Global Types with Internal Delegation

To the memory of Maurice Nivat [1]

Ilaria Castellani[a], Mariangiola Dezani-Ciancaglini[2b], Paola Giannini[3c], Ross Horne[d]

[a] *INRIA, Université Côte d'Azur, Sophia Antipolis, France*
[b] *Dipartimento di Informatica, Università di Torino, Italy*
[c] *DiSIT, Università del Piemonte Orientale, Alessandria, Italy*
[d] *Computer Science and Communications Research Unit, University of Luxembourg*

Abstract

This paper investigates a new form of delegation for multiparty session calculi. Usually, delegation allows a session participant to appoint a participant in another session to act on her behalf. This means that delegation is inherently an inter-session mechanism, which requires session interleaving. Hence delegation falls outside the descriptive power of global types, which specify single sessions. As a consequence, properties such as deadlock-freedom or lock-freedom are difficult to ensure in the presence of delegation. Here we adopt a different view of delegation, by allowing participants to delegate tasks to each other within the same multiparty session. This way, delegation occurs within a single session (internal delegation) and may be captured by its global type. To increase flexibility in the use of delegation, our calculus uses connecting communications, which allow optional participants in the branches of choices. By these means, we are able to express conditional delegation. We present a session type system based on global types with internal delegation, and show that it ensures the usual safety properties of multiparty sessions, together with a progress property.

Keywords: Communication-centric Systems, Process Calculi, Multiparty Session Types.

1. Introduction

Session types model the sequence and types of messages exchanged between a number of parties. In early work [16], session types were enhanced with the ability to *delegate* interactions from one participant in a session to another participant in a different session. Thereby, by using delegation, a participant involved in a session can at any point request that some participant in a different session conducts part of the interaction on her behalf.

Typically delegation is modelled by sending a channel, on which the session is conducted, over another channel. For example, suppose two participants — *Client* and *Seller* — are initially engaged in a session on a channel, say *cs*. During the session, the participant seller would like another participant, *Bank*, which does not belong to the current session, to process a payment on her behalf. For a seamless experience for the client, the seller opens a new session with the bank, say on channel *sb*, and sends the initial session channel *cs* over the new channel *sb* to the bank, who is then able to complete the session with the client on behalf of the seller. This avoids the client directly opening a new session with the bank; it also avoids any need for the seller to act as an intermediary, forwarding messages back and forth between the client and the bank. After delegation, all interactions occur directly between the client and the bank. The client does not know that

[2]Partially supported by EU H2020-644235 Rephrase project, EU H2020-644298 HyVar project, IC1402 ARVI and Ateneo/CSP project RunVar.
[3]This original research has the financial support of the Università del Piemonte Orientale.

the seller has subcontracted part of its interaction to the bank. Such subcontracting may be useful both for abstraction purposes, to hide useless details from some participant (the client does not need to know how the transaction is implemented) or to provide additional guarantees on sensitive parts of the interaction (the credit card should only be processed by a trusted entity like the bank).

The idea of delegation originates in early research into concurrent object-oriented programming [35, 2]. Hence, delegation via channels, as described above, has been natural to implement in object-oriented languages [11], such as Java [20], F# [9] and Scala [32]. With regards to session calculi, delegation for binary sessions was introduced in [16]. The first paper with global types for multiparty sessions [17] also considers delegation. Delegation increases the expressiveness of a session calculus regardless of whether the sessions are binary or multiparty. In fact, channel-based delegation allows communications between participants which operate in different sessions and are unknown to each other. This behaviour cannot be obtained in session calculi without delegation.

This work takes a slightly more abstract approach to delegation than the channel-passing approach described above. In our calculus, there are no channels representing sessions, and hence delegation cannot be explicitly modelled by passing channels over channels. Instead, delegation is modelled by enabling one participant in a given multiparty session (the principal) to temporarily "lend" her behaviour to another participant in the same session (the deputy). The approach adopted has two advantages:

- Firstly, by being more abstract, we are less explicit about how delegation is implemented; thereby permitting a wide range of delegation mechanisms to be modelled. For example, delegation to a bank may be implemented as a browser redirection secured by a delegation certificate [1], which is quite different from the object-oriented approach. Thus, as with related work on extending Scribble with explicit connections [19], the global types in this work are largely language-independent.

- Secondly, by "localising" delegation within a single multiparty session, we get around two well-known issues of multiparty sessions types, both of which may prevent progress: session interleaving (which is required to model delegation in the channel-based approach), and the "non-local" character of channel-based delegation, which is inherently shared between two sessions, and hence cannot be captured by standard global types.

Global types provide a view of the behaviour of all participants partaking in a session [18, 27]. They are a required feature of all multiparty session frameworks. Without global types, using standard techniques, deadlock cannot be prevented even within a single multiparty session: one can easily program a 3-philosopher deadlock by assuming three participants in the same session, which are all waiting to receive from their left neighbour before sending to their right neighbour. These participants are all pairwise dual but the session is stuck. The problem can be solved by using global types, and requiring the session types of all participants to be projections of the same global type.

Global types are closely related to message sequence charts [22, 24], widely understood by engineers as a tool for describing the flow of messages in a protocol. This work explores how global types can be enhanced with mechanisms for describing internal delegation (i.e., delegation within the same session), thus going beyond the expressiveness of message sequence charts.

Existing approaches to global types with delegation include the interleaving approach where several sessions are described separately and their actions may be interleaved [17, 5]. A problem with having several separate global types is that it is not immediately clear how the actions in one session are related to the actions in another session. Worse still, without a view of how all interactions are related, interleaving between different sessions may cause deadlock, preventing progress of typed sessions. Proposals for ensuring progress in the presence of session interleaving have been put forward in [8, 29], but they require sophisticated additional machinery. The approach presented in the current paper has the advantage that there is a single global type expressing the relationships between the behaviours of all participants in a session.

To increase flexibility in the use of delegation, our calculus accommodates *connecting communications*, a notion introduced by Hu and Yoshida [19] to describe protocols with optional participants. Here we adopt the variant of this notion proposed in [7]. The intuition behind connecting communications is that in some parts of the protocol, delimited by a choice construct, some participants may be optional, namely they are "invited" to join the interaction only in some branches of the choice, by means of connecting communications. As argued in [19] and [7], this feature allows for a more natural description of typical communication protocols. In the present setting, it will also enable us to express conditional delegation: this will be obtained by writing a choice where the delegation appears only in some branches of the choice, following a connecting communication.

Outline. Section 2 provides a motivating example describing a global type with internal delegation and its end-point projections. Section 3 introduces a core process calculus with delegation actions. Section 4 introduces global types and session types reflecting our notion of internal delegation. Section 5 formalises the type system which assures lock-freedom for networks with internal delegation and connecting communications. Section 6 establishes the main properties of our typed calculus: subject reduction, session fidelity and progress. Section 7 sketches future work on multiparty compatibility and reversible sessions. Section 8 discusses related work, followed by concluding remarks in Section 9. In the Appendix we prove that projection is well defined.

2. A Motivating Example of a Global Type with Internal Delegation

Here we revisit a typical example of delegation involving three participants: Client, Seller, and Bank. In the protocol, the client and seller engage in a session in which they agree on the terms of a purchase. If the client decides to purchase, the seller delegates the processing of the client's credit card to the bank.

$$
\begin{array}{l}
\mathsf{C} \xrightarrow{\mathtt{title}} \mathsf{S};\\
\mathsf{S} \xrightarrow{\mathtt{price}} \mathsf{C};\\
\quad (\ \mathsf{C} \xrightarrow{\mathtt{ok}} \mathsf{S};\\
\qquad \mathsf{S} \xrightarrow{\underset{\leftrightarrow}{\mathtt{price}}} \mathsf{B};\\
\qquad \mathsf{S}\circ\langle\!\langle\bullet\mathsf{B};\\
\qquad \mathsf{C} \xrightarrow{\mathtt{card}} \mathsf{S};\\
\qquad \mathsf{B}\bullet\rangle\!\rangle\circ\mathsf{S};\\
\qquad \mathsf{S} \xrightarrow{\mathtt{date}} \mathsf{C}; \mathsf{End}\\
\qquad \boxplus\\
\qquad \mathsf{C} \xrightarrow{\mathtt{ko}} \mathsf{S}; \mathsf{End}\ )
\end{array}
$$

Figure 1: A global type for the client-seller-bank protocol.

