Open Multiparty Sessions

Franco Barbanera*
Dipartimento di Matematica e Informatica
Università di Catania, Catania Italy
barba@dmi.unict.it

Mariangiola Dezani-Ciancaglini†
Dipartimento di Informatica,
Università di Torino, Torino Italy
dezani@di.unito.it

Multiparty sessions are systems of concurrent processes, which allow several participants to communicate by sending and receiving messages. Their overall behaviour can be described by means of global types. Typable multiparty session enjoy lock-freedom. We look at multiparty sessions as *open* systems by a suitable definition of *connection* transforming *compatible* processes into *gateways* (forwarders). A relation resembling the standard subtyping relation for session types is used to formalise compatibility. We show that the session obtained by connection can be typed by manipulating the global types of the starting sessions. This allows us to prove that lock-freedom is preserved by connection.

1 Introduction

Distributed systems are seldom developed as independent entities and, either directly in their design phase or even after their deployment, they should be considered as open entities ready for interaction with an environment. In general, it is fairly natural to expect to connect open systems as if they were composable modules, and in doing that we should rely on "safe" methodologies and techniques, guaranteeing the composition not to "break" any relevant property of the single systems.

In [1] a methodology has been proposed for the connection of open systems, consisting in replacing *any* two participants - if their behaviours are "compatible" - by two forwarders, dubbed *gateways*, enabling the systems to interact. The behaviour of any participant can be looked at as an *interface* since, without loss of generality, the notion of interface is interpreted not as the description of the interactions "offered" by a system but, dually, as those "required" by a possible environment (usually another system).

Inspired by [22], the aim of the present paper is to look for a choreography formalism enabling to "lift" the connection-by-gateways by means of a proper function definable on protocol descriptions. The function should yield the protocol of the system obtained by connecting the systems described by the arguments of the function itself. Connected systems would hence enjoy all the good communication properties guaranteed by the formalism itself, which - with no ad-hoc extension of the syntax - could be seen as a choreography formalism for open systems.

We took into account the choreography model of MultiParty Session Types (MPST) [17, 18]. Of course not all of the MPST formalisms are suitable for our aim. For instance, in the formalisn of [9] the requirements imposed by its type system are too strong for the gateway processes to be typed, so preventing the function we are looking for to be definable.

The MPST formalism that we introduce in the present paper (inspired by [27]) proved to be a right candidate. The simplicity of the calculus allows to get rid of channels and local types. Moreover,

*Partially supported by the project "Piano Triennale Ricerca" DMI-Università di Catania.
†Partially supported by Ateneo/Compagnia di San Paolo 2016/2018 project "MnemoComputing - Components for Processing In Memory".

To appear in EPTCS.

its abstract point of view for what concerns global and local behaviours (looked at as infinite regular trees) also enables to focus on the relevant aspects of the investigation without the hindering syntactic descriptions of recursion. In particular, with respect to [27], we relax the conditions imposed on global types in order to be projectable, so ensuring projectability of "connected" global types. (a property that does not hold in the formalism of [27]). In our formalism, typable systems are guaranteed to be lock-free [20]. The systems obtained by connecting typable systems are lock-free too. The main tool is a function from the global types of the original systems to the global type of the system obtained by connection.
In the present setting it is also possible to investigate in a clean way the notion of *interface compatibility*: we show that the compatibility relation used in [1] can be relaxed to a relation closely connected to the observational preorder of [27], in turn corresponding to the subtyping relation for session types of [14, 11].

Outline The first three sections introduce our calculus of multiparty sessions, together with their global types, and prove the properties of well-typed sessions. In the following two sections we define the compatibility relations and the gateway connections for sessions and global types, respectively. Our main result, i.e. the typability and hence the lock-freedom of the session obtained by gateway connection, is Theorem 6.7. Sections 7 and 8 conclude discussing related and future works, respectively.

2 Processes and Multiparty Sessions

We use the following base sets and notation: *messages*, ranged over by $\ell, \ell', \ldots$; *session participants*, ranged over by $\mathsf{p}, \mathsf{q}, \ldots$; *processes*, ranged over by $P, Q, \ldots$; *multiparty sessions*, ranged over by $\mathscr{M}, \mathscr{M}', \ldots$; *integers*, ranged over by $n, m, i, j, , \ldots$.

Processes implement the behaviours of single participants. The input process $\mathsf{p}?\{\ell_i.P_i \mid 1{\leq}i{\leq}n\}$ waits for one of the messages ℓ_i from participant p; the output process $\mathsf{p}!\{\ell_i.P_i \mid 1{\leq}i{\leq}n\}$ chooses one message ℓ_i and sends it to participant p. We use Λ as shorthand for $\{\ell_i.P_i \mid 1{\leq}i{\leq}n\}$. We define the multiset of messages in Λ as $msg(\{\ell_i.P_i \mid 1{\leq}i{\leq}n\}) = \{\ell_i \mid 1 \leq i \leq n\}$. After sending or receiving the message ℓ_i the process reduces to P_i ($1 \leq i \leq n$). The set Λ in $\mathsf{p}?\Lambda$ acts as an *external choice*, while the same set in $\mathsf{p}!\Lambda$ acts as an *internal choice*. In a full-fledged calculus, messages would carry values, namely they would be of the form $\ell(\mathsf{v})$. Here for simplicity we consider only pure messages. This agrees with the focus of session calculi, which is on process interactions that do not depend on actual transmitted values.
For the sake of abstraction, we do not take into account any explicit syntax for recursion, but rather consider processes as, possibly infinite, regular trees.

It is handy to first define pre-processes, since the processes must satisfy conditions which can be easily given using the tree representation of pre-processes.

Definition 2.1 (Processes) *(i) We say that P is a* pre-process *and Λ is a* pre-choice of messages *if they are generated by the grammar:*

$$P \ ::=^{coinductive} \ \mathbf{0} \quad | \quad \mathsf{p}?\Lambda \quad | \quad \mathsf{p}!\Lambda \qquad\qquad \Lambda \ ::= \{\ell_i.P_i \mid 1{\leq}i{\leq}n\}$$

and all messages in $msg(\Lambda)$ are pairwise distinct.

(ii) The tree representation *of a pre-process is a directed rooted tree, where: (a) each internal node is labelled by* $\mathsf{p}?$ *or* $\mathsf{p}!$ *and has as many children as the number of messages, (b) the edge from* $\mathsf{p}?$ *or* $\mathsf{p}!$ *to the child P_i is labelled by ℓ_i and (c) the leaves of the tree (if any) are labelled by* $\mathbf{0}$.

(iii) We say that a pre-process P is a process *if the tree representation of P is regular (namely, it has finitely many distinct sub-trees). We say that a pre-choice of messages Λ is a* choice of messages *if all the pre-processes in Λ are processes.*

We identify processes with their tree representations and we shall sometimes refer to the trees as the processes themselves. The regularity condition implies that we only consider processes admitting a finite description. This is equivalent to writing processes with μ-notation and an equality which allows for an infinite number of unfoldings. This is also called the *equirecursive approach*, since it views processes as the unique solutions of (guarded) recursive equations [26, Section 20.2]. The existence and uniqueness of a solution follow from known results (see [10] and [5, Theorem 7.5.34]). It is natural to use *coinduction* as the main logical tool, as we do in most of the proofs. In particular, we adopt the coinduction style advocated in [21] which, without any loss of formal rigour, promotes readability and conciseness.

We define the set $\mathtt{ptp}(P)$ of participants of process P by: $\mathtt{ptp}(\mathbf{0}) = \emptyset$ and

$$\mathtt{ptp}(\mathtt{p}?\{\ell_i.P_i \mid 1\leq i\leq n\}) = \mathtt{ptp}(\mathtt{p}!\{\ell_i.P_i \mid 1\leq i\leq n\}) = \{\mathtt{p}\}\cup\mathtt{ptp}(P_1)\cup\ldots\cup\mathtt{ptp}(P_n)$$

The regularity of processes assures that the set of participants is finite.

We shall write $\ell.P\uplus\Lambda$ for $\{\ell.P\}\cup\Lambda$ if $\ell\notin msg(\Lambda)$ and $\Lambda_1\uplus\Lambda_2$ for $\Lambda_1\cup\Lambda_2$ if $msg(\Lambda_1)\cap msg(\Lambda_2)=\emptyset$. We shall also omit curly brackets in choices with only one branch and trailing $\mathbf{0}$ processes.

A *multiparty session* is the parallel composition of pairs participants/processes.

Definition 2.2 (Multiparty Sessions) *A* multiparty session $\mathscr{M}$ *is defined by the following grammar:*

$$\mathscr{M} \quad ::=^{inductive} \quad \mathtt{p}\triangleright P \quad \mid \quad \mathscr{M}\mid\mathscr{M}$$

and it should satisfy the following conditions:
(a) In $\mathtt{p}_1\triangleright P_1 \mid \ldots \mid \mathtt{p}_n\triangleright P_n$ *all the* $\mathtt{p}_i$*'s* $(1\leq i\leq n)$ *are distinct;*
(b) In $\mathtt{p}\triangleright P$ *we require* $\mathtt{p}\notin\mathtt{ptp}(P)$ *(we do not allow self-communication).*

We shall use $\prod_{1\leq i\leq n}\mathtt{p}_i\triangleright P_i$ as shorthand for $\mathtt{p}_1\triangleright P_1 \mid \ldots \mid \mathtt{p}_n\triangleright P_n$.
We define $\mathtt{pts}(\mathtt{p}\triangleright P)=\{\mathtt{p}\}$ and $\mathtt{pts}(\mathscr{M}\mid\mathscr{M}')=\mathtt{pts}(\mathscr{M})\cup\mathtt{pts}(\mathscr{M}')$.

Operational Semantics The structural congruence $\equiv$ between two multiparty sessions establishes that parallel composition is commutative, associative and has neutral elements $\mathtt{p}\triangleright\mathbf{0}$ for any fresh $\mathtt{p}$.

The reduction for multiparty sessions allows participants to choose and communicate messages.

Definition 2.3 (LTS for Multiparty Sessions) *The* labelled transition system (LTS) for multiparty sessions *is the closure under structural congruence of the reduction specified by the unique rule:*

$$[\textsc{comm}]\quad \frac{msg(\Lambda)\subseteq msg(\Lambda')}{\mathtt{p}\triangleright\mathtt{q}!(\ell.P\uplus\Lambda) \;\mid\; \mathtt{q}\triangleright\mathtt{p}?(\ell.Q\uplus\Lambda')\mid\mathscr{M} \xrightarrow{\mathtt{p}\ell\mathtt{q}} \mathtt{p}\triangleright P \;\mid\; \mathtt{q}\triangleright Q\mid\mathscr{M}}$$

Rule [COMM] makes the communication possible: participant $\mathtt{p}$ sends message ℓ to participant $\mathtt{q}$. This rule is non-deterministic in the choice of messages. The condition $msg(\Lambda)\subseteq msg(\Lambda')$ assures that the sender can freely choose the message, since the receiver must offer all sender messages and possibly more. This allows us to distinguish in the operational semantics between internal and external choices. We use $\mathscr{M}\xrightarrow{\lambda}\mathscr{M}'$ as shorthand for $\mathscr{M}\xrightarrow{\mathtt{p}\ell\mathtt{q}}\mathscr{M}'$. We sometimes omit the label writing $\longrightarrow$. As usual, $\longrightarrow^*$ denotes the reflexive and transitive closure of $\longrightarrow$.

Example 2.4 Let us consider a system (inspired by a similar one in [1]) with participants $\mathtt{p}$, $\mathtt{q}$, and $\mathtt{h}$ interacting according the following protocol. Participant $\mathtt{p}$ keeps on sending text messages to $\mathtt{q}$, which has to deliver them to $\mathtt{h}$. After a message has been sent by $\mathtt{p}$, the next one can be sent only if the previous has been received by $\mathtt{h}$ and its propriety of language ascertained, i.e if it does not contain, say, rude or offensive words. Participant $\mathtt{h}$ acknowledges to $\mathtt{q}$ the propriety of language of a received text by means of the message *ack*. In such a case $\mathtt{q}$ sends to $\mathtt{p}$ an *ok* message so that $\mathtt{p}$ can proceed by sending a further message. More precisely:

1. p sends a text message to q in order to be delivered to h, which accepts only texts possessing a good propriety of language;
2. then h either
   (a) sends an *ack* to q certifying the reception of the text and its propriety. In this case q sends back to p an *ok* message and the protocol goes back to 1., so that p can proceed by sending a further text message;
   (b) sends a *nack* message to inform q that the text has not the required propriety of language. In such a case q produces *transf* (a semantically invariant reformulation of the text), sends it back to h and the protocol goes to 2. again. Before doing that, q informs p (through the *notyet* message) that the text has not been accepted yet and a reformulation has been requested;
   (c) sends a *stop* message to inform q that no more text will be accepted. In such a case q informs of that also p.

A multiparty session implementing this protocol is: $\mathscr{M} = \mathsf{p} \triangleright P \mid \mathsf{q} \triangleright Q \mid \mathsf{h} \triangleright H$ where

$P = \mathsf{q}!\mathit{text}.P_1$ $\quad$ $P_1 = \mathsf{q}?\{\mathit{ok}.P, \mathit{notyet}.P_1, \mathit{stop}\}$

$Q = \mathsf{p}?\mathit{text}.\mathsf{h}!\mathit{text}.Q_1$ $\quad$ $Q_1 = \mathsf{h}?\{\mathit{ack}.\mathsf{p}!\mathit{ok}.Q, \mathit{nack}.\mathsf{p}!\mathit{notyet}.\mathsf{h}!\mathit{transf}.Q_1, \mathit{stop}.\mathsf{p}!\mathit{stop}\}$

$H = \mathsf{q}?\mathit{text}.H_1$ $\quad$ $H_1 = \mathsf{q}!\{\mathit{ack}.H, \mathit{nack}.\mathsf{q}?\mathit{transf}.H_1, \mathit{stop}\}$

We end this section by defining the property of lock-freedom for multiparty session as in [20]. Lock-freedom ensures both progress and no starvation (under fairness assumption). I.e. it guarantees the absence of deadlock and that all participants willing to communicate can do it. Recall that $\mathsf{p} \triangleright \mathbf{0}$ is the neutral element of parallel composition.

