Composition and decomposition of multiparty sessions

Composition and Decomposition of Multiparty Sessions

Franco Barbanera[a,1], Mariangiola Dezani-Ciancaglini[b,2],
Ivan Lanese[c], Emilio Tuosto[d,3]

[a] *Dipartimento di Mat. e Inf., Università di Catania (Italy), barba@dmi.unict.it*
[b] *Dipartimento di Informatica, Università di Torino (Italy), dezani@di.unito.it*
[c] *Focus Team, University of Bologna/INRIA (Italy), ivan.lanese@gmail.com*
[d] *Gran Sasso Science Institute (Italy), emilio.tuosto@gssi.it*

Abstract

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 sessions enjoy lock-freedom.

We look at multiparty sessions as *open* systems by allowing one to *compose* multiparty sessions by transforming two of their participants into a pair of coupled *gateways*, forwarding messages between the two sessions. Gateways need to be *compatible*. We show that the session resulting from the composition can be typed, and its type can be computed from the global types of the starting sessions. As a consequence, lock-freedom is preserved by composition. Compatibility between global types is *necessary*, since systems obtained by composing sessions with incompatible global types have locks (or they are not sessions). We also define *direct composition*, which allows one to connect two global types without using gateways. Finally, we propose a *decomposition* operator, to split a global type into two, which is the left inverse of direct composition. Direct composition and decomposition on global types prepare the ground for a novel framework allowing for the modular design and implementation of distributed systems.

[1] Partially supported by the project "Piano Triennale Ricerca" DMI-Università di Catania.

[2] Partially supported by Ateneo/Compagnia di San Paolo 2016/2018 project "Mnemo-Computing - Components for Processing In Memory".

[3] Research partly supported by the EU H2020 RISE programme under the Marie Skodowska-Curie grant agreement No 778233.

1. Introduction

Distributed systems are seldom developed as independent entities. Indeed, in many cases they should be considered as open entities ready for interaction with an environment. Composition may be specified either statically, for a fixed environment, or dynamically, if the environment only becomes known upon deployment. In general, it is fairly natural to expect to connect open systems as if they were composable modules, and in doing that one 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 composition of open systems, consisting in replacing *any* two participants (one per system) - if their behaviours are "compatible" - by a pair of coupled forwarders, dubbed *gateways*, enabling the two systems to exchange messages. In this approach, the behaviour of any participant can be looked at as an *interface*. In this setting, 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).

One of the contributions of the present paper is to present a choreography formalism enabling to "lift" the composition-by-gateways approach of [1] from systems to protocol descriptions, represented as MultiParty Session Types (MPST) [23, 24]. More precisely, we define a function that, given two MPSTs and two participants (one per MPST) to be used as interfaces, computes the MPST of the composed system. This allows one to lift to the composed system all the guarantees on the soundness of communication provided by the chosen MPST formalism. We remark that our approach does not change the syntax of the chosen MPST approach, but allows one to look at it as a formalism to describe open systems (while MPST normally describe only closed systems).

While the idea of the approach is rather general, not all MPST formalisms in the literature are suitable for its application. For instance, the requirements imposed by the type system in [13] are too strong for the gateway processes to be typed, thus we could not get any guarantee on the result of the composition.

In the present paper we apply the approach in a setting as simple as possible (inspired by [34]), to highlight its features and avoid unrelated complexity. In particular, the simplicity of the calculus allows us to get rid of channels and local types. Moreover, the fact that global and local behaviours are both

represented as infinite regular trees avoids the hindering issues caused by a syntactic description of recursion.

With respect to [34], we relax the conditions imposed on global types in order to be projectable, in particular ensuring projectability of global types resulting from our composition operation (a property that does not hold in the formalism of [34]). In our formalism, typable systems are guaranteed to be lock-free [26]. Hence, the systems obtained by composing typable systems are lock-free too. Beyond this, the global type of the composed system also provides a global view of the new system. These two benefits are direct consequences of one of our main contributions: the function from the global types of the original systems to the global type of the system obtained by composition.

In order to ensure the correctness of composition, we need to require a condition of *interface compatibility* on the two participants chosen as interfaces. We show that the compatibility relation used in [1] can be relaxed to a relation closely connected to the observational preorder of [34], in turn corresponding to the subtyping relation for session types of [20, 16].

The notion of compatibility between global types is further investigated by showing it to be, not only a sufficient condition to get lock-freeness preservation under composition, but also a necessary condition.

The use of gateways enables to connect systems in a safe way by means of the minimal modification, corresponding to the transformation of interface participants into forwarders, while the other participants are unchanged. This property is quite important when composition happens dynamically. However this approach is less suitable when composition is performed statically, to support a modular design and development process. Indeed, in this case one would like to avoid the overhead due to gateways. To answer this need we define a second function which performs *direct composition* (both at the level of global types and at the level of systems), that is composition without the need for gateways. One can see direct composition as a simplification of composition by gateways, where the behaviour of gateways is internalised in the participants willing to communicate with the other system, hence gateways are no more needed. Of course, this approach requires some (limited) changes to the other participants of the systems to be composed.

We complement direct composition with an operation of *decomposition*, which is, on global types, its left inverse. Decomposition allows one to take a typed MPST and split its participants into two groups (to each group an interface participant is also added), thus creating two MPSTs. A MPST with

the original global type can be recovered by direct composition.

Direct composition and decomposition form the basis of a modular design and implementation development process, where one can take a global description, divide it into descriptions of subsystems to be implemented separately, and then safely compose the resulting implementations.

In the standard subtyping for session types supertypes have less inputs and more outputs than their subtypes [16]. In our type system we use a preorder on processes in which larger processes have less inputs than smaller ones, but the same outputs. This allows us to get: *(i)* a stronger notion of session fidelity, *(ii)* a better correspondence between composition by gateways at the level of multiparty sessions and the one at the level of global types, *(iii)* the necessity of compatibility between global types to ensure that lock-freedom is preserved under composition. Besides, we show that such a restriction does not change the class of multiparty sessions that can be typed (but may change the types assigned to them).

Outline. Sections 2, 3 and 4 respectively introduce our calculus of multiparty sessions, their global types, and prove the properties that well-typed sessions enjoy. Sections 5 and 6 define the compatibility relation and the composition via gateways for sessions and global types, respectively. The key result that sessions obtained from composition are typable (and hence lock free) is presented in Theorem 6.11. In Section 7 compatibility between global types is shown to be not only a sufficient, but also a necessary condition for lock-freedom of composition. Direct composition, allowing one to compose systems without the need for gateways, is defined in Section 8, where related properties are also discussed and proved. The decomposition operation which complements the direct composition operation is defined and investigated in Section 9. In Section 10 we discuss how all previous results change if we consider a preorder on processes mimicking the standard subtyping relation on session types, that allows larger processes to have more outputs than smaller ones. Section 11 concludes by a recap of the paper and by discussing related and future work.

Comparison with the workshop version. The present paper is a revised and extended version of [3]. The definition of the composition operation, first presented in [3], is considerably improved here, enabling to simplify and make more readable also the proofs of most of the related results. This paper includes also a number of new contributions. First, we show that compatibility

between global types is not only a sufficient, but also a necessary condition in order to get lock-freedom preservation for composition of typable multiparty sessions. Also, both the direct composition and the decomposition operations, and all the related results, are new. Lastly, in the workshop version we did not consider how the results change when the processes are compared using a preorder mimicking the standard subtyping preorder.

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 $\mathcal{M}, \mathcal{M}', \ldots$; *integers*, ranged over by $n, m, i, j, \ldots$.

Processes implement the behaviour of 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 for some i, $1 \leq i \leq n$, 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 for some i, the process reduces to P_i. The set Λ acts as an *external choice* in $\mathsf{p}?\Lambda$ and as an *internal choice* in $\mathsf{p}!\Lambda$. 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:*

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

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.*

The fact that the grammar for pre-processes has to be interpreted coinductively implies that infinite derivations are allowed. 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 having 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 [33, Section 20.2]. The existence and uniqueness of a solution follow from known results (see [14] and [9, Theorem 7.5.34]). With such an approach, 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 [27] which, without compromising 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

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

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, as well as trailing $\mathbf{0}$ processes.

A *multiparty session* is the parallel composition of pairs participant/process.

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

$$\mathcal{M} \;::=^{inductive}\; \mathsf{p} \triangleright P \;\mid\; \mathcal{M} \mid \mathcal{M}$$

and it satisfies the following conditions:
(a) in $\mathsf{p}_1 \triangleright P_1 \mid \ldots \mid \mathsf{p}_n \triangleright P_n$*, for all* i, j*,* $1 \leq i \neq j \leq n$*,* $\mathsf{p}_i \neq \mathsf{p}_j$*;*
(b) in $\mathsf{p} \triangleright P$ *we require* $\mathsf{p} \notin \mathtt{ptp}(P)$ *(we do not allow self-communication).*

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

Operational Semantics. We assume to have a structural congruence $\equiv$ on multiparty sessions, establishing that parallel composition is commutative, associative and has neutral elements $\mathsf{p} \triangleright \mathbf{0}$ for any fresh p.
It is convenient to adopt the notation $\mathsf{p} \triangleright P \in \mathcal{M}$ for $\mathcal{M} \equiv \mathsf{p} \triangleright P \mid \mathcal{M}'$.

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:*

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

Rule [COMM] makes the communication possible: participant p sends message ℓ to participant 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 $\mathcal{M} \xrightarrow{\lambda} \mathcal{M}'$ as shorthand for $\mathcal{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathcal{M}'$. 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 p, q, and h interacting according to the following protocol. Participant p keeps on sending text messages to q, which has to deliver them to h. After a message has been sent by p, the next one can be sent only if the previous has been received by h and its propriety of language ascertained, i.e if it does not contain, say, rude or offensive words. Participant h acknowledges to q the propriety of language of a received text by means of the message

ack. In such a case q sends to p an *ok* message so that 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 step 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 step 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 p of that.

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

$$
\begin{array}{ll}
P = \mathsf{q}!\mathit{text}.P_1 & P_1 = \mathsf{q}?\{\mathit{ok}.P, \mathit{notyet}.P_1, \mathit{stop}\} \\
Q = \mathsf{p}?\mathit{text}.\mathsf{h}!\mathit{text}.Q_1 & Q_1 = \mathsf{h}?\{\mathit{ack}.\mathsf{p}!\mathit{ok}.Q, \\
 & \qquad \mathit{nack}.\mathsf{p}!\mathit{notyet}.\mathsf{h}!\mathit{transf}.Q_1, \\
 & \qquad \mathit{stop}.\mathsf{p}!\mathit{stop}\} \\
H = \mathsf{q}?\mathit{text}.H_1 & H_1 = \mathsf{q}!\{\mathit{ack}.H, \mathit{nack}.\mathsf{q}?\mathit{transf}.H_1, \mathit{stop}\}
\end{array}
$$

$\diamond$

We end this section by defining the property of lock-freedom for multiparty sessions as in [26, 32]. Lock-freedom guarantees progress for each participant (and hence deadlock-freedom). In other words, each participant ready to communicate will eventually find her partner exposing a dual communication action. Recall that, by structural congruence, $\mathsf{p} \triangleright \mathbf{0}$ is the neutral element of parallel composition, for any fresh p.

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

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

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

To get a stronger lock-freedom assuring that a communication involving p will occur in all reductions starting from $\mathsf{p} \triangleright P \mid \mathcal{M}$ when $P \neq \mathbf{0}$ we need the following fairness assumption:

> *in a session, any pair of participants ready to communicate will exchange a message after a finite number of steps*

In this way the multiparty session $\mathcal{M} = \mathsf{p} \triangleright P \mid \mathsf{q} \triangleright Q \mid \mathsf{r} \triangleright \mathsf{s}!\ell \mid \mathsf{s} \triangleright \mathsf{r}?\ell$ with $P = \mathsf{q}!\ell'.P$ and $Q = \mathsf{p}?\ell'.Q$ will not have the infinite reduction

$$\mathcal{M} \xrightarrow{\mathsf{p}\ell'\mathsf{q}} \mathcal{M} \xrightarrow{\mathsf{p}\ell'\mathsf{q}} \cdots$$

where r and s never communicate.
This assumption can be implemented, e.g., by a maximally parallel reduction in which at each step all possible communications are done [34].
Notice that we do not require participants to choose outputs in a fair way, since our type system excludes starvation due to the choice of labels, which is exemplified below.

Example 2.6. The multiparty session $\mathcal{M} = \mathsf{p} \triangleright P \mid \mathsf{q} \triangleright Q \mid \mathsf{r} \triangleright R$ with $P = \mathsf{q}!\{\ell_1.\mathsf{r}?\ell_3, \ell_2.P\}$, $Q = \mathsf{p}?\{\ell_1, \ell_2.Q\}$, and $R = \mathsf{p}!\ell_3$, has the infinite reduction

$$\mathcal{M} \xrightarrow{\mathsf{p}\ell_2\mathsf{q}} \mathcal{M} \xrightarrow{\mathsf{p}\ell_2\mathsf{q}} \cdots$$

in which participant r can never send her message. Notice that this multiparty session is rejected by our type system, see Remark 3.11. ◇

3. Global Types and Typing System

The behaviour of multiparty sessions can be disciplined by means of types. Global types describe the whole conversation scenarios of multiparty sessions. As in [34] we directly assign global types to multiparty sessions without the usual detour around session types and subtyping [23, 24].

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 i, $1 \leq i \leq n$, (and q must receive it) 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 $msg(\{\ell_i.\mathsf{G}_i \mid {}_{1\leq i\leq n}\}) = \{\ell_i \mid 1 \leq i \leq n\}$. 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:*

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

where $\mathsf{p} \neq \mathsf{q}$ *and all messages in* $msg(\Gamma)$ *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 $\mathtt{End}$.

(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 representations 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 those of processes and sessions. 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 assume that $\uplus$ has the highest precedence, i.e. $\mathsf{p} \to \mathsf{q} : \Gamma_1 \uplus \Gamma_2$ means $\mathsf{p} \to \mathsf{q} : (\Gamma_1 \uplus \Gamma_2)$. We shall omit curly brackets in choices with only one branch, as well as trailing $\mathtt{End}$ terms.

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 $\mathtt{End}$. 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 (ϵ is the empty sequence):

$$\begin{array}{lcl} \mathit{paths}(\mathtt{End}) & = & \{\epsilon\} \\ \mathit{paths}(\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}) & = & \bigcup_{1 \leq i \leq n} \{(\mathsf{p} \to \mathsf{q})\, \ell_i\, \rho \mid \rho \in \mathit{paths}(\mathsf{G}_i)\} \end{array}$$

Note that every infinite path of a global type has infinitely many occurrences of '$\to$'. We use "$\mathsf{p} \notin \rho$" as a shorthand for "p *does not occur in* ρ".
The function *length*, which returns the number of "interactions" in a path, is defined as expected:

$$\begin{array}{lcl} \mathit{length}(\epsilon) & = & 0 \\ \mathit{length}((\mathsf{p} \to \mathsf{q})\, \ell_i\, \rho) & = & 1 + \mathit{length}(\rho) \end{array}$$

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

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

Examples of paths in G are
$\rho_1 = (\mathsf{p} \to \mathsf{q})\mathit{text}(\mathsf{q} \to \mathsf{h})\mathit{text}(\mathsf{h} \to \mathsf{q})\mathit{stop}(\mathsf{q} \to \mathsf{p})\mathit{stop}$ and
$\rho_2 = (\mathsf{p} \to \mathsf{q})\mathit{text}(\mathsf{q} \to \mathsf{h})\mathit{text}(\mathsf{h} \to \mathsf{q})\mathit{ack}(\mathsf{q} \to \mathsf{p})\mathit{ok}\rho_2$.
We get $\mathit{length}(\rho_1) = 4$ and $\mathit{length}(\rho_2) = \infty$. ◇

Usually, as in [23, 24], projection of global types onto participants produces session types, and session types are assigned to processes by a type system. The simple shape of our messages, instead, allows us to define a projection of global types onto participants producing processes directly.

The projection of a global type onto a participant, if defined, returns the process that the participant should run to comply with the protocol specified by the global type. If the global type begins with a message 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 r. However, such a communication may be part of a choice, and since r is not part of it, she is not immediately aware of which branch is taken. Then, there are two possibilities: either r will behave in the same way in all the branches, hence

she has no need to know which one has been taken, or she will discover later on which branch has been taken, by receiving distinct messages in each branch. In the first case, the projections on r of all the branches must be the same, and this projection defines her behaviour. In the second case, the projections yield input processes receiving from the same sender, and we can allow the process of r to combine all these processes, proviso the messages are all different.

Definition 3.3 (Projection). *Given a participant* p*, we coinductively define the partial function* $\_\upharpoonright_{\mathsf{p}}$ *on global types* G *as follows:*

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

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 [17, 21], where processes are replaced by local types.

As mentioned above, if p is not involved in the first communication of G, then in all branches the process of participant p must either behave in the same way or be an input from the same sender of different messages, so that p can understand which branch was chosen.

Example 3.4. 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 $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \{\ell_1.\mathsf{r} \to \mathsf{p} : \ell_3, \ell_2.\mathsf{G}'\}$ is projectable: $\mathsf{G}'\upharpoonright_{\mathsf{p}} = P$, $\mathsf{G}'\upharpoonright_{\mathsf{q}} = Q$, $\mathsf{G}'\upharpoonright_{\mathsf{r}} = R$, where P, Q and R are defined in Example 2.6. 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. $\diamond$

In order to assure lock-freedom by typing we require each participant to occur in all the paths from the root, and that for each participant the first occurrences in the various paths are at bounded depth. This is formalised by requiring the *depth* of each participant, defined as below, to be finite.

Definition 3.5 (Depth). *Let*

$$depth(\rho, \mathsf{p}) = \begin{cases} length(\rho_1) & \text{if } \rho = \rho_1\,(\mathsf{q} \to \mathsf{r})\,\ell\,\rho_2,\ \mathsf{p} \notin \rho_1 \text{ and } \mathsf{p} \in \{\mathsf{q}, \mathsf{r}\} \\ 0 & \text{if } \mathsf{p} \notin \rho \end{cases}$$

We then define $depth : \mathsf{G} \times \mathtt{ptg}(\mathsf{G}) \to \mathbb{N} \cup \{\infty\}$ *by*

$$depth(\mathsf{G}, \mathsf{p}) = sup\{depth(\rho, \mathsf{p}) \mid \rho \in paths(\mathsf{G})\}$$

Example 3.6. Let G be as in Example 3.2; then $depth(\mathsf{G}, \mathsf{p}) = depth(\mathsf{G}, \mathsf{q}) = 0$, and $depth(\mathsf{G}, \mathsf{h}) = 1$.

Let G' be as in Example 3.4, then $depth(\mathsf{G}', \mathsf{r}) = \infty$. ◇

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

Example 3.8. The global type G of Example 3.2 is well formed, while the global type G' of Example 3.4 is not well formed, since its depth is not bounded. ◇

In the following, **we will only consider well-formed global types**.

