulation. We identify the condition when the two semantics exactly coincide (Theorem 4.2). Interestingly our theorem (Theorem 4.3) shows this difference appears only when processes are running under more than two interleaved global types. This feature makes the theory applicable to real situations where multiple choreographies are used in 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]. The appendix includes auxiliary definitions, the full proofs and a full derivation of a usecase from [1].

Acknowledgement. The work has been partially sponsored by the Ocean Observatories Initiative and EPSRC EP/K011715/1, EP/K034413/1 and EP/G015635/1.

2 Synchronous Multiparty Sessions

This section defines a synchronous version of the multiparty session types. The syntax and typing follows [4] except we eliminate queues for asynchronous communication. We chose synchrony since it allows the simplest formulations for demonstrating the essential concepts of bisimulations. The extension to asynchrony is given in [7].

Syntax Below we define the syntax of the synchronous multiparty session calculus.

P	$::= \overline{u}[\mathtt{p}](x).P$	Request	$\mid$	$\mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q$	Conditional
	$\mid\ u[\mathtt{p}](x).P$	Accept	$\mid$	$P \mid Q$	Parallel
	$\mid\ c[\mathtt{p}]!\langle e\rangle;P$	Sending	$\mid$	$\mathbf{0}$	Inaction
	$\mid\ c[\mathtt{p}]?(x);P$	Receiving	$\mid$	$(\nu\ n)P$	Hiding
	$\mid\ c[\mathtt{p}]\oplus l;P$	Selection	$\mid$	$\mu X.P$	Recursion
	$\mid\ c[\mathtt{p}]\&\{l_i : P_i\}_{i\in I}$	Branching	$\mid$	X	Variable
u	$::= x \mid a$	Identifier	c	$::= s[\mathtt{p}] \mid x$	Session
n	$::= s \mid a$	Name	v	$::= a \mid \mathtt{tt} \mid \mathtt{ff} \mid s[\mathtt{p}]$	Value
e	$::= v \mid x \mid e\ \mathtt{and}\ e' \mid e = e' \mid \ldots$	Expression			

Note that expressions includes name matching ($n = n$). We call $\mathtt{p}, \mathtt{p}', \mathtt{q}, \ldots$ (ranging over the natural numbers) the *participants*. For the primitives for session initiation,

$\overline{u}[\mathtt{p}](x).P$ initiates a new session through an identifier u (which represents a shared interaction point) with the other multiple participants, each of shape $u[\mathtt{p}](x).Q_{\mathtt{q}}$ where $1 \leq \mathtt{q} \leq \mathtt{p}-1$. The (bound) variable x is the channel used to do the communications. Session communications (communications that take place inside an established session) are performed using the next two pairs: the sending and receiving of a value and the selection and branching (where the former chooses one of the branches offered by the latter). Process $c[\mathtt{p}]!\langle e\rangle;P$ sends a value to $\mathtt{p}$; accordingly, process $c[\mathtt{p}]?(x);P$ denotes the intention of receiving a value from the participant $\mathtt{p}$. The same holds for selection/branching. Process $\mathbf{0}$ is the inactive process. Other processes are standard. We say that a process is closed if it does not contain free variables. $\mathtt{fn}(P)$/$\mathtt{bn}(P)$ and $\mathtt{fv}(P)$/$\mathtt{bv}(P)$ denote a set of free/bound names and free/bound variables, respectively. We use the standard structure rules (denoted by $\equiv$) including $\mu X.P \equiv P\{\mu X.P/X\}$.

Operational semantics Operational semantics of the calculus are defined below.

$$a[1](x).P_1 \mid \Pi_{i=\{2,..,n\}} a[i](x).P_i \longrightarrow (\nu\, s)(P_1\{s[1]/x\} \mid \Pi_{i=\{2,..,n\}} P_i\{s[i]/x\}) \quad [\text{Link}]$$

$$s[\mathtt{p}][\mathtt{q}]!\langle e\rangle;P \mid s[\mathtt{q}][\mathtt{p}]?(x);Q \longrightarrow P \mid Q\{v/x\} \quad (e \downarrow v) \qquad [\text{Comm}]$$

$$s[\mathtt{p}][\mathtt{q}] \oplus l_k;P \mid s[\mathtt{q}][\mathtt{p}]\&\{l_i : P_i\}_{i\in I} \longrightarrow P \mid P_k \quad (k \in I) \qquad [\text{Label}]$$

$$\mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q \longrightarrow P \quad (e \downarrow \mathtt{tt}) \qquad \mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q \longrightarrow Q \quad (e \downarrow \mathtt{ff}) \quad [\text{If}]$$

$$\frac{P \longrightarrow P'}{(\nu\, n)P \longrightarrow (\nu\, n)P'}\,[\text{Res}] \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}\,[\text{Par}] \qquad \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}\,[\text{Str}]$$

Rule [Link] defines synchronous session initiation. All session roles must be present to synchronously reduce each role $\mathtt{p}$ on a fresh session name $s[\mathtt{p}]$. Rule [Comm] is for sending a value to the corresponding receiving process where $e \downarrow v$ means expression e evaluates to value v. The interaction between selection and branching is defined via rule [Label]. Other rules are standard. We write $\twoheadrightarrow$ for $(\longrightarrow \cup \equiv)^*$.