A global type for such a client-seller-bank protocol is presented in Figure 1. Note sequential composition (;) binds tighter than choice ($\boxplus$). The protocol follows the following message sequence:

- Firstly, Client sends a title to Seller, indicated by action $\mathsf{C} \xrightarrow{\mathtt{title}} \mathsf{S}$;
- Seller responds by sending a price to Client, indicated by $\mathsf{S} \xrightarrow{\mathtt{price}} \mathsf{C}$;
- If the price is within Client's budget, then the following occurs:
  - Client sends a message ok to Seller;
  - Seller sends the price to Bank. The annotation on the message $\underset{\leftrightarrow}{\mathtt{price}}$ indicates that the receiver Bank connects to the session at that point.
  - After the above connecting communication, Seller **delegates** her interaction to Bank, indicated by $\mathsf{S}\circ\langle\!\langle\bullet\mathsf{B}$.

- Client sends her credit card number apparently to the untrusted party Seller but actually — thanks to delegation — to the trusted party Bank;
- Bank delegates back to Seller, indicated by $\mathtt{B}\bullet\rangle\!\rangle\circ\mathtt{S}$;
- Seller sends a `date` to Client. Since the delegation has ended, this is a communication between the actual seller and the client;

• Otherwise, Client sends message `ko` to Seller, terminating the session.

Notice, in the above protocol, it is possible that the bank is never contacted, in the case the buyer terminates the protocol by giving up the purchase. In other words, the bank is an optional participant. For this reason, it receives its first message via a connecting communication. The local view of the bank in this protocol can be described by the following session type.

$$\text{Participant Bank:}\quad \mathtt{S}?\overset{\mathtt{price}}{\leftrightarrow}\ ;\ \mathtt{S}\circ\langle\!\langle\bullet\ ;\ \mathtt{C}?\,\mathtt{card}\ ;\ \bullet\rangle\!\rangle\circ\mathtt{S}\ ;\ \mathsf{End}$$

In the above session type for the bank, the first action $\mathtt{S}?\overset{\mathtt{price}}{\leftrightarrow}$ indicates the bank is ready to receive a message containing a `price` from the seller via a connecting communication. The construct $\mathtt{S}\circ\langle\!\langle\bullet$, pronounced *passive forward delegation*, represents that the bank is delegated to act as a deputy for the seller. On behalf of the seller it receives a card number from the client, indicated by $\mathtt{C}?\,\mathtt{card}$. Finally, it returns control to the seller, as indicated by the construct $\bullet\rangle\!\rangle\circ\mathtt{S}$, pronounced *active backward delegation*.

The session type of the bank, presented above, is obtained automatically by *projection* from the global type. This projection always succeeds under some mild constraints removing ambiguity and ensuring that delegations begin and end correctly. For the other two participants — Client and Seller — we obtain the following session types.

Participant Client:	Participant Seller:
$\mathtt{S}!\,\mathtt{title}\,;$	$\mathtt{C}?\,\mathtt{title}\,;$
$\mathtt{S}?\,\mathtt{price}\,;$	$\mathtt{C}!\,\mathtt{price}\,;$
$(\ \ \mathtt{S}!\,\mathtt{ok}\,;$	$(\ \ \mathtt{C}?\,\mathtt{ok}\,;$
$\mathtt{S}!\,\mathtt{card}\,;$	$\mathtt{B}!\overset{\mathtt{price}}{\leftrightarrow}\,;$
$\mathtt{S}?\,\mathtt{date}\,;\mathsf{End}$	$\circ\langle\!\langle\bullet\mathtt{B};\mathtt{B}\bullet\rangle\!\rangle\circ;$
$\vee$	$\mathtt{C}!\,\mathtt{date}\,;\mathsf{End}$
$\mathtt{S}!\,\mathtt{ko}\,;\mathsf{End}\ \ )$	$\wedge$
	$\mathtt{C}?\,\mathtt{ko}\,;\mathsf{End}\ \ )$

In the above two types, observe that the client always thinks she is interacting with the seller. Also observe that the seller does not perform any actions between delegating her behaviour to the bank (indicated by $\circ\langle\!\langle\bullet\mathtt{B}$, pronounced *active forward delegation*) and receiving it back (indicated by $\mathtt{B}\bullet\rangle\!\rangle\circ$, pronounced *passive backward delegation*).

As standard for session types, since the client decides whether to purchase or not by sending the message `ok` or `ko` respectively, the branches for the client are composed using the union operator ($\vee$). The union indicates that at compile-time we do not know what decision the client will make. In contrast, we know that the seller will react in response to the decision the client makes — by either receiving `ok` or `ko` — and hence the branches of the seller are composed using intersection ($\wedge$).

2.1. Comparison to the approach with session interleaving

In previous work [17], a quite different approach to global types involving delegation is employed. To model the scenario defined above, two global types are employed. The first describes the session involving interactions between the client and the seller. The second describes the session between the seller and the bank, in which some actions of the seller from the former session are delegated to the bank. As discussed in Section 1, this delegation is implemented via channel passing and it appears in the global type of the

G_{cs}

$$\mathtt{C} \xrightarrow{\mathtt{title}} \mathtt{S};$$
$$\mathtt{S} \xrightarrow{\mathtt{price}} \mathtt{C};$$
$$(\,(\,\mathtt{C} \xrightarrow{\mathtt{ok}} \mathtt{S};$$
$$\mathtt{C} \xrightarrow{\mathtt{card}} \mathtt{S};$$
$$\mathtt{S} \xrightarrow{\mathtt{date}} \mathtt{C}\,)$$
$$\boxplus$$
$$\mathtt{C} \xrightarrow{\mathtt{ko}} \mathtt{S};\mathsf{End}$$
$$)$$

G_{sb}

$$\mathtt{S} \xrightarrow{\mathtt{price}} \mathtt{B};$$
$$\mathtt{S} \xrightarrow{\mathsf{T}} \mathtt{B};$$
$$\mathtt{B} \xrightarrow{\mathsf{T}'} \mathtt{S};\mathsf{End}$$

$\mathsf{T} = \mathtt{C}?\,\mathtt{card}\,;\mathsf{T}'$ $\qquad$ $\mathsf{T}' = \mathtt{C}!\,\mathtt{date}\,;\mathsf{End}$

Figure 2: Two global types for the client-seller-bank protocol in the channel-based approach.

deputy's session as a communication labelled by a channel type. The two separate types are presented in Figure 2.

All communications in Figure 1 are also accounted for in Figure 2. However, there is information missing in Figure 2, regarding how the communications in the two global types are related to each other. Moreover, it is not clear whether the delegation in the first session is enforced, since G_{cs} describes a binary session that can complete independently of G_{sb}.

To infer the connection between the two global types in Figure 2 it is required to observe that the transmitted type $\mathtt{C}?\,\mathtt{card}\,;\mathtt{C}!\,\mathtt{date}\,;\mathsf{End}$ is the type of the seller in G_{cs} after having received the message `ok`. This however is still ambiguous. For example, we may ask whether the message `price` is sent from the seller to the bank before or after receiving the message `ok`. In order to guarantee progress for the bank once it engages in the session the seller must send the `price` after `ok` is received from the client. Unfortunately, this is not immediately obvious from the types in Figure 2. In contrast, there is no such ambiguity in interpreting the global type in Figure 1.

3. A Core Calculus Modelling Processes with Internal Delegation

We begin the formal development by introducing untyped networks of processes. 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; and *session participants*, ranged over by $\mathsf{p}, \mathsf{q}, \mathsf{r}$ and forming the set Part. We use Λ to range over both messages and connecting messages.

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. An input action $\mathsf{p}?\Lambda$ whose message Λ is connecting is called a *connecting input* (and similarly for output actions). As in [19, 7], 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 example of Section 2.

In addition to exchanging messages, processes may perform delegation actions. Delegation involves two participants: the *principal*, say p in what follows, and the *deputy*, say q. The deputy q acts on behalf of the principal p in the piece of code delimited by two matching delegation actions $\mathsf{p}\circ\langle\!\langle\bullet$ and $\bullet\rangle\!\rangle\circ\mathsf{p}$. Dually, the principal p suspends execution between two matching delegation actions $\circ\langle\!\langle\bullet\mathsf{q}$ and $\mathsf{q}\bullet\rangle\!\rangle\circ$.

Definition 3.1 (Processes). Processes *are defined by:*

$$P \quad ::= \quad \Sigma_{i\in I}\pi_i;P_i \;\mid\; \oplus_{i\in I}\,\pi_i;P_i \;\mid\; \mathsf{p}\circ\langle\!\langle\bullet;P \;\mid\; \bullet\rangle\!\rangle\circ\mathsf{p};P$$
$$\mid\; \circ\langle\!\langle\bullet\mathsf{q};P \;\mid\; \mathsf{q}\bullet\rangle\!\rangle\circ;P \;\mid\; \mu X.P \;\mid\; X \;\mid\; \mathbf{0}$$

External choice ($\sum$) and internal choice ($\oplus$) are assumed to be associative, commutative, and non-empty (except when combined with binary choices). A process prefixed by an atomic action 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 [30] (Chapter 21). We omit choice symbols in one-branch choices, and trailing $\mathbf{0}$ processes.

