Reversible Multiparty Sessions with Checkpoints*

Mariangiola Dezani-Ciancaglini † Paola Giannini‡

Reversible interactions model different scenarios, like biochemical systems and human as well as automatic negotiations. We abstract interactions via multiparty sessions enriched with named checkpoints. Computations can either go forward or roll back to some checkpoints, where possibly different choices may be taken. In this way communications can be undone and different conversations may be tried. Interactions are typed with global types, which control also rollbacks. Typeability of session participants in agreement with global types ensures session fidelity and progress of reversible communications.

1 Introduction

Reversibility is an essential feature in the construction of reliable systems. If a system reaches an undesired state, some actions may be undone and the computation may be restarted from a consistent state. Several studies, see [6, 15, 16, 20], have investigated the theoretical foundations of reversible computations. The relevance of these papers for our work is briefly discussed at the beginning of Section 5.

Our focus is in the context of structured communications, more precisely multiparty sessions, see [12, 13]. The choreography of communications is described by global types, which are projected on the participants to get their interaction patterns [5, 10]. In order to fix the points of the computations we may revert to, we add checkpoints to the syntax of global types. In contrast to previous work, see [2, 22, 23], our checkpoints are named and rollbacks specify the name of the checkpoints to which we revert.

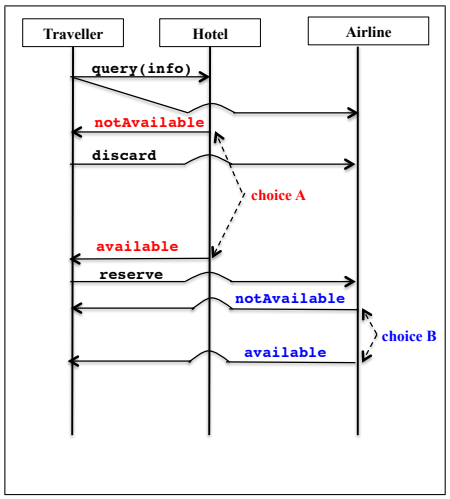

Figure 1: Traveller planning a trip.

We illustrate our approach by discussing an example. Consider the UML sequence diagram of Figure 1. In this example, there are three interacting participants, named Traveller (`Tr`), Hotel (`Ht`) and Airline (`Al`), that establish a session. `Tr`, planning a trip, sends a message labelled `query`, to `Ht` and to `Al` with the details of his journey (abstracted as a string `info`). `Ht` answers to `Tr` with either the message `notAvailable` or `available`. If `Ht` answers `notAvailable`, `Tr` sends to `Al` a message `discard` saying to ignore the previous `query`. If `Ht` answers `available`, then `Tr` send a message to `Al` asking to `reserve` flights for the journey, to which `Al` answers to `Tr` with either the message `notAvailable` or `available`.

The choice made by `Ht`, named A, and the one made by `Al`, named B, are checkpointed choices. This means that the computation could revert to one of these points of the interaction and the given participant could make a different choice. Rolling back to a choice point involves all the participants which crossed this choice point. Moreover, rollback may happen only when all the participants that have this choice point in their future have crossed it. Typing ensures that not involved participants are terminated.

*Partially supported by EU H2020-644235 Rephrase project, EU H2020-644298 HyVar project, ICT COST Actions IC1201 BETTY, IC1402 ARVI and Ateneo/CSP project RunVar.

†Dipartimento di Informatica, Università di Torino, dezani@di.unito.it

‡Computer Science Institute, DiSIT, Università del Piemonte Orientale, paola.giannini@uniupo.it

Assume that Ht, after sending the message available to Tr, discovers that instead for the required dates it is fully booked, and wants to roll back to the choice point A. Before rolling back, it has to make sure that Tr has sent the reserve message to Al and Al has received the message. So that, they can all go back to the interaction before A. Otherwise, if Tr has not sent the reserve message, and Ht goes back to sending the message notAvailable to Tr which is not expecting a message from Ht, there could be an unpleasant misunderstanding, which is formally represented by a "stuck" computation.

Outline Section 2 introduces our calculus, completed by the type system of Section 3. Subject reduction, session fidelity and progress are proved in Section 4. Section 5 discusses related papers and future work.

2 Calculus

In this section we introduce the syntax and the semantics of multiparty sessions with *named checkpoints*.

Syntax A *multiparty session* is a series of interactions between a fixed number of participants, possibly with branching and recursion [13].

We use the following base sets: *values*, ranged over by $v, v', \ldots$; *expressions*, ranged over by $\mathsf{e}, \mathsf{e}', \ldots$; *expression variables*, ranged over by $x, y, z \ldots$; *labels*, ranged over by $\ell, \ell', \ldots$; *checkpoint names*, ranged over by $A, B, \ldots$; *session participants*, ranged over by $\mathsf{p}, \mathsf{q}, \ldots$; *process variables*, ranged over by $X, Y, \ldots$; *processes*, ranged over by $P, Q, \ldots$; *configurations*, ranged over by $\mathbb{C}, \mathbb{C}', \ldots$; *multiparty sessions*, ranged over by $\mathbb{M}, \mathbb{M}', \ldots$; *networks*, ranged over by $\mathbb{N}, \mathbb{N}', \ldots$.

Our processes are obtained from the processes of [10] by adding named checkpoints before external and internal choices.

Definition 2.1 Processes *are defined by:*

$$P ::= \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{e}_i).P_i \mid \blacktriangle_A \sum_{j\in J} \mathsf{p}?\ell_j(x_j).P_j \mid \blacktriangle_A \bigoplus_{j\in J} \mathsf{p}!\ell_j(\mathsf{e}_j).P_j \mid \mu X.P \mid X \mid \mathbf{0}$$

We say that $\blacktriangle_A P$ is a *process checkpointed by A*.

The input process $\sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i$ waits for a value and a label ℓ_i with $i \in I$ from participant p and the output process $\bigoplus_{i\in I} \mathsf{q}!\ell_i(\mathsf{e}_i).P_i$ sends the value of an expression e_i and a label ℓ_i with $i \in I$ to participant q. Checkpointed input and output processes behave in a similar way, except that, when sending/reading a message the checkpointed process is memorised, and can be executed again after a rollback. As usual, in writing processes we omit trailing $\mathbf{0}$'s, and empty parameters.

Example 2.2 *Consider the example of Figure 1. The processes associated with the participants* Tr, Ht, *and* Al *are defined as follows (we abbreviate the labels of messages with their first two consonants):*

$$\begin{array}{lcl} P_{\mathtt{Tr}} & = & \mathtt{Ht!qr(in).Al!qr(in).}P'_{\mathtt{Tr}} \text{ where } P'_{\mathtt{Tr}} = \blacktriangle_A \sum\{\mathtt{Ht?nAv.Al!ds}\ ,\ \mathtt{Ht?av.Al!rs.}\blacktriangle_B \sum\{\mathtt{Al?nAv}\ ,\ \mathtt{Al?av}\}\} \\ P_{\mathtt{Ht}} & = & \mathtt{Tr?qr(}x\mathtt{).}\blacktriangle_A \bigoplus\{\mathtt{Tr!nAv}\ ,\ \mathtt{Tr!av}\} \\ P_{\mathtt{Al}} & = & \mathtt{Tr?qr(}x\mathtt{).}P'_{\mathtt{Al}} \text{ where } P'_{\mathtt{Al}} = \blacktriangle_A \sum\{\mathtt{Tr?ds}\ ,\ \mathtt{Tr?rs.}\blacktriangle_B \bigoplus\{\mathtt{Tr!nAv}\ ,\ \mathtt{Tr!av}\}\} \end{array}$$

In order to allow backward reductions, the configurations of session participants contain both active processes and sequences of checkpointed internal and external choices, denoting the processes that should run in case of rollbacks.

Definition 2.3 Configurations*, ranged over by* $\mathbb{C}$*, are pairs* $R \prec P$*, where* P *is a process,* the active process*, and*

$$R ::= \varepsilon \mid R \cdot \blacktriangle_A P$$

is a (possibly empty) sequence of checkpointed processes, dubbed checkpointed sequence*.*

In the sequence $R \cdot P$ we call P the *top process*.

Multiparty sessions, ranged over by $\mathbb{M}$, are parallel compositions of pairs participant/configuration (denoted by $\mathsf{p} \triangleleft \mathbb{C}$):

$$\mathbb{M} ::= \mathsf{p} \triangleleft \mathbb{C} \mid \mathbb{M} \mid \mathbb{M}$$

Networks, ranged over by $\mathbb{N}$, are parallel composition of sessions:

$$\mathbb{N} ::= \mathbb{M} \mid \mathbb{N} \| \mathbb{N}$$

Operational Semantics The LTS of configurations is given in Figure 2. The forward rules are as expected, only internal choices with more than one branch can silently reduce. When the active process crosses a checkpoint, it is memorised at the top of the checkpointed sequence (rules [CkChc] and [CkRcv]). The backward rule can choose as the new active process an arbitrary process P in the current checkpointed sequence: the name of the checkpoint of P decorates the transition (rule [RbP]). This is essential in order to guarantee that the backward reduction of multiparty sessions produces well-behaved sessions, see rule [RbM] in Figure 3.

$$R \prec \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{e}_i).P_i \xrightarrow{\tau} R \prec \mathsf{p}!\ell_k(\mathsf{e}_k).P_k \quad k \in I \neq \{k\} \quad [\text{Chc}]$$

$$R \prec {}_{\blacktriangle_A}\bigoplus_{j\in J} \mathsf{p}!\ell_j(\mathsf{e}_j).P_j \xrightarrow{\tau} R \cdot {}_{\blacktriangle_A}\bigoplus_{j\in J} \mathsf{p}!\ell_j(x_j).P_j \prec \mathsf{p}!\ell_k(\mathsf{e}_k).P_k \quad k \in J \neq \{k\} \quad [\text{CkChc}]$$

$$R \prec \mathsf{p}!\ell(\mathsf{e}).P \xrightarrow{\mathsf{p}!\ell(v)} R \prec P \quad \mathsf{e} \downarrow v \quad [\text{Snd}]$$

$$R \prec \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \xrightarrow{\mathsf{p}?\ell_j(v)} R \prec P_j\{v/x\} \quad j \in I \quad [\text{Rcv}]$$

$$R \prec {}_{\blacktriangle_A}\sum_{j\in J} \mathsf{p}?\ell_j(x_j).P_j \xrightarrow{\mathsf{p}?\ell_k(v)} R \cdot {}_{\blacktriangle_A}\sum_{j\in J} \mathsf{p}?\ell_j(x_j).P_j \prec P_k\{v/x\} \quad k \in J \quad [\text{CkRcv}]$$

$$R \cdot {}_{\blacktriangle_A}P \cdot R' \prec P' \xrightarrow{A} R \prec {}_{\blacktriangle_A}P \quad [\text{RbP}]$$

Figure 2: Reduction rules of configurations.

The operational semantics of sessions and network is shown in Figure 3, where α ranges over $\tau, \mathsf{p}!\ell(v), \mathsf{p}?\ell(v), A$. This semantics relies on a structural equivalence $\equiv$ for which the parallel operators $\mid$ and $\|$ are commutative and associative and have $\mathsf{p} \triangleleft \varepsilon \prec \mathbf{0}$ as neutral element.