To type multiparty sessions we use the preorder $\leqslant$ on processes defined below and inspired by the subtyping of [11].

Definition 3.9 (Structural Preorder). *We define the* structural preorder on processes*,* $P \leqslant Q$*, by coinduction:*

$$[\textsc{sub-}\mathbf{0}]\quad \mathbf{0} \leqslant \mathbf{0} \qquad\qquad \frac{P_i \leqslant Q_i \quad \forall 1 \leq i \leq n}{\mathsf{p}!\{\ell_i.P_i \mid {\scriptstyle 1 \leq i \leq n}\} \leqslant \mathsf{p}!\{\ell_i.Q_i \mid {\scriptstyle 1 \leq i \leq n}\}}\ [\textsc{sub-out}]$$

$$\frac{P_i \leqslant Q_i \quad \forall 1 \leq i \leq n}{\mathsf{p}?(\{\ell_i.P_i \mid {\scriptstyle 1 \leq i \leq n}\} \uplus \Lambda) \leqslant \mathsf{p}?\{\ell_i.Q_i \mid {\scriptstyle 1 \leq i \leq n}\}}\ [\textsc{sub-in}]$$

The double-line in rules indicates that the rules are interpreted *coinductively*. Rule [SUB-IN] allows larger processes to offer fewer inputs than smaller ones, 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. Usually, subtyping relations for output allow more branches in the supertype. We discuss this option in Section 10.

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

Definition 3.10 (Typing System). *The only typing rule is:*

$$\frac{\forall \mathsf{p} \in \mathtt{pts}(\mathcal{M}).\ \left(\ \mathsf{p} \triangleright P \in \mathcal{M} \implies P \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}\ \right)}{\vdash \mathcal{M} : \mathsf{G}}\ [\text{T-SESS}]$$

A session $\mathcal{M}$ is well typed *if there exists G such that $\vdash \mathcal{M} : \mathsf{G}$.*

The rule above 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 same global type. Note that [T-SESS] is not coinductive since a multiparty session is a finite parallel composition of (possibly infinite) processes; hence coinduction is necessary only for the subtyping relation.
The quantification $\forall \mathsf{p} \in \mathtt{pts}(\mathcal{M})$ (instead of a quantification over $\mathsf{p} \triangleright P$) in the premise makes [T-SESS] finitary. Indeed, $\mathsf{p} \triangleright \mathbf{0} \in \mathcal{M}$ for all $\mathsf{p} \notin \mathtt{pts}(\mathcal{M})$, but no check is needed on those participants. Remark that $\vdash \mathcal{M} : \mathsf{G}$ implies $\mathtt{ptg}(\mathsf{G}) \subseteq \mathtt{pts}(\mathcal{M})$. In fact if $\mathsf{p} \triangleright P \in \mathcal{M}$ and $P \neq \mathbf{0}$ and $\mathsf{p} \notin \mathtt{ptg}(\mathsf{G})$, then $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathbf{0}$ and $P \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$ cannot hold.

Remark 3.11. Since the projections of G' as defined in Example 3.4 are exactly the processes of the multiparty session in Example 2.6 and G' is not well formed, our type system rejects the multiparty session in Example 2.6.

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 $\vdash \mathcal{M} : \mathsf{G}$ *and* $\mathsf{p} \triangleright P \in \mathcal{M}$*, then* $P \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$.

Lemma 4.2 (Canonical Form Lemma). *If* $\vdash \mathcal{M} : \mathsf{G}$ *and* $\mathsf{p} \in \mathtt{ptg}(\mathsf{G})$*, then there is* $\mathsf{p} \triangleright P \in \mathcal{M}$ *and* $P \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$.

To formalise the properties of Subject Reduction and Session Fidelity [23, 24], we use the standard LTS for global types given below.

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

$$[\textsc{ecomm}] \quad \mathsf{p} \to \mathsf{q} : \ell.\mathsf{G} \uplus \Gamma \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}$$

$$[\textsc{icomm}] \quad \frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}_i' \quad \forall 1 \leq i \leq n \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid {\scriptstyle 1 \leq i \leq n}\} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i' \mid {\scriptstyle 1 \leq i \leq n}\}}$$

Rule [Icomm] makes sense since, in a projectable global type $\mathsf{r} \to \mathsf{s} : \Gamma$, behaviours involving a participant p doing an output as first action and different from r are the same in all branches. Hence they are independent from the choice of r, and may be executed before it.

In the remaining of this section we show the main properties of our type system, i.e. Subject Reduction, Sessione Fidelity and Lock-Freedom. We start with two lemmas. The first lemma says that the depths of participants not occurring in the root of a global type decrease along the branches of the tree. The second lemma relates projections and reductions of global types.

Lemma 4.4. *If* $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ *and* $\mathsf{r} \in \mathtt{ptg}(\mathsf{G}) \setminus \{\mathsf{p},\mathsf{q}\}$*, then* $\mathit{depth}(\mathsf{G},\mathsf{r}) > \mathit{depth}(\mathsf{G}',\mathsf{r})$ *for all* $\ell.\mathsf{G}' \in \Gamma$.

Proof. All paths of G are of the shape $(\mathsf{p} \to \mathsf{q})\,\ell\,\rho$, where ρ is a path of G'. This gives $\mathit{depth}(\mathsf{G},\mathsf{r}) > \mathit{depth}(\mathsf{G}',\mathsf{r})$. □

Lemma 4.5 (Key Lemma).

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

(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 $w = depth(\mathsf{G}, \mathsf{p})$.
If $w = 0$, then $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$. Hence, by definition of projection (Definition 3.3), we have $msg(\Lambda) = msg(\Lambda') = msg(\Gamma)$ and, for all $\ell.\mathsf{G}^\ell \in \Gamma$, $\ell.\mathsf{G}^\ell\upharpoonright_{\mathsf{p}} \in \Lambda$ and $\ell.\mathsf{G}^\ell\upharpoonright_{\mathsf{q}} \in \Lambda'$. Moreover $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}^\ell$ using rule [ECOMM].
If $w > 0$, then $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}$ and $\mathsf{p} \notin \{\mathsf{r}, \mathsf{s}\}$. From $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$ we get $\mathsf{q} \notin \{\mathsf{r}, \mathsf{s}\}$. Moreover $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ and $\mathsf{G}_i\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$ for all i, $1 \leq i \leq n$, by definition of projection. By Lemma 4.4 $w > depth(\mathsf{G}_i, \mathsf{p})$ for all i, $1 \leq i \leq n$. By the induction hypothesis, $msg(\Lambda) = msg(\Lambda')$. 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, $1 \leq i \leq n$, and all $\ell \in msg(\Lambda)$. We get $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}^\ell$ using rule [ICOMM], where $\mathsf{G}^\ell = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i^\ell \mid 1 \leq i \leq n\}$.
(ii). The proof is by induction on $depth(\mathsf{G}, \mathsf{p})$ 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, $1 \leq i \leq n$. By definition of projection $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{G}_1\upharpoonright_{\mathsf{p}}$ and $\mathsf{G}_i\upharpoonright_{\mathsf{q}} = \mathsf{G}_1\upharpoonright_{\mathsf{q}}$ for all i, $1 \leq i \leq n$. 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 4.6 (Subject Reduction).
If $\vdash \mathcal{M} : \mathsf{G}$ *and* $\mathcal{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathcal{M}'$, *then* $\mathsf{G} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathsf{G}'$ *and* $\vdash \mathcal{M}' : \mathsf{G}'$.

Proof. If $\mathcal{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathcal{M}'$, then $\mathsf{p} \triangleright \mathsf{q}!(\ell.P \uplus \Lambda) \in \mathcal{M}$, $\mathsf{q} \triangleright \mathsf{p}?(\ell.Q \uplus \Lambda') \in \mathcal{M}$ and $\mathsf{p} \triangleright P \in \mathcal{M}'$, $\mathsf{q} \triangleright Q \in \mathcal{M}'$. Moreover $\mathsf{r} \triangleright R \in \mathcal{M}$ iff $\mathsf{r} \triangleright R \in \mathcal{M}'$ for all $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$. Since $\vdash \mathcal{M} : \mathsf{G}$, we have that $\mathsf{q}!(\ell.P \uplus \Lambda) \leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$, and $\mathsf{p}?(\ell.Q \uplus \Lambda') \leqslant \mathsf{G}\upharpoonright_{\mathsf{q}}$, and for all $\mathsf{r} \triangleright R \in \mathcal{M}$ such that $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$ we have $R \leqslant \mathsf{G}\upharpoonright_{\mathsf{r}}$ by Lemma 4.1. 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 4.5(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 $\mathsf{G}\upharpoonright_{\mathsf{r}} \leqslant \mathsf{G}'\upharpoonright_{\mathsf{r}}$ for each $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$ and $\mathsf{r} \triangleright R \in \mathcal{M}$ by induction on $depth(\mathsf{G}, \mathsf{r})$ 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 3.3 either $\mathsf{G}\upharpoonright_{\mathsf{r}} = \mathsf{G}'\upharpoonright_{\mathsf{r}}$ or $\mathsf{G}\upharpoonright_{\mathsf{r}} \leqslant \mathsf{G}'\upharpoonright_{\mathsf{r}}$. For rule [ICOMM] $\mathsf{G}_i\upharpoonright_{\mathsf{r}} \leqslant \mathsf{G}_i'\upharpoonright_{\mathsf{r}}$ for all i, $1 \leq i \leq n$, by the induction hypothesis. This implies $\mathsf{G}\upharpoonright_{\mathsf{r}} \leqslant \mathsf{G}'\upharpoonright_{\mathsf{r}}$. We conclude $\vdash \mathcal{M}' : \mathsf{G}'$. □

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

Theorem 4.7 (Session Fidelity). *Let* $\vdash \mathcal{M} : \mathsf{G}$.

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

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

Proof. (i). It is the Subject Reduction Theorem.
(ii). By Lemma 4.5(ii), $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ and $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?\Lambda'$ and $\ell \in msg(\Lambda) = msg(\Lambda')$. By Lemma 4.5(i) $\ell.\mathsf{G}'\upharpoonright_{\mathsf{p}} \in \Lambda$ and $\ell.\mathsf{G}'\upharpoonright_{\mathsf{q}} \in \Lambda'$. By Lemma 4.2 $\mathsf{p} \triangleright P \in \mathcal{M}$ and $\mathsf{q} \triangleright Q \in \mathcal{M}$ 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 $\{\ell\} \cup msg(\Lambda_1) = msg(\Lambda)$ and $P' \leqslant \mathsf{G}'\upharpoonright_{\mathsf{p}}$,
- $Q = \mathsf{p}?(\ell.Q' \uplus \Lambda_2)$ with $\{\ell\} \cup msg(\Lambda_2) \supseteq msg(\Lambda')$ and $Q' \leqslant \mathsf{G}'\upharpoonright_{\mathsf{q}}$.

Hence we have $\mathcal{M} \xrightarrow{\mathsf{p}\ell\mathsf{q}} \mathcal{M}'$ with $\mathsf{p} \triangleright P' \in \mathcal{M}'$, $\mathsf{q} \triangleright Q' \in \mathcal{M}'$ and $\mathsf{r} \triangleright R \in \mathcal{M}$ iff $\mathsf{r} \triangleright R \in \mathcal{M}'$ for all $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$. We conclude $\vdash \mathcal{M}' : \mathsf{G}'$. □

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. Both follow from Session Fidelity, and the latter uses also Lemma 4.4.

Theorem 4.8 (Lock-Freedom). *If* $\mathcal{M}$ *is well typed, then* $\mathcal{M}$ *is lock-free.*

Proof. Let $\hat{\mathcal{M}}$ be the session obtained after a sequence of reductions from $\mathcal{M}$ as in conditions (a) and (b) of Definition 2.5. By Subject Reduction $\hat{\mathcal{M}}$ can be typed. Let G be a type for $\hat{\mathcal{M}}$. If $\hat{\mathcal{M}} \not\equiv \mathsf{p} \triangleright \mathbf{0}$, then $\mathsf{G} \neq \mathtt{End}$.
Let $\mathsf{G} = \mathsf{q} \to \mathsf{r} : \Gamma$. By rule [ECOMM], $\mathsf{G} \xrightarrow{\mathsf{q}\ell\mathsf{r}} \mathsf{G}'$ for some $\ell.\mathsf{G}' \in \Gamma$, and this implies $\hat{\mathcal{M}} \xrightarrow{\mathsf{q}\ell\mathsf{r}} \mathcal{M}'$ by Theorem 4.7(ii). This shows condition (a) of Definition 2.5.
The proof of condition (b) of Definition 2.5 is by induction on $w = depth(\mathsf{G}, \mathsf{p})$. If $w = 0$ 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]. Then $\hat{\mathcal{M}} \xrightarrow{\lambda} \mathcal{M}''$ by Theorem 4.7(ii). If $w > 0$ then $\mathsf{G} = \mathsf{q} \to \mathsf{r} : \Gamma$ with $\mathsf{p} \notin \{\mathsf{q}, \mathsf{r}\}$ and $\mathsf{G} \xrightarrow{\mathsf{q}\ell\mathsf{r}} \mathsf{G}^\ell$ for all $\ell.\mathsf{G}^\ell \in \Gamma$ by rule [ECOMM]. By Lemma 4.4 $depth(\mathsf{G}, \mathsf{p}) > depth(\mathsf{G}^\ell, \mathsf{p})$ and induction applies. □

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

5. Composition of Multiparty-Sessions via Gateways

Two multiparty sessions can be *composed via gateways* when they possess two *compatible* participants, i.e. participants that offer communications which can be paired. Hence, the two participants can 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 in these examples 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' name 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 "AN" the operation abstracting from the participants' name inside processes. So, in our example we would get

$$\begin{array}{lcl} \text{AN}(H) & = & \circ?\mathit{text}.\text{AN}(H_1) \\ \text{AN}(H_1) & = & \circ!\{\mathit{ack}.\text{AN}(H), \mathit{nack}.\circ?\mathit{transf}.\text{AN}(H_1), \mathit{stop}\} \end{array}$$

where $\circ$ stands for an abstracted participant name.
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. Assume that such a system is formed by participants k, r and s interacting according to 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

$$\mathcal{M}' = \mathsf{r} \triangleright R \mid \mathsf{s} \triangleright S \mid \mathsf{k} \triangleright K_{\mathsf{r}}$$

where

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

The "behaviour as interface" of participant k corresponds to

$$\begin{array}{rcl} \textsc{an}(K_{\mathsf{r}}) = \textsc{an}(K_{\mathsf{s}}) & = & \circ!\mathit{text}.\textsc{an}(K'_{\mathsf{r}}) \\ \textsc{an}(K'_{\mathsf{r}}) = \textsc{an}(K'_{\mathsf{s}}) & = & \circ?\{\mathit{ack}.\textsc{an}(K_{\mathsf{r}}), \mathit{nack}.\circ!\mathit{transf}.\textsc{an}(K'_{\mathsf{r}})\} \end{array}$$

Notice that the mapping $\textsc{an}$ equates K_{r} and K_{s}, i.e. $\textsc{an}(K_{\mathsf{r}}) = \textsc{an}(K_{\mathsf{s}})$. The interactions "offered" and "requested" by $\textsc{an}(H)$ and $\textsc{an}(K_{\mathsf{r}})$ do not precisely match each other, that is $\overline{\textsc{an}(H)} \neq \textsc{an}(K_{\mathsf{r}})$ (where $\overline{(\cdot)}$ is the standard syntactic duality function replacing '!' by '?' and vice versa [22]). 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 such a message never arrives.

In the following definition, instead of explicitly introducing the "$\textsc{an}$" function, we simply formalise the compatibility relation in such a way that two processes are compatible (as interfaces) whenever they offer dual communications to *arbitrary* participants, and, for each communication, the set of input labels is a subset of the set of output labels.

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

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

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. Process compatibility is similar to, but simpler than, the subtyping relation defined in [20]. Therefore one can easily adapt the algorithm for subtyping in [20] so to check process compatibility.
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 $P \leftrightarrow \mathsf{p}?\Lambda$ *and* $P \leftrightarrow \mathsf{p}?\Lambda'$ *and* $\Lambda \cap \Lambda' = \emptyset$*, then* $P \leftrightarrow \mathsf{p}?(\Lambda \uplus \Lambda')$*.*

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

As 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 $\mathcal{M}$ and $\mathcal{M}'$ enables to connect them by transforming H and K into a pair of gateways. This means that each message received by H is immediately sent to K, each message sent by H needs to be first received by K, and similarly for what concerns K. Hence, we define below a function $\mathsf{gw}(P, \mathsf{h})$ which transforms an arbitrary process P (representing the behaviour of K above) not containing a fixed participant h into a gateway towards h, namely a process which:

1. sends to h each message received in P;
2. receives from h each message to be sent in P.

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

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

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 it is a process.*

Proof. The proof is by coinduction on P and by cases on its shape. The case $P = \mathbf{0}$ is trivial. If $P = \mathsf{p}?\{\ell_i.P_i \mid {\scriptstyle 1\leq i\leq n}\}$ then

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

For each i, $1 \leq i \leq n$, by coinduction $\mathsf{gw}(P_i,\mathsf{h})$ is defined and it is a process, since $\mathsf{h} \notin \mathtt{ptp}(P_i)$. Hence, by definition also $\mathsf{gw}(\mathsf{p}?\{\ell_i.P_i \mid {\scriptstyle 1\leq i\leq n}\},\mathsf{h})$ is a process. The proof for the case where P is an output process is similar. □

The gateway process construction is monotone with respect to the structural preorder. This property is key to get Theorem 6.11 (ensuring that composition via gateways preserves typability) and it essentially relies on the fact that larger processes offer the same output messages than smaller ones.

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. The proof is by coinduction on the derivation of $P \leqslant Q$. We only consider the case of input processes, the proof for output processes is similar and simpler, the one for $\mathbf{0}$ is trivial.
If $P = \mathsf{p}?\{\ell_i.P_i \mid {\scriptstyle 1\leq i\leq n}\}$ and $Q = \mathsf{p}?\{\ell_i.Q_i \mid {\scriptstyle 1\leq i\leq m}\}$ with $m \leq n$, then

$$\begin{aligned}\mathsf{gw}(P,\mathsf{h}) =& \mathsf{p}?\{\ell_i.\mathsf{h}!\ell_i.\mathsf{gw}(P_i,\mathsf{h}) \mid {\scriptstyle 1\leq i\leq n}\} \text{ and}\\ \mathsf{gw}(Q,\mathsf{h}) =& \mathsf{p}?\{\ell_i.\mathsf{h}!\ell_i.\mathsf{gw}(Q_i,\mathsf{h}) \mid {\scriptstyle 1\leq i\leq m}\}\end{aligned}$$

From $P \leqslant Q$ we get $P_i \leqslant Q_i$ for all i, $1 \leq i \leq m$. 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, $1 \leq i \leq m$. Hence $\mathsf{gw}(P,\mathsf{h}) \leqslant \mathsf{gw}(Q,\mathsf{h})$, by definition of $\leqslant$ (Definition 3.9). □

The following relationship between compatibility and structural preorder of processes will be essential in the proof of one of our main results (Theorem 6.11).