Definition 2.5 (Lock-Freedom) *We say that a multiparty session* $\mathscr{M}$ is a lock-free session *if*

(a) $\mathscr{M} \longrightarrow^* \mathscr{M}'$ *implies either* $\mathscr{M}' \equiv \mathsf{p} \triangleright \mathbf{0}$ *or* $\mathscr{M}' \longrightarrow \mathscr{M}''$*, and*

(b) $\mathscr{M} \longrightarrow^* \mathsf{p} \triangleright P \mid \mathscr{M}'$ *and* $P \neq \mathbf{0}$ *imply* $\mathsf{p} \triangleright P \mid \mathscr{M}' \longrightarrow^* \mathscr{M}'' \xrightarrow{\lambda}$ *and* p *occurs in* λ.

3 Global Types and Typing System

The behaviour of multiparty sessions can be disciplined by means of types, as usual. Global types describe the whole conversation scenarios of multiparty sessions. As in [27] we directly assign global types to multiparty sessions without the usual detour around session types and subtyping [17, 18].

The type $\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}$ formalises a protocol where participant p must send to q a message ℓ_i for some $1 \leq i \leq n$ and then, depending on which ℓ_i was chosen by p, the protocol continues as G_i. We use Γ as shorthand for $\{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}$ and define the multiset $\mathit{msg}(\{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}) = \{\ell_i \mid 1 \leq i \leq n\}$. As for processes, we define first pre-global types and then global types.

Definition 3.1 (Global Types) *(i) We say that* G *is a* pre-global type *and* Γ *is a* pre-choice of communications *if they are generated by the grammar:*

$$\mathsf{G} ::=^{\mathit{coinductive}} \mathtt{end} \quad \| \quad \mathsf{p} \to \mathsf{q} : \Gamma \qquad\qquad \Gamma := \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}$$

and all messages in $\mathit{msg}(\Gamma)$ *are pairwise distinct.*

(ii) The tree representation *of a pre-global type is built as follows: (a) each internal node is labelled by* $\mathsf{p} \to \mathsf{q}$ *and has as many children as the number of messages, (b) the edge from* $\mathsf{p} \to \mathsf{q}$ *to the child* G_i *is labelled by* ℓ_i *and (c) the leaves of the tree (if any) are labelled by* end.

(iii) We say that a pre-global type G *is a* global type *if the tree representation of* G *is* regular. *We say that a pre-choice of communications* Γ *is a* choice of communications *if all the pre-global types in* Γ *are global types.*

We identify pre-global types and global types with their tree representations and we shall sometimes refer to the tree representation as the global types themselves. As for processes, the regularity condition implies that we only consider global types admitting a finite representation.

The set $\mathtt{ptg}(\mathsf{G})$ of participants of global type G is defined similarly to that of processes. The regularity of global types assures that the set of participants is finite. We shall write $\ell.\mathsf{G} \uplus \Gamma$ for $\{\ell.\mathsf{G}\} \cup \Gamma$ if $\ell \notin msg(\Gamma)$ and $\Gamma_1 \uplus \Gamma_2$ for $\Gamma_1 \cup \Gamma_2$ if $msg(\Gamma_1) \cap msg(\Gamma_2) = \emptyset$. We shall omit curly brackets in choices with only one branch and trailing `end`s.

Since all messages in communication choices are pairwise distinct, the set of paths in the trees representing global types are determined by the labels of nodes and edges found on the way, omitting the leaf label `end`. Let ρ range over paths of global types. Formally the set of paths of a global type can be defined as a set of sequences as follows (ε is the empty sequence):

$$paths(\mathtt{end}) = \{\varepsilon\} \qquad paths(\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}) = \bigcup_{1 \leq i \leq n} \{\mathsf{p} \to \mathsf{q}\, \ell_i\, \rho \mid \rho \in paths(\mathsf{G}_i)\}$$

Note that every infinite path of a global type has infinitely many occurrences of $\to$.

Example 3.2 A global type representing the protocol of Example 2.4 is:

$$\begin{array}{lcl} \mathsf{G} & = & \mathsf{p} \to \mathsf{q} : \mathit{text}.\mathsf{q} \to \mathsf{h} : \mathit{text}.\mathsf{G}_1 \\ \mathsf{G}_1 & = & \mathsf{h} \to \mathsf{q} : \{\mathit{ack} : \mathsf{q} \to \mathsf{p} : \mathit{ok}.\mathsf{G}, \\ & & \qquad \mathit{nack} : \mathsf{q} \to \mathsf{p} : \mathit{notyet}.\mathsf{q} \to \mathsf{h} : \mathit{transf}.\mathsf{G}_1, \\ & & \qquad \mathit{stop} : \mathsf{q} \to \mathsf{p} : \mathit{stop}\} \end{array}$$

In order to assure lock-freedom by typing we require that the first occurrences of participants in global types are at a bounded depth in all paths starting from the root. This is formalised by the following definition of *weight*.

Definition 3.3 (Weight) *Let* $weight(\rho_1\,(\mathsf{q} \to \mathsf{r})\,\ell\,\rho_2, \mathsf{p}) = length(\rho_1)$ *if* $\mathsf{p} \notin \rho_1$ *and* $\mathsf{p} \in \{\mathsf{q}, \mathsf{r}\}$*, then*

$$weight(\mathsf{G}, \mathsf{p}) = \begin{cases} max\{weight(\rho, \mathsf{p}) \mid \rho \in paths(\mathsf{G})\} & \text{if } \mathsf{p} \in \mathtt{ptg}(\mathsf{G}), \\ 0 & \text{otherwise} \end{cases}$$

Example 3.4 If G is as in Example 3.2, then $weight(\mathsf{G}, \mathsf{p}) = weight(\mathsf{G}, \mathsf{q}) = 0$, and $weight(\mathsf{G}, \mathsf{h}) = 1$. If $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \{\ell_1.\mathsf{r} \to \mathsf{p} : \ell_3, \ell_2.\mathsf{G}'\}$, then $weight(\mathsf{G}', \mathsf{r}) = \infty$.

The standard projection of global types onto participants produces session types and session types are assigned to processes by a type system [17, 18]. The present simplified shape of messages allows us to define a projection of global types onto participants producing processes instead of local types.

The projection of a global type onto a participant returns, if any, the process that the participant should run to follow the protocol specified by the global type. If the global type begins by establishing a communication from p to q, then the projection onto p should send one message to q, and the projection onto q should receive one message from p. The projection onto a third participant r skips the initial communication, that does not involve her. This implies that the behaviour of r must be independent of the branch chosen by p, that is the projections on r of all the branches must be the same. However, in case of projections yielding input processes from the same sender, we can allow the process of r to combine all these processes, proviso the messages are all distinct.

Definition 3.5 (Projection) *Given a global type* G *and a participant* p*, we define the partial function* $\upharpoonright_{\mathsf{p}}$ *coinductively as follows:*

$$
\begin{array}{rcl}
\mathsf{G}\upharpoonright_{\mathsf{p}} & = & \mathbf{0} \quad \textit{if } \mathsf{p} \notin \mathtt{ptg}(\mathsf{G}) \\
(\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid 1{\leq}i{\leq}n\}) \upharpoonright_{\mathsf{p}} & = & \mathsf{q}!\{\ell_i.\mathsf{G}_i\upharpoonright_{\mathsf{p}} \mid 1{\leq}i{\leq}n\} \\
(\mathsf{q} \to \mathsf{p} : \{\ell_i.\mathsf{G}_i \mid 1{\leq}i{\leq}n\}) \upharpoonright_{\mathsf{p}} & = & \mathsf{q}?\{\ell_i.\mathsf{G}_i\upharpoonright_{\mathsf{p}} \mid 1{\leq}i{\leq}n\} \\
(\mathsf{q} \to \mathsf{r} : \{\ell_i.\mathsf{G}_i \mid 1{\leq}i{\leq}n\}) \upharpoonright_{\mathsf{p}} & = & \begin{cases} \mathsf{G}_1\upharpoonright_{\mathsf{p}} & \textit{if } \mathsf{p} \notin \{\mathsf{q},\mathsf{r}\} \textit{ and } \mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{G}_j\upharpoonright_{\mathsf{p}} \ (1 \leq i,j \leq n) \\ \mathsf{s}?(\Lambda_1 \uplus \ldots \uplus \Lambda_n) & \textit{if } \mathsf{p} \notin \{\mathsf{q},\mathsf{r}\},\ \mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{s}?\Lambda_i \ (1 \leq i \leq n) \textit{ and} \\ & \quad msg(\Lambda_i) \cap msg(\Lambda_j) = \emptyset \textit{ for } 1 \leq i \neq j \leq n \end{cases}
\end{array}
$$

We say that $\mathsf{G}\upharpoonright_{\mathsf{p}}$ *is the* projection *of* G *onto* p *if* $\mathsf{G}\upharpoonright_{\mathsf{p}}$ *is defined. We say that* G *is* projectable *if* $\mathsf{G}\upharpoonright_{\mathsf{p}}$ *is defined for all participants* p.

This projection is the coinductive version of the projection given in [12, 15], where processes are replaced by local types.

As mentioned above, if p is not involved in the first communication of G, and G starts with a choice between distinct messages, then in all branches the process of participant p must either behave in the same way or be a different input, so that p can understand which branch was chosen.

Example 3.6 The global type G of Example 3.2 is projectable, and by projecting it we obtain $\mathsf{G}\upharpoonright_{\mathsf{p}} = P$, $\mathsf{G}\upharpoonright_{\mathsf{q}} = Q$, $\mathsf{G}\upharpoonright_{\mathsf{h}} = H$, where P, Q, and H are as defined in Example 2.4. Also the global type G' of Example 3.4 is projectable, $\mathsf{G}'\upharpoonright_{\mathsf{p}} = \mathsf{q}!\{\ell_1.\mathsf{r}?\ell_3, \ell_2.\mathsf{G}'\upharpoonright_{\mathsf{p}}\}$, $\mathsf{G}'\upharpoonright_{\mathsf{q}} = \mathsf{p}?\{\ell_1,\ell_2.\mathsf{G}'\upharpoonright_{\mathsf{q}}\}$, $\mathsf{G}'\upharpoonright_{\mathsf{r}} = \mathsf{p}!\ell_3$. Notice that G' has two branches, the projection of the first branch onto r is $\mathsf{p}!\ell_3$, the projection of the second branch onto r is just the projection of G' onto r, so $\mathsf{p}!\ell_3$ is the (coinductive) projection of G' onto r.

Definition 3.7 (Well-formed Global Types) *A global type* G *is well formed if weight*(G,p) *is finite and* $\mathsf{G}\upharpoonright_{\mathsf{p}}$ *is defined for all* $\mathsf{p} \in \mathtt{ptg}(\mathsf{G})$.

The global type G of Example 3.2 is well formed, while the global type G' of Example 3.4 is not well formed. In the following **we only consider well-formed global types**.

To type multiparty sessions we use the preorder $\leqslant$ on processes below, inspired by the subtyping of [7].

Definition 3.8 (Structural Preorder) *We define the* structural preorder on processes, *$P \leqslant Q$, by coinduction:*

$$
[\text{SUB-}\mathbf{0}]\quad \mathbf{0} \leqslant \mathbf{0}
\qquad
\frac{[\text{SUB-IN}]\quad P_i \leqslant Q_i \quad \forall 1 \leq i \leq n}{\mathsf{p}?(\{\ell_i.P_i \mid 1{\leq}i{\leq}n\} \uplus \Lambda) \leqslant \mathsf{p}?\{\ell_i.Q_i \mid 1{\leq}i{\leq}n\}}
\qquad
\frac{[\text{SUB-OUT}]\quad P_i \leqslant Q_i \quad \forall 1 \leq i \leq n}{\mathsf{p}!\{\ell_i.P_i \mid 1{\leq}i{\leq}n\} \leqslant \mathsf{p}!\{\ell_i.Q_i \mid 1{\leq}i{\leq}n\}}
$$

The double line in rules indicates that the rules are interpreted *coinductively*. Rule [SUB-IN] allows bigger processes to offer fewer inputs, while Rule [SUB-OUT] requires the output messages to be the same. The regularity condition on processes is crucial to guarantee the termination of algorithms for checking structural preorder. As it will be further discussed in Remark 6.8, the current proof fails if we type processes used as gateways by means of the preorder $\leqslant^+$ obtained by substituting rule

$$
\frac{[\text{SUB-OUT}^+]\quad P_i \leqslant Q_i \quad \forall 1 \leq i \leq n}{\mathsf{p}!\{\ell_i.P_i \mid 1{\leq}i{\leq}n\} \leqslant \mathsf{p}!(\{\ell_i.Q_i \mid 1{\leq}i{\leq}n\} \uplus \Lambda)}
$$

for rule [SUB-OUT].

The typing judgments associate global types to sessions: they are of the shape $\vdash \mathscr{M} : \mathsf{G}$.

Definition 3.9 (Typing system) *The only typing rule is:*

$$
\frac{[\text{T-SESS}]\quad \forall i \in I \quad P_i \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}_i} \quad \mathtt{ptg}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \prod_{i \in I} \mathsf{p}_i \triangleright P_i : \mathsf{G}}
$$

This rule requires that the processes in parallel can play as participants of a whole communication protocol or they are the terminated process, i.e. they are smaller or equal (according to the structural preorder) to the projections of a unique global type. The condition $\mathtt{ptg}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}$ allows to type also sessions containing $\mathsf{p} \triangleright \mathbf{0}$, a property needed to assure invariance of types under structural congruence. Notice that this typing rule allows to type multiparty session only with global types which can be projected on all their participants. A session $\mathscr{M}$ is *well typed* if there exists G such that $\vdash \mathscr{M} : \mathsf{G}$.