The only interesting rule is rule [RbM]. In this rule we use the mapping $\mathscr{A}$, that associates to a configuration the set of the checkpoint names of processes belonging to its checkpointed sequence. Formally:

$$\mathscr{A}(R \prec \mathbf{0}) = \mathscr{A}(R) \qquad \mathscr{A}(\varepsilon) = \emptyset \qquad \mathscr{A}(R \cdot {}_{\blacktriangle_A}P) = \mathscr{A}(R) \cup \{A\}$$

This mapping is defined only for configurations having $\mathbf{0}$ as their active process. This is enough since the typing rules ensure that the processes which did not traverse some checkpoints are terminated. A multiparty session can roll back to processes at the checkpoint named A only if all the sets $\mathscr{A}$ of the configurations which remain unchanged are defined (i.e. $\mathbf{0}$ is the active process of these configurations) and they do not contain A.

In networks the different sessions reduce independently. For this reason the same participant can interact in different sessions belonging to the same network. We use $\xrightarrow{\tau}{}^*$ to denote the transitive and reflexive closure of the $\xrightarrow{\tau}$ relation.

$$\frac{\mathbb{C} \xrightarrow{\alpha} \mathbb{C}'}{\mathsf{p} \lhd \mathbb{C} \xrightarrow{\alpha} \mathsf{p} \lhd \mathbb{C}'}\ [\text{PC}] \qquad \frac{\mathbb{M} \xrightarrow{\tau} \mathbb{M}'}{\mathbb{M} \mid \mathbb{M}'' \xrightarrow{\tau} \mathbb{M}' \mid \mathbb{M}''}\ [\text{PRM}] \qquad \frac{\mathbb{M}_1 \equiv \mathbb{M}_1' \quad \mathbb{M}_1' \xrightarrow{\tau} \mathbb{M}_2' \quad \mathbb{M}_2' \equiv \mathbb{M}_2}{\mathbb{M}_1 \xrightarrow{\tau} \mathbb{M}_2}\ [\text{EQM}]$$

$$\frac{\mathsf{p} \lhd \mathbb{C}_\mathsf{p} \xrightarrow{\mathsf{q}!\ell(v)} \mathsf{p} \lhd \mathbb{C}_\mathsf{p}' \qquad \mathsf{q} \lhd \mathbb{C}_\mathsf{q} \xrightarrow{\mathsf{p}?\ell(v)} \mathsf{q} \lhd \mathbb{C}_\mathsf{q}'}{\mathsf{p} \lhd \mathbb{C}_\mathsf{p} \mid \mathsf{q} \lhd \mathbb{C}_\mathsf{q} \xrightarrow{\tau} \mathsf{p} \lhd \mathbb{C}_\mathsf{p}' \mid \mathsf{q} \lhd \mathbb{C}_\mathsf{q}'}\ [\text{COM}]$$

$$\frac{\mathsf{p}_i \lhd \mathbb{C}_{\mathsf{p}_i} \xrightarrow{A} \mathsf{p}_i \lhd \mathbb{C}_{\mathsf{p}_i}' \ \ \forall i \in I \qquad A \notin \mathscr{A}(\mathbb{C}_{\mathsf{q}_j}) \ \ \forall j \in J}{\Pi_{i\in I}\mathsf{p}_i \lhd \mathbb{C}_{\mathsf{p}_i} \mid \Pi_{j\in J}\mathsf{q}_j \lhd \mathbb{C}_{\mathsf{q}_j} \xrightarrow{\tau} \Pi_{i\in I}\mathsf{p}_i \lhd \mathbb{C}_{\mathsf{p}_i}' \mid \Pi_{j\in J}\mathsf{q}_j \lhd \mathbb{C}_{\mathsf{q}_j}}\ [\text{RBM}]$$

$$\frac{\mathbb{N} \xrightarrow{\tau} \mathbb{N}'}{\mathbb{N} \| \mathbb{N}'' \xrightarrow{\tau} \mathbb{N}' \| \mathbb{N}''}\ [\text{PRN}] \qquad \frac{\mathbb{N}_1 \equiv \mathbb{N}_1' \quad \mathbb{N}_1' \xrightarrow{\tau} \mathbb{N}_2' \quad \mathbb{N}_2' \equiv \mathbb{N}_2}{\mathbb{N}_1 \xrightarrow{\tau} \mathbb{N}_2}\ [\text{EQN}]$$

Figure 3: Reduction rules of sessions and networks.

Example 2.4 *Consider the processes of Example 2.2. We first give some reductions possible for the configurations of the three participants starting from empty checkpointed sequences.*

$$\begin{array}{rll}
\varepsilon \prec P_{\mathtt{Tr}} \xrightarrow{\mathtt{Ht!qr(in)}} \varepsilon \prec \mathtt{Al!qr(in)}.P'_{\mathtt{Tr}} = \mathbb{C}_1 & [\text{SND}] & (1)\\
\xrightarrow{\mathtt{Al!qr(in)}} \varepsilon \prec P'_{\mathtt{Tr}} = \mathbb{C}_2 & [\text{SND}] & (2)\\
\xrightarrow{\mathtt{Ht?av}} P'_{\mathtt{Tr}} \prec \mathtt{Al!rs}.(\blacktriangle_B \sum\{\mathtt{Al?nAv}\ ,\ \mathtt{Al?av}\}) = \mathbb{C}_3 & [\text{CKRCV}] & (3)\\
\xrightarrow{\mathtt{Al!rs}} P'_{\mathtt{Tr}} \prec \blacktriangle_B \sum\{\mathtt{Al?nAv}\ ,\ \mathtt{Al?av}\} = \mathbb{C}_4 & [\text{SND}] & (4)\\
\xrightarrow{\mathtt{Al?nAv}} P'_{\mathtt{Tr}} \cdot (\blacktriangle_B \sum\{\mathtt{Al?nAv}\ ,\ \mathtt{Al?av}\}) \prec \mathbf{0} = \mathbb{C}_5 & [\text{CKRCV}] & (5)\\
\varepsilon \prec P_{\mathtt{Ht}} \xrightarrow{\mathtt{Tr?qr(in)}} \varepsilon \prec \blacktriangle_A \bigoplus\{\mathtt{Tr!nAv}, \mathtt{Tr!av}\} = \mathbb{C}_6 & [\text{RCV}] & (6)\\
\xrightarrow{\tau} \blacktriangle_A \bigoplus\{\mathtt{Tr!nAv}, \mathtt{Tr!av}\} \prec \mathtt{Tr!av} = \mathbb{C}_7 & [\text{CKCHC}] & (7)\\
\xrightarrow{\mathtt{Tr!av}} \blacktriangle_A \bigoplus\{\mathtt{Tr!nAv}, \mathtt{Tr!av}\} \prec \mathbf{0} = \mathbb{C}_8 & [\text{SND}] & (8)\\
\varepsilon \prec P_{\mathtt{Al}} \xrightarrow{\mathtt{Tr?qr(in)}} \varepsilon \prec P'_{\mathtt{Al}} = \mathbb{C}_9 & [\text{RCV}] & (9)\\
\xrightarrow{\mathtt{Tr?rs}} P'_{\mathtt{Al}} \prec \blacktriangle_B \bigoplus\{\mathtt{Tr!nAv}, \mathtt{Tr!av}\} = \mathbb{C}_{10} & [\text{CKRCV}] & (10)\\
\xrightarrow{\tau} P'_{\mathtt{Al}} \cdot \blacktriangle_B \bigoplus\{\mathtt{Tr!nAv}, \mathtt{Tr!av}\} \prec \mathtt{Tr!av} = \mathbb{C}_{11} & [\text{CKCHC}] & (11)\\
\xrightarrow{\mathtt{Tr!av}} P'_{\mathtt{Al}} \cdot \blacktriangle_B \bigoplus\{\mathtt{Tr!nAv}, \mathtt{Tr!av}\} \prec \mathbf{0} = \mathbb{C}_{12} & [\text{SND}] & (12)
\end{array}$$

Starting from the initial session $\mathbb{M} = \mathtt{Tr} \lhd \varepsilon \prec P_{\mathtt{Tr}} \mid \mathtt{Ht} \lhd \varepsilon \prec P_{\mathtt{Ht}} \mid \mathtt{Al} \lhd \varepsilon \prec P_{\mathtt{Al}}$*, in which all participants have their associated processes as active processes and empty checkpointed sequences, we can have the reductions shown in Figure 4.*

$\mathbb{M}$	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_1 \mid \mathtt{Ht} \lhd \mathbb{C}_6 \mid \mathtt{Al} \lhd \varepsilon \prec P_{\mathtt{Al}}$	using (1) and (6) and rules [PC], [COM], and [PRM]
	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_2 \mid \mathtt{Ht} \lhd \mathbb{C}_6 \mid \mathtt{Al} \lhd \mathbb{C}_9$	using (2) and (9) and rules [PC], [COM], and [PRM]
	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_2 \mid \mathtt{Ht} \lhd \mathbb{C}_7 \mid \mathtt{Al} \lhd \mathbb{C}_9$	using (7) and rules [PC], and [PRM]
	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_3 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_9$	using (3) and (8) and rules [PC], [COM], and [PRM]
	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_4 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_{10}$	using (4) and (10) and rules [PC], [COM], and [PRM]
	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_4 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_{11}$	using (11) and rules [PC], and [PRM]
	$\xrightarrow{\tau} \mathtt{Tr} \lhd \mathbb{C}_5 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_{12}$	using (5) and (12) and rules [PC], [COM], and [PRM]

Figure 4: Example of multiparty session reductions.

From the final session we can do a rollback to B as follows:

$$\dfrac{\dfrac{\mathbb{C}_5 \xrightarrow{B} \mathbb{C}_4 \quad [\text{RBP}]}{\mathtt{Tr} \triangleleft \mathbb{C}_5 \xrightarrow{B} \mathtt{Tr} \triangleleft \mathbb{C}_4}\,[\text{PART}] \qquad \dfrac{\mathbb{C}_{12} \xrightarrow{B} \mathbb{C}_{10} \quad [\text{RBP}]}{\mathtt{Al} \triangleleft \mathbb{C}_{12} \xrightarrow{B} \mathtt{Al} \triangleleft \mathbb{C}_{10}}\,[\text{PART}] \qquad B \notin \mathscr{A}(\mathbb{C}_8)}{\mathtt{Tr} \triangleleft \mathbb{C}_5 \mid \mathtt{Ht} \triangleleft \mathbb{C}_8 \mid \mathtt{Al} \triangleleft \mathbb{C}_{12} \xrightarrow{\tau} \mathtt{Tr} \triangleleft \mathbb{C}_4 \mid \mathtt{Ht} \triangleleft \mathbb{C}_8 \mid \mathtt{Al} \triangleleft \mathbb{C}_{10}}\,[\text{RBM}]$$

In a similar way, we can do a rollback to A from the session $\mathtt{Tr} \triangleleft \mathbb{C}_5 \mid \mathtt{Ht} \triangleleft \mathbb{C}_8 \mid \mathtt{Al} \triangleleft \mathbb{C}_{12}$ *producing* $\mathtt{Tr} \triangleleft \mathbb{C}_2 \mid \mathtt{Ht} \triangleleft \mathbb{C}_6 \mid \mathtt{Al} \triangleleft \mathbb{C}_9$.

3 Type System

Types *Sorts* are ranged over by S and defined by: $S ::= \mathtt{Int} \mid \mathtt{Bool} \mid \ldots$

Single-threaded global types describe the whole conversation scenarios of multiparty sessions, when they reduce forward. The communications can be either without or with named checkpoints.