Global types, ranged over by $G, G', \ldots$ describe the whole conversation scenario of a multiparty session as a type signature. Its grammar is given below.

$$\begin{array}{rrcll|rrcll}
\text{Global} & G & ::= & \mathtt{p} \to \mathtt{q} : \langle U\rangle.G' & \text{exchange} & \text{Local} & T & ::= & [\mathtt{p}]!\langle U\rangle;T & \text{send}\\
 & & \mid & \mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I} & \text{branching} & & & \mid & [\mathtt{p}]?(U);T & \text{receive}\\
 & & \mid & \mu\mathtt{t}.G & \text{recursion} & & & \mid & [\mathtt{p}] \oplus \{l_i : T_i\}_{i\in I} & \text{selection}\\
 & & \mid & \mathtt{t} & \text{variable} & & & \mid & [\mathtt{p}]\&\{l_i : T_i\}_{i\in I} & \text{branching}\\
 & & \mid & \mathtt{end} & \text{end} & & & \mid & \mu\mathtt{t}.T & \text{recursion}\\
\text{Exchange} & U & ::= & S \mid T & & & & \mid & \mathtt{t} & \text{variable}\\
\text{Sort} & S & ::= & \mathtt{bool} \mid \langle G\rangle & & & & \mid & \mathtt{end} & \text{end}
\end{array}$$

The global type $\mathtt{p} \to \mathtt{q} : \langle U\rangle.G'$ says that participant $\mathtt{p}$ sends a message of type U to the participant $\mathtt{q}$ and then interactions described in G' take place. *Exchange types* $U, U', \ldots$ consist of *sorts* types $S, S', \ldots$ for values (either base types or global types), and *local session* types $T, T', \ldots$ for channels (defined in the next paragraph). Type $\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I}$ says participant $\mathtt{p}$ sends one of the labels l_i to $\mathtt{q}$. If l_j is sent, interactions described in G_j take place. In both cases we assume $\mathtt{p} \neq \mathtt{q}$. Type $\mu\mathtt{t}.G$ is a recursive type, assuming type variables $(\mathtt{t}, \mathtt{t}', \ldots)$ are guarded in the standard way, i.e., type variables only appear under some prefix. We take an *equi-recursive* view of recursive types, not distinguishing between $\mu\mathtt{t}.G$ and its unfolding $G\{\mu\mathtt{t}.G/\mathtt{t}\}$.We assume that G in the grammar of sorts has no free type variables. Type $\mathtt{end}$ represents the termination of the session.

Local types correspond to the communication actions, representing sessions from the view-points of single participants. The *send type* $[\mathrm{p}]!\langle U\rangle;T$ expresses the sending to p of a value of type U, followed by the communications of T. The *selection type* $[\mathrm{p}]\oplus\{l_i : T_i\}_{i\in I}$ represents the transmission to p of a label l_i chosen in the set $\{l_i \mid i \in I\}$ followed by the communications described by T_i. The *receive* and *branching* are dual. Other types are the same as global types.

The relation between global and local types is formalised by the standard projection function [11]. For example, $(\mathrm{p}' \to \mathrm{q} : \langle U\rangle.G)\lceil\mathrm{p}$ is defined as: $[\mathrm{q}]!\langle U\rangle;(G\lceil\mathrm{p})$ if $\mathrm{p}=\mathrm{p}'$, $[\mathrm{p}']?(U);(G\lceil\mathrm{p})$ if $\mathrm{p}=\mathrm{q}$ and $G\lceil\mathrm{p}$ otherwise. Then the *projection set* of $s:G$ is defined as $\mathtt{proj}(s:G) = \{s[\mathrm{p}] : G\lceil\mathrm{p} \mid \mathrm{p}\in\mathtt{roles}(G)\}$ where $\mathtt{roles}(G)$ denotes the set of the roles appearing in G.

Typing system 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 standard environment which associates variables to sort types, shared names to global types and process variables to session environments; and Δ is the session environment which associates channels to session types. Formally we define: $\Gamma ::= \emptyset \mid \Gamma\cdot u : S \mid \Gamma\cdot X : \Delta$ and $\Delta ::= \emptyset \mid \Delta\cdot c : T$, assuming we can write $\Gamma\cdot u : S$ if $u \notin \mathtt{dom}(\Gamma)$. We extend this to a concatenation for typing environments as $\Delta\cdot\Delta' = \Delta\cup\Delta'$. Typing Δ is *coherent with respect to session s* (notation $\mathtt{co}(\Delta(s))$) if for all $s[\mathrm{p}] : T_{\mathrm{p}}, s[\mathrm{q}] : T_{\mathrm{q}} \in \Delta$, T_{p} and T_{q} are dual each other (it is given by exchanging ! by ? and $\oplus$ by & [9]). A typing Δ is *coherent* (notation $\mathtt{co}(\Delta)$) if it is coherent with respect to all s in its domain. The typing judgement $\Gamma \vdash P \triangleright \Delta$ is *coherent* if $\mathtt{co}(\Delta)$.

The typing rules are essentially identical to the communication typing system for programs in [4] (since we do not require queues). The rest of the paper can be read without knowing the typing system.

Type soundness Next we define the reduction semantics for local types. Since session environments represent the forthcoming communications, by reducing processes session environments can change. This can be formalised as in [4, 11] by introducing the notion of reduction of session environments, whose rules are:

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. $\{s[\mathrm{p}] : [\mathrm{q}]\oplus\{l_i : T_i\}_{i\in I}\cdot s[\mathrm{q}] : [\mathrm{p}]\&\{l_j : T'_j\}_{j\in J}\} \longrightarrow \{s[\mathrm{p}] : T_k\cdot s[\mathrm{q}] : T'_k\}$ $I\subseteq J, k\in I$.
3. $\Delta\cup\Delta' \longrightarrow \Delta\cup\Delta''$ if $\Delta' \longrightarrow \Delta''$.

We write $\twoheadrightarrow = \longrightarrow^*$. Note that $\Delta \twoheadrightarrow \Delta'$ is non-deterministic (i.e. not always confluent) by the second rule. Then the typing system satisfies the subject reduction theorem [4]: if $\Gamma \vdash P\triangleright\Delta$ is coherent and $P \twoheadrightarrow P'$ then $\Gamma \vdash P'\triangleright\Delta'$ is coherent with $\Delta \twoheadrightarrow \Delta'$.

3 Synchronous Multiparty Session Semantics

This section presents the standard typed behavioural theory for the synchronous multiparty sessions.