4 Properties of Well-Typed Sessions

We start with the standard lemmas of inversion and canonical form, easily following from Rule [T-SESS].

Lemma 4.1 (Inversion Lemma) *If* $\vdash \prod_{i\in I} \mathsf{p}_i \triangleright P_i : \mathsf{G}$, *then* $P_i \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}_i}$ *for all* $i \in I$ *and* $\mathtt{ptg}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}$.

Lemma 4.2 (Canonical Form Lemma)
If $\vdash \mathscr{M} : \mathsf{G}$ *and* $\mathtt{ptg}(\mathsf{G}) = \{\mathsf{p}_i \mid i \in I\}$, *then* $\mathscr{M} \equiv \prod_{i\in I} \mathsf{p}_i \triangleright P_i$ *and* $P_i \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}_i}$ *for all* $i \in I$.

To formalise the properties of Subject Reduction and Session Fidelity [17, 18], we use the standard LTS for global types given below. Rule [ICOMM] is justified by the fact that in a projectable global type $\mathsf{r} \to \mathsf{s} : \Gamma$, the behaviours of a participant p different from r and s and starting with an output are the same in all branches, and hence they are independent from the choice of r, and may be executed before it.

Definition 4.3 (LTS for Global Types) *The* labelled transition system (LTS) for global types *is specified by the rules:*

$$[\text{ECOMM}] \quad \mathsf{p} \to \mathsf{q} : (\ell.\mathsf{G} \uplus \Gamma) \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G} \qquad [\text{ICOMM}] \quad \frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}_i' \quad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \quad \text{for all } i\ (1 \le i \le n)}{\mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1{\le}i{\le}n\} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i' \mid 1{\le}i{\le}n\}}$$

The following lemma relates projections and reductions of global types.

Lemma 4.4 (Key Lemma) *(i) If* $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ *and* $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$, *then* $msg(\Lambda) = msg(\Lambda')$. *Moreover* $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}^\ell$ *and* $\ell.\mathsf{G}^\ell\upharpoonright_{\mathsf{p}} \in \Lambda$ *and* $\ell.\mathsf{G}^\ell\upharpoonright_{\mathsf{q}} \in \Lambda'$ *for all* $\ell \in msg(\Lambda)$.

(ii) If $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$, *then* $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ *and* $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$ *and* $\ell \in msg(\Lambda) = msg(\Lambda')$.

Proof. (i). The proof is by induction on $n = weight(\mathsf{G},\mathsf{p})$. If $n = 0$, then we have $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ and $msg(\Gamma) = msg(\Lambda) = msg(\Lambda')$ and $\ell.\mathsf{G}^\ell\upharpoonright_{\mathsf{p}} \in \Lambda$ and $\ell.\mathsf{G}^\ell\upharpoonright_{\mathsf{q}} \in \Lambda'$ by definition of projection. If $n > 0$, then $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1{\le}i{\le}n\}$ and $\{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset$ and $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ and $\mathsf{G}_i\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$ for all i, $1 \le i \le n$, by definition of projection. By the induction hypothesis, $msg(\Lambda) = msg(\Lambda')$. Moreover, again by the induction hypothesis, $\mathsf{G}_i \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}_i^\ell$ and $\ell.\mathsf{G}_i^\ell\upharpoonright_{\mathsf{p}} \in \Lambda$ and $\ell.\mathsf{G}_i^\ell\upharpoonright_{\mathsf{q}} \in \Lambda'$ for all i, $1 \le i \le n$. We get $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}^\ell$ using rule [ICOMM], where $\mathsf{G}^\ell = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i^\ell \mid 1{\le}i{\le}n\}$. The definition of projection implies $\mathsf{G}^\ell\upharpoonright_{\mathsf{p}} = \mathsf{G}_1^\ell\upharpoonright_{\mathsf{p}}$ and $\mathsf{G}^\ell\upharpoonright_{\mathsf{q}} = \mathsf{G}_1^\ell\upharpoonright_{\mathsf{q}}$.
(ii). The proof is by induction on $weight(\mathsf{G},\mathsf{p})$ and by cases on the reduction rules.
The case of rule [ECOMM] is easy. For rule [ICOMM], by the induction hypothesis, $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda_i$ and $\mathsf{G}_i\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda_i'$ and $\ell \in msg(\Lambda_i) = msg(\Lambda_i')$ for all i, $1 \le i \le n$. By definition of projection $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{G}_j\upharpoonright_{\mathsf{p}}$ and $\mathsf{G}_i\upharpoonright_{\mathsf{q}} = \mathsf{G}_j\upharpoonright_{\mathsf{q}}$ for $1 \le i, j \le n$. Again by definition of projection $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{G}_1\upharpoonright_{\mathsf{p}}$ and $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{G}_1\upharpoonright_{\mathsf{q}}$. □

Subject Reduction says that the transitions of well-typed sessions are mimicked by those of global types.

Theorem 4.5 (Subject Reduction) *If* $\vdash \mathscr{M} : \mathsf{G}$ *and* $\mathscr{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathscr{M}'$, *then* $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$ *and* $\vdash \mathscr{M}' : \mathsf{G}'$.

Proof. If $\mathscr{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathscr{M}'$, then

$$\begin{array}{rcl} \mathscr{M} & \equiv & \mathsf{p}\triangleright\mathsf{q}!(\ell.P \uplus \Lambda) \mid \mathsf{q}\triangleright\mathsf{p}?(\ell.Q \uplus \Lambda') \mid \prod\limits_{1\le j\le m} \mathsf{r}_j \triangleright R_j \\ \mathscr{M}' & \equiv & \mathsf{p}\triangleright P \mid \mathsf{q}\triangleright Q \mid \prod\limits_{1\le j\le m} \mathsf{r}_j \triangleright R_j \end{array}$$

Since $\vdash \mathscr{M} : \mathsf{G}$, by Lemma 4.1 we have that $\mathsf{q}!(\ell.P \uplus \Lambda) \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$, $\mathsf{p}?(\ell.Q \uplus \Lambda') \leqslant \mathsf{G}\upharpoonright_{\mathsf{q}}$ and $R_j \leqslant \mathsf{G}\upharpoonright_{\mathsf{r}_j}$ $(1 \le j \le m)$. By definition of $\leqslant$, from $\mathsf{q}!(\ell.P \uplus \Lambda) \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$ we get $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!(\ell.P_0 \uplus \Lambda_0)$ and $P \leqslant P_0$. Similarly from $\mathsf{p}?(\ell.Q \uplus \Lambda') \leqslant \mathsf{G}\upharpoonright_{\mathsf{q}}$ we get $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?(\ell.Q_0 \uplus \Lambda'_0)$ and $Q \leqslant Q_0$. Lemma 4.4(i) implies $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$ and $\mathsf{G}'\upharpoonright_{\mathsf{p}} = P_0$ and $\mathsf{G}'\upharpoonright_{\mathsf{q}} = Q_0$. We show $\mathsf{G}\upharpoonright_{\mathsf{r}_j} \leqslant \mathsf{G}'\upharpoonright_{\mathsf{r}_j}$ for each j, $1 \le j \le m$ by induction on $\mathit{weight}(\mathsf{G}, \mathsf{r}_j)$ and by cases on the reduction rules. For rule [ECOMM] we get $\mathsf{G} = \mathsf{p} \to \mathsf{q} : (\ell.\mathsf{G}' \uplus \Gamma)$. By Definition 3.5 either $\mathsf{G}\upharpoonright_{\mathsf{r}_j} = \mathsf{G}'\upharpoonright_{\mathsf{r}_j}$ or $\mathsf{G}\upharpoonright_{\mathsf{r}_j} \leqslant \mathsf{G}'\upharpoonright_{\mathsf{r}_j}$. For rule [ICOMM] $\mathsf{G}_i\upharpoonright_{\mathsf{r}_j} \leqslant \mathsf{G}'_i\upharpoonright_{\mathsf{r}_j}$ for $1 \le i \le n$ by the induction hypothesis. In both cases $\mathsf{G}\upharpoonright_{\mathsf{r}_j} \leqslant \mathsf{G}'\upharpoonright_{\mathsf{r}_j}$. We conclude $\vdash \mathscr{M}' : \mathsf{G}'$. □

Session fidelity assures that the communications in a session typed by a global type are done as prescribed by the global type.

Theorem 4.6 (Session Fidelity) *Let* $\vdash \mathscr{M} : \mathsf{G}$.

(i) If $\mathscr{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathscr{M}'$, *then* $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$ *and* $\vdash \mathscr{M}' : \mathsf{G}'$.

(ii) If $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$, *then* $\mathscr{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathscr{M}'$ *and* $\vdash \mathscr{M}' : \mathsf{G}'$.

Proof. (i). It is the Subject Reduction Theorem.
(ii). By Lemma 4.4(ii), $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ and $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$ and $\ell \in \mathit{msg}(\Lambda) = \mathit{msg}(\Lambda')$. By Lemma 4.4(i), $\ell.\mathsf{G}'\upharpoonright_{\mathsf{p}} \in \Lambda$ and $\ell.\mathsf{G}'\upharpoonright_{\mathsf{q}} \in \Lambda'$. By Lemma 4.2, $\mathscr{M} \equiv \mathsf{p}\triangleright P \mid \mathsf{q}\triangleright Q \mid \mathscr{M}_0$ and $P \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$ and $Q \leqslant \mathsf{G}\upharpoonright_{\mathsf{q}}$. By definition of $\leqslant$ we get $P = \mathsf{q}!(\ell.P' \uplus \Lambda_1)$ with $\mathit{msg}(\Lambda) = \{\ell\} \cup \mathit{msg}(\Lambda_1)$ and $P' \leqslant \mathsf{G}'\upharpoonright_{\mathsf{p}}$, and $Q = \mathsf{p}?(\ell.Q' \uplus \Lambda_2)$ with $\mathit{msg}(\Lambda_2) \cup \{\ell\} \supseteq \mathit{msg}(\Lambda')$ and $Q' \leqslant \mathsf{G}'\upharpoonright_{\mathsf{q}}$. Hence $\mathscr{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{p}\triangleright P' \mid \mathsf{q}\triangleright Q' \mid \mathscr{M}_0 = \mathscr{M}'$ and $\vdash \mathscr{M}' : \mathsf{G}'$. □

Let $\vdash^+$ be the typing system obtained by using $\leqslant^+$ (as defined on page 6) in rule [T-SESS]. Session Fidelity for $\vdash^+$ is weaker than for $\vdash$. If $\mathscr{M} = \mathsf{p}\triangleright\mathsf{q}!.\ell_1 \mid \mathsf{q}\triangleright\mathsf{p}?\{\ell_1, \ell_2\}$ and $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_1, \ell_2\}$, then $\vdash^+ \mathscr{M} : \mathsf{G}$ and $\mathsf{G} \xrightarrow{\mathsf{p}\ell_i\mathsf{q}} \mathtt{end}$ with $i = 1, 2$, but the only reduction of $\mathscr{M}$ is $\mathscr{M} \xrightarrow{\mathsf{p}\ell_1\mathsf{q}} \mathsf{p}\triangleright\mathbf{0}$. Notice that $\mathsf{q}!.\ell_1 \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$ but $\mathsf{q}!.\ell_1 \not\leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$ and $\mathsf{p}?\{\ell_1, \ell_2\} = \mathsf{G}\upharpoonright_{\mathsf{q}}$.
Clearly $\vdash \mathscr{M} : \mathsf{G}$ implies $\vdash^+ \mathscr{M} : \mathsf{G}$, and a weakening of the vice versa is shown below.

Theorem 4.7 *If* $\vdash^+ \mathscr{M} : \mathsf{G}$, *then* $\vdash \mathscr{M} : \mathsf{G}'$ *for some* G'.

Proof. The proof is by coinduction on G. Let $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$. Then, by Lemmas 4.2 and 4.1 (which easily extend to $\vdash^+$), Definition 3.5 and the definition of $\leqslant^+$, $\mathscr{M} \equiv \mathsf{p}\triangleright\mathsf{q}!\Lambda \mid \mathsf{q}\triangleright\mathsf{p}?\Lambda' \mid \mathscr{M}'$ and $\mathsf{q}!\Lambda \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$ and $\mathsf{p}?\Lambda' \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{q}}$. Again by the definition of $\leqslant^+$, $\mathit{msg}(\Lambda) \subseteq \mathit{msg}(\Gamma) \subseteq \mathit{msg}(\Lambda')$. Let $\Lambda = \{\ell_i.P_i \mid {}_{1\le i\le n}\}$, $\Gamma = \{\ell_i.\mathsf{G}_i \mid {}_{1\le i\le n}\} \uplus \Gamma'$ and $\Lambda' = \{\ell_i.Q_i \mid {}_{1\le i\le n}\} \uplus \Lambda''$. Then $\mathscr{M} \xrightarrow{\mathsf{p}\ell_i\mathsf{q}} \mathsf{p}\triangleright P_i \mid \mathsf{q}\triangleright Q_i \mid \mathscr{M}'$ and $\mathsf{G} \xrightarrow{\mathsf{p}\ell_i\mathsf{q}} \mathsf{G}_i$ for all i, $1 \le i \le n$. Since the proof of Theorem 4.5 easily adapts to $\vdash^+$, we get $\vdash^+ \mathsf{p}\triangleright P_i \mid \mathsf{q}\triangleright Q_i \mid \mathscr{M}' : \mathsf{G}_i$ for all i, $1 \le i \le n$. By coinduction there are G'_i such that $\vdash \mathsf{p}\triangleright P_i \mid \mathsf{q}\triangleright Q_i \mid \mathscr{M}' : \mathsf{G}'_i$ for all i, $1 \le i \le n$. We can choose $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}'_i \mid {}_{1\le i\le n}\}$. □

We end this section by showing that the type system $\vdash$ assures lock-freedom. By Subject Reduction it is enough to prove that well-typed sessions are deadlock-free and no participant waits forever. The former follows from Session Fidelity, while the latter follows from the following lemma that says that reducing by rule [ECOMM] the weights of the not involved participants strictly decrease.