Global types instead take into account both forward and backward reductions of multiparty sessions. They have therefore a structure which mimics the structure of configurations.

Definition 3.1 *1.* Single-threaded global types *are defined by:*

$$\mathsf{G} ::= \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I} \mid {\blacktriangle_A}\mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J} \mid \mu \mathbf{t}.\mathsf{G} \mid \mathbf{t} \mid \mathsf{end}$$

2. Global types *are pairs* $\Upsilon \prec \mathsf{G}$*, where* Υ *is a (possibly empty) sequence of single-threaded global types with checkpoints having distinct names:*

$$\Upsilon ::= \varepsilon \mid \Upsilon \cdot {\blacktriangle_A}\mathsf{G}$$

We say that G is the *active type* of $\Upsilon \prec \mathsf{G}$. The condition in point (2) of previous definition ensures that, if $\Upsilon \prec \mathsf{G}$ is a global type, and $\Upsilon = \Upsilon' \cdot {\blacktriangle_A}\mathsf{G}' \cdot \Upsilon''$, then no single-threaded global type in Υ', Υ'' can be checkpointed by A.

Session types correspond to projections of single-threaded global types onto the individual participants. Therefore, they can be decorated by named checkpoints. Inspired by [19], we use intersection and union types instead of standard branching and selection, see [13], to take advantage of the subtyping induced by subset inclusion. The grammar of session types, ranged over by T, is then

$$\mathsf{T} ::= \bigwedge_{i\in I} \mathsf{p}?\ell_i(S_i).\mathsf{T}_i \mid \bigvee_{i\in I} \mathsf{p}!\ell_i(S_i).\mathsf{T}_i \mid {\blacktriangle_A}\bigwedge_{j\in J} \mathsf{p}?\ell_j(S_j).\mathsf{T}_j \mid {\blacktriangle_A}\bigvee_{j\in J} \mathsf{p}!\ell_j(S_j).\mathsf{T}_j \mid \mu \mathbf{t}.\mathsf{T} \mid \mathbf{t} \mid \mathsf{end}$$

In both global and session types we require that:

- I, J are not empty sets and J is not a singleton;
- $\ell_h \neq \ell_k$ if $h,k \in I$ or $h,k \in J$;
- the name A does not occur in a type checkpointed by A;
- recursion is guarded.

We constrain J to contain at least two elements since it makes sense to reverse only when an alternative branch could be taken.

Recursive types with the same regular tree are considered equal [21, Chapter 20, Section 2]. In writing types we omit unnecessary brackets, intersections, unions, and end.

We extend the original definition of projection of single-threaded global types onto participants of [13] in the line of [10]. This generalisation allows session participants to behave differently in alternative branches of the same global type, after they have received a message identifying the branch. We define the partial operator $\between$ from sets of session types - all uncheckpointed or checkpointed by the same name - to (possibly checkpointed) intersection types. The operator is defined when the set of session types contains only intersection of types with same sender and different labels. Otherwise, it is undefined. More precisely:

$$\bigwedge\!\!\!\!\bigwedge(\{\mathsf{T}_i\}_{i\in I}) = \begin{cases} \bigwedge_{i\in I}\mathsf{T}_i & \text{if there is } \mathsf{p} \text{ such that } \mathsf{T}_i = \bigwedge_{h\in H_i}\mathsf{p}?\ell_h^{(i)}(S_h^{(i)}).\mathsf{T}_h^{(i)} \text{ for all } i\in I, \\ & \text{and } \ell_h^{(i)} \neq \ell_k^{(j)} \text{for all } h\in H_i, k\in H_j \text{ and } i,j\in I, i\neq j \\ \blacktriangle_A \bigwedge\!\!\!\!\bigwedge(\{\mathsf{T}'_i\}_{i\in I}) & \text{if } \mathsf{T}_i = \blacktriangle_A \mathsf{T}'_i \text{ for all } i\in I \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Notice that in defining $\bigwedge\!\!\!\!\bigwedge$ we could allow identical types in T_i and T_j by the idempotence of intersection. We prefer the current choice for its simplicity.

Figure 5 gives the projection of single-threaded global types onto participants. Notice that, the projection of a checkpointed type onto a participant not involved in the initial communication is defined only when it receives uncheckpointed messages in all the branches. For this reason, and since $\bigwedge\!\!\!\!\bigwedge$ is a partial operator, projections onto some participants may be undefined. A single-threaded global type G is *well formed* if all the projections of G onto all participants are defined. In the following we assume that all single-threaded global types are well formed.

$$(\mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\}_{i\in I})\upharpoonright\mathsf{r} = \begin{cases} \bigvee_{i\in I}\mathsf{q}!\ell_i(S_i).\mathsf{G}_i\upharpoonright\mathsf{r} & \text{if } \mathsf{r}=\mathsf{p} \\ \bigwedge_{i\in I}\mathsf{p}?\ell_i(S_i).\mathsf{G}_i\upharpoonright\mathsf{r} & \text{if } \mathsf{r}=\mathsf{q} \\ \mathsf{end} & \text{if } \mathsf{r}\neq\mathsf{p} \text{ and } \mathsf{r}\neq\mathsf{q} \text{ and } \mathsf{G}_i\upharpoonright\mathsf{r}=\mathsf{end} \text{ for all } i\in I \\ \bigwedge\!\!\!\!\bigwedge(\{\mathsf{G}_i\upharpoonright\mathsf{r}\mid i\in I\}) & \text{if } \mathsf{r}\neq\mathsf{p} \text{ and } \mathsf{r}\neq\mathsf{q} \end{cases}$$

$$(\blacktriangle_A\mathsf{p}\to\mathsf{q}:\{\ell_j(S_j).\mathsf{G}_j\}_{j\in J})\upharpoonright\mathsf{r} = \begin{cases} \blacktriangle_A\bigvee_{j\in J}\mathsf{q}!\ell_j(S_j).\mathsf{G}_j\upharpoonright\mathsf{r} & \text{if } \mathsf{r}=\mathsf{p} \\ \blacktriangle_A\bigwedge_{j\in J}\mathsf{p}?\ell_j(S_j).\mathsf{G}_j\upharpoonright\mathsf{r} & \text{if } \mathsf{r}=\mathsf{q} \\ \mathsf{end} & \text{if } \mathsf{r}\neq\mathsf{p} \text{ and } \mathsf{r}\neq\mathsf{q} \text{ and } \mathsf{G}_j\upharpoonright\mathsf{r}=\mathsf{end} \text{ for all } j\in J \\ \blacktriangle_A\bigwedge\!\!\!\!\bigwedge(\{\mathsf{G}_j\upharpoonright\mathsf{r}\mid j\in J\}) & \text{if } \mathsf{r}\neq\mathsf{p} \text{ and } \mathsf{r}\neq\mathsf{q} \\ & \text{and } \bigwedge\!\!\!\!\bigwedge(\{\mathsf{G}_j\upharpoonright\mathsf{r}\mid j\in J\}) \text{ is not checkpointed} \end{cases}$$

$$(\mu\mathbf{t}.\mathsf{G})\upharpoonright\mathsf{p} = \begin{cases} \mu\mathbf{t}.\mathsf{G}\upharpoonright\mathsf{p} & \text{if } \mathsf{p}\in\mathsf{G}, \\ \mathsf{end} & \text{otherwise.} \end{cases} \qquad \mathbf{t}\upharpoonright\mathsf{p}=\mathbf{t} \qquad \mathsf{end}\upharpoonright\mathsf{p}=\mathsf{end}$$

Figure 5: Projection of single-threaded global types onto participants.

Example 3.2 *Assuming that* Str *is the sort of strings, the global type for the interaction of Figure 1 is* G *defined as follows:*

$$\mathsf{G} = \mathtt{Tr}\to\mathtt{Ht}:\mathtt{qr}(\mathsf{Str}).\mathtt{Tr}\to\mathtt{Al}:\mathtt{qr}(\mathsf{Str}).\mathsf{G}_1 \textit{ where}$$

$$\mathsf{G}_1 = \blacktriangle_A\mathtt{Ht}\to\mathtt{Tr}:\{\mathtt{nAv}.\mathtt{Tr}\to\mathtt{Al}:\mathtt{ds}\ ,\ \mathtt{av}.\mathtt{Tr}\to\mathtt{Al}:\mathtt{rs}.\mathsf{G}_2\} \textit{ and } \mathsf{G}_2 = \blacktriangle_B\mathtt{Al}\to\mathtt{Tr}:\{\mathtt{nAv}\ ,\ \mathtt{av}\}$$

G *is well formed since the projections onto its participants are:*

$$\begin{array}{lcl} \mathsf{G}\upharpoonright\mathtt{Tr} & = & \mathtt{Ht}!\mathtt{qr}(\mathsf{Str}).\mathtt{Al}!\mathtt{qr}(\mathsf{Str}).\blacktriangle_A\bigwedge\{\mathtt{Ht}?\mathtt{nAv}.\mathtt{Al}!\mathtt{ds}\ ,\ \mathtt{Ht}?\mathtt{av}.\mathtt{Al}!\mathtt{rs}.\blacktriangle_B\bigwedge\{\mathtt{Al}?\mathtt{nAv}\ ,\ \mathtt{Al}?\mathtt{av}\}\} \\ \mathsf{G}\upharpoonright\mathtt{Ht} & = & \mathtt{Tr}?\mathtt{qr}(\mathsf{Str}).\blacktriangle_A\bigvee\{\mathtt{Tr}!\mathtt{nAv}\ ,\ \mathtt{Tr}!\mathtt{av}\} \\ \mathsf{G}\upharpoonright\mathtt{Al} & = & \mathtt{Tr}?\mathtt{qr}(\mathsf{Str}).\blacktriangle_A\bigwedge\{\mathtt{Tr}?\mathtt{ds}\ ,\ \mathtt{Tr}?\mathtt{rs}.\blacktriangle_B\bigvee\{\mathtt{Tr}!\mathtt{nAv}\ ,\ \mathtt{Tr}!\mathtt{av}\}\} \end{array}$$

In order to type checkpointed sequences and configurations we need (possibly empty) *sequences of checkpointed session types*, ranged over by ρ:

$$\rho ::= \varepsilon \mid \rho\cdot\blacktriangle_A\mathsf{T}$$

and pairs $\rho\prec\mathsf{T}$, dubbed *configuration types*.

The typing is made more flexible by a subtyping relation on session types exploiting the standard inclusions for intersection and union. A checkpointed type can be a subtype only of a type checkpointed by the same name. Figure 6 gives the subtyping rules: the double line in rules indicates that the rules are interpreted *coinductively* [21, Chapter 21]. Subtyping can be easily decided, see for example [11].

$$[\text{SUB-END}]\quad \mathsf{end} \leqslant \mathsf{end} \qquad [\text{SUB-CK}]\quad \frac{\mathsf{T} \leqslant \mathsf{T}'}{\blacktriangle_A \mathsf{T} \leqslant \blacktriangle_A \mathsf{T}'}$$

$$[\text{SUB-IN}]\quad \frac{\forall i \in I: \quad \mathsf{T}_i \leqslant \mathsf{T}'_i}{\bigwedge_{i\in I\cup J} \mathsf{p}?\ell_i(S_i).\mathsf{T}_i \leqslant \bigwedge_{i\in I} \mathsf{p}?\ell_i(S_i).\mathsf{T}'_i} \qquad [\text{SUB-OUT}]\quad \frac{\forall i \in I: \quad \mathsf{T}_i \leqslant \mathsf{T}'_i}{\bigvee_{i\in I} \mathsf{p}!\ell_i(S_i).\mathsf{T}_i \leqslant \bigvee_{i\in I\cup J} \mathsf{p}!\ell_i(S_i).\mathsf{T}'_i}$$

Figure 6: Subtyping rules.