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

Proof. Let us assume

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

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

Essentially the result above shows that compatibility is upward-closed with respect to the structural preorder. However, it is not downward-closed. For example $\mathsf{p}!\ell \leftrightarrow \mathsf{q}?\ell$ and $\mathsf{q}?\{\ell, \ell'\} \leqslant \mathsf{q}?\ell$, but $\mathsf{p}!\ell \leftrightarrow \mathsf{q}?\{\ell, \ell'\}$ is false.

The formal definition of composition of multiparty sessions via gateways is based on the notion of process compatibility (Definition 5.1) and on the transformation of a process into a gateway (Definition 5.3).

Definition 5.7 (Compatible Multiparty Sessions).
Two multiparty sessions $\mathcal{M}$, $\mathcal{M}'$ are compatible via the participants h and k *if* $\mathtt{pts}(\mathcal{M}) \cap \mathtt{pts}(\mathcal{M}') = \emptyset$ *and* $\mathsf{h} \triangleright H \in \mathcal{M}$ *and* $\mathsf{k} \triangleright K \in \mathcal{M}'$ *with* $H \leftrightarrow K$. *We write* $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ *when $\mathcal{M}$, $\mathcal{M}'$ are compatible via* h *and* k.

Definition 5.8 (Composition Via Gateways of Multiparty Sessions).
Let $\mathcal{M} \equiv \mathcal{M}_1 \mid \mathsf{h} \triangleright H$, $\mathcal{M}' \equiv \mathcal{M}'_1 \mid \mathsf{k} \triangleright K$ *and* $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$. *We define* $\mathcal{M} \overset{\mathsf{h}}{\leftrightarrow\!\ast} \mathcal{M}'$, *the composition of $\mathcal{M}$ and $\mathcal{M}'$ via gateways, through* h *and* k, *by*

$$\mathcal{M} \overset{\mathsf{h}}{\leftrightarrow\!\ast} \mathcal{M}' = \mathcal{M}_1 \mid \mathcal{M}'_1 \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h})$$

Example 5.9. Consider $\mathcal{M}$ of Example 2.4 and $\mathcal{M}'$ defined on page 19. It is not difficult to check that

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

where

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

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

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

◇

We have shown above how to compose two multiparty sessions, but we have not provided any guarantee on the behaviour of the resulting multiparty session. In the next section we prove that lock-freedom is preserved under session composition via gateways. To show this, we define an operator taking the global types of two sessions and the name of a participant in each of them (the two participants need to be compatible) and building a type for the composed multiparty session.
Notice that preservation of lock-freedom cannot be inferred from the results of [2], where lock-freedom was not taken into account.

6. Composition of Global Types via Gateways

The composition defined in the previous section can be shown to be lock-freedom preserving by means of Theorem 4.8. In fact, it is possible to lift the construction in Definition 5.8 to the level of global types. We start by defining the compatibility of global types via two participants, which mimics the compatibility of multiparty sessions via two participants

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

It is easy to verify that $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$ implies

$$\mathsf{h} \in \mathtt{ptg}(\mathsf{G}) \text{ if, and only if, } \mathsf{k} \in \mathtt{ptg}(\mathsf{G}').$$

The compatibility of typed sessions implies the compatibility of the corresponding global types.

Lemma 6.2. *If* $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ *and* $\vdash \mathcal{M} : \mathsf{G}$ *and* $\vdash \mathcal{M}' : \mathsf{G}'$, *then*

$$(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$$

Proof. The typings $\vdash \mathcal{M} : \mathsf{G}$ and $\vdash \mathcal{M}' : \mathsf{G}'$ imply $\mathtt{ptg}(\mathsf{G}) \subseteq \mathtt{pts}(\mathcal{M})$ and $\mathtt{ptg}(\mathsf{G}') \subseteq \mathtt{pts}(\mathcal{M}')$ as observed after Definition 3.10. Therefore we get that $\mathtt{pts}(\mathcal{M}) \cap \mathtt{pts}(\mathcal{M}') = \emptyset$ implies $\mathtt{ptg}(\mathsf{G}) \cap \mathtt{ptg}(\mathsf{G}') = \emptyset$. Let $\mathsf{h} \triangleright H \in \mathcal{M}$ and $\mathsf{k} \triangleright K \in \mathcal{M}'$. By the Inversion Lemma (Lemma 4.1) from $\vdash \mathcal{M} : \mathsf{G}$ we get $H \leqslant \mathsf{G}\upharpoonright_{\mathsf{h}}$ and from $\vdash \mathcal{M}' : \mathsf{G}'$ we get $K \leqslant \mathsf{G}'\upharpoonright_{\mathsf{k}}$. From $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ we get $H \leftrightarrow K$ by Definition 5.7. Lemma 5.6 implies $\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$. □

It is worth noticing that $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$ and $\vdash \mathcal{M} : \mathsf{G}$ and $\vdash \mathcal{M}' : \mathsf{G}'$ do not imply $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$, as shown in the following example.

Example 6.3. Take $\mathcal{M} = \mathsf{p} \triangleright \mathsf{h}?\ell \mid \mathsf{h} \triangleright \mathsf{p}!\ell$, $\mathcal{M}' = \mathsf{q} \triangleright \mathsf{k}!\ell \mid \mathsf{k} \triangleright \mathsf{q}?\{\ell, \ell'\}$, $\mathsf{G} = \mathsf{h} \to \mathsf{p} : \ell$, $\mathsf{G}' = \mathsf{q} \to \mathsf{k} : \ell$. We get $\mathsf{p}!\ell \leftrightarrow \mathsf{q}?\ell$, but $\mathsf{p}!\ell \leftrightarrow \mathsf{q}?\{\ell, \ell'\}$ does not hold. ◇

We are now ready to define the composition of global types via gateways. To avoid cumbersome parentheses, in the following we assume

$$\begin{gathered}
\star \to \star : \Gamma \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \star \to \star : \Gamma' \text{ reads } (\star \to \star : \Gamma) \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} (\star \to \star : \Gamma') \\
\star \to \star : \Gamma \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G} \text{ reads } (\star \to \star : \Gamma) \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G} \\
\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \star \to \star : \Gamma \text{ reads } \mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} (\star \to \star : \Gamma) \\
\ell.\star \to \star : \ell.\star \to \star : \ell.\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}' \text{ reads } \ell.\star \to \star : \ell.\star \to \star : \ell.(\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}') \\
\ell.\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}' \text{ reads } \ell.(\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}')
\end{gathered}$$

where $\star$ stands for arbitrary, possibly different participant names.

Definition 6.4 (Composition of Global Types via Gateways). *Let* $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$. *We define*

$$\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}'$$

coinductively by the clauses of Figure 1, 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) $\mathtt{End} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G} = \mathsf{G}$

(2) $\mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{G}_i \mid 1 \le i \le n\} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{k} \to \mathsf{s} : \{\ell_i.\mathsf{G}'_i \mid 1 \le i \le n\} \uplus \Gamma =$
$\mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{h} \to \mathsf{k} : \ell_i.\mathsf{k} \to \mathsf{s} : \ell_i.\mathsf{G}_i \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}'_i \mid 1 \le i \le n\}$

(3) $\mathsf{h} \to \mathsf{p} : \{\ell_i.\mathsf{G}_i \mid 1 \le i \le n\} \uplus \Gamma \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{G}'_i \mid 1 \le i \le n\} =$
$\mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{k} \to \mathsf{h} : \ell_i.\mathsf{h} \to \mathsf{p} : \ell_i.\mathsf{G}_i \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G}'_i \mid 1 \le i \le n\}$

(4) $\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid 1 \le i \le n\} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G} \mathbin{{}^{\mathsf{k}}\!\!\not\leftrightarrow^{\mathsf{h}}} \mathsf{G}_i \mid 1 \le i \le n\}$

(5) $\mathsf{G} \mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}} \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1 \le i \le n\} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mathbin{{}^{\mathsf{k}}\!\!\not\leftrightarrow^{\mathsf{h}}} \mathsf{G} \mid 1 \le i \le n\}$

Figure 1: Definition of $\mathbin{{}^{\mathsf{h}}\!\!\not\leftrightarrow^{\mathsf{k}}}$ on global types

The core rules of our construction are rules (2) and (3), where the communication received by one of the two gateways is sent to the other one, which in turns outputs it. In global types, the order of interactions between pairs of unrelated participants is irrelevant, since we would get the very same projections. In clauses (4) and (5), however, we swap roles h and k, as well as

their corresponding global types in the "recursive call". We do that in order to spare the axiom $\mathsf{G} \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathtt{End} = \mathsf{G}$. Moreover, this swapping avoids that in $\mathsf{G} \,{}^{\mathsf{k}\nrightarrow\mathsf{h}}\, \mathsf{G}'$ the interactions following 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 6.5. The protocol implemented by the multiparty session $\mathcal{M}'$ defined on page 19 can be represented by the following global type G_r:

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

Then, by Definition 6.4, the composition, via h and k, of the global type G of Example 3.2 and the above global type G_r is:

$$\mathsf{G} \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}_\mathsf{r} = \mathsf{p} \to \mathsf{q} : \mathit{text}.\mathsf{q} \to \mathsf{h} : \mathit{text}.\mathsf{h} \to \mathsf{k} : \mathit{text}.\mathsf{k} \to \mathsf{r} : \mathit{text}.\mathsf{G}'_\mathsf{r} \,{}^{\mathsf{k}\nrightarrow\mathsf{h}}\, \mathsf{G}_1$$

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

$$\mathsf{G}_\mathsf{s} \,{}^{\mathsf{k}\nrightarrow\mathsf{h}}\, \mathsf{G} = \mathsf{p} \to \mathsf{q} : \mathit{text}.\mathsf{q} \to \mathsf{h} : \mathit{text}.\mathsf{h} \to \mathsf{k} : \mathit{text}.\mathsf{k} \to \mathsf{s} : \mathit{text}.\mathsf{G}_1 \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}'_\mathsf{s}$$

$$\begin{aligned}
\mathsf{G}_1 \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}'_\mathsf{s} = \mathsf{s} \to \mathsf{k} : \{&\mathit{ack}.\mathsf{k} \to \mathsf{h} : \mathit{ack}.\mathsf{h} \to \mathsf{q} : \mathit{ack}.\mathsf{q} \to \mathsf{p} : \mathit{ok}.\mathsf{s} \to \mathsf{r} : \mathit{go}.\mathsf{G} \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}_\mathsf{r}, \\
&\mathit{nack}.\mathsf{k} \to \mathsf{h} : \mathit{nack}.\mathsf{h} \to \mathsf{q} : \mathit{nack}.\mathsf{q} \to \mathsf{p} : \mathit{notyet}.\mathsf{s} \to \mathsf{r} : \mathit{wait}. \\
&\mathsf{q} \to \mathsf{h} : \mathit{transf}.\mathsf{h} \to \mathsf{k} : \mathit{transf}.\mathsf{k} \to \mathsf{s} : \mathit{transf}.\mathsf{G}_1 \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}'_\mathsf{s}\}
\end{aligned}$$

In $\mathsf{G} \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}_\mathsf{r}$ 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 on one side and r and s on the other side are exchanged by passing through the coupled gateways h and k.

It is worth pointing out that in $\mathsf{G} \,{}^{\mathsf{h}\nrightarrow\!\!\!\ast}\, \mathsf{G}_\mathsf{r}$, 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 $\mathbin{\mathsf{h}\!\!\not\leftrightarrow\!\!\ast}$ 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 6.10, where it is shown that $\mathsf{gw}(\mathsf{G}\!\upharpoonright_{\mathsf{h}}, \mathsf{k})$ and $\mathsf{gw}(\mathsf{G}'\!\upharpoonright_{\mathsf{k}}, \mathsf{h})$ are $\leqslant$ of the projections on h and k of $\mathsf{G} \mathbin{\mathsf{h}\!\!\not\leftrightarrow\!\!\ast} \mathsf{G}'$, 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. $\diamond$

In order to show the soundness of Definition 6.4, it is handy to express the condition $\mathsf{G}\!\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\!\upharpoonright_{\mathsf{k}}$ by means of a relation, dubbed agreement, between the pairs (G, h) and (G, k).

Definition 6.6 (Agreement Relation). *]*
The agreement relation $(\mathsf{G}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}, \mathsf{k})$ *is the largest symmetric relation coinductively defined by the rules:*

$$[\textsc{rel-End}] \quad (\mathtt{End}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}', \mathsf{k}) \ \textit{if}\ \mathsf{k} \notin \mathtt{ptg}(\mathsf{G}')$$

$$[\textsc{rel-go}] \quad \frac{\mathsf{h} \notin \{\mathsf{p}, \mathsf{q}\} \qquad (\mathsf{G}^{\ell}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}', \mathsf{k}) \quad \forall \ell . \mathsf{G}^{\ell} \in \Gamma}{(\mathsf{p} \to \mathsf{q} : \Gamma, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}', \mathsf{k})}$$

$$[\textsc{rel-comm}] \quad \frac{msg(\Gamma) = msg(\Gamma') \quad (\mathsf{G}^{\ell}, \mathsf{h}) \leftrightsquigarrow (\widehat{\mathsf{G}}^{\ell}, \mathsf{k}) \quad \forall \ell . \mathsf{G}^{\ell} \in \Gamma,\ \ell . \widehat{\mathsf{G}}^{\ell} \in \Gamma'}{(\mathsf{p} \to \mathsf{h} : \Gamma, \mathsf{h}) \leftrightsquigarrow (\mathsf{k} \to \mathsf{q} : \Gamma' \uplus \Gamma'', \mathsf{k})}$$

The following lemma connects the agreement relation between pairs global type/participant with the compatibility of the projections of global types on the participants.

Lemma 6.7. $(\mathsf{G}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}', \mathsf{k})$ *iff* $\mathsf{G}\!\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\!\upharpoonright_{\mathsf{k}}$.

Proof. From $(\mathsf{G}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}', \mathsf{k})$ we can show $\mathsf{G}\!\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\!\upharpoonright_{\mathsf{k}}$ by coinduction and by cases on the rules of Definition 6.6.

If the last applied rule is [REL-GO], let $\Gamma = \{\ell_i.\mathsf{G}_i \mid 1{\leq}i{\leq}n\}$. By coinduction $\mathsf{G}_i\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$ for all i, $1 \leq i \leq n$. By Definition 3.3 we have two cases:

1. $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{G}_1\upharpoonright_{\mathsf{h}}$ if $\mathsf{G}_i\upharpoonright_{\mathsf{h}} = \mathsf{G}_1\upharpoonright_{\mathsf{h}}$ for all i, $1 \leq i \leq n$;
2. $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{s}?(\Lambda_1 \uplus \ldots \uplus \Lambda_n)$ if $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{s}?\Lambda_i$ for all i, $1 \leq i \leq n$ and $msg(\Lambda_i) \cap msg(\Lambda_j) = \emptyset$ for all i,j, $1 \leq i \neq j \leq n$.

In case 1 we are done. In case 2 we conclude by repeatedly applying Proposition 5.2(ii).
If the last applied rule is [REL-COMM], then by coinduction $\mathsf{G}^{\ell}\upharpoonright_{\mathsf{h}} \leftrightarrow \widehat{\mathsf{G}}^{\ell}\upharpoonright_{\mathsf{k}}$ for all $\ell \in msg(\Gamma) = msg(\Gamma')$. By Definition 3.3 $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{p}?\{\ell.\mathsf{G}^{\ell}\upharpoonright_{\mathsf{h}} \mid \ell.\mathsf{G}^{\ell} \in \Gamma\}$ and $\mathsf{G}'\upharpoonright_{\mathsf{k}} = \mathsf{q}!\{\ell.\widehat{\mathsf{G}}^{\ell}\upharpoonright_{\mathsf{k}} \mid \ell.\widehat{\mathsf{G}}^{\ell} \in (\Gamma' \uplus \Gamma'')\}$. We can then derive $\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$ using rule [COMP-O/I] of Definition 5.1.

Vice versa, if $\neg\big((\mathsf{G},\mathsf{h}) \leftrightsquigarrow (\mathsf{G}',\mathsf{k})\big)$ there exists a "failing derivation" using rules of Definition 6.6 with conclusion $(\mathsf{G},\mathsf{h}) \leftrightsquigarrow (\mathsf{G}',\mathsf{k})$ and with at least one finite branch ending with a judgment having one of the following forms (or the symmetric ones):

1. $(\mathtt{End},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y},\mathsf{k})$ with $\mathsf{k} \in \mathtt{ptg}(\mathsf{Y})$;
2. $(\mathsf{p} \to \mathsf{h} : \Gamma, \mathsf{h}) \leftrightsquigarrow (\mathsf{s} \to \mathsf{k} : \Gamma', \mathsf{k})$;
3. $(\mathsf{h} \to \mathsf{p} : \Gamma, \mathsf{h}) \leftrightsquigarrow (\mathsf{k} \to \mathsf{s} : \Gamma', \mathsf{k})$;
4. $(\mathsf{p} \to \mathsf{h} : \Gamma, \mathsf{h}) \leftrightsquigarrow (\mathsf{k} \to \mathsf{s} : \Gamma', \mathsf{k})$ and there exists $\ell \in msg(\Gamma) \setminus msg(\Gamma')$.

In order to show $\neg(\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}})$ it is enough to assume $\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$ and to derive a contradiction by checking that the following statements hold:

a) $\mathsf{k} \in \mathtt{ptg}(\mathsf{Y})$ implies $\neg(\mathtt{End}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}\upharpoonright_{\mathsf{k}})$;

b) If $\mathsf{Y} = \mathsf{p} \to \mathsf{h} : \Gamma$ and $\mathsf{Y}' = \mathsf{s} \to \mathsf{k} : \Gamma'$, then $\neg(\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}})$;

c) If $\mathsf{Y} = \mathsf{h} \to \mathsf{p} : \Gamma$ and $\mathsf{Y}' = \mathsf{k} \to \mathsf{s} : \Gamma'$, then $\neg(\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}})$;

d) If $\mathsf{Y} = \mathsf{p} \to \mathsf{h} : \Gamma$, $\mathsf{Y}' = \mathsf{k} \to \mathsf{s} : \Gamma'$, and there exists $\ell \in msg(\Gamma) \setminus msg(\Gamma')$, then $\neg(\mathsf{Y}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{Y}'\upharpoonright_{\mathsf{k}})$.

a), *b*), *c*) and *d*) above descend easily from Definition 3.3 of projection and Definition 5.1 of compatibility between processes. □

Using the previous lemma we can give the following definition of compatible global types, which is alternative to Definition 6.1:

$$(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k}) \textit{ if } \mathtt{ptg}(\mathsf{G}) \cap \mathtt{ptg}(\mathsf{G}') = \emptyset \textit{ and } (\mathsf{G},\mathsf{h}) \leftrightsquigarrow (\mathsf{G}',\mathsf{k}).$$

The agreement relation is preserved by recursive calls performed during the composition of global types.

