Acta Informatica
https://doi.org/10.1007/s00236-019-00332-y

ORIGINAL ARTICLE

Reversible sessions with flexible choices

Ilaria Castellani[1] · Mariangiola Dezani-Ciancaglini[2] · Paola Giannini[3]

Received: 16 April 2018 / Accepted: 8 February 2019

Abstract

We propose a calculus for concurrent reversible multiparty sessions, equipped with a flexible choice operator allowing for different sets of participants in each branch. This operator is inspired by the notion of *connecting action* recently introduced by Hu and Yoshida to describe protocols with optional participants. We argue that this choice operator allows for a natural description of typical communication protocols. Our calculus also supports a compact representation of the history of processes and types, which facilitates the definition of rollback. Moreover, it implements a fine-tuned strategy for backward computation. We present a session type system for the calculus and show that it enforces the expected properties of session fidelity, forward progress and backward progress.

1 Introduction

Session types are a simple but expressive type formalism that specifies the structure of interactions. Traditionally, session types have been used to ensure safety properties of interactions, such as absence of communication errors, deadlock freedom and race freedom.

Reversibility is a means to improve system flexibility and reliability. Reversing a computation may be defined as the act of undoing some suffix of the computation, in order to return to a previously visited state. In a nondeterministic computation, this possibility may be used

Mariangiola Dezani-Ciancaglini: Partially supported by EU H2020-644235 Rephrase project, EU H2020-644298 HyVar project, IC1402 ARVI and Ateneo/CSP project RunVar.
Paola Giannini: This original research has the financial support of the Università del Piemonte Orientale.
The authors acknowledge a partial support of COST Action IC1405 on Reversible Computation—extending horizons of computing.

✉ Paola Giannini
paola.giannini@uniupo.it

Ilaria Castellani
ilaria.castellani@inria.fr

Mariangiola Dezani-Ciancaglini
dezani@di.unito.it

[1] INRIA, Université Côte d'Azur, 2004 Route des Lucioles, 06902 Sophia Antipolis, France

[2] Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10131 Turin, Italy

[3] DiSIT, Università del Piemonte Orientale, Via Teresa Michel 11, 15121 Alessandria, Italy

to return to a previous branching point, in case the chosen branch has led to an unsuccessful state.

In the setting of structured communications, reversibility has been first studied for contracts [2,3] and transactions [11,12,21]. More recently, it has started to be investigated also for session calculi, both binary [24,34] and multiparty [14,26,28,35] (see Sect. 6 for more discussion on related work).

When reversing a structured interaction, one has to preserve the consistency of the global state: if one partner triggers a rollback, then all its communicating partners should roll back accordingly. Session types turn out to be very useful here, since they specify both the functionality of communications (sender, receiver and type of message), and the order in which they should occur.

We present a calculus for concurrent reversible multiparty sessions, whose distinctive feature is a flexible choice operator, allowing for different sets of participants in its branches. The only participant which is required to occur in all branches is the one which solves the choice, henceforth called the *choice leader*.

Our choice operator is inspired by the notion of *connecting action* recently introduced by Hu and Yoshida to describe protocols with optional participants [20]. The intuition behind connecting actions is that in some parts of the protocol, delimited by a choice construct, some participants are required to take part in the interaction while some others may be optional. Connecting actions are used to invite optional participants to join the interaction along some branches of the choice. For instance, in a PC meeting, it could happen that in case of divergent views about a paper, the PC chair launches a discussion among the concerned PC members, but also invites some additional PC members to join the discussion. These additional members are optional in the sense that they are not required to discuss on that particular paper, but they may be invited to do so in some cases.

Here we shall make a more permissive use of connecting actions than in [20].

The differences between our connecting actions and those in [20] are discussed in Sect. 6.

Compared to the standard choice operator of multiparty session calculi, our flexible choice allows for a more natural description of typical communication protocols, such as the one mentioned above. Another example is the vacation protocol discussed below, where Alice has to decide between two destinations, and depending on her decision she will wish to contact either an airline or a railway company but not both. This will be modelled by a choice between two connecting actions, one with the airline and one with the railway company.

Another notable feature of our calculus is that it gives a compact representation of the *history* of processes and types, which facilitates the definition of rollback. It also implements a fine-tuned strategy for backward computation, which is geared towards achieving compliance. In essence, a backward move can only return to a past choice point, and it can only be triggered by the leader of that choice; moreover, the past choice state is restored without the already explored branch, thus forcing the choice leader to engage into a different branch.

The main contributions of our paper may be summarised as follows:

– the introduction of a flexible choice operator based on *connecting actions* in a reversible multiparty session framework;
– a fine-tuned strategy for rollback to checkpointed choices, whereby rollback can only be triggered by choice leaders in predefined states of the computation, leading back to the choice state stripped off the unsuccessful path.

This work builds on our previous papers [8,14]. As regards the treatment of reversibility in multiparty sessions, the general principles and the use of checkpoints for return points are taken from [14], while the formalisation of histories and the specific rollback mechanism are borrowed from [8]. The formalisation of histories is pushed a little further here than in [8], including notions of causality and conflict for communication occurrences. On the other hand, the use of connecting actions is new with respect to [8,14], and so is the study of their interplay with reversible computations. This interplay is not entirely trivial, as it requires revisiting the definition of projection of a global type onto its participants. A relevant new notion to this purpose is that of *affecting choice* for a participant: intuitively, a choice specified by a global type affects a participant if the whole choice is needed to determine the projection on that participant. By definition, a choice affects all its required participants while it does not affect optional participants which appear in only one branch. If we ignore connecting actions, the expressive power of our calculus is comprised between that of [14] and that of [8]. Indeed, [8] allows for both parallel and sequential composition within processes and types, yielding a powerful but somewhat complex calculus. Here we decided to stick to a more standard syntax, in order to be able to focus on the main topic of the paper, namely backward computation in the presence of connecting actions.

Vacation protocol example To illustrate our approach, we present a simple protocol involving five parties, Alice, Bob, Carol, the airline company Alitalia and the railway company Trenitalia. This protocol, henceforth called the *vacation protocol*, will serve as our running example throughout the paper.

For Easter holidays, Alice receives two independent invitations from Bob and Carol. Bob proposes to Alice to visit him in Paris, while Carol offers to host her in her seaside house in Amalfi. Since Alice lives in Rome, depending on her decision she will need to book a flight to go to Paris or a train to reach Amalfi. To do so, Alice will either contact Alitalia or Trenitalia to buy a ticket, and then inform both Bob and Carol of her decision. In case of a plane or train strike, Alice is allowed to change her mind. Letting $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{f}, \mathsf{t}$ denote respectively Alice, Bob, Carol, Alitalia and Trenitalia, a global type describing this communication protocol is:

$$\mathsf{b} \xrightarrow{iP} \mathsf{a};\, \mathsf{c} \xrightarrow{iA} \mathsf{a};\, (\mathsf{a} \overset{tk}{\leftrightarrow} \mathsf{f};\, \mathsf{a} \xrightarrow{yes} \mathsf{b};\, \mathsf{a} \xrightarrow{no} \mathsf{c}\ {}_{C}\boxplus\ \mathsf{a} \overset{tk}{\leftrightarrow} \mathsf{t};\, \mathsf{a} \xrightarrow{no} \mathsf{b};\, \mathsf{a} \xrightarrow{yes} \mathsf{c})$$

where ${}_{C}\boxplus$ is a choice with checkpoint label C, meaning that the choice leader (i.e. the participant who makes the choice, Alice in this case) can rollback to take a different branch of the choice. The communications between Alice and Alitalia/Trenitalia are *connecting communications* (represented by messages of the form $\overset{\lambda}{\leftrightarrow}$). This means that there is no assurance for Alitalia and Trenitalia that they will receive a message from Alice. With standard global types, Alitalia and Trenitalia should receive a message from Alice in both branches of the choice, and this is clearly not realistic. A different way out would be to use three global types instead of one, respectively with sets of participants {Alice, Bob, Carol}, {Alice, Alitalia} and {Alice, Trenitalia}. The drawback of the latter solution is that, in the presence of more than one global type, a simple session type system is not sufficient to ensure progress, and one needs to recourse to more refined type systems [29].

Thanks to connecting actions, the vacation protocol may be described by a single global type, without the constraints of standard global types.

Outline The rest of the paper is organised as follows. In Sect. 2 we define the syntax and operational semantics—both forward and backward—of our calculus. In Sect. 3 we introduce our syntax for global types and session types and we establish well-formedness conditions for global types. In Sect. 4 we present our type system and prove some preliminary properties. In Sect. 5 we prove the soundness of our type system, namely that it ensures the expected semantic properties of session fidelity and forward and backward progress. We conclude in Sect. 6 with some discussion on related and future work. The Appendix proves the correctness of a possible implementation of backward reduction.

2 Calculus

We assume the following base sets: *simple messages*, ranged over by $\lambda, \lambda', \ldots$ and forming the set Msg; *connecting messages*, ranged over by $\overset{\lambda}{\leftrightarrow}, \overset{\lambda'}{\leftrightarrow}, \ldots$ and forming the set CMsg; *checkpoint labels*, ranged over by C, C' and forming the set ChLa; and *session participants*, ranged over by $\mathsf{p}, \mathsf{q}, \mathsf{r}$ and forming the set Part. We use Λ to range over both simple messages and connecting messages.

We use Δ to range over sets of checkpoint labels, and $\overline{\gamma}$ to stand for either the empty set or a singleton consisting of an overlined checkpoint label (curly brackets will be omitted around singletons):

$$\Delta ::= \emptyset \mid \Delta, C \qquad\qquad \overline{\gamma} ::= \emptyset \mid \overline{C}$$

Sets Δ and $\overline{\gamma}$ are associated with choices in processes. More precisely, a set Δ is associated with an external choice and said to be *passive*, while a set $\overline{\gamma}$ is associated with an internal choice and said to be *active*. Intuitively, an overlined label $\overline{C}$ is the handle for a backward move: a participant who crossed an internal choice (henceforth called the *choice leader*) with checkpoint label $\overline{C}$, and then proceeded in the computation, may decide to return to that choice whenever she has the ability to send a message. Within a network, this backward move of the choice leader will have to be matched by backward moves of all the participants who did some action after crossing the matching external choice, whose checkpoint set contains label C. The asymmetry between Δ and $\overline{\gamma}$ is justified by the fact that there is only one choice leader who can send messages to various participants.

Let $\pi \in \{\mathsf{p}?\Lambda, \mathsf{p}!\Lambda \mid \mathsf{p} \in \mathsf{Part}, \Lambda \in \mathsf{Msg} \cup \mathsf{CMsg}\}$ denote an *atomic action*, namely a simple input/output action or an input/output action establishing a connection. As in [20], connecting inputs may be dangling forever, whereas simple inputs will eventually take place. This gives a natural freedom in the definition of communication protocols as illustrated in the examples of the introduction. An atomic action can bear a hat, in which case it represents an already *executed action*. We use $\widetilde{\pi}$ to stand for either π or $\widehat{\pi}$. External choices and internal choices are denoted by $\sum$ and $\bigoplus$, respectively.

Definition 1 (*Processes*) *Processes* are defined by:

$$P ::= {}_{\Delta}\sum_{i\in I} \widetilde{\mathsf{p}_i?\Lambda_i}; P_i \;\mid\; {}_{\overline{\gamma}}\bigoplus_{i\in I} \widetilde{\mathsf{p}_i!\Lambda_i}; P_i \;\mid\; \mu X.P \;\mid\; X \;\mid\; \mathsf{end}$$

where in both kinds of choice the pairs $(\mathsf{p}_i, \Lambda_i)$ are assumed to be all distinct. Moreover, for external choices we also assume that the Λ_i's are either all simple or all connecting messages. This condition allows us to distinguish between simple and connecting external choices according to the kind of their inputs. This is essential for requiring that at least one of the inputs in a simple external choice will eventually be matched by a message. Instead all inputs in a connecting external choice can wait forever. Processes without hats are called *user processes* and the others are *runtime processes*.

We will omit empty sets of checkpoint labels, choice symbols in one-branch choices, and trailing end processes.

External and internal choices are assumed to be associative, commutative, and non-empty (except when combined with binary choices). A prefixed process may be either an *input process* or an *output process*. We require recursion to be guarded. Processes are treated equi-recursively, i.e. they are identified with their generated tree [33]. In other words, we consider processes up to the standard structural equivalence of processes.

The typing rules of Sect. 4 will ensure that in a choice, at most one of the first atomic actions bears a hat. This condition expresses the fact that executing a choice amounts to executing one of its branches.

In a full-fledged calculus, messages would carry values, namely they would be of the form $\Lambda(v)$. Here, for simplicity we consider only pure messages.

Networks are parallel compositions of pairs $\mathsf{p}[\![\ P\ ]\!]$, where participant p has behaviour P.

Definition 2 (*Networks*) *Networks* are defined by: $\quad \mathbb{N} \quad ::= \quad \mathsf{p}[\![\ P\ ]\!] \quad \mathbf{|} \quad \mathbb{N} \parallel \mathbb{N}$

The operator $\parallel$ is associative and commutative, with neutral element $\mathsf{p}[\![\ \mathsf{end}\ ]\!]$ for each p. These laws, together with the structural equivalence of processes, give the structural equivalence of networks.

Example 1 (*Networks*) A network for the vacation example of Sect. 1 is as follows:

$$\mathbb{N} = \mathsf{a}[\![\ P^{\mathsf{a}}\ ]\!] \parallel \mathsf{b}[\![\ P^{\mathsf{b}}\ ]\!] \parallel \mathsf{c}[\![\ P^{\mathsf{c}}\ ]\!] \parallel \mathsf{f}[\![\ P^{\mathsf{f}}\ ]\!] \parallel \mathsf{t}[\![\ P^{\mathsf{t}}\ ]\!]$$

where

$$P^{\mathsf{a}} = \mathsf{b}?iP;\ \mathsf{c}?iA;\ (P^{\mathsf{a}}_1 \ {}_{\overline{C}}\!\oplus\ P^{\mathsf{a}}_2) \text{ with } P^{\mathsf{a}}_1 = \mathsf{f}!\overset{tk}{\leftrightarrow};\ \mathsf{b}!yes;\ \mathsf{c}!no \text{ and } P^{\mathsf{a}}_2 = \mathsf{t}!\overset{tk}{\leftrightarrow};\ \mathsf{b}!no;\ \mathsf{c}!yes$$
$$P^{\mathsf{b}} = \mathsf{a}!iP;\ (\mathsf{a}?yes\ {}_C\!+\ \mathsf{a}?no) \qquad P^{\mathsf{c}} = \mathsf{a}!iA;\ (\mathsf{a}?no\ {}_C\!+\ \mathsf{a}?yes) \qquad P^{\mathsf{f}} = P^{\mathsf{t}} = {}_C\mathsf{a}?\overset{tk}{\leftrightarrow}$$

The operational semantics is given by two LTSs, one for processes and one for networks. In the LTS for processes, *forward transitions* have the form $P \xrightarrow{\pi} P'$ and *backward transitions* have the form $P \overset{\overline{C}}{\circlearrowright} P'$ or $P \overset{C}{\circlearrowright} P'$. We define $P \downarrow_{out}$ if $P \xrightarrow{\mathsf{p}!\Lambda} P'$ for some p, Λ, P'. In the LTS for networks, forward and backward transitions have respectively the form $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$ and $\mathbb{N} \overset{C}{\circlearrowright} \mathbb{N}'$.

The LTSs for processes and networks are given in Fig. 1. Rules [ExtCh] and [IntCh] allow an action to be extracted from one of the summands, as usual, but instead of discarding the other summands they record the fact that the choice has been crossed by marking the executed action with a hat. With this technique, inspired by [6] and already used for reversible computations in [8,26], all the dynamic operators are turned into static operators and nothing is lost of the original user process. Notice that when $I = \{j\}$ these rules become $\pi_j;\ P_j \xrightarrow{\pi_j} \widehat{\pi_j};\ P_j$. Rule [BackA] is the main backward rule: it applies to a past internal choice in which one branch has been partially executed, and it allows the process to roll back to the original choice where the executed branch is removed.[1] For this to be possible, the choice must have at least one alternative branch P_i and an overlined label $\overline{C}$, which will label the back transition. This is essential to ensure (by means of typing) that the choice leader will be the only participant habilitated to trigger a rollback to this choice. The condition $P \downarrow_{out}$

[1] We could also leave the executed branch, just erasing its hats. This change would preserve all the properties of the calculus. We prefer the current formulation, since it avoids the possibility of going back and fourth several times along the same branch (which could yield a livelock).

$$ {}_{\Delta}\textstyle\sum_{i\in I}\pi_i;P_i \xrightarrow{\pi_j} {}_{\Delta}(\sum_{i\in I\setminus\{j\}}\pi_i;P_i \;+\; \widehat{\pi_j};P_j) \quad j\in I \qquad [\text{ExtCh}]$$

$$ {}_{\overline{\gamma}}\textstyle\bigoplus_{i\in I}\pi_i;P_i \xrightarrow{\pi_j} {}_{\overline{\gamma}}(\bigoplus_{i\in I\setminus\{j\}}\pi_i;P_i \;\oplus\; \widehat{\pi_j};P_j) \quad j\in I \qquad [\text{IntCh}]$$

$$\frac{P\downarrow_{out} \qquad I\neq\emptyset}{{}_{\overline{C}}(\bigoplus_{i\in I}\pi_i;P_i \;\oplus\; \widehat{\pi};P) \overset{\overline{C}}{\circlearrowright} {}_{\overline{C}}\bigoplus_{i\in I}\pi_i;P_i}\,[\text{BackA}]$$

$$\frac{C\in\Delta}{{}_{\Delta}\sum_{i\in I}\widetilde{\pi_i};P_i \overset{C}{\circlearrowright} \wr{}_{\Delta}\sum_{i\in I}\widetilde{\pi_i};P_i\int}\,[\text{BackP}]$$

$$\frac{P\xrightarrow{\pi}P'}{\mathcal{E}[P]\xrightarrow{\pi}\mathcal{E}[P']}\,[\text{CtFAt}] \qquad \frac{P\overset{\overline{C}}{\circlearrowright}P' \quad \mathcal{E}\ \mathsf{ok}\ \text{for}\ \overline{C}}{\mathcal{E}[P]\overset{\overline{C}}{\circlearrowright}\mathcal{E}[P']}\,[\text{CtBA}] \qquad \frac{P\overset{C}{\circlearrowright}P' \quad \mathcal{E}\ \mathsf{ok}\ \text{for}\ C}{\mathcal{E}[P]\overset{C}{\circlearrowright}\mathcal{E}[P']}\,[\text{CtBP}]$$

$$\frac{P\xrightarrow{\mathsf{q}!\Lambda}P' \qquad Q\xrightarrow{\mathsf{p}?\Lambda}Q'}{\mathsf{p}[\![\,P\,]\!]\parallel\mathsf{q}[\![\,Q\,]\!]\parallel\mathbb{N}\xrightarrow{\mathsf{p}\Lambda\mathsf{q}}\mathsf{p}[\![\,P'\,]\!]\parallel\mathsf{q}[\![\,Q'\,]\!]\parallel\mathbb{N}}\,[\text{Com}]$$

$$\frac{P\overset{\overline{C}}{\circlearrowright}P' \quad P_h\overset{C}{\circlearrowright}P'_h \;\; h\in H \quad P_k\overset{C}{\not\circlearrowright} \;\; k\in K}{\mathsf{p}[\![\,P\,]\!]\parallel\Pi_{h\in H}\mathsf{p}_h[\![\,P_h\,]\!]\parallel\Pi_{k\in K}\mathsf{p}_k[\![\,P_k\,]\!]\overset{C}{\circlearrowright}\mathsf{p}[\![\,P'\,]\!]\parallel\Pi_{h\in H}\mathsf{p}_h[\![\,P'_h\,]\!]\parallel\Pi_{k\in K}\mathsf{p}_k[\![\,P_k\,]\!]}\,[\text{Back}]$$

Fig. 1 LTS for processes and networks

means that in order to trigger a rollback, the process P must be "in lead", namely able to do an output. In this way, rolling back acts as an alternative to one of its possible outputs.

Rule [BackP] is needed to allow the remaining participants to roll back. The mapping $\wr\ \int$ erases hats from processes, yielding user processes, i.e.

$$\wr{}_{\Delta}\sum_{i\in I}\widetilde{\pi_i};\,P_i\int = {}_{\Delta}\left(\sum_{i\in I}\pi_i;\ \wr\,P_i\int\right) \qquad \wr{}_{\overline{\gamma}}\bigoplus_{i\in I}\widetilde{\pi_i};\,P_i\int = {}_{\overline{\gamma}}\left(\bigoplus_{i\in I}\pi_i;\ \wr\,P_i\int\right)$$

and $\wr\ \int$ acts homomorphically otherwise. Note that the rollback rules can only be applied to processes that are not user processes: in particular, one branch can be erased only if at least one of its actions has been executed.

Evaluation contexts are as expected.