Typing Rules We distinguish six kinds of typing judgements

$$\Gamma \vdash \mathsf{e} : S \qquad \Gamma \vdash P : \mathsf{T} \qquad \vdash R : \rho \qquad \vdash \mathbb{C} : \rho \prec \mathsf{T} \qquad \vdash \mathbb{M} : \Upsilon \prec \mathsf{G} \qquad \vdash \mathbb{N}\ \checkmark$$

where Γ is the environment $\Gamma ::= \emptyset \mid \Gamma, x : S \mid \Gamma, X : \mathsf{T}$ that associates expression variables with sorts and process variables with session types.

Figure 7 gives the typing rules for processes. Processes typing exploits the correspondence between external choices and intersections, internal choices and unions. A checkpointed process has a type checkpointed by the same name.

$$\frac{\Gamma, x : S \vdash P_i : \mathsf{T}_i}{\Gamma \vdash \sum_{i\in I} \mathsf{p}?\ell_i(\mathsf{e}_i).P_i : \bigwedge_{i\in I} \mathsf{p}?\ell_i(S_i).\mathsf{T}_i}\ [\text{T-IN}] \qquad \frac{\Gamma \vdash \mathsf{e}_i : S_i \quad \Gamma \vdash P_i : \mathsf{T}_i}{\Gamma \vdash \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{e}_i).P_i : \bigvee_{i\in I} \mathsf{p}!\ell_i(S_i).\mathsf{T}_i}\ [\text{T-OUT}]$$

$$\frac{\Gamma \vdash P : \mathsf{T}}{\Gamma \vdash \blacktriangle_A P : \blacktriangle_A \mathsf{T}}\ [\text{T-CK}] \qquad \Gamma \vdash \mathbf{0} : \mathsf{end}\ \ [\text{T-}\mathbf{0}]$$

$$\frac{\Gamma, X : \mathsf{T} \vdash P : \mathsf{T}}{\Gamma \vdash \mu X.P : \mathsf{T}}\ [\text{T-REC}] \qquad \Gamma, X : \mathsf{T} \vdash X : \mathsf{T}\ \ [\text{T-VAR}]$$

Figure 7: Typing rules for processes.

Figure 8 gives the remaining typing rules. Sequences of checkpointed processes are typed by sequences of checkpointed types (rule [T-SP]). Configurations are typed by configuration types (rule [T-C]).

The most interesting rule is rule [T-M] for typing multiparty sessions. The set $\mathsf{pt}(\mathsf{G})$ of participants

$$\frac{\vdash R : \rho \quad \vdash P : \mathsf{T}}{\vdash R \cdot P : \rho \cdot \mathsf{T}}\ [\text{T-SP}] \qquad \frac{\vdash R : \rho \quad \vdash P : \mathsf{T}}{\vdash \mathsf{p} \triangleleft R \prec P : \rho \prec \mathsf{T}}\ [\text{T-C}]$$

$$\frac{\vdash \mathsf{p}_i \triangleleft \mathbb{C}_i : \rho_i \prec \mathsf{T}_i \quad \rho_i \prec \mathsf{T}_i \ltimes_{\mathsf{p}_i} \Upsilon \prec \mathsf{G} \quad i \in I \quad |\Upsilon| = \max\{|\rho_i| \mid i \in I\} \quad \mathsf{pt}(\Upsilon) \cup \mathsf{pt}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \Pi_{i\in I} \mathsf{p}_i \triangleleft \mathbb{C}_i : \Upsilon \prec \mathsf{G}}\ [\text{T-M}]$$

$$\frac{\vdash \mathbb{M} : \Upsilon \prec \mathsf{G}}{\vdash \mathbb{M}\ \checkmark}\ [\text{T-}\checkmark] \qquad \frac{\vdash \mathbb{N}\ \checkmark \quad \vdash \mathbb{N}'\ \checkmark}{\vdash \mathbb{N} \| \mathbb{N}'\ \checkmark}\ [\text{T-N}]$$

Figure 8: Typing rules for checkpointed sequences, multiparty sessions and networks.

of a single-threaded global type is defined by

$$\mathsf{pt}(\mathsf{p}\to\mathsf{q}:\{\ell_i(S_i).\mathsf{G}_i\}_{i\in I})=\{\mathsf{p},\mathsf{q}\}\cup\mathsf{pt}(\mathsf{G}_i)\ (i\in I)^{1}\quad \mathsf{pt}(\blacktriangle_A\mathsf{G})=\mathsf{pt}(\mu\mathbf{t}.\mathsf{G})=\mathsf{pt}(\mathsf{G})\quad \mathsf{pt}(\mathsf{end})=\mathsf{pt}(\mathbf{t})=\emptyset$$

The definition is extended to sequences of checkpointed single-threaded global types by

$$\mathsf{pt}(\varepsilon)=\emptyset\qquad\qquad \mathsf{pt}(\Upsilon\cdot\blacktriangle_A\mathsf{G})=\mathsf{pt}(\Upsilon)\cup\mathsf{pt}(\mathsf{G})$$

The condition $\mathsf{pt}(\Upsilon)\cup\mathsf{pt}(\mathsf{G})\subseteq\{\mathsf{p}_i\mid i\in I\}$ ensures the presence of all session participants and allows the typing of sessions containing $\mathsf{p}\triangleleft\varepsilon\prec\mathbf{0}$ (also if $\mathsf{p}\notin\mathsf{pt}(\Upsilon)\cup\mathsf{pt}(\mathsf{G})$), a property needed to guarantee invariance of types under structural equivalence. Rule [T-M] requires that the types of the active processes are subtypes of the projections of a unique global type. This condition must hold also after a rollback, in which all the processes checkpointed by A, in the respective checkpointed sequences, become the active processes (reduction rule [RBM]). For this reason we type a multiparty session by a global type such that the sequence of single-threaded global types has length equal to the maximum of the lengths of the sequences of session types in the types of configurations. This is expressed by the condition $|\Upsilon|=\max\{|\rho_i|\mid i\in I\}$ in the premise of rule [T-M], where $|\Upsilon|$ is the length of the sequence Υ and $|\rho|$ is the length of the sequence ρ. This requirement is clearly not enough. Further constraints are prescribed by the agreement between global types and configuration types of session participants. This agreement is made more flexible by the use of subtyping.

Definition 3.3 *Let $\rho=\mathsf{T}_1\cdot\ldots\cdot\mathsf{T}_n$ and $\Upsilon=\mathsf{G}_1\cdot\ldots\cdot\mathsf{G}_m$. The configuration type $\rho\prec\mathsf{T}$* p-agrees *with the global type $\Upsilon\prec\mathsf{G}$ (notation $\rho\prec\mathsf{T}\ltimes_{\mathsf{p}}\Upsilon\prec\mathsf{G}$) if all the following conditions hold:*

1. *$\mathsf{T}_i\leq\mathsf{G}_i\upharpoonright\mathsf{p}$ for $1\leq i\leq n$;*
2. *if $\mathsf{T}=\mathsf{end}$, then $n\leq m$ and $\mathsf{G}_i\upharpoonright\mathsf{p}=\mathsf{G}\upharpoonright\mathsf{p}=\mathsf{end}$ for $n+1\leq i\leq m$;*
3. *if T is a union type, then $n=m$ and $\mathsf{T}\leq\mathsf{G}\upharpoonright\mathsf{p}$;*
4. *if T is an intersection type, then either $n=m$ and $\mathsf{T}\leq\mathsf{G}\upharpoonright\mathsf{p}$ or $n=m-1$ and $\mathsf{T}\leq\mathsf{G}_m\upharpoonright\mathsf{p}$ and $\mathsf{T}=\blacktriangle_A\mathsf{T}'$ and $\mathsf{T}'\leq\mathsf{G}\upharpoonright\mathsf{p}$.*

Condition 1, typing rule [T-CK], and the fact that the checkpoints in Υ have distinct names (see Definition 3.1), ensure that, after a rollback, the processes becoming active are in the same position inside the checkpointed sequences. Moreover, these processes have types which are subtypes of the projections of the same single-threaded global type. Condition 2 deals with the case in which the active process of p is $\mathbf{0}$. Once the active process of a participant is $\mathbf{0}$, no more process are added to its checkpointed sequence, so, in case of a rollback to a checkpoint crossed afterwards, by some other participants, its active process would remain $\mathbf{0}$. Therefore the projection of the global type should be end. Conditions 3 and 4 deal with the case in which the type of the active process of p is a union or an intersection. If $n=m$, then these conditions require that its type is subtype of the projection of the global type G, i.e., $\mathsf{T}\leq\mathsf{G}\upharpoonright\mathsf{p}$. If the active process of p is typed by an intersection, then it must be an external choice of input processes (rule [T-IN]). In this case the global type G could capture the situation in which a participant q has internally chosen one branch, memorised in the checkpointed sequence its active process, and the active process of q has became an uncheckpointed output (reduction rule [CKCHC]). This means that, the length of the checkpointed sequence of q must be m, while the length of the checkpointed sequence of p must be $m-1$. To deal with this situation, since an uncheckpointed and a checkpointed type cannot be projections of the same global type, in condition 4, we require that $\mathsf{T}\leq\mathsf{G}_m\upharpoonright\mathsf{p}$ and $\mathsf{T}=\blacktriangle_A\mathsf{T}'$ and $\mathsf{T}'\leq\mathsf{G}\upharpoonright\mathsf{p}$. Notice that, G must be uncheckpointed and G_m must be checkpointed by A. This condition is illustrated in Example 3.4.

Rule [T-M] requires that all types of the configurations which built the multiparty session agree with the global type of the session itself, conditions $\vdash\mathsf{p}_i\triangleleft\mathbb{C}_i:\rho_i\prec\mathsf{T}_i$ and $\rho_i\prec\mathsf{T}_i\ltimes_{\mathsf{p}_i}\Upsilon\prec\mathsf{G}$ for $i\in I$.

A network is well typed if and only if all its multiparty sessions are well typed.

[1] The projectability of G ensures $\mathsf{G}_i=\mathsf{G}_j$ for all $i,j\in I$.