The prefixes $\mathsf{p}\circ\langle\!\langle\bullet,\bullet\rangle\!\rangle\circ\mathsf{p}, \circ\langle\!\langle\bullet\mathsf{q},\mathsf{q}\bullet\rangle\!\rangle\circ$ are used for delegation and should be read as follows: the open/closed parentheses mean forward/backward delegation, while the white/black circle next to the participant indicates whether she is the principal/deputy in the delegation (as a rule of thumb, the colour of the principal is white since she plays first, as in a chess game).

The process $\circ\langle\!\langle\bullet\mathsf{q}; P$ is *active forward delegation*: it represents the behaviour of a principal, which delegates deputy q to act on her behalf, and freezes her continuation P until q returns the delegation. The process $\mathsf{q}\bullet\rangle\!\rangle\circ; P$ is *passive backward delegation*: it represents the behaviour of a principal which gets back the delegation from the deputy q and resumes executing its continuation P. An active forward delegation and a passive backward delegation appearing together in the code for p, $\circ\langle\!\langle\bullet\mathsf{q};\mathsf{q}\bullet\rangle\!\rangle\circ; P$, indicate that p is inactive while her behaviour is delegated to q, as in participant Seller in Section 2.

Dually, the process $\mathsf{p}\circ\langle\!\langle\bullet; P$ is *passive forward delegation*: it represents the behaviour of the deputy, which will execute the continuation P as if it were the principal p until the end of the delegation. The process $\bullet\rangle\!\rangle\circ\mathsf{p}; P$ is *active backward delegation*: it represents the behaviour of a deputy which returns the delegation to the principal p and resumes executing P under her own identity. For example, if a participant q is modelled as process $\mathsf{r}?\lambda_1; \mathsf{p}\circ\langle\!\langle\bullet; \mathsf{r}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{r}?\lambda_3$, then messages λ_1 and λ_3 at the beginning and end are received under the participant's own identity q; however, for sending message λ_2, the participant acts as a deputy for p. See for example participant Bank in Section 2.

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

Networks are comprised of pairs $\mathsf{p}[\![\, P \,]\!]$ or $\overset{*}{\mathsf{p}}[\![\, P \,]\!]$ composed in parallel. The pair $\mathsf{p}[\![\, P \,]\!]$ indicates participant p is *active* and has behaviour P; while in $\overset{*}{\mathsf{p}}[\![\, P \,]\!]$ the participant p is *frozen* and is ready to resume P after unfreezing.

Definition 3.2 (Networks). Networks *are defined by:*

$$\mathbb{N} \quad ::= \quad \mathsf{p}[\![\, P \,]\!] \quad | \quad \overset{*}{\mathsf{p}}[\![\, P \,]\!] \quad | \quad \mathbb{N} \parallel \mathbb{N}$$

The operator $\parallel$ is associative and commutative, with neutral elements $\mathsf{p}[\![\, \mathbf{0} \,]\!]$ for each p. These laws give rise to the structural equivalence on networks.

To express the operational semantics, it is convenient to record on labels the overall behaviour when a send and receive interact. For this purpose, we define *atomic communications*, with sender p, ranging over $\alpha^{\mathsf{p}} \in \{\mathsf{p}\Lambda\mathsf{q} \mid \mathsf{q} \in \mathsf{Part}, \Lambda \in \mathsf{Msg} \cup \mathsf{CMsg}\}$. The communication $\mathsf{p}\lambda\mathsf{q}$ is *simple*, while the communication $\mathsf{p} \overset{\lambda}{\leftrightarrow} \mathsf{q}$ is *connecting*. We write α instead of α^{p} when the sender is not relevant. Given a communication $\alpha = \mathsf{p}\Lambda\mathsf{q}$, we define the sender and receiver of α as $\mathsf{sender}(\alpha) = \mathsf{p}$ and $\mathsf{rec}(\alpha) = \mathsf{q}$. Moreover, we denote by $\mathsf{part}(\alpha) = \{\mathsf{rec}(\alpha), \mathsf{sender}(\alpha)\}$ the participants of the communication α.

We also wish to record on labels when a principal and deputy pair cooperate during delegation. We let *atomic interactions* range over $\phi \in \{\mathsf{p}\Lambda\mathsf{q}, \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}, \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p} \mid \mathsf{p},\mathsf{q} \in \mathsf{Part}, \Lambda \in \mathsf{Msg} \cup \mathsf{CMsg}\}$. An atomic interaction is either an atomic communication from a participant p to a participant q, or the start of a delegation from a principal p to a deputy q, or the end of a delegation from a deputy q to a principal p. In the delegation opening and closing, the participant to the left is the one which triggers the interaction, while the white and black colours denote respectively the principal and the deputy.

The operational semantics is given by two labelled transition systems, one for processes and one for networks. The transitions for processes are labelled by atomic actions, while those for networks are labelled by atomic interactions. As a convention, when we want to discuss sequences of multiple interactions, we write $\mathbb{N} \xrightarrow{\phi_1\cdots\phi_n} \mathbb{N}'$ for the transitive closure of $\xrightarrow{\phi}$. Formally, for $n \geq 1$ we let $\mathbb{N} \xrightarrow{\phi_1\cdots\phi_n} \mathbb{N}'$ if there are $\mathbb{N}_i$, $0 \leq i \leq n$, such that $\mathbb{N}_0 = \mathbb{N}$ and $\mathbb{N}_n = \mathbb{N}'$ and $\mathbb{N}_i \xrightarrow{\phi_{i+1}} \mathbb{N}_{i+1}$ for $0 \leq i < n$.

$$\Sigma_{i\in I}\pi_i; P_i \xrightarrow{\pi_j} P_j \quad j \in I \quad [\textsc{ExtCh}] \qquad \oplus_{i\in I}\pi_i; P_i \xrightarrow{\pi_j} P_j \quad j \in I \quad [\textsc{IntCh}]$$

$$\frac{P \xrightarrow{\mathsf{q}!\Lambda} P' \qquad Q \xrightarrow{\mathsf{p}?\Lambda} Q'}{\mathsf{p}\llbracket\, P \,\rrbracket \parallel \mathsf{q}\llbracket\, Q \,\rrbracket \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathsf{p}\llbracket\, P' \,\rrbracket \parallel \mathsf{q}\llbracket\, Q' \,\rrbracket}\ [\textsc{Com}]$$

$$\mathsf{p}\llbracket\, \circ\langle\!\langle\bullet\mathsf{q}; P \,\rrbracket \parallel \mathsf{q}\llbracket\, \mathsf{p}\circ\langle\!\langle\bullet; Q \,\rrbracket \xrightarrow{\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}} \overset{*}{\mathsf{p}}\llbracket\, P \,\rrbracket \parallel \mathsf{p}\llbracket\, Q \,\rrbracket \quad [\text{BD}\textsc{el}]$$

$$\overset{*}{\mathsf{p}}\llbracket\, \mathsf{q}\bullet\rangle\!\rangle\circ; P \,\rrbracket \parallel \mathsf{p}\llbracket\, \bullet\rangle\!\rangle\circ\mathsf{p}; Q \,\rrbracket \xrightarrow{\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathsf{p}\llbracket\, P \,\rrbracket \parallel \mathsf{q}\llbracket\, Q \,\rrbracket \quad [\text{ED}\textsc{el}]$$

$$\frac{\mathbb{N} \xrightarrow{\phi} \mathbb{N}'}{\mathbb{N} \parallel \mathbb{N}'' \xrightarrow{\phi} \mathbb{N}' \parallel \mathbb{N}''}\ [\textsc{Ct}]$$

Figure 3: LTS for processes and networks.

The LTSs for processes and networks are given in Figure 3. Rules [ExtCh] and [IntCh] allow an action to be extracted from one of the summands, as usual. When the delegation from p to q begins, p becomes frozen and q becomes p. When the delegation from p to q ends, the active p becomes q again and the frozen p becomes active.

4. Global Types and Session Types

The untyped networks in the previous section offer few guarantees regarding their behaviours. It is trivial to construct networks where progress is not guaranteed, or which exhibit wild delegation behaviours, where we lose track of which participants are the principals and which are the deputies. Such problems are addressed by the types in this section, which constrain networks.

4.1. Unambiguous global types and session pre-types

A *multiparty session* is a series of atomic interactions among participants [17, 18], which follows a predefined protocol specified by a *global type*. Global types are built from choices among atomic communications with the same sender (*the choice leader*) and forward/backward delegations, possibly using recursion.

Definition 4.1 (Global types). Global types G *are defined by:*

$$\mathsf{G} \quad ::= \quad \boxplus_{i\in I}\, \alpha_i^{\mathsf{p}}; \mathsf{G}_i \ \mid\ \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}; \mathsf{G} \ \mid\ \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{G} \ \mid\ \mu\mathbf{t}.\mathsf{G} \ \mid\ \mathbf{t} \ \mid\ \mathsf{End}$$

Sequential composition (;) has higher precedence than choice ($\boxplus$). Recursion must be guarded and it is treated equi-recursively.