Labels We use the following labels $(\ell, \ell', \ldots)$:

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

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

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

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

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

$$\langle\text{Res}\rangle \frac{P \xrightarrow{\ell} P' \quad n \notin \mathtt{fn}(\ell)}{(\nu\ n)P \xrightarrow{\ell} (\nu\ n)P'} \quad \langle\text{OpenS}\rangle \frac{P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle s'[\mathtt{p}']\rangle} P'}{(\nu\ s')P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])} P'} \quad \langle\text{OpenN}\rangle \frac{P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle a\rangle} P'}{(\nu\ a)P \xrightarrow{s[\mathtt{p}][\mathtt{q}]!(a)} P'}$$

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

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

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

We omit the synmetric case of ⟨Par⟩ and conditionals.

Fig. 1. Labelled transition system for processes

A role set A is a set of multiparty session types 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 labels $s[\mathtt{p}][\mathtt{q}]!\langle v\rangle$ and $s[\mathtt{p}][\mathtt{q}]?\langle v\rangle$ for output and input of value v from $\mathtt{p}$ to $\mathtt{q}$ on session s. Bound output values can be shared channels or session roles (delegation). $s[\mathtt{p}][\mathtt{q}] \oplus l$ and $s[\mathtt{p}][\mathtt{q}]\& l$ define the selection and branching respectively. Label τ is the standard hidden transition.

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

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

Dual labels are input and output (resp. selection and branching) on the same session channel and on complementary roles. For example, in $s[\mathtt{p}][\mathtt{q}]!\langle v\rangle$ and $s[\mathtt{q}][\mathtt{p}]?\langle v\rangle$, role $\mathtt{p}$ sends to $\mathtt{q}$ and role $\mathtt{q}$ receives from $\mathtt{p}$. Another important definition for the session initiation is the notion of the complete role set. We say the role set A is *complete with respect to n* if $n = \max(A)$ and $A = \{1, 2, \ldots, n\}$. The 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. 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 1 gives the untyped labelled transition system. Rules ⟨Req⟩ and ⟨Acc⟩ define the accept and request actions for a fresh session s on role $\{\mathtt{p}\}$. Rules ⟨Send⟩ and ⟨Rcv⟩ give the send and receive respectively for value v from role $\mathtt{p}$ to role $\mathtt{q}$ in session s. Rules ⟨Sel⟩ and ⟨Bra⟩ define selecting and branching labels.

$$
\begin{array}{rrcl}
\Gamma(a) = \langle G\rangle, s \text{ fresh implies} & (\Gamma,\Delta) & \xrightarrow{a[A](s)} & (\Gamma,\Delta\cdot\{s[i]:G\lceil i\}_{i\in A}) \\
\Gamma(a) = \langle G\rangle, s \text{ fresh implies} & (\Gamma,\Delta) & \xrightarrow{\overline{a}[A](s)} & (\Gamma,\Delta\cdot\{s[i]:G\lceil i\}_{i\in A}) \\
\Gamma\vdash v:U, s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle} & (\Gamma,\Delta\cdot s[\mathtt{p}]:T) \\
s[\mathtt{q}]\notin \mathtt{dom}(\Delta), a\notin\mathtt{dom}(\Gamma) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]!(a)} & (\Gamma\cdot a:U,\Delta\cdot s[\mathtt{p}]:T) \\
\Gamma\vdash v:U, s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]?(U);T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]?\langle v\rangle} & (\Gamma,\Delta\cdot s[\mathtt{p}]:T) \\
a\notin\mathtt{dom}(\Gamma), s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]?(U);T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]?\langle a\rangle} & (\Gamma\cdot a:U,\Delta\cdot s[\mathtt{p}]:T) \\
s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s'[\mathtt{p}']:T'\cdot s[\mathtt{p}]:[\mathtt{q}]!\langle T'\rangle;T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle s'[\mathtt{p}']\rangle} & (\Gamma,\Delta\cdot s[\mathtt{p}]:T) \\
s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]!\langle T'\rangle;T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])} & (\Gamma,\Delta\cdot s[\mathtt{p}]:T\cdot\{s'[\mathtt{p}_i]:T_i\}) \\
s[\mathtt{q}], s'[\mathtt{p}']\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]?(T');T) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]?\langle s'[\mathtt{p}']\rangle} & (\Gamma,\Delta\cdot s'[\mathtt{p}']:T'\cdot s[\mathtt{p}]:T) \\
s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]\oplus\{l_i:T_i\}_{i\in I}) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]\oplus l_k} & (\Gamma,\Delta\cdot s[\mathtt{p}]:T_k) \\
s[\mathtt{q}]\notin \mathtt{dom}(\Delta) \text{ implies} & (\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]\&\{l_i:T_i\}_{i\in I}) & \xrightarrow{s[\mathtt{p}][\mathtt{q}]\& l_k} & (\Gamma,\Delta\cdot s[\mathtt{p}]:T_k) \\
\Delta\longrightarrow\Delta' \quad\text{or}\quad \Delta=\Delta' \text{ implies} & (\Gamma,\Delta) & \xrightarrow{\tau} & (\Gamma,\Delta')
\end{array}
$$

Fig. 2. Labelled Transition Relation for Environments

The last three rules are for collecting and synchronising the multiparty participants together. Rule $\langle$AccPar$\rangle$ accumulates the accept participants and records them into role set A. Rule $\langle$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 $\langle$TauS$\rangle$ checks that a role set is complete, thus a new session can be created under the τ-action (synchronisation). Other rules are standard. See Example 3.1. We write $\Longrightarrow$ for the reflexive and transitive closure of $\longrightarrow$, $\overset{\ell}{\Longrightarrow}$ for the transitions $\Longrightarrow\xrightarrow{\ell}\Longrightarrow$ and $\overset{\hat{\ell}}{\Longrightarrow}$ for $\overset{\ell}{\Longrightarrow}$ if $\ell\neq\tau$ otherwise $\Longrightarrow$.