Lemma 4.8 *If* $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$ *by rule* [ECOMM] *and* $\mathsf{r} \notin \{\mathsf{p},\mathsf{q}\}$ *and* $\mathsf{r} \in \mathtt{ptg}(\mathsf{G})$*, then* $\mathit{weight}(\mathsf{G},\mathsf{r}) > \mathit{weight}(\mathsf{G}',\mathsf{r})$.

Proof. Rule [ECOMM] implies $\mathsf{G} = \mathsf{p} \to \mathsf{q} : (\ell.\mathsf{G}' \uplus \Gamma)$. If ρ is a path of G', then $(\mathsf{p} \to \mathsf{q})\,\ell\,\rho$ is a path in G. This gives $\mathit{weight}(\mathsf{G},\mathsf{r}) > \mathit{weight}(\mathsf{G}',\mathsf{r})$. □

Multiparty session typability guarantees lock-freedom.

Theorem 4.9 (Lock-Freedom) *If session* $\mathscr{M}$ *is well typed, then* $\mathscr{M}$ *is lock-free.*

Proof. Let G be a type for $\mathscr{M}$. If $\mathscr{M} \not\equiv \mathsf{p} \triangleright \mathbf{0}$, then $\mathsf{G} \neq \mathtt{end}$. Let $\mathsf{G} = \mathsf{q} \to \mathsf{r} : \Gamma$. By definition of reduction $\mathsf{G} \xrightarrow{\mathsf{q}\ell\mathsf{r}} \mathsf{G}'$ for some ℓ, and this implies $\mathscr{M} \xrightarrow{\mathsf{q}\ell\mathsf{r}} \mathscr{M}'$ by Theorem 4.6(ii). This shows condition (a) of Definition 2.5. The proof of condition (b) of Definition 2.5 is by induction on $n = \mathit{weight}(\mathsf{G},\mathsf{p})$. If $n = 0$ then either $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ or $\mathsf{G} = \mathsf{q} \to \mathsf{p} : \Gamma$ and $\mathsf{G} \xrightarrow{\lambda} \mathsf{G}'$ with p in λ by rule [ECOMM]. If $n > 0$ then $\mathsf{G} = \mathsf{q} \to \mathsf{r} : \Gamma$ with $\mathsf{p} \notin \{\mathsf{q},\mathsf{r}\}$ and $\mathsf{G} \xrightarrow{\mathsf{q}\ell\mathsf{r}} \mathsf{G}'$ for all $\ell \in \mathit{msg}(\Gamma)$ by rule [ECOMM]. By Lemma 4.8 $\mathit{weight}(\mathsf{G},\mathsf{p}) > \mathit{weight}(\mathsf{G}',\mathsf{p})$ and induction applies. □

It is easy to check that $\vdash \mathscr{M} : \mathsf{G}$, where $\mathscr{M}$ and G are the multiparty session and the global type of Examples 2.4 and 3.2, respectively. By the above result, $\mathscr{M}$ of Example 2.4 is hence provably lock-free.

5 Connection of Multiparty-Sessions via Gateways

Given two multiparty sessions, they can be *connected via gateways* when they possess two *compatible* participants, i.e. participants that offer communications which can be paired and can hence be transformed into forwarders, that we dub "gateways". We start by discussing the relation of compatibility between processes by elaborating on Examples 2.4 and 3.2.

If we decide to look at the participant h as an interface, the messages sent by her have to be considered as those actually provided by an external environment; and the received messages as messages expected by such an environment. In a sense, this means that, if we abstract from participants' names in the process H, we get a description of an interface (in the more usual sense) of an external system, rather than an interface of our system.

In order to better grasp the notion of compatibility hinted at above, let us dub "ERS" the operation abtracting from the participants' names inside processes. So, in our example we would get

$$\mathrm{ERS}(H) = \circ?\mathit{text}.\mathrm{ERS}(H_1) \qquad \mathrm{ERS}(H_1) = \circ!\{\mathit{ack}.\mathrm{ERS}(H), \mathit{nack}.\circ?\mathit{transf}.\mathrm{ERS}(H_1), \mathit{stop}\}$$

Let us now take into account another system that could work as the environment of the system having the G of Example 3.2 as global type. Let assume such a system to be formed by participants k, r and s interacting according the following protocol:

Participant k sends text messages to r and s in an alternating way, starting with r.

Participants r and s inform k that a text has been accepted or refused by sending back, respectively, either *ack* or *nack*.

- In the first case it is the other receiver's turn to receive the text: a message *go* is exchanged between r and s to signal this case;
- in the second case, the sender has to resend the text until it is accepted. Meanwhile the involved participant between r and s informs the other one that she needs to *wait* since the previous message is being resent in a *transf*ormed form.

This protocol can be implemented by the multiparty session $\mathscr{M}' = \mathsf{r} \triangleright R \mid \mathsf{s} \triangleright S \mid \mathsf{k} \triangleright K_\mathsf{r}$
where

$$\begin{array}{lcllcllcl} R & = & \mathsf{k}?\mathit{text}.R_1 & R_1 & = & \mathsf{k}!\{\mathit{ack}.\mathsf{s}!\mathit{go}.R_2, \mathit{nack}.\mathsf{s}!\mathit{wait}.\mathsf{k}?\mathit{transf}.R_1\} & R_2 & = & \mathsf{s}?\{\mathit{go}.R, \mathit{wait}.R_2\} \\ S & = & \mathsf{r}?\{\mathit{go}.\mathsf{k}?\mathit{text}.S_1, \mathit{wait}.S\} & S_1 & = & \mathsf{k}!\{\mathit{ack}.\mathsf{r}!\mathit{go}.S, \mathit{nack}.\mathsf{r}!\mathit{wait}.\mathsf{k}?\mathit{transf}.S_1\} & & & \\ K_\mathsf{r} & = & \mathsf{r}!\mathit{text}.K'_\mathsf{r} & & & & K'_\mathsf{r} & = & \mathsf{r}?\{\mathit{ack}.K_\mathsf{s}, \mathit{nack}.\mathsf{r}!\mathit{transf}.K'_\mathsf{r}\} \\ K_\mathsf{s} & = & \mathsf{s}!\mathit{text}.K'_\mathsf{s} & & & & K'_\mathsf{s} & = & \mathsf{s}?\{\mathit{ack}.K_\mathsf{r}, \mathit{nack}.\mathsf{s}!\mathit{transf}.K'_\mathsf{s}\} \end{array}$$

The "behaviour as interface" of participant k corresponds to
$\mathrm{ERS}(K_\mathsf{r}) = \mathrm{ERS}(K_\mathsf{s}) = \circ!\mathit{text}.\mathrm{ERS}(K'_\mathsf{r}) \quad \mathrm{ERS}(K'_\mathsf{r}) = \mathrm{ERS}(K'_\mathsf{s}) = \circ?\{\mathit{ack}.\mathrm{ERS}(K_\mathsf{r}), \mathit{nack}.\circ!\mathit{transf}.\mathrm{ERS}(K'_\mathsf{r})\}$
Notice that the mapping ERS equates K_r and K_s, i.e. $\mathrm{ERS}(K_\mathsf{r}) = \mathrm{ERS}(K_\mathsf{s})$.
The interactions "offered" and "requested" by $\mathrm{ERS}(H)$ and $\mathrm{ERS}(K_\mathsf{r})$ do not precisely match each other, that is $\overline{\mathrm{ERS}(H)} \neq \mathrm{ERS}(K_\mathsf{r})$ (where $\overline{(\cdot)}$ is the standard syntactic duality function replacing '!' by '?' and vice versa [16]). Nonetheless it is easy to check that, even if the system $\mathsf{p} \triangleright P \mid \mathsf{q} \triangleright Q$ of Example 2.4 can safely deal with a message *stop* coming from its environment, no problem arises in case no such a message will ever arrive.
In the following definition, instead of explicitly introduce the "ERS" function, we simply formalise the compatibility relation in such a way two processes are compatible (as interfaces) whenever they offer dual communications to *arbitrary* participants, and the set of input labels is a subset of the set of output labels.

Definition 5.1 (Processes' Compatibility) *The* interface compatibility relation $P \leftrightarrow Q$ *on processes (*compatibility *for short), is the largest symmetric relation coinductively defined by:*

$$[\text{COMP-}\mathbf{0}] \quad \mathbf{0} \leftrightarrow \mathbf{0} \qquad\qquad [\text{COMP- O/I }] \quad \frac{P_i \leftrightarrow Q_i \quad \forall 1 \leq i \leq n}{\mathsf{p}!(\{\ell_i.P_i \mid {\scriptstyle 1 \leq i \leq n}\} \uplus \Lambda) \leftrightarrow \mathsf{q}?\{\ell_i.Q_i \mid {\scriptstyle 1 \leq i \leq n}\}}$$

The double line in rule [COMP-O/I] indicates that the rule is *coinductive*. Notice that the relation $\leftrightarrow$ is insensitive to the names of senders and receivers. It is immediate to verify that process compatibility is similar and simpler than the subtyping defined in [14]. Therefore an algorithm for checking process compatibility can be an easy adaptation of the algorithm given in [14].
For what concerns our example, it is straightforward to verify that $H \leftrightarrow K_\mathsf{r}$.

Useful properties of compatibility are stated in the following proposition, whose proof is simple.

Proposition 5.2 *(i) If* $P \leftrightarrow \mathsf{p}?(\Lambda \uplus \Lambda')$*, then* $P \leftrightarrow \mathsf{p}?\Lambda$.

(ii) If $\mathsf{p}!(\ell.P \uplus \Lambda) \leftrightarrow \mathsf{q}?\ell.Q$*, then* $P \leftrightarrow Q$.

Similarly to what is done in [1] for the setting of Communicating Finite State Machines (CFSMs), the presence of two compatible processes H and K in two multiparty sessions $\mathscr{M}$ and $\mathscr{M}'$ enables to connect these systems by transforming H and K in such a way each message received by H is immediately sent to K, and each message sent by H is first received from K. And similarly for what concerns K. In the following definition we hence transform an arbitrary process P not containing a fixed participant h into a process which: 1. sends to h each message received in P; 2. receives from h each message sent in P. We call $\mathsf{gw}(P, \mathsf{h})$ the so obtained process.

Definition 5.3 (Gateway Process) *Let* $\mathsf{h} \notin \mathtt{ptp}(P)$. *We define* $\mathsf{gw}(P, \mathsf{h})$ *coinductively as follows*

$$\begin{array}{lcl} \mathsf{gw}(\mathbf{0}, \mathsf{h}) & = & \mathbf{0} \\ \mathsf{gw}(\mathsf{p}?\{\ell_i.P_i \mid {\scriptstyle 1 \leq i \leq n}\}, \mathsf{h}) & = & \mathsf{p}?\{\ell_i.\mathsf{h}!\ell_i.\mathsf{gw}(P_i, \mathsf{h}) \mid {\scriptstyle 1 \leq i \leq n}\} \\ \mathsf{gw}(\mathsf{p}!\{\ell_i.P_i \mid {\scriptstyle 1 \leq i \leq n}\}, \mathsf{h}) & = & \mathsf{h}?\{\ell_i.\mathsf{p}!\ell_i.\mathsf{gw}(P_i, \mathsf{h}) \mid {\scriptstyle 1 \leq i \leq n}\} \end{array}$$

A first lemma assures the soundness of the previous definition.

Lemma 5.4 *If* $\mathsf{h} \notin \mathtt{ptp}(P)$*, then* $\mathsf{gw}(P,\mathsf{h})$ *is defined and is a function.*

Proof. The proof is by coinduction. If $P = \mathsf{p}?\{\ell_i.P_i \mid 1\leq i\leq n\}$, then

$$\mathsf{gw}(\mathsf{p}?\{\ell_i.P_i \mid 1\leq i\leq n\},\mathsf{h}) = \mathsf{p}?\{\ell_i.\mathsf{h}!\ell_i.\mathsf{gw}(P_i,\mathsf{h}) \mid 1\leq i\leq n\}$$

By coinduction $\mathsf{gw}(P_i,\mathsf{h})$ is defined and is a function for $1 \leq i \leq n$. The thesis hence follows. Similarly when P is an output process. □

The gateway process construction enjoys the preservation of the structural preorder. This property is the key to get Theorem 6.7 below and it essentially relies on the fact that bigger processes offer the same output messages.

Lemma 5.5 *Let* $\mathsf{h} \notin \mathtt{ptp}(P) \cup \mathtt{ptp}(Q)$*. If* $P \leqslant Q$*, then* $\mathsf{gw}(P,\mathsf{h}) \leqslant \mathsf{gw}(Q,\mathsf{h})$.

Proof. We only consider the case of input processes, the proof for output processes is similar and simpler.
If $P = \mathsf{p}?\{\ell_i.P_i \mid 1\leq i\leq n\}$ and $Q = \mathsf{p}?\{\ell_i.Q_i \mid 1\leq i\leq n'\}$ with $n' \leq n$, then $\mathsf{gw}(P,\mathsf{h}) = \mathsf{p}?\{\ell_i.\mathsf{h}!\ell_i.\mathsf{gw}(P_i,\mathsf{h}) \mid 1\leq i\leq n\}$ and $\mathsf{gw}(Q,\mathsf{h}) = \mathsf{p}?\{\ell_i.\mathsf{h}!\ell_i.\mathsf{gw}(Q_i,\mathsf{h}) \mid 1\leq i\leq n'\}$. From $P \leqslant Q$ we get $P_i \leqslant Q_i$ for all i, $1 \leq i \leq n'$. By coinduction $\mathsf{gw}(P_i,\mathsf{h}) \leqslant \mathsf{gw}(Q_i,\mathsf{h})$, which implies $\mathsf{h}!\ell_i.\mathsf{gw}(P_i,\mathsf{h}) \leqslant \mathsf{h}!\ell_i.\mathsf{gw}(Q_i,\mathsf{h})$ for all i, $1 \leq i \leq n'$, and hence $\mathsf{gw}(P,\mathsf{h}) \leqslant \mathsf{gw}(Q,\mathsf{h})$, by definition of $\leqslant$ (Definition 3.8). □

Lemma 5.5 fails for $\leqslant^+$. For example, if $P = \mathsf{p}!\ell_1$ and $Q = \mathsf{p}!\{\ell_1,\ell_2\}$, then $P \leqslant^+ Q$, but $\mathsf{gw}(P,\mathsf{h}) = \mathsf{h}?\ell_1.\mathsf{p}!\ell_1 \not\leqslant^+ \mathsf{h}?\{\ell_1.\mathsf{p}!\ell_1,\ell_2.\mathsf{p}!\ell_1\} = \mathsf{gw}(Q,\mathsf{h})$.