In the figures of Section 2 we used a more verbose syntax for global types, rendering $\mathsf{p}\Lambda\mathsf{q}$ as $\mathsf{p} \xrightarrow{\Lambda} \mathsf{q}$, as this seemed easier to understand in an informal description of the protocol.

To avoid non-determinism, we require all atomic communications in choices to be different, a condition formalised by the following definition.

Definition 4.2 (Non-ambiguous global type). *A choice $\boxplus_{i\in I}\, \alpha_i^{\mathsf{p}}; \mathsf{G}_i$ is ambiguous if $\alpha_h^{\mathsf{p}} = \alpha_k^{\mathsf{p}}$ for some $h, k \in I$. A global type is* non-ambiguous *if it does not contain ambiguous choices.*

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 $\mathbf{0}$ with End, with similar conventions.

$\mathsf{part}(\boxplus_{i\in I}\,\alpha_i;\mathsf{G}_i) = \bigcup_{i\in I}(\mathsf{part}(\alpha_i)\cup\mathsf{part}(\mathsf{G}_i))$
$\mathsf{part}(\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q};\mathsf{G}) = \mathsf{part}(\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{G}) = \mathsf{part}(\mathsf{G})\cup\{\mathsf{p},\mathsf{q}\}$
$\mathsf{part}(\mu\mathbf{t}.\mathsf{G}) = \mathsf{part}(\mathsf{G})$
$\mathsf{part}(\mathbf{t}) = \mathsf{part}(\mathsf{End}) = \emptyset$

Figure 4: Participants of global types.

Definition 4.3 (Session pre-types). Session pre-types *are defined by:*

$$\mathsf{T} \quad ::= \quad \bigwedge_{i\in I}\pi_i;\mathsf{T}_i \mid \bigvee_{i\in I}\pi_i;\mathsf{T}_i \mid \mathsf{p}\circ\langle\!\langle\bullet;\mathsf{T} \mid \bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{T} \mid \circ\langle\!\langle\bullet\mathsf{q};\mathsf{T} \mid \mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T} \mid \mu\mathbf{t}.\mathsf{T} \mid \mathbf{t} \mid \mathsf{End}$$

As with global types and processes, session type recursion must be guarded and treated equi-recursively.

4.2. Projection of a global type onto participants

Session types are defined as projections of unambiguous global types onto their participants (when such a projection is well defined), where *the set of participants of* G, denoted part(G), is defined in Figure 4. Each session type represents the contribution of an individual participant to the session. The projection of a choice yields a union for the choice leader and intersections for all other participants.

The projection of global types is defined in terms of two projection operators, the *direct projection* "$\restriction$", which projects a global type on participants when no delegation is underway, and the *delegation projection* "$\restriction_i$" with $i = 1, 2$, which projects a global type after a delegation has started. The latter is parameterised by a pair of participants: the pair identifies the principal and the deputy of the delegation, while the subscript 1 or 2 identifies the participant on which the projection is performed. The delegation projection can only be performed on the two participants involved in the delegation. Both projection operators are partial and they have higher precedence than ";".

Projection employs a partial operator $\sqcap$, ensuring that, in a projection from a global choice, the choice leader makes the decision and all the other participants act accordingly. This is achieved 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* in the sense that the following partial operator, *the meet of a set of types* $\bigsqcap_{i\in I}\mathsf{T}_i$, is defined.

Definition 4.4 (Non-ambiguous input intersection). *An intersection of inputs* $\bigwedge_{i\in I}\mathsf{p}_i?\Lambda_i;\mathsf{T}_i$ *is* non-ambiguous *if its initial inputs are either all simple inputs or all connecting inputs, and moreover all such inputs are distinct, namely* $h, k \in I$ *and* $h \neq k$ *imply either* $\mathsf{p}_h \neq \mathsf{p}_k$ *or* $\Lambda_h \neq \Lambda_k$.

Definition 4.5 (Meet of types). *To build the meet of intersection types, we start by defining the meet between a non-ambiguous input intersection and an input type or an* End *type, as well as the meet between two* End *types:*

1. $(\bigwedge_{i\in I}\mathsf{p}_i?\Lambda_i;\mathsf{T}_i)\sqcap\mathsf{p}?\Lambda;\mathsf{T} = \bigwedge_{i\in I}\mathsf{p}_i?\Lambda_i;\mathsf{T}_i \,\wedge\, \mathsf{p}?\Lambda;\mathsf{T}$
   if the resulting intersection is not ambiguous
2. $(\bigwedge_{i\in I}\mathsf{p}_i?\Lambda_i;\mathsf{T}_i)\sqcap\mathsf{p}?\Lambda;\mathsf{T} = \bigwedge_{i\in I}\mathsf{p}_i?\Lambda_i;\mathsf{T}_i$
   if $\mathsf{p} = \mathsf{p}_j$ *and* $\Lambda = \Lambda_j$ *and* $\mathsf{T} = \mathsf{T}_j$ *for some* $j \in I$
3. $(\bigwedge_{i\in I}\mathsf{p}_i?\overset{\lambda_i}{\leftrightarrow};\mathsf{T}_i)\sqcap\mathsf{End} = \bigwedge_{i\in I}\mathsf{p}_i?\overset{\lambda_i}{\leftrightarrow};\mathsf{T}_i$
4. $\mathsf{End}\sqcap\mathsf{End} = \mathsf{End}$

To define the meet of two intersection types, we simply iterate the above definition on the members of one of the intersections. Thereby, meet can be extended to a set of types.

The above meet $\bigsqcap_{i\in I}\mathsf{T}_i$ checks that the T_i's are consistent 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. Note we do not need to define the meet when the arguments are recursive types, since we consider recursion equi-recursively. In other cases, meet is not defined, i.e. when unions or delegations are involved.

$$(\mathsf{p} \Lambda \mathsf{q}; \mathsf{G}) \upharpoonright \mathsf{r} = \begin{cases} \mathsf{q}!\Lambda; \mathsf{G} \upharpoonright \mathsf{p} & \text{if } \mathsf{r} = \mathsf{p} \\ \mathsf{p}?\Lambda; \mathsf{G} \upharpoonright \mathsf{q} & \text{if } \mathsf{r} = \mathsf{q} \\ \mathsf{G} \upharpoonright \mathsf{r} & \text{if } \mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\} \end{cases}$$

$$(\boxplus_{i \in I} \alpha_i^{\mathsf{p}}; \mathsf{G}_i) \upharpoonright \mathsf{r} = \begin{cases} \bigvee_{i \in I} (\alpha_i^{\mathsf{p}}; \mathsf{G}_i) \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{p} \\ \bigsqcap_{i \in I} (\alpha_i^{\mathsf{p}}; \mathsf{G}_i) \upharpoonright \mathsf{r} & \text{otherwise} \end{cases} \qquad \text{where } |I| > 1$$

$$(\mu \mathbf{t}.\mathsf{G}) \upharpoonright \mathsf{p} = \begin{cases} \mathsf{G} \upharpoonright \mathsf{p} & \text{if } \mathbf{t} \text{ does not occur in } \mathsf{G} \\ \mu \mathbf{t}.\mathsf{G} \upharpoonright \mathsf{p} & \text{if } \mathsf{p} \in \mathsf{part}(\mathsf{G}) \\ \mathsf{End} & \text{otherwise} \end{cases} \qquad \mathbf{t} \upharpoonright \mathsf{p} = \mathbf{t} \qquad \mathsf{End} \upharpoonright \mathsf{p} = \mathsf{End}$$

$$(\mathsf{p} \circ\langle\!\langle \bullet \mathsf{q}; \mathsf{G}) \upharpoonright \mathsf{r} = \begin{cases} \circ\langle\!\langle \bullet\, \mathsf{q}; \mathsf{G} \upharpoonright_1 (\mathsf{p}, \mathsf{q}) & \text{if } \mathsf{r} = \mathsf{p} \\ \mathsf{p} \circ\langle\!\langle \bullet; \mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q}) & \text{if } \mathsf{r} = \mathsf{q} \\ \mathsf{G} \upharpoonright \mathsf{r} & \text{otherwise} \end{cases}$$

$$(\mathsf{q} \bullet\rangle\!\rangle \circ \mathsf{p}; \mathsf{G}) \upharpoonright \mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r} \qquad \text{if } \mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$$

Figure 5: Direct projection of global types onto participants.