Typed labelled transition relation We define the typed LTS on the basis of the untyped one. This is realised by introducing the definition of an environment labelled transition system, defined in Figure 2. $(\Gamma,\Delta)\xrightarrow{\ell}(\Gamma',\Delta')$ means that an environment (Γ,Δ) allows an action to take place, and the resulting environment is (Γ',Δ').

The intuition for this definition is that 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.

The first rule says that 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 ($\{s[i]:G\lceil i\}_{i\in A}$). The second rule 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, bound name output, free value input and name input. Rest of rules are free session output, bound session output, and session input as well as selection and branching rules. The bound session output records a set of session types $s'[\mathtt{p}_i]$ at opened session s'. The final rule ($\ell=\tau$) follows the reduction rules for linear session environment defined in § 2 ($\Delta=\Delta'$ 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 3.1 (Typed transition). *Typed transition* relation is defined as $\Gamma_1 \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} \Gamma_2 \vdash P_2 \triangleright \Delta_2$ if (1) $P_1 \xrightarrow{\ell} P_2$ and (2) $(\Gamma_1, \Delta_1) \xrightarrow{\ell} (\Gamma_2, \Delta_2)$ with $\Gamma_i \vdash P_i \triangleright \Delta_i$.

Synchronous multiparty behavioural theory We first define a relation $\mathscr{R}$ as *typed relation* if it relates two closed, coherent typed terms $\Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} \Gamma \vdash P_2 \triangleright \Delta_2$. We often write $\Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$.

Next we define the *barb* [2]: we write $\Gamma \vdash P \triangleright \Delta \downarrow_{s[\mathtt{p}][\mathtt{q}]}$ if $P \equiv (\nu\ \tilde{a}\tilde{s})(s[\mathtt{p}][\mathtt{q}]!\langle v\rangle; R \mid Q)$ with $s \notin \tilde{s}$ and $s[\mathtt{q}] \notin \mathtt{dom}(\Delta)$; and $\Gamma \vdash P \triangleright \Delta \downarrow_a$ if $P \equiv (\nu\ \tilde{a}\tilde{s})(\overline{a}[n](s).R \mid Q)$ with $a \notin \tilde{a}$. Then we write m for either a or $s[\mathtt{p}][\mathtt{q}]$. We define $\Gamma \vdash P \triangleright \Delta \Downarrow_m$ if there exists Q such that $P \twoheadrightarrow Q$ and $\Gamma \vdash Q \triangleright \Delta' \downarrow_m$.

We write $\Delta_1 \rightleftharpoons \Delta_2$ if there exists Δ such that $\Delta_1 \twoheadrightarrow \Delta$ and $\Delta_2 \twoheadrightarrow \Delta$. We now define the contextual congruence based on the barb and [10].

Definition 3.2 (Reduction congruence). A typed relation $\mathscr{R}$ is *reduction congruence* if it satisfies the following conditions for each $\Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$ with $\Delta_1 \rightleftharpoons \Delta_2$.

1. $\Gamma \vdash P_1 \triangleright \Delta_1 \Downarrow_m$ iff $\Gamma \vdash P_2 \triangleright \Delta_2 \Downarrow_m$
2. Whenever $\Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$ holds, $P_1 \twoheadrightarrow P_1'$ implies $P_2 \twoheadrightarrow P_2'$ such that $\Gamma \vdash P_1' \triangleright \Delta_1' \mathrel{\mathscr{R}} P_2' \triangleright \Delta_2'$ holds with $\Delta_1' \rightleftharpoons \Delta_2'$.
3. For all closed context C, such that $\Gamma \vdash C[P_1'] \triangleright \Delta_1'$ and $\Gamma \vdash C[P_2'] \triangleright \Delta_2'$ where $\Delta_1' \rightleftharpoons \Delta_2'$, $\Gamma \vdash C[P_1] \triangleright \Delta_1' \mathrel{\mathscr{R}} \Gamma \vdash C[P_2] \triangleright \Delta_2'$.

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

Definition 3.3 (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 $\Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$, it holds:

1. $\Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} \Gamma' \vdash P_1' \triangleright \Delta_1'$ implies $\Gamma \vdash P_2 \triangleright \Delta_2 \xRightarrow{\hat{\ell}} \Gamma' \vdash P_2' \triangleright \Delta_2'$ such that $\Gamma' \vdash P_1' \triangleright \Delta_1' \mathrel{\mathscr{R}} P_2' \triangleright \Delta_2'$.
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. $P \approx^s Q$. We also write $\approx$ for untyped synchronous bisimilarity which is defined by the untyped LTS in Figure 1 but without the environment LTS in Figure 2.

Theorem 3.1 (Soundness and completeness). $\cong^s = \approx^s$.

Example 3.1 (Synchronous multiparty bisimulation). We use the running example from § 1. First we explain the session initialisation from Figure 1. By $\langle\text{Acc}\rangle$ and $\langle\text{Req}\rangle$,

$$P_1 \xrightarrow{a[\{1\}](s_1)} P_1' = b[1](y).s_1[1][3]!\langle v\rangle; y[2]!\langle w\rangle; \mathbf{0}$$
$$P_2 \xrightarrow{a[\{2\}](s_1)} P_2' = \overline{b}[2](y).(y[1]?(z); \mathbf{0} \mid s_1[2][3]!\langle v\rangle; \mathbf{0}) \qquad P_3 \xrightarrow{\overline{a}[\{3\}](s_1)} P_3' = s_1[3][1]?(z); s_1[3][2]?(y); \mathbf{0}$$

with

$\Gamma \vdash P_1' \triangleright s_1[1] : [3]!\langle U\rangle;\mathtt{end}, \quad \Gamma \vdash P_2' \triangleright s_1[2] : [3]!\langle U\rangle;\mathtt{end}, \quad \Gamma \vdash P_3' \triangleright s_1[3] : [1]?(U);[2]?(U);\mathtt{end}$