The following relationship between compatibility and structural preorder of processes will be essential in the proof of our main result (Theorem 6.7).

Lemma 5.6 *If* $P \leftrightarrow Q$*, then* $P \leqslant P'$ *and* $Q \leqslant Q'$ *imply* $P' \leftrightarrow Q'$.

Proof. Let us assume

$$\begin{array}{ccc} P = \mathsf{p}!\{\ell_i.P_i \mid 1\leq i\leq n\} & \leqslant & P' = \mathsf{p}!\{\ell_i.P'_i \mid 1\leq i\leq n\} \\ \updownarrow & & \\ Q = \mathsf{q}?\{\ell_i.Q_i \mid 1\leq i\leq n'\} & \leqslant & Q' = \mathsf{q}?\{\ell_i.Q'_i \mid 1\leq i\leq n''\} \quad \text{with } n'' \leq n' \leq n \end{array}$$

From $P \leftrightarrow Q$ we get $P_i \leftrightarrow Q_i$ for all i, $1 \leq i \leq n'$. From $P \leqslant P'$ we get $P_i \leqslant P'_i$ for all i, $1 \leq i \leq n$. From $Q \leqslant Q'$ we get $Q_i \leqslant Q'_i$ for all i, $1 \leq i \leq n''$. By coinduction we have $P'_i \leftrightarrow Q'_i$ for all i, $1 \leq i \leq n''$. We can then conclude $P' \leftrightarrow Q'$. □

The vice versa does not hold. For example $\mathsf{p}!\ell \leftrightarrow \mathsf{q}?\ell$ and $\mathsf{q}?\{\ell,\ell'\} \leqslant \mathsf{q}?\ell$, but $\mathsf{p}!\ell \not\leftrightarrow \mathsf{q}?\{\ell,\ell'\}$ is false.

The formal definition of connection of multiparty sessions via gateways is based on the notion of process compatibility (Definition 5.1) and on the addition of communications to a process (Definition 5.3).

Definition 5.7 (Multiparty-Sessions' Compatibility)
Two multiparty sessions $\mathscr{M}$*,* $\mathscr{M}'$ *are* compatible via the participants h, k (notation $(\mathscr{M},\mathsf{h}) \leftrightarrow (\mathscr{M}',\mathsf{k})$) if

$$\mathtt{pts}(\mathscr{M}) \cap \mathtt{pts}(\mathscr{M}') = \emptyset \quad \text{and} \quad \mathscr{M} \equiv \mathscr{M}_1 \mid \mathsf{h} \triangleright H \quad \text{and} \quad \mathscr{M}' \equiv \mathscr{M}_2 \mid \mathsf{k} \triangleright K \quad \text{with } H \leftrightarrow K.$$

Definition 5.8 (Multiparty-Sessions' Connection via Gateways)
Let $\mathscr{M} \equiv \mathscr{M}_1 \mid \mathsf{h} \triangleright H$*,* $\mathscr{M}' \equiv \mathscr{M}_2 \mid \mathsf{k} \triangleright K$ *and* $(\mathscr{M},\mathsf{h}) \leftrightarrow (\mathscr{M}',\mathsf{k})$*. We define* $\mathscr{M}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathscr{M}'$*, the connection of* $\mathscr{M}$ *and* $\mathscr{M}'$ *via gateways, through* h *and* k*, by*

$$\mathscr{M}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathscr{M}' \triangleq \mathscr{M}_1 \mid \mathscr{M}_2 \mid \mathsf{h} \triangleright \mathsf{gw}(H,\mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K,\mathsf{h})$$

Example 5.9 For what concerns $\mathscr{M}$ of Example 2.4, $\mathscr{M}'$ defined on page 10, h and k, it is not difficult to check that

$$\mathscr{M}\,{}^{\mathsf{h}\leftrightarrow\mathsf{k}}\!\mathscr{M}' \quad = \quad \mathsf{p}\triangleright P \mid \mathsf{q}\triangleright Q \mid \mathsf{r}\triangleright R \mid \mathsf{s}\triangleright S \mid \mathsf{h}\triangleright \hat{H} \mid \mathsf{k}\triangleright \hat{K}_{\mathsf{r}}$$

where

$\hat{H} = \mathsf{gw}(H,\mathsf{k}) = \mathsf{q}?\mathit{text}.\mathsf{k}!\mathit{text}.\hat{H}_1 \quad \hat{H}_1 = \mathsf{k}?\{\mathit{ack}.\mathsf{q}!\mathit{ack}.\hat{H},\ \mathit{nack}.\mathsf{q}!\mathit{nack}.\mathsf{q}?\mathit{transf}.\mathsf{k}!\mathit{transf}.\hat{H}_1,\ \mathit{stop}.\mathsf{q}!\mathit{stop}\}$

$\hat{K}_{\mathsf{r}} = \mathsf{gw}(K_{\mathsf{r}},\mathsf{h}) = \mathsf{h}?\mathit{text}.\mathsf{r}!\mathit{text}.\hat{K}'_{\mathsf{r}} \quad \hat{K}'_{\mathsf{r}} = \mathsf{r}?\{\mathit{ack}.\mathsf{h}!\mathit{ack}.\hat{K}_{\mathsf{s}},\ \mathit{nack}.\mathsf{h}!\mathit{nack}.\mathsf{h}?\mathit{transf}.\mathsf{r}!\mathit{transf}.\hat{K}'_{\mathsf{r}}\}$

$\hat{K}_{\mathsf{s}} = \mathsf{gw}(K_{\mathsf{s}},\mathsf{h}) = \mathsf{h}?\mathit{text}.\mathsf{s}!\mathit{text}.\hat{K}'_{\mathsf{s}} \quad \hat{K}'_{\mathsf{s}} = \mathsf{s}?\{\mathit{ack}.\mathsf{h}!\mathit{ack}.\hat{K}_{\mathsf{r}},\ \mathit{nack}.\mathsf{h}!\mathit{nack}.\mathsf{h}?\mathit{transf}.\mathsf{s}!\mathit{transf}.\hat{K}'_{\mathsf{s}}\}$

In the following section we shall prove that lock-freedom is preserved by the session connection via gateways. This follows from the fact that we define an operator building a global type such that the participant processes of the session obtained by connection via gateways are smaller than or equal to the projections of this global type.

6 Connection of Global Types via Gateways

The composition defined in the previous section can be shown to be lock-freedom preserving by means of Theorem 4.9. In fact it is possible to define a function on global types with compatible participants, which corresponds to the lifting of the construction in Definition 5.8 to the level of global types.

Definition 6.1 (Global-Types' Compatibility) *Two global types* G, G' *are* compatible via the participants h, k (notation $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$) if $\mathtt{ptg}(\mathsf{G}) \cap \mathtt{ptg}(\mathsf{G}') = \emptyset$ and $\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$.

Definition 6.2 (Global-Types' Connection via Gateways) *Let* $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$. *We define*

$$\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}' \triangleq \mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$$

where CN *is coinductively given by the following clauses, assuming* $\{\mathsf{p},\mathsf{q},\mathsf{r},\mathsf{s}\} \cap \{\mathsf{h},\mathsf{k}\} = \emptyset$. *The clauses must be applied in the given order.*

(1) $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathtt{end},\mathsf{G}') = \mathsf{G}'$

(2) $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{p}\to\mathsf{h}:\{\ell_i.\mathsf{G}_i \mid 1\le i\le n\},\mathsf{G}') = \mathsf{p}\to\mathsf{h}:\{\ell_i.\mathrm{CN}(\mathsf{h},\mathsf{k},\ell_i^{\rightarrow},\mathsf{G}_i,\mathsf{G}') \mid 1\le i\le n\}$

(3) $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{G},\mathsf{k}\to\mathsf{s}:\{\ell'_j.\mathsf{G}'_j \mid 1\le j\le m\}) = \mathsf{h}\to\mathsf{k}:\ell.\mathsf{k}\to\mathsf{s}:\ell.\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}'_\iota)$ *if* $\ell=\ell'_\iota$ *with* $1\le\iota\le m$

(4) $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{G},\mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathsf{G}'_j \mid 1\le j\le m\}) = \mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{G},\mathsf{G}'_j) \mid 1\le j\le m\}$

(5) $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{r}\to\mathsf{k}:\{\ell'_j.\mathsf{G}'_j \mid 1\le j\le m\}) = \mathsf{r}\to\mathsf{k}:\{\ell'_j.\mathrm{CN}(\mathsf{h},\mathsf{k},\ell'^{\leftarrow}_j,\mathsf{G},\mathsf{G}'_j) \mid 1\le j\le m\}$

(6) $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{h}\to\mathsf{q}:\{\ell_i.\mathsf{G}_i \mid 1\le i\le n\},\mathsf{G}') = \mathsf{k}\to\mathsf{h}:\ell.\mathsf{h}\to\mathsf{q}:\ell.\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G}_\iota,\mathsf{G}')$ *if* $\ell=\ell_\iota$ *with* $1\le\iota\le n$

(7) $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{p}\to\mathsf{q}:\{\ell_i.\mathsf{G}_i \mid 1\le i\le n\},\mathsf{G}') = \mathsf{p}\to\mathsf{q}:\{\ell_i.\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{G}_i,\mathsf{G}') \mid 1\le i\le n\}$

(8) $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{p}\to\mathsf{q}:\{\ell_i.\mathsf{G}_i \mid 1\le i\le n\},\mathsf{G}') = \mathsf{p}\to\mathsf{q}:\{\ell_i.\mathrm{CN}(\mathsf{k},\mathsf{h},\#,\mathsf{G}',\mathsf{G}_i) \mid 1\le i\le n\}$

(9) $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathsf{G}'_j \mid 1\le j\le m\}) = \mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathrm{CN}(\mathsf{k},\mathsf{h},\#,\mathsf{G}'_j,\mathsf{G}) \mid 1\le j\le m\}$

The argument '$\ell^{\rightarrow}$' (resp. '$\ell^{\leftarrow}$') in $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{G},\mathsf{G}')$ (resp. $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{G},\mathsf{G}')$) is used when a sending of the message 'ℓ' from k (resp. h) is expected in the second (resp. first) global type in the subsequent recursive calls.
The argument '$\#$' is used instead when all other possible interactions can occur in either the first or the second global type in the subsequent recursive calls.
In global types, the order of interactions between pairs of unrelated participants is irrelevant, since we

would get the very same projections. In clauses (8) and (9), however, we swap roles h and k, as well as their corresponding global types in the "recursive call". We do that in order to avoid that in $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$ the interactions preceding a communication via gateway all belong to G (or G') and that the communication is completed after the description of interactions all belonging to G' (or G). In this way the parallel nature of the interactions in G and G' that are not affected by the communications via gateways is made visually more evident.

Example 6.3 The protocol implemented by the multiparty session $\mathscr{M}'$ defined on page 10 can be represented by the following global type G_r:

$$\begin{array}{ll}\mathsf{G}_\mathsf{r} = \mathsf{k} \to \mathsf{r} : \mathit{text}.\mathsf{G}'_\mathsf{r} & \mathsf{G}_\mathsf{s} = \mathsf{k} \to \mathsf{s} : \mathit{text}.\mathsf{G}'_\mathsf{s}\\ \mathsf{G}'_\mathsf{r} = \mathsf{r} \to \mathsf{k} : \{\mathit{ack}.\mathsf{r} \to \mathsf{s} : \mathit{go}.\mathsf{G}_\mathsf{s}, & \mathsf{G}'_\mathsf{s} = \mathsf{s} \to \mathsf{k} : \{\mathit{ack}.\mathsf{s} \to \mathsf{r} : \mathit{go}.\mathsf{G}_\mathsf{r},\\ \quad \mathit{nack}.\mathsf{r} \to \mathsf{s} : \mathit{wait}.\mathsf{k} \to \mathsf{r} : \mathit{transf}.\mathsf{G}'_\mathsf{r}\} & \quad \mathit{nack}.\mathsf{s} \to \mathsf{r} : \mathit{wait}.\mathsf{k} \to \mathsf{s} : \mathit{transf}.\mathsf{G}'_\mathsf{s}\}\end{array}$$