$$(\mathsf{r} \Lambda \mathsf{s}; \mathsf{G}) \upharpoonright_2 (\mathsf{p}, \mathsf{q}) = \begin{cases} \mathsf{s}!\Lambda; \mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q}) & \text{if } \mathsf{r} = \mathsf{p} \text{ and } \mathsf{s} \neq \mathsf{q} \\ \mathsf{r}?\Lambda; \mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q}) & \text{if } \mathsf{s} = \mathsf{p} \text{ and } \mathsf{r} \neq \mathsf{q} \\ \mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q}) & \text{if } \{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset \end{cases}$$

$$(\mathsf{r} \Lambda \mathsf{s}; \mathsf{G}) \upharpoonright_1 (\mathsf{p}, \mathsf{q}) = \mathsf{G} \upharpoonright_1 (\mathsf{p}, \mathsf{q}) \ \text{ if } \mathsf{r} \neq \mathsf{q} \text{ and } \mathsf{s} \neq \mathsf{q}$$

$$(\mathsf{q} \bullet\rangle\!\rangle \circ \mathsf{p}; \mathsf{G}) \upharpoonright_1 (\mathsf{p}, \mathsf{q}) = \mathsf{q} \bullet\rangle\!\rangle \circ; \mathsf{G} \upharpoonright \mathsf{p} \qquad (\mathsf{q} \bullet\rangle\!\rangle \circ \mathsf{p}; \mathsf{G}) \upharpoonright_2 (\mathsf{p}, \mathsf{q}) = \bullet\rangle\!\rangle \circ \mathsf{p}; \mathsf{G} \upharpoonright \mathsf{q}$$

$$(\mathsf{r} \circ\langle\!\langle \bullet \mathsf{s}; \mathsf{G}) \upharpoonright_1 (\mathsf{p}, \mathsf{q}) = (\mathsf{r} \bullet\rangle\!\rangle \circ \mathsf{s}; \mathsf{G}) \upharpoonright_1 (\mathsf{p}, \mathsf{q}) = \mathsf{G} \upharpoonright_1 (\mathsf{p}, \mathsf{q}) \ \text{ if } \{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$$

$$(\mathsf{r} \circ\langle\!\langle \bullet \mathsf{s}; \mathsf{G}) \upharpoonright_2 (\mathsf{p}, \mathsf{q}) = (\mathsf{r} \bullet\rangle\!\rangle \circ \mathsf{s}; \mathsf{G}) \upharpoonright_2 (\mathsf{p}, \mathsf{q}) = \mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q}) \ \text{ if } \{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$$

Figure 6: Delegation projection of global types onto participants.

The direct projection, see Figure 5 where $|I|$ denotes the cardinality of I, can be defined for all kinds of global types. The projection of a choice (with more than one branch) is a union of outputs for the choice leader, and it is otherwise computed as the meet of the projections of the branches. From the definition of meet we can see that the branches of a choice can differ for the presence of participants whose first communication is a connecting input. Note this definition of meet is more liberal than the standard meet typically used in the literature, since we permit a global choice with distinct receiving participants to be projected.

The projection of a forward delegation from principal p to deputy q is an active forward delegation on p and a passive forward delegation on q, followed in both cases by the delegation projection of the rest of the global type (this is where the direct projection turns into the delegation projection). In the case that the projected participant is neither the principal nor the deputy, the forward delegation is ignored. Similarly, the backward delegation from deputy q to principal p is ignored when projected on participants different from p and q. Notice that the direct projection of backward delegation is not defined for p and q, since this case is handled by delegation projection.

The delegation projection, see Figure 6, comes into play when a delegation is encountered. It is not defined for (proper) choices, nor for $\mathsf{End}, \mu \mathbf{t}.\mathsf{G}$ and $\mathbf{t}$. Therefore, the portion of computation between a

forward delegation from p to q and a backward delegation from q to p must consist of a sequence of atomic interactions. Moreover, these interactions should not involve q, since q is supposed to perform exclusively p's actions in this portion of computation. As hinted above, the delegation projections $\mathsf{G}\upharpoonright_1(\mathsf{p},\mathsf{q})$ and $\mathsf{G}\upharpoonright_2(\mathsf{p},\mathsf{q})$ represent the projections of G on p and on q respectively, within the scope of a delegation from p to q. The first two clauses of Figure 6 concern the delegation projection of atomic communications. An atomic communication involving the principal p (and not involving the deputy q, as just argued) projects to a communication action for the deputy q and to no action for the principal p. An atomic communication not involving the principal p projects to no action for either p or q.
The subsequent line defines the projection of backward delegation on one of the two involved participants. Notice that, as soon as the delegation ends, one goes back to the direct projection. The last two lines say that the projections on p and q ignore any forward/backward delegation between other participants.

Example 4.6. *Let us see our notion of projection at work on a few examples.*

1. *Let* $\mathsf{G} = \mathsf{p}\lambda_1\mathsf{q} \boxplus \mathsf{p} \overset{\lambda_2}{\leftrightarrow} \mathsf{q}$. *Then* $\mathsf{G}\upharpoonright\mathsf{q}$ *is not defined, since the intersection* $\mathsf{p}?\lambda_1 \wedge \mathsf{p}? \overset{\lambda_2}{\leftrightarrow}$ *between a simple and a connecting input is ambiguous.*

2. *Let* $\mathsf{G} = \mu\mathbf{t}.(\mathsf{p}\lambda_1\mathsf{q};\mathbf{t} \boxplus \mathsf{p}\lambda_2\mathsf{q})$ *and* $\mathsf{G}' = \mathsf{p}\lambda_1\mathsf{q};\mathsf{G} \boxplus \mathsf{p}\lambda_2\mathsf{q}$. *Then* $\mathsf{G}\upharpoonright\mathsf{q} = \mu\mathbf{t}.(\mathsf{p}?\lambda_1;\mathbf{t} \wedge \mathsf{p}?\lambda_2)$ *and* $\mathsf{G}'\upharpoonright\mathsf{q} = \mathsf{p}?\lambda_1;\mathsf{G}\upharpoonright\mathsf{q} \wedge \mathsf{p}?\lambda_2$. *Notice that* G' *is the unfolding of* G.

3. *Let* $\mathsf{G} = \mu\mathbf{t}.\mathsf{p}\lambda\mathsf{q};\mathbf{t}$ *and* $\mathsf{G}' = \mu\mathbf{t}.\mathsf{p}\lambda\mathsf{q};\mathsf{p}\lambda\mathsf{q};\mathbf{t}$. *Then* $\mathsf{G}\upharpoonright\mathsf{q} = \mu\mathbf{t}.\mathsf{p}?\lambda;\mathbf{t}$ *and* $\mathsf{G}'\upharpoonright\mathsf{q} = \mu\mathbf{t}.\mathsf{p}?\lambda;\mathsf{p}?\lambda;\mathbf{t}$. *Notice that* G *and* G' *generate the same tree.*

4. *Let* $\mathsf{G} = \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q};\mathsf{p}\lambda_1\mathsf{r};\mathsf{s}\lambda_2\mathsf{p};\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{q}\lambda_3\mathsf{p}$. *Then* $\mathsf{G}\upharpoonright\mathsf{q} = \mathsf{p}\circ\langle\!\langle\bullet;\mathsf{r}!\lambda_1;\mathsf{s}?\lambda_2;\bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{p}!\lambda_3$, *where* q *communicates with* r,s *as deputy of* p.

The soundness of the direct projection requires that global types generating the same tree are projected into session types generating the same tree. Notice that the syntactic differences between global types generating the same tree can be:

- fold/unfold of recursion, like in Example 4.6(2);
- different occurrences of μ bindings and bound variables, like in Example 4.6(3).

The projection of global types with the same tree yields session types with the same tree, since it preserves the occurrences of μ bindings and bound variables. The formal proof is given in Appendix A.

Definition 4.7 (Projectable global type). *A global type* G *is* projectable *if for all* $\mathsf{p} \in \mathsf{part}(\mathsf{G})$ *either the direct projection* $\mathsf{G}\upharpoonright\mathsf{p}$ *is defined, or there exists* $\mathsf{q} \in \mathsf{part}(\mathsf{G})$ *such that one of the two delegation projections* $\mathsf{G}\upharpoonright_1(\mathsf{p},\mathsf{q})$ *and* $\mathsf{G}\upharpoonright_2(\mathsf{q},\mathsf{p})$ *is defined.*

In the following we will consider only *non-ambiguous and projectable global types.* Projectability assures the property of well-delegation, defined as follows:

Definition 4.8 (Well-delegated global type). *A global type is* well delegated *if:*

1. *each occurrence of* $\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}$ *is followed by an occurrence of* $\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}$;
2. *no atomic interaction involving* q *occurs between* $\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}$ *and* $\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}$;
3. *no choice occurs between* $\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}$ *and* $\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}$;
4. *no delegation involving* p *occurs between* $\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}$ *and* $\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}$.