Definition 3 (*Evaluation contexts*) *Evaluation contexts* $\mathcal{E}$ are defined by:

$$\mathcal{E} ::= {}_{\Delta}\left(\sum_{h\in H}\mathsf{p}_h?\Lambda_h;\,P_h \;+\; \widehat{\mathsf{p}?\Lambda;\,\mathcal{E}}\right) \;\mid\; {}_{\overline{\gamma}}\left(\bigoplus_{h\in H}\mathsf{p}_h!\Lambda_h;\,P_h \;\oplus\; \widehat{\mathsf{p}!\Lambda;\,\mathcal{E}}\right) \;\mid\; [\;]$$

An evaluation context $\mathcal{E}$ is ok for C ($\overline{C}$) if $C\notin\Delta$ ($\overline{C}\neq\overline{\gamma}$) whenever $\mathcal{E}$ has a sub-context of the shape ${}_{\Delta}(\sum_{h\in H}P_h \;+\; \widehat{\pi};\,\mathcal{E}')$ (${}_{\overline{\gamma}}(\bigoplus_{h\in H}P_h \;\oplus\; \widehat{\pi};\,\mathcal{E}')$). We use this condition in rules [CtBA] and [CtBP] to assure that all participants involved in a recursion go back to the same checkpoint, namely to the outermost one, as in [28]. This is needed to assure subject reduction, see Example 6.

Rule [Com] is standard and deals with both simple and connecting messages. We write $P \overset{C}{\not\circlearrowright}$ if Rule [CtBP] (with label C) cannot be applied to P. This means that C can only occur in user processes within P. In other words, P does not contain an executed ${}_{\Delta}+$ with $C \in \Delta$. In a well-typed network, Rule [Back] will make participant p roll back to an internal choice and moreover, all participants that can roll back to corresponding external choices will do so in the same step. This will be the basis for our soundness result in Sect. 5. Notice that the only requirements of rule [Back] concern process P, since it is easy to verify that either $Q \overset{C}{\circlearrowright} Q'$ or $Q \overset{C}{\not\circlearrowright}$ for any other process Q. Note that a direct implementation of rule [Back] would be unrealistic. We discuss a possible asynchronous implementation of this rule at the end of the section.

When the labels of transitions are not relevant, we write them simply as $\longrightarrow$ and $\circlearrowright$. In this case, we use $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$ and $\overset{\circlearrowright}{\longrightarrow}{}^*$ to denote the reflexive and transitive closure of $\longrightarrow \cup \circlearrowright$.

Example 2 (*Reduction of networks*) We describe the evolution of the network $\mathbb{N}$ of Example 1. At each step we only show the participants that are modified by the reduction.

$$
\begin{aligned}
\mathbb{N} &\xrightarrow{\mathsf{b}\, iP\, \mathsf{a}} \mathsf{a}[\![\, \widehat{\mathsf{b}?iP}; \mathsf{c}?iA; (P^{\mathsf{a}}_1 \,{}_{\overline{C}}\!\oplus\; P^{\mathsf{a}}_2) \,]\!] \;\;\|\;\; \mathsf{b}[\![\, \widehat{\mathsf{a}!iP}; (\mathsf{a}?yes \;{}_{C}\!+\; \mathsf{a}?no) \,]\!] \;\;\|\;\; \cdots \\
&\xrightarrow{\mathsf{c}\, iA\, \mathsf{a}} \mathsf{a}[\![\, \widehat{\mathsf{b}?iP}; \widehat{\mathsf{c}?iA}; (P^{\mathsf{a}}_1 \,{}_{\overline{C}}\!\oplus\; P^{\mathsf{a}}_2) \,]\!] \;\;\|\;\; \mathsf{c}[\![\, \widehat{\mathsf{a}!iA}; (\mathsf{a}?no \;{}_{C}\!+\; \mathsf{a}?yes) \,]\!] \;\;\|\;\; \cdots \\
&\xrightarrow{\mathsf{a} \overset{tk}{\leftrightarrow} \mathsf{f}} \mathsf{a}[\![\, \widehat{\mathsf{b}?iP}; \widehat{\mathsf{c}?iA}; (\widehat{\mathsf{f}! \overset{tk}{\leftrightarrow}}; \mathsf{b}!yes; \mathsf{c}!no \;{}_{\overline{C}}\!\oplus\; P^{\mathsf{a}}_2) \,]\!] \;\;\|\;\; \mathsf{f}[\![\, \widehat{{}_{C}\mathsf{a}? \overset{tk}{\leftrightarrow}} \,]\!] \;\;\|\;\; \cdots \\
&\xrightarrow{\mathsf{a}\, yes\, \mathsf{b}} \mathsf{a}[\![\, \widehat{\mathsf{b}?iP}; \widehat{\mathsf{c}?iA}; (\widehat{\mathsf{f}! \overset{tk}{\leftrightarrow}}; \widehat{\mathsf{b}!yes}; \mathsf{c}!no \;{}_{\overline{C}}\!\oplus\; P^{\mathsf{a}}_2) \,]\!] \;\;\|\;\; \mathsf{b}[\![\, \widehat{\mathsf{a}!iP}; (\widehat{\mathsf{a}?yes} \;{}_{C}\!+\; \mathsf{a}?no) \,]\!] \;\;\|\;\; \cdots \\
&\overset{\overset{C}{\circlearrowright}}{\longrightarrow} \mathsf{a}[\![\, \widehat{\mathsf{b}?iP}; \widehat{\mathsf{c}?iA}; {}_{\overline{C}}P^{\mathsf{a}}_2 \,]\!] \;\;\|\;\; \mathsf{b}[\![\, \widehat{\mathsf{a}!iP}; (\mathsf{a}?yes \;{}_{C}\!+ \mathsf{a}?no) \,]\!] \;\;\|\;\; \mathsf{f}[\![\, {}_{C}\mathsf{a}? \overset{tk}{\leftrightarrow} \,]\!] \;\;\|\;\; \cdots
\end{aligned}
$$

The last rollback is triggered by a [BackA] move of Alice synchronised with [BackP] moves of Bob and Alitalia.

In the rest of this section we discuss a possible implementation of rule [Back], which decomposes the simultaneous rollback of a set of participants into a sequence of backward moves, one for each participant in the set. The first move of the sequence is always that of the choice leader.

We introduce states to define systems, which are pairs of networks and states.

States are defined by $\sigma \;:=\; \langle c, \mathbb{P}_1, \mathbb{P}_2 \rangle$, where c denotes either a checkpoint label C or '$-$', and $\mathbb{P}_1, \mathbb{P}_2$ form a partition on the set of network participants. In a state of the form $\langle -, \mathbb{P}_1, \mathbb{P}_2 \rangle$, the set $\mathbb{P}_2$ is always $\emptyset$. A state is *ready* if it is of the shape $\langle -, \mathbb{P}, \emptyset \rangle$. A communication can only take place in a ready state, yielding a ready state again. Similarly, the starting move of a rollback can only take place in a ready state, but it gives rise to a state of the form $\langle C, \mathbb{P}_1, \mathbb{P}_2 \rangle$. In a state $\langle C, \mathbb{P}_1, \mathbb{P}_2 \rangle$, a rollback is underway: the participants in $\mathbb{P}_1$ have already contributed to the rollback, while the participants in $\mathbb{P}_2$ still need to respond to the rollback "call" from the choice leader.

Systems have the form $\mathbb{N} \between \langle c, \mathbb{P}_1, \mathbb{P}_2 \rangle$, where $\mathbb{P}_1 \cup \mathbb{P}_2$ is the participant set of $\mathbb{N}$. A system $\mathbb{N} \between \sigma$ is *ready* if the state σ is ready. Figure 2 gives the LTS for systems. We use the labelled double arrows $\overset{\alpha}{\Rightarrow}$ and the labelled curly arrows $\overset{\mathsf{s}_C}{\circlearrowright}, \overset{\mathsf{y}_C}{\circlearrowright}, \overset{\mathsf{n}_C}{\circlearrowright}, \overset{\mathsf{e}_C}{\circlearrowright}$ for communication and backward moves of systems, respectively. In this way the labelled arrows for networks and systems are disjoint.

The reverse rule [Back] of Fig. 1 splits into four rules for systems:

$$\frac{P_1 \xrightarrow{\mathsf{p}?\Lambda} P_1' \quad P_2 \xrightarrow{\mathsf{q}!\Lambda} P_2'}{\mathsf{q}\llbracket\, P_1 \,\rrbracket \parallel \mathsf{p}\llbracket\, P_2 \,\rrbracket \parallel \mathbb{N} \between \langle -, \mathbb{P}, \emptyset\rangle \xRightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathsf{q}\llbracket\, P_1' \,\rrbracket \parallel \mathsf{p}\llbracket\, P_2' \,\rrbracket \parallel \mathbb{N} \between \langle -, \mathbb{P}, \emptyset\rangle}\ [\textsc{ComS}]$$

$$\frac{P \overset{\overline{C}}{\circlearrowleft} P'}{\mathsf{p}\llbracket\, P \,\rrbracket \parallel \mathbb{N} \between \langle -, \mathbb{P} \cup \{\mathsf{p}\}, \emptyset\rangle \overset{\mathsf{s}C}{\circlearrowleft} \mathsf{p}\llbracket\, P' \,\rrbracket \parallel \mathbb{N} \between \langle C, \{\mathsf{p}\}, \mathbb{P}\rangle}\ [\textsc{BackS}]$$

$$\frac{P \overset{C}{\circlearrowleft} P'}{\mathsf{p}\llbracket\, P \,\rrbracket \parallel \mathbb{N} \between \langle C, \mathbb{P}, \mathbb{P}' \cup \{\mathsf{p}\}\rangle \overset{\mathsf{y}C}{\circlearrowleft} \mathsf{p}\llbracket\, P' \,\rrbracket \parallel \mathbb{N} \between \langle C, \mathbb{P} \cup \{\mathsf{p}\}, \mathbb{P}'\rangle}\ [\textsc{BackY}]$$

$$\frac{P \overset{C}{\not\circlearrowleft}}{\mathsf{p}\llbracket\, P \,\rrbracket \parallel \mathbb{N} \between \langle C, \mathbb{P}, \mathbb{P}' \cup \{\mathsf{p}\}\rangle \overset{\mathsf{n}C}{\circlearrowleft} \mathsf{p}\llbracket\, P \,\rrbracket \parallel \mathbb{N} \between \langle C, \mathbb{P} \cup \{\mathsf{p}\}, \mathbb{P}'\rangle}\ [\textsc{BackN}]$$

$$\mathbb{N} \between \langle C, \mathbb{P}, \emptyset\rangle \overset{\mathsf{e}C}{\circlearrowleft} \mathbb{N} \between \langle -, \mathbb{P}, \emptyset\rangle \qquad [\textsc{BackE}]$$

Fig. 2 LTS for systems

- rule [BACKS] allows the choice leader p to start a rollback: the new state contains the checkpoint label and all participants but p are required to roll back, if possible;
- rule [BACKY] allows participant p to roll back to the checkpoint label memorised in the state: the new state records the rollback of p;
- rule [BACKN] modifies the state recording that participant p cannot roll back with the current checkpoint label;
- rule [BACKE] ends the rollback by erasing from the state the current checkpoint label when all participants have become aware of the rollback.

The participants different from the choice leader are split into those that may roll back and those that may not.

The first ones reduce by rule [BACKY] and the second ones reduce by rule [BACKN].

In “Appendix A” we prove the equivalence between the LTS of Fig. 1 and the one of Fig. 2.

Example 3 (*Reduction of systems*) We show now how the rules of Fig. 2 may be used to simulate the execution of the network $\mathbb{N}$ in Example 2. First observe that, since the initial state is $\sigma = \langle -, \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{f}, \mathsf{t}\}, \emptyset\rangle$, the first four communications of the system can be derived by rule [COMS], leaving the state σ unchanged and yielding the same network as in Example 2 (fourth line), namely $\mathbb{N}' = \mathbb{N}_1 \parallel \mathbb{N}_2$ where:

$$\mathbb{N}_1 = \mathsf{a}\llbracket\, \widehat{\mathsf{b}?iP}; \widehat{\mathsf{c}?iA}; (\widehat{\mathsf{f}! \overset{tk}{\leftrightarrow}}; \widehat{\mathsf{b}!yes}; \mathsf{c}!no\ _{\overline{C}} \oplus\ P_2^{\mathsf{a}}) \,\rrbracket \ \parallel\ \mathsf{b}\llbracket\, \widehat{\mathsf{a}!iP}; (\widehat{\mathsf{a}?yes}\ _C +\ \mathsf{a}?no) \,\rrbracket$$

$$\mathbb{N}_2 = \mathsf{c}\llbracket\, \widehat{\mathsf{a}!iA}; (\mathsf{a}?no\ _C +\ \mathsf{a}?yes) \,\rrbracket \ \parallel\ \mathsf{f}\llbracket\, \widehat{_C\mathsf{a}? \overset{tk}{\leftrightarrow}} \,\rrbracket \ \parallel\ \mathsf{t}\llbracket\, _C\mathsf{a}? \overset{tk}{\leftrightarrow} \,\rrbracket$$

Then, an application of rule [BACKS] starts the rollback and a possible evolution of the system $\mathbb{N}' \between \sigma$ is the following:

$$\begin{array}{llr}
\mathbb{N}' \between \langle -, \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{f},\mathsf{t}\}, \emptyset\rangle & \overset{\mathsf{s}_C}{\circlearrowright}\ \mathsf{a}[\![\ \widehat{\mathsf{b}?iP}; \widehat{\mathsf{c}?iA}; {}_{\overline{C}}P^{\mathsf{a}}_2\ ]\!]\ \parallel \cdots \between \langle C, \{\mathsf{a}\}, \{\mathsf{b},\mathsf{c},\mathsf{f},\mathsf{t}\}\rangle & [\textsc{BackS}]\\
& \overset{\mathsf{y}_C}{\circlearrowright}\ \mathsf{b}[\![\ \widehat{\mathsf{a}!iP}; (\mathsf{a}?yes\ {}_C\!+\ \mathsf{a}?no)\ ]\!]\ \parallel \cdots \between \langle C, \{\mathsf{a},\mathsf{b}\}, \{\mathsf{c},\mathsf{f},\mathsf{t}\}\rangle & [\textsc{BackY}]\\
& \overset{\mathsf{n}_C}{\circlearrowright}\ \mathsf{c}[\![\ \widehat{\mathsf{a}!iA}; (\mathsf{a}?no\ {}_C\!+\ \mathsf{a}?yes)\ ]\!]\ \parallel \cdots \between \langle C, \{\mathsf{a},\mathsf{b},\mathsf{c}\}, \{\mathsf{f},\mathsf{t}\}\rangle & [\textsc{BackN}]\\
& \overset{\mathsf{y}_C}{\circlearrowright}\ \mathsf{f}[\![\ {}_C\mathsf{a}?\overset{tk}{\leftrightarrow}\ ]\!]\ \parallel \cdots \between \langle C, \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{f}\}, \{\mathsf{t}\}\rangle & [\textsc{BackY}]\\
& \overset{\mathsf{n}_C}{\circlearrowright}\ \mathsf{t}[\![\ {}_C\mathsf{a}?\overset{tk}{\leftrightarrow}\ ]\!]\ \parallel \cdots \between \langle C, \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{f},\mathsf{t}\}, \emptyset\rangle & [\textsc{BackN}]\\
& \overset{\mathsf{e}_C}{\circlearrowright}\ \cdots \between \langle -, \{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{f},\mathsf{t}\}, \emptyset\rangle & [\textsc{BackE}]
\end{array}$$

The application of rule [BackE] closes the rollback and brings the system back to the ready state σ. Note that the [BackN] and [BackY] moves can be performed in any order as long as they follow the [BackS] move by Alice. The network $\mathbb{N}$ is typable in the system of Sect. 4 by the global type given in the Introduction. By the Subject Reduction Theorem (Theorem 1) also the network $\mathbb{N}'$ is typable. This example shows that the four back rules of Fig. 2 are all used for reducing typable networks.

3 Global types and session types

According to [18,19], a *multiparty session* is a series of communications among participants [20], which follows a predefined protocol specified by a *global type*. Global types are built from choices among communications with the same sender, possibly using recursion. The choices are required to be non-ambiguous, i.e. to have either different receivers or different messages. As in processes, choices in global types have *checkpoint labels* which mark them as return points for rollbacks, and the executed part is highlighted with hats. For consistency, when a communication has a hat, then also all the communications that cause it (see Definition 7) must have a hat. This condition is expressed by Definition 8.

The inputs/outputs of each participant are determined by her session type, which is obtained by projecting the global type of the whole conversation. To define session types we start with the syntax of session pre-types (Definition 9) and then we single out session types (Definition 11) using the projection of global types (Fig. 5). The crucial point in the definition of projection is to determine when a participant which is not the choice leader of a choice may be realised by an external choice of processes. To this aim we define a meet operation on session pre-types (Definition 10).

We use γ to denote either the empty set or a singleton made of a checkpoint label:

$$\gamma ::= \emptyset \ \mid\ C$$

Sets γ will be associated with choices in global types.

Let $\alpha^{\mathsf{p}} \in \{\mathsf{p} \xrightarrow{\Lambda} \mathsf{q} \mid \mathsf{q} \in \mathsf{Part}, \Lambda \in \mathsf{Msg} \cup \mathsf{CMsg}\}$ denote an *atomic communication* with sender p. The communication $\mathsf{p} \xrightarrow{\lambda} \mathsf{q}$ is *simple*, while the communication $\mathsf{p} \xrightarrow{\overset{\lambda}{\leftrightarrow}} \mathsf{q}$ is *connecting*. An atomic communication can bear a hat, in notation $\widehat{\alpha^{\mathsf{p}}}$, in which case it represents a past or *executed* communication. The symbol $\widetilde{\alpha^{\mathsf{p}}}$ stands for either α^{p} or $\widehat{\alpha^{\mathsf{p}}}$. We write $\widetilde{\alpha}$ instead of $\widetilde{\alpha^{\mathsf{p}}}$ when the sender is not relevant. Given a communication $\alpha = \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$, we define the sender and receiver of α to be $\mathsf{sender}(\alpha) = \mathsf{p}$ and $\mathsf{recv}(\alpha) = \mathsf{q}$. Moreover, we denote by $\mathsf{part}(\alpha) = \{\mathsf{recv}(\alpha), \mathsf{sender}(\alpha)\}$ the participants of the communication α.

A global type K specifies an interaction that is still to start. A general global type G specifies a partially executed interaction, whose parts that have been discarded in choices or remain to be executed are specified by subterms K. By $|I|$ we denote the cardinality of the set I.

$$\boxed{\begin{array}{l}\mathsf{part}({}_\gamma \boxdot_{i\in I}\ \widetilde{\alpha_i};\mathsf{G}_i) = \bigcup_{i\in I}(\mathsf{part}(\alpha_i)\cup \mathsf{part}(\mathsf{G}_i))\\ \mathsf{part}(\mu\mathbf{t}.\mathsf{K}) = \mathsf{part}(\mathsf{K})\\ \mathsf{part}(\mathbf{t}) = \mathsf{part}(\mathsf{End}) = \emptyset\end{array}}$$

Fig. 3 Participants of global types

Definition 4 (*Global types*) *Global types* G are defined by:

$$\begin{array}{lcl}\mathsf{K} & ::= & {}_\gamma\,\alpha;\mathsf{K} \mid {}_\gamma\boxplus_{j\in J}\alpha^{\mathsf{p}}_j;\mathsf{K}_j \mid \mu\mathbf{t}.\mathsf{K} \mid \mathbf{t} \mid \mathsf{End}\\ \mathsf{G} & ::= & \mathsf{K} \mid {}_\gamma\,\widetilde{\alpha};\mathsf{G} \mid {}_\gamma(\boxplus_{i\in I}\alpha^{\mathsf{p}}_i;\mathsf{K}_i \boxplus \widehat{\alpha^{\mathsf{p}}}\ ;\ \mathsf{G})\end{array}$$

where $|J| > 1, |I| > 0$ and all atomic communications in a choice are different. I.e. $\alpha^{\mathsf{p}}_h \neq \alpha^{\mathsf{p}}_k$ for all $h \neq k \in J$ in a choice ${}_\gamma\boxplus_{j\in J}\alpha^{\mathsf{p}}_j;\mathsf{K}_j$, and $\alpha^{\mathsf{p}}_h \neq \alpha^{\mathsf{p}}_k$ for all $h \neq k \in I$ and $\alpha^{\mathsf{p}}_h \neq \alpha^{\mathsf{p}}$ for all $h \in I$ in a partially executed choice ${}_\gamma(\boxplus_{i\in I}\alpha^{\mathsf{p}}_i;\mathsf{K}_i \boxplus \widehat{\alpha^{\mathsf{p}}}\ ;\ \mathsf{G})$.

The type ${}_\gamma\widetilde{\alpha};\mathsf{G}$ represents a prefixing communication. The choice operator $\boxplus$ is n-ary ($n \geq 2$) and commutative, and ";" has higher precedence than $\boxplus$. The notation ${}_\gamma(\boxplus_{i\in I}\alpha^{\mathsf{p}}_i;\mathsf{K}_i \boxplus \widehat{\alpha^{\mathsf{p}}}\ ;\ \mathsf{G})$ stands for a choice where the branch initiating with $\widehat{\alpha^{\mathsf{p}}}$ has started to be executed, while the other branches have been discarded.

By abuse of notation, we will write ${}_\gamma\boxplus_{j\in J}\ \widetilde{\alpha^{\mathsf{p}}_j};\mathsf{G}_j$ for either ${}_\gamma\boxplus_{j\in J}\alpha^{\mathsf{p}}_j;\mathsf{G}_j$ or ${}_\gamma(\boxplus_{j\in J\setminus\{k\}}\alpha^{\mathsf{p}}_j;\mathsf{G}_j \boxplus \widehat{\alpha^{\mathsf{p}}_k}\ ;\ \mathsf{G}_k)$. In the examples we will write $\widetilde{\alpha^{\mathsf{p}}_1};\mathsf{G}_1\ {}_\gamma\boxplus\ \widetilde{\alpha^{\mathsf{p}}_2};\mathsf{G}_2$ and ${}_\gamma(\widetilde{\alpha^{\mathsf{p}}_1};\mathsf{G}_1 \boxplus \ldots \boxplus \widetilde{\alpha^{\mathsf{p}}_n};\mathsf{G}_n)$. (Note however that only one of the $\widetilde{\alpha^{\mathsf{p}}_i}$ may have a hat.)