Then, by Definition 6.2, the composition, via h and k, of the G of Example 3.2 and the above G_r is:

$$\begin{array}{lcl}\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}_\mathsf{r} & = & \mathsf{p} \to \mathsf{q} : \mathit{text}.\mathsf{q} \to \mathsf{h} : \mathit{text}.\mathsf{h} \to \mathsf{k} : \mathit{text}.\mathsf{k} \to \mathsf{r} : \mathit{text}.\mathsf{G}'^{\mathsf{k}\leftrightarrow\mathsf{h}}_\mathsf{r}\mathsf{G}_1\\ \mathsf{G}'^{\mathsf{k}\leftrightarrow\mathsf{h}}_\mathsf{r}\mathsf{G}_1 & = & \mathsf{r} \to \mathsf{k} : \{\mathit{ack}.\mathsf{k} \to \mathsf{h} : \mathit{ack}.\mathsf{h} \to \mathsf{q} : \mathit{ack}.\mathsf{r} \to \mathsf{s} : \mathit{go}.\mathsf{q} \to \mathsf{p} : \mathit{ok}.\mathsf{G}_\mathsf{s}{}^{\mathsf{k}\leftrightarrow\mathsf{h}}\mathsf{G},\\ & & \quad \mathit{nack}.\mathsf{k} \to \mathsf{h} : \mathit{nack}.\mathsf{h} \to \mathsf{q} : \mathit{nack}.\mathsf{r} \to \mathsf{s} : \mathit{wait}.\mathsf{q} \to \mathsf{p} : \mathit{notyet}.\\ & & \quad\quad \mathsf{q} \to \mathsf{h} : \mathit{transf}.\mathsf{h} \to \mathsf{k} : \mathit{transf}.\mathsf{k} \to \mathsf{r} : \mathit{transf}.\mathsf{G}'^{\mathsf{k}\leftrightarrow\mathsf{h}}_\mathsf{r}\mathsf{G}_1\}\\ \mathsf{G}_\mathsf{s}{}^{\mathsf{k}\leftrightarrow\mathsf{h}}\mathsf{G} & = & \mathsf{p} \to \mathsf{q} : \mathit{text}.\mathsf{q} \to \mathsf{h} : \mathit{text}.\mathsf{h} \to \mathsf{k} : \mathit{text}.\mathsf{k} \to \mathsf{s} : \mathit{text}.\mathsf{G}_1{}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'_\mathsf{s}\\ \mathsf{G}_1{}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'_\mathsf{s} & = & \mathsf{s} \to \mathsf{k} : \{\mathit{ack}.\mathsf{k} \to \mathsf{h} : \mathit{ack}.\mathsf{h} \to \mathsf{q} : \mathit{ack}.\mathsf{q} \to \mathsf{p} : \mathit{ok}.\mathsf{s} \to \mathsf{r} : \mathit{go}.\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}_\mathsf{r},\\ & & \quad \mathit{nack}.\mathsf{k} \to \mathsf{h} : \mathit{nack}.\mathsf{h} \to \mathsf{q} : \mathit{nack}.\mathsf{q} \to \mathsf{p} : \mathit{notyet}.\mathsf{s} \to \mathsf{r} : \mathit{wait}.\\ & & \quad\quad \mathsf{q} \to \mathsf{h} : \mathit{transf}.\mathsf{h} \to \mathsf{k} : \mathit{transf}.\mathsf{k} \to \mathsf{s} : \mathit{transf}.\mathsf{G}_1{}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'_\mathsf{s}\}\end{array}$$

In $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}_\mathsf{r}$ the text messages coming from p are delivered to q and, alternately, to r and s till they are accepted (*ack*). Participant p is informed when text messages are accepted (*ok*). During the cycle, q transforms a not yet accepted text into a more suitable form. The messages between q and r and s are exchanged by passing through the coupled forwarders h and k.

It is worth pointing out that in $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}_\mathsf{r}$, the *stop* branch of G disappeared. In fact, since any message coming from h in G does now come from k (which is now the gateway forwarding the messages coming in turn from either r or s), the function CN takes care of the fact that only *ack* or *nack* can be received by (the gateway) h. This fact is reflected in the following Theorem 6.6, where it is shown that the projections on h and k of $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'$ are a "supertype" of $\mathsf{gw}(\mathsf{G}\upharpoonright_\mathsf{h},\mathsf{k})$ and $\mathsf{gw}(\mathsf{G}'\upharpoonright_\mathsf{k},\mathsf{h})$, respectively.

We could look at both h and p as interfaces: h representing a social-network system, which does not accept rude language, and p a social-network client sending text messages and requiring to be informed about their delivery status. From this point of view, the global type G of Example 3.2 actually describes a "delivery-guaranteed" service for text messages, assuring messages to be eventually delivered by means of a text-transformation policy.

The following lemma assures that the global types obtained during the evaluation of CN are always compatible.

Lemma 6.4 *Let* $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$. *Then for any call in the tree of the recursive calls of* $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$*:*

(a) if the call is $\mathrm{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}')$*, then* $\mathsf{Y}\upharpoonright_\mathsf{h} \leftrightarrow \mathsf{Y}'\upharpoonright_\mathsf{k}$*;*

(b) if the call is $\mathrm{CN}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{Y},\mathsf{Y}')$*, then* $\mathsf{p}?\ell.\mathsf{Y}\upharpoonright_\mathsf{h} \leftrightarrow \mathsf{Y}'\upharpoonright_\mathsf{k}$ *for some* p*;*

(c) if the call is $\text{CN}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{Y},\mathsf{Y}')$*, then* $\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{p}?\ell.\mathsf{Y}'\upharpoonright_{\mathsf{k}}$ *for some* p.

Proof. We show (a), (b) and (c) simultaneously by induction on the depth of the call in the tree, and by cases on the applied rule. For rule (1) the proof is immediate, since no new call is generated.
Rule (2). By induction on (a), $(\mathsf{p}\to\mathsf{h}:\{\ell_i.\mathsf{Y}_i \mid 1\le i\le n\})\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$. By Definition 3.5

$$(\mathsf{p}\to\mathsf{h}:\{\ell_i.\mathsf{Y}_i \mid 1\le i\le n\})\upharpoonright_{\mathsf{h}} = \mathsf{p}?\{\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \mid 1\le i\le n\}$$

Then $\mathsf{p}?\{\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \mid 1\le i\le n\} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$, which implies $\mathsf{p}?\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$ for $1\le i\le n$ by Proposition 5.2(i).
Rule (3). By induction on (b), $\mathsf{p}?\ell.\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow (\mathsf{k}\to\mathsf{s}:\{\ell'_j.\mathsf{Y}'_j \mid 1\le j\le m\})\upharpoonright_{\mathsf{k}}$. By Definition 3.5

$$(\mathsf{k}\to\mathsf{s}:\{\ell'_j.\mathsf{Y}'_j \mid 1\le j\le m\})\upharpoonright_{\mathsf{k}} = \mathsf{s}!\{\ell'_j.\mathsf{Y}'_j\upharpoonright_{\mathsf{k}} \mid 1\le j\le m\}$$

By Proposition 5.2(ii) $\ell=\ell'_\iota$ with $1\le\iota\le m$ implies $\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'_\iota\upharpoonright_{\mathsf{k}}$.
Rule (4). By induction on (b), $\mathsf{p}?\ell.\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow (\mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathsf{Y}'_j \mid 1\le j\le m\})\upharpoonright_{\mathsf{k}}$. By Definition 5.1 the projection

$$(\mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathsf{Y}'_j \mid 1\le j\le m\})\upharpoonright_{\mathsf{k}}$$

must be an output, which implies $(\mathsf{r}\to\mathsf{s}:\{\ell'_j.\mathsf{Y}'_j \mid 1\le j\le m\})\upharpoonright_{\mathsf{k}} = \mathsf{Y}'_1\upharpoonright_{\mathsf{k}}$ and $\mathsf{Y}'_j\upharpoonright_{\mathsf{k}} = \mathsf{Y}'_l\upharpoonright_{\mathsf{k}}$ for $1\le j,l\le m$ by Definition 3.5. We conclude $\mathsf{p}?\ell.\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'_j\upharpoonright_{\mathsf{k}}$ for $1\le j\le m$.
Rule (8). By induction on (a), $(\mathsf{p}\to\mathsf{q}:\{\ell_i.\mathsf{Y}_i \mid 1\le i\le n\})\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$. By Definition 3.5 either

$$(\mathsf{p}\to\mathsf{q}:\{\ell_i.\mathsf{Y}_i \mid 1\le i\le n\})\upharpoonright_{\mathsf{h}} = \mathsf{Y}_1\upharpoonright_{\mathsf{h}}$$

and $\mathsf{Y}_i\upharpoonright_{\mathsf{h}} = \mathsf{Y}_l\upharpoonright_{\mathsf{h}}$ for $1\le i,l\le n$ or $(\mathsf{p}\to\mathsf{q}:\{\ell_i.\mathsf{Y}_i \mid 1\le i\le n\})\upharpoonright_{\mathsf{h}} = \mathsf{t}?(\Lambda_1\uplus\ldots\uplus\Lambda_n)$ and $\mathsf{Y}_i\upharpoonright_{\mathsf{h}} = \mathsf{t}?\Lambda_i$ for $1\le i\le n$. In the first case we get immediately $\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$ for $1\le i\le n$. In the second case by Proposition 5.2(i) $\mathsf{t}?(\Lambda_1\uplus\ldots\uplus\Lambda_n) \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$ implies $\mathsf{t}?\Lambda_i \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$, i.e. $\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$, for $1\le i\le n$.
The proofs for rules (5), (6),(7) and (9) are similar to those of rules (2), (3),(4) and (8), respectively. □

Using the previous lemma we can show the soundness of Definition 6.2.

Lemma 6.5 *Let* $(\mathsf{G},\mathsf{h})\leftrightarrow(\mathsf{G}',\mathsf{k})$*. Then* $\text{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$ *is defined and it is a global type, i.e. a regular pre-global type.*

Proof. To show that $\text{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$ is defined, let us assume, towards a contradiction, that in the tree of the recursive calls of $\text{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$ there is one leaf on which no rule of Definition 6.2 can be applied. If the recursive call is $\text{CN}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}')$, then the applicable rules are (1), (2), (5), (8) and (9). So the only deadlock would be for $\mathsf{Y}=\mathsf{h}\to\mathsf{p}:\Gamma$ and $\mathsf{Y}'\neq\mathsf{r}\to\mathsf{k}:\Gamma'$ and $\mathsf{Y}'\neq\mathsf{r}\to\mathsf{s}:\Gamma''$. This is impossible since by Lemma 6.4(a) $\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$. If the recursive call is $\text{CN}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{Y},\mathsf{Y}')$, then the applicable rules are (3) and (4). So the only deadlock would be for $\mathsf{Y}'\neq\mathsf{r}\to\mathsf{s}:\Gamma$ and $\mathsf{Y}'\neq\mathsf{r}\to\mathsf{k}:\Gamma'$. This is impossible since by Lemma 6.4(b) $\mathsf{p}?\ell.\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}}$. The proof for the recursive call $\text{CN}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{Y},\mathsf{Y}')$ uses Lemma 6.4(c) and it is similar to the previous one.
The regularity of the obtained pre-global type follows from observing that the regularity of G and G' forbid an infinite path, in the tree of the recursive calls, in which no two calls are identical. □

We can now prove the main result concerning projections of types obtained by connecting via gateways.

Theorem 6.6 *If* $(\mathsf{G},\mathsf{h})\leftrightarrow(\mathsf{G}',\mathsf{k})$*, then* $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'$ *is well formed. Moreover*

(i) $\mathsf{gw}(\mathsf{G}\upharpoonright_{\mathsf{h}},\mathsf{k}) \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_{\mathsf{h}}$ *and* $\mathsf{gw}(\mathsf{G}'\upharpoonright_{\mathsf{k}},\mathsf{h}) \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_{\mathsf{k}}$*;*

(ii) $\mathsf{G}\upharpoonright_{\mathsf{p}} \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_{\mathsf{p}}$ *and* $\mathsf{G}'\upharpoonright_{\mathsf{q}} \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_{\mathsf{q}}$,
for any $\mathsf{p}\in\mathtt{ptg}(\mathsf{G})$ *and* $\mathsf{q}\in\mathtt{ptg}(\mathsf{G}')$ *such that* $\mathsf{p}\neq\mathsf{h}$ *and* $\mathsf{q}\neq\mathsf{k}$.

Proof. It is easy to verify that if

$$w = max\{weight(\mathsf{G},\mathsf{p}) \mid \mathsf{p}\in\mathtt{ptg}(\mathsf{G})\} \text{ and } w' = max\{weight(\mathsf{G}',\mathsf{p}) \mid \mathsf{p}\in\mathtt{ptg}(\mathsf{G}')\}$$

then $weight(\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}',\mathsf{p}) \le 2(w+w')$ for all $\mathsf{p}\in\mathtt{ptg}(\mathsf{G})\cup\mathtt{ptg}(\mathsf{G}')$. Since (i) and (ii) imply that $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'$ is projectable for all $\mathsf{p}\in\mathtt{ptg}(\mathsf{G})\cup\mathtt{ptg}(\mathsf{G}')$, then $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'$ is well formed. Let $\star\in\{\#,\ell^{\rightarrow},\ell^{\leftarrow}\}$.

(i). We only show $\mathsf{gw}(\mathsf{G}\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_\mathsf{h}$, the proof of $\mathsf{gw}(\mathsf{G}'\upharpoonright_\mathsf{k},\mathsf{h}) \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_\mathsf{k}$ is specular. We prove that, for any recursive call $\textsc{cn}(\mathsf{h},\mathsf{k},\star,\mathsf{Y},\mathsf{Y}')$ in $\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$, the following relations between processes hold:

(a) $\mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}')\upharpoonright_\mathsf{h}$;

(b) $\mathsf{k}!\ell.\mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{Y},\mathsf{Y}')\upharpoonright_\mathsf{h}$;

(c) $\mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{Y},\mathsf{Y}')\upharpoonright_\mathsf{h}$.

We prove (a), (b) and (c) simultaneously by coinduction on Y and Y' and by cases on the rule applied to get $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}' = \textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$. Rules (1), (4), (5), (7), (8) and (9) do not modify the communications of participant h, so coinduction easily applies.