By $\langle$AccPar$\rangle$, we have $P_1 \mid P_2 \xrightarrow{a[\{1,2\}](s_1)} P_1' \mid P_2'$. We have another possible initialisation: $P_1 \mid P_3 \xrightarrow{\overline{a}[\{1,3\}](s_1)} P_1' \mid P_3'$. From both of them, if we compose another process, the set $\{1,2,3\}$ becomes complete so that by synchronisation $\langle$TauS$\rangle$, $\Gamma \vdash P_1 \mid P_2 \mid P_3 \triangleright \emptyset \xrightarrow{\tau} \Gamma \vdash (\nu\ s_1)(P_1' \mid P_2' \mid P_3') \triangleright \emptyset$. Further we have:

$$\Gamma \vdash P_1' \mid P_2' \triangleright \Delta_0 \xrightarrow{\tau}$$
$$\Gamma \vdash (\nu\ s_2)(s_1[1][3]!\langle v\rangle;s_2[1][2]!\langle w\rangle;\mathbf{0} \mid s_2[2][1]?(z);\mathbf{0} \mid s_1[2][3]!\langle v\rangle;\mathbf{0}) = Q_1 \triangleright \Delta_0$$

with $\Delta_0 = s_1[1] : [3]!\langle U\rangle;\mathtt{end} \cdot s_1[2] : [3]!\langle U\rangle;\mathtt{end}$. Then

$$\Gamma \vdash Q_1 \mid P_3' \triangleright \Delta_0 \cdot s_1[3] : [1]?(U);[2]?(U);\mathtt{end} \approx^s \mathbf{0} \triangleright s_1[1] : \mathtt{end} \cdot s_1[2] : \mathtt{end} \cdot s_1[3] : \mathtt{end}$$

since $(\Gamma,\Delta) \not\xrightarrow{\ell}$ for any $\ell \neq \tau$ with $\Delta = \Delta_0 \cdot s_1[3] : [1]?(U);[2]?(U);\mathtt{end}$ (by the condition of Line 3 in Figure 2). However by the untyped LTS, $Q_1 \mid P_3' \not\approx \mathbf{0}$ since $Q_1 \mid P_3' \xrightarrow{s_1[1][3]!\langle v\rangle}$.

4 Globally Governed Behavioural Theory

We introduce the semantics for globally governed behavioural theory. In the previous section, the local typing (Δ) constrains the untyped LTS to give rise to a local typed LTS. In a multiparty distributed environment, communications follow 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.

Global environments and configurations We define a *global environment* $(E, E', ...)$ as a mapping from session names to global types.

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

The projection definition is extended to include E as $\mathtt{proj}(E) = \bigcup_{s:G \in E} \mathtt{proj}(s : G)$.

We define a labelled reduction relation over global environments which corresponds to $\Delta \longrightarrow \Delta'$ defined in § 2. We use the labels $\lambda \in \{s : \mathtt{p} \to \mathtt{q} : U, s : \mathtt{p} \to \mathtt{q} : l\}$ to annotate reductions over global environments. We define $\mathtt{out}(\lambda)$ and $\mathtt{inp}(\lambda)$ as $\mathtt{out}(s : \mathtt{p} \to \mathtt{q} : U) = \mathtt{out}(s : \mathtt{p} \to \mathtt{q} : l) = \mathtt{p}$ and as $\mathtt{inp}(s : \mathtt{p} \to \mathtt{q} : U) = \mathtt{inp}(s : \mathtt{p} \to \mathtt{q} : l) = \mathtt{q}$ and $\mathtt{p} \in \ell$ if $\mathtt{p} \in \mathtt{out}(\ell) \cup \mathtt{inp}(\ell)$. We often omit the label λ by writing $\longrightarrow$ for $\xrightarrow{\lambda}$ and $\longrightarrow^*$ for $(\xrightarrow{\lambda})^*$. The first rule is the axiom for the input and output interaction between two parties; the second rule is for the choice; the third and forth rules formulate the case that the action λ can be performed under $\mathtt{p} \to \mathtt{q}$ if $\mathtt{p}$ and $\mathtt{q}$ are not related to the participants in λ; and the fifth rule is a congruent rule.

$$\{s : \mathtt{p} \to \mathtt{q} : \langle U\rangle.G\} \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} \{s : G\} \qquad \{s : \mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I}\} \xrightarrow{s:\mathtt{p}\to\mathtt{q}:l_k} \{s : G_k\}$$

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

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

As a simple example of the above LTS, consider $s:\mathtt{p}\to\mathtt{q}:\langle U_1\rangle.\mathtt{p}'\to\mathtt{q}':\{l_1:\mathtt{end},l_2:\mathtt{p}'\to\mathtt{q}':\langle U_2\rangle.\mathtt{end}\}$. Since $\mathtt{p},\mathtt{q},\mathtt{p}',\mathtt{q}'$ are pairwise distinct, we can apply the second and third rules to obtain: $s:\mathtt{p}\to\mathtt{q}:\langle U_1\rangle.\mathtt{p}'\to\mathtt{q}':\{l_1:\mathtt{end},l_2:\mathtt{p}'\to\mathtt{q}':\langle U_2\rangle.\mathtt{end}\} \xrightarrow{s:\mathtt{p}'\to\mathtt{q}':l_1} s:\mathtt{p}\to\mathtt{q}:\langle U_1\rangle.\mathtt{end}$

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

Definition 4.1 (Governance judgement). *Let $\Gamma\vdash P\triangleright\Delta$ be coherent. We write $E,\Gamma\vdash P\triangleright\Delta$ if $\exists E'\cdot E\longrightarrow^* E'$ 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).

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

Definition 4.2 (Environment configuration). *We write (E,Γ,Δ) if $\exists E'\cdot E\longrightarrow^* E'$ and $\Delta\subseteq\mathtt{proj}(E')$.*

Figure 3 defines a LTS over environment configurations that refines the LTS over environments (i.e $(\Gamma,\Delta)\xrightarrow{\ell}(\Gamma',\Delta')$) in § 3.