We use ${}_\gamma\boxdot_{i\in I}\ \widetilde{\alpha^{\mathsf{p}}_i};\mathsf{G}_i$ to stand for ${}_\gamma\widetilde{\alpha^{\mathsf{p}}};\mathsf{G}$ when I is a singleton and for ${}_\gamma\boxplus_{i\in I}\ \widetilde{\alpha^{\mathsf{p}}_i};\mathsf{G}_i$ otherwise. Moreover, ${}_\gamma(\boxdot_{h\in H}\alpha^{\mathsf{p}}_h;\mathsf{G}_h\boxdot\widehat{\alpha^{\mathsf{p}}}\ ;\ \mathsf{G})$ stands for ${}_\gamma\widehat{\alpha^{\mathsf{p}}};\mathsf{G}$ when $H = \emptyset$. We write ${}_\gamma(\boxdot_{j\in J\setminus\{k\}}\widetilde{\alpha^{\mathsf{p}}_j};\mathsf{G}_j\boxdot\widetilde{\alpha^{\mathsf{p}}_k};\mathsf{G}_k)$ when we want to emphasise the branch $\widetilde{\alpha^{\mathsf{p}}_k};\mathsf{G}_k$.

When the checkpoint set γ associated with a choice is empty, we omit it. We call *rooted interaction* a subterm $\widetilde{\alpha};\mathsf{G}$. So $\widetilde{\alpha};\mathsf{G}$ can denote either a rooted interaction or a prefixing communication (the context will disambiguate if needed). A type ${}_\gamma\boxplus_{j\in J}\ \widetilde{\alpha_j};\mathsf{G}_j$ is a choice among more than one rooted interaction with the same sender, at most one of which bears a hat. Recursion must be guarded and it is treated equi-recursively.

We use $\mathsf{part}(\mathsf{G})$ to denote *the set of participants of* G, as defined in Fig. 3.

We represent global types as trees, with ${}_\gamma\boxplus$ or ${}_\gamma;$ on internal nodes, atomic communications on the branches, and End on the leaves. For simplicity we assume that each index set I, J, H is of the form $\{1,\ldots,n\}$, where n is its cardinality. The jth branch of a ${}_\gamma\boxplus$ node is labelled by $\widetilde{\alpha_j}$, and the unique branch of a ${}_\gamma;$ node, called 0th branch to distinguish it from proper choice branches, is labelled by the corresponding communication. In case the global type has some recursive subtype, the tree is an infinite (regular) tree.

Example 4 (*Trees*) Figure 4 shows the tree representing the global type

$$\mathsf{G} = \widehat{\mathsf{p} \xrightarrow{\lambda_0} \mathsf{q}};\mu\mathbf{t}.(\mathsf{q} \xrightarrow{\overset{\lambda_1}{\leftrightarrow}} \mathsf{r};\mathsf{r} \xrightarrow{\lambda_2} \mathsf{p};\mathbf{t}\ {}_C\boxplus \mathsf{q} \xrightarrow{\lambda_3} \mathsf{p};\mathsf{p} \xrightarrow{\lambda_4} \mathsf{q})$$

where α_i denotes the communication with message $\Lambda_i = \lambda_i$ or $\Lambda_i = \overset{\lambda_i}{\leftrightarrow}$ for $i = 0,\ldots,4$.

We introduce now a notation to distinguish different occurrences of the same communication $\widetilde{\alpha}$ within a type G.

Communication occurrences in G, denoted by ξ,ξ', have the form $\sigma\tilde{\alpha}$, where $\tilde{\alpha}$ is a possibly hatted communication and σ is a finite string over Nat, which represents the *path*

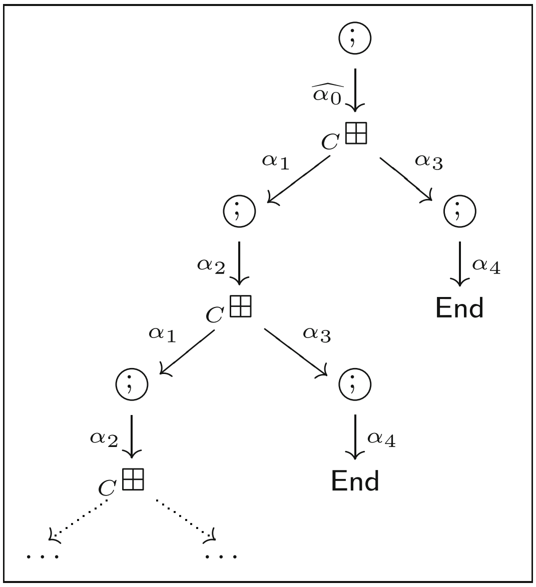

Fig. 4 Tree representation of $\mathsf{G} = \widehat{\alpha_0}; \mu\mathbf{t}.(\alpha_1; \alpha_2; \mathbf{t}\,{}_C\boxplus \alpha_3; \alpha_4)$

leading to that particular occurrence of α in the tree of G. Formally, the path σ records the branch chosen at each ${}_\gamma\boxplus$ or ${}_\gamma;$ node in G. In particular, the last element of σ specifies the branch labelled by the communication $\tilde{\alpha}$. (Note that in $\sigma\tilde{\alpha}$ the path σ is not empty since it contains at least the index of the branch labelled by $\tilde{\alpha}$.) We use $\sqsubseteq$ ($\sqsubset$) to denote the prefix ordering (strict prefix ordering) between strings of naturals.

For instance, the two shown occurrences of α_2 in the tree of Fig. 4 are identified by $010\alpha_2$ and $01010\alpha_2$.

Definition 5 (*Occurrences*) The set of *communication occurrences* of G, written *Occ*(G), is defined by:

- $Occ(\mathsf{End}) = \emptyset \qquad Occ(\mu\mathbf{t}.\mathsf{K}) = Occ(\mathsf{K}\{\mu\mathbf{t}.\mathsf{K}/\mathbf{t}\})$
- $Occ({}_\gamma\,\widetilde{\alpha}; \mathsf{G}) = \{0\widetilde{\alpha}\} \cup \{0\xi \mid \xi \in Occ(\mathsf{G})\}$
- $Occ({}_\gamma \boxplus_{j\in J}\ \widetilde{\alpha_j}; \mathsf{G}_j) = \{j\widetilde{\alpha_j} \mid j \in J\} \cup \{j\xi \mid j \in J \text{ and } \xi \in Occ(\mathsf{G}_j)\}$

In the following, communication occurrences will be simply referred to as *occurrences*. Given an occurrence $\xi = \sigma\widetilde{\alpha}$ of G, we define the path of ξ in G to be $\mathsf{path}(\xi) = \sigma$, the communication of ξ to be $\mathsf{comm}(\xi) = \widetilde{\alpha}$ and the participants of ξ to be $\mathsf{part}(\xi) = \mathsf{part}(\alpha)$.

An occurrence ξ is *executed* in G if $\mathsf{comm}(\xi)$ bears a hat. We define *ExOcc*(G), the *set of executed occurrences of* G, as follows:

$$ExOcc(\mathsf{G}) = \{\xi \mid \xi \in Occ(\mathsf{G}) \text{ and } \mathsf{comm}(\xi) = \widehat{\alpha} \text{ for some } \alpha\}$$

The relation of conflict on occurrences is handy. Intuitively, two occurrences are in conflict if they mutually exclude each other in any computation.

Definition 6 (*Conflict*) The *conflict relation* in G, denoted by $\#_\mathsf{G}$, is the symmetric irreflexive relation on *Occ*(G) defined by: $\xi \#_\mathsf{G} \xi'$ if there exist $\sigma_0, \sigma_1, \sigma_2$ such that $\mathsf{path}(\xi) = \sigma_0 i \sigma_1$ and $\mathsf{path}(\xi') = \sigma_0 j \sigma_2$ with $i \neq j$.

Two conflicting occurrences have paths that diverge at some choice node of the tree. Since the syntax of global types does not allow hats on more than one branch of a choice, this implies that two conflicting occurrences can never be both executed.

We now formalise the notion of causality on occurrences by defining the set of causes of a given occurrence. Intuitively, the set of causes of ξ in G is the set of occurrences in G on which ξ depends, namely those that need to be executed before ξ in any computation allowed by G.

Definition 7 (*Causes*)
Given an occurrence $\xi \in \mathit{Occ}(\mathsf{G})$, the *set of causes of* ξ *in* G, written $\mathit{Causes}(\mathsf{G}, \xi)$, is the smallest set that contains an occurrence ξ' if:

- either $\mathsf{path}(\xi') \sqsubset \mathsf{path}(\xi)$ and $\mathsf{part}(\xi') \cap \mathsf{part}(\xi) \neq \emptyset$;
- or $\xi' \in \mathit{Causes}(\mathsf{G}, \xi'')$ and $\xi'' \in \mathit{Causes}(\mathsf{G}, \xi)$ for some ξ''.

Our notions of conflict and causality are similar to those defined in [37] for Prime Event Structures (although simpler, since we only deal with trees here).

We require global types to respect causality of occurrences, as formalised in the following definition.

Definition 8 (*Causally correct global type*) A global type G is *causally correct* if, whenever an occurrence ξ is executed in G, then every occurrence in $\mathit{Causes}(\mathsf{G}, \xi)$ is executed in G, namely if $\xi \in \mathit{ExOcc}(\mathsf{G})$ implies $\mathit{Causes}(\mathsf{G}, \xi) \subseteq \mathit{ExOcc}(\mathsf{G})$.

The conflict relation is *hereditary*, namely if $\xi \;\#_G\; \xi'$ and $\xi' \in \mathit{Causes}(\mathsf{G}, \xi'')$, then $\xi \;\#_G\; \xi''$. To show this property (which is an axiom in Prime Event Structures), it is enough to observe that if $\mathsf{path}(\xi) = \sigma_0 i \sigma_1$ and $\mathsf{path}(\xi') = \sigma_0 j \sigma_2$, then $\sigma_0 j \sigma_2 \sqsubset \mathsf{path}(\xi'')$ implies $\mathsf{path}(\xi'') = \sigma_0 j \sigma_2 \sigma_3$ for some σ_3.

Session types are projections of global types onto participants. They represent the contributions of individual participants to the session. The projection of a choice yields a union for the choice leader and intersections for the receivers. Checkpoint labels of global types are preserved by the projection onto session types, and the checkpoint label of the choice leader is distinguished by overlining it.

We now define *session pre-types*, which are a superset of session types. Session types will be session pre-types which are projections of global types. Session pre-types are obtained from processes by replacing external and internal choices with intersections and unions, X with $\mathbf{t}$ and end with End, with similar conventions.

Definition 9 (*Session pre-types*) *Session pre-types* are defined by:

$$\mathsf{T} \;::=\; {}_{\Delta}\bigwedge_{i\in I} \widetilde{\mathsf{p}_i?\Lambda_i};\mathsf{T}_i \;\mid\; {}_{\overline{\gamma}}\bigvee_{i\in I} \widetilde{\mathsf{p}_i!\Lambda_i};\mathsf{T}_i \;\mid\; \mu\mathbf{t}.\mathsf{T} \;\mid\; \mathbf{t} \;\mid\; \mathsf{End}$$

As for processes, we assume that intersections and unions are not ambiguous, i.e. that the pairs $(\mathsf{p}_i, \Lambda_i)$ $(i \in I)$ are all distinct and the Λ_i's in intersections are either all simple or all connecting messages.

We want projection to ensure that in a global choice, the choice leader makes the decision and all the other participants act accordingly. We do so by requiring that, for any participant except the choice leader, the set of projections of the choice branches on that participant, say $\{\mathsf{T}_i \mid i \in I\}$, be consistent, i.e., the *meet of the types in the set,* $\bigsqcap_{i\in I} \mathsf{T}_i$, be defined.

The meet $\bigsqcap_{i\in I} \mathsf{T}_i$ is a partial operator, which checks that the T_i's are compatible behaviours and then combines them into a single session type. Intuitively, $\bigsqcap_{i\in I} \mathsf{T}_i$ is defined if the concerned participant receives a message that "notifies" her about the chosen T_i. If one of the T_i's is an intersection of simple inputs, then so must be all the other T_i's. Instead, intersections of connecting inputs can be combined with End.

To build the meet of intersection types, we define an auxiliary operator $\Cap$, which takes an intersection $\bigwedge_{i\in I} \widetilde{\mathsf{p}_i ?\Lambda_i}; \mathsf{T}_i$ and an input type (after removing checkpoint labels in both of them) and combines them, if possible:

$$\left(\bigwedge_{i\in I} \widetilde{\mathsf{p}_i ?\Lambda_i}; \mathsf{T}_i\right) \Cap \widetilde{\mathsf{p}?\Lambda}; \mathsf{T} = \bigwedge_{i\in I} \widetilde{\mathsf{p}_i ?\Lambda_i}; \mathsf{T}_i \ \wedge \ \widetilde{\mathsf{p}?\Lambda}; \mathsf{T}$$

if the resulting intersection is not ambiguous

$$\left(\bigwedge_{i\in I} \widetilde{\mathsf{p}_i ?\Lambda_i}; \mathsf{T}_i\right) \Cap \widetilde{\mathsf{p}?\Lambda}; \mathsf{T} = \bigwedge_{i\in I} \widetilde{\mathsf{p}_i ?\Lambda_i}; \mathsf{T}_i$$

if $\mathsf{p} = \mathsf{p}_j$ and $\Lambda = \Lambda_j$ and $\mathsf{T} = \mathsf{T}_j$ for some $j \in I$

$$\left(\bigwedge_{i\in I} \widetilde{\mathsf{p}_i ? \overset{\lambda_i}{\leftrightarrow}}; \mathsf{T}_i\right) \Cap \mathsf{End} = \bigwedge_{i\in I} \widetilde{\mathsf{p}_i ? \overset{\lambda_i}{\leftrightarrow}}; \mathsf{T}_i$$

To define $\Cap$ on two intersection types, we just iterate the above definition on the members of one of the intersections. In a similar way, we can extend $\Cap$ to a set of types. Using $\Cap$ we are able to build meets.

Definition 10 (*Meet*) The *meet of a set of session pre-types* is defined by:

$$\prod_{i\in I} \mathsf{T}_i = \begin{cases} \Delta \Cap_{i\in I} \mathsf{T}'_i & \text{if } \mathsf{T}_i = {}_{\Delta_i}\mathsf{T}'_i \text{ for all } i \in I \text{ and } \Delta = \bigcup_{i\in I} \Delta_i \\ \mathsf{End} & \text{if } \mathsf{T}_i = \mathsf{End} \text{ for all } i \in I \end{cases}$$

If possible we combine the T_i with $\Cap$ as explained above. The set of checkpoint labels of the resulting intersection is the union of the corresponding sets for the T_i's. If some T_i is End, it means that the participant terminates in the branch T_i. Then it must terminate or be connecting in all the other branches.

We do not need to define the meet when the arguments are recursive types, since we consider recursion equi-recursively. The meet is not defined in all other cases, i.e. between intersections and unions, between unions etc.

To define projection we need one last auxiliary operator, the *labelling* of a session pre-type T by a set γ of at most one checkpoint label (notation $\gamma\lfloor\mathsf{T}\rfloor$), defined by:

$$\gamma\lfloor_{\Delta} \bigwedge_{i\in I} \widetilde{\pi_i}; \mathsf{T}_i\rfloor = {}_{\gamma\cup\Delta} \bigwedge_{i\in I} \widetilde{\pi_i}; \mathsf{T}_i \qquad \gamma\lfloor\mathsf{End}\rfloor = \mathsf{End}$$

Labelling adds γ to the first (possibly empty) set Δ found in T. Intuitively, the checkpoint labels that are spread along successive choices in the global type may get grouped together on a single local choice represented by an intersection when projected on participants. Labelling of union and recursive types is not defined since this operator is only used with types resulting from the application of the meet.

The projection of global types uses the projections of rooted interactions, see the first line of Fig. 5, where the projection operator "$\upharpoonright$" has higher precedence than "$;$". The projection of a choice is a union for the sending participant, and it is otherwise computed as the meet of the projections on the branches. By abuse of notation, in Fig. 5 $\overline{\gamma}$ means $\overline{C}$ if $\gamma = C$ and $\emptyset$ if $\gamma = \emptyset$. The meet operation can be undefined, and therefore also the projection of global types can be undefined. The definition of projection does not ensure that all the branches of a

$$(\widetilde{\mathsf{p} \xrightarrow{\Lambda} \mathsf{q}}; \mathsf{G}) \upharpoonright \mathsf{p} = \widetilde{\mathsf{q}!\Lambda}; \mathsf{G} \upharpoonright \mathsf{p} \qquad (\widetilde{\mathsf{p} \xrightarrow{\Lambda} \mathsf{q}}; \mathsf{G}) \upharpoonright \mathsf{q} = \widetilde{\mathsf{p}?\Lambda}; \mathsf{G} \upharpoonright \mathsf{q} \qquad (\widetilde{\mathsf{p} \xrightarrow{\Lambda} \mathsf{q}}; \mathsf{G}) \upharpoonright \mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r}\ ,\ \text{if } \mathsf{r} \neq \mathsf{p}, \mathsf{q}$$

$$({}_{\gamma}\widetilde{\alpha^{\mathsf{p}}}; \mathsf{G}) \upharpoonright \mathsf{r} = \begin{cases} {}_{\overline{\gamma}}(\widetilde{\alpha^{\mathsf{p}}}; \mathsf{G}) \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{p} \\ {}_{\gamma}(\widetilde{\alpha^{\mathsf{p}}}; \mathsf{G}) \upharpoonright \mathsf{r} & \text{otherwise} \end{cases}$$

$$({}_{\gamma} \boxplus_{i \in I} \widetilde{\alpha_i^{\mathsf{p}}}; \mathsf{G}_i) \upharpoonright \mathsf{r} = \begin{cases} {}_{\overline{\gamma}} \bigvee_{i \in I} (\widetilde{\alpha_i^{\mathsf{p}}}; \mathsf{G}_i) \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{p} \\ {}_{\gamma}\lfloor \prod_{i \in I} (\widetilde{\alpha_i^{\mathsf{p}}}; \mathsf{G}_i) \upharpoonright \mathsf{r} \rfloor & \text{otherwise} \end{cases}$$

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

Fig. 5 Projection of global types onto participants

choice ${}_{\gamma} \boxplus_{i \in I} \widetilde{\alpha_i^{\mathsf{p}}}; \mathsf{G}_i$ have the same participants. They can differ for participants whose first communication is a connecting input. Notice that projection respects hats and checkpoint labels (transforming C to $\overline{C}$ for the leader of a C-labelled choice).

Example 5 (*Projection*) Let us see our notion of projection at work on a few examples.

1. If G is the global type shown in Example 4 we get

$$\mathsf{G} \upharpoonright \mathsf{p} = \widehat{\mathsf{q}!\lambda_0}; \mu \mathbf{t}.(\mathsf{r}?\lambda_2; \mathbf{t}\, {}_{C}\!\wedge \mathsf{q}?\lambda_3; \mathsf{q}!\lambda_4) \qquad \mathsf{G} \upharpoonright \mathsf{r} = \mu \mathbf{t}.{}_{C}\mathsf{q}? \overset{\lambda_1}{\leftrightarrow}; \mathsf{p}!\lambda_2; \mathbf{t}$$
$$\mathsf{G} \upharpoonright \mathsf{q} = \widehat{\mathsf{p}?\lambda_0}; \mu \mathbf{t}.(\mathsf{r}! \overset{\lambda_1}{\leftrightarrow}; \mathbf{t}\, {}_{\overline{C}}\!\vee \mathsf{p}!\lambda_3; \mathsf{p}?\lambda_4)$$

2. An example showing how projections can decorate intersections by more checkpoint labels is $\mathsf{G} = \mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; (\mathsf{p} \xrightarrow{\overset{\lambda_2}{\leftrightarrow}} \mathsf{r}_{C_1} \boxplus \mathsf{p} \xrightarrow{\overset{\lambda_3}{\leftrightarrow}} \mathsf{s}) \boxplus \mathsf{p} \xrightarrow{\lambda_4} \mathsf{q}; (\mathsf{q} \xrightarrow{\overset{\lambda_5}{\leftrightarrow}} \mathsf{r}_{C_2} \boxplus \mathsf{q} \xrightarrow{\overset{\lambda_6}{\leftrightarrow}} \mathsf{s})$ where $\mathsf{G} \upharpoonright \mathsf{r} = \mathsf{p}? \overset{\lambda_2}{\leftrightarrow}_{\{C_1, C_2\}}\!\wedge \mathsf{q}? \overset{\lambda_5}{\leftrightarrow}$.
3. Let $\mathsf{G} = \mu \mathbf{t}.(\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; \mathbf{t} \boxplus \mathsf{p} \xrightarrow{\lambda_2} \mathsf{q})$ and $\mathsf{G}' = \mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; \mathsf{G} \boxplus \mathsf{p} \xrightarrow{\lambda_2} \mathsf{q}$. Then $\mathsf{G} \upharpoonright \mathsf{q} = \mu \mathbf{t}.(\mathsf{p}?\lambda_1; \mathbf{t} \wedge \mathsf{p}?\lambda_2)$ and $\mathsf{G}' \upharpoonright \mathsf{q} = \mathsf{p}?\lambda_1; \mathsf{G} \upharpoonright \mathsf{q} \wedge \mathsf{p}?\lambda_2$.