Lemma 6.8. *Let* $(\mathsf{G},\mathsf{h}) \leftrightsquigarrow (\mathsf{G}',\mathsf{k})$. *Then for any call* $\mathsf{Y} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}'$ *or* $\mathsf{Y}' \overset{\mathsf{k}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}$ *in the tree of the recursive calls of* $\mathsf{G} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{G}'$ *we get* $(\mathsf{Y},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$.

Proof. The proof is by coinduction on the rules of Definition 6.6 and by cases on the rules of Definition 6.4.
Rule (2): $\mathsf{Y} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}' = \mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{h} \to \mathsf{k} : \ell_i.\mathsf{k} \to \mathsf{s} : \ell_i.\mathsf{Y}_i \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}'_i \mid {}_{1\leq i\leq n}\}$ where $\mathsf{Y} = \mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{Y}_i \mid {}_{1\leq i\leq n}\}$ and $\mathsf{Y}' = \mathsf{k} \to \mathsf{s} : \{\ell_i.\mathsf{Y}'_i \mid {}_{1\leq i\leq n}\} \uplus \Gamma$. The relation $(\mathsf{Y},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$ is the conclusion of rule [REL-COMM] (cf. Definition 6.6) with premises $(\mathsf{Y}_i,\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}'_i,\mathsf{k})$ for all i, $1 \leq i \leq n$.
The proof for rule (3) is similar to the proof for rule (2).
Rule (4): $\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{Y}_i \mid {}_{1\leq i\leq n}\} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}' = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{Y}' \overset{\mathsf{k}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}_i \mid {}_{1\leq i\leq n}\}$. Let $\mathsf{Y} = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{Y}_i \mid {}_{1\leq i\leq n}\}$. The relation $(\mathsf{Y},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$ is the conclusion of rule [REL-GO] with premises $(\mathsf{Y}_i,\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$ for all i, $1 \leq i \leq n$.
The proof for rule (5) is similar to the proof for rule (4). □

We can now prove the soundness of Definition 6.4.

Lemma 6.9. *Let* $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$. *Then* $\mathsf{G} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{G}'$ *is defined and it is a global type, i.e. a regular pre-global type.*

Proof. From $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$ we have $(\mathsf{G},\mathsf{h}) \leftrightsquigarrow (\mathsf{G}',\mathsf{k})$. By Lemma 6.8 for all recursive calls $\mathsf{Y} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{Y}'$ in $\mathsf{G} \overset{\mathsf{h}}{\nrightarrow\!\!\!\!\!\!\ast}\ \mathsf{G}'$ we always get global types Y,Y' such that $(\mathsf{Y},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$. We show that $(\mathsf{Y},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$ assures the applicability of one rule in Definition 6.4. The proof is by cases on the rules of Definition 6.6.
Rule [REL-End]: we can apply rule (1).
Rule [REL-GO]: we can apply rule (4) or (5).
Rule [REL-COMM]: we can apply rule (2) or (3).
Notice that the applicability of the rules in alternative is due to the symmetry of the relation $(\mathsf{Y},\mathsf{h}) \leftrightsquigarrow (\mathsf{Y}',\mathsf{k})$.

The regularity of the obtained pre-global type follows by observing that the regularity of G and G' implies that the tree of the recursive calls has no infinite path with pairwise distinct calls. □

The following theorem gives the key result concerning projections of types obtained by composing via gateways.

Theorem 6.10. *If* $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$*, then* $\mathsf{G} \between \mathsf{G}'$ *is well formed. Moreover*

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

(ii) $\mathsf{G}\upharpoonright_{\mathsf{p}} \leqslant (\mathsf{G} \between \mathsf{G}')\upharpoonright_{\mathsf{p}}$ *and* $\mathsf{G}'\upharpoonright_{\mathsf{q}} \leqslant (\mathsf{G} \between \mathsf{G}')\upharpoonright_{\mathsf{q}}$ *,*
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. We recall that by Definition 3.7 a global type is well formed iff it is projectable on each participant and each participant has finite depth. It is easy to verify that if

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

then $depth(\mathsf{G} \between \mathsf{G}',\mathsf{p}) \leq 2(w+w')$ for all $\mathsf{p} \in \mathtt{ptg}(\mathsf{G}) \cup \mathtt{ptg}(\mathsf{G}')$. We show the projectability of the composition by proving (i) and (ii).
(i). We consider only the case $\mathsf{gw}(\mathsf{G}\upharpoonright_{\mathsf{h}},\mathsf{k}) \leqslant (\mathsf{G} \between \mathsf{G}')\upharpoonright_{\mathsf{h}}$ as the proof of $\mathsf{gw}(\mathsf{G}'\upharpoonright_{\mathsf{k}},\mathsf{h}) \leqslant (\mathsf{G} \between \mathsf{G}')\upharpoonright_{\mathsf{k}}$ is specular. We prove $\mathsf{gw}(\mathsf{Y}\upharpoonright_{\mathsf{h}},\mathsf{k}) \leqslant (\mathsf{Y} \between \mathsf{Y}')\upharpoonright_{\mathsf{h}}$ for any recursive call $\mathsf{Y} \between \mathsf{Y}'$ in $\mathsf{G} \between \mathsf{G}'$ by coinduction and by cases on the last applied rule.
Rules (1), (4) and (5) do not modify the communications of participant h, so coinduction easily applies.
Rule (2): $\mathsf{Y} \between \mathsf{Y}' = \mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{h} \to \mathsf{k} : \ell_i.\mathsf{k} \to \mathsf{s} : \ell_i.\mathsf{Y}_i \between \mathsf{Y}'_i \mid {\scriptstyle 1\leq i\leq n}\}$ where $\mathsf{Y} = \mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{Y}_i \mid {\scriptstyle 1\leq i\leq n}\}$ and $\mathsf{Y}' = \mathsf{k} \to \mathsf{s} : \{\ell_i.\mathsf{Y}'_i \mid {\scriptstyle 1\leq i\leq n}\} \uplus \Gamma$.
Then $\mathsf{Y}\upharpoonright_{\mathsf{h}} = \mathsf{p}?\{\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \mid {\scriptstyle 1\leq i\leq n}\}$ by Definition 3.3.

$$\begin{array}{rcll}
\mathsf{gw}(\mathsf{Y}\upharpoonright_{\mathsf{h}},\mathsf{k}) & = & \mathsf{p}?\{\ell_i.\mathsf{k}!\ell_i.\mathsf{gw}(\mathsf{Y}_i\upharpoonright_{\mathsf{h}},\mathsf{k}) \mid {\scriptstyle 1\leq i\leq n}\} & \text{by Definition 5.3} \\
 & \leqslant & \mathsf{p}?\{\ell_i.\mathsf{k}!\ell_i.(\mathsf{Y}_i \between \mathsf{Y}'_i)\upharpoonright_{\mathsf{h}} \mid {\scriptstyle 1\leq i\leq n}\} & \text{by rules [SUB-IN] and [SUB-OUT]} \\
 & & & \text{of Definition 3.9 since} \\
 & & & \text{by coinduction} \\
 & & & \mathsf{gw}(\mathsf{Y}_i\upharpoonright_{\mathsf{h}},\mathsf{k}) \leqslant (\mathsf{Y}_i \between \mathsf{Y}'_i)\upharpoonright_{\mathsf{h}} \\
 & & & \text{for all } i,\ 1 \leq i \leq n \\
 & = & (\mathsf{Y} \between \mathsf{Y}')\upharpoonright_{\mathsf{h}} & \text{by Definition 3.3.}
\end{array}$$

Rule (3): $\mathsf{Y} \between \mathsf{Y}' = \mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{k} \to \mathsf{h} : \ell_i.\mathsf{h} \to \mathsf{p} : \ell_i.\mathsf{Y}_i \between \mathsf{Y}'_i \mid {\scriptstyle 1\leq i\leq n}\}$, where $\mathsf{Y} = \mathsf{h} \to \mathsf{p} : \{\ell_i.\mathsf{Y}_i \mid {\scriptstyle 1\leq i\leq m}\}$ and $\mathsf{Y}' = \mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{Y}'_i \mid {\scriptstyle 1\leq i\leq n}\}$ with $m \geq n$.
Then $\mathsf{Y}\upharpoonright_{\mathsf{h}} = \mathsf{p}!\{\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{h}} \mid {\scriptstyle 1\leq i\leq m}\}$ by Definition 3.3.

$$
\begin{array}{rcll}
\mathsf{gw}(\mathsf{Y}\upharpoonright_{\mathsf{h}},\mathsf{k}) & = & \mathsf{k}?\{\ell_i.\mathsf{p}!\ell_i.\mathsf{gw}(\mathsf{Y}_i\upharpoonright_{\mathsf{h}},\mathsf{k}) \mid {\scriptstyle 1\leq i\leq m}\} & \text{by Definition 5.3}\\
& \leqslant & \mathsf{k}?\{\ell_i.\mathsf{p}!\ell_i.\mathsf{gw}(\mathsf{Y}_i\upharpoonright_{\mathsf{h}},\mathsf{k}) \mid {\scriptstyle 1\leq i\leq n}\} & \text{by rule [SUB-IN]}\\
& \leqslant & \mathsf{k}?\{\ell_i.\mathsf{p}!\ell_i.(\mathsf{Y}_i \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}'_i)\upharpoonright_{\mathsf{h}}\mid {\scriptstyle 1\leq i\leq n}\} & \text{by rules [SUB-IN] and [SUB-OUT]}\\
& & & \text{of Definition 3.9 since}\\
& & & \text{by coinduction}\\
& & & \mathsf{gw}(\mathsf{Y}_i\upharpoonright_{\mathsf{h}},\mathsf{k}) \leqslant (\mathsf{Y}_i \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}'_i)\upharpoonright_{\mathsf{h}}\\
& & & \text{for all } i,\ 1\leq i\leq n\\
& = & (\mathsf{Y} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}')\upharpoonright_{\mathsf{h}} & \text{by Definition 3.3.}
\end{array}
$$

(ii). We only show $\mathsf{G}\upharpoonright_{\mathsf{p}} \leqslant (\mathsf{G} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{G}')\upharpoonright_{\mathsf{p}}$ for $\mathsf{p} \in \mathtt{ptg}(\mathsf{G})$ and $\mathsf{p} \neq \mathsf{h}$. The proof of $\mathsf{G}\upharpoonright_{\mathsf{q}} \leqslant (\mathsf{G} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{G}')\upharpoonright_{\mathsf{q}}$ for $\mathsf{q} \in \mathtt{ptg}(\mathsf{G}')$ and $\mathsf{q} \neq \mathsf{k}$ is specular. Consider the recursive calls $\mathsf{Y} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}'$ in $\mathsf{G} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{G}'$. We prove $\mathsf{Y}\upharpoonright_{\mathsf{p}} \leqslant (\mathsf{Y} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}')\upharpoonright_{\mathsf{p}}$ by coinduction on Y, Y' and by cases on the last applied rule. The only rule which modifies the communications of p is rule (3):
$\mathsf{Y} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}' = \mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{k} \to \mathsf{h} : \ell_i.\mathsf{h} \to \mathsf{p} : \ell_i.\mathsf{Y}_i \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}'_i \mid {\scriptstyle 1\leq i\leq n}\}$, where $\mathsf{Y} = \mathsf{h} \to \mathsf{p} : \{\ell_i.\mathsf{Y}_i \mid {\scriptstyle 1\leq i\leq m}\}$ and $\mathsf{Y}' = \mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{Y}'_i \mid {\scriptstyle 1\leq i\leq n}\}$ with $m \geq n$.

$$
\begin{array}{rcll}
\mathsf{Y}\upharpoonright_{\mathsf{p}} & = & \mathsf{h}?\{\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{p}}\mid {\scriptstyle 1\leq i\leq m}\} & \text{by Definition 3.3}\\
& \leqslant & \mathsf{h}?\{\ell_i.\mathsf{Y}_i\upharpoonright_{\mathsf{p}}\mid {\scriptstyle 1\leq i\leq n}\} & \text{by rule [SUB-IN]}\\
& \leqslant & \mathsf{h}?\{\ell_i.(\mathsf{Y}_i \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}'_i)\upharpoonright_{\mathsf{p}}\mid {\scriptstyle 1\leq i\leq n}\} & \text{by rule [SUB-IN] since by coinduction}\\
& & & \mathsf{Y}_i\upharpoonright_{\mathsf{p}} \leqslant (\mathsf{Y}_i \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}'_i)\upharpoonright_{\mathsf{p}} \text{ for all } i,\ 1\leq i\leq n\\
& = & (\mathsf{Y} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{Y}')\upharpoonright_{\mathsf{p}} & \text{by Definition 3.3.}
\end{array}
$$

□

We now show that by composing via gateways two well-typed sessions which are compatible, we get a session which is also well typed. This is relevant, since well-typed sessions enjoy lock-freedom (Theorem 4.8).

Theorem 6.11. *If* $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ *and* $\vdash \mathcal{M} : \mathsf{G}$ *and* $\vdash \mathcal{M}' : \mathsf{G}'$*, then* $\vdash \mathcal{M} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathcal{M}' : \mathsf{G} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{G}'$*.*

Proof. Let $\mathcal{M} \equiv \mathcal{M}_1 \mid \mathsf{h} \triangleright H$ and $\mathcal{M}' \equiv \mathcal{M}'_1 \mid \mathsf{k} \triangleright K$. By construction

$$\mathcal{M} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathcal{M}' = \mathcal{M}_1 \mid \mathcal{M}'_1 \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h})$$

From $\vdash \mathcal{M} : \mathsf{G}$ we get $H \leqslant \mathsf{G}\upharpoonright_{\mathsf{h}}$ and from $\vdash \mathcal{M}' : \mathsf{G}'$ we get $K \leqslant \mathsf{G}'\upharpoonright_{\mathsf{k}}$, both thanks to the Inversion Lemma (Lemma 4.1). Lemma 5.5 gives $\mathsf{gw}(H, \mathsf{k}) \leqslant \mathsf{gw}(\mathsf{G}\upharpoonright_{\mathsf{h}}, \mathsf{k})$ and $\mathsf{gw}(K, \mathsf{h}) \leqslant \mathsf{gw}(\mathsf{G}'\upharpoonright_{\mathsf{k}}, \mathsf{h})$. From $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ by Lemma 6.2 we get $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$. We conclude $\vdash \mathcal{M} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathcal{M}' : \mathsf{G} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{G}'$ using the relations on the projections of $\mathsf{G} \mathbin{\text{h}\!\!\leftrightarrow\!\!\ast} \mathsf{G}'$ given in Theorem 6.10. □

As a general application of the previous results, let us suppose to 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 3.10). At this point, we can "deploy" the composed system (following Definition 5.8) without any further verification step, since Theorem 6.11 ensures that under such conditions we have a well-typed and hence lock-free composed system. Besides, we are able to provide the documentation (the global type) of the resulting system.

7. Compatibility of Global Types is Necessary

We have shown that the composition of typeable multiparty sessions, via the transformation of two participants into gateways, does preserve lock-freedom in case the two participants are compatible. One could wonder whether session compatibility, besides being a sufficient condition for ensuring such a preservation property, is also a necessary one. Example 6.3 shows that this is not the case. In fact taking $\mathcal{M}$ and $\mathcal{M}'$ as in Example 6.3, $\mathcal{M}\ {}_{\mathsf{h}}\!\!\leftrightarrow\!\!{}_{\mathsf{k}}\ \mathcal{M}'$ is undefined, since $(\mathcal{M},\mathsf{h}) \leftrightarrow (\mathcal{M}',\mathsf{k})$ does not hold; nonetheless we can build

$$\mathsf{p} \triangleright \mathsf{h}?\ell \mid \mathsf{h} \triangleright \mathsf{k}?\ell.\mathsf{p}!\ell \mid \mathsf{q} \triangleright \mathsf{k}!\ell \mid \mathsf{k} \triangleright \mathsf{q}?\{\ell.\mathsf{h}!\ell, \ell'.\mathsf{h}!\ell'\}$$

which is lock-free, since its only reduction terminates with $\mathsf{p}\triangleright\mathbf{0}$. The reason is that even if potentially k could try to send ℓ' to p (via h), creating a deadlock, in this session such a possibility is never taken, since q will never send ℓ' to k. Notice that in the same example $(\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})$ instead holds.

We will look then at compatibility of global types. Let $\neg\big((\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})\big)$ and $\vdash \mathcal{M} : \mathsf{G}$ and $\vdash \mathcal{M}' : \mathsf{G}'$. Then $\neg\big((\mathcal{M},\mathsf{h}) \leftrightarrow (\mathcal{M}',\mathsf{k})\big)$ by Lemma 6.2. This implies that $\mathcal{M}\ {}_{\mathsf{h}}\!\!\leftrightarrow\!\!{}_{\mathsf{k}}\ \mathcal{M}'$ is undefined. Anyway if $\mathcal{M} \equiv \mathcal{M}_1 \mid \mathsf{h} \triangleright H$ and $\mathcal{M}' \equiv \mathcal{M}'_1 \mid \mathsf{k} \triangleright K$ we can build $\mathcal{M}_1 \mid \mathcal{M}'_1 \mid \mathsf{gw}(H,\mathsf{k}) \mid \mathsf{gw}(K,\mathsf{h})$. Theorem 7.4 below shows that this parallel composition is not lock-free (or not even a session) when $\neg\big((\mathsf{G},\mathsf{h}) \leftrightarrow (\mathsf{G}',\mathsf{k})\big)$ and $\vdash \mathcal{M} : \mathsf{G}$ and $\vdash \mathcal{M}' : \mathsf{G}'$.

We proceed by providing an inductive characterisation of the relation of non-compatibility between processes.