Each rule requires a corresponding environment transition (Figure 2 in § 3) and a corresponding labelled global environment transition in order to control a transition following the global protocol. [Acc] is the rule for accepting a session initialisation so that it creates a new mapping $s:G$ which matches Γ in a governed environment E. [Req] is the rule for requesting a new session and it is dual to [Acc].

The next seven rules are the transition relations on session channels and we assume the condition $\mathtt{proj}(E_1)\supseteq\Delta$ 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 selection and branching, which is similar to [Out] and [In]. In [Tau] rule, we refined the reduction relation on Δ in § 2 as:

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

$$[\text{Acc}]\,\frac{\Gamma \vdash a:\langle G\rangle \quad (\Gamma,\Delta_1) \xrightarrow{a[A](s)} (\Gamma,\Delta_2)}{(E,\Gamma,\Delta_1) \xrightarrow{a[A](s)} (E\cdot s:G,\Gamma,\Delta_2)} \qquad [\text{Req}]\,\frac{\Gamma \vdash a:\langle G\rangle \quad (\Gamma,\Delta_1) \xrightarrow{\overline{a}[A](s)} (\Gamma,\Delta_2)}{(E,\Gamma,\Delta_1) \xrightarrow{\overline{a}[A](s)} (E\cdot s:G,\Gamma,\Delta_2)}$$

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

$$[\text{ResN}]\,\frac{\begin{array}{c}(\Gamma,\Delta_1)\xrightarrow{s[\mathtt{p}][\mathtt{q}]!(a)}(\Gamma\cdot a:\langle G\rangle,\Delta_2)\\ E_1\xrightarrow{s:\mathtt{q}\to\mathtt{p}:\langle G\rangle}E_2\end{array}}{(E_1,\Gamma,\Delta_1)\xrightarrow{s[\mathtt{p}][\mathtt{q}]!(a)}(E_2,\Gamma\cdot a:\langle G\rangle,\Delta_2)} \qquad [\text{ResS}]\,\frac{\begin{array}{c}(\Gamma,\Delta_1)\xrightarrow{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])}(\Gamma,\Delta_2\cdot\{s'[\mathtt{p}_i]:T_i\})\\ E_1\xrightarrow{s:\mathtt{q}\to\mathtt{p}:T}E_2 \quad \cdot\forall i.G\lceil \mathtt{p}_i = T_i\end{array}}{(E_1,\Gamma,\Delta_1)\xrightarrow{s[\mathtt{p}][\mathtt{q}]!(s'[\mathtt{p}'])}(E_2\cdot s':G,\Gamma,\Delta_2\cdot\{s'[\mathtt{p}_i]:T_i\})}$$

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

$$[\text{Tau}]\,\frac{(\Delta_1=\Delta_2,\ E_1=E_2)\vee(\Delta_1\xrightarrow{\lambda}\Delta_2,\ E_1\xrightarrow{\lambda}E_2)}{(E_1,\Gamma,\Delta_1)\xrightarrow{\tau}(E_2,\Gamma,\Delta_2)} \qquad [\text{Inv}]\,\frac{E_1\longrightarrow^* E_1'\quad (E_1',\Gamma_1,\Delta_1)\xrightarrow{\ell}(E_2,\Gamma_2,\Delta_2)}{(E_1,\Gamma_1,\Delta_1)\xrightarrow{\ell}(E_2,\Gamma_2,\Delta_2)}$$

Fig. 3. The LTS for the environment configuations

[Inv] is the key rule: the 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 4.1 (LTS for environment configuration). Let $E = s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.\mathtt{p}\to\mathtt{q}:\langle U\rangle.G$, $\Gamma = v:U$ and $\Delta = s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T_\mathtt{p}$ with $G\lceil\mathtt{p}=T_\mathtt{p}$, $G\lceil\mathtt{q}=T_\mathtt{q}$ and $\mathtt{roles}(G) = \{\mathtt{p},\mathtt{q}\}$. Then (E,Γ,Δ) is an environment configuration since if $E\longrightarrow E'$ then $\mathtt{proj}(E')\supset\Delta$ because $E\xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G$, $\mathtt{proj}(s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G) = s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T_\mathtt{p}\cdot s[\mathtt{q}]:[\mathtt{p}]?(U);T_\mathtt{q}$ and $\mathtt{proj}(s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G)\supset\Delta$. Then we can apply [Out] to $s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G \xrightarrow{s:\mathtt{p}\to\mathtt{q}:U} s:G$ and $(\Gamma,s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T_\mathtt{p})\xrightarrow{s[\mathtt{p}][\mathtt{q}]!\langle v\rangle}(\Gamma,s[\mathtt{p}]:T_\mathtt{p})$ to 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 this and $E\longrightarrow s:\mathtt{p}\to\mathtt{q}:\langle U\rangle.G$, using [Inv], we can 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 congruency To define the reduction-closed congruency, 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.

$$\begin{array}{rl}
(E,\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]!\langle U\rangle;T)\downarrow_{s[\mathtt{p}][\mathtt{q}]} & \text{if } s[\mathtt{q}]\notin\mathtt{dom}(\Delta),\exists E'\cdot E\longrightarrow^* E'\xrightarrow{s:\mathtt{p}\to\mathtt{q}:U},\Delta\subseteq\mathtt{proj}(E')\\
(E,\Gamma,\Delta\cdot s[\mathtt{p}]:[\mathtt{q}]\oplus\{l_i:T_i\}_{i\in I})\downarrow_{s[\mathtt{p}][\mathtt{q}]} & \text{if } s[\mathtt{q}]\notin\mathtt{dom}(\Delta),\exists E'\cdot E\longrightarrow^* E'\xrightarrow{s:\mathtt{p}\to\mathtt{q}:l_k},k\in I,\Delta\subseteq\mathtt{proj}(E'),\\
(E,\Gamma,\Delta)\downarrow_a & \text{if } a\in\mathtt{dom}(\Gamma)
\end{array}$$