Example 3.4 *To show the typings for the networks produced by the reduction of Example 2.4 consider the global type* G *of Example 3.2. We can derive*

1. $\vdash \mathtt{Tr} \lhd \varepsilon \prec P_{\mathtt{Tr}} \mid \mathtt{Ht} \lhd \varepsilon \prec P_{\mathtt{Ht}} \mid \mathtt{Al} \lhd \varepsilon \prec P_{\mathtt{Al}} : \varepsilon \prec \mathsf{G}$
2. $\vdash \mathtt{Tr} \lhd \mathbb{C}_1 \mid \mathtt{Ht} \lhd \mathbb{C}_6 \mid \mathtt{Al} \lhd \varepsilon \prec P_{\mathtt{Al}} : \varepsilon \prec \mathtt{Tr} \to \mathtt{Al} : \mathtt{qr}(\mathtt{in}).\mathsf{G}_1$
3. $\vdash \mathtt{Tr} \lhd \mathbb{C}_2 \mid \mathtt{Ht} \lhd \mathbb{C}_6 \mid \mathtt{Al} \lhd \mathbb{C}_9 : \varepsilon \prec \mathsf{G}_1$
4. $\vdash \mathtt{Tr} \lhd \mathbb{C}_2 \mid \mathtt{Ht} \lhd \mathbb{C}_7 \mid \mathtt{Al} \lhd \mathbb{C}_9 : \mathsf{G}_1 \prec \mathtt{Ht} \to \mathtt{Tr} : \mathtt{av}.\mathtt{Tr} \to \mathtt{Al} : \mathtt{rs}.\mathsf{G}_2$
5. $\vdash \mathtt{Tr} \lhd \mathbb{C}_3 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_9 : \mathsf{G}_1 \prec \mathtt{Tr} \to \mathtt{Al} : \mathtt{rs}.\mathsf{G}_2$
6. $\vdash \mathtt{Tr} \lhd \mathbb{C}_4 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_{10} : \mathsf{G}_1 \prec \mathsf{G}_2$
7. $\vdash \mathtt{Tr} \lhd \mathbb{C}_4 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_{11} : \mathsf{G}_1 \cdot \mathsf{G}_2 \prec \mathtt{Al} \to \mathtt{Tr} : \mathtt{av}$
8. $\vdash \mathtt{Tr} \lhd \mathbb{C}_5 \mid \mathtt{Ht} \lhd \mathbb{C}_8 \mid \mathtt{Al} \lhd \mathbb{C}_{12} : \mathsf{G}_1 \cdot \mathsf{G}_2 \prec \mathsf{end}$

In typing 6, the active type G_2 *(see Example 3.2) is checkpointed by B and the type of* $\mathtt{Tr}$ *is* $\mathsf{G}_2 \upharpoonright \mathtt{Tr}$*, see the definition of* $\mathbb{C}_4$ *in Example 2.4. In typing 7, the active type is* $\mathtt{Al} \to \mathtt{Tr} : \mathtt{av}$*. The type of* $\mathtt{Tr}$ *remain the same and without the checkpoint B is a subtype of* $(\mathtt{Al} \to \mathtt{Tr} : \mathtt{av}) \upharpoonright \mathtt{Tr}$*. In these two typings the type of* $\mathtt{Tr}$ *agrees with the first and the second case of condition 4 in Definition 3.3, respectively.*

4 Main Properties

In this section we present the technical results of the paper. First we prove subject reduction for multiparty sessions (Theorem 4.3). This implies that well-typed networks respect the choreographies described by global types (Theorem 4.4). This property is usually called session fidelity, see [9]. The progress theorem (Theorem 4.5) establishes reachability of all communications and backward reductions.

As standard we start with an inversion lemma for processes, checkpointed sequences, configurations, multiparty sessions and networks.

Lemma 4.1 *[Inversion]*

1. *Let* $\Gamma \vdash P : \mathsf{T}$.
   - *(a) If* $P = \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i$*, then* $\mathsf{T} = \bigwedge_{i\in I} \mathsf{p}?\ell_i(S_i).\mathsf{T}_i$*, and* $\Gamma, x{:}S_i \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
   - *(b) If* $P = \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{e}_i).P_i$*, then* $\mathsf{T} = \bigvee_{i\in I} \mathsf{p}!\ell_i(S_i).\mathsf{T}_i$*,* $\Gamma \vdash \mathsf{e}_i : S_i$*, and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
   - *(c) If* $P = \blacktriangle_A \sum_{j\in J} \mathsf{p}?\ell_j(x_j).P_j$*, then* $\mathsf{T} = \blacktriangle_A \bigwedge_{j\in J} \mathsf{p}?\ell_j(S_j).\mathsf{T}_j$*, and* $\Gamma, x{:}S_j \vdash P_j : \mathsf{T}_j$ *for* $j \in J$.
   - *(d) If* $P = \blacktriangle_A \bigoplus_{j\in J} \mathsf{p}!\ell_j(\mathsf{e}_j).P_j$*, then* $\mathsf{T} = \blacktriangle_A \bigvee_{j\in J} \mathsf{p}!\ell_j(S_j).\mathsf{T}_j$*,* $\Gamma \vdash \mathsf{e}_j : S_j$*, and* $\Gamma \vdash P_j : \mathsf{T}_j$ *for* $j \in J$.
   - *(e) If* $P = \mu X.Q$*, then* $\Gamma, X{:}\mathsf{T} \vdash Q : \mathsf{T}$.
   - *(f) If* $P = X$*, then* $\Gamma = \Gamma', X{:}\mathsf{T}$.
   - *(g) If* $P = \mathbf{0}$*, then* $\mathsf{T} = \mathsf{end}$.
2. *If* $\vdash R \cdot P : \rho$*, then* $\rho = \rho' \cdot \mathsf{T}$ *and* $\vdash R : \rho'$ *and* $\vdash P : \mathsf{T}$.
3. *If* $\vdash \mathsf{p} \lhd R \prec P : \rho \prec \mathsf{T}$*, then* $\vdash R : \rho$ *and* $\vdash P : \mathsf{T}$.
4. *If* $\vdash \Pi_{i\in I} \mathsf{p}_i \lhd \mathbb{C}_i : \Upsilon \prec \mathsf{G}$*, then* $\vdash \mathsf{p}_i \lhd \mathbb{C}_i : \rho_i \prec \mathsf{T}_i$ *and* $\rho_i \prec \mathsf{T}_i \ltimes_{\mathsf{p}_i} \Upsilon \prec \mathsf{G}$ *for* $i \in I$ *and* $|\Upsilon| = max\{|\rho_i| \mid i \in I\}$ *and* $\mathsf{pt}(\Upsilon) \cup \mathsf{pt}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}$.
5. *If* $\vdash \mathbb{N}\ \checkmark$*, then either* $\vdash \mathbb{N} : \Upsilon \prec \mathsf{G}$ *or* $\mathbb{N} = \mathbb{N}' \mid \mathbb{N}''$ *and* $\vdash \mathbb{N}'\ \checkmark$ *and* $\vdash \mathbb{N}''\ \checkmark$.

Proof Easy from the definition of the typing relation.

The inversion lemma gives some important properties.

Points (1a), (1b), (1c),(1d) and (1g) ensure that the processes are checkpointed iff their types are checkpointed with the same name. Moreover, input processes have intersection types, output processes have union types and the process $\mathbf{0}$ has type end.

Point (2) says that the length of checkpointed sequences is equal to the length of the sequences of their checkpointed session types.

Point (4) and Definition 3.3 imply that in a well-typed multiparty session:

- exactly one of the active processes is an output process;

- at least one of the active processes is an input process.

More precisely, if $\mathsf{G} = {}_{\blacktriangle_A}\mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J}$, then the active processes of participants p and q must have types which are subtypes of $\mathsf{G}\upharpoonright\mathsf{p}$ and $\mathsf{G}\upharpoonright\mathsf{q}$, respectively. E.g., this is the case of typing 3 of Example 3.4, where the active processes of `Ht` and `Tr` have types equal to the projections the active global type onto the respective participant. (Similarly for typing 6.) If $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I}$, then the active process of participant p must have a type which is a subtype of $\mathsf{G}\upharpoonright\mathsf{p}$, as before. Instead, the active process of participant q can have either a type which is a subtype of $\mathsf{G}\upharpoonright\mathsf{q}$, or a checkpointed type ${}_{\blacktriangle_A}\mathsf{T}$ such that $\mathsf{T} \leqslant \mathsf{G}\upharpoonright\mathsf{q}$. E.g., this is the case of typing 7 in Example 3.4, where the active processes of `Al` has an uncheckpointed output type (a union of a single type), whereas the active process of `Tr` has an intersection type checkpointed by A. If G is checkpointed by A, then an active process different from $\mathbf{0}$ is checkpointed by A and the checkpointed sequence has length $|\Upsilon|$. Instead, if G is uncheckpointed, then the output process is uncheckpointed and its checkpointed sequence has length $|\Upsilon|$, while an input process can be:

- either uncheckpointed: in this case its checkpointed sequence has length $|\Upsilon|$,
- or checkpointed: in this case its checkpointed sequence has length $|\Upsilon| - 1$.

E.g., for the case of typing 3 of Example 3.4, the length of the checkpointed sequences of all participants is 0, and all participants have checkpointed active processes, whereas for typing 7, the length of the checkpointed sequence of `Al` is 2, whereas the ones of the other participants is 1. Moreover, the active process of `Tr` is checkpointed and the one of `Ht` is $\mathbf{0}$. These properties follow from conditions 3 and 4 of Definition 3.3. Condition 4 of Definition 3.3 implies also that, if G is uncheckpointed and the type of an active input process is checkpointed, then the name of its checkpoint is the name of the checkpoint of the global type on the top of Υ. Lastly, condition 1 of Definition 3.3 ensures that all the processes in the checkpointed sequences are checkpointed by the same names as the global types in Υ, in the exact order. The only difference can be the length of the sequences, which must satisfy the other conditions of Definition 3.3.

Global types are not preserved under multiparty session reductions: this is expected, as they evolve according to the silent actions, the communications and the rollbacks performed by the session participants. This evolution is formalised by the *reduction of global types*, which is the smallest pre-order relation closed under the rules of Figure 9. Rule [G-CkChc] corresponds to the reduction of the output process of participant p by rule [CkChc] of Figure 2. Notably, the process is checkpointed with name A. Rule [G-Com] corresponds to the communication performed by rule [Com] of Figure 3. I.e., the output process of participant p sends a message labelled ℓ_k (rule [Snd] of Figure 2) and the input process of participant q receives this message (rule [Rcv] or [CkRcv] of Figure 2). Notice that, when rule [CkRcv] is used, the checkpointed input process is added to the checkpointed sequence of participant q. Lastly, rule [G-Rb] is used when the multiparty session rolls back by means of rule [RbM] of Figure 3. Let A be the name of the checkpoint of G, the participants must either reduce by rule [RbP] of Figure 2 with a transition labelled A or remain unchanged. In the unchanged configurations the processes belonging to the checkpointed sequences are not checkpointed by A and the active processes are $\mathbf{0}$. We use $\Longrightarrow^*$ to denote the transitive and reflexive closure of the $\Longrightarrow$ relation.

A standard substitution lemma is handy.

Lemma 4.2 *If* $\Gamma, x : S \vdash P : \mathsf{T}$ *and* $\Gamma \vdash \mathsf{e} : S$ *and* $\mathsf{e} \downarrow v$, *then* $\Gamma \vdash P\{v/x\} : \mathsf{T}$.

We can show subject reduction for well-typed multiparty sessions, which implies subject reduction for well-typed networks.

Theorem 4.3 (SR) *If* $\vdash \mathbb{M} : \Upsilon \prec \mathsf{G}$ *and* $\mathbb{M} \xrightarrow{\tau}{}^* \mathbb{M}'$, *then* $\vdash \mathbb{M}' : \Upsilon' \prec \mathsf{G}'$ *and* $\Upsilon \prec \mathsf{G} \Longrightarrow^* \Upsilon' \prec \mathsf{G}'$.

$$\Upsilon \prec {}_{\blacktriangle_A}\mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J} \Longrightarrow \Upsilon \cdot ({}_{\blacktriangle_A}\mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J}) \prec \mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J} \quad \text{[G-CkChc]}$$

$$\Upsilon \prec \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I} \Longrightarrow \Upsilon \prec \mathsf{G}_k \quad k \in I \quad \text{[G-Com]}$$

$$\Upsilon \cdot \mathsf{G} \cdot \Upsilon' \prec \mathsf{G}' \Longrightarrow \Upsilon \prec \mathsf{G} \quad \text{[G-Rb]}$$

Figure 9: Reduction rules of global types.