Definition 7.1 (Non-Compatibility of Processes).
The non-compatibility relation between processes, dubbed $\not\leftrightarrow$, *is the symmetric*

closure of the relation inductively defined as follows.

$$\frac{P \neq \mathbf{0}}{P \not\leftrightarrow \mathbf{0}}\ [\text{NOCOMP-}\mathbf{0}] \qquad \frac{}{\mathsf{p}?\Lambda \not\leftrightarrow \mathsf{q}?\Lambda'}\ [\text{NOCOMP-I}] \qquad \frac{}{\mathsf{p}!\Lambda \not\leftrightarrow \mathsf{q}!\Lambda'}\ [\text{NOCOMP-O}]$$

$$\frac{msg(\Lambda') \not\subseteq msg(\Lambda)}{\mathsf{p}!\Lambda \not\leftrightarrow \mathsf{q}?\Lambda'}\ [\text{NOCOMP-LAB}] \qquad \frac{P \not\leftrightarrow Q}{\mathsf{p}!(\ell.P \uplus \Lambda) \not\leftrightarrow \mathsf{q}?(\ell.Q \uplus \Lambda')}\ [\text{NOCOMP-HERED}]$$

Lemma 7.2. *$P \not\leftrightarrow Q$ iff $\neg(P \leftrightarrow Q)$.*

Proof. If $P \not\leftrightarrow Q$, then we can show $\neg(P \leftrightarrow Q)$ by induction on the derivation of $P \not\leftrightarrow Q$. We just develop the inductive case [NOCOMP-HERED]. In this case, $P = \mathsf{p}!(\ell.P' \uplus \Lambda)$ and $Q = \mathsf{q}?(\ell.Q' \uplus \Lambda')$; moreover, $P' \not\leftrightarrow Q'$ and thus, by the induction hypothesis, $\neg(P' \leftrightarrow Q')$. We now notice that $P \leftrightarrow Q$ could only possibly hold by rule [COMP-O/I] but, since $\neg(P' \leftrightarrow Q')$, at least one of the coinductive premises of such a rule is not satisfied. Hence, we conclude $\neg(P \leftrightarrow Q)$.
Vice versa, if $\neg(P \leftrightarrow Q)$ we construct a derivation of $P \not\leftrightarrow Q$ by looking at a "failing derivation" of $P \leftrightarrow Q$. If we try to apply rule [COMP-O/I] to show $P \leftrightarrow Q$, there exists a derivation branch that fails after n steps, i.e. that reaches two processes P', Q' which do *not* match the conclusion of [COMP-O/I]. We prove $P \not\leftrightarrow Q$ by induction on n, turning the failing coinductive derivation branch into a derivation of depth $n+1$ which concludes $P \not\leftrightarrow Q$:

- base case $n = 0$. The derivation fails immediately, i.e. $P' = P$ and $Q' = Q$. By cases on the possible shapes of P and Q, we construct a derivation which concludes $P \not\leftrightarrow Q$ in $1 = n + 1$ steps, by one of the axioms [NOCOMP-$\mathbf{0}$], [NOCOMP-I], [NOCOMP-O], or [NOCOMP-LAB];

- inductive case $n = m + 1$. The shapes of P, Q match the conclusion of rule [NOCOMP-HERED], but there is some coinductive premise $P' \leftrightarrow Q'$ whose sub-derivation has a branch that fails after m steps. By the induction hypothesis, there exists a derivation of depth $m + 1$ that concludes $P' \not\leftrightarrow Q'$; using this as a premise of rule [NOCOMP-HERED] we construct a derivation of depth $(m + 1) + 1 = n + 1$ which concludes $P \not\leftrightarrow Q$. □

A lemma connecting projections of global types with reductions of typed sessions is handy.

Lemma 7.3. *Let* $\vdash \mathcal{M} : \mathsf{G}$.

(i) If $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}?\Lambda$ *with* $\ell \in msg(\Lambda)$*, then there is a reduction*

$$\mathcal{M} \longrightarrow^* \mathcal{M}' \mid \mathsf{q} \triangleright \mathsf{p}!\Lambda' \text{ such that } \ell \in msg(\Lambda').$$

(ii) If $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\Lambda$ *with* $\ell \in msg(\Lambda)$*, then there is a reduction*

$$\mathcal{M} \longrightarrow^* \mathcal{M}' \mid \mathsf{q} \triangleright \mathsf{p}?\Lambda' \text{ such that } \ell \in msg(\Lambda').$$

Proof. The proofs of both items are by induction on $w = depth(\mathsf{G}, \mathsf{p})$.
(i). If $w = 0$, then $\mathsf{G} = \mathsf{q} \to \mathsf{p} : \Gamma$ and $\mathsf{q} \triangleright Q \in \mathcal{M}$ and $Q \leqslant \mathsf{G}\upharpoonright_{\mathsf{q}}$. We get the statement without reducing $\mathcal{M}$, since $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}!\Lambda'$ and $msg(\Lambda') = msg(\Lambda) = msg(\Gamma)$.
If $w > 0$, then $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}$ with $\mathsf{p} \notin \{\mathsf{r}, \mathsf{s}\}$, then by Definition 3.3 there is j, $1 \leq j \leq n$, such that $\mathsf{G}_j\upharpoonright_{\mathsf{p}} = \mathsf{q}?\Lambda_j$ with $\ell \in \Lambda_j$. By rule [ECOMM] of Definition 4.3 $\mathsf{G} \xrightarrow{\mathsf{r}\ell_j\mathsf{s}} \mathsf{G}_j$. By Theorem 4.7(ii) $\mathcal{M} \xrightarrow{\mathsf{r}\ell_j\mathsf{s}} \mathcal{M}''$ and $\vdash \mathcal{M}'' : \mathsf{G}_j$. We conclude using induction, since by Lemma 4.4 $depth(\mathsf{G}_j, \mathsf{p}) < w$.

The proof of (ii) is simpler than the proof of (i), since when $w > 0$ and $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\}$ with $\mathsf{p} \notin \{\mathsf{r}, \mathsf{s}\}$, then by Definition 3.3 we have $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{G}_1\upharpoonright_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. □

We can now prove that, whereas compatibility between sessions is not a necessary condition for lock-freedom preservation (as shown at the beginning of this section using Example 6.3), compatibility between global types is.

Theorem 7.4.
If $\neg\big((\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})\big)$ *and* $\vdash \mathcal{M} \mid \mathsf{h} \triangleright H : \mathsf{G}$ *and* $\vdash \mathcal{M}' \mid \mathsf{k} \triangleright K : \mathsf{G}'$*, then either* $\mathcal{M} \mid \mathcal{M}' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h})$ *is not a session or it is not lock-free.*

Proof. If $\mathtt{ptg}(\mathsf{G}) \cap \mathtt{ptg}(\mathsf{G}') \neq \emptyset$, then $\mathtt{pts}(\mathcal{M} \mid \mathsf{h} \triangleright H) \cap \mathtt{pts}(\mathcal{M}' \mid \mathsf{k} \triangleright K) \neq \emptyset$. In this case $\mathcal{M} \mid \mathcal{M}' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h})$ has at least one repeated participant, so it is not a session. Otherwise $\neg(\mathsf{G}\upharpoonright_{\mathsf{h}} \leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}})$, i.e. $\mathsf{G}\upharpoonright_{\mathsf{h}} \not\leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$ by Lemma 7.2. We show that $\mathcal{M} \mid \mathcal{M}' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h})$ is not lock-free by induction on the derivation of $\mathsf{G}\upharpoonright_{\mathsf{h}} \not\leftrightarrow \mathsf{G}'\upharpoonright_{\mathsf{k}}$ (cf. Definition 7.1). In each case we use that $\vdash \mathcal{M} \mid \mathsf{h} \triangleright H : \mathsf{G}$ implies $H \leqslant \mathsf{G}\upharpoonright_{\mathsf{h}}$ and $\vdash \mathcal{M} \mid \mathsf{k} \triangleright K : \mathsf{G}'$

implies $K \leqslant \mathsf{G}'\upharpoonright_{\mathsf{k}}$ by Lemma 4.1.
If $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathbf{0}$ and $\mathsf{G}'\upharpoonright_{\mathsf{k}} \neq \mathbf{0}$, then $H = \mathbf{0}$ and $K \neq \mathbf{0}$. This implies $\mathsf{gw}(H, \mathsf{k}) = \mathbf{0}$ but $\mathsf{gw}(K, \mathsf{h}) \neq \mathbf{0}$. The messages in $\mathsf{gw}(K, \mathsf{h})$ with sender or receiver h will never be consumed.
If $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{p}?\Lambda_1$ and $\mathsf{G}'\upharpoonright_{\mathsf{k}} = \mathsf{q}?\Lambda_2$, then $H = \mathsf{p}?\Lambda_1'$ and $K = \mathsf{q}?\Lambda_2'$. This implies $\mathsf{gw}(H, \mathsf{k})$ (after receiving a message from p) wants to send a message to k but $\mathsf{gw}(K, \mathsf{h})$ (after receiving a message from q) wants to send a message to h.
If $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{p}!\Lambda_1$ and $\mathsf{G}'\upharpoonright_{\mathsf{k}} = \mathsf{q}!\Lambda_2$, then $H = \mathsf{p}!\Lambda_1'$ and $K = \mathsf{q}!\Lambda_2'$. This implies $\mathsf{gw}(H, \mathsf{k})$ (before sending a message to p) waits for a message from k but $\mathsf{gw}(K, \mathsf{h})$ (before sending a message to q) waits for a message from h.
Let $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{p}!\Lambda_1$ and $\mathsf{G}'\upharpoonright_{\mathsf{k}} = \mathsf{q}?(\ell.P \uplus \Lambda_2)$ and $\ell \notin msg(\Lambda_1)$, then $H = \mathsf{p}!\Lambda_1'$ with $msg(\Lambda_1') = msg(\Lambda_1)$ and $K = \mathsf{q}?(\ell.P' \uplus \Lambda_2')$ with $msg(\Lambda_2') \supseteq msg(\Lambda_2)$. From $\mathsf{G}'\upharpoonright_{\mathsf{k}} = \mathsf{q}?(\ell.P \uplus \Lambda_2)$ we get $\mathcal{M}' \longrightarrow^* \mathcal{M}_1' \mid \mathsf{q} \triangleright Q$ with $Q = \mathsf{k}!(\ell.Q' \uplus \Lambda_3)$ by Lemma 7.3(i). After Q exchanges the message ℓ with $\mathsf{gw}(K, \mathsf{h})$, the process $\mathsf{gw}(K, \mathsf{h})$ wants to send the message ℓ to h, but $\mathsf{gw}(H, \mathsf{k})$ cannot receive this message. More precisely

$$\begin{array}{l}\mathcal{M} \mid \mathcal{M}' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h}) \longrightarrow^* \\ \qquad \mathcal{M} \mid \mathcal{M}_1' \mid \mathsf{q} \triangleright Q \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h}) \xrightarrow{\mathsf{q}\ell\mathsf{k}} \\ \qquad\qquad \mathcal{M} \mid \mathcal{M}_1' \mid \mathsf{q} \triangleright Q' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{h}!\ell.\mathsf{gw}(P, \mathsf{h})\end{array}$$

which cannot reduce since $\mathsf{gw}(H, \mathsf{k}) = \mathsf{k}?\Lambda$ with $\ell \notin msg(\Lambda) = msg(\Lambda_1)$.
Let $\mathsf{G}\upharpoonright_{\mathsf{h}} = \mathsf{p}!(\ell.\mathsf{G}_1\upharpoonright_{\mathsf{h}} \uplus \Lambda_1)$ and $\mathsf{G}'\upharpoonright_{\mathsf{k}} = \mathsf{q}?(\ell.\mathsf{G}_2\upharpoonright_{\mathsf{k}} \uplus \Lambda_2)$ and $\mathsf{G}_1\upharpoonright_{\mathsf{h}} \not\leftrightarrows \mathsf{G}_2\upharpoonright_{\mathsf{k}}$. Then $H = \mathsf{p}!(\ell.P \uplus \Lambda_1')$ and $K = \mathsf{q}?(\ell.Q \uplus \Lambda_2')$ and $P \leqslant \mathsf{G}_1\upharpoonright_{\mathsf{h}}$, $Q \leqslant \mathsf{G}_2\upharpoonright_{\mathsf{k}}$. By Lemma 7.3 $\mathcal{M} \longrightarrow^* \mathcal{M}_1 \mid \mathsf{p} \triangleright \mathsf{h}?(\ell.P' \uplus \Lambda_3)$ and $\mathcal{M}' \longrightarrow^* \mathcal{M}_1' \mid \mathsf{q} \triangleright \mathsf{k}!(\ell.Q' \uplus \Lambda_3')$. Then

$$\mathcal{M} \mid \mathsf{h} \triangleright H \longrightarrow^* \mathcal{M}_1 \mid \mathsf{p} \triangleright \mathsf{h}?(\ell.P' \uplus \Lambda_3) \mid \mathsf{h} \triangleright H \xrightarrow{\mathsf{h}\ell\mathsf{p}} \mathcal{M}_1 \mid \mathsf{p} \triangleright P' \mid \mathsf{h} \triangleright P$$

and by the Subject Reduction Theorem (Theorem 4.6) there is G_1' such that $\vdash \mathcal{M}_1 \mid \mathsf{p} \triangleright P' \mid \mathsf{h} \triangleright P : \mathsf{G}_1'$. Similarly

$$\mathcal{M}' \mid \mathsf{k} \triangleright K \longrightarrow^* \mathcal{M}_1' \mid \mathsf{q} \triangleright \mathsf{k}!(\ell.Q' \uplus \Lambda_3') \mid \mathsf{k} \triangleright K \xrightarrow{\mathsf{q}\ell\mathsf{k}} \mathcal{M}_1' \mid \mathsf{q} \triangleright Q' \mid \mathsf{k} \triangleright Q$$

and $\vdash \mathcal{M}_1' \mid \mathsf{q} \triangleright Q' \mid \mathsf{k} \triangleright Q : \mathsf{G}_2'$ for some G_2'. Building G_1' and G_2' as in the proof of Theorem 4.6 we get $\mathsf{G}_1'\upharpoonright_{\mathsf{h}} = \mathsf{G}_1\upharpoonright_{\mathsf{h}}$ and $\mathsf{G}_2'\upharpoonright_{\mathsf{k}} = \mathsf{G}_2\upharpoonright_{\mathsf{k}}$, which imply $\mathsf{G}_1'\upharpoonright_{\mathsf{h}} \not\leftrightarrows \mathsf{G}_2'\upharpoonright_{\mathsf{k}}$. When $\mathsf{gw}(K, \mathsf{h})$ and $\mathsf{gw}(H, \mathsf{k})$ exchange the message ℓ they become $\mathsf{gw}(P, \mathsf{k})$ and $\mathsf{gw}(Q, \mathsf{h})$, so the network is not lock-free by induction hypothesis. More

precisely

$$\begin{array}{l}
\mathcal{M} \mid \mathcal{M}' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h}) \longrightarrow^* \\
\quad \mathcal{M} \mid \mathcal{M}'_1 \mid \mathsf{q} \triangleright \mathsf{k}!(\ell.Q' \uplus \Lambda'_3) \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(K, \mathsf{h}) \xrightarrow{\mathsf{q}\ell\mathsf{k}} \\
\quad\quad \mathcal{M} \mid \mathcal{M}'_1 \mid \mathsf{q} \triangleright Q' \mid \mathsf{h} \triangleright \mathsf{gw}(H, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{h}!\ell.\mathsf{gw}(Q, \mathsf{h}) \xrightarrow{\mathsf{k}\ell\mathsf{h}} \\
\quad\quad\quad \mathcal{M} \mid \mathcal{M}'_1 \mid \mathsf{q} \triangleright Q' \mid \mathsf{h} \triangleright \mathsf{p}!\ell.\mathsf{gw}(P, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(Q, \mathsf{h}) \longrightarrow^* \\
\quad\quad\quad\quad \mathcal{M}_1 \mid \mathsf{p} \triangleright \mathsf{h}?(\ell.P' \uplus \Lambda_3) \mid \mathcal{M}'_1 \mid \mathsf{q} \triangleright Q' \mid \mathsf{h} \triangleright \mathsf{p}!\ell.\mathsf{gw}(P, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(Q, \mathsf{h}) \xrightarrow{\mathsf{h}\ell\mathsf{p}} \\
\quad\quad\quad\quad\quad \mathcal{M}_1 \mid \mathsf{p} \triangleright P' \mid \mathcal{M}'_1 \mid \mathsf{q} \triangleright Q' \mid \mathsf{h} \triangleright \mathsf{gw}(P, \mathsf{k}) \mid \mathsf{k} \triangleright \mathsf{gw}(Q, \mathsf{h})
\end{array}$$

is not lock-free by induction hypothesis. □

Notice that the theorem above is stated for typeable multiparty sessions, since it uses compatibility of their global types.

8. Direct Composition of Typed Multiparty Sessions

The use of gateways enables us to get a "safe" composition of systems by minimally affecting the components themselves, since just the interface participants need to be modified.

Therefore the composition via gateways is useful after deployment, since it minimally affects systems that are already implemented. On the other hand, one would prefer to avoid the overhead due to gateways when composition is performed statically, as part of a modular design of systems via global types. In this setting one could easily modify the involved participants, since they have not been implemented yet.

One could hence wonder whether gateways are strictly necessary to get safe compositions in our multiparty-sessions setting. It is quite natural to expect that one could "bypass" the use of gateways by removing them and simply applying a renaming function on the other participants' "code". This however would be too naive an approach. As pointed out in [2] (Sect. 5.3), a "safe" composition of their systems which bypasses the use of gateways could require an heavy redesign of the participants involved in the inter-systems interactions. A disciplined setting as the present one, instead, enables to handle direct composition more easily, both at the level of sessions and of global types, but some care is anyway needed. The following example, in fact, shows that just a renaming would not work in general.

Example 8.1. Consider the following global types and multiparty sessions.

$$\begin{array}{rcl@{\qquad\qquad}rcl} \mathsf{G} & = & \mathsf{p} \to \mathsf{h} : \ell.\mathsf{G} & \mathsf{G}' & = & \mathsf{k} \to \mathsf{r} : \ell.\mathsf{k} \to \mathsf{s} : \ell.\mathsf{G}' \\ \mathcal{M} & = & \mathsf{p} \triangleright P \mid \mathsf{h} \triangleright H & \mathcal{M}' & = & \mathsf{r} \triangleright R \mid \mathsf{s} \triangleright S \mid \mathsf{k} \triangleright K \end{array}$$

where

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

It is easy to check that $\vdash \mathcal{M} : \mathsf{G}$ and $\vdash \mathcal{M}' : \mathsf{G}'$.

In order to compose the above multiparty sessions, bypassing the gateways, we could take K out on the side of $\mathcal{M}'$, and rename some senders' and receivers' names in R and S so that they can receive the message ℓ directly from p, obtaining $\tilde{R} = \mathsf{p}?\ell.\tilde{R}$ and $\tilde{S} = \mathsf{p}?\ell.\tilde{S}$. On the side of $\mathcal{M}$, instead, after taking out H, we could not get a sound composition 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 direct composition would hence imply a less straightforward modification of P as follows: $\tilde{P} = \mathsf{r}!\ell.\mathsf{s}!\ell.\tilde{P}$. $\diamond$

We implement a direct composition at the level of global types by means of the function $\_ \not\bowtie \_$ defined below and we shall then use this function to achieve a similar composition of multiparty sessions as well.

Definition 8.2 (Direct Composition of Global Types).
Let $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$. *We define*

$$\mathsf{G} \not\bowtie \mathsf{G}'$$

coinductively by the clauses of Figure 2, assuming $\{\mathsf{p}, \mathsf{q}, \mathsf{r}, \mathsf{s}\} \cap \{\mathsf{h}, \mathsf{k}\} = \emptyset$. *The clauses must be applied in the given order.*

Taking G and G' of Example 8.1, it is easy to check that $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$ and hence that we can apply direct composition on these global types, obtaining:

$$\mathsf{G} \not\bowtie \mathsf{G}' \quad = \quad \mathsf{p} \to \mathsf{r} : \ell.\mathsf{p} \to \mathsf{s} : \ell.(\mathsf{G} \not\bowtie \mathsf{G}')$$

We prove now that the direct composition function on global types is well defined.