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

Let us write $T_1\sqsubseteq T_2$ if the syntax tree of T_2 includes T_1. For example, $[\mathtt{q}]?(U');T\sqsubseteq[\mathtt{p}]!\langle U\rangle;[\mathtt{q}]?(U');T$. Then we define: $E_1\sqcup E_2=\{E_i(s)\ \mid\ E_j(s)\sqsubseteq E_i(s), i,j\in\{1,2\}, i\neq$

$j\} \cup E_1 \setminus \mathtt{dom}(E_2) \cup E_2 \setminus \mathtt{dom}(E_1)$. As an example of $E_1 \sqcup E_2$, let us define:

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

Then $E_1 \sqcup E_2 = \mathtt{p} \to \mathtt{q} : \langle U_1 \rangle.\mathtt{p}' \to \mathtt{q}' : \langle U_2 \rangle.\mathtt{p} \to \mathtt{q} : \langle U_3 \rangle.\mathtt{end} \cdot s_2 : \mathtt{p}' \to \mathtt{q}' : \langle W_1 \rangle.\mathtt{p} \to \mathtt{q} : \langle W_2 \rangle.\mathtt{end}$. The behavioural relation w.r.t. a global whiteness is defined below.

Definition 4.3 (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 $E_1 \sqcup E_2, \Gamma \vdash P \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$ if $E_1 \sqcup E_2$ is defined.

Proposition 4.1 (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; and (2) Given E, a set $\{E' \mid E \longrightarrow^* E'\}$ is finite.*

Definition 4.4 (Global configuration transition). We write $E_1, \Gamma \vdash P_1 \triangleright \Delta_1 \xrightarrow{\ell} E_2, \Gamma' \vdash P_2 \triangleright \Delta_2$ if $E_1, \Gamma \vdash P_1 \triangleright \Delta_1$, $P_1 \xrightarrow{\ell} P_2$ and $(E_1, \Gamma, \Delta_1) \xrightarrow{\ell} (E_2, \Gamma', \Delta_2)$.

Proposition 4.2. *(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; and (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 $\mathtt{co}(\Delta')$.*

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

Definition 4.5 (Governed reduction congruence). A configuration relation $\mathscr{R}$ is *governed reduction congruence* if $E, \Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$ 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 and only if $P_2 \twoheadrightarrow P_2'$ and $E, \Gamma \vdash P_1' \triangleright \Delta_1' \mathrel{\mathscr{R}} P_2' \triangleright \Delta_2'$
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 $E, \Gamma \vdash C[P_1] \triangleright \Delta_1' \mathrel{\mathscr{R}} C[P_2] \triangleright \Delta_2'$.

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

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

Definition 4.6 (Globally governed bisimulation). A configuration relation $\mathscr{R}$ is a *globally governed weak bisimulation* (or governed bisimulation) if whenever $E, \Gamma \vdash P_1 \triangleright \Delta_1 \mathrel{\mathscr{R}} P_2 \triangleright \Delta_2$, 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 $E_1' \sqcup E_2', \Gamma' \vdash P_1' \triangleright \Delta_1' \mathrel{\mathscr{R}} P_2' \triangleright \Delta_2'$.
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. $P \approx_g^s Q$.

Theorem 4.1 (Sound and completeness). $\approx_g^s = \cong_g^s$.

The relationship between $\approx^s$ and $\approx^s_g$ is given as follows.

Theorem 4.2. *If for all E, $E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ then $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$. Also if $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$, then for all E, $E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$.*

To justify the above theorem, consider the following processes:

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

then we have $P_1 \approx^s P_2$. By the above theorem, 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:

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

Note that all E that are instances up-to weakening are 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 [11]. A simple process contains only a single session so that it satisfies the progress property as proved in [11]. 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 [11]). Since there is no interleaving of sessions in simple processes, the difference between $\approx^s$ and $\approx^s_g$ disappears.

Theorem 4.3 (Coincidence). *Assume P_1 and P_2 are simple. If $\exists E \cdot E,\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s_g P_2 \triangleright \Delta_2$ then $\Gamma \vdash P_1 \triangleright \Delta_1 \approx^s P_2 \triangleright \Delta_2$.*

To justify the above theorem, consider: $P_1 = s[1][2]?(x);s[1][3]!\langle x\rangle;\mathbf{0} \mid s[2][1]!\langle v\rangle;\mathbf{0}$ and $P_2 = s[1][3]!\langle v\rangle;\mathbf{0}$. It holds that for $E = s : 2 \to 1 : \langle U\rangle.1 \to 3 : \langle U\rangle.\mathtt{end}$ then $E \vdash P_1 \approx^s_g P_2$. We can easily reason $P_1 \approx^s P_2$.

Example 4.2 (Governed bisimulation). Recall the example from § 1 and Example 3.1. Q_1 is the process corresponding to Example 3.1, while Q_2 has a parallel thread instead of the sequential composition (this corresponds to $P_1 \mid R_2$ in § 1).

$$\begin{aligned} Q_1 &= s_1[1][3]!\langle v\rangle;s_2[1][2]!\langle w\rangle;\mathbf{0} \mid s_2[2][1]?(x);\mathbf{0} \mid s_1[2][3]!\langle v\rangle;\mathbf{0} \\ Q_2 &= s_1[1][3]!\langle v\rangle;s_2[1][2]!\langle w\rangle;\mathbf{0} \mid s_2[2][1]?(x);s_1[2][3]!\langle v\rangle;\mathbf{0} \end{aligned}$$

Assume: $\Gamma = v : S \cdot w : S$
$\Delta = s_1[1] : [3]!\langle S\rangle;\mathtt{end}\cdot s_1[2] : [3]!\langle S\rangle;\mathtt{end}\cdot s_2[1] : [2]!\langle S\rangle;\mathtt{end}\cdot s_2[2] : [1]?(S);\mathtt{end}$