In the following we will consider only *causally correct and projectable global types*. The Subject Reduction Theorem (Theorem 1) shows that these conditions are preserved by reducing networks that can be typed using the rules introduced in the next section.

We end this section by defining session types.

Definition 11 (*Session types*) A session pre-type T is a *session type* if $\mathsf{T} = \mathsf{G} \upharpoonright \mathsf{p}$ for some global type G and some participant p.

4 Type system

In this section we define our type system and prove some initial results about it. The shape of *typing judgements* is $\Gamma \vdash P : \mathsf{T}$, where the environment Γ associates process variables with session types: $\Gamma ::= \emptyset \mid \Gamma, X : \mathsf{T}$. Process typing exploits the correspondence between external choices and intersections, internal choices and unions. The typing rules for both processes and networks are given in Fig. 6. Typing respects hats: in rules [T- ExtCh], and [T- IntCh], the hats in processes and types are exactly on the same actions.

$$\frac{\Gamma \vdash P_i : \mathsf{T}_i \ (i \in I)}{\Gamma \vdash {}_{\Delta}\sum_{i\in I} \widetilde{\pi_i}; P_i : {}_{\Delta}\bigwedge_{i\in I} \widetilde{\pi_i}; \mathsf{T}_i}\ [\text{T-ExtCh}] \qquad \frac{\Gamma \vdash P_i : \mathsf{T}_i \ (i \in I)}{\Gamma \vdash {}_{\overline{\gamma}}\bigoplus_{i\in I} \widetilde{\pi_i}; P_i : {}_{\overline{\gamma}}\bigvee_{i\in I} \widetilde{\pi_i}; \mathsf{T}_i}\ [\text{T-IntCh}]$$

$$\Gamma \vdash \mathsf{end} : \mathsf{End}\ [\text{T-end}] \qquad \frac{\Gamma, X : \mathbf{t} \vdash P : \mathsf{T}}{\Gamma \vdash \mu X.P : \mu\mathbf{t}.\mathsf{T}}\ [\text{T-Rec}] \qquad \Gamma, X : \mathbf{t} \vdash X : \mathbf{t}\ [\text{T-Var}]$$

$$\frac{\vdash P_i : \mathsf{T}_i \quad \mathsf{T}_i \leqslant \mathsf{G}\restriction \mathsf{p}_i \quad (1 \le i \le n) \quad \mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}}{\vdash \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!] : \mathsf{G}}\ [\text{T-Net}]$$

Fig. 6 Typing rules for processes and networks

[Sub-In]
$$\frac{\forall i \in I : \mathsf{T}_i \leqslant \mathsf{T}'_i}{{}_{\Delta}\bigwedge_{i\in I\cup I'} \widetilde{\pi_i}; \mathsf{T}_i \leqslant {}_{\Delta}\bigwedge_{i\in I} \widetilde{\pi_i}; \mathsf{T}'_i}$$

[Sub-Out]
$$\frac{\forall i \in I : \mathsf{T}_i \leqslant \mathsf{T}'_i}{{}_{\overline{\gamma}}\bigvee_{i\in I} \widetilde{\pi_i}; \mathsf{T}_i \leqslant {}_{\overline{\gamma}}\bigvee_{i\in I} \widetilde{\pi_i}; \mathsf{T}'_i}$$

[Sub-In-Skip]
$$\frac{\forall i \in I : \pi_i \text{ is a connecting input}}{{}_{\Delta}\bigwedge_{i\in I} \widetilde{\pi_i}; \mathsf{T}_i \leqslant \mathsf{End}}$$

[Sub- End]
$\mathsf{End} \leqslant \mathsf{End}$

Fig. 7 Subtyping rules

Figure 7 gives the subtyping rules, where the double line indicates that the rules are interpreted *coinductively* [33] (Chapter 21). Subtyping takes into account the rules for intersection and union and preserves hats. Rule [Sub- In- Skip] reflects the fact that connecting inputs can be added without causing problems. Rule [T- Net] is the only rule for typing networks: it requires that the types of all processes be subtypes of the projections of a unique global type. The condition $\mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ ensures the presence of all session participants and allows the typing of sessions containing $\mathsf{p}[\![\, \mathsf{end} \,]\!]$ for any p, a property needed to guarantee invariance of types under structural equivalence of networks. Clearly, typing imposes constraints on the way hats and checkpoint labels are placed within processes.

Example 6 (ok *condition on evaluation contexts*) The following example shows the need of the ok condition in reduction rules to assure subject reduction. Let $P = \mu X.(\mathsf{q}!\lambda_1; \mathsf{r}!\lambda_2; X \ {}_{C}\!\oplus\ \mathsf{q}!\lambda_3; \mathsf{r}!\lambda_4; X)$ and $Q = \mu Y.(\mathsf{p}?\lambda_1; Y \ {}_{C}\!+\ \mathsf{p}?\lambda_3; Y)$ and $R = \mu Z.(\mathsf{p}?\lambda_2; Z \ {}_{C}+\ \mathsf{p}?\lambda_4; Z)$. The network $\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!]$ reduces by forward reductions to $\mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathsf{r}[\![\, R' \,]\!]$ where

$P' = \widehat{\mathsf{q}!\lambda_1}; \widehat{\mathsf{r}!\lambda_2}; (\mathsf{q}!\lambda_1; \mathsf{r}!\lambda_2; P \ {}_{C}\!\oplus\ \widehat{\mathsf{q}!\lambda_3}; \mathsf{r}!\lambda_4; P) \ {}_{C}\!\oplus\ \mathsf{q}!\lambda_3; \mathsf{r}!\lambda_4; P$ and $Q' = \widehat{\mathsf{p}?\lambda_1}; (\mathsf{p}?\lambda_1; Q \ {}_{C}\!+\ \widehat{\mathsf{p}?\lambda_3}; Q)_{C} +\ \mathsf{p}?\lambda_3; Q$ and $R' = \widehat{\mathsf{p}?\lambda_2}; R \ {}_{C}\!+\ \mathsf{p}?\lambda_4; R$. By rule [Back], $\mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathsf{r}[\![\, R' \,]\!] \overset{C}{\curvearrowleft} \mathsf{p}[\![\, P'' \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!]$ where $P'' = \mathsf{q}!\lambda_3; \mathsf{r}!\lambda_4; P$. Without the condition $\mathcal{E}$ ok for $\overline{C}$ on rule [CtBA], we could have also the backward move:

$\mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathsf{r}[\![\, R' \,]\!] \overset{C}{\curvearrowleft} \mathsf{p}[\![\, P''' \,]\!] \parallel \mathsf{q}[\![\, Q'' \,]\!] \parallel \mathsf{r}[\![\, R \,]\!]$, where $P''' = (\widehat{\mathsf{q}!\lambda_1}; \widehat{\mathsf{r}!\lambda_2}; \mathsf{q}!\lambda_1; \mathsf{r}!\lambda_2; P) \ {}_{C}\oplus\ \mathsf{q}!\lambda_3; \mathsf{r}!\lambda_4; P$ and $Q'' = \widehat{\mathsf{p}?\lambda_1}; Q \ {}_{C}\!+\ \mathsf{p}?\lambda_3; Q$. Then Subject Reduction would fail, since the network $\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!]$ is typable with the global type $\mu\mathbf{t}.\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; \mathsf{p} \xrightarrow{\lambda_2} \mathsf{r}; \mathbf{t}\, {}_{C}\!\boxplus \mathsf{p} \xrightarrow{\lambda_3} \mathsf{q}; \mathsf{p} \xrightarrow{\lambda_4} \mathsf{r}; \mathbf{t}$, while $\mathsf{p}[\![\, P''' \,]\!] \parallel \mathsf{q}[\![\, Q'' \,]\!] \parallel \mathsf{r}[\![\, R \,]\!]$ is not typable. In fact the output $\mathsf{r}!\lambda_2$ has a hat in the session type of P''', while the corresponding input $\mathsf{p}?\lambda_2$ does not have a hat in the session type of R. This example shows also why it would not be possible to roll back to the innernost checkpoints in recursive processes, since they could be different for different participants.

In the remainder of this section we prove some properties of our type system which will be used to show the soundness results in the next section. The reader not interested in proofs can go directly to Sect. 5. We start with the classical lemmas of inversion and canonical forms.

Lemma 1 (Inversion Lemma)

1. *If* $\Gamma \vdash {}_{\Delta}\sum_{i\in I} \widetilde{\pi}_i;\, P_i : \mathsf{T}$, *then* $\mathsf{T} = {}_{\Delta}\bigwedge_{i\in I} \widetilde{\pi}_i; \mathsf{T}_i$ *and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
2. *If* $\Gamma \vdash {}_{\overline{\gamma}}\bigoplus_{i\in I} \widetilde{\pi}_i;\, P_i : \mathsf{T}$, *then* $\mathsf{T} = {}_{\overline{\gamma}}\bigvee_{i\in I} \widetilde{\pi}_i; \mathsf{T}_i$ *and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
3. *If* $\Gamma \vdash \mu X.P : \mathsf{T}$, *then* $\mathsf{T} = \mu \boldsymbol{t}.\mathsf{T}'$ *and* $\Gamma, X : \boldsymbol{t} \vdash P : \mathsf{T}'$.
4. *If* $\Gamma \vdash X : \mathsf{T}$, *then* $\mathsf{T} = \boldsymbol{t}$ *and* $\Gamma = \Gamma', X : \boldsymbol{t}$.
5. *If* $\Gamma \vdash \mathsf{end} : \mathsf{T}$, *then* $\mathsf{T} = \mathsf{End}$.
6. *If* $\vdash \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!] : \mathsf{G}$, *then* $\vdash P_i : \mathsf{T}_i$ *and* $\mathsf{T}_i \leqslant \mathsf{G}\upharpoonright \mathsf{p}_i$ *for* $1 \le i \le n$ *and* $\mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$.

Lemma 2 (Canonical Form Lemma)

1. *If* $\Gamma \vdash P : {}_{\Delta}\bigwedge_{i\in I} \widetilde{\pi}_i; \mathsf{T}_i$, *then* $P = {}_{\Delta}\sum_{i\in I} \widetilde{\pi}_i;\, P_i$ *and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
2. *If* $\Gamma \vdash P : {}_{\overline{\gamma}}\bigvee_{i\in I} \widetilde{\pi}_i; \mathsf{T}_i$, *then* $P = {}_{\overline{\gamma}}\bigoplus_{i\in I} \widetilde{\pi}_i;\, P_i$ *and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
3. *If* $\Gamma \vdash P : \mu \boldsymbol{t}.\mathsf{T}$, *then* $P = \mu X.Q$ *and* $\Gamma, X : \boldsymbol{t} \vdash Q : \mathsf{T}$.
4. *If* $\Gamma \vdash P : \boldsymbol{t}$, *then* $P = X$ *and* $\Gamma = \Gamma', X : \boldsymbol{t}$.
5. *If* $\Gamma \vdash P : \mathsf{End}$, *then* $P = \mathsf{end}$.
6. *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathsf{part}(\mathsf{G}) = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$, *then* $N \equiv p_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$ *and* $\vdash P_i : \mathsf{T}_i$ *and* $\mathsf{T}_i \leqslant \mathsf{G}\upharpoonright \mathsf{p}_i$ *for* $1 \le i \le n$.

The mapping $\wr\ \int$ defined at page 7 for processes may be extended in the obvious way to session types. As expected, this mapping preserves typing.

Lemma 3 *If* $\vdash P : \mathsf{T}$, *then* $\vdash \wr P \int : \wr \mathsf{T} \int$.

Another classical lemma we need for proving Subject Reduction is the following, which retrieves the shape of processes and networks from their labelled transitions.

Lemma 4 1. *If* $P \xrightarrow{\mathsf{p}?\Lambda} P'$, *then* $P = \mathcal{E}[{}_{\Delta}\sum_{i\in I} \pi_i;\, P_i]$, *where* $\pi_j = \mathsf{p}?\Lambda$ *for some* $j \in I$, *and* $P' = \mathcal{E}[{}_{\Delta}(\sum_{i\in I\setminus\{j\}} \pi_i;\, P_i \;+\; \widehat{\pi}_j;\, P_j)]$.

2. *If* $P \xrightarrow{\mathsf{p}!\Lambda} P'$, *then* $P = \mathcal{E}[{}_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i;\, P_i]$, *where* $\pi_j = \mathsf{p}!\Lambda$ *for some* $j \in I$, *and* $P' = \mathcal{E}[{}_{\overline{\gamma}}(\bigoplus_{i\in I\setminus\{j\}} \pi_i;\, P_i \;\oplus\; \widehat{\pi}_j;\, P_j)]$.
3. *If* $P \overset{\overline{C}}{\circlearrowright} P'$, *then* $P = \mathcal{E}[{}_{\overline{C}}(\bigoplus_{i\in I} \pi_i;\, Q_i \;\oplus\; \widehat{\pi};\, Q)]$ *and* $Q \downarrow_{out}$ *and* $I \neq \emptyset$ *and* $\mathcal{E}$ ok *for* $\overline{C}$ *and* $P' = \mathcal{E}[{}_{\overline{C}}\bigoplus_{i\in I} \pi_i;\, Q_i]$.
4. *If* $P \overset{C}{\circlearrowright} P'$, *then* $P = \mathcal{E}[{}_{\Delta}\sum_{i\in I} P_i]$ *and* $C \in \Delta$ *and* $\mathcal{E}$ ok *for* C *and* $P' = \mathcal{E}[\wr_{\Delta}\sum_{i\in I} P_i \int]$.
5. *If* $P \overset{C}{\not\circlearrowright}$, *then* P *does not contain an executed external choice with checkpoint label* Δ *and* $C \in \Delta$.
6. *If* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$, *then* $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0$ *and* $P \xrightarrow{\mathsf{q}!\Lambda} P'$ *and* $Q \xrightarrow{\mathsf{p}?\Lambda} Q'$ *and* $\mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathbb{N}_0$.
7. *If* $\mathbb{N} \overset{C}{\circlearrowright} \mathbb{N}'$, *then* $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \Pi_{h\in H}\mathsf{p}_h[\![\, P_h \,]\!] \parallel \Pi_{k\in K}\mathsf{p}_k[\![\, P_k \,]\!]$ *and* $P \overset{\overline{C}}{\circlearrowright} P'$ *and* $P_h \overset{C}{\circlearrowright} P'_h$ *for all* $h \in H$ *and* $P_k \overset{C}{\not\circlearrowright}$ *for all* $k \in K$ *and*
$$\mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \Pi_{h\in H}\mathsf{p}_h[\![\, P'_h \,]\!] \parallel \Pi_{k\in K}\mathsf{p}_k[\![\, P_k \,]\!]$$

We now introduce some notions that are specific to the present type system, and prove some results about them. More precisely:

- we define session contexts (Definition 12) and we show that they correspond to evaluation contexts (Lemma 5);
- by extending the definition of meet to session contexts (Definition 14), we show how session contexts may be retrieved as projections of global contexts (Definition 13);
- we show that the maximal hatted path in the tree of a global type splits the global type into a global context and the global type filling the hole (Definition 15);
- the main result is Lemma 6: it devises the shape of global types starting from one of its projections. It uses the notion of "global type affecting a participant" (Definition 16).

We start by defining session contexts, which mirror process evaluation contexts (Definition 3):

Definition 12 (*Session contexts*) *Session contexts* are defined by:

$$\mathcal{T} ::= {}_{\Delta}(\textstyle\bigwedge_{h\in H} \mathsf{p}_h?\Lambda_h;\mathsf{T}_h \wedge \widehat{\mathsf{p}?\Lambda};\mathcal{T}) \;\;|\;\; {}_{\overline{\gamma}}(\bigvee_{h\in H} \mathsf{p}_h!\Lambda_h;\mathsf{T}_h \vee \widehat{\mathsf{p}!\Lambda};\mathcal{T}) \;\;|\;\; [\;]$$

We generalise typing, subtyping, and the definitions of ok for C ($\overline{C}$) to session contexts. We omit these definitions, which are trivial due to the correspondence between process evaluation contexts and session contexts.

The following lemma gives easy relations between contexts and typing/subtyping.

Lemma 5 1. *If* $\mathcal{T}[\mathsf{T}] \leqslant \mathsf{T}'$ *and* $\mathsf{T}' \neq \mathsf{End}$, *then* $\mathsf{T}' = \mathcal{T}'[\mathsf{T}'']$ *and* $\mathcal{T} \leqslant \mathcal{T}'$ *and* $\mathsf{T} \leqslant \mathsf{T}''$. *Moreover if* $\mathcal{T}$ *is* ok *for* C ($\overline{C}$), *then* $\mathcal{T}'$ *is* ok *for* C ($\overline{C}$).
2. *If* $\mathsf{T} \leqslant \mathcal{T}[\mathsf{T}']$, *then* $\mathsf{T} = \mathcal{T}'[\mathsf{T}'']$ *and* $\mathcal{T}' \leqslant \mathcal{T}$ *and* $\mathsf{T}'' \leqslant \mathsf{T}'$. *Moreover if* $\mathcal{T}$ *is* ok *for* C ($\overline{C}$), *then* $\mathcal{T}'$ *is* ok *for* C ($\overline{C}$).
3. *If* $\Gamma \vdash \mathcal{E}[P] : \mathsf{T}$, *then* $\mathsf{T} = \mathcal{T}[\mathsf{T}']$ *and* $\Gamma \vdash \mathcal{E} : \mathcal{T}$ *and* $\Gamma \vdash P : \mathsf{T}'$. *Moreover if* $\mathcal{E}$ *is* ok *for* C ($\overline{C}$), *then* $\mathcal{T}$ *is* ok *for* C ($\overline{C}$).
4. *If* $\Gamma \vdash \mathcal{E} : \mathcal{T}$ *and* $\Gamma \vdash P : \mathsf{T}$ *and* $\mathcal{T}[\mathsf{T}]$ *is a type, then* $\Gamma \vdash \mathcal{E}[P] : \mathcal{T}[\mathsf{T}]$.
5. *If* $\Gamma \vdash P : \mathcal{T}[\mathsf{T}]$, *then* $P = \mathcal{E}[Q]$ *and* $\Gamma \vdash \mathcal{E} : \mathcal{T}$ *and* $\Gamma \vdash Q : \mathsf{T}$. *Moreover if* $\mathcal{T}$ *is* ok *for* C ($\overline{C}$), *then* $\mathcal{E}$ *is* ok *for* C ($\overline{C}$).

Proof By induction on contexts.

We define now global contexts, which identify subtypes of global types which do not occur in discarded branches.

Definition 13 (*Global contexts*) *Global contexts* are defined by:

$$\mathcal{G} ::= {}_{\gamma}\widetilde{\alpha};\mathcal{G} \;\;|\;\; {}_{\gamma}(\boxplus_{i\in I}\,\alpha_i;\mathsf{K}_i \boxplus \widetilde{\alpha};\mathcal{G}) \;\;|\;\; [\;]$$

We extend to session contexts the labelling defined at page 14 by $\gamma\lfloor[\;]\rfloor = [\;]$. This allows us to define projections of global contexts into session contexts using $[\;]\upharpoonright\mathsf{p} = [\;]$ and generalising the definition of the meet operator as follows:

Definition 14 (*Meet with contexts*) The *meet of session contexts and session types* is defined by:

$${}_{\Delta}(\textstyle\bigwedge_{h\in H}\mathsf{T}_h \wedge \widehat{\pi};\mathcal{T}) \sqcap {}_{\Delta'}\mathsf{T} = {}_{\Delta\cup\Delta'}((\bigwedge_{h\in H}\mathsf{T}_h \Cap \mathsf{T}) \wedge \widehat{\pi};\mathcal{T}) \qquad \text{if the resulting intersection is a session context;}$$

$${}_{\Delta}(\textstyle\bigwedge_{h\in H}\mathsf{T}_h \wedge \widehat{\pi};\mathcal{T}) \sqcap \mathsf{End} = {}_{\Delta}(\bigwedge_{h\in H}\mathsf{T}_h \wedge \widehat{\pi};\mathcal{T}) \qquad \text{if } \mathsf{T}_h \text{ for } h\in H \text{ and } \widehat{\pi} \text{ are all connecting inputs.}$$

Projections of global contexts define session contexts only for a subset of participants. A simple example is $\mathcal{G} = \mathsf{p} \xrightarrow{\lambda} \mathsf{q}; [\ ]$, whose projections on all $\mathsf{r} \neq \mathsf{p}, \mathsf{q}$ is $\mathcal{G} \restriction \mathsf{r} = [\ ]$. However, $\mathcal{G} \restriction \mathsf{p} = \mathsf{q}!\lambda; [\ ]$ and $\mathcal{G} \restriction \mathsf{q} = \mathsf{p}?\lambda; [\ ]$ are not session contexts because the actions before the hole are not hatted. A more interesting example is the ternary choice:

$$\mathcal{G}' = \mathsf{p} \overset{\overset{\lambda_1}{\leftrightarrow}}{\longrightarrow} \mathsf{q} \boxplus \mathsf{p} \overset{\overset{\lambda_2}{\leftrightarrow}}{\longrightarrow} \mathsf{r} \boxplus \widehat{\mathsf{p} \overset{\overset{\lambda_3}{\leftrightarrow}}{\longrightarrow} \mathsf{q}}; [\ ]$$

In fact $\mathcal{G}' \restriction \mathsf{p} = \mathsf{q}! \overset{\lambda_1}{\leftrightarrow} \vee \mathsf{r}! \overset{\lambda_2}{\leftrightarrow} \vee \widehat{\mathsf{q}! \overset{\lambda_3}{\leftrightarrow}}; [\ ]$ and $\mathcal{G}' \restriction \mathsf{q} = \mathsf{p}? \overset{\lambda_1}{\leftrightarrow} \wedge \widehat{\mathsf{p}? \overset{\lambda_3}{\leftrightarrow}}; [\ ]$ are session contexts, while $\mathcal{G}' \restriction \mathsf{r} = \mathsf{p}? \overset{\lambda_2}{\leftrightarrow} \wedge [\ ]$ is not. Notice that without connecting communications such kind of example would be longer, since the participants q and r would have to occur in all branches of the choice.