Lemma 8.3. *Let* $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$. *Then* $\mathsf{G} \not\bowtie \mathsf{G}'$ *is defined and it is a global type, i.e. a regular pre-global type.*

(1) $\mathtt{End} \not\bowtie \mathsf{G} = \mathsf{G}$

(2) $\mathsf{p} \to \mathsf{h} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\} \not\bowtie \mathsf{k} \to \mathsf{s} : \{\ell_i.\mathsf{G}'_i \mid 1 \leq i \leq n\} \uplus \Gamma = \mathsf{p} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \not\bowtie \mathsf{G}'_i \mid 1 \leq i \leq n\}$

(3) $\mathsf{h} \to \mathsf{p} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\} \uplus \Gamma \not\bowtie \mathsf{s} \to \mathsf{k} : \{\ell_i.\mathsf{G}'_i \mid 1 \leq i \leq n\} = \mathsf{s} \to \mathsf{p} : \{\ell_i.\mathsf{G}_i \not\bowtie \mathsf{G}'_i \mid 1 \leq i \leq n\}$

(4) $\mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\} \not\bowtie \mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G} \not\bowtie \mathsf{G}_i \mid 1 \leq i \leq n\}$

(5) $\mathsf{G} \not\bowtie \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid 1 \leq i \leq n\} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \not\bowtie \mathsf{G} \mid 1 \leq i \leq n\}$

Figure 2: Definition of $\not\bowtie$ on global types

Proof. This lemma is the analogous of Lemma 6.9, whose proof uses Lemma 6.8. Notice that Figure 1 and Figure 2 have exactly the same global types on the left side of the equalities. The proofs of Lemmas 6.8 and 6.9 only use these global types. Therefore we can easily adapt these proofs to $\mathsf{G} \not\bowtie \mathsf{G}'$. □

The global type $\mathsf{G} \not\bowtie \mathsf{G}'$ can be shown to be well formed and to preserve the projections of those participants which do not communicate with h or k.

Theorem 8.4. *If* $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$*, then* $\mathsf{G} \not\bowtie \mathsf{G}'$ *is well formed. If* $\mathsf{p} \in \mathtt{ptg}(\mathsf{G})$ *and* $\mathsf{h} \notin \mathtt{ptp}(\mathsf{G}\upharpoonright_{\mathsf{p}})$*, then* $\mathsf{G}\upharpoonright_{\mathsf{p}} = (\mathsf{G} \not\bowtie \mathsf{G}')\upharpoonright_{\mathsf{p}}$*. If* $\mathsf{p} \in \mathtt{ptg}(\mathsf{G}')$ *and* $\mathsf{k} \notin \mathtt{ptp}(\mathsf{G}'\upharpoonright_{\mathsf{p}})$*, then* $\mathsf{G}'\upharpoonright_{\mathsf{p}} = (\mathsf{G} \not\bowtie \mathsf{G}')\upharpoonright_{\mathsf{p}}$*.*

Proof. It is easy to verify that if

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

then $depth(\mathsf{G} \not\bowtie \mathsf{G}', \mathsf{p}) \leq max\{w, w'\}$ for all $\mathsf{p} \in \mathtt{ptg}(\mathsf{G}) \cup \mathtt{ptg}(\mathsf{G}')$. We can show that $\mathsf{G} \not\bowtie \mathsf{G}'$ is projectable using the projectability of G, G' and the definition of $\mathsf{G} \not\bowtie \mathsf{G}'$.
If $\mathsf{p} \in \mathtt{ptg}(\mathsf{Y})$ and $\mathsf{h} \notin \mathtt{ptp}(\mathsf{Y}\upharpoonright_{\mathsf{p}})$ we prove that $\mathsf{Y}\upharpoonright_{\mathsf{p}} = (\mathsf{Y} \not\bowtie \mathsf{Y}')\upharpoonright_{\mathsf{p}}$ for any recursive call $\mathsf{Y} \not\bowtie \mathsf{Y}'$ in $\mathsf{G} \not\bowtie \mathsf{G}'$ by coinduction and by cases on the last applied rule. Since $\mathsf{h} \notin \mathtt{ptp}(\mathsf{Y}\upharpoonright_{\mathsf{p}})$ only rules (4) and (5) can be used and these rules do not modify the communications of p, so we get the statement. The case of $\mathsf{p} \in \mathtt{ptg}(\mathsf{Y}')$ and $\mathsf{k} \notin \mathtt{ptp}(\mathsf{Y}'\upharpoonright_{\mathsf{p}})$ is analogous. □

We can hence obtain a direct composition function at the multiparty-session level through the use of the function $\not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow$ on global types.

Definition 8.5 (Direct Composition of Multiparty Sessions). *Let* $\vdash \mathcal{M} : \mathsf{G}$, $\vdash \mathcal{M}' : \mathsf{G}'$, $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$, *with* $\mathcal{M} \equiv \mathcal{M}_1 \mid \mathcal{M}_2$ *and* $\mathcal{M}' \equiv \mathcal{M}'_1 \mid \mathcal{M}'_2$ *such that:*

- $\mathsf{h} \in \mathtt{ptp}(P)$ *for all* $\mathsf{p} \triangleright P \in \mathcal{M}_1$
- $\mathsf{h} \notin \mathtt{ptp}(P)$ *for all* $\mathsf{p} \triangleright P \in \mathcal{M}_2$
- $\mathsf{k} \in \mathtt{ptp}(Q)$ *for all* $\mathsf{q} \triangleright Q \in \mathcal{M}'_1$
- $\mathsf{k} \notin \mathtt{ptp}(Q)$ *for all* $\mathsf{q} \triangleright Q \in \mathcal{M}'_2$

We define

$$\mathcal{M} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathcal{M}' = \prod_{\mathsf{p} \in \mathtt{pts}(\mathcal{M}_1)} \mathsf{p} \triangleright (\mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}')\restriction_{\mathsf{p}} \mid \mathcal{M}_2 \mid \prod_{\mathsf{q} \in \mathtt{pts}(\mathcal{M}'_1)} \mathsf{q} \triangleright (\mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}')\restriction_{\mathsf{q}} \mid \mathcal{M}'_2$$

The main result of this section connects the direct composition of global types and the direct composition of multiparty sessions.

Theorem 8.6. *If* $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ *and* $\vdash \mathcal{M} : \mathsf{G}$ *and* $\vdash \mathcal{M}' : \mathsf{G}'$, *then* $\vdash \mathcal{M} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathcal{M}' : \mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}'$.

Proof. Lemma 6.2 gives $(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$. By Theorem 8.4 $\mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}'$ is well formed. We conclude $\vdash \mathcal{M} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathcal{M}' : \mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}'$ using the relations on the projections of $\mathsf{G} \not\leftrightarrow\!\!\!\!\not\leftrightarrow \mathsf{G}'$ given in Theorem 8.4. □

Example 8.7. It is almost immediate to check that $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ for Example 8.1. We obtain

$$\mathcal{M} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathcal{M}' = \mathsf{p} \triangleright \tilde{P} \mid \mathsf{r} \triangleright \tilde{R} \mid \mathsf{s} \triangleright \tilde{S}$$

where

$$\tilde{R} = \mathsf{p}?\ell.\tilde{R} \qquad \tilde{S} = \mathsf{p}?\ell.\tilde{S} \qquad \tilde{P} = \mathsf{r}!\ell.\mathsf{s}!\ell.\tilde{P}$$

and $\vdash \mathcal{M} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathcal{M}' : \mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}'$. ◇

For direct compositions we cannot compose $\mathcal{M}$ and $\mathcal{M}'$ when $\vdash \mathcal{M} : \mathsf{G}$ and $\vdash \mathcal{M}' : \mathsf{G}'$ and $\neg((\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k}))$, since $\mathsf{G} \not\leftrightarrow\!\!\!\!\!\!\not\leftrightarrow \mathsf{G}'$ is undefined. So we cannot have a result similar to Theorem 7.4.

9. Decomposition of Typed Multiparty Sessions

In order to use direct composition in the modular development of systems one needs first to decompose a global specification into modules, then to develop the various modules in isolation, and finally to combine the modules using direct composition. We still miss an operation to decompose a global specification: this will be introduced in the present section. In particular, we will define a function enabling to decompose a global type into two global types,n whose direct composition represents the same system of processes represented by the original global type.

Definition 9.1. *The projection of a global type* G *with respect to a set of participants* $\mathcal{P} \subseteq \mathtt{ptg}(\mathsf{G})$ *and one interface participant* $\mathsf{h} \notin \mathtt{ptg}(\mathsf{G})$ *is the global type* $\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}$ *(if any) defined in Figure 3.*

$$\mathsf{G} \Uparrow^{\mathsf{h}}_{\mathcal{P}} = \mathtt{End} \qquad \text{if } \mathtt{ptg}(\mathsf{G}) \cap \mathcal{P} = \emptyset$$

$$\mathsf{p} \to \mathsf{q} : \Gamma \Uparrow^{\mathsf{h}}_{\mathcal{P}} = \begin{cases} \mathsf{p} \to \mathsf{q} : \{\ell.\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}} \mid \ell.\mathsf{G} \in \Gamma\} & \text{if } \mathsf{p},\mathsf{q} \in \mathcal{P} \\ \mathsf{p} \to \mathsf{h} : \{\ell.\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}} \mid \ell.\mathsf{G} \in \Gamma\} & \text{if } \mathsf{p} \in \mathcal{P}, \mathsf{q} \notin \mathcal{P} \\ \mathsf{h} \to \mathsf{q} : \{\ell.\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}} \mid \ell.\mathsf{G} \in \Gamma\} & \text{if } \mathsf{p} \notin \mathcal{P}, \mathsf{q} \in \mathcal{P} \\ \widehat{\mathsf{G}} & \text{if } \{\mathsf{p},\mathsf{q}\} \cap \mathcal{P} = \emptyset \text{ and} \\ & \widehat{\mathsf{G}} = \mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}} \text{ for all } \ell.\mathsf{G} \in \Gamma \end{cases}$$

Figure 3: Definition of $\Uparrow$ for global types

It is easy to check that if $\mathtt{ptg}(\mathsf{G}) = \mathcal{P}$, then $\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}} = \mathsf{G}$. Moreover if $\mathcal{P} = \emptyset$, then $\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}} = \mathtt{End}$. Clearly this projection is a partial function. In fact the condition in case $\mathsf{p},\mathsf{q} \notin \mathcal{P}$ is quite demanding. This requirement could be relaxed (similarly to what is done in Definition 3.3), but this would complicate the definition and we preferred simplicity. This condition reflects on the projection of global types as stated in the following proposition.

Proposition 9.2.
If $\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}}$ *is defined and* G *contains* $\mathsf{p} \to \mathsf{q} : \Gamma$ *with* $\{\mathsf{p},\mathsf{q}\} \cap \mathcal{P} = \emptyset$*, then there is* $\widehat{\mathsf{G}}$ *such that* $\mathsf{G}^{\ell}\upharpoonright_{\mathsf{r}} = \widehat{\mathsf{G}}\upharpoonright_{\mathsf{r}}$ *for all* $\mathsf{r} \in \mathtt{ptg}(\mathsf{G})$ *and all* $\ell.\mathsf{G}^{\ell} \in \Gamma$*.*

Proof. By coinduction on G and by cases on Definition 9.1. □

We can now define the decomposition function on global types. Basically, the participants are split in two different global types, and two fresh interface participants are created to manage the communications between the two groups.

Definition 9.3 (Decomposition of Global Types).
Let G *be a global type,* $\{\mathcal{P}, \mathcal{Q}\}$ *a partition of* $\mathtt{ptg}(\mathsf{G})$ *and* $\mathsf{h}, \mathsf{k} \notin \mathtt{ptg}(\mathsf{G})$ *with* $\mathsf{h} \neq \mathsf{k}$. *We set*

$$\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}, \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}\big)$$

if $\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}$ *and* $\mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}$ *are defined.*

A first lemma relates decomposition and compatibility of global types.

Lemma 9.4. *If* $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\mathsf{G}_1, \mathsf{G}_2\big)$, *then* $(\mathsf{G}_1, \mathsf{h}) \leftrightarrow (\mathsf{G}_2, \mathsf{k})$.

Proof. Clearly $\mathtt{ptg}(\mathsf{G}_1) \cap \mathtt{ptg}(\mathsf{G}_2) = \emptyset$. By Lemma 6.7 it is enough to show $(\mathsf{G}_1, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}_2, \mathsf{k})$. The proof is by coinduction on G and by cases on Definition 9.1.

If $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ with $\mathsf{p} \in \mathcal{P}$ and $\mathsf{q} \in \mathcal{Q}$, then by Definition 9.3

$$\begin{aligned}
\mathsf{G}_1 = \mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} &= \mathsf{p} \to \mathsf{h} : \{\ell.\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid \ell.\mathsf{G}^{\ell} \in \Gamma\} \quad \text{and} \\
\mathsf{G}_2 = \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} &= \mathsf{k} \to \mathsf{q} : \{\ell.\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} \mid \ell.\mathsf{G}^{\ell} \in \Gamma\}
\end{aligned}$$

By coinduction we have that, for all $\ell.\mathsf{G}^{\ell} \in \Gamma$, $(\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}, \mathsf{k})$ and hence we can derive $(\mathsf{G}_1, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}_2, \mathsf{k})$ using rule [REL-COMM] of Definition 6.6.

If $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ with $\mathsf{p}, \mathsf{q} \in \mathcal{P}$, then

$$\mathsf{G}_1 = \mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \mathsf{p} \to \mathsf{q} : \{\ell.\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid \ell.\mathsf{G}^{\ell} \in \Gamma\} \quad \text{and} \quad \mathsf{G}_2 = \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} = \widehat{\mathsf{G}}$$

for some global type $\widehat{\mathsf{G}}$ such that $\widehat{\mathsf{G}} = \mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}$ for all $\ell.\mathsf{G}^{\ell} \in \Gamma$. By coinduction we have that $(\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}, \mathsf{k})$ for all $\ell.\mathsf{G}^{\ell} \in \Gamma$. Then $(\mathsf{G}_1, \mathsf{h}) \leftrightsquigarrow (\mathsf{G}_2, \mathsf{k})$ using rule [REL-GO] of Definition 6.6. □

The following theorem shows the desired duality between decomposition and direct composition.

Theorem 9.5. *If* $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\mathsf{G}_1, \mathsf{G}_2\big)$, *then* $\mathsf{G}_1 \,{}^{\mathsf{h}}\!\!\not\bowtie\!\!{}^{\mathsf{k}}\, \mathsf{G}_2 = \mathsf{G}$.

Proof. We show the thesis by coinduction on the structure of G and by cases on Definition 9.1.

If $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ with $\mathsf{p} \in \mathcal{P}$ and $\mathsf{q} \in \mathcal{Q}$ then

$$\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \mathsf{p} \to \mathsf{h} : \{\ell.\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid \ell.\mathsf{G}^{\ell} \in \Gamma\} \quad \text{and} \quad \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} = \mathsf{k} \to \mathsf{q} : \{\ell.\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} \mid \ell.\mathsf{G}^{\ell} \in \Gamma\}$$

By Definition 8.2 we get

$$\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} = \mathsf{p} \to \mathsf{q} : \{\ell.(\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}) \mid \ell.\mathsf{G}^{\ell} \in \Gamma\}$$

By coinduction we have that $\mathsf{G}^{\ell} = \mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}$ for all $\ell.\mathsf{G}^{\ell} \in \Gamma$ and hence the thesis.

If $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$ with $\mathsf{p}, \mathsf{q} \in \mathcal{P}$ then

$$\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \mathsf{p} \to \mathsf{q} : \{\ell.\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid 1 \leq i \leq n\} \quad \text{and} \quad \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} = \widehat{\mathsf{G}}$$

for some global type $\widehat{\mathsf{G}}$ such that $\widehat{\mathsf{G}} = \mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}$ for all $\ell.\mathsf{G}^{\ell} \in \Gamma$. By Definition 8.2 we get

$$\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} = \mathsf{p} \to \mathsf{q} : \{\ell.(\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \widehat{\mathsf{G}}) \mid \ell.\mathsf{G}^{\ell} \in \Gamma\}$$

which implies

$$\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \mathsf{G}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} = \mathsf{p} \to \mathsf{q} : \{\ell.(\mathsf{G}^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \between \mathsf{G}^{\ell}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}}) \mid \ell.\mathsf{G}^{\ell} \in \Gamma\}$$

and we can conclude as in previous case. □

In the remaining of this section we discuss the decomposition of typed sessions. A notion of process projection is handy.

Definition 9.6. *The projection of a process P with respect to a set of participants $\mathcal{P}$ and one interface participant $\mathsf{h} \notin \mathcal{P}$ is the process $P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}$ defined by:*

$$\mathbf{0}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \mathbf{0} \qquad \mathsf{p}\dagger\Lambda{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \begin{cases} \mathsf{p} \dagger \{\ell.P^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid \ell.P^{\ell} \in \Lambda\} & \text{if } \mathsf{p} \in \mathcal{P}, \\ \mathsf{h} \dagger \{\ell.P^{\ell}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid \ell.P^{\ell} \in \Lambda\} & \text{if } \mathsf{p} \notin \mathcal{P} \end{cases}$$

where $\dagger$ stands for $?$ or $!$.

It is easy to verify that if $\mathcal{P} = \emptyset$, then $\mathtt{ptp}(P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}) = \{\mathsf{h}\}$.

We look for a decomposition procedure at the multiparty-session level, through the use of the projection in Definition 9.1. Notice that if $\vdash \mathcal{M} : \mathsf{G}$ and $\{\mathcal{P}, \mathcal{Q}\}$ is a partition of $\mathtt{ptg}(\mathsf{G})$, then $\mathcal{M} \equiv \Pi_{\mathsf{p} \in \mathcal{P}} \mathsf{p} \triangleright P_{\mathsf{p}} \mid \Pi_{\mathsf{q} \in \mathcal{Q}} \mathsf{q} \triangleright Q_{\mathsf{q}}$.

Definition 9.7. *Let* $\vdash \mathcal{M} : \mathsf{G}$ *and* $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}, \mathcal{P}, \mathcal{Q}) = (\mathsf{G}_1, \mathsf{G}_2)$ *and* $\mathcal{M} \equiv \Pi_{\mathsf{p}\in\mathcal{P}}\mathsf{p} \triangleright P_{\mathsf{p}} \mid \Pi_{\mathsf{q}\in\mathcal{Q}}\mathsf{q} \triangleright Q_q$. *We define:*

$$\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathcal{M}, \mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\Pi_{\mathsf{p}\in\mathcal{P}}\mathsf{p} \triangleright P_{\mathsf{p}}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \ \mid \mathsf{h} \triangleright \mathsf{G}_1{\restriction}_{\mathsf{h}}, \Pi_{\mathsf{q}\in\mathcal{Q}}\mathsf{q} \triangleright Q_{\mathsf{q}}{\Uparrow}^{\mathsf{k}}_{\mathcal{Q}} \ \mid \mathsf{k} \triangleright \mathsf{G}_2{\restriction}_{\mathsf{k}} \ \big)$$

Notably, the decomposition function above exploits the type of the session.