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

$$\begin{aligned} E_1 &= s_1 : 1 \to 3 : \langle S\rangle.2 \to 3 : \langle S\rangle.\mathtt{end}\cdot s_2 : 1 \to 2 : \langle S\rangle.\mathtt{end} \\ E_2 &= s_1 : 2 \to 3 : \langle S\rangle.1 \to 3 : \langle S\rangle.\mathtt{end}\cdot s_2 : 1 \to 2 : \langle S\rangle.\mathtt{end} \end{aligned}$$

Then the projection of E_1 and E_2 are given as:

$$\begin{aligned} \mathtt{proj}(E_1) = {} & s_1[1] : [3]!\langle S\rangle;\mathtt{end}\cdot s_1[2] : [3]!\langle S\rangle;\mathtt{end}\cdot s_1[3] : [1]?(S);[2]?(S);\mathtt{end} \\ & s_2[1] : [2]!\langle S\rangle;\mathtt{end}\cdot s_2[2] : [1]?(S);\mathtt{end} \\ \mathtt{proj}(E_2) = {} & s_1[1] : [3]!\langle S\rangle;\mathtt{end}\cdot s_1[2] : [3]!\langle S\rangle;\mathtt{end}\cdot s_1[3] : [2]?(S);[1]?(S);\mathtt{end}\cdot \\ & s_2[1] : [2]!\langle S\rangle;\mathtt{end}\cdot s_2[2] : [1]?(S);\mathtt{end} \end{aligned}$$

with $\Delta \subset \mathtt{proj}(E_1)$ and $\Delta \subset \mathtt{proj}(E_2)$. The reader should note that the difference between E_1 and E_2 is the type of the participant 3 at s_1.

By definition, we can write: $E_i, \Gamma \vdash Q_1 \triangleright \Delta$ and $E_i, \Gamma \vdash Q_2 \triangleright \Delta$ for $i = 1, 2$. Both processes are well-formed global configurations under both witnesses. Now we can observe $\Gamma \vdash Q_1 \triangleright \Delta \xrightarrow{s[2][3]!\langle v\rangle} \Gamma \vdash Q_1' \triangleright \Delta'$ but $\Gamma \vdash Q_2 \triangleright \Delta \overset{s[2][3]!\langle v\rangle}{\not\longrightarrow}$. Hence $\Gamma \vdash Q_1 \triangleright \Delta \not\approx^s Q_2 \triangleright \Delta$. By the same argument, we have: $E_2, \Gamma \vdash Q_1 \triangleright \Delta \not\approx^s_g Q_2 \triangleright \Delta$. On the other hand, since E_1 *forces* to wait for $s[2][3]!\langle v\rangle$, $E_1, \Gamma \vdash Q_1 \triangleright \Delta \overset{s[2][3]!\langle v\rangle}{\not\longrightarrow}$. Hence Q_1 and Q_2 are bisimilar, i.e. $E_1, \Gamma \vdash Q_1 \triangleright \Delta \approx^s_g Q_2 \triangleright \Delta$. This concludes the optimisation is correct.

5 Related and Future Work

As a typed foundation for structured communications programming, session types [9, 17] have been studied over the last decade for a wide range of process calculi and programming languages. 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 the existing ones only focus on binary (two-party) sessions. Our work [14] develops an *asynchronous binary* session typed behavioural theory with event operations. An LTS is defined on session type process judgements and ensures session typed properties, such as linearity in the presence of asynchronous queues. The work [15] proves the proof conversions induced by Linear Logic interpretation coincide with an observational equivalence over a strict subset of the binary synchronous session processes. 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 [13], 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 [6] gives a fully abstract encoding of a *binary* synchronous session typed calculus into a linearly typed π-calculus [3]. 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.

The constructions of our work are hinted by [8] which studies typed behavioural semantics for the π-calculus with IO-subtyping where a LTS for pairs of typing environments and processes is used for defining typed testing equivalences and barbed congruence. On the other hand, in [8], 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. Among them, a recent paper [12] 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 3 and Definition 4.4). Global types can give a guidance how to coordinate parallel threads giving explicit protocols, hence it is applicable to a semantic-preserving optimisation (cf. Example 4.2 and [7]). 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 for the governed bisimulations for some subset of multiparty session processes.

References

1. Ocean Observatories Initiative (OOI). `http://www.oceanobservatories.org/`.
2. R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi-calculus. *TCS*, 195(2):291–324, 1998.
3. M. Berger, K. Honda, and N. Yoshida. Sequentiality and the π-calculus. In *Proc. TLCA'01*, volume 2044 of *LNCS*, pages 29–45, 2001.
4. L. Bettini et al. Global progress in dynamically interleaved multiparty sessions. In *CONCUR*, volume 5201 of *LNCS*, pages 418–433. Springer, 2008.
5. W3C Web Services Choreography. `http://www.w3.org/2002/ws/chor/`.
6. R. Demangeon and K. Honda. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR*, volume 6901 of *LNCS*, pages 280–296. Springer, 2011.
7. Technical report of this paper. Department of Computing, Imperial College, DTR 2013/4.
8. 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.
9. 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.
10. K. Honda and N. Yoshida. On reduction-based process semantics. *TCS*, 151(2):437–486, 1995.
11. K. Honda, N. Yoshida, and M. Carbone. Multiparty Asynchronous Session Types. In *POPL'08*, pages 273–284. ACM, 2008.
12. V. Koutavas and M. Hennessy. A testing theory for a higher-order cryptographic language. In *ESOP*, volume 6602 of *LNCS*, pages 358–377, 2011.
13. D. Kouzapas. *A study of bisimulation theory for session types*. PhD thesis, Department of Computing, Imperial College London, May 2013.
14. 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.
15. 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.
16. B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *MSCS*, 6(5):409–454, 1996.
17. 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.
18. N. Yoshida. Graph types for monadic mobile processes. In *FSTTCS*, volume 1180 of *LNCS*, pages 371–386. Springer, 1996.