Rule (2): $\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{p}\to\mathsf{h}:\{\ell_i.\mathsf{Y}_i \mid 1{\leq}i{\leq}n\},\mathsf{Y}') = \mathsf{p}\to\mathsf{h}:\{\ell_i.\textsc{cn}(\mathsf{h},\mathsf{k},\ell_i^{\rightarrow},\mathsf{Y}_i,\mathsf{Y}') \mid 1{\leq}i{\leq}n\}$.
Let $\mathsf{Y} = \mathsf{p}\to\mathsf{h}:\{\ell_i.\mathsf{Y}_i \mid 1{\leq}i{\leq}n\}$, then $\mathsf{Y}\upharpoonright_\mathsf{h} = \mathsf{p}?\{\ell_i.\mathsf{Y}_i\upharpoonright_\mathsf{h} \mid 1{\leq}i{\leq}n\}$ by Definition 3.5.

$$\begin{array}{rcll}
\mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) & = & \mathsf{p}?\{\ell_i.\mathsf{k}!\ell_i.\mathsf{gw}(\mathsf{Y}_i\upharpoonright_\mathsf{h},\mathsf{k}) \mid 1{\leq}i{\leq}n\} & \text{by Definition 5.3}\\
& \leqslant & \mathsf{p}?\{\ell_i.\textsc{cn}(\mathsf{h},\mathsf{k},\ell_i^{\rightarrow},\mathsf{Y}_i,\mathsf{Y}')\upharpoonright_\mathsf{h} \mid 1{\leq}i{\leq}n\} & \text{by rule [SUB-IN] of Definition 3.8 since}\\
& & & \mathsf{k}!\ell_i.\mathsf{gw}(\mathsf{Y}_i\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\ell_i^{\rightarrow},\mathsf{Y}_i,\mathsf{Y}')\upharpoonright_\mathsf{h}\\
& & & \text{for } 1\leq i\leq n \text{ by coinduction on (b)}\\
& = & (\mathsf{p}\to\mathsf{h}:\{\ell_i.\textsc{cn}(\mathsf{h},\mathsf{k},\ell_i^{\rightarrow},\mathsf{Y}_i,\mathsf{Y}') \mid 1{\leq}i{\leq}n\})\upharpoonright_\mathsf{h} & \text{by Definition 3.5}
\end{array}$$

Rule (3): $\textsc{cn}(\mathsf{h},\mathsf{k},\ell^{\rightarrow},\mathsf{Y},\mathsf{k}\to\mathsf{s}:\{\ell'_j.\mathsf{Y}'_j \mid 1{\leq}j{\leq}m\}) = \mathsf{h}\to\mathsf{k}:\ell.\mathsf{k}\to\mathsf{s}:\ell.\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}'_\iota)$, where $\ell = \ell'_\iota$ with $1\leq\iota\leq m$.

$$\begin{array}{rcll}
\mathsf{k}!\ell.\mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) & \leqslant & \mathsf{k}!\ell.\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}'_\iota)\upharpoonright_\mathsf{h} & \text{by rule [SUB-OUT] of Definition 3.8 since}\\
& & & \mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}'_\iota)\upharpoonright_\mathsf{h}\\
& & & \text{by coinduction on (a)}\\
& = & (\mathsf{h}\to\mathsf{k}:\ell.\mathsf{k}\to\mathsf{s}:\ell.\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y},\mathsf{Y}'_\iota))\upharpoonright_\mathsf{h} & \text{by Definition 3.5}
\end{array}$$

Rule (6): $\textsc{cn}(\mathsf{h},\mathsf{k},\ell^{\leftarrow},\mathsf{h}\to\mathsf{q}:\{\ell_i.\mathsf{Y}_i \mid 1{\leq}i{\leq}n\},\mathsf{Y}') = \mathsf{k}\to\mathsf{h}:\ell.\mathsf{h}\to\mathsf{q}:\ell.\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}')$, where $\ell = \ell_\iota$ with $1\leq\iota\leq n$. Let $\mathsf{Y} = \mathsf{h}\to\mathsf{q}:\{\ell_i.\mathsf{Y}_i \mid 1{\leq}i{\leq}n\}$, then $\mathsf{Y}\upharpoonright_\mathsf{h} = \mathsf{q}!\{\ell_i.\mathsf{Y}_i\upharpoonright_\mathsf{h} \mid 1{\leq}i{\leq}n\}$ by Definition 3.5.

$$\begin{array}{rcll}
\mathsf{gw}(\mathsf{Y}\upharpoonright_\mathsf{h},\mathsf{k}) & = & \mathsf{k}?\{\ell_i.\mathsf{q}!\ell_i.\mathsf{gw}(\mathsf{Y}_i\upharpoonright_\mathsf{h},\mathsf{k}) \mid 1{\leq}i{\leq}n\} & \text{by Definition 5.3}\\
& \leqslant & \mathsf{k}?\ell.\mathsf{q}!\ell.\mathsf{gw}(\mathsf{Y}_\iota\upharpoonright_\mathsf{h},\mathsf{k}) & \text{by rule [SUB-IN] and } \ell=\ell_\iota\\
& \leqslant & \mathsf{k}?\ell.\mathsf{q}!\ell.(\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}'))\upharpoonright_\mathsf{h} & \text{by rules [SUB-IN] and [SUB-OUT] since}\\
& & & \mathsf{gw}(\mathsf{Y}_\iota\upharpoonright_\mathsf{h},\mathsf{k}) \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}')\upharpoonright_\mathsf{h}\\
& & & \text{by coinduction on (a)}\\
& = & (\mathsf{k}\to\mathsf{h}:\ell.\mathsf{h}\to\mathsf{q}:\ell.\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}'))\upharpoonright_\mathsf{h} & \text{by Definition 3.5}
\end{array}$$

(ii). We only show $\mathsf{G}\upharpoonright_\mathsf{q} \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_\mathsf{q}$ for $\mathsf{q}\in\mathtt{ptg}(\mathsf{G})$ and $\mathsf{q}\neq\mathsf{h}$. The proof of $\mathsf{G}\upharpoonright_\mathsf{s} \leqslant (\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}')\upharpoonright_\mathsf{s}$ for $\mathsf{s}\in\mathtt{ptg}(\mathsf{G}')$ and $\mathsf{s}\neq\mathsf{k}$ is specular. We consider the recursive calls $\textsc{cn}(\mathsf{h},\mathsf{k},\star,\mathsf{Y},\mathsf{Y}')$ in $\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{G},\mathsf{G}')$. We prove $\mathsf{Y}\upharpoonright_\mathsf{q} \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\star,\mathsf{Y},\mathsf{Y}')\upharpoonright_\mathsf{q}$ by coinduction on Y,Y' and by cases on the applied rule. The only rule which modifies the communications of q is rule (6). Let $\mathsf{Y} = \mathsf{h}\to\mathsf{q}:\{\ell_i.\mathsf{Y}_i \mid 1{\leq}i{\leq}n\}$, then

$$\begin{array}{rcll}
\mathsf{Y}\upharpoonright_\mathsf{q} & = & \mathsf{h}?\{\ell_i.\mathsf{Y}_i\upharpoonright_\mathsf{q} \mid 1{\leq}i{\leq}n\} & \text{by Definition 3.5}\\
& \leqslant & \mathsf{h}?\ell.\mathsf{Y}_\iota\upharpoonright_\mathsf{q} & \text{by rule [SUB-IN] and } \ell=\ell_\iota\\
& \leqslant & \mathsf{h}?\ell.(\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}'))\upharpoonright_\mathsf{q} & \text{by rule [SUB-IN] since by coinduction}\\
& & & \mathsf{Y}_\iota\upharpoonright_\mathsf{q} \leqslant \textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}')\upharpoonright_\mathsf{q}\\
& = & (\mathsf{k}\to\mathsf{h}:\ell.\mathsf{h}\to\mathsf{q}:\ell.\textsc{cn}(\mathsf{h},\mathsf{k},\#,\mathsf{Y}_\iota,\mathsf{Y}'))\upharpoonright_\mathsf{q} & \text{by Definition 3.5}
\end{array}$$

□

We now show that if we start from two well-typed sessions which are compatible, then by building their connection via gateways we get a well-typed session too. This is relevant, since well-typed sessions enjoy lock-freedom (Theorem 4.9).

Theorem 6.7 *If* $(\mathscr{M},\mathsf{h})\leftrightarrow(\mathscr{M}',\mathsf{k})$ *and* $\vdash\mathscr{M}:\mathsf{G}$ *and* $\vdash\mathscr{M}':\mathsf{G}'$, *then* $\vdash\mathscr{M}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathscr{M}':\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'$.

Proof. The typing $\vdash \mathscr{M} : \mathsf{G}$ implies $\mathtt{ptg}(\mathsf{G}) \subseteq \mathtt{pts}(\mathscr{M})$. The typing $\vdash \mathscr{M}' : \mathsf{G}'$ implies $\mathtt{ptg}(\mathsf{G}') \subseteq \mathtt{pts}(\mathscr{M}')$. Then $\mathtt{pts}(\mathscr{M}) \cap \mathtt{pts}(\mathscr{M}') = \emptyset$ gives $\mathtt{ptg}(\mathsf{G}) \cap \mathtt{ptg}(\mathsf{G}') = \emptyset$. Let $\mathscr{M} = \mathscr{M}_1 \mid \mathsf{h} \triangleright H$ and $\mathscr{M}' = \mathscr{M}_2 \mid \mathsf{k} \triangleright K$. By construction

$$\mathscr{M}^{\mathsf{h} \leftrightarrow \mathsf{k}} \mathscr{M}' = \mathscr{M}_1 \mid \mathscr{M}_2 \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h})$$

From $\vdash \mathscr{M} : \mathsf{G}$ we get $H \leqslant \mathsf{G}\upharpoonright_{\mathsf{h}}$. From $\vdash \mathscr{M}' : \mathsf{G}'$ we get $K \leqslant \mathsf{G}'\upharpoonright_{\mathsf{k}}$. Lemma 5.6 implies $\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$. Lemma 5.5 implies $\mathsf{gw}(H, \mathsf{k}) \leqslant \mathsf{gw}(\mathsf{G}\upharpoonright_{\mathsf{h}}, \mathsf{k})$ and $\mathsf{gw}(K, \mathsf{h}) \leqslant \mathsf{gw}(\mathsf{G}'\upharpoonright_{\mathsf{k}}, \mathsf{h})$.
We conclude $\vdash \mathscr{M}^{\mathsf{h} \leftrightarrow \mathsf{k}} \mathscr{M}' : \mathsf{G}^{\mathsf{h} \leftrightarrow \mathsf{k}} \mathsf{G}'$ using the projections of $\mathsf{G}^{\mathsf{h} \leftrightarrow \mathsf{k}} \mathsf{G}'$ given in Theorem 6.6. □

It is worth noticing that $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$ and $\vdash \mathscr{M} : \mathsf{G}$ and $\vdash \mathscr{M}' : \mathsf{G}'$ do not imply $(\mathscr{M}, \mathsf{h}) \leftrightarrow (\mathscr{M}', \mathsf{k})$. Take as example $\mathscr{M} = \mathsf{p} \triangleright \mathsf{h}?\ell \mid \mathsf{h} \triangleright \mathsf{p}!\ell$, $\mathscr{M}' = \mathsf{q} \triangleright \mathsf{k}!\ell \mid \mathsf{k} \triangleright \mathsf{q}?\{\ell, \ell'\}$, $\mathsf{G} = \mathsf{h} \to \mathsf{p} : \ell$, $\mathsf{G}' = \mathsf{q} \to \mathsf{k} : \ell$. In fact $\mathsf{p}!\ell \leftrightarrow \mathsf{q}?\{\ell, \ell'\}$ does not hold.

Remark 6.8 The proof of Theorem 6.7 uses Lemma 5.5 which fails for the typing system $\vdash^+$. In spite of this, we conjecture that Theorem 6.7 holds for $\vdash^+$ as well. The main reason is that compatibility requires all inputs to have corresponding outputs, and this forbids to exploit the difference between $\leqslant$ and $\leqslant^+$.

Of course we could relax our typing system so that, in Rule [T-SESS], $\leqslant$ is used for interface processes (i.e. those that are transformed into gateways when systems are connected), while $\leqslant^+$ is used for all other processes. This would result, however, in a fairly serious restriction of the flexibility of system connections, since we should establish a priori the interfaces of systems.

As a possible general applications of our results, let us suppose we have two systems that correspond to multiparty-sessions that are compatible via some participants (according to Definition 5.7) and that are well typed (according to Definition 3.9). At this point we can "deploy" the connected system (following Definition 5.8) without any further verification step, since Theorem 6.7 ensures that in such conditions we have a well-typed and hence lock-free connected system. Besides, we are able to provide the documentation (the global type) of the resulting systems.

7 Related Works

The distinguishing feature of an *open* system of concurrent components is its capacity of communicating with the "outside", i.e. with an environment of the system. This ability provides means for composing open systems to larger systems (which may still be open). In order to compose systems "safely", it is common practice to rely on interface descriptions.
MPST systems [18, 8, 25] are usually assumed to be *closed*, since all the components needed for the functioning of the system must be already there. In [1] a novel approach to open systems has been proposed where, according to the current needs, the behaviour of any participant can be regarded as an "interface". An interface is hence intended to represent - somehow dually with respect to the standard notion of interface - part of the expected communication behaviour of the environment. Identifying a participant behaviour as interface corresponds to expecting such a behaviour to be realised by the environment rather than by an actual component of the system. Then, according to such an approach, there is actually no distinction between a closed and an open system. In particular, once two systems possess two "compatible" interfaces, they can be connected. The connecting mechanism of [1] uses suitable forwarders, dubbed "gateways", for this purpose. The gateways are automatically synthesised out of the compatible interfaces and the connection of two systems simply consists in replacing the latter by the former.

In the present paper we have provided a multiparty formalism and we have adapted the approach of [1] to it. Our calculus of multiparty sessions is like those of [12, 15], but for the use of coinduction instead of induction which is inspired by [6, 27]. As in [27] we get rid of local types, which in many calculi are similar to processes [7, 12, 15]. The syntax of global types is the coinductive version of the standard syntax [18] and the notion of projection is an extension of both the standard projection [18] and the projection given in [27]. Our global types assure lock-freedom of multiparty sessions.

A relevant feature of our formalism is that the connection operation on systems can be "lifted" to the level of global types. In [1], where systems of CFSMs were taken into account, such a lifting was done by extending the syntax of global descriptions with a *new* symbol, whose semantics is indeed the connection-by-gateways at system level. Instead in the present paper we can use the standard syntax to build the global type of the session obtained by connecting. Moreover, we have shown that the compatibility relation of [1], which requires duality, can be relaxed to a relation strongly similar to session-types' subtyping [14, 11]. Our structural preorder on processes mimics the subtyping relation between session types of [7], which is a restriction of the subtyping of [11]. This choice is justified by the fact that the subtyping of [11] allows process substitution, while the subtyping of [14] allows channel substitution, as observed in [13].