A last proposition on subtyping between input processes (with a trivial proof) is handy in showing the final theorem.

Proposition 9.8. *If* $P \leqslant \mathsf{p}?\Lambda$ *and* $P \leqslant \mathsf{p}?\Lambda'$ *and* $msg(\Lambda) \cap msg(\Lambda') = \emptyset$, *then* $P \leqslant \mathsf{p}?(\Lambda \uplus \Lambda')$.

The main result of this section is that the sessions obtained by decomposition can be typed by the global types obtained by decomposition.

Theorem 9.9.
If $\vdash \mathcal{M} : \mathsf{G}$ *and* $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}, \mathcal{P}, \mathcal{Q}) = (\mathsf{G}_1, \mathsf{G}_2)$ *and* $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathcal{M}, \mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\mathcal{M}_1, \mathcal{M}_2\big)$, *then* $\vdash \mathcal{M}_1 : \mathsf{G}_1$ *and* $\vdash \mathcal{M}_2 : \mathsf{G}_2$.

Proof. It is enough to show that $P \leqslant \mathsf{G}{\restriction}_{\mathsf{p}}$ implies $P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \leqslant (\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}){\restriction}_{\mathsf{p}}$ with $\mathsf{p} \in \mathcal{P}$. The proof is by coinduction on G and by cases on its shape. The case $\mathsf{G} = \mathtt{End}$ is trivial.

If $\mathsf{G} = \mathsf{q} \to \mathsf{p} : \{\ell_i.\mathsf{G}_i \mid {\scriptstyle 1\leq i\leq n}\}$, by Definition 3.3 and rule [SUB-IN] (cf. Definition 3.9) we have

$$P = \mathsf{q}?\{\ell_i.P_i \mid {\scriptstyle 1\leq i\leq m}\} \quad \text{and} \quad \mathsf{G}{\restriction}_{\mathsf{p}} = \mathsf{q}?\{\ell_i.\mathsf{G}_i{\restriction}_{\mathsf{p}} \mid {\scriptstyle 1\leq i\leq n}\}$$

with $n \leq m$ and $P_i \leqslant \mathsf{G}_i{\restriction}_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. By coinduction we get $P_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \leqslant (\mathsf{G}_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}){\restriction}_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. By Definitions 9.6 and 9.1

$$P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \mathsf{s}?\{\ell_i.P_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid {\scriptstyle 1\leq i\leq m}\} \quad \text{and} \quad (\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}){\restriction}_{\mathsf{p}} = \mathsf{s}?\{\ell_i.(\mathsf{G}_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}){\restriction}_{\mathsf{p}} \mid {\scriptstyle 1\leq i\leq n}\}$$

where $\mathsf{s} = \mathsf{q}$ if $\mathsf{q} \in \mathcal{P}$ and $\mathsf{s} = \mathsf{h}$ otherwise. Then we get $P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \leqslant (\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}){\restriction}_{\mathsf{p}}$ by using the rule [SUB-IN] of Definition 3.9.

If $\mathsf{G} = \mathsf{q} \to \mathsf{p} : \Gamma$ the proof is similar to the previous case using the rule [SUB-OUT] of Definition 3.9.

If $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid {\scriptstyle 1\leq i\leq n}\}$ and $\mathsf{p} \notin \{\mathsf{r}, \mathsf{s}\}$, then we have two cases. If $\{\mathsf{r}, \mathsf{s}\} \cap \mathcal{P} = \emptyset$, then by Definition 9.1 $(\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}){\restriction}_{\mathsf{p}} = \widehat{\mathsf{G}}{\restriction}_{\mathsf{p}}$ for some global type $\widehat{\mathsf{G}}$ such that $\widehat{\mathsf{G}} = \mathsf{G}_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}}$ for all i, $1 \leq i \leq n$. By Proposition 9.2 $\mathsf{G}_i{\restriction}_{\mathsf{p}} = \mathsf{G}_1{\restriction}_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. So coinduction applies. Otherwise, when $\{\mathsf{r}, \mathsf{s}\} \cap \mathcal{P} \neq \emptyset$, by Definition 3.3 of projection, we have two subcases:

1. $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{G}_1\upharpoonright_{\mathsf{p}}$ if $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{G}_1\upharpoonright_{\mathsf{p}}$ for all i, $1 \leq i \leq n$;

2. $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{t}?(\Lambda_1 \uplus \ldots \uplus \Lambda_n)$ with $\mathsf{G}_i\upharpoonright_{\mathsf{p}} = \mathsf{t}?\Lambda_i$ for all i, $1 \leq i \leq n$, and $msg(\Lambda_i) \cap msg(\Lambda_j) = \emptyset$ for all i, j, $1 \leq i \neq j \leq n$.

In case 1 the proof is as before. In case 2, from $\Lambda = \Lambda_1 \uplus \ldots \uplus \Lambda_n$ using rule [SUB-IN] of Definition 3.9, we get $\mathsf{G}\upharpoonright_{\mathsf{p}} \leqslant \mathsf{G}_i\upharpoonright_{\mathsf{p}}$, which implies $P \leqslant \mathsf{G}_i\upharpoonright_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. By coinduction $P \Uparrow^{\mathsf{h}}_{\mathcal{P}} \leqslant (\mathsf{G}_i \Uparrow^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. By Definition 9.1 $(\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}} = \mathsf{t}'?(\Lambda'_1 \uplus \ldots \uplus \Lambda'_n)$ and $(\mathsf{G}_i\Uparrow^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}} = \mathsf{t}'?\Lambda'_i$ (where either $\mathsf{t}' = \mathsf{t}$ or $\mathsf{t}' = \mathsf{h}$) and $msg(\Lambda'_i) = msg(\Lambda_i)$ for all i, $1 \leq i \leq n$. We conclude $P \Uparrow^{\mathsf{h}}_{\mathcal{P}} \leqslant (\mathsf{G}\Uparrow^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}}$ by repeated application of Proposition 9.8. □

If $\mathsf{G}_0 = \mathsf{p} \to \mathsf{r} : \ell.\mathsf{p} \to \mathsf{s} : \ell.\mathsf{G}_0$ and $\mathcal{P} = \{\mathsf{p}\}$ and $\mathcal{Q} = \{\mathsf{r}, \mathsf{s}\}$, then we get $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}_0, \mathcal{P}, \mathcal{Q}) = \big(\mathsf{G}, \mathsf{G}'\big)$ where G and G' are as in the Example 8.1. Let $\tilde{P}$, $\tilde{R}$ and $\tilde{S}$ be as in Example 8.7. We have

$$\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{p} \triangleright \tilde{P} \mid \mathsf{r} \triangleright \tilde{R} \mid \mathsf{s} \triangleright \tilde{S}, \mathsf{G}_0, \mathcal{P}, \mathcal{Q}) = \big(\mathsf{p} \triangleright P \mid \mathsf{h} \triangleright H, \mathsf{r} \triangleright R \mid \mathsf{s} \triangleright S \mid \mathsf{k} \triangleright K\big)$$

where P, H, R, S, and K are as in Example 8.1. This shows an instance of Theorem 9.9.

For typed multiparty sessions the decomposition is not the left inverse of the direct composition. For example take

$$\mathcal{M} \equiv \mathsf{p} \triangleright \mathsf{r}!\ell_1 \mid \mathsf{q} \triangleright \mathsf{s}?\ell_2 \mid \mathsf{r} \triangleright \mathsf{p}?\{\ell_1, \ell_3\} \mid \mathsf{s} \triangleright \mathsf{q}!\ell_2$$

and $\mathcal{P} = \{\mathsf{p}, \mathsf{q}\}$ and $\mathcal{Q} = \{\mathsf{r}, \mathsf{s}\}$ and $\mathsf{G} = \mathsf{p} \to \mathsf{r} : \ell_1.\mathsf{s} \to \mathsf{q} : \ell_2$. We get $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathcal{M}, \mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\mathcal{M}_1, \mathcal{M}_2\big)$ where $\mathcal{M}_1 \equiv \mathsf{p} \triangleright \mathsf{h}!\ell_1 \mid \mathsf{q} \triangleright \mathsf{h}?\ell_2 \mid \mathsf{h} \triangleright \mathsf{p}?\ell_1.\mathsf{q}!\ell_2$, $\mathcal{M}_2 \equiv \mathsf{r} \triangleright \mathsf{k}?\{\ell_1, \ell_3\} \mid \mathsf{s} \triangleright \mathsf{k}!\ell_2 \mid \mathsf{k} \triangleright \mathsf{r}!\ell_1.\mathsf{s}?\ell_2$. Then $\vdash \mathcal{M}_1 : \mathsf{G}_1$ and $\vdash \mathcal{M}_2 : \mathsf{G}_2$ where $\mathsf{G}_1 = \mathsf{p} \to \mathsf{h} : \ell_1.\mathsf{h} \to \mathsf{q} : \ell_2$ and $\mathsf{G}_2 = \mathsf{k} \to \mathsf{r} : \ell_1.\mathsf{s} \to \mathsf{k} : \ell_2$. We obtain

$$\mathcal{M}_1 \not\bowtie \mathcal{M}_2 \equiv \mathsf{p} \triangleright \mathsf{r}!\ell_1 \mid \mathsf{q} \triangleright \mathsf{s}?\ell_2 \mid \mathsf{r} \triangleright \mathsf{p}?\ell_1 \mid \mathsf{s} \triangleright \mathsf{q}!\ell_2$$

The difference between $\mathcal{M}$ and $\mathcal{M}_1 \not\bowtie \mathcal{M}_2$ is that participant r in $\mathcal{M}$ waits for a message ℓ_3 from participant p. However, p will never send this message. As in this example, decomposing followed by composing typed sessions eliminates useless inputs. This is due to the replacement of some original processes by the projections of global types into the corresponding participants, see Definition 8.5.

10. Standard Preorder on Processes

We now discuss the impact of adopting a structural preorder on processes, that we denote as $\leqslant^+$, mimicking the standard subtyping relation [16].

The relation $\leqslant^+$ is obtained by replacing [SUB-OUT] with [SUB-OUT$^+$] below in Definition 3.9, hence it allows more outputs in larger processes than in smaller ones.

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

Let $\vdash^+$ be the typing system obtained by using $\leqslant^+$ in rule [T-SESS] (cf. Definition 3.10).

Lemmas 4.1 and 4.2 easily adapts to the system $\vdash^+$.

Lemma 10.1 (Inversion Lemma for $\vdash^+$). *If $\vdash^+ \mathcal{M} : \mathsf{G}$ and $\mathsf{p} \triangleright P \in \mathcal{M}$, then $P \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$.*

Lemma 10.2 (Canonical Form Lemma for $\vdash^+$).
If $\vdash^+ \mathcal{M} : \mathsf{G}$ and $\mathsf{p} \in \mathtt{ptg}(\mathsf{G})$, then there is $\mathsf{p} \triangleright P \in \mathcal{M}$ and $P \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$.

Clearly $\vdash \mathcal{M} : \mathsf{G}$ implies $\vdash^+ \mathcal{M} : \mathsf{G}$, and a weakening of the vice versa is shown below.

Theorem 10.3. *If $\vdash^+ \mathcal{M} : \mathsf{G}$, then $\vdash \mathcal{M} : \mathsf{G}'$ for some G'.*

Proof. The proof is by coinduction on G. Let $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \Gamma$. Then, by Lemma 10.2 we get $\mathcal{M} \equiv \mathsf{p} \triangleright \mathsf{q}!\Lambda \mid \mathsf{q} \triangleright \mathsf{p}?\Lambda' \mid \mathcal{M}'$ and $\mathsf{q}!\Lambda \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$ and $\mathsf{p}?\Lambda' \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{q}}$. By the definition of $\leqslant^+$, $msg(\Lambda) \subseteq msg(\Gamma) \subseteq msg(\Lambda')$. Let $\Lambda = \{\ell_i.P_i \mid {}_{1\leq i\leq n}\}$, $\Gamma = \{\ell_i.\mathsf{G}_i \mid {}_{1\leq i\leq n}\} \uplus \Gamma'$ and $\Lambda' = \{\ell_i.Q_i \mid {}_{1\leq i\leq n}\} \uplus \Lambda''$. By definition of $\leqslant^+$ we get $P_i \leqslant^+ \mathsf{G}_i\upharpoonright_{\mathsf{p}}$ and $Q_i \leqslant^+ \mathsf{G}_i\upharpoonright_{\mathsf{q}}$ for all i, $1 \leq i \leq n$. Then we can derive $\vdash^+ \mathsf{p} \triangleright P_i \mid \mathsf{q} \triangleright Q_i \mid \mathcal{M}' : \mathsf{G}_i$ for all i, $1 \leq i \leq n$. By coinduction there are G'_i such that $\vdash \mathsf{p} \triangleright P_i \mid \mathsf{q} \triangleright Q_i \mid \mathcal{M}' : \mathsf{G}'_i$ for all i, $1 \leq i \leq n$. We can choose $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}'_i \mid {}_{1\leq i\leq n}\}$. □

Thanks to the theorem above, the classes of multiparty sessions that can be typed by $\vdash$ and $\vdash^+$ do coincide.

A first effect of adopting $\leqslant^+$ is a weaker notion of session fidelity than the one for $\leqslant$. This is easily illustrated by the following example: using $\vdash^+$, we can type session $\mathcal{M} = \mathsf{p}\triangleright\mathsf{q}!\ell_1 \mid \mathsf{q}\triangleright\mathsf{p}?\{\ell_1, \ell_2\}$ with $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_1, \ell_2\}$, namely

$\vdash^+ \mathcal{M} : \mathsf{G}$ holds. However $\mathsf{G} \xrightarrow{\mathsf{p}\ell_i\mathsf{q}} \mathtt{End}$ with $i = 1, 2$, while the only reduction of $\mathcal{M}$ is $\mathcal{M} \xrightarrow{\mathsf{p}\ell_1\mathsf{q}} \mathsf{p} \triangleright \mathbf{0}$. Notice that $\mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\{\ell_1, \ell_2\}$ and $\mathsf{G}\upharpoonright_{\mathsf{q}} = \mathsf{p}?\{\ell_1, \ell_2\}$, hence $\mathsf{q}!\ell_1 \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$ but $\mathsf{q}!\ell_1 \not\leqslant \mathsf{G}\upharpoonright_{\mathsf{p}}$.

Lemma 5.5 does not hold for $\vdash^+$, as shown by the following counterexample. If $P = \mathsf{p}!\ell_1$ and $Q = \mathsf{p}!\{\ell_1, \ell_2\}$, then $P \leqslant^+ Q$, but $\mathsf{gw}(P, \mathsf{h}) = \mathsf{h}?\ell_1.\mathsf{p}!\ell_1 \not\geqslant \mathsf{h}?\{\ell_1.\mathsf{p}!\ell_1, \ell_2.\mathsf{p}!\ell_1\} = \mathsf{gw}(Q, \mathsf{h})$. The reason is that $\leqslant^+$ allows larger processes to contain additional messages in send operations, but the gateway construction would introduce additional messages in inputs.

Instead we can show the analogous of Lemmas 5.6 and 6.2 for $\leqslant^+$.

Lemma 10.4. *If $P \leftrightarrow Q$, then $P \leqslant^+ P'$ and $Q \leqslant^+ Q'$ imply $P' \leftrightarrow Q'$.*

Proof. Let us assume

$$\begin{array}{lcll} P = \mathsf{p}!\{\ell_i.P_i \mid {\scriptstyle 1 \le i \le n}\} & \leqslant^+ & P' = \mathsf{p}!\{\ell_i.P'_i \mid {\scriptstyle 1 \le i \le m}\} & \\ \updownarrow & & & \\ Q = \mathsf{q}?\{\ell_i.Q_i \mid {\scriptstyle 1 \le i \le n'}\} & \leqslant^+ & Q' = \mathsf{q}?\{\ell_i.Q'_i \mid {\scriptstyle 1 \le i \le n''}\} & \text{with } n'' \le n' \le n \le m \end{array}$$

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

Lemma 10.5. *If $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ and $\vdash^+ \mathcal{M} : \mathsf{G}$ and $\vdash^+ \mathcal{M}' : \mathsf{G}'$, then*

$$(\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})$$

Proof. The proof mimics that of Lemma 6.2, just using Lemma 10.4 instead of Lemma 5.6. □

A main drawback of rule [SUB-OUT$^+$] is that Theorem 6.11 fails to hold due to the fact that $\vdash^+$ invalidates Lemma 5.5. We can anyway prove a similar result for $\vdash^+$.

Theorem 10.6. *If $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ and $\vdash^+ \mathcal{M} : \mathsf{G}$ and $\vdash^+ \mathcal{M}' : \mathsf{G}'$, then $\vdash^+ \mathcal{M} \bowtie \mathcal{M}' : \mathsf{G}''$ for some G''.*

Proof. Theorem 10.3 implies $\vdash \mathcal{M} : \mathsf{G}_1$ and $\vdash \mathcal{M}' : \mathsf{G}_2$ for some $\mathsf{G}_1, \mathsf{G}_2$. Theorem 6.11 gives $\vdash \mathcal{M} \between \mathcal{M}' : \mathsf{G}_1 \between \mathsf{G}_2$. We can choose $\mathsf{G}'' = \mathsf{G}_1 \between \mathsf{G}_2$. □

Compatibility of global types is not necessary for the type system $\vdash^+$. Let

$$\begin{array}{rclcrcl} \mathsf{G} & = & \mathsf{p} \to \mathsf{h} : \{\ell_1, \ell_2\} & \qquad & \mathsf{G}' & = & \mathsf{k} \to \mathsf{s} : \{\ell_1, \ell_2.\mathsf{s} \to \mathsf{k} : \ell_3\} \\ \mathcal{M} & = & \mathsf{p} \triangleright \mathsf{h}!\ell_1 \mid \mathsf{h} \triangleright \mathsf{p}?\{\ell_1, \ell_2\} & & \mathcal{M}' & = & \mathsf{s} \triangleright \mathsf{k}?\{\ell_1, \ell_2.\mathsf{k}!\ell_3\} \mid \mathsf{k} \triangleright \mathsf{s}!\ell_1 \end{array}$$

Then $\neg\big((\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})\big)$ and $\vdash^+ \mathcal{M} : \mathsf{G}$ and $\vdash^+ \mathcal{M}' : \mathsf{G}'$, but

$$\mathcal{M}'' = \mathsf{p} \triangleright \mathsf{h}!\ell_1 \mid \mathsf{s} \triangleright \mathsf{k}?\{\ell_1, \ell_2.\mathsf{k}!\ell_3\} \mid \mathsf{h} \triangleright \mathsf{p}?\{\ell_1.\mathsf{k}!\ell_1, \ell_2.\mathsf{k}!\ell_2\} \mid \mathsf{k} \triangleright \mathsf{h}?\ell_1.\mathsf{s}!\ell_1$$

reduces only as follows:

$$\begin{array}{rcl} \mathcal{M}'' & \xrightarrow{\mathsf{p}\ell_1\mathsf{h}} & \mathsf{s} \triangleright \mathsf{k}?\{\ell_1, \ell_2.\mathsf{k}!\ell_3\} \mid \mathsf{h} \triangleright \mathsf{k}!\ell_1 \mid \mathsf{k} \triangleright \mathsf{h}?\ell_1.\mathsf{s}!\ell_1 \\ & \xrightarrow{\mathsf{h}\ell_1\mathsf{k}} & \mathsf{s} \triangleright \mathsf{k}?\{\ell_1, \ell_2.\mathsf{k}!\ell_3\} \mid \mathsf{k} \triangleright \mathsf{s}!\ell_1 \\ & \xrightarrow{\mathsf{k}\ell_1\mathsf{s}} & \mathsf{s} \triangleright \mathbf{0} \end{array}$$

As this example shows, if the cause of $\neg\big((\mathsf{G}, \mathsf{h}) \leftrightarrow (\mathsf{G}', \mathsf{k})\big)$ occurs after an output that is not present in $\mathcal{M}$ and $\mathcal{M}'$, then no deadlock arises by composing them via gateways.