Let us briefly comment on the above definition. Condition 1 simply requires that every forward delegation be matched by the corresponding backward delegation. Conditions 2–4 impose some restrictions on the behaviour that can be delegated: while it should be clear why this behaviour should not involve the deputy q (Condition 2), the fact that the deputy is not allowed to perform choices (Condition 3) nor to perform further delegations on behalf of the principal (Condition 4) simplifies the technical treatment and may be justified by the intention of the principal to retain some control on the deputy.

Lemma 4.9. *Each projectable global type is well delegated.*

Proof Observe that the projection $\upharpoonright_i(\mathsf{p},\mathsf{q})$ with $i = 1, 2$ is only defined for:

- delegations not involving p,q;
- atomic communications not involving q;
- the backward delegation $\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}$.

Notice that well-delegation only considers delegation projection, so it does not assure projectability as shown by Example 4.6(1). Projectability does not assure non-ambiguity, since for example $\mathsf{p}\Lambda\mathsf{q} \boxplus \mathsf{p}\Lambda\mathsf{q}$ is ambiguous and projectable.

We end this section by defining session types.

Definition 4.10 (Session types). *A session pre-type* T *is a* session type *if* $\mathsf{T} = \mathsf{G} \upharpoonright \mathsf{p}$ *or* $\mathsf{T} = \mathsf{G} \upharpoonright_1(\mathsf{p},\mathsf{q})$ *or* $\mathsf{T} = \mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{q})$ *for some (non-ambiguous and projectable) global type* G *and some participants* $\mathsf{p},\mathsf{q} \in \mathsf{part}(\mathsf{G})$.

5. Guaranteeing Progress for Networks with Internal Delegation

In this section, we define our type system and state the main strong progress result that the type system guarantees. As usual, we assume an environment Γ that associates process variables with session types: $\Gamma ::= \emptyset \mid \Gamma, X : \mathsf{T}$. Then the *typing judgements* for processes and networks have the form $\Gamma \vdash P : \mathsf{T}$ and $\vdash \mathbb{N} : \mathsf{G}$, respectively.

Process typing exploits the correspondence between external choices and intersections, internal choices and unions. The typing rules for processes and networks are given in Figure 7. In rule [T-DEL], we denote by δ either one of the prefixes $\mathsf{p}\circ\langle\!\langle\bullet, \bullet\rangle\!\rangle\circ\mathsf{p}$,$\circ\langle\!\langle\bullet\mathsf{q}$ or $\mathsf{q}\bullet\rangle\!\rangle\circ$. Since $\pi_h \neq \pi_k$ with $h, k \in I$ for intersections/unions in session types, this is true also for choices in processes.
Figure 8 gives the subtyping rules, where the double line indicates that the rules are interpreted *coinductively* [30] (Chapter 21). Subtyping takes into account the rules for intersection and union. Rule [SUB-IN-SKIP] reflects the fact that unused 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 typing rules for external choice and internal choice, together with the subtyping of intersections and unions, assure that choice leaders can freely select both receivers and messages. The condition $\mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\} \cup \{\mathsf{q}_i \mid i \in I\} \cup \{\mathsf{r}_j \mid j \in J\}$ ensures all participants appear in the network. Furthermore we require all such participants in the network to be distinct to avoid multiple processes assuming the same role in a network (this is essential for linearity of sessions). Note rule [T-NET] permits additional participants that do not appear in the global type, allowing the typing of sessions containing $\mathsf{p}[\![\,\mathbf{0}\,]\!]$ for fresh p — a property required to guarantee invariance of types under structural equivalence of networks. The rule [T-NET] assures that:

- the principals partaking in a delegation are frozen and their first interaction is the end of the delegation;
- the deputies partaking in a delegation have the behaviours required by the global type for the corresponding principals (recall that, in the labelled transition system, the deputy takes on the name of the principal);

$$\frac{\Gamma \vdash P_i : \mathsf{T}_i \ (i \in I)}{\Gamma \vdash \Sigma_{i\in I}\pi_i; P_i : \bigwedge_{i\in I} \pi_i; \mathsf{T}_i} \ [\textsc{t-ExtCh}] \qquad \frac{\Gamma \vdash P_i : \mathsf{T}_i \ (i \in I)}{\Gamma \vdash \oplus_{i\in I}\pi_i; P_i : \bigvee_{i\in I} \pi_i; \mathsf{T}_i} \ [\textsc{t-IntCh}]$$

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

$$\frac{\Gamma \vdash P : \mathsf{T}}{\Gamma \vdash \delta; P : \delta; \mathsf{T}} \ [\textsc{t-Del}] \qquad \Gamma \vdash \mathbf{0} : \mathsf{End} \ [\textsc{t-}\mathbf{0}]$$

$$\frac{\begin{array}{lll} \vdash P_i : \mathsf{q}_i\bullet\rangle\!\rangle\circ; \mathsf{T}_i & \mathsf{q}_i\bullet\rangle\!\rangle\circ; \mathsf{T}_i \leq \mathsf{G}\restriction_1(\mathsf{p}_i, \mathsf{q}_i) & (i \in I) \\ \vdash Q_i : \mathsf{S}_i & \mathsf{S}_i \leq \mathsf{G}\restriction_2(\mathsf{p}_i, \mathsf{q}_i) & (i \in I) \\ \vdash R_j : \mathsf{U}_j & \mathsf{U}_j \leq \mathsf{G}\restriction \mathsf{r}_j & (j \in J) \\ \multicolumn{2}{l}{\mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\} \cup \{\mathsf{q}_i \mid i \in I\} \cup \{\mathsf{r}_j \mid j \in J\}} & \text{all participants distinct} \end{array}}{\vdash \Pi_{i\in I}\overset{*}{\mathsf{p}}_i[\![P_i]\!] \parallel \Pi_{i\in I}\, \mathsf{p}_i[\![Q_i]\!] \parallel \Pi_{j\in J}\, \mathsf{r}_j[\![R_j]\!] : \mathsf{G}} \ [\textsc{t-Net}]$$

Figure 7: Typing rules for processes and networks.

$$[\textsc{Sub-In}] \ \frac{\forall i \in I : \mathsf{T}_i \leq \mathsf{T}'_i}{\bigwedge_{i\in I\cup I'} \pi_i; \mathsf{T}_i \leq \bigwedge_{i\in I} \pi_i; \mathsf{T}'_i} \qquad [\textsc{Sub-Out}] \ \frac{\forall i \in I : \mathsf{T}_i \leq \mathsf{T}'_i}{\bigvee_{i\in I} \pi_i; \mathsf{T}_i \leq \bigvee_{i\in I\cup I'} \pi_i; \mathsf{T}'_i}$$

$$[\textsc{Sub-In-Skip}] \ \frac{\forall i \in I : \pi_i \text{ is a connecting input}}{\bigwedge_{i\in I} \pi_i; \mathsf{T}_i \leq \mathsf{End}} \qquad [\textsc{Sub-Del}] \ \frac{\mathsf{T} \leq \mathsf{T}'}{\delta; \mathsf{T} \leq \delta; \mathsf{T}'} \qquad [\textsc{Sub-Skip}] \ \mathsf{End} \leq \mathsf{End}$$

Figure 8: Subtyping rules.

- all other participants follow the prescription of the global type.

Example 5.1. *The following are examples of typed networks with different delegation patterns. The first does not require connecting inputs but the third does; although, the only difference is to whom a message is sent during the delegated part of the session.*

1. *Global type* $\mathsf{G} = \mathsf{p}\lambda_1\mathsf{q}; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}; \mathsf{p}\lambda_2\mathsf{r}; \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p} \boxplus \mathsf{p}\lambda_1\mathsf{r}; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{r}; \mathsf{p}\lambda_2\mathsf{q}; \mathsf{r}\bullet\rangle\!\rangle\circ\mathsf{p}$ *can be used to type the following network.*

$$\begin{array}{ll} & \mathsf{p}[\![\mathsf{q}!\lambda_1; \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{q}\bullet\rangle\!\rangle\circ \oplus \mathsf{r}!\lambda_1; \circ\langle\!\langle\bullet\mathsf{r}; \mathsf{r}\bullet\rangle\!\rangle\circ]\!] \\ \parallel & \mathsf{q}[\![\mathsf{p}?\lambda_1; \mathsf{p}\circ\langle\!\langle\bullet; \mathsf{r}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p} + \mathsf{p}?\lambda_2]\!] \\ \parallel & \mathsf{r}[\![\mathsf{p}?\lambda_1; \mathsf{p}\circ\langle\!\langle\bullet; \mathsf{q}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p} + \mathsf{p}?\lambda_2]\!] \end{array}$$

   Notice in the above network, one of the branches of each participant will trigger in every execution. We call this a balanced session since the set of active participants is the same regardless of the choices made.