Proof By induction on multiparty session reductions. It is easy to verify that typing is invariant under structural equivalence of multiparty sessions, so we will omit the application of rule [EqM].
If $\mathbb{M} \xrightarrow{\tau} \mathbb{M}'$, then there are three cases:

1. $\mathbb{M} = \mathsf{p} \triangleleft \mathbb{C} \mid \mathbb{M}''$ and $\mathbb{M}' = \mathsf{p} \triangleleft \mathbb{C}' \mid \mathbb{M}''$ and $\mathsf{p} \triangleleft \mathbb{C} \xrightarrow{\tau} \mathsf{p} \triangleleft \mathbb{C}'$, i.e., rule [PrM] has been applied,
2. $\mathbb{M} = \mathsf{p} \triangleleft \mathbb{C}_\mathsf{p} \mid \mathsf{q} \triangleleft \mathbb{C}_\mathsf{q} \mid \mathbb{M}''$ and $\mathbb{M}' = \mathsf{p} \triangleleft \mathbb{C}'_\mathsf{p} \mid \mathsf{q} \triangleleft \mathbb{C}'_\mathsf{q} \mid \mathbb{M}''$ and $\mathsf{p} \triangleleft \mathbb{C}_\mathsf{p} \xrightarrow{\mathsf{q}!\ell(v)} \mathsf{p} \triangleleft \mathbb{C}'_\mathsf{p}$ and $\mathsf{q} \triangleleft \mathbb{C}_\mathsf{q} \xrightarrow{\mathsf{p}?\ell(v)} \mathsf{q} \triangleleft \mathbb{C}'_\mathsf{q}$, i.e., rule [PrM], with rule [Com] on the premise, has been applied,
3. $\mathbb{M} = \Pi_{i\in I}\mathsf{p}_i \triangleleft \mathbb{C}_{\mathsf{p}_i} \mid \Pi_{j\in J}\mathsf{p}_j \triangleleft \mathbb{C}_{\mathsf{p}_j}$ and $\mathbb{M}' = \Pi_{i\in I}\mathsf{p}_i \triangleleft \mathbb{C}'_{\mathsf{p}_i} \mid \Pi_{j\in J}\mathsf{p}_j \triangleleft \mathbb{C}_{\mathsf{p}_j}$ and $\mathsf{p}_i \triangleleft \mathbb{C}_{\mathsf{p}_i} \xrightarrow{A} \mathsf{p}_i \triangleleft \mathbb{C}'_{\mathsf{p}_i}$ for all $i \in I$ and $A \notin \mathscr{A}(\mathbb{C}_{\mathsf{p}_j})$ for all $j \in J$, i.e., rule [RbM] has been applied.

Case (1). From $\mathsf{p} \triangleleft \mathbb{C} \xrightarrow{\tau} \mathsf{p} \triangleleft \mathbb{C}'$ we get $\mathbb{C} \xrightarrow{\tau} \mathbb{C}'$. Let $\mathbb{C} = R \prec P$ and $\mathbb{C}' = R' \prec P'$. Therefore

(a) either $\mathbb{C} \xrightarrow{\tau} \mathbb{C}'$ with rule [Chc], which implies $P = \bigoplus_{i\in I} \mathsf{q}!\ell_i(\mathsf{e}_i).P_i$ and $R' = R$ and $P' = \mathsf{q}!\ell_k(\mathsf{e}_k).P_k$ for $k \in I \neq \{k\}$,

(b) or $\mathbb{C} \xrightarrow{\tau} \mathbb{C}'$ with rule [CkChc], which implies $P = {}_{\blacktriangle_A}\bigoplus_{j\in J} \mathsf{q}!\ell_j(\mathsf{e}_j).P_j$ and $R' = R \cdot P$ and $P' = \mathsf{q}!\ell_k(\mathsf{e}_k).P_k$ for $k \in J \neq \{k\}$.

By Lemma 4.1(4) $\vdash \mathsf{p} \triangleleft \mathbb{C} : \rho \prec \mathsf{T}$ and $\rho \prec \mathsf{T} \ltimes_\mathsf{p} \Upsilon \prec \mathsf{G}$. Then $\vdash R : \rho$ and $\vdash P : \mathsf{T}$ by Lemma 4.1(3). Lemma 4.1(1b) and (1d) imply that T is a union type, then we get $|\rho| = |\Upsilon|$ and $\mathsf{T} \leqslant \mathsf{G} \upharpoonright \mathsf{p}$ by condition 3 of Definition 3.3.
Case (1a). Lemma 4.1(1b) applied to $\vdash P : \mathsf{T}$ gives $\mathsf{T} = \bigvee_{i\in I} \mathsf{q}!\ell_i(S_i).\mathsf{T}_i$ and $\vdash \mathsf{e}_i : S_i$ and $\vdash P_i : \mathsf{T}_i$ for $i \in I$. Then $\mathsf{T} \leqslant \mathsf{G} \upharpoonright \mathsf{p}$ implies $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_i(S_i).\mathsf{G}_i\}_{i\in I\cup L}$. We choose $\Upsilon' = \Upsilon$ and $\mathsf{G}' = \mathsf{G}$. In fact, we can derive $\vdash P' : \mathsf{q}!\ell_k(S_k).\mathsf{T}_k$ and $\mathsf{q}!\ell_k(S_k).\mathsf{T}_k \leqslant \mathsf{G} \upharpoonright \mathsf{p}$. Therefore, we derive $\vdash \mathbb{M}' : \Upsilon' \prec \mathsf{G}'$ by rule [T-M].
Case (1b). Lemma 4.1(1d) applied to $\vdash P : \mathsf{T}$ gives $\mathsf{T} = {}_{\blacktriangle_A}\bigvee_{j\in J} \mathsf{q}!\ell_j(S_j).\mathsf{T}_j$ and $\vdash \mathsf{e}_j : S_j$ and $\vdash P_j : \mathsf{T}_j$ for $j \in J$. Then $\mathsf{T} \leqslant \mathsf{G} \upharpoonright \mathsf{p}$ implies $\mathsf{G} = {}_{\blacktriangle_A}\mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J\cup L}$. We can choose $\Upsilon' = \Upsilon \cdot \mathsf{G}$ and $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \{\ell_j(S_j).\mathsf{G}_j\}_{j\in J\cup L}$. In fact $\Upsilon \prec \mathsf{G} \Longrightarrow \Upsilon \cdot \mathsf{G} \prec \mathsf{G}'$ by rule [G-CkChc] and we can derive $\vdash P' : \mathsf{q}!\ell_k(S_k).\mathsf{T}_k$ and $\mathsf{q}!\ell_k(S_k).\mathsf{T}_k \leqslant \mathsf{G}' \upharpoonright \mathsf{p}$.
If $\mathsf{r} \neq \mathsf{p}$ and $\mathsf{r} \triangleleft R_\mathsf{r} \prec P_\mathsf{r}$ occurs in $\mathbb{M}$, then by Lemma 4.1(4) $\vdash \mathsf{r} \triangleleft R_\mathsf{r} \prec P_\mathsf{r} : \rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r}$ and $\rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r} \ltimes_\mathsf{r} \Upsilon \prec \mathsf{G}$. Lemma 4.1(3) gives $\vdash R_\mathsf{r} : \rho_\mathsf{r}$ and $\vdash P_\mathsf{r} : \mathsf{T}_\mathsf{r}$. If $P_\mathsf{r} = \mathbf{0}$, then $\mathsf{T}_\mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r} = \mathsf{G}' \upharpoonright \mathsf{r} = \mathsf{end}$ and $\rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r} \ltimes_\mathsf{r} \Upsilon' \prec \mathsf{G}'$, since condition 2 of Definition 3.3 is satisfied. If P_r is an input process, then T_r is an intersection type and $|\rho_\mathsf{r}| = |\Upsilon|$ and $\mathsf{T}_\mathsf{r} \leqslant \mathsf{G} \upharpoonright \mathsf{r}$ by the first alternative in condition 4 of Definition 3.3. We have $|\rho_\mathsf{r}| = |\Upsilon'| - 1$ since $|\rho_\mathsf{r}| = |\Upsilon|$. From $\mathsf{T}_\mathsf{r} \leqslant \mathsf{G} \upharpoonright \mathsf{r}$ we get $\mathsf{T}_\mathsf{r} = {}_{\blacktriangle_A}\mathsf{T}'_\mathsf{r}$ and $\mathsf{T}'_\mathsf{r} \leqslant \mathsf{G}' \upharpoonright \mathsf{r}$. Then $\rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r} \ltimes_\mathsf{r} \Upsilon' \prec \mathsf{G}'$ since the second alternative of condition 4 in Definition 3.3 is satisfied. We can then derive $\vdash \mathbb{M}' : \Upsilon' \prec \mathsf{G}'$ by rule [T-M].
Case (2). From $\mathsf{p} \triangleleft \mathbb{C}_\mathsf{p} \xrightarrow{\mathsf{q}!\ell(v)} \mathsf{p} \triangleleft \mathbb{C}'_\mathsf{p}$ and $\mathsf{q} \triangleleft \mathbb{C}_\mathsf{q} \xrightarrow{\mathsf{p}?\ell(v)} \mathsf{q} \triangleleft \mathbb{C}'_\mathsf{q}$ we get that $\mathbb{C}_\mathsf{p} \xrightarrow{\mathsf{q}!\ell(v)} \mathbb{C}'_\mathsf{p}$ and $\mathbb{C}_\mathsf{q} \xrightarrow{\mathsf{p}?\ell(v)} \mathbb{C}'_\mathsf{q}$. Let $\mathbb{C}_\mathsf{p} = R_\mathsf{p} \prec P_\mathsf{p}$ and $\mathbb{C}'_\mathsf{p} = R'_\mathsf{p} \prec P'_\mathsf{p}$ and $\mathbb{C}_\mathsf{q} = R_\mathsf{q} \prec P_\mathsf{q}$ and $\mathbb{C}'_\mathsf{q} = R'_\mathsf{q} \prec P'_\mathsf{q}$. Then $\mathbb{C}_\mathsf{p}$ reduces with rule [Snd], which implies $P_\mathsf{p} = \mathsf{q}!\ell(\mathsf{e}).P$ and $\mathsf{e} \downarrow v$ and $R'_\mathsf{p} = R_\mathsf{p}$ and $P'_\mathsf{p} = P$. The reduction of $\mathbb{C}_\mathsf{q}$ can be done

(a) either with rule [Rcv], which implies $P_\mathsf{q} = \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i$ with $\ell_k = \ell$ and $R'_\mathsf{q} = R_\mathsf{q}$ and $P'_\mathsf{q} = P_k\{v/x\}$,

(b) or with rule [CKRCV], which implies $P_\mathsf{q} = \blacktriangle_A \sum_{j\in J} \mathsf{p}?\ell_j(x_j).P_j$ with $\ell_k = \ell$ and $R'_\mathsf{q} = R_\mathsf{q} \cdot P_\mathsf{q}$ and $P'_\mathsf{q} = P_k\{v/x\}$.