Instead we can show that the direct composition of multiparty sessions can be typed using the direct composition of global types, as in the system $\vdash$, and differently with respect to the composition via gateways. We define $\mathcal{M} \between\!\!\!\!\!\!+\!\!\!\!\!\!\between \mathcal{M}'$ as $\mathcal{M} \between\!\!\!\!\!\!\between \mathcal{M}'$ (cf. Definition 8.5), but using $\vdash^+$ instead of $\vdash$.

Theorem 10.7. *If* $(\mathcal{M}, \mathsf{h}) \leftrightarrow (\mathcal{M}', \mathsf{k})$ *and* $\vdash^+ \mathcal{M} : \mathsf{G}$ *and* $\vdash^+ \mathcal{M}' : \mathsf{G}'$*, then* $\vdash^+ \mathcal{M} \between\!\!\!\!\!\!+\!\!\!\!\!\!\between \mathcal{M}' : \mathsf{G} \between\!\!\!\!\!\!\between \mathsf{G}'$.

Proof. The proof is as the proof of Theorem 8.6 using Lemma 10.5 instead of Lemma 6.2. □

The decomposition of global types and multiparty sessions is insensible to the difference between $\leqslant$ and $\leqslant^+$ as well.

Proposition 10.8. *If* $P \leqslant^+ \mathsf{p}?\Lambda$ *and* $P \leqslant^+ \mathsf{p}?\Lambda'$ *and* $msg(\Lambda) \cap msg(\Lambda') = \emptyset$*, then* $P \leqslant^+ \mathsf{p}?(\Lambda \uplus \Lambda')$.

Theorem 10.9.
If $\vdash^+ \mathcal{M} : \mathsf{G}$, $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathsf{G}, \mathcal{P}, \mathcal{Q}) = (\mathsf{G}_1, \mathsf{G}_2)$ *and* $\textsc{Dec}^{\mathsf{h},\mathsf{k}}(\mathcal{M}, \mathsf{G}, \mathcal{P}, \mathcal{Q}) = \big(\mathcal{M}_1, \mathcal{M}_2\big)$, *then* $\vdash^+ \mathcal{M}_1 : \mathsf{G}_1$ *and* $\vdash^+ \mathcal{M}_2 : \mathsf{G}_2$.

Proof. Like in the proof of Theorem 9.9, it is enough to show that $P \leqslant^+ \mathsf{G}\upharpoonright_{\mathsf{p}}$ implies $P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \leqslant^+ (\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}}$ with $\mathsf{p} \in \mathcal{P}$. Here we show only the cases whose proofs differ from those considered in Theorem 9.9.

If $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_i.\mathsf{G}_i \mid {\scriptstyle 1 \leq i \leq n}\}$, by Definition 3.3 we have

$$P = \mathsf{q}!\{\ell_i.P_i \mid {\scriptstyle 1 \leq i \leq n}\} \quad \text{and} \quad \mathsf{G}\upharpoonright_{\mathsf{p}} = \mathsf{q}!\{\ell_i.\mathsf{G}_i\upharpoonright_{\mathsf{p}} \mid {\scriptstyle 1 \leq i \leq m}\}$$

with $n \leq m$ and $P_i \leqslant^+ \mathsf{G}_i\upharpoonright_{\mathsf{p}}$ for all i, $1 \leq i \leq n$, by [SUB-OUT$^+$] (cf. rule (1)). By coinduction, $P_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \leqslant^+ (\mathsf{G}_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}}$ for all i, $1 \leq i \leq n$. By Definitions 9.6 and 9.1

$$P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} = \mathsf{s}!\{\ell_i.P_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \mid {\scriptstyle 1 \leq i \leq n}\} \quad \text{and} \quad (\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}} = \mathsf{s}!\{\ell_i.(\mathsf{G}_i{\Uparrow}^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}} \mid {\scriptstyle 1 \leq i \leq m}\}$$

where $\mathsf{s} = \mathsf{q}$ if $\mathsf{q} \in \mathcal{P}$ and $\mathsf{s} = \mathsf{h}$ otherwise. Then we get $P{\Uparrow}^{\mathsf{h}}_{\mathcal{P}} \leqslant^+ (\mathsf{G}{\Uparrow}^{\mathsf{h}}_{\mathcal{P}})\upharpoonright_{\mathsf{p}}$ by using rule [SUB-OUT$^+$].

If $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\ell_i.\mathsf{G}_i \mid {\scriptstyle 1 \leq i \leq n}\}$ the proof is as for Theorem 9.9, but using Proposition 10.8 instead of Proposition 9.8.

□

11. Concluding Remarks, Related Work, and Future Developments

The distinguishing feature of an *open* system of concurrent components is its capacity of communicating with the "outside", i.e. with some environment of the system. This ability provides means for composing open systems into larger systems (which may still be open). In order to compose systems "safely", it is common practice to rely on interface descriptions.
MPST systems [24, 12, 31] are usually assumed to be *closed*, since all the components needed for the functioning of the system must be already there. As a matter of fact, the MPST framework does work fairly well for the design of closed systems, but does not possess the flexibility open systems can offer. In [1, 2] 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 composed. The composition mechanism of [1, 2] uses suitable forwarders, dubbed "gateways", for this purpose. The gateways are automatically synthesised out of the compatible interfaces and the composition of two systems simply consists in replacing the latter by the former.

In the present paper we have provided a MPST formalism where global types are used to describe the overall behaviour of "safe" systems of processes – dubbed *multiparty sessions* – directly related to the global types projections. We have then adapted and extended to such a formalism the approach of [1, 2], both at the global type and multiparty session levels.
In general, managing to look at global types as overall descriptions of open systems results in a MPST-based approach to the modular design of systems. From another point of view, by means of such an approach, one could develop systems where some participants, instead of representing actual processes, describe API calls, along the line of what some researchers refer to as Behavioural API [4]. Moreover, this approach is helpful also after the system implementation phase. Let us assume to have a system developed using a 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 composition with the other system, and the composition operation on global types enables to get a global view of what is going on in the resulting system.

Our calculus of multiparty sessions is like those of [17, 21], but for the use of coinduction instead of induction which is inspired by [10, 34]. As in [34] we get rid of local types, which in many calculi are similar to processes [11, 17, 21]. The syntax of global types is the coinductive version of the standard syntax [24] and the notion of projection in Definition 3.3 is an extension of both the standard projection [24] and the projection given in [34]. Our global types assure lock-freedom of multiparty sessions.

As mentioned above, a relevant feature of our formalism is that the composition by gateways operation on systems can be lifted to the level of global types. In [1, 2], where systems of CFSMs were considered, such a lifting was

done by extending the syntax of global descriptions with a *new* symbol, whose semantics is indeed the composition by gateways at system level. Instead, in the present paper, the session obtained by composition is represented using the standard syntax of global types. Moreover, we have shown that the compatibility relation of [1, 2], which requires duality, can be relaxed to a relation very close to session types subtyping [20, 16]. We further investigated the notion of compatibility by also showing that compatibility of global types is actually also a necessary condition in order to get a "safe" composition of systems.

Our formalism is equipped with a structural preorder on processes akin to the subtyping relation between session types of [11], which in turn is a restriction of the subtyping of [16]. This choice is justified by the fact that the subtyping of [16] allows process substitution, while the subtyping of [20] allows channel substitution, as observed in [19].

We have also shown that the composition by gateways technique can be extended so that the gateways can be actually dropped. This alternative approach has led to the definition of another composition operation that we dubbed direct. The use of direct composition - in particular at type level - yields a powerful tool. In fact, it enables the modular design and development of distributed systems, expecially in case such an operation can be paired – as we did – with another one enabling to decompose a global description. Small systems can be separately implemented out of the types obtained by decomposing a global type. Then by direct composition such systems can be merged into a system implementing the original global description, as shown by our results (some constraints have to be imposed on the global type to be decomposed).

A final contribution of the present paper is the discussion on how the properties of the given constructions change when we consider a structural preorder on processes mimicking the subtyping relation of [16].

In [29, 28] global types are built out of several local types (under certain conditions). We also aim at getting global types, but we obtain them from the global types describing the two systems which are composed.

The dynamic addition/removal of participants (they can join/leave the session after it has been set up) is supported in the calculus of [25] and recently in [6]. In [25] 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 decided in advance. This is also the point of view adopted in [6], however the

type system proposed in [6] aims to guarantee data-flow properties instead of control properties. For instance, in [6] well-typed systems may be not lock-free. Both those approaches to the dynamic system extension issue look orthogonal to the work presented here.

Another approach to safe system composition allows one to replace a fragment of a choreography (essentially a program structured as a multiparty session type) with a new one at run-time [15], possibly also introducing new participants [18]. However, the new fragment could not communicate by exchanging messages with the context choreography, but only through shared state. Also, the change may deeply impact all the participants, and a complex middleware managing the change is needed. Hence, this approach is orthogonal to ours as well.

Both arbiter processes [8] and mediums [7] 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 composing two multiparty systems. Nonetheless, our work could be further developed in the logical context of [8]: in the logical interpretation of multiparty sessions one could introduce a "composition cut" corresponding to a sort of composition-by-gateways operator. Then the good properties of the system corresponding to the proof containing the cut should be guaranteed by proving that the "composition cut" is actually an admissible rule. The proof should consist in a "composition cut elimination" procedure corresponding to our direct composition on global types, which bypasses the use of gateways.

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 that compatibility could be extended, since the subtyping for asynchronous multiparty sessions is more permissive than the subtyping for the synchronous ones [31]. Of course this extension requires care, being the subtyping of [31] undecidable, as shown in [5, 30].

The composition via gateways proposed by the authors of [1, 2] and exploited in the present paper in a multiparty session 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 composition to a multiple interfaces one. Such an extension, however, immediately reveals itself to be unsound: by composing via gateways more than one pair of compatible interfaces one could obtain a deadlocked system. A

very simple example for that is

$$\mathsf{G} = \mathsf{p} \to \mathsf{h} : \ell \quad \text{and} \quad \mathsf{G}' = \mathsf{k} \to \mathsf{s} : \ell$$

By projection we get the systems

$$\mathcal{M} = \mathsf{p} \triangleright \mathsf{h}!\ell \mid \mathsf{h} \triangleright \mathsf{p}?\ell \quad \text{and} \quad \mathcal{M}' = \mathsf{k} \triangleright \mathsf{s}!\ell \mid \mathsf{s} \triangleright \mathsf{k}?\ell$$

It is immediate to check that p and s are compatible, as well as h and k. Simultaneously connecting $\mathcal{M}$ and $\mathcal{M}'$ through both the compatible pairs (p, s) and (h, k) would result in the following deadlocked system

$$\mathsf{p} \triangleright \mathsf{s}?\ell.\mathsf{h}!\ell \quad | \quad \mathsf{h} \triangleright \mathsf{p}?\ell.\mathsf{k}!\ell \quad | \quad \mathsf{k} \triangleright \mathsf{h}?\ell.\mathsf{s}!\ell \quad | \quad \mathsf{s} \triangleright \mathsf{k}?\ell.\mathsf{p}!\ell$$

(This example is similar to one developed in the CFSMs setting of [2]).
It is easy to realise that this sort of difficulty stays the same also in case direct composition is considered. 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 [13] could be investigated in future for such an aim.

Acknowledgments. We gratefully acknowledge the fruitful interactions and communications with the anonymous referees through the ICE Forum and thank them for their reports.

References

[1] Barbanera, F., de'Liguoro, U., Hennicker, R., 2018. Global Types for Open Systems, in: ICE, Open Publishing Association. pp. 4–20.

[2] Barbanera, F., de'Liguoro, U., Hennicker, R., 2019. Connecting Open Systems of Communicating Finite State Machines. Journal of Logical and Algebraic Methods in Programming 109, 1–34.

[3] Barbanera, F., Dezani-Ciancaglini, M., 2019. Open Multiparty Sessions, in: ICE, Open Publishing Association. pp. 77–96.

[4] BEHAPI website, 2019. Behapi website. `https://www.um.edu.mt/projects/behapi/`.

[5] Bravetti, M., Carbone, M., Zavattaro, G., 2017. Undecidability of Asynchronous Session Subtyping. Information and Computation 256, 300–320.

[6] Bruni, R., Corradini, A., Gadducci, F., Melgratti, H.C., Montanari, U., Tuosto, E., 2019. Data-Driven Choreographies à la Klaim, in: Models, Languages, and Tools for Concurrent and Distributed Programming, Springer. pp. 170–190. Essays Dedicated to Rocco De Nicola on the Occasion of His 65th Birthday.

[7] Caires, L., Pérez, J.A., 2016. Multiparty Session Types Within a Canonical Binary Theory, and Beyond, in: FORTE, Springer. pp. 74–95.

[8] Carbone, M., Lindley, S., Montesi, F., Schürmann, C., Wadler, P., 2016. Coherence Generalises Duality: A Logical Explanation of Multiparty Session Types, in: CONCUR, Schloss Dagstuhl. pp. 33:1–33:15.

[9] Cardone, F., Coppo, M., 2013. Recursive Types, in: Barendregt, H., Dekkers, W., Statman, R. (Eds.), Lambda Calculus with Types. Cambridge University Press. Perspectives in Logic, pp. 377–576.

[10] Castagna, G., Gesbert, N., Padovani, L., 2009. A Theory of Contracts for Web Services. ACM Transactions on Programming Languages and Systems 31, 19:1–19:61.

[11] Castellani, I., Dezani-Ciancaglini, M., Giannini, P., 2019. Reversible Sessions with Flexible Choices. Acta Informatica , 553–583.

[12] Coppo, M., Dezani-Ciancaglini, M., Padovani, L., Yoshida, N., 2015. A Gentle Introduction to Multiparty Asynchronous Session Types, in: Formal Methods for Multicore Programming, Springer. pp. 146–178.

[13] Coppo, M., Dezani-Ciancaglini, M., Yoshida, N., Padovani, L., 2016. Global Progress for Dynamically Interleaved Multiparty Sessions. Mathematical Structures in Computer Science 26, 238–302.

[14] Courcelle, B., 1983. Fundamental Properties of Infinite Trees. Theoretical Computer Science 25, 95–169.

[15] Dalla Preda, M., Gabbrielli, M., Giallorenzo, S., Lanese, I., Mauro, J., 2017. Dynamic Choreographies: Theory And Implementation. Logical Methods in Computer Science 13, 1–57.

[16] Demangeon, R., Honda, K., 2011. Full Abstraction in a Subtyped Pi-Calculus with Linear Types, in: CONCUR, Springer. pp. 280–296.

[17] Dezani-Ciancaglini, M., Ghilezan, S., Jaksic, S., Pantovic, J., Yoshida, N., 2015. Precise Subtyping for Synchronous Multiparty Sessions, in: PLACES, Open Publishing Association. pp. 29–43.

[18] Gabbrielli, M., Giallorenzo, S., Lanese, I., Mauro, J., 2019. Guess Who's Coming: Runtime Inclusion of Participants in Choreographies, in: The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy - Essays Dedicated to Catuscia Palamidessi on the Occasion of Her 60th Birthday, Springer. pp. 118–138.

[19] Gay, S., 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, Springer. pp. 95–108.

[20] Gay, S., Hole, M., 2005. Subtyping for Session Types in the Pi Calculus. Acta Informatica 42, 191–225.

[21] Ghilezan, S., Jaksic, S., Pantovic, J., Scalas, A., Yoshida, N., 2019. Precise Subtyping for Synchronous Multiparty Sessions. Journal of Logic and Algebraic Methods in Programming 104, 127–173.

[22] Honda, K., Vasconcelos, V.T., Kubo, M., 1998. Language Primitives and Type Discipline for Structured Communication-Based Programming, in: ESOP, Springer. pp. 122–138.

[23] Honda, K., Yoshida, N., Carbone, M., 2008. Multiparty Asynchronous Session Types, in: POPL, ACM Press. pp. 273–284.

[24] Honda, K., Yoshida, N., Carbone, M., 2016. Multiparty Asynchronous Session Types. Journal of the ACM 63, 9:1–9:67.

[25] Hu, R., Yoshida, N., 2017. Explicit Connection Actions in Multiparty Session Types, in: FASE, Springer. pp. 116–133.

[26] Kobayashi, N., 2002. A Type System for Lock-Free Processes. Information and Computation 177, 122–159.

[27] Kozen, D., Silva, A., 2017. Practical Coinduction. Mathematical Structures in Computer Science 27, 1132–1152.

[28] Lange, J., 2014. On the Synthesis of Choreographies. Ph.D. thesis. Department of Computer Science, University of Leicester.

[29] Lange, J., Tuosto, E., 2012. Synthesising Choreographies from Local Session Types, in: CONCUR, Springer. pp. 225–239.

[30] Lange, J., Yoshida, N., 2017. On the Undecidability of Asynchronous Session Subtyping, in: FOSSACS, Springer. pp. 441–457.

[31] Mostrous, D., Yoshida, N., Honda, K., 2009. Global Principal Typing in Partially Commutative Asynchronous Sessions, in: ESOP, Springer. pp. 316–332.

[32] Padovani, L., 2014. Deadlock and Lock Freedom in the Linear π-calculus, in: LICS, pp. 72:1–72:10.

[33] Pierce, B.C., 2002. Types and Programming Languages. MIT Press.

[34] Severi, P., Dezani-Ciancaglini, M., 2019. Observational Equivalence for Multiparty Sessions. Fundamenta Informaticae 167, 267–305.