The paths that are not on discarded branches in the tree representation of a global type G may be used to split G between a global context and the subtype filling the hole. This is formalised in the following definition.

Definition 15 (*Contexts and subtypes determined by paths*) Let σ be a path in a global type G.

1. The *context determined by* σ, $\mathit{Ctx}(\mathsf{G}, \sigma)$, is defined by:
   - $\mathit{Ctx}(\mathsf{G}, \epsilon) = [\ ]$
   - $\mathit{Ctx}({}_\gamma\, \widetilde{\alpha}; \mathsf{G}', 0\sigma) = {}_\gamma\, \widetilde{\alpha}; \mathit{Ctx}(\mathsf{G}', \sigma)$
   - $\mathit{Ctx}({}_\gamma (\boxplus_{j \in J \setminus \{k\}} \alpha_j; \mathsf{K}_j \boxplus \widetilde{\alpha_k}; \mathsf{G}_k), k\sigma) = {}_\gamma (\boxplus_{j \in J \setminus \{k\}} \alpha_j; \mathsf{K}_j \boxplus \widetilde{\alpha_k}; \mathit{Ctx}(\mathsf{G}_k, \sigma))$
   - $\mathit{Ctx}(\mu \mathbf{t}.\mathsf{K}, \sigma) = \mathit{Ctx}(\mathsf{K}\{\mu \mathbf{t}.\mathsf{K}/\mathbf{t}\}, \sigma)$
2. The *subtype determined by* σ, $\mathit{SubT}(\mathsf{G}, \sigma)$, is defined by:
   - $\mathit{SubT}(\mathsf{G}, \epsilon) = \mathsf{G}$
   - $\mathit{SubT}({}_\gamma\, \widetilde{\alpha}; \mathsf{G}', 0\sigma) = \mathit{SubT}(\mathsf{G}', \sigma)$
   - $\mathit{SubT}({}_\gamma (\boxplus_{j \in J \setminus \{k\}} \alpha_j; \mathsf{K}_j \boxplus \widetilde{\alpha_k}; \mathsf{G}_k), k\sigma) = \mathit{SubT}(\mathsf{G}_k, \sigma)$
   - $\mathit{SubT}(\mu \mathbf{t}.\mathsf{K}, \sigma) = \mathit{SubT}(\mathsf{K}\{\mu \mathbf{t}.\mathsf{K}/\mathbf{t}\}, \sigma)$

Note that $\mathit{Ctx}(\mathsf{G}, \sigma)$ and $\mathit{SubT}(\mathsf{G}, \sigma)$ are defined if and only if the occurrences on σ are not in conflict with executed occurrences, in other words if all $\xi \in \mathit{ExOcc}(\mathsf{G})$ are such that $\mathsf{path}(\xi) \sqsubseteq \sigma$ or $\sigma \sqsubset \mathsf{path}(\xi)$.

It is easy to verify that $\mathsf{G} = \mathit{Ctx}(\mathsf{G}, \sigma)[\mathit{SubT}(\mathsf{G}, \sigma)]$ whenever $\mathit{Ctx}(\mathsf{G}, \sigma)$ is defined.

The properties of our calculus mainly depend on the possibility of deriving information on the shape of a global type from its projections on participants. A useful notion is that of global type "affecting" a participant, which essentially means that the whole type is needed in order to obtain the projection on that participant. A global type G affects p if:

- either there is one branch of G whose first communication has participant p
- or there are two branches of G whose projections on p are both different from End.

Definition 16 (*Affecting global types*) A *global type* ${}_\gamma \boxdot_{i \in I}\ \widetilde{\alpha_i}; \mathsf{G}_i$ *affects participant* p if one of the following holds:

- $\mathsf{p} \in \mathsf{part}(\alpha_i)$ for some $i \in I$;
- $|\{i \in I \mid \widetilde{\alpha_i}; \mathsf{G}_i \restriction \mathsf{p} \neq \mathsf{End}\}| > 1$.

For instance, if G is the global type of Example 4, then G affects participants p and q but it does not affect participant r, as shown by their projections given in Example 5(1).

In the projection of a global type G on p, the intersections have sets of checkpoint labels. We know that at most one of these checkpoint labels is the label of the subtype of G which affects p. But which one? The projection does not give us enough information to decide. We overcome this problem by means of an equality relation on session types which disregards the sets of checkpoint labels decorating intersections.

We say that two session types T and T' are *equal up to intersection labelling*, dubbed $\mathsf{T} \doteq \mathsf{T}'$, if they are equal ($\mathsf{T} = \mathsf{T}'$) or they are the same intersection with different sets of checkpoint labels ($\mathsf{T} = {}_{\Delta}\mathsf{T}''$, $\mathsf{T}' = {}_{\Delta'}\mathsf{T}''$ for some T'').

We have now enough machinery to prove that if the projection of G on p is split into a session context and a session type T which is either a union or an intersection of simple inputs, then this is mirrored by a splitting of G into a global context and a subtype which affects p. Moreover, if the session context is a hole, then all the choices along the path leading to the hole in G must be unary choices. Distinguishing the two shapes of T we can also show that:

- if T is a union type, then the subtype of G is a choice with leader p;
- if T is an intersection of simple inputs, then each "long enough" path in G leads to a communication that projects onto one of these inputs.

Lemma 6 1. *If* $\mathsf{G} \upharpoonright \mathsf{p} = \mathcal{T}[{}_{\overline{\gamma}}\bigvee_{i\in I} \mathsf{T}_i]$, *then there is a unique path* σ *in* G *such that* $\mathit{SubT}(\mathsf{G},\sigma) = {}_{\gamma} \boxdot_{i\in I}\ \widetilde{\alpha_i^{\mathsf{p}}}; \mathsf{G}_i$ *affects* p *and*

$$\mathit{Ctx}(\mathsf{G},\sigma)\upharpoonright \mathsf{p} = \mathcal{T} \textit{and } \mathit{SubT}(\mathsf{G},\sigma)\upharpoonright \mathsf{p} = {}_{\overline{\gamma}}\bigvee_{i\in I} \mathsf{T}_i.$$

2. *If* $\mathsf{G}\upharpoonright \mathsf{p} = \mathcal{T}[{}_{\Delta} \bigwedge_{i\in I} \widetilde{\mathsf{q}_i ?\lambda_i}; \mathsf{T}_i]$, *then there is a unique path* σ *in* G *such that* $\mathit{SubT}(\mathsf{G},\sigma)$ *affects* p *and*

$$\mathit{Ctx}(\mathsf{G},\sigma)\upharpoonright \mathsf{p} = \mathcal{T} \textit{and } \mathit{SubT}(\mathsf{G},\sigma)\upharpoonright \mathsf{p} \doteq {}_{\Delta}\bigwedge_{i\in I} \widetilde{\mathsf{q}_i ?\lambda_i}; \mathsf{T}_i.$$

Moreover for each path σ' *in* $\mathit{SubT}(\mathsf{G},\sigma)$ *there are* $j \in I$ *and* σ_j *such that*

$$\widetilde{\sigma\sigma_j \mathsf{q}_j \xrightarrow{\lambda_j} \mathsf{p}} \in \mathit{Occ}(\mathsf{G}) \textit{ and either } \sigma_j \sqsubseteq \sigma' \textit{ or } \sigma' \sqsubset \sigma_j.$$

In both cases if $\mathcal{T}$ *is the hole, then* σ *is a possibly empty string of* 0*'s.*

Proof (1) By induction on session type contexts. If $\mathcal{T} = [\ ]$, then $\mathsf{G}\upharpoonright \mathsf{p} = {}_{\overline{\gamma}}\bigvee_{i\in I} \mathsf{T}_i$ and by definition of projection $\mathsf{G} = {}_{\gamma} \boxdot_{i\in I}\ \widetilde{\alpha_i^{\mathsf{p}}}; \mathsf{G}_i$, so G affects p. In this case $\sigma = \epsilon$ is the required path, since $\mathit{SubT}(\mathsf{G},\epsilon) = \mathsf{G}$ affects p and $\mathit{Ctx}(\mathsf{G},\epsilon)\upharpoonright \mathsf{p} = [\ ]\upharpoonright \mathsf{p} = [\ ]$ and $\mathit{SubT}(\mathsf{G},\epsilon)\upharpoonright \mathsf{p} = \mathsf{G}\upharpoonright \mathsf{p} = {}_{\overline{\gamma}}\bigvee_{i\in I} \mathsf{T}_i$.

Moreover, σ is unique because for any path $\sigma' \neq \epsilon$ the first communication along σ' would be an output from p, which would imply $\mathit{Ctx}(\mathsf{G},\sigma')\upharpoonright \mathsf{p} \neq [\ ]$ whenever $\mathit{Ctx}(\mathsf{G},\sigma')\upharpoonright \mathsf{p}$ is defined.

If $\mathcal{T} = {}_{\Delta}(\bigwedge_{i\in I} \mathsf{T}_i \wedge \widehat{\mathsf{q}?\Lambda}; \mathcal{T}')$, then $\widehat{\mathsf{q} \xrightarrow{\Lambda} \mathsf{p}}$ occurs in G by definition of projection. This means that there exists $\xi \in \mathit{Occ}(\mathsf{G})$ such that $\xi = \sigma' \widehat{\mathsf{q} \xrightarrow{\Lambda} \mathsf{p}}$ and there does not exist $\xi' \in \mathit{Occ}(\mathsf{G})$ such that $\mathsf{path}(\xi') \sqsubset \sigma'$ and $\mathsf{p} \in \mathsf{part}(\xi')$. The definition of meet implies that there is no $j \in I$ such that T_j starts by $\mathsf{q}?\Lambda$. Let $\mathsf{G}' = \mathit{SubT}(\mathsf{G},\sigma')$. By construction $\mathsf{G}'\upharpoonright \mathsf{p} = \mathcal{T}'[{}_{\overline{\gamma}}\bigvee_{i\in I} \mathsf{T}_i]$. By induction hypothesis on $\mathcal{T}'$ there is a unique path σ'' such that $\mathit{SubT}(\mathsf{G}',\sigma'')$ affects p and $\mathit{Ctx}(\mathsf{G}',\sigma'')\upharpoonright \mathsf{p} = \mathcal{T}'$ and $\mathit{SubT}(\mathsf{G}',\sigma'')\upharpoonright \mathsf{p} = {}_{\overline{\gamma}}\bigvee_{i\in I} \mathsf{T}_i$. We can then choose $\sigma = \sigma'\sigma''$. The proof of the other inductive case is similar.

(2) By induction on session type contexts. Let $\mathsf{T} =_{\Delta} \bigwedge_{i\in I} \widetilde{\mathsf{q}_i?\lambda_i}; \mathsf{T}_i$. If $\mathcal{T} = [\ ]$ the proof proceeds by an inner induction on the path $\mathsf{ap}(\mathsf{p},\mathsf{G})$ defined by:

$$\mathsf{ap}(\mathsf{p},\mathsf{G}) = \begin{cases} 0\,\mathsf{ap}(\mathsf{p},\mathsf{G}') & \text{if } \mathsf{G} = \widetilde{\alpha}; \mathsf{G}' \text{ and } \mathsf{p} \notin \mathsf{part}(\alpha) \\ \epsilon & \text{otherwise} \end{cases}$$

Intuitively, $\mathsf{ap}(\mathsf{p},\mathsf{G})$ represents the access path to the first choice affecting p in G. Let $\sigma = \mathsf{ap}(\mathsf{p},\mathsf{G})$. If $\sigma = \epsilon$, then $\mathsf{G} =_{\gamma} \boxdot_{h\in H}\ \widetilde{\alpha_h}; \mathsf{G}_h$ must satisfy either $|H| > 1$ or $\mathsf{p} \in \mathsf{part}(\alpha_j)$ for the unique $j \in H$. In the latter case, G trivially affects p. If $|H| > 1$, then by definition of projection and the fact that all inputs are simple, there are two branches whose projections on p are different from End. Therefore G affects p. In both cases we have $\mathit{Ctx}(\mathsf{G},\epsilon)\restriction\mathsf{p} = [\ ]\restriction\mathsf{p} = [\ ]$ and $\mathit{SubT}(\mathsf{G},\epsilon)\restriction\mathsf{p} = \mathsf{G}\restriction\mathsf{p} = \mathsf{T}$.

We show that σ is unique. Let $\sigma' = j\sigma''$ for some $j \in H$ and suppose that $\mathit{Ctx}(\mathsf{G},\sigma')\restriction\mathsf{p}$ is defined. If $\mathsf{p} \in \mathsf{part}(\alpha_j)$, then clearly $\mathit{Ctx}(\mathsf{G},\sigma')\restriction\mathsf{p} \neq [\ ]$. If $|H| > 1$, then $\mathit{Ctx}(\mathsf{G},\sigma')\restriction\mathsf{p}$ contains the projection on p of all the paths of G starting with a branch different from the jth one. Since p appears as receiver along all these paths as shown below, this again implies $\mathit{Ctx}(\mathsf{G},\sigma')\restriction\mathsf{p} \neq [\ ]$.

If $\sigma = 0\,\sigma'$ where $\sigma' = \mathsf{ap}(\mathsf{p},\mathsf{G}')$ and $\mathsf{G} = \widetilde{\alpha}; \mathsf{G}'$, then by inductive hypothesis on σ' we have that $\mathit{SubT}(\mathsf{G}',\sigma')$ affects p and $\mathit{Ctx}(\mathsf{G}',\sigma')\restriction\mathsf{p} = [\ ]$ and $\mathit{SubT}(\mathsf{G}',\sigma')\restriction\mathsf{p} = \mathsf{T}$. Therefore, $\mathit{SubT}(\mathsf{G},0\sigma')$ affects p and $\mathit{Ctx}(\mathsf{G},0\sigma')\restriction\mathsf{p} = [\ ]$ and $\mathit{SubT}(\mathsf{G},0\sigma')\restriction\mathsf{p} = \mathsf{T}$. By induction hypothesis σ' is unique and so is σ.
The proof for $\mathcal{T} \neq [\ ]$ is as in (1).
In the remaining we prove that for each path σ' in $\mathit{SubT}(\mathsf{G},\sigma)$ there are σ'' and $j \in I$ such that $\sigma\sigma''\widetilde{\mathsf{q}_j \xrightarrow{\lambda_j} \mathsf{p}} \in \mathit{Occ}(\mathsf{G})$ and either $\sigma'' \sqsubseteq \sigma'$ or $\sigma' \sqsubset \sigma''$. The path σ' crosses different choices. By the assumption that all inputs are simple and by the definition of $\Cap$, the projections of the branches of these choices on p must be intersections of types starting with inputs with different senders or messages, namely $\widetilde{\mathsf{q}_j?\lambda_j}$ $(j \in I' \subseteq I)$. Moreover, the corresponding communications $\widetilde{\mathsf{q}_j \xrightarrow{\lambda_j} \mathsf{p}}$ $(j \in I')$ must be the first communications involving p in paths of $\mathit{SubT}(\mathsf{G},\sigma)$. Therefore, given σ' in $\mathit{SubT}(\mathsf{G},\sigma)$ there are σ'' and α such that $\xi = \sigma\sigma''\widetilde{\alpha} \in \mathit{Occ}(\mathsf{G})$ and ξ is the first occurrence with $\mathsf{p} \in \mathsf{part}(\xi)$ and either $\sigma'' \sqsubseteq \sigma'$ or $\sigma' \sqsubset \sigma''$. The definition of projection implies $\widetilde{\alpha} = \widetilde{\mathsf{q}_j \xrightarrow{\lambda_j} \mathsf{p}}$ for some $j \in I'$. □

5 Soundness

This section is devoted to the formulation and the proof of the properties of our calculus, i.e. subject reduction, session fidelity and progress. These proofs are not trivial due to the presence of both reverse computations and connecting communications.

Lemma 7 deals with subtypes of global contexts. It uses the notions of "alive" and "enabled" for communication occurrences. An occurrence of a communication is alive if it can be executed in the future, and it is enabled if it can be immediately executed. We also define the notions of "alive" and "enabled" for checkpoint labels as they are needed for session fidelity. A checkpoint label is alive if it can be the target of a rollback in the future, and it is enabled if the next action of the choice leader is an output, possibly triggering a backward reduction of the network.

Definition 17 (*Alive and enabled*)

1. *An occurrence ξ is alive in* G if $\xi \in Occ(\mathsf{G}) \backslash ExOcc(\mathsf{G})$ and there is no $\xi' \in ExOcc(\mathsf{G})$ such that $\xi' \#_{\mathsf{G}}\, \xi$.
2. *An occurrence ξ is enabled in* G if it is alive in G and $Causes(\mathsf{G}, \xi) \subseteq ExOcc(\mathsf{G})$.
3. *A path σ is* ok *for the checkpoint label C in* G if C does not occur along the path σ in the tree of G.
4. *A checkpoint label C is alive in* G if for some path σ ok for C (in G)
$$SubT(\mathsf{G}, \sigma) = {}_{C}(\boxplus_{j \in J \backslash \{k\}}\, \alpha_j^{\mathsf{p}}; \mathsf{K}_j \boxplus \widehat{\alpha_k^{\mathsf{p}}}; \mathsf{G}_k)$$
and $|J| > 1$ and there are σ' and α^{p} such that $\sigma k \sigma' \alpha^{\mathsf{p}} \in Occ(\mathsf{G})$ is alive in G.
5. *A checkpoint label C is enabled in* G if it is alive in G and $\sigma k \sigma' \alpha^{\mathsf{p}}$ is enabled in G, where σ, k, σ' and α^{p} are as in (3).

For instance in the global type of Example 4 the occurrence $01\alpha_1$ is enabled and the occurrence $0101\alpha_1$ is alive but not enabled. The checkpoint label C is enabled for the path ϵ in the global type $\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}\, {}_{C}\boxplus \widehat{\mathsf{p} \xrightarrow{\lambda_2} \mathsf{q}}; \mathsf{p} \overset{\overset{\lambda_3}{\leftrightarrow}}{\longrightarrow} \mathsf{r}$, since the occurrence $20\mathsf{p} \overset{\overset{\lambda_3}{\leftrightarrow}}{\longrightarrow} \mathsf{r}$ is enabled in G. The checkpoint label C is alive but not enabled for the path ϵ in the global type $\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}\, {}_{C}\boxplus \widehat{\mathsf{p} \xrightarrow{\lambda_2} \mathsf{q}}; \mathsf{q} \xrightarrow{\lambda_3} \mathsf{p}; \mathsf{p} \overset{\overset{\lambda_4}{\leftrightarrow}}{\longrightarrow} \mathsf{r}$, since the occurrence $200\mathsf{p} \overset{\overset{\lambda_4}{\leftrightarrow}}{\longrightarrow} \mathsf{r}$ is alive, but not enabled in G.

Lemma 7 *1. If $\sigma j \alpha$ is enabled in* G *and* $\mathsf{p} \in \mathsf{part}(\alpha)$, *then* $Ctx(\mathsf{G}, \sigma) \restriction \mathsf{p}$ *is a session context.*
2. *If $SubT(\mathsf{G}, \sigma) = {}_{\gamma}\, \boxdot_{i \in I}\ \alpha_i; \mathsf{G}_i$, then $\sigma i \alpha_i$ is alive in* G *for all $i \in I$.*
3. *If $SubT(\mathsf{G}, \sigma) = {}_{\gamma}(\boxdot_{i \in I \backslash \{j\}}\, \alpha_i^{\mathsf{p}}; \mathsf{K}_i \boxdot \alpha_j^{\mathsf{p}}; \mathsf{G}_j)$ and* $\mathsf{recv}(\alpha_j^{\mathsf{p}}) = \mathsf{q}$ *and both $Ctx(\mathsf{G}, \sigma) \restriction \mathsf{p}$ and $Ctx(\mathsf{G}, \sigma) \restriction \mathsf{q}$ are session contexts, then $\sigma j \alpha_j^{\mathsf{p}}$ is enabled in* G.