By Lemma 4.1(4) $\vdash \mathsf{p} \triangleleft \mathbb{C}_\mathsf{p} : \rho_\mathsf{p} \prec \mathsf{T}_\mathsf{p}$ and $\rho_\mathsf{p} \prec \mathsf{T}_\mathsf{p} \ltimes_\mathsf{p} \Upsilon \prec \mathsf{G}$ and $\vdash \mathsf{q} \triangleleft \mathbb{C}_\mathsf{q} : \rho_\mathsf{q} \prec \mathsf{T}_\mathsf{p}$ and $\rho_\mathsf{q} \prec \mathsf{T}_\mathsf{q} \ltimes_\mathsf{q} \Upsilon \prec \mathsf{G}$. Then $\vdash R_\mathsf{p} : \rho_\mathsf{p}$ and $\vdash P_\mathsf{p} : \mathsf{T}_\mathsf{p}$ and $\vdash R_\mathsf{q} : \rho_\mathsf{q}$ and $\vdash P_\mathsf{q} : \mathsf{T}_\mathsf{q}$ by Lemma 4.1(3). Lemma 4.1(1b) applied to $\vdash P_\mathsf{p} : \mathsf{T}_\mathsf{p}$ gives $\mathsf{T}_\mathsf{p} = \mathsf{q}!\ell(S).\mathsf{T}$ and $\vdash \mathsf{e} : S$ and $\vdash P : \mathsf{T}$. We have $|\rho_\mathsf{p}| = |\Upsilon|$ and $\mathsf{T}_\mathsf{p} \leqslant \mathsf{G} \upharpoonright \mathsf{p}$ by condition 3 of Definition 3.3. This implies $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\ell_h(S_h).\mathsf{G}_h\}_{h\in H}$ with $\ell_k = \ell$, $S = S_k$ and $\mathsf{T} \leqslant \mathsf{G}_k \upharpoonright \mathsf{p}$. We derive $\vdash \mathsf{p} \triangleleft \mathbb{C}'_\mathsf{p} : \rho_\mathsf{p} \prec \mathsf{T}$.

We can choose $\Upsilon' = \Upsilon$ and $\mathsf{G}' = \mathsf{G}_k$ since $\Upsilon \prec \mathsf{G} \Longrightarrow \Upsilon \prec \mathsf{G}_k$ by rule [G-COM] and we will show that $\vdash \mathbb{M}' : \Upsilon' \prec \mathsf{G}'$ is derivable by checking the agreement conditions of Definition 3.3 for all pairs participant/configuration of $\mathbb{M}'$. From $\rho_\mathsf{p} \prec \mathsf{T}_\mathsf{p} \ltimes_\mathsf{p} \Upsilon \prec \mathsf{G}$ and $\mathsf{T} \leqslant \mathsf{G}_k \upharpoonright \mathsf{p}$ we get $\rho_\mathsf{p} \prec \mathsf{T} \ltimes_\mathsf{p} \Upsilon' \prec \mathsf{G}'$.

Case (2a). Lemma 4.1(1a) applied to $\vdash P_\mathsf{q} : \mathsf{T}_\mathsf{q}$ gives $\mathsf{T}_\mathsf{q} = \bigwedge_{i\in I} \mathsf{q}?\ell_i(S_i).\mathsf{T}_i$ and $x_i : S_i \vdash P_i : \mathsf{T}_i$ for $i \in I$. We get $|\rho_\mathsf{p}| = |\Upsilon|$ and $\mathsf{T}_\mathsf{q} \leqslant \mathsf{G} \upharpoonright \mathsf{q}$ by condition 4 of Definition 3.3. This implies $H \subseteq I$ and in particular $k \in I$. The Substitution Lemma implies $\vdash P'_\mathsf{q} : \mathsf{T}_k$. We derive $\vdash \mathsf{q} \triangleleft \mathbb{C}'_\mathsf{q} : \rho_\mathsf{q} \prec \mathsf{T}_k$. From $\rho_\mathsf{q} \prec \mathsf{T}_\mathsf{q} \ltimes_\mathsf{q} \Upsilon \prec \mathsf{G}$ and $\mathsf{T}_k \leqslant \mathsf{G}_k \upharpoonright \mathsf{q}$ we get $\rho_\mathsf{q} \prec \mathsf{T}_k \ltimes_\mathsf{q} \Upsilon' \prec \mathsf{G}'$.

Case (2b). Lemma 4.1(1c) applied to $\vdash P_\mathsf{q} : \mathsf{T}_\mathsf{q}$ gives $\mathsf{T}_\mathsf{q} = \blacktriangle_A \bigwedge_{j\in J} \mathsf{q}?\ell_j(S_j).\mathsf{T}_j$ and $x_j : S_j \vdash P_j : \mathsf{T}_j$ for $j \in J$. Let $\Upsilon = \Upsilon'' \cdot \mathsf{G}''$. We get $|\rho_\mathsf{p}| = |\Upsilon| - 1$ and $\mathsf{T}_\mathsf{q} \leqslant \mathsf{G}'' \upharpoonright \mathsf{q}$ and $\bigwedge_{j\in J} \mathsf{q}?\ell_j(S_j).\mathsf{T}_j \leqslant \mathsf{G} \upharpoonright \mathsf{q}$ by condition 4 of Definition 3.3. This implies $H \subseteq J$ and in particular $k \in J$. As in previous case the Substitution Lemma implies $\vdash P'_\mathsf{q} : \mathsf{T}_k$. We derive $\vdash \mathsf{q} \triangleleft \mathbb{C}'_\mathsf{q} : \rho_\mathsf{q} \cdot \mathsf{T}_\mathsf{q} \prec \mathsf{T}_k$. From $\rho_\mathsf{q} \prec \mathsf{T}_\mathsf{q} \ltimes_\mathsf{q} \Upsilon'' \cdot \mathsf{G}'' \prec \mathsf{G}$ and $\mathsf{T}_\mathsf{q} \leqslant \mathsf{G}'' \upharpoonright \mathsf{q}$ and $\mathsf{T}_k \leqslant \mathsf{G}_k \upharpoonright \mathsf{q}$ we get $\rho_\mathsf{q} \cdot \mathsf{T}_\mathsf{q} \prec \mathsf{T}_k \ltimes_\mathsf{q} \Upsilon' \prec \mathsf{G}'$.

Consider a participant $\mathsf{r} \neq \mathsf{p}, \mathsf{q}$. If $\mathsf{r} \triangleleft R_\mathsf{r} \prec P_\mathsf{r}$ occurs in $\mathbb{M}$, then by Lemma 4.1(4) $\vdash \mathsf{r} \triangleleft R_\mathsf{r} \prec P_\mathsf{r} : \rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r}$ and $\rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r} \ltimes_\mathsf{r} \Upsilon \prec \mathsf{G}$. Lemma 4.1(3) gives $\vdash R_\mathsf{r} : \rho_\mathsf{r}$ and $\vdash P_\mathsf{r} : \mathsf{T}_\mathsf{r}$. If $P_\mathsf{r} = \mathbf{0}$, then $\mathsf{T}_\mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r} = \mathsf{G}' \upharpoonright \mathsf{r} = \mathsf{end}$ and $\rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r} \ltimes_\mathsf{r} \Upsilon' \prec \mathsf{G}'$, since condition 2 of Definition 3.3 is satisfied. If P_r is an input process, then T_r is an intersection type and either $|\rho_\mathsf{r}| = |\Upsilon|$ and $\mathsf{T}_\mathsf{r} \leqslant \mathsf{G} \upharpoonright \mathsf{r}$ or $|\rho_\mathsf{r}| = |\Upsilon| - 1$ and $\mathsf{T}_\mathsf{r} \leqslant \mathsf{G}'' \upharpoonright \mathsf{r}$ and $\mathsf{T}_\mathsf{r} = \blacktriangle_A \mathsf{T}'$ and $\mathsf{T}' \leqslant \mathsf{G} \upharpoonright \mathsf{r}$ by condition 4 of Definition 3.3. In both cases $\mathsf{G} \upharpoonright \mathsf{r} \leqslant \mathsf{G}_k \upharpoonright \mathsf{r}$. If G_k is uncheckpointed we conclude $\rho_\mathsf{r} \prec \mathsf{T}_\mathsf{r} \ltimes_\mathsf{r} \Upsilon' \prec \mathsf{G}'$. If G_k is checkpointed we must have $|\rho_\mathsf{r}| = |\Upsilon|$ and $\mathsf{T}_\mathsf{r} \leqslant \mathsf{G} \upharpoonright \mathsf{r}$. In fact otherwise $\mathsf{T}' \leqslant \mathsf{G}_k \upharpoonright \mathsf{r}$ would imply $\mathsf{T}' = \blacktriangle_B \mathsf{T}''$, where B is the name of the checkpoint of G_k. We would get $\mathsf{T}_\mathsf{r} = \blacktriangle_A \blacktriangle_B \mathsf{T}''$ and this is not a session type according to our syntax.

Case (3). Let $\Upsilon = \Upsilon' \cdot \mathsf{G}' \cdot \Upsilon''$ and A be the name of the checkpoint of G'. Lemma 4.1(4) implies that $\vdash \mathsf{p}_l \triangleleft \mathbb{C}_{\mathsf{p}_l} : \rho_{\mathsf{p}_l} \prec \mathsf{T}_{\mathsf{p}_l}$ and $\rho_{\mathsf{p}_l} \prec \mathsf{T}_{\mathsf{p}_l} \ltimes_{\mathsf{p}_l} \Upsilon \prec \mathsf{G}$ for all $l \in I \cup J$. If $|\rho_{\mathsf{p}_l}| \leq |\Upsilon'|$, then $A \notin (\mathbb{C}_{\mathsf{p}_l})$ by Definition 3.1(2) and Definition 3.3. This implies $l \in J$ and $\rho_{\mathsf{p}_l} \prec \mathsf{T}_{\mathsf{p}_l} \ltimes_{\mathsf{p}_l} \Upsilon' \prec \mathsf{G}'$. Otherwise $l \in I$ and $\rho_{\mathsf{p}_l} = \rho'_{\mathsf{p}_l} \cdot \mathsf{T}_l \cdot \rho''_{\mathsf{p}_l}$ and $\mathsf{T}_l \leqslant \mathsf{G}' \upharpoonright \mathsf{p}_l$ by condition 1 of Definition 3.3. Let $\mathbb{C}_{\mathsf{p}_i} = R_{\mathsf{p}_i} \prec P_{\mathsf{p}_i}$. From $\vdash \mathsf{p}_i \triangleleft \mathbb{C}_{\mathsf{p}_i} : \rho_{\mathsf{p}_i} \prec \mathsf{T}_{\mathsf{p}_i}$ we get $\vdash R_{\mathsf{p}_i} : \rho'_{\mathsf{p}_i} \cdot \mathsf{T}_i \cdot \rho''_{\mathsf{p}_i}$ by Lemma 4.1(3). Then $R_{\mathsf{p}_i} = R'_{\mathsf{p}_i} \cdot P_i \cdot R''_{\mathsf{p}_i}$ and $\vdash P_i : \mathsf{T}_i$ by Lemma 4.1(2). The name of the checkpoint of P_i is A since $\mathsf{T}_i \leqslant \mathsf{G}' \upharpoonright \mathsf{p}_i$ and then $\mathbb{C}'_{\mathsf{p}_l} = R'_{\mathsf{p}_i} \prec P_i$. This implies $\rho_{\mathsf{p}_i} \prec \mathsf{T}_{\mathsf{p}_i} \ltimes_{\mathsf{p}_i} \Upsilon' \prec \mathsf{G}'$. We can then derive $\vdash \mathbb{M}' : \Upsilon' \prec \mathsf{G}'$ by rule [T-M].