In [23] global types are build out of several local types (under certain conditions). We also aim at getting global types, the difference being that this is obtained out of the global types describing the two systems which are connected. The "dynamic" addition of participants (they can join/leave the session after it's been set up) is supported in the calculus of [19]. In that work the extension of a system is part of the global protocol. The extension operation is sort of "internalised". We take instead the standard point of view of open systems, where the possible extensions cannot be "programmed" in advance. The two approaches to the system-extension issue look orthogonal.

Both "arbiter processes" [4] and "mediums" [3] coordinate communications described by global types. A difference with the present paper is that their aim is to reduce the interactions in multiparty sessions to interactions in binary sessions. Our gateways do instead act as simple "forwarders", with the aim of connecting two multiparty systems. Nonetheless, our work could be further developed and investigated in the logical context of [4]: in the logical interpretation of multiparty sessions one could introduce a "connection-cut" corresponding to a sort of connection-by-gateways-operator. Then the good properties of the system corresponding to the proof containing the cut should be guaranteed by proving that the "connection-cut" is actually an admissible rule. The proof should consist in a "connection-cut elimination" procedure corresponding to our CN function on global types, once extended (as we claim it can be, see next Section) in order to "bypass" the use of gateways.

8 Future Works and Conclusion

The MPST framework does work fairly well for the design of closed systems, but does not possess the flexibility open systems can offer. Managing to look at global types as overall descriptions of open systems results in the possibility of a modular design of systems. From another point of view, by means of our approach one could develop systems where some participants, instead of representing actual processes, describe sort of "API calls", along the line of what some researchers refer to as Behavioural-API. Moreover, the theory we propose could be helpful also after the system implementation phase. Let us assume to have a system developed using the MPST software-development approach. After the implementation phase, one could realise that the service corresponding to a participant of the system can be more suitably provided by another system. The participant can then be safely replaced by a gateway

connection with the other system and the connection operation on global types enables to get a global view of what is going on in the resulting system.

We conjecture the completeness of our process compatibility, i.e. that the session $(\mathscr{M},\mathsf{h}) \leftrightarrow (\mathscr{M}',\mathsf{k})$ can reduce to a stuck session whenever h and k are not compatible. This could be shown by taking inspiration from the completeness proofs for subtyping of [12, 15].

The use of gateways enables us to get a "safe" systems' composition by minimally affecting the systems themselves, being just the interfaces to be modified. One could however wonder whether gateways are strictly necessary to get safe connections in our multiparty-sessions' setting. As suggested in [22], one could try to "bypass" the use of gateways by taking the interface participants out and changing some senders' and receivers' names in the other participants' "code". The following simple example shows that just a renaming would not work in general. Let us consider the following global types.

$$\mathsf{G} \;=\; \mathsf{p} \to \mathsf{h} : \ell.\mathsf{G} \qquad\qquad \mathsf{G}' \;=\; \mathsf{k} \to \mathsf{r} : \ell.\mathsf{k} \to \mathsf{s} : \ell.\mathsf{G}'$$

It is immediate to check that the multiparty sessions corresponding to G and G' are

$$\mathscr{M} \;=\; \mathsf{p} \triangleright P \mid \mathsf{h} \triangleright H \qquad\qquad \mathscr{M}' \;=\; \mathsf{k} \triangleright K \mid \mathsf{r} \triangleright R \mid \mathsf{s} \triangleright S$$

where $\quad P = \mathsf{h}!\ell.P \qquad H = \mathsf{p}?\ell.H \qquad K = \mathsf{r}!\ell.\mathsf{s}!\ell.K \qquad R = \mathsf{k}?\ell.R \qquad S = \mathsf{k}?\ell.S$

On the side of $\mathscr{M}'$ we could take K out and change some senders' and receivers' names in R and S in order they can receive the message ℓ directly from p, so obtaining $\tilde{R} = \mathsf{p}?\ell.\tilde{R}$ and $\tilde{S} = \mathsf{p}?\ell.\tilde{S}$. On the side of $\mathscr{M}$, instead, after taking out H, we could not get a sound connection by a simple renaming for the recipient h in $P = \mathsf{h}!\ell.P$, since the message ℓ should be delivered, alternately, to $\tilde{R}$ and $\tilde{S}$. A safe connection would hence imply also a modification of the "code" of P as follows: $\tilde{P} = \mathsf{r}!\ell.\mathsf{s}!\ell.\tilde{P}$. We conjecture that the function $^{\mathsf{h}\leftrightarrow\mathsf{k}}$ on global types can be redefined in such a way that, in the present example, by projecting $\mathsf{G}^{\mathsf{h}\leftrightarrow\mathsf{k}}\mathsf{G}'$ we exactly obtain $\mathsf{p} \triangleright \tilde{P} \mid \mathsf{r} \triangleright \tilde{R} \mid \mathsf{s} \triangleright \tilde{S}$. This new connection operation on global types would result in a useful tool for the modular design of systems via global types. We leave the investigation of such alternative definition of $^{\mathsf{h}\leftrightarrow\mathsf{k}}$ for future work.

The results of this paper would be more applicable accounting for asynchronous communications. In particular, a first relevant step would consist in allowing gateways to interact asynchronously. We expect the compatibility could be extended, since the subtyping for asynchronous multiparty sessions is more permissive than the subtyping for the synchronous ones [25]. Of course this extension requires care, being the subtyping of [25] undecidable, as shown in [2, 24].

The connection via gateways proposed by the authors of [1] and exploited in the present paper in a multiparty sessions setting does produce networks of systems possessing a tree-like topology. In order to get general graphs topology, it sounds natural to extend the present "single interface" connection to a "multiple interfaces" one. Such an extension, however, immediately reveals itself to be unsound: by connecting via gateways more than one pair of compatible interfaces one could obtain a deadlocked system. A very simple example for that is $\quad \mathsf{G} = \mathsf{p} \to \mathsf{h} : \ell \quad$ and $\quad \mathsf{G}' = \mathsf{k} \to \mathsf{s} : \ell$

By projection we get the systems $\quad \mathscr{M} = \mathsf{p} \triangleright \mathsf{h}!\ell \mid \mathsf{h} \triangleright \mathsf{p}?\ell \quad$ and $\quad \mathscr{M}' = \mathsf{k} \triangleright \mathsf{s}!\ell \mid \mathsf{s} \triangleright \mathsf{k}?\ell$

It is immediate to check that p and s are compatible, as well as h and k. Simultaneously connecting $\mathscr{M}$ and $\mathscr{M}'$ through both the compatible pairs (p,s) and (h,k) would result in the following deadlocked system $\qquad \mathsf{p} \triangleright \mathsf{s}?\ell.\mathsf{h}!\ell \quad | \quad \mathsf{h} \triangleright \mathsf{p}?\ell.\mathsf{k}!\ell \quad | \quad \mathsf{k} \triangleright \mathsf{h}?\ell.\mathsf{s}!\ell \quad | \quad \mathsf{s} \triangleright \mathsf{k}?\ell.\mathsf{p}!\ell$

(A similar example can be developed also in the CFSMs setting of [1]). In order to guarantee "safeness" of multiple connections, suitable requirements have hence to be devised. An adaptation to the present setting of the *interaction type system* of [9] could be investigated in future for such an aim.

Acknowledgments We are indebted to Ivan Lanese and Emilio Tuosto for many enlightening discussions on the subject of this paper. We gratefully acknowledge the anonymous referees for the interaction

through the ICE web site and for their reports. The final version of this paper strongly improved in clarity and correctness thanks to their observations and suggestions.

References

[1] Franco Barbanera, Ugo de'Liguoro & Rolf Hennicker (2018): *Global Types for Open Systems*. In: *ICE*, *EPTCS* 279, Open Publishing Association, pp. 4–20, doi:10.4204/EPTCS.279.4.

[2] Mario Bravetti, Marco Carbone & Gianluigi Zavattaro (2017): *Undecidability of Asynchronous Session Subtyping*. *Information and Computation* 256, pp. 300–320, doi:10.1016/j.ic.2017.07.010.

[3] Luís Caires & Jorge A. Pérez (2016): *Multiparty Session Types Within a Canonical Binary Theory, and Beyond*. In: *FORTE*, *LNCS* 9688, Springer, pp. 74–95, doi:10.1007/978-3-319-39570-8_6.

[4] Marco Carbone, Sam Lindley, Fabrizio Montesi, Carsten Schürmann & Philip Wadler (2016): *Coherence Generalises Duality: A Logical Explanation of Multiparty Session Types*. In: *CONCUR*, *LIPIcs* 59, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 33:1–33:15, doi:10.4230/LIPIcs.CONCUR.2016.33.

[5] Felice Cardone & Mario Coppo (2013): *Recursive Types*. In Henk Barendregt, Wil Dekkers & Richard Statman, editors: *Lambda Calculus with Types*, Perspectives in Logic, Cambridge University Press, pp. 377–576, doi:10.1017/CBO9781139032636.011.

[6] Giuseppe Castagna, Nils Gesbert & Luca Padovani (2009): *A Theory of Contracts for Web Services*. *ACM Transactions on Programming Languages and Systems* 31(5), pp. 19:1–19:61, doi:10.1145/1538917.1538920.

[7] Ilaria Castellani, Mariangiola Dezani-Ciancaglini & Paola Giannini (2019): *Reversible Sessions with Flexible Choices*. *Acta Informatica*, doi:10.1007/s00236-019-00332-y. To appear.

[8] Mario Coppo, Mariangiola Dezani-Ciancaglini, Luca Padovani & Nobuko Yoshida (2015): *A Gentle Introduction to Multiparty Asynchronous Session Types*. In: *Formal Methods for Multicore Programming*, LNCS, Springer, pp. 146–178, doi:10.1007/978-3-319-18941-3_4.

[9] Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida & Luca Padovani (2016): *Global Progress for Dynamically Interleaved Multiparty Sessions*. *Mathematical Structures in Computer Science* 26(2), pp. 238–302, doi:10.1017/S0960129514000188.

[10] Bruno Courcelle (1983): *Fundamental Properties of Infinite Trees*. *Theoretical Computer Science* 25, pp. 95–169, doi:10.1016/0304-3975(83)90059-2.

[11] Romain Demangeon & Kohei Honda (2011): *Full Abstraction in a Subtyped Pi-Calculus with Linear Types*. In: *CONCUR*, *LNCS* 6901, Springer, pp. 280–296, doi:10.1007/978-3-642-23217-6_19.

[12] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic & Nobuko Yoshida (2015): *Precise Subtyping for Synchronous Multiparty Sessions*. In: *PLACES*, *EPTCS* 203, Open Publishing Association, pp. 29–43, doi:10.4204/EPTCS.203.3.

[13] Simon Gay (2016): *Subtyping Supports Safe Session Substitution*. In: *A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday*, *LNCS* 9600, Springer, pp. 95–108, doi:10.1007/978-3-319-30936-1_5.

[14] Simon Gay & Malcolm Hole (2005): *Subtyping for Session Types in the Pi Calculus*. *Acta Informatica* 42(2/3), pp. 191–225, doi:10.1007/s00236-005-0177-z.

[15] Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, Alceste Scalas & Nobuko Yoshida (2019): *Precise Subtyping for Synchronous Multiparty Sessions*. *Journal of Logic and Algebraic Methods in Programming* 104, pp. 127–173, doi:10.1016/j.jlamp.2018.12.002.

[16] Kohei Honda, Vasco Thudichum Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Discipline for Structured Communication-Based Programming*. In: *ESOP*, *LNCS* 1381, Springer, pp. 122–138, doi:10.1007/BFb0053567.

[17] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty Asynchronous Session Types*. In: *POPL*, ACM Press, pp. 273–284, doi:10.1145/1328438.1328472.

[18] Kohei Honda, Nobuko Yoshida & Marco Carbone (2016): *Multiparty Asynchronous Session Types*. *Journal of the ACM* 63(1), p. 9, doi:10.1145/2827695.

[19] Raymond Hu & Nobuko Yoshida (2017): *Explicit Connection Actions in Multiparty Session Types*. In: *FASE*, *LNCS* 10202, Springer, pp. 116–133, doi:10.1007/978-3-662-54494-5_7.

[20] Naoki Kobayashi (2002): *A Type System for Lock-Free Processes*. *Information and Computation* 177(2), pp. 122–159, doi:10.1006/inco.2002.3171.

[21] Dexter Kozen & Alexandra Silva (2017): *Practical Coinduction*. *Mathematical Structures in Computer Science* 27(7), pp. 1132–1152, doi:10.1017/S0960129515000493.

[22] Ivan Lanese & Emilio Tuosto: *Personal Communication*.

[23] Julien Lange & Emilio Tuosto (2012): *Synthesising Choreographies from Local Session Types*. In: *CONCUR*, *LNCS* 7454, Springer, pp. 225–239, doi:10.1007/978-3-642-32940-1_17.

[24] Julien Lange & Nobuko Yoshida (2017): *On the Undecidability of Asynchronous Session Subtyping*. In: *FOSSACS*, *LNCS* 10203, Springer, pp. 441–457, doi:10.1007/978-3-662-54458-7_26.

[25] Dimitris Mostrous, Nobuko Yoshida & Kohei Honda (2009): *Global Principal Typing in Partially Commutative Asynchronous Sessions*. In: *ESOP*, *LNCS* 5502, Springer, pp. 316–332, doi:10.1007/978-3-642-00590-9_23.

[26] Benjamin C. Pierce (2002): *Types and Programming Languages*. MIT Press.

[27] Paula Severi & Mariangiola Dezani-Ciancaglini (2019): *Observational Equivalence for Multiparty Sessions*. *Fundamenta Informaticae* 167. To appear.