Proof (1) If $\sigma j \alpha$ is enabled in G, then $Causes(\mathsf{G}, \sigma j \alpha) \subseteq ExOcc(\mathsf{G})$ and $\xi \in ExOcc(\mathsf{G})$ implies $\mathsf{path}(\xi) \sqsubseteq \sigma$. Hence $Ctx(\mathsf{G}, \sigma)$ is defined. If $\mathsf{p} \in \mathsf{part}(\alpha)$, define $Causes_{\mathsf{p}}(\mathsf{G}, \sigma j \alpha) = Causes(\mathsf{G}, \sigma j \alpha) \cap \{\xi \in Occ(\mathsf{G}) \mid \mathsf{p} \in \mathsf{part}(\xi)\}$. Since the projection of $Ctx(\mathsf{G}, \sigma)$ on p retains only $Causes_{\mathsf{p}}(\mathsf{G}, \sigma j \alpha)$ and forgets everything else on the path σ, and moreover it preserves and reflects hats, $Ctx(\mathsf{G}, \sigma) \restriction \mathsf{p}$ is a session context.

(2) From $SubT(\mathsf{G}, \sigma) = {}_{\gamma}\, \boxdot_{i \in I}\ \alpha_i; \mathsf{G}_i$ it follows that $\sigma i \alpha_i \in Occ(\mathsf{G}) \backslash ExOcc(\mathsf{G})$. Moreover, since $SubT(\mathsf{G}, \sigma)$ is defined, the path σ must follow the executed branches of G, if any, thus $\xi \in Causes(\mathsf{G}, \sigma i \alpha_i)$ implies $\mathsf{path}(\xi) \sqsubseteq \sigma$. Hence $\sigma i \alpha_i$ is alive in G.

(3) By (2) $\sigma j \alpha_j^{\mathsf{p}}$ is alive in G. We prove that $Causes(\mathsf{G}, \sigma j \alpha_j^{\mathsf{p}}) \subseteq ExOcc(\mathsf{G})$ by induction on the definition of $Causes$. If $\xi \in Causes(\mathsf{G}, \sigma j \alpha_j^{\mathsf{p}})$ because $\mathsf{part}(\xi) \cap \{\mathsf{p}, \mathsf{q}\} \neq \emptyset$, then it must be $\xi \in ExOcc(\mathsf{G})$, otherwise the projection of the communication of ξ would appear unhatted in $Ctx(\mathsf{G}, \sigma) \restriction \mathsf{p}$ or in $Ctx(\mathsf{G}, \sigma) \restriction \mathsf{q}$. Suppose now that $\xi \in Causes(\mathsf{G}, \sigma j \alpha_j^{\mathsf{p}})$ because $\xi \in Causes(\mathsf{G}, \xi')$ and $\xi' \in Causes(\mathsf{G}, \sigma j \alpha_j^{\mathsf{p}})$. By induction hypothesis $\xi' \in ExOcc(\mathsf{G})$, hence by causal correctness of G also $\xi \in ExOcc(\mathsf{G})$. □

Theorem 1 (Subject reduction) *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathbb{N} \overset{\curvearrowleft\,*}{\longrightarrow} \mathbb{N}'$, *then* $\vdash \mathbb{N}' : \mathsf{G}'$ *for some* G'.

Proof By case analysis on the reduction rules for networks. Since the case of structural equivalence is trivial, there are only two rules to consider, Rule [COM] and Rule [BACK]. Let the applied rule be [COM], then

$$\frac{P \xrightarrow{\mathsf{q}!\Lambda} P' \qquad Q \xrightarrow{\mathsf{p}?\Lambda} Q'}{\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathbb{N}}$$

By Lemma 4(2) $P = \mathcal{E}[{}_{\overline{\gamma}}\bigoplus_{i\in I}\pi_i; P_i]$, where $\pi_j = \mathsf{p}!\Lambda$ for some $j \in I$, and $P' = \mathcal{E}[{}_{\overline{\gamma}}(\bigoplus_{i\in I\setminus\{j\}}\pi_i; P_i \oplus \widehat{\pi_j}; P_j)]$. By Lemma 4(1) $Q = \mathcal{E}'[{}_{\Delta}\sum_{h\in H}\pi'_h; Q_h]$, where $\pi'_k = \mathsf{p}?\Lambda$ for some $k \in H$, and $Q' = \mathcal{E}'[{}_{\Delta}(\sum_{h\in H\setminus\{k\}}\pi'_h; Q_h + \widehat{\pi'_k}; Q_k)]$.

By Lemma 1(6) $\vdash P : \mathsf{T}$ and $\vdash Q : \mathsf{S}$ and $\mathsf{T} \leqslant \mathsf{G}\restriction\mathsf{p}$ and $\mathsf{S} \leqslant \mathsf{G}\restriction\mathsf{q}$. By Lemma 5(3) $\mathsf{T} = \mathcal{T}[\mathsf{T}']$ and $\vdash \mathcal{E} : \mathcal{T}$ and $\vdash {}_{\overline{\gamma}}\bigoplus_{i\in I}\pi_i; P_i : \mathsf{T}'$ and $\mathsf{S} = \mathcal{T}'[\mathsf{S}']$ and $\vdash \mathcal{E}' : \mathcal{T}'$ and $\vdash {}_{\Delta}\sum_{h\in H}\pi'_h; Q_h : \mathsf{S}'$. By Lemma 1(2) and (1) $\mathsf{T}' = {}_{\overline{\gamma}}\bigvee_{i\in I}\pi_i; \mathsf{T}_i$ and $\mathsf{S}' = {}_{\Delta}\bigwedge_{h\in H}\pi'_h; \mathsf{S}_h$.

By Lemma 5(1) $\mathsf{G}\restriction\mathsf{p} = \mathcal{T}_0[\mathsf{T}'']$ and $\mathcal{T} \leqslant \mathcal{T}_0$ and ${}_{\overline{\gamma}}\bigvee_{i\in I}\pi_i; \mathsf{T}_i \leqslant \mathsf{T}''$ and $\mathsf{G}\restriction\mathsf{q} = \mathcal{T}'_0[\mathsf{S}'']$ and $\mathcal{T}' \leqslant \mathcal{T}'_0$ and ${}_{\Delta}\bigwedge_{h\in H}\pi'_h; \mathsf{S}_h \leqslant \mathsf{S}''$. By definition of $\leqslant$ we get $\mathsf{T}'' = {}_{\overline{\gamma}}\bigvee_{i\in I}\pi_i; \mathsf{T}'_i$ with $\mathsf{T}_i \leqslant \mathsf{T}'_i$ for $i \in I$ and $\mathsf{S}'' = {}_{\Delta}\bigwedge_{h\in H'}\pi'_h; \mathsf{S}'_h$ with $H \supseteq H'$ and $\mathsf{S}_h \leqslant \mathsf{S}'_h$ for $h \in H'$.

From $\mathsf{G}\restriction\mathsf{p} = \mathcal{T}_0[{}_{\overline{\gamma}}\bigvee_{i\in I}\pi_i; \mathsf{T}'_i]$ it follows by Lemma 6(1) that there is a unique path σ of G such that $\mathit{SubT}(\mathsf{G},\sigma) = {}_{\gamma}\boxdot_{i\in I}\ \alpha^{\mathsf{p}}_i; \mathsf{G}_i$ affects p and $\mathit{Ctx}(\mathsf{G},\sigma)\restriction\mathsf{p} = \mathcal{T}_0$ and $\mathit{SubT}(\mathsf{G},\sigma)\restriction\mathsf{p} = {}_{\overline{\gamma}}\bigvee_{i\in I}\pi_i; \mathsf{T}'_i$. Since $\pi_j = \mathsf{q}!\Lambda$, we have $\alpha^{\mathsf{p}}_j = \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$. Consider now the projection $\mathit{SubT}(\mathsf{G},\sigma)\restriction\mathsf{q} = ({}_{\gamma}\boxdot_{i\in I}\ \alpha^{\mathsf{p}}_i; \mathsf{G}_i)\restriction\mathsf{q}$. It must be $\mathit{SubT}(\mathsf{G},\sigma)\restriction\mathsf{q} = \gamma\lfloor\bigsqcap_{i\in I}(\ \alpha^{\mathsf{p}}_i; \mathsf{G}_i\ )\restriction\mathsf{q}\rfloor \doteq {}_{\Delta}(\bigwedge_{h\in H'\setminus\{k\}}\pi'_h; \mathsf{S}'_h \wedge \pi'_k; \mathsf{S}'_k) = \mathsf{S}''$, where $\gamma \in \Delta$ and $\pi'_k = \mathsf{p}?\Lambda$.

By definition we have $\mathsf{G} = \mathit{Ctx}(\mathsf{G},\sigma)[\mathit{SubT}(\mathsf{G},\sigma)]$. Then we can choose $\mathsf{G}' = \mathit{Ctx}(\mathsf{G},\sigma)[{}_{\gamma}(\boxdot_{i\in I\setminus\{j\}}\alpha^{\mathsf{p}}_i; \mathsf{K}_i \boxdot \widehat{\alpha^{\mathsf{p}}_j}; \mathsf{K}_j)]$. Indeed $\mathsf{G}'\restriction\mathsf{p} = \mathcal{T}_0[\mathsf{T}_0]$ and $\mathsf{G}'\restriction\mathsf{q} = \mathcal{T}'_0[\mathsf{S}_0]$, where $\mathsf{T}_0 = {}_{\overline{\gamma}}(\bigvee_{i\in I\setminus\{j\}}\pi_i; \mathsf{T}_i \vee \widehat{\pi_j}; \mathsf{T}_j)$ and $\mathsf{S}_0 = {}_{\Delta}(\bigwedge_{h\in H'\setminus\{k\}}\pi_h; \mathsf{S}_h \wedge \widehat{\pi'_k}; \mathsf{S}'_k)$, while $\mathsf{G}'\restriction\mathsf{r} = \mathsf{G}\restriction\mathsf{r}$ for $\mathsf{r} \neq \mathsf{p},\mathsf{q}$. By Lemma 5(4) $\vdash P' : \mathcal{T}[\mathsf{T}_0]$ and $\vdash Q' : \mathcal{T}'[\mathsf{S}_0]$. Since $\mathcal{T}[\mathsf{T}_0] \leqslant \mathsf{G}'\restriction\mathsf{p}$ and $\mathcal{T}'[\mathsf{S}_0] \leqslant \mathsf{G}'\restriction\mathsf{q}$ we can derive $\vdash \mathsf{p}[\![\ P'\ ]\!] \parallel \mathsf{q}[\![\ Q'\ ]\!] \parallel \mathbb{N} : \mathsf{G}'$. By Lemma 7(3) $\sigma\, j\alpha^{\mathsf{p}}_j$ is enabled in G, being both $\mathit{Ctx}(\mathsf{G},\sigma)\restriction\mathsf{p}$ and $\mathit{Ctx}(\mathsf{G},\sigma)\restriction\mathsf{q}$ session contexts. This implies the causal correctness of G'. The projectability of G' is immediate.

Let now the applied reduction rule be [Back], then:

$$\frac{P \overset{\overline{C}}{\circlearrowright} P' \qquad P_h \overset{C}{\circlearrowright} P'_h \quad h \in H \qquad P_k \overset{C}{\not\circlearrowright} \quad k \in K}{\mathsf{p}[\![\ P\ ]\!] \parallel \Pi_{h\in H}\mathsf{p}_h[\![\ P_h\ ]\!] \parallel \Pi_{k\in K}\mathsf{p}_k[\![\ P_k\ ]\!] \overset{C}{\circlearrowright} \mathsf{p}[\![\ P'\ ]\!] \parallel \Pi_{h\in H}\mathsf{p}_h[\![\ P'_h\ ]\!] \parallel \Pi_{k\in K}\mathsf{p}_k[\![\ P_k\ ]\!]}$$

By Lemma 4(3) $P = \mathcal{E}[{}_{\overline{C}}(\bigoplus_{i\in I}\pi_i; Q_i \oplus \widehat{\pi}; Q)]$ and $Q \downarrow_{out}$ and $I \neq \emptyset$ and $\mathcal{E}$ ok for $\overline{C}$ and $P' = \mathcal{E}[{}_{\overline{C}}\bigoplus_{i\in I}\pi_i; Q_i]$. By Lemma 4(4) $P_h = \mathcal{E}_h[S_h]$, and $P'_h = \mathcal{E}_h[\wr S_h \int]$, where $S_h = {}_{\Delta_h}\sum_{\ell\in L_h} R_\ell$ and $C \in \Delta_h$ and $\mathcal{E}_h$ ok for C with $h \in H$. By Lemma 4(5), in a P_k with $k \in K$ there cannot be executed external choices whose checkpoint labels contain C.

By Lemma 1(6) $\vdash P : \mathsf{T}$ for some $\mathsf{T} \leqslant \mathsf{G}\restriction\mathsf{p}$, and $\vdash P_j : \mathsf{T}_j$ for some $\mathsf{T}_j \leqslant \mathsf{G}\restriction\mathsf{p}_j$ for each $j \in H \cup K$. By Lemma 1(1), in a T_k with $k \in K$ there cannot be executed intersections whose checkpoint labels contain C.

By Lemma 5(3) $\mathsf{T} = \mathcal{T}[\mathsf{T}']$, where $\vdash \mathcal{E} : \mathcal{T}$ and $\mathcal{T}$ ok for $\overline{C}$ and

$$\vdash {}_{\overline{C}}(\bigoplus_{i\in I}\pi_i; Q_i \oplus \widehat{\pi}; Q) : \mathsf{T}'$$

and $\mathsf{T}_h = \mathcal{T}_h[\mathsf{S}_h]$, where $\vdash \mathcal{E}_h : \mathcal{T}_h$ and $\mathcal{T}_h$ ok for C and $\vdash S_h : \mathsf{S}_h$ for each $h \in H$. By Lemma 1(2) $\mathsf{T}' = {}_{\overline{C}}(\bigvee_{i\in I}\pi_i; \mathsf{T}_i \vee \widehat{\pi}; \mathsf{T}_Q)$ and by Lemma 1(1) S_h is an intersection with Δ_h as set of checkpoint labels ($h \in H$).

By Lemma 5(1) $\mathsf{G}\restriction\mathsf{p} = \mathcal{T}'[\mathsf{T}'']$ and $\mathcal{T} \leqslant \mathcal{T}'$ and $\mathcal{T}'$ ok for $\overline{C}$ and

$${}_{\overline{C}}\left(\bigvee_{i\in I}\pi_i; \mathsf{T}_i \vee \widehat{\pi}; \mathsf{T}_Q\right) \leqslant \mathsf{T}''$$

and $\mathsf{G}\restriction\mathsf{p}_h = \mathcal{T}'_h[\mathsf{S}'_h]$ and $\mathcal{T}_h \leqslant \mathcal{T}'_h$ and $\mathcal{T}'_h$ ok for C and $\mathsf{S}_h \leqslant \mathsf{S}'_h$ for $h \in H$. By definition of $\leqslant$ we get $\mathsf{T}'' = {}_{\overline{C}}(\bigvee_{i\in I}\pi_i; \mathsf{T}'_i \vee \widehat{\pi}; \mathsf{T}'_Q)$ with $\mathsf{T}_i \leqslant \mathsf{T}'_i$ for $i \in I$ and $\mathsf{T}_Q \leqslant \mathsf{T}'_Q$ and S'_h is an intersection with Δ_h as set of checkpoint labels ($h \in H$).

By Lemma 6(1), from $\mathsf{G} \restriction \mathsf{p} = \mathcal{T}'[_{\overline{C}}(\bigvee_{i\in I} \pi_i; \mathsf{T}'_i \vee \widehat{\pi}; \mathsf{T}'_Q)]$ it follows that there is a unique path σ of G such that $\mathit{SubT}(\mathsf{G}, \sigma) = {}_C(\boxplus_{i\in I} \alpha_i^{\mathsf{p}}; \mathsf{K}_i \boxplus \widehat{\alpha^{\mathsf{p}}}; \mathsf{G}_Q)$ affects p and $\mathit{Ctx}(\mathsf{G}, \sigma) \restriction \mathsf{p} = \mathcal{T}'$ and $\mathit{SubT}(\mathsf{G}, \sigma) \restriction \mathsf{p} = {}_{\overline{C}}(\bigvee_{i\in I} \pi_i; \mathsf{T}'_i \vee \widehat{\pi}; \mathsf{T}'_Q)$. The conditions $\mathcal{T}'$ ok for $\overline{C}$ and $\mathcal{T}_h$ ok for C $(h \in H)$ and T_k without executed intersections whose checkpoint labels contain C $(k \in K)$ assure that the path σ is ok for C, i.e. that $\mathit{Ctx}(\mathsf{G}, \sigma)$ does not contain occurrences of C. Therefore the occurrence of C in $\mathit{SubT}(\mathsf{G}, \sigma)$ is the outermost one and it is projected on the checkpoint labels of T'' and S_h for $h \in H$. Then we can choose $\mathsf{G}' = \mathit{Ctx}(\mathsf{G}, \sigma)[_C(\boxplus_{i\in I} \alpha_i^{\mathsf{p}}; \mathsf{K}_i)]$. In fact $\mathsf{G}' \restriction \mathsf{p} = \mathcal{T}'[_{\overline{C}} \bigvee_{i\in I} \pi_i; \mathsf{T}'_i]$ and $\mathsf{G}' \restriction \mathsf{p}_h = \mathcal{T}'_h[\wr \mathsf{S}'_h \int]$ for $h \in H$ and $\mathsf{G}' \restriction \mathsf{p}_k = \mathsf{G} \restriction \mathsf{p}_k$ for $k \in K$. We can derive $\vdash P' : \mathcal{T}[_{\overline{C}} \bigvee_{i\in I} \pi_i; \mathsf{T}_i]$ by Lemma 5(4) and $\vdash P'_h : \mathcal{T}_h[\wr \mathsf{S}_h \int]$ for $h \in H$ by Lemma 3. Since $\mathcal{T}[_{\overline{C}} \bigvee_{i\in I} \pi_i; \mathsf{T}_i] \leqslant \mathsf{G}' \restriction \mathsf{p}$ and $\mathcal{T}_h[\wr \mathsf{S}_h \int] \leqslant \mathsf{G}' \restriction \mathsf{p}_h$ for $h \in H$ we may conclude that $\vdash \mathsf{p}[\![P']\!] \parallel \Pi_{h\in H} \mathsf{p}_h [\![P'_h]\!] \parallel \Pi_{k\in K} \mathsf{p}_k [\![P_k]\!] : \mathsf{G}'$. The causal correctness of G implies the causal correctness of G' since G' is obtained from G by erasing one branch in a choice. The projectability of G' is immediate. □

A standard property enforced by session types is session fidelity: all communications occur as specified by global types. In our case this applies also to backward reductions. The proof relies on the proof of subject reduction.

Theorem 2 (Session fidelity) *Let* $\vdash \mathbb{N} : \mathsf{G}$.

1. *If* $\sigma \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$ *is enabled in* G, *then* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$.
2. *If* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$, *then* $\sigma \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$ *is enabled in* G *for some* σ.
3. *If* C *is enabled in* G, *then* $\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}'$.
4. *If* $\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}'$, *then* C *is enabled in* G.

Proof (1) Let $\sigma = \sigma' j$. Since $\sigma' j \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$ is enabled in G, from Lemma 7(1) for some session contexts $\mathcal{T}$ and $\mathcal{T}'$ we have $\mathcal{T} = \mathit{Ctx}(\mathsf{G}, \sigma') \restriction \mathsf{p}$ and $\mathcal{T}' = \mathit{Ctx}(\mathsf{G}, \sigma') \restriction \mathsf{q}$. Therefore the projections of G on p and q may be written as $\mathsf{G} \restriction \mathsf{p} = \mathcal{T}[_{\overline{\gamma}}(\bigvee_{i\in I\setminus\{j\}} \pi_i; \mathsf{T}_i \vee \mathsf{q}!\Lambda; \mathsf{T}_j)]$ and $\mathsf{G} \restriction \mathsf{q} = \mathcal{T}'[_{\Delta}(\bigwedge_{h\in H\setminus\{k\}} \pi'_h; \mathsf{S}_h \wedge \mathsf{p}?\Lambda; \mathsf{S}_k)]$.

By Lemma 2(6) $\mathbb{N} = \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \mathbb{N}''$ and $\vdash P : \mathsf{T}$ and $\vdash Q : \mathsf{S}$ and $\mathsf{T} \leqslant \mathsf{G} \restriction \mathsf{p}$ and $\mathsf{S} \leqslant \mathsf{G} \restriction \mathsf{q}$. By Lemma 5(2) and the definition of $\leqslant$ we have :

– $\mathsf{T} = \mathcal{T}_1[_{\overline{\gamma}} \bigvee_{i\in I} \pi_i; \mathsf{T}'_i]$ where $\pi_j = \mathsf{q}!\Lambda$ with $j \in I$ and $\mathsf{T}'_i \leqslant \mathsf{T}_i$ $(i \in I)$ and $\mathcal{T} \leqslant \mathcal{T}_1$
– $\mathsf{S} = \mathcal{T}_2[_{\Delta} \bigwedge_{h\in H'} \pi'_h; \mathsf{S}'_h]$ where $\pi_k = \mathsf{p}?\Lambda$ with $k \in H$ and $H' \supseteq H$ and $\mathsf{S}'_h \leqslant \mathsf{S}_h$ $(h \in H)$ and $\mathcal{T}' \leqslant \mathcal{T}_2$.