From the proof of the previous theorem we get the following properties of well-typed networks, which are usual for session calculi [13]. We say that an application of the reduction rule [COM] has a *type mismatch*, if there is no sort that can be derived both for the communicated value and for variable associated to the communicated label in the input process.

Global types describe interaction protocols. The communications in well-typed networks evolve following the exact order of the associated global types.

Theorem 4.4 (Session Fidelity) *If* $\vdash \mathbb{N}$ ✓, *then reducing* $\mathbb{N}$

1. there is never a type mismatch;
2. the communications occur in the order prescribed by global types.

Notably property 1 holds in spite of the fact that session participants may exchange messages of different types. Property 2 says that session participants behave according to established communication protocols.

The standard definition of progress only ensures absence of deadlocks [21, Section 8.3]. Progress for session calculi means that all the requested interactions may happen [5]. In reversible sessions it is also natural to guarantee that all possible rollbacks may take place. This leads us to the following formulation of the progress theorem.

Theorem 4.5 (Progress) *If* $\vdash \mathbb{N} \checkmark$*, then:*

1. *if* $\mathbb{N}$ *contains an input or output process, then* $\mathbb{N}$ *forward reduces to* $\mathbb{N}'$ *and that input or output prefix does not occur in* $\mathbb{N}'$*;*
2. *if* $\mathbb{N}$ *contains a checkpoint named A, then there is a reduction of* $\mathbb{N}$ *in which the last step is a rollback making the processes checkpointed by A active processes.*

Proof As proved in [5], a single multiparty session in a standard calculus with global and session types, like the calculus in [13], always enjoys progress whenever it is well typed. In fact, by the Subject Reduction Theorem (Theorem 4.3), reduction preserves well-typedness of sessions. Moreover, all required session participants are present, as ensured by the condition $\mathsf{pt}(\Upsilon) \cup \mathsf{pt}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}$ in the premise of rule [T-M]. Thus, all communications among participants in a unique session will take place, in the order prescribed by the single-threaded active global type. This ensures that property 1 holds. For property 2 observe that, if $\mathbb{N}$ contains checkpoints named A, then there is at least one multiparty session $\mathbb{M}$ in $\mathbb{N}$ which is typed by a global type $\Upsilon \prec \mathsf{G}$ which contains A. If A occurs in G, then we can reduce forward $\mathbb{M}$ until the processes checkpointed by A will be all in the checkpointed sequences, and then apply the desired rollback. If A does not occur in G let $\Upsilon = \Upsilon' \cdot {}_{\blacktriangle_B}\mathsf{G}' \cdot \Upsilon''$ and A occurs in ${}_{\blacktriangle_B}\mathsf{G}'$. Then the checkpointed sequences of $\mathbb{M}$ have processes checkpointed by B. We can then apply the rollback which makes the processes checkpointed by B to become active processes. If $A = B$ we are done. Otherwise the active global type of the obtained session contains A and we can conclude as in previous case.

5 Related Work and Conclusions

Since the pioneering work by Danos and Krivine [6], reversible computations in process algebras have been widely studied. The calculus of [6] adds a distributed monitoring system to CCS [18] allowing computations to be rewound. Phillips and Ulidowski [20] propose a method for reversing process operators that are definable by SOS rules in a general format, using keys to bind synchronised actions together. A reversible variant of the higher-order π-calculus is defined in [16], using name tags for identifying threads and explicit memory processes. In [15], Lanese et al. enrich the calculus of [16] with a fine-grained rollback primitive. To the best of our knowledge the earliest work dealing with rollback of communicating systems are [7, 8, 14]. In these papers an extension of CCS models the combination of rollback recovery and coordinated checkpoints.

As pointed out in [20], reversibility in process calculi is challenging, since we cannot distinguish between the processes $a\|a$ and $a.a$ by simply recording the past actions. For this reason both histories and unique identifiers for threads have been used to track information. We do not have this problem in our calculus since each session participant reduces in a sequential way. A key requirement, dubbed *causal consistency* in [6], is that of undoing actions only if no other action depending on them has been executed (and not undone). In the present work causal consistency follows from the linearity of the interactions described by single-threaded global types.

The most widespread models of structured communication-based programming are session behaviours [3, 4] and session calculi [12, 13]. Reversibility has been incorporated into both these models.

Compliance and sub-behaviour for session behaviours with checkpoints has been first studied in [1]. There a process has the possibility, after a rollback, of resuming the computation along the very same branch of the computation on which the rollback has been performed. From a different point of view, instead, rollbacks could be used as a strategy to get compliance. For instance assuming the interacting processes to roll back whenever the current branch of the computation cannot proceed and a different branch could work instead. This approach has been investigated in [2].

The papers closer to ours are [22, 23, 17]. Tiezzi and Yoshida [22] use tags and memories to allow reversibility of binary sessions with delegation. Reversibility is full, i.e. each interaction can be undone and causal consistency is preserved. An extension of this calculus allows computation to go forward and backward until the session is committed by means of a specific irreversible action. Only processes are typed, but this is enough to ensure absence of errors. Two forms of reversibility are considered in [23]. Either a session can be completely reversed with one backward step, or any intermediate state can be restored with either one backward step or multiple ones. In the first case the memory is just the initial process, while in the second case the sequence of all the processes generated by the reduction is needed. Both binary and multiparty sessions are taken into account under the hypothesis that they are "single". A session is single when all participants interact only along that session. Mezzina and Pérez [17] use monitors as memories. A key novelty of [17] are session types with present and past, which allow the semantics of reversible actions to be streamlined.

The main contributions of this paper are the treatment of checkpointed interactions and the role played by global types in controlling reversibility. In defining of our calculus, we made some design choices. For simplicity, we did not consider

- session initialisation by means of request/accept,
- subsorting and covariance/contravariance of messages types in the subtyping of session types,
- asynchronous communications using message queues.

Including these features, which are present in [13, 11, 23], would be easy.
Moreover we did the following assumptions:

- the rollback to a checkpoint is done non-deterministically and simultaneously by all participants which traversed that checkpoint,
- all the communications can be undone.

In future work we plan to address the issue of communication that cannot be undone, such as "money dispensed by an ATM machine", and also add to the process language primitives triggering the rollback. In our calculus, when crossing a checkpoint we memorise all the branches of the choice. Including only the branches not taken, would get us also checkpointed single inputs/outputs, and this would require some care.

We will also study rollbacks with checkpoints for interleaved multiparty sessions with delegation. In this case, a crucial point is the dependency between different sessions when backward reductions are done, see [22].

Acknowledgments. We are grateful to the anonymous reviewers for their useful suggestions, which led to substantial improvements.

References

[1] Franco Barbanera, Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2016): *Reversible client/server interactions*. *Formal Aspects of Computing* 28(4), pp. 697–722, doi:10.1007/s00165-016-0358-2.

[2] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese & Ugo de' Liguoro (2016): *Retractable Contracts*. In: *PLACES*, *EPTCS* 203, pp. 61–72, doi:10.4204/EPTCS.203.

[3] Franco Barbanera & Ugo de' Liguoro (2015): *Sub-behaviour relations for session-based client/server systems*. Mathematical Structures in Computer Science 25(6), pp. 1339–1381, doi:10.1017/S096012951400005X.

[4] Giovanni Bernardi & Matthew Hennessy (2016): *Modelling session types using contracts*. Mathematical Structures in Computer Science 26(3), pp. 510–560, doi:10.1017/S0960129514000243.

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

[6] Vincent Danos & Jean Krivine (2004): *Reversible Communicating Systems*. In: *CONCUR*, *LNCS* 3170, Springer, pp. 292–307, doi:10.1007/978-3-540-28644-8_19.

[7] Edsko de Vries, Vasileios Koutavas & Matthew Hennessy (2010): *Communicating Transactions - (Extended Abstract)*. In: *CONCUR*, *LNCS* 6269, Springer, pp. 569–583, doi:10.1007/978-3-642-15375-4_39.

[8] Edsko de Vries, Vasileios Koutavas & Matthew Hennessy (2010): *Liveness of Communicating Transactions - (Extended Abstract)*. In: *APLAS*, *LNCS* 6461, Springer, pp. 392–407, doi:10.1007/978-3-642-17164-2_27.

[9] Pierre-Malo Deniélou & Nobuko Yoshida (2011): *Dynamic Multirole Session Types*. In: *POPL*, ACM Press, pp. 435–446, doi:10.1145/1926385.1926435.

[10] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic & Nobuko Yoshida (2016): *Precise subtyping for synchronous multiparty sessions*. In: *PLACES*, *EPTCS* 203, pp. 29–43, doi:10.4204/EPTCS.203.3.

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

[12] Kohei Honda, Vasco T. Vasconcelos & Makoto Kubo (1998): *Language Primitives and Type Disciplines for Structured Communication-based Programming*. In: *ESOP*, *LNCS* 1381, Springer, pp. 22–138, doi:10.1007/BFb0053567.

[13] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty Asynchronous Session Types*. In: *POPL*, ACM Press, pp. 273–284, doi:10.1145/1328897.1328472.

[14] Vasileios Koutavas, Carlo Spaccasassi & Matthew Hennessy (2014): *Bisimulations for Communicating Transactions - (Extended Abstract)*. In: *FOSSACS*, *LNCS* 8412, Springer, pp. 320–334, doi:10.1007/978-3-642-54830-7_21.

[15] Ivan Lanese, Claudio Antares Mezzina, Alan Schmitt & Jean-Bernard Stefani (2011): *Controlling Reversibility in Higher-Order Pi*. In: *CONCUR*, *LNCS* 6901, Springer, pp. 297–311, doi:10.1007/978-3-642-23217-6_20.

[16] Ivan Lanese, Claudio Antares Mezzina & Jean-Bernard Stefani (2010): *Reversing Higher-Order Pi*. In: *CONCUR*, *LNCS* 6269, Springer, pp. 478–493, doi:10.1007/978-3-642-15375-4_33.

[17] Claudio A. Mezzina & Jorge A. Pérez (2016): *Reversible Sessions Using Monitors*. In: *PLACES*, *EPTCS* 211, pp. 56–64, doi:10.4204/EPTCS.211.6.

[18] Robin Milner (1989): *Communication and concurrency*. PHI Series in computer science, Prentice Hall.

[19] Luca Padovani (2011): *Session Types = Intersection Types + Union Types*. In: *ITRS*, *EPTCS* 45, pp. 71–89, doi:10.4204/EPTCS.45.6.

[20] Iain C. C. Phillips & Irek Ulidowski (2007): *Reversing algebraic process calculi*. Journal of Logic and Algebraic Methods in Programming 73(1-2), pp. 70–96, doi:10.1016/j.jlap.2006.11.002.

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

[22] Francesco Tiezzi & Nobuko Yoshida (2015): *Reversible Session-Based Pi-Calculus*. Journal of Logical and Algebraic Methods in Programming 84(5), pp. 684–707, doi:10.1016/j.jlamp.2015.03.004.

[23] Francesco Tiezzi & Nobuko Yoshida (2016): *Reversing Single Sessions*. In: *RC*, *LNCS* 9720, Springer, pp. 52–69, doi:10.1007/978-3-319-40578-0_4.