2. *The type* G *in the example above can be used to type also the above network modified such that participant* p *has process* $P = \mathsf{q}!\lambda_1; \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{q}\bullet\rangle\!\rangle\circ$ *since* $\vdash P : \mathsf{T}$, *where* $\mathsf{T} = \mathsf{q}!\lambda_1; \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{q}\bullet\rangle\!\rangle\circ \leq \mathsf{G}\restriction\mathsf{p} = \mathsf{q}!\lambda_1; \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{q}\bullet\rangle\!\rangle\circ \vee \mathsf{r}!\lambda_1; \circ\langle\!\langle\bullet\mathsf{r}; \mathsf{r}\bullet\rangle\!\rangle\circ$.

3. *In contrast to the above, connecting inputs are necessary for the existence of a network with type* $\mathsf{G} = \mathsf{p} \overset{\lambda_1}{\leftrightarrow} \mathsf{q}; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}; \mathsf{p}\lambda_2\mathsf{s}; \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p} \boxplus \mathsf{p} \overset{\lambda_1}{\leftrightarrow} \mathsf{r}; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{r}; \mathsf{p}\lambda_2\mathsf{s}; \mathsf{r}\bullet\rangle\!\rangle\circ\mathsf{p}$. *Observe the use of connecting inputs, which allow one of*

the roles q *or* r *to hang forever, without causing deadlock. Thus the following network is typable.*

$$\begin{aligned} &\mathsf{p}[\![\, \mathsf{q}!\overset{\lambda_1}{\leftrightarrow}; \circ\langle\!\langle \bullet\mathsf{q}; \mathsf{q}\bullet\rangle\!\rangle\circ \oplus \mathsf{r}!\overset{\lambda_1}{\leftrightarrow}; \circ\langle\!\langle \bullet\mathsf{r}; \mathsf{r}\bullet\rangle\!\rangle\circ \,]\!] \parallel \mathsf{s}[\![\, \mathsf{p}?\lambda_2 \,]\!] \\ \parallel\ &\mathsf{q}[\![\, \mathsf{p}?\overset{\lambda_1}{\leftrightarrow}; \mathsf{p}\circ\langle\!\langle \bullet; \mathsf{s}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p} \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\overset{\lambda_1}{\leftrightarrow}; \mathsf{p}\circ\langle\!\langle \bullet; \mathsf{s}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p} \,]\!] \end{aligned}$$

When typing the above observe $\mathsf{G} \upharpoonright \mathsf{q} = \mathsf{p}?\overset{\lambda_1}{\leftrightarrow}; \mathsf{p}\circ\langle\!\langle \bullet\mathsf{s}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p} \sqcap \mathsf{End}$*, which, by Definition 4.5 clause 3, is* $\mathsf{p}?\overset{\lambda_1}{\leftrightarrow}; \mathsf{p}\circ\langle\!\langle \bullet\mathsf{s}!\lambda_2; \bullet\rangle\!\rangle\circ\mathsf{p}$.

4. *A single session may delegate to a participant infinitely often, as long as each delegation is opened and closed within each unfolding of a recursion. For example* $\mathsf{G} = \mu\mathbf{t}.(\mathsf{p}\circ\langle\!\langle \bullet\mathsf{q}; \mathsf{r}\lambda\mathsf{p}; \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}; \mathbf{t})$ *is a type for the following network.*

$$\mathsf{p}[\![\, \mu X.\circ\langle\!\langle \bullet\mathsf{q}; \mathsf{q}\bullet\rangle\!\rangle\circ; X \,]\!] \parallel \mathsf{q}[\![\, \mu Y.\mathsf{p}\circ\langle\!\langle \bullet; \mathsf{r}?\lambda; \bullet\rangle\!\rangle\circ\mathsf{p}; Y \,]\!] \parallel \mathsf{r}[\![\, \mu Z.\mathsf{p}?\lambda; Z \,]\!]$$

5.1. Main result: strong progress

We are now in a position to state the main result of the paper. Well-typed networks are guaranteed a strong notion of progress, where there are no hanging actions — inputs, outputs, or delegations — except when an input is connecting. Specifically we have the following:

- Every participant offering a choice of outputs will eventually be able to choose to perform **any** of those outputs;
- Every participant offering a choice of simple inputs will eventually be asked to perform **one** of those inputs;
- Every principal finds the deputy and will get back the delegation;
- Every deputy finds the principal and will return back the delegation.

The progress theorem can be stated as follows.

Theorem 5.2 (Progress). *The following hold for well-typed networks:*

1. *If* $\mathbb{N} = \mathsf{p}[\![\, \oplus_{i\in I}\mathsf{q}_i!\Lambda_i; P_i \,]\!] \parallel \mathbb{N}_0$*, then* $\mathbb{N} \xrightarrow{\vec{\phi}\ \mathsf{p}\Lambda_i\mathsf{q}_i} \mathbb{N}'$ *for some* $\vec{\phi}$ *and for all* $i \in I$.
2. *If* $\mathbb{N} = \mathsf{p}[\![\, \Sigma_{i\in I}\mathsf{q}_i?\lambda_i; P_i \,]\!] \parallel \mathbb{N}_0$*, then* $\mathbb{N} \xrightarrow{\vec{\phi}\ \mathsf{q}_i\lambda_i\mathsf{p}} \mathbb{N}'$ *for some* $\vec{\phi}$ *and for some* $i \in I$.
3. *If* $\mathbb{N} = \mathsf{p}[\![\, \circ\langle\!\langle \bullet\mathsf{q}; P \,]\!] \parallel \mathbb{N}_0$*, then* $\mathbb{N} \xrightarrow{\vec{\phi}\ \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}\,\vec{\phi}'\,\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$ *for some* $\vec{\phi}$ *and* $\vec{\phi}'$.
4. *If* $\mathbb{N} = \mathsf{q}[\![\, \mathsf{p}\circ\langle\!\langle \bullet; Q \,]\!] \parallel \mathbb{N}_0$*, then* $\mathbb{N} \xrightarrow{\vec{\phi}\ \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}\,\vec{\phi}'\,\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$ *for some* $\vec{\phi}$ *and* $\vec{\phi}'$.

Notice, in the above theorem, the labels are sequences of atomic interactions, each of which results either from a pair of matching send and receive actions, or from a pair of matching delegation actions. The first two clauses in the theorem above state that any participant in a typed network with a choice of actions available will eventually perform one of those actions as a communication with another party. Notice that in the internal choice of outputs in Clause 1 above, the choice leader can freely choose any of its branches; as formalised by the universal quantification over branches. In contrast, in the external choice of inputs in Clause 2 above, the receiver responds to the decision made by the choice leader, hence cannot freely choose which branch is taken; as formalised by the existential quantification over branches. These guarantees distinguishing internal and external choice are enforced by the type system, rather than by the operational rules for external and internal choice.

The second two clauses of the progress theorem above state that, if a principal (resp., a deputy) in a typed network is ready to send (resp., receive) a forward delegation, then the forward delegation will eventually happen and furthermore the corresponding backward delegation will also eventually happen. The next section provides the necessary lemmas to establish the above theorem.

6. Technical Results Required for Strong Progress

We present intermediate results required to establish progress. We employ lemmas for inferring the structure of a type from the structure of a well-typed network (Lemma 6.1), for inferring the structure of a network from its type (Lemma 6.2), and the structure of a network from labelled transitions (Lemma 6.3). Such lemmas are employed to discover the forms in each case of the key theorems *subject reduction* (Theorem 6.6) and *session fidelity* (Theorem 6.7).

As standard, subject reduction ensures that a well-typed network always reduces to a well-typed network. The novel cases in this work are those that involve interactions where at least one participant is a deputy acting on behalf of a principal, in which case Lemma 6.4 ensures the form of types reflects the restrictions on what actions a deputy can perform. Session fidelity refines subject reduction, by allowing us to be more precise about the form of the global types before and after a specific atomic interaction. The proof of progress employs session fidelity. A novelty of the proof of progress is that, in the case of simple inputs, we must check that the receiving participant appears on every control flow path and the first action encountered for that participant is a simple input (Lemma 6.8). Note this subtlety arises due to the style of definition of session projection using the meet operator for combining consistent inputs (Definition 4.5).

6.1. Standard technical lemmas about the type system

We start by proving the classical lemmas for inversion and canonical forms.