By Lemma 5(5) we get $P = \mathcal{E}[P']$ and $\vdash \mathcal{E} : \mathcal{T}_1$ and $\vdash P' : {}_{\overline{\gamma}} \bigvee_{i\in I} \pi_i; \mathsf{T}'_i$ and $Q = \mathcal{E}'[Q']$ and $\vdash \mathcal{E}' : \mathcal{T}_2$ and $\vdash Q' : {}_{\Delta} \bigwedge_{h\in H'} \pi'_h; \mathsf{S}'_h$. By Lemma 2(2) $P' = {}_{\overline{\gamma}} \bigoplus_{i\in I} \pi_i; P_i$ and by Lemma 2(1) $Q' = {}_{\Delta} \sum_{h\in H'} \pi'_h; Q_h$. Therefore

$$P \xrightarrow{\mathsf{q}!\Lambda} \mathcal{E}[P''] \text{ where } P'' = {}_{\overline{\gamma}}(\textstyle\bigoplus_{i\in I\setminus\{j\}} \pi_i;\, P_i \;\oplus\; \widehat{\mathsf{q}!\Lambda};\, P_j)$$

and

$$Q \xrightarrow{\mathsf{p}?\Lambda} \mathcal{E}'[Q''] \text{ where } Q'' = {}_{\Delta}(\textstyle\sum_{h\in H'\setminus\{k\}} \pi'_h;\, Q_h \;+\; \widehat{\mathsf{p}?\Lambda};\, Q_k),$$

which imply $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathsf{p}[\![\mathcal{E}[P'']]\!] \parallel \mathsf{q}[\![\mathcal{E}'[Q'']]\!] \parallel \mathbb{N}''$.

(2) By Lemma 4(6) $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0$ and $P \xrightarrow{\mathsf{q}!\Lambda} P'$ and $Q \xrightarrow{\mathsf{p}?\Lambda} Q'$. Therefore rule [Com] has been applied and from the proof of Theorem 1 we know that $\sigma \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$ is enabled in G for some σ.

(3) From Definition 17(4) we know that there exists a path σ ok for C such that $\mathit{SubT}(\mathsf{G}, \sigma) = {}_C(\boxplus_{j\in J\setminus\{k\}} \alpha_j^{\mathsf{p}}; \mathsf{K}_j \boxplus \widehat{\alpha_k^{\mathsf{p}}}; \mathsf{G}_k)$. Since $\mathsf{G} = \mathit{Ctx}(\mathsf{G}, \sigma)[\mathit{SubT}(\mathsf{G}, \sigma)]$ is causally correct all the communications involving p in $\mathit{Ctx}(\mathsf{G}, \sigma)$ are executed communications. This implies that $\mathcal{T} = \mathit{Ctx}(\mathsf{G}, \sigma) \restriction \mathsf{p}$ is a session context and thus the projection of G on p may be written as $\mathsf{G} \restriction \mathsf{p} = \mathcal{T}[{}_{\overline{\gamma}}(\bigvee_{j\in J\setminus\{k\}} \pi_j; \mathsf{T}_j \vee \widehat{\pi_k}; \mathsf{T})]$. Therefore $\mathcal{T}$ is ok for C and $(\alpha_j^{\mathsf{p}}; \mathsf{K}_j) \restriction \mathsf{p} = \pi_j; \mathsf{T}_j$ for $j \in J\setminus\{k\}$ and $(\widehat{\alpha_k^{\mathsf{p}}}; \mathsf{G}_k) \restriction \mathsf{p} = \widehat{\pi_k}; \mathsf{T}$. By Lemma 2(6) $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathbb{N}''$ and $\vdash P : \mathsf{T}'$ and $\mathsf{T}' \leqslant \mathsf{G} \restriction \mathsf{p}$. By Lemma 5(2) $\mathsf{T}' = \mathcal{T}'[{}_{\overline{\gamma}}(\bigvee_{j\in J\setminus\{k\}} \mathsf{S}_i \vee \widehat{\pi_k}; \mathsf{S})]$ and $\mathcal{T}' \leqslant \mathcal{T}$ and $\pi_j; \mathsf{S}_j \leqslant \pi_j; \mathsf{T}_j$ for $j \in J\setminus\{k\}$ and $\widehat{\pi_k}; \mathsf{S} \leqslant \widehat{\pi_k}; \mathsf{T}$.

By Lemma 5(5) $P = \mathcal{E}[P']$ and $\vdash \mathcal{E} : \mathcal{T}'$ and $\vdash P' : {}_{\overline{\gamma}}(\bigvee_{j\in J\setminus\{k\}} \pi_j; \mathsf{S}_j \vee \widehat{\pi_k}; \mathsf{S})$. By Lemma 2(2) $P' = \mathcal{E}[{}_{\overline{\gamma}}(\bigoplus_{j\in J\setminus\{k\}} \pi_j; Q_j \,\oplus\, \widehat{\pi_k}; Q)]$ and $\vdash \widehat{\pi_k}; Q : \widehat{\pi_k}; \mathsf{S}$. Let σ' and h be such that $\sigma k \sigma' h \alpha^{\mathsf{p}} \in \mathit{Occ}(\mathsf{G})$ is enabled in G. Then $\mathit{SubT}(\mathsf{G}, \sigma k \sigma') = {}_{\gamma'} \boxdot_{i\in I}\, \alpha_i^{\mathsf{p}}; \mathsf{K}_i$ with $h \in I$ and $\alpha_h^{\mathsf{p}} = \alpha^{\mathsf{p}}$ and for some $\mathcal{T}''$ we have that $\mathsf{T} = \mathcal{T}''[{}_{\overline{\gamma'}}(\bigvee_{i\in I} \mathsf{T}_i')]$. By Lemma 5(2) $\mathsf{S} = \mathcal{T}'''[{}_{\overline{\gamma'}}(\bigvee_{i\in I} \mathsf{S}_i')]$ for some $\mathcal{T}'''$. By Lemma 2(2) $Q = \mathcal{E}'[{}_{\overline{\gamma'}}\bigoplus_{i\in I} Q_i']$, which implies $Q \downarrow_{out}$. Then $P \overset{\overline{C}}{\circlearrowright} P'$ and $\mathbb{N} \overset{C}{\circlearrowright} \mathbb{N}'$, since all processes in $\mathbb{N}$ may contain or may not contain executed internal choices labelled C.

(4) By Lemma 4(7) $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \Pi_{h\in H} \mathsf{p}_h[\![\, P_h \,]\!] \parallel \Pi_{k\in K} \mathsf{p}_k[\![\, P_k \,]\!]$ and $P \overset{\overline{C}}{\circlearrowright} P'$ and $P_h \overset{C}{\circlearrowright} P_h'$ for all $h \in H$ and $P_k \overset{C}{\not\circlearrowright}$ for all $k \in K$ and

$$\mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \Pi_{h\in H} \mathsf{p}_h[\![\, P_h' \,]\!] \parallel \Pi_{k\in K} \mathsf{p}_k[\![\, P_k \,]\!]$$

Therefore rule [Back] has been applied and from the proof of Theorem 1 we know that:

- $P = \mathcal{E}[{}_{\overline{C}}(\bigoplus_{i\in I} \pi_i; Q_i \,\oplus\, \widehat{\pi}; Q)]$ and $Q \downarrow_{out}$ and $I \neq \emptyset$ and $\mathcal{E}$ ok for C
- $\mathit{SubT}(\mathsf{G}, \sigma) = {}_C(\boxplus_{i\in I} \alpha_i^{\mathsf{p}}; \mathsf{K}_i \boxplus \widehat{\alpha^{\mathsf{p}}}; \mathsf{G}_Q)$ and σ is ok for C
- $\vdash P : \mathsf{T}$ and $\mathsf{T} \leqslant \mathsf{G} \restriction \mathsf{p}$ and $\vdash Q : \mathsf{T}_Q$ and $\mathsf{T}_Q \leqslant \mathsf{G}_Q \restriction \mathsf{p}$
- $\mathsf{G} \restriction \mathsf{p} = \mathcal{T}'[{}_{\overline{C}}(\bigvee_{i\in I} \pi_i; \mathsf{T}_i' \vee \widehat{\pi}; \mathsf{T}_Q')]$ and $\mathsf{T}_Q \leqslant \mathsf{T}_Q'$.

Let $\widehat{\alpha^{\mathsf{p}}}; \mathsf{G}_Q$ be the ℓth branch in $\mathit{SubT}(\mathsf{G}, \sigma)$. By definition $Q \downarrow_{out}$ implies $Q = \mathcal{E}'[{}_{\overline{\gamma'}}\bigoplus_{j\in I'} Q_j']$.

From $\vdash Q : \mathsf{T}_Q$ and Lemma 5(3) $\mathsf{T}_Q = \mathcal{T}_Q[\mathsf{S}_Q]$ and $\vdash {}_{\overline{\gamma'}}\bigoplus_{j\in I'} Q_j' : \mathsf{S}_Q$. By Lemma 1(2) $\mathsf{S}_Q = {}_{\overline{\gamma'}} \bigvee_{j\in I'} \mathsf{S}_j$, which together with $\mathsf{T}_Q \leqslant \mathsf{T}_Q'$ imply $\mathsf{T}_Q' = \mathcal{T}_Q'[{}_{\overline{\gamma'}} \bigvee_{j\in I'} \mathsf{S}_j']$ by Lemma 5(1). Let $\mathcal{T}'' = \mathcal{T}'[{}_{\overline{C}}(\bigvee_{i\in I} \pi_i; \mathsf{T}_i' \vee \widehat{\pi}; \mathcal{T}_Q')]$, then $\mathsf{G} \restriction \mathsf{p} = \mathcal{T}''[{}_{\overline{\gamma'}} \bigvee_{j\in I'} \mathsf{S}_j']$. By Lemma 6(1) there is σ' such that $\mathit{SubT}(\mathsf{G}, \sigma \ell \sigma') = {}_{\overline{\gamma'}} \boxdot_{j\in I'}\, \beta_j^{\mathsf{p}}; \mathsf{K}_j'$ affects p and $\mathit{Ctx}(\mathsf{G}, \sigma \ell \sigma') \restriction \mathsf{p} = \mathcal{T}''$ and $\mathit{SubT}(\mathsf{G}, \sigma \ell \sigma') \restriction \mathsf{p} = {}_{\overline{\gamma'}} \bigvee_{j\in I'} \mathsf{S}_j'$. From Lemma 7(2) $\sigma \ell \sigma' j \beta_j^{\mathsf{p}}$ is alive in G for all $j \in I'$ and then C is alive in G. We finally conclude that C is enabled in G, since $\mathit{Ctx}(\mathsf{G}, \sigma \ell \sigma') \restriction \mathsf{p}$ is a session context, and this implies that $\sigma \ell \sigma' j \beta_j^{\mathsf{p}}$ is enabled in G for all $j \in I'$. $\square$

The remainder of this section is devoted to the proof of forward and backward progress.

For forward progress we first show that a communication which is alive in a global type has at least one enabled cause, and will eventually become enabled.

Lemma 8 *1. If $\xi \in Occ(\mathsf{G})$ is alive in G, then either ξ is enabled in G or there is some occurrence in $Causes(\mathsf{G}, \xi)$ which is enabled in G.*

2. If $\vdash \mathbb{N} : \mathsf{G}$ and $\xi \in Occ(\mathsf{G})$ is alive in G, then $\mathbb{N} \longrightarrow^* \mathbb{N}'$ with $\vdash \mathbb{N}' : \mathsf{G}'$ and ξ is enabled in G'.

Proof (1) By induction on the cardinality of the set of non executed occurrences in $Causes(\mathsf{G}, \xi)$, namely on $n = |Causes(\mathsf{G}, \xi)\backslash ExOcc(\mathsf{G})|$. If $n = 0$, then ξ is enabled in G by definition. Let $n > 0$. Note that every $\xi' \in Causes(\mathsf{G}, \xi)\backslash ExOcc(\mathsf{G})$ must be alive, because otherwise there would exist an executed ξ'' such that $\xi'' \#_{\mathsf{G}} \xi'$, which by conflict heredity would imply $\xi'' \#_{\mathsf{G}} \xi$, contradicting the assumption that ξ is alive. So, let us choose a non executed $\xi' \in Causes(\mathsf{G}, \xi)$. If ξ' is enabled we are done, otherwise, since ξ' is alive and $Causes(\mathsf{G}, \xi') \subset Causes(\mathsf{G}, \xi)$, by inductive hypothesis there is ξ'' in $Causes(\mathsf{G}, \xi')$ which is enabled in G.

(2) Again, we proceed by induction on $n = |Causes(\mathsf{G}, \xi)\backslash ExOcc(\mathsf{G})|$. If $n = 0$, then ξ is enabled in G and the result is immediate by Theorem 2(1). Let now $n > 0$. By (1) there exists ξ' in $Causes(\mathsf{G}, \xi)$ which is enabled in G. If $\xi' = \sigma \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$, by Theorem 2(1) we have $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}''$ and then by Theorem 1 we have $\vdash \mathbb{N}'' : \mathsf{G}''$. Let $\xi'' = \widehat{\sigma \mathsf{p} \xrightarrow{\Lambda} \mathsf{q}}$. Since $Causes(\mathsf{G}'', \xi) = Causes(\mathsf{G}, \xi) \setminus \{\xi'\} \cup \{\xi''\}$ and $ExOcc(\mathsf{G}'') = ExOcc(\mathsf{G}) \cup \xi''$, we have $Causes(\mathsf{G}'', \xi)\backslash ExOcc(\mathsf{G}'') \subset Causes(\mathsf{G}, \xi)\backslash ExOcc(\mathsf{G})$. Then by induction $\mathbb{N}'' \longrightarrow^* \mathbb{N}'$ (whence $\mathbb{N} \longrightarrow^* \mathbb{N}'$) with $\vdash \mathbb{N}' : \mathsf{G}'$ and ξ is enabled in G'. □

Connecting communications cause the failure of the standard progress property, since a participant offering an external choice between connecting communications can wait forever. Instead, processes offering outputs or simple inputs can always communicate.

Theorem 3 (Forward progress)

1. *If $\vdash \mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N} : \mathsf{G}$, then for every $j \in I$ there exists an $\mathbb{N}_j$ such that $\mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N} \longrightarrow^* \mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}(\bigoplus_{i\in I\backslash\{j\}} \pi_i; P_i \oplus \widehat{\pi_j}; P_j)]\,]\!] \parallel \mathbb{N}_j$.*
2. *If $\vdash \mathsf{p}[\![\, \mathcal{E}[_{\Delta}\sum_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N} : \mathsf{G}$ and all π_i are simple inputs, then there is $\mathbb{N}'$ such that $\mathsf{p}[\![\, \mathcal{E}[_{\Delta}\sum_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N} \longrightarrow^* \mathsf{p}[\![\, \mathcal{E}[_{\Delta}(\sum_{i\in I\backslash\{j\}} \pi_i; P_i + \widehat{\pi_j}; P_j)]\,]\!] \parallel \mathbb{N}'$ for some $j \in I$.*

Proof (1) Let $P = \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]$, by Lemma 1(6) $\vdash P : \mathsf{T}$ and $\mathsf{T} \leqslant \mathsf{G}\restriction\mathsf{p}$. By Lemma 5(3) $\mathsf{T} = \mathcal{T}[\mathsf{T}']$ and $\vdash \mathcal{E} : \mathcal{T}$ and $\vdash {}_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i : \mathsf{T}'$. By Lemma 1(2) $\mathsf{T}' = {}_{\overline{\gamma}}\bigvee_{i\in I} \pi_i; \mathsf{T}_i$. By Lemma 5(1) $\mathsf{G}\restriction\mathsf{p} = \mathcal{T}'[\mathsf{T}'']$ and $\mathcal{T} \leqslant \mathcal{T}'$ and ${}_{\overline{\gamma}}\bigvee_{i\in I} \pi_i; \mathsf{T}_i \leqslant \mathsf{T}''$. By definition of $\leqslant$ we get $\mathsf{T}'' = {}_{\overline{\gamma}}\bigvee_{i\in I} \pi_i; \mathsf{T}'_i$ with $\mathsf{T}_i \leqslant \mathsf{T}'_i$ for $i \in I$. Therefore $\mathsf{G}\restriction\mathsf{p} = \mathcal{T}'[{}_{\overline{\gamma}}\bigvee_{i\in I} \pi_i; \mathsf{T}'_i]$.

From Lemma 6(1), for some σ we have that $SubT(\mathsf{G}, \sigma) = {}_{\gamma} \boxdot_{i\in I}\, \alpha_i^{\mathsf{p}}; \mathsf{K}_i$ and $Ctx(\mathsf{G}, \sigma)\restriction \mathsf{p} = \mathcal{T}'$ and $SubT(\mathsf{G}, \sigma)\restriction\mathsf{p} = {}_{\overline{\gamma}}\bigvee_{i\in I} \pi_i; \mathsf{T}'_i$. This implies, by Lemma 7(2), that $\sigma\, j\alpha_j^{\mathsf{p}}$ is alive in G. Therefore from Lemma 8(2) we get that

$$\mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N} \longrightarrow^* \mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N}'_j$$

with $\vdash \mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N}'_j : \mathsf{G}'$ and $\sigma\, j\alpha_j^{\mathsf{p}}$ enabled in G'. From Theorem 2(1) we conclude $\mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}\bigoplus_{i\in I} \pi_i; P_i]\,]\!] \parallel \mathbb{N}'_j \xrightarrow{\alpha_j^{\mathsf{p}}} \mathsf{p}[\![\, \mathcal{E}[_{\overline{\gamma}}(\bigoplus_{i\in I\backslash\{j\}} \pi_i; P_i \oplus \widehat{\pi_j}; P_j)]\,]\!] \parallel \mathbb{N}_j$.

(2) Let $P = \mathcal{E}[_{\Delta}\sum_{i\in I} \pi_i; P_i]$, by Lemma 1(6) $\vdash P : \mathsf{T}$ and $\mathsf{T} \leqslant \mathsf{G}\restriction\mathsf{p}$. By Lemma 5(3) $\mathsf{T} = \mathcal{T}[\mathsf{T}']$ and $\vdash \mathcal{E} : \mathcal{T}$ and $\vdash {}_{\Delta}\sum_{i\in I} \pi_i; P_i : \mathsf{T}'$. By Lemma 1(1) $\mathsf{T}' = {}_{\Delta}\bigwedge_{i\in I} \pi_i; \mathsf{T}_i$. By Lemma 5(1) $\mathsf{G}\restriction\mathsf{p} = \mathcal{T}'[\mathsf{T}'']$ and $\mathcal{T} \leqslant \mathcal{T}'$ and ${}_{\Delta}\bigwedge_{i\in I} \pi_i; \mathsf{T}_i \leqslant \mathsf{T}''$. By definition

of $\leqslant$ we get and $\mathsf{T}'' = {}_{\Delta}\bigwedge_{i\in I'}\pi_i; \mathsf{T}'_i$ with $I \supseteq I'$ and $\mathsf{T}_i \leqslant \mathsf{T}'_i$ for $i \in I'$. Therefore $\mathsf{G}\upharpoonright\mathsf{p} = \mathcal{T}'[{}_{\Delta}\bigwedge_{i\in I'}\pi_i; \mathsf{T}'_i]$. From Lemma 6(2) we have that there is σ such that $\mathit{SubT}(\mathsf{G},\sigma)$ affects p and $\mathit{Ctx}(\mathsf{G},\sigma)\upharpoonright\mathsf{p} = \mathcal{T}'$ and $\mathit{SubT}(\mathsf{G},\sigma)\upharpoonright\mathsf{p} \doteq {}_{\Delta}\bigwedge_{i\in I'}\pi_i; \mathsf{T}'_i$. For each $i \in I'$, the action π_i must have the form $\pi_i = \mathsf{q}_i?\lambda_i$ and be obtained projecting a communication $\alpha_i = \mathsf{q}_i \xrightarrow{\lambda_i} \mathsf{p}$. We consider two cases: either $\mathit{SubT}(\mathsf{G},\sigma)$ is an executed choice, or it is either a non executed choice or a sequence starting with a non executed communication.

- If $\mathit{SubT}(\mathsf{G},\sigma) = {}_{\gamma}(\boxdot_{h\in H\setminus\{k\}}\beta_h; \mathsf{K}_h \boxdot \widehat{\beta_k}; \mathsf{G}_k)$, then let σ_k be the longest path in $\mathit{SubT}(\mathsf{G},\sigma)$ such that $k \sqsubseteq \sigma_k$ and $\sigma\sigma_k = \mathsf{path}(\xi)$ for some $\xi \in \mathit{ExOcc}(\mathsf{G})$. By Lemma 6(2) there are σ' and $j \in I'$ such that $\sigma\sigma'\alpha_j \in \mathit{Occ}(\mathsf{G})$ and either $\sigma' \sqsubseteq \sigma_k$ or $\sigma_k \sqsubset \sigma'$. Since $\xi' \in \mathit{ExOcc}(\mathsf{G})$ implies $\mathsf{path}(\xi') \sqsubseteq \sigma\sigma_k$, then ξ' is not in conflict with $\sigma\sigma'\alpha_j$. Therefore $\sigma\sigma'\alpha_j$ is alive in G.
- If $\mathit{SubT}(\mathsf{G},\sigma) = {}_{\gamma}\boxdot_{h\in H}\ \beta_h; \mathsf{G}_h$, then $\sigma h\beta_h$ is alive in G for all $h \in H$ by Lemma 7(2). Then by Lemma 6(2) for any $h \in H$ we can find σ_h such that $\sigma\sigma_h\alpha_j$ is alive in G for some $j \in I'$.

In both cases there is σ_{p} such that $\sigma_{\mathsf{p}}\alpha_j$ is alive in G for some $j \in I'$. Lemma 8(2) implies

$$\mathsf{p}[\![\,\mathcal{E}[{}_{\Delta}\textstyle\sum_{i\in I}\pi_i; P_i]\,]\!] \parallel \mathbb{N} \longrightarrow^* \mathsf{p}[\![\,\mathcal{E}[{}_{\Delta}\textstyle\sum_{i\in I}\pi_i; P_i]\,]\!] \parallel \mathbb{N}'_j$$

and $\vdash \mathsf{p}[\![\,\mathcal{E}[{}_{\Delta}\sum_{i\in I}\pi_i; P_i]\,]\!] \parallel \mathbb{N}'_j : \mathsf{G}'$ with $\sigma_{\mathsf{p}}\alpha_j$ enabled in G'. From Theorem 2(1) we conclude $\mathsf{p}[\![\,\mathcal{E}[{}_{\Delta}\sum_{i\in I}\pi_i; P_i]\,]\!] \parallel \mathbb{N}'_j \xrightarrow{\alpha_j} \mathsf{p}[\![\,\mathcal{E}[{}_{\Delta}(\sum_{i\in I\setminus\{j\}}\pi_i; P_i \oplus \widehat{\pi_j}; P_j)]\,]\!] \parallel \mathbb{N}'_j$. □

Notice that the standard formulation of progress [13], which requires that each *simple* input and each output that is persistently offered be eventually consumed, is an easy consequence of this theorem.

Theorem 4 (Backward progress) *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* C *is alive in* G*, then there is* $\mathbb{N}'$ *such that* $\mathbb{N} \longrightarrow^* \mathbb{N}'$ *and* $\mathbb{N}' \overset{C}{\circlearrowleft} \mathbb{N}''$.

Proof By definition $\mathsf{G} = \mathit{Ctx}(\mathsf{G},\sigma)[{}_{C}(\boxplus_{j\in J\setminus\{k\}}\alpha^{\mathsf{p}}_j; \mathsf{K}_j \boxplus \widehat{\alpha^{\mathsf{p}}_k}; \mathsf{G}_k)]$ and $|J| > 1$ and σ is ok for C (in G) and there are σ' and α^{p} such that $\sigma k\sigma'\alpha^{\mathsf{p}} \in \mathit{Occ}(\mathsf{G})$ is alive in G. Lemma 8(2) implies that $\mathbb{N} \longrightarrow^* \mathbb{N}'$ and $\vdash \mathbb{N} : \mathsf{G}'$ with $\sigma k\sigma'\alpha^{\mathsf{p}}$ enabled in G'.

This implies that C is enabled in G. We conclude $\mathbb{N}' \overset{C}{\circlearrowleft} \mathbb{N}''$ by Theorem 2(3). □

To sum up, our calculus enjoys:

- subject reduction for both forward and backward computations;
- session fidelity for enabled communications and for rollbacks to enabled checkpoint labels;
- forward progress, assuring the absence of dangling actions but for connecting inputs;
- backward progress for alive checkpoint labels.

We end this section with a remark on causal consistency and the loop lemma [10]. Causal consistency states that a cause cannot be reversed without first reversing its effects. Clearly, our calculus enjoys causal consistency since a rollback to a checkpointed choice removes all communications that were done after that checkpoint. The loop lemma prescribes that each transition has an inverse. The loop lemma does not hold for our calculus, since a reverse computation can only go back to a checkpointed choice whose leader currently offers an output.

6 Related work and conclusion

Since the seminal work by Danos and Krivine on reversible CCS [10], reversible computation has been widely studied in process calculi. In [31], Phillips and Ulidowski proposed a method for reversing process operators defined in a general SOS format, and noted that thread identifiers and histories were needed to record the past of computations. In [30] the authors use a cursor to mark past actions, thus avoiding the use of extra memory information as in [10].

In [23], Lanese et al. extended the approach of Danos and Krivine by defining a reversible variant of the higher-order π-calculus, using tags to identify threads, and explicit memory processes. This calculus was enriched with a fine-grained rollback primitive in [22]. In [9], Cristescu et al. proposed a causal semantic model for the reversible π-calculus.

Reversibility for structured communications was first studied in [11,12,21], where *transactions* with rollback and coordinated checkpoints were modelled in an extended CCS. More recently, reversibility has been incorporated into *contracts* [1,4] and *session calculi* [17,18]. In [2,3], the authors investigated the notions of compliance and sub-behaviour for contracts with checkpoints. While rollbacks are forgetful in [2], in [3] they are used as a strategy to achieve compliance: in this case, after a rollback a process cannot engage again in the previously explored branch, presumably unsuccessful. We used a similar idea for defining our rollback here.

Our backward mechanism is most closely related to the recent proposals [24–26,34,35]. Tiezzi and Yoshida [34] use tags and memories to allow full reversibility of binary sessions with delegation. In [35], two forms of reversibility are considered: either a session is completely reversed in a single backward step, or any intermediate state is restored. Mezzina and Pérez [24,25] use monitors as memories for reversing binary sessions. A key novelty of this work is the use of session types with present and past. In [26], this approach is generalised to multiparty sessions, asynchronous higher-order communications, and decoupled rollbacks.

In [28], Neykova and Yoshida provide an algorithm to analyse and extract causal dependencies from a given multiparty global type, and use it to ensure that communicating processes are safely recovered from consistent states in the presence of a failure. In [27], Mezzina and Tuosto propose a semantic control of reversibility: a computation along a branch is reversed according to the guards on the current configuration. A feature of [27] is that inputs are potentially irreversible actions, unless they appear within a loop.

Our work builds on our previous papers [8,14]. In [14] we introduced a multiparty session calculus in which choices could be labelled with checkpoints. Participants could revert to one of these checkpointed choices in order to make a different choice. Global types were used to control reversibility, and shown to enforce the properties of fidelity and progress (both forward and backward). In [8] we enriched the syntax for types and processes with parallel and sequential composition, and used a more compact representation for past communications and a more refined strategy for backward moves. The current work borrows ideas and techniques from both papers while sticking to a simpler syntax than [8], and improves on both of them by allowing connecting communications.

As regards dynamic participants, in [13,15] global types prescribe the behaviours of fixed roles, and each role includes an arbitrary number of participants, which can dynamically join and leave roles. Instead, in the Conversation Calculus [7,36] the protocols describe communications between participants which can dynamically join and leave conversations. The model based on conversation contexts and labelled message-passing primitives is quite different from multiparty session types and this makes it difficult to adapt their approach to our setting.

The paper [20] has been our inspiration for connecting messages, but our calculus is more permissive than the original one. In fact, in [20] a connecting message can be exchanged only between two participants that did not communicate before, unless they have been disconnected by an ad hoc primitive. This means for example that $\mathsf{p} \xrightarrow{\overset{\lambda_1}{\leftrightarrow}} \mathsf{q}; \mathsf{q} \xrightarrow{\overset{\lambda_2}{\leftrightarrow}} \mathsf{p}$ is not allowed. Another difference of [20] with respect to our calculus is that in a choice, the first message received by a participant must be a connecting message, which forbids for instance $\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q} \boxplus \mathsf{p} \xrightarrow{\lambda_2} \mathsf{r}$. A further restriction is that all the inputs in an external choice must have the same sender. A final major difference is that communication in [20] is asynchronous.

The properties of our calculus are standard for reversible session calculi, but their proofs require some ingenuity due to the presence of connecting communications and to the specificity of our rollback mechanism. As argued already in Sect. 3, connecting communications may be used to avoid the use of multiple global types. They could also be useful for incorporating delegation into global types. For example, a seller could delegate a bank to receive a credit card number only when the client wants to buy an item, otherwise the bank is not involved in the transaction (so delegation would appear in one branch of a choice but not in the others). This is a topic we plan to investigate. As regards the conditions for reversing a computation, a current limitation of our work is that the starting points of rollbacks are statically determined. By contrast, these points are determined dynamically in [27], offering a more realistic solution. We plan to introduce similar runtime conditions for rollback in our calculus.

Finally, we would like to study the interpretation of global types into a model of Event Structures, for which reversible variants have already been proposed [16,32]. We plan to explore a reversible variant of *Flow Event Structures* [5], a model that has already been used to interpret CCS processes with past in [6].

Acknowledgements We would like to thank the anonymous referees for their helpful comments.

A: Reductions of networks and systems

It is easy to define a bijection between networks and initial systems:

$$\mathcal{S}(\mathbb{N}) = \mathbb{N} \between \langle -, \mathbb{P}, \emptyset \rangle \qquad\qquad \mathcal{N}(\mathbb{N} \between \langle -, \mathbb{P}, \emptyset \rangle) = \mathbb{N}$$

where $\mathbb{P}$ is the set of participants in $\mathbb{N}$.

We establish now an operational correspondence between networks and their associated systems. More precisely we prove the following:

- *Communication preservation and reflection:* every network communication is simulated by a communication in the associated system and vice-versa;
- *Rollback preservation:* every network rollback is simulated by a sequence of backward moves in the associated system, initiating with a starting backward move and terminating with an ending backward move;
- *Rollback reflection:* every starting backward move in a system can be extended to a complete sequence of backward moves such that the resulting system is the image of a rollback in the source network.

In the following theorem we denote by $\overset{C}{\circlearrowright\!\!\circlearrowright}$ the transitive closure of $\overset{\mathsf{y}_C}{\circlearrowright}$ and $\overset{\mathsf{n}_C}{\circlearrowright}$.

Theorem 5 1. *If* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$, *then* $\mathcal{S}(\mathbb{N}) \xRightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathcal{S}(\mathbb{N}')$.

2. *If* $\mathcal{S}(\mathbb{N}) \overset{\mathsf{p}\Lambda\mathsf{q}}{\Longrightarrow} \mathcal{S}(\mathbb{N}')$*, then* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$.
3. *If* $\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}'$*, then* $\mathcal{S}(\mathbb{N}) \overset{\mathsf{s}_C}{\circlearrowleft}\overset{C}{\circlearrowleft^{*}}\overset{\mathsf{e}_C}{\circlearrowleft} \mathcal{S}(\mathbb{N}')$.
4. *If* $\mathbb{N} \between \langle -, \mathbb{P}, \emptyset\rangle \overset{\mathsf{s}_C}{\circlearrowleft} \mathbb{N}' \between \langle C, \mathbb{P}_1, \mathbb{P}_2\rangle$*, then* $\mathbb{N}' \between \langle C, \mathbb{P}_1, \mathbb{P}_2\rangle \overset{C}{\circlearrowleft^{*}}\overset{\mathsf{e}_C}{\circlearrowleft} \mathbb{N}'' \between \langle -, \mathbb{P}, \emptyset\rangle$ *and* $\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}''$.

Proof (1) and (2). The result is immediate since Rule [COM] of Fig. 1 and Rule [COMS] of Fig. 2 have the same antecedents.

(3). If $\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}'$, then the applied rule is

$$\frac{P \overset{\overline{C}}{\circlearrowleft} P' \quad P_i \overset{C}{\circlearrowleft} P_i' \;\; 1 \le i \le m \quad P_i \overset{C}{\not\circlearrowleft} \;\; m+1 \le i \le n}{\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}'}\,[\textsc{Back}]$$

where

$$\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \Pi_{1\le i\le n}\mathsf{p}_i[\![\, P_i \,]\!] \quad \text{and} \quad \mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \Pi_{1\le i\le m}\mathsf{p}_i[\![\, P_i' \,]\!] \parallel \Pi_{m+1\le i\le n}\mathsf{p}_i[\![\, P_i \,]\!]$$

Then $\mathcal{S}(\mathbb{N}) = \mathbb{N} \between \sigma$ and $\mathcal{S}(\mathbb{N}') = \mathbb{N}' \between \sigma$ with $\sigma = \langle -, \{\mathsf{p}\} \cup \{\mathsf{p}_i \mid 1 \le i \le n\}, \emptyset\rangle$. We get

$$\begin{aligned}
\mathbb{N} \between \sigma &\overset{\mathsf{s}_C}{\circlearrowleft} \mathsf{p}[\![\, P' \,]\!] \parallel \Pi_{1\le i\le n}\mathsf{p}_i[\![\, P_i \,]\!] \between \langle C, \{\mathsf{p}\}, \{\mathsf{p}_i \mid 1 \le i \le n\}\rangle \\
&\overset{\mathsf{y}_C}{\circlearrowleft} \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{p}_1[\![\, P_1' \,]\!] \parallel \Pi_{2\le i\le n}\mathsf{p}_i[\![\, P_i \,]\!] \between \langle C, \{\mathsf{p}, \mathsf{p}_1\}, \{\mathsf{p}_i \mid 2 \le i \le n\}\rangle \\
&\overset{\mathsf{y}_C}{\circlearrowleft} \cdots \\
&\overset{\mathsf{y}_C}{\circlearrowleft} \mathbb{N}' \between \langle C, \{\mathsf{p}\} \cup \{\mathsf{p}_i \mid 1 \le i \le m\}, \{\mathsf{p}_i \mid m+1 \le i \le n\}\rangle \\
&\overset{\mathsf{n}_C}{\circlearrowleft} \mathbb{N}' \between \langle C, \{\mathsf{p}\} \cup \{\mathsf{p}_i \mid 1 \le i \le m+1\}, \{\mathsf{p}_i \mid m+2 \le i \le n\}\rangle \\
&\overset{\mathsf{n}_C}{\circlearrowleft} \cdots \\
&\overset{\mathsf{n}_C}{\circlearrowleft} \mathbb{N}' \between \langle C, \{\mathsf{p}\} \cup \{\mathsf{p}_i \mid 1 \le i \le n\}, \emptyset\rangle \\
&\overset{\mathsf{e}_C}{\circlearrowleft} \mathbb{N}' \between \sigma
\end{aligned}$$

(4). If $\mathbb{N} \between \langle -, \mathbb{P}, \emptyset\rangle \overset{\mathsf{s}_C}{\circlearrowleft} \mathbb{N}' \between \langle C, \mathbb{P}_1, \mathbb{P}_2\rangle$, then Rule [BACKS] has been applied, so

$$P \overset{\overline{C}}{\circlearrowleft} P' \qquad \mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathbb{N}_1 \qquad \mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \mathbb{N}_1 \qquad \mathbb{P}_1 = \{\mathsf{p}\} \qquad \mathbb{P}_2 = \mathbb{P} \setminus \{\mathsf{p}\}$$

Since $\mathbb{P}_2$ can be split in two subsets of participants, according to whether the associated processes satisfy the premise of Rule [BACKY] or of Rule [BACKN] in Fig. 2, we may assume without loss of generality that

$$\mathbb{N}_1 = \Pi_{1\le i\le m}\mathsf{p}_i[\![\, P_i \,]\!] \parallel \Pi_{m+1\le i\le n}\mathsf{p}_i[\![\, P_i \,]\!]$$

where for all i, $1 \le i \le n$, $P_i \overset{C}{\circlearrowleft} P_i'$ if $i \le m$ and $P_i \overset{C}{\not\circlearrowleft}$ otherwise. With a sequence of reductions as in the proof of point (3), we then obtain for some network $\mathbb{N}''$:

$$\mathbb{N}' \between \langle C, \{\mathsf{p}\}, \mathbb{P} \setminus \{\mathsf{p}\}\rangle \overset{C}{\circlearrowleft^{*}}\overset{\mathsf{e}_C}{\circlearrowleft} \mathbb{N}'' \between \langle -, \mathbb{P}, \emptyset\rangle.$$

Since P and all the P_i for $1 \le i \le n$ satisfy the premises of Rule [BACK] in Fig. 1, we may apply this rule to $\mathbb{N}$ to conclude $\mathbb{N} \overset{C}{\circlearrowleft} \mathbb{N}''$, as required. □

References

1. Barbanera, F., de' Liguoro, U.: Sub-behaviour relations for session-based client/server systems. Math. Struct. Comput. Sci. **25**(6), 1339–1381 (2015)
2. Barbanera, F., Dezani-Ciancaglini, M., de'Liguoro, U.: Reversible client/server interactions. Form. Asp. Comput. **28**(4), 697–722 (2016)
3. Barbanera, F., Dezani-Ciancaglini, M., Lanese, I., de' Liguoro, U.: Retractable contracts. In: PLACES, volume 203 of EPTCS, pp. 61–72 (2016)
4. Bernardi, G., Hennessy, M.: Modelling session types using contracts. Math. Struct. Comput. Sci. **26**(3), 510–560 (2016)
5. Boudol, G., Castellani, I.: Permutation of transitions: an event structure semantics for CCS and SCCS. In: REX School, volume 354 of LNCS, pp. 411–427. Springer (1988)
6. Boudol, G., Castellani, I.: Flow models of distributed computations: three equivalent semantics for CCS. Inf. Comput. **114**(2), 247–314 (1994)
7. Caires, L., Vieira, H.T.: Analysis of service oriented software systems with the conversation calculus. In: FACS, volume 6921 of LNCS, pp. 6–33. Springer (2010)
8. Castellani, I., Dezani-Ciancaglini, M., Giannini, P.: Concurrent reversible sessions. In: CONCUR, volume 85 of LIPIcs, pp. 30:1–30:17. Schloss Dagstuhl (2017)
9. Cristescu, I., Krivine, J., Varacca, D.: Rigid families for the reversible π-calculus. In: RC, volume 9720 of LNCS, pp. 3–19. Springer (2016)
10. Danos, V., Krivine, J.: Reversible communicating systems. In: CONCUR, volume 3170 of LNCS, pp. 292–307. Springer (2004)
11. de Vries, E., Koutavas, V., Hennessy, M.: Communicating transactions—(extended abstract). In: CONCUR, volume 6269 of LNCS, pp. 569–583. Springer (2010)
12. de Vries, E., Koutavas, V., Hennessy, M.: Liveness of communicating transactions—(extended abstract). In: APLAS, volume 6461 of LNCS, pp. 392–407. Springer (2010)
13. Deniélou, P.-M., Yoshida, N.: Dynamic multirole session types. In: POPL, pp. 435–446. ACM Press (2011)
14. Dezani-Ciancaglini, M., Giannini, P.: Reversible multiparty sessions with checkpoints. In: EXPRESS/SOS, volume 222 of EPTCS, pp. 60–74 (2016)
15. Giachino, E., Sackman, M., Drossopoulou, S., Eisenbach, S.: Softly safely spoken: Role playing for session types. In: PLACES (2009)
16. Graversen, E., Phillips, I., Yoshida, N.: Towards a categorical representation of reversible event structures. In: PLACES, volume 246 of EPTCS, pp. 49–60 (2017)
17. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type disciplines for structured communication-based programming. In: ESOP, volume 1381 of LNCS, pp. 22–138. Springer (1998)
18. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: POPL, pp. 273–284. ACM Press (2008)
19. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. J. ACM **63**(1), 9 (2016)
20. Hu, R., Yoshida, N.: Explicit connection actions in multiparty session types. In: Fundamental Approaches to Software Engineering, volume 10202 of LNCS, pp. 116–133. Springer (2017)
21. Koutavas, V., Spaccasassi, C., Hennessy, M.: Bisimulations for communicating transactions—(extended abstract). In: FOSSACS, volume 8412 of LNCS, pp. 320–334. Springer (2014)
22. Lanese, I., Mezzina, C.A., Schmitt, A., Stefani, J.-B.: Controlling reversibility in higher-order pi. In: CONCUR, volume 6901 of LNCS, pp. 297–311. Springer (2011)
23. Lanese, I., Mezzina, C.A., Stefani, J.-B.: Reversing higher-order pi. In: CONCUR, volume 6269 of LNCS, pp. 478–493. Springer (2010)
24. Mezzina, C.A., Pérez, J.A.: Reversible semantics in session-based concurrency. In: ICTCS, volume 1720 of CEUR, pp. 221–226 (2016). CEUR-WS.org
25. Mezzina, C.A., Pérez, J.A.: Reversible sessions using monitors. In: PLACES, volume 211 of EPTCS, pp. 56–64 (2016)
26. Mezzina, C.A., Pérez, J.A.: Causally consistent reversible choreographies: a monitors-as-memories approach. In: PPDP, pp. 127–138. ACM Press (2017)
27. Mezzina, C.A., Tuosto, E.: Choreographies for automatic recovery (2017). CoRR, arXiv:1705.09525
28. Neykova, R., Yoshida, N.: Let it recover: multiparty protocol-induced recovery. In: CC, pp. 98–108. ACM Press (2017)
29. Padovani, L.: Type reconstruction for the linear π-calculus with composite regular types. Log. Methods Comput. Sci. **11**(4), 1–23 (2015)
30. Phillips, I., Ulidowski, I.: Operational semantics of reversibility in process algebra. In: APC, volume 162 of ENTCS, pp. 281–286 (2006)

31. Phillips, I., Ulidowski, I.: Reversing algebraic process calculi. J. Log. Algebr. Methods Program. **73**(1–2), 70–96 (2007)
32. Phillips, I., Ulidowski, I.: Reversibility and asymmetric conflict in event structures. J. Log. Algebr. Methods Program. **84**(6), 781–805 (2015)
33. Pierce, B.C.: Types and Programming Languages. MIT Press, Cambridge (2002)
34. Tiezzi, F., Yoshida, N.: Reversible session-based pi-calculus. J. Log. Algebr. Methods Program. **84**(5), 684–707 (2015)
35. Tiezzi, F., Yoshida, N.: Reversing single sessions. In: RC, volume 9720 of LNCS, pp. 52–69. Springer (2016)
36. Vieira, H.T., Caires, L., Seco, J.C.: The conversation calculus: a model of service-oriented computation. In: ESOP, volume 4960 of LNCS, pp. 269–283. Springer (2008)
37. Winskel, G.: Events in Computation. PhD thesis, Department of Computer Science, University of Edinburgh (1980)