Lemma 6.1 (Inversion Lemma). *The following hold:*

1. *If* $\Gamma \vdash \Sigma_{i\in I}\pi_i; P_i : \mathsf{T}$, *then* $\mathsf{T} = \bigwedge_{i\in I} \pi_i; \mathsf{T}_i$ *and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
2. *If* $\Gamma \vdash \oplus_{i\in I}\pi_i; P_i : \mathsf{T}$, *then* $\mathsf{T} = \bigvee_{i\in I} \pi_i; \mathsf{T}_i$ *and* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
3. *If* $\Gamma \vdash \delta; P : \mathsf{T}$, *then* $\mathsf{T} = \delta; \mathsf{T}'$ *and* $\Gamma \vdash P : \mathsf{T}'$.
4. *If* $\Gamma \vdash \mu X.P : \mathsf{T}$, *then* $\mathsf{T} = \mu\mathbf{t}.\mathsf{T}'$ *and* $\Gamma, X : \mathbf{t} \vdash P : \mathsf{T}'$.
5. *If* $\Gamma \vdash X : \mathsf{T}$, *then* $\mathsf{T} = \mathbf{t}$ *and* $\Gamma = \Gamma', X : \mathbf{t}$.
6. *If* $\Gamma \vdash \mathbf{0} : \mathsf{T}$, *then* $\mathsf{T} = \mathsf{End}$.
7. *If* $\vdash \Pi_{i\in I}\overset{*}{\mathsf{p}}_i[\![\, P_i \,]\!] \parallel \Pi_{i\in I}\, \mathsf{p}_i[\![\, Q_i \,]\!] \parallel \Pi_{j\in J}\, \mathsf{r}_j[\![\, R_j \,]\!] : \mathsf{G}$, *then* $\mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\} \cup \{\mathsf{q}_i \mid i \in I\} \cup \{\mathsf{r}_j \mid j \in J\}$ *and*
   (a) $\vdash P_i : \mathsf{q}_i{\bullet}\!\rangle\!\rangle{\circ}; \mathsf{T}_i$ *where* $\mathsf{q}_i{\bullet}\!\rangle\!\rangle{\circ}; \mathsf{T}_i \leq \mathsf{G}\restriction_1(\mathsf{p}_i, \mathsf{q}_i)$ *for* $i \in I$;
   (b) $\vdash Q_i : \mathsf{S}_i$ *where* $\mathsf{S}_i \leq \mathsf{G}\restriction_2(\mathsf{p}_i, \mathsf{q}_i)$ *for* $i \in I$;
   (c) $\vdash R_j : \mathsf{U}_j$ *where* $\mathsf{U}_j \leq \mathsf{G}\restriction \mathsf{r}_j$ *for* $j \in J$.

Lemma 6.2 (Canonical Form Lemma). *The following hold:*

1. *If* $\Gamma \vdash P : \bigwedge_{i\in I} \pi_i; \mathsf{T}_i$ *then* $P = \Sigma_{i\in I}\pi_i; P_i$ *with* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
2. *If* $\Gamma \vdash P : \bigvee_{i\in I} \pi_i; \mathsf{T}_i$ *then* $P = \oplus_{i\in I}\pi_i; P_i$ *with* $\Gamma \vdash P_i : \mathsf{T}_i$ *for* $i \in I$.
3. *If* $\Gamma \vdash P : \delta; \mathsf{T}$ *then* $P = \delta; Q$ *with* $\Gamma \vdash Q : \mathsf{T}$.
4. *If* $\Gamma \vdash P : \mu\mathbf{t}.\mathsf{T}$, *then* $P = \mu X.Q$ *and* $\Gamma, X : \mathbf{t} \vdash Q : \mathsf{T}$.
5. *If* $\Gamma \vdash P : \mathbf{t}$, *then* $P = X$ *and* $\Gamma = \Gamma', X : \mathbf{t}$.
6. *If* $\Gamma \vdash P : \mathsf{End}$, *then* $P = \mathbf{0}$.

7. *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathsf{part}(\mathsf{G}) = \{\mathsf{p}_i \mid i \in I\} \cup \{\mathsf{q}_i \mid i \in I\} \cup \{\mathsf{r}_j \mid j \in J\}$ *and* $\mathsf{G} \upharpoonright_1(\mathsf{p}_i,\mathsf{q}_i)$, $\mathsf{G} \upharpoonright_2(\mathsf{p}_i,\mathsf{q}_i)$, $\mathsf{G} \upharpoonright \mathsf{r}_j$ *are defined for* $i \in I$ *and* $j \in J$, *then*

$$\mathbb{N} = \Pi_{i\in I}\overset{*}{\mathsf{p}}_i[\![\, P_i \,]\!] \parallel \Pi_{i\in I}\, \mathsf{p}_i[\![\, Q_i \,]\!] \parallel \Pi_{j\in J}\, \mathsf{r}_j[\![\, R_j \,]\!]$$

and

(a) $\vdash P_i : \mathsf{q}_i\bullet\!\rangle\!\rangle\circ;\mathsf{T}_i$ *where* $\mathsf{q}_i\bullet\!\rangle\!\rangle\circ;\mathsf{T}_i \leq \mathsf{G} \upharpoonright_1(\mathsf{p}_i,\mathsf{q}_i)$ *for* $i \in I$;

(b) $\vdash Q_i : \mathsf{S}_i$ *where* $\mathsf{S}_i \leq \mathsf{G} \upharpoonright_2(\mathsf{p}_i,\mathsf{q}_i)$ *for* $i \in I$;

(c) $\vdash R_j : \mathsf{U}_j$ *where* $\mathsf{U}_j \leq \mathsf{G} \upharpoonright \mathsf{r}_j$ *for* $j \in J$.

The following lemma allows us to recover the shapes of processes and networks from their labelled transitions.

Lemma 6.3 (Synthesis Lemma). *The following hold:*

1. *If* $P \xrightarrow{\mathsf{p}?\Lambda} P'$, *then* $P = \Sigma_{i\in I}\pi_i; P_i$, *where* $\pi_j = \mathsf{p}?\Lambda$ *and* $P' = P_j$ *for some* $j \in I$.

2. *If* $P \xrightarrow{\mathsf{p}!\Lambda} P'$, *then* $P = \oplus_{i\in I}\pi_i; P_i$, *where* $\pi_j = \mathsf{p}!\Lambda$ *and* $P' = P_j$ *for some* $j \in I$.

3. *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* $\mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathbb{N}_0$.

4. *If* $\mathbb{N} \xrightarrow{\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}} \mathbb{N}'$, *then* $\mathbb{N} = \mathsf{p}[\![\, \circ\langle\!\langle\bullet\mathsf{q}; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\circ\langle\!\langle\bullet; Q \,]\!] \parallel \mathbb{N}_0$ *and*
$$\mathbb{N}' = \overset{*}{\mathsf{p}}[\![\, P \,]\!] \parallel \mathsf{p}[\![\, Q \,]\!] \parallel \mathbb{N}_0.$$

5. *If* $\mathbb{N} \xrightarrow{\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$, *then* $\mathbb{N} = \overset{*}{\mathsf{p}}[\![\, \mathsf{q}\bullet\!\rangle\!\rangle\circ; P \,]\!] \parallel \mathsf{p}[\![\, \bullet\!\rangle\!\rangle\circ\mathsf{p}; Q \,]\!] \parallel \mathbb{N}_0$ *and*
$$\mathbb{N}' = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0.$$

In the proof of the following lemma we use global type contexts $\mathcal{G}$ defined by:

$$\mathcal{G} \quad ::= \quad \boxplus_{i\in I}\, \alpha_i^{\mathsf{p}};\mathsf{G}_i \boxplus \alpha^{\mathsf{p}};\mathcal{G} \;\mid\; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q};\mathcal{G} \;\mid\; \mathsf{p}\bullet\rangle\!\rangle\circ\mathsf{q};\mathcal{G} \;\mid\; [\,]$$

where I could be empty.

A context $\mathcal{G}$ that does not contain any proper choices (i.e., choices with more than one branch) will be called *flat global type context*, since in this case $\mathcal{G}[\mathsf{G}]$ can be flattened to a sequence of atomic interactions followed by G, i.e., $\mathcal{G}[\mathsf{G}] = \phi_1; \ldots; \phi_n; \mathsf{G}$. Such contexts typically arise in the scope of a delegation.

We end this section with a lemma which allows us to guess the shape of a global type starting from the shapes of its projections. More precisely, this lemma consists of five statements which may summarised as follows:

- Statement (1) says that, if the direct projection on p is a union type, then the global type is a sequence of atomic interactions not involving p followed by a choice with leader p;

- Statement (2) says that, if the direct projection on the principal or on the deputy is a forward delegation action, then the global type is a sequence of atomic interactions not involving that participant followed by the forward delegation;

- Statement (3) says that if the delegation projection on the principal is defined, then it starts with a passive backward delegation action; namely, when a well-typed process sends a delegation, then it cannot do anything else than wait for the delegation to return.

- Statement (4) says that, if the delegation projection on the deputy is not a backward delegation action, then it starts by an input or an output, and the global type is a sequence of atomic interactions not involving the principal nor the deputy, followed by a communication involving the principal;

- Statement (5) says that, if the delegation projection on the principal or on the deputy is a backward delegation action, then the global type is a sequence of atomic interactions not involving the deputy followed by the backward delegation.

Lemma 6.4 (Key Lemma).

1. *If* $\mathsf{G} \upharpoonright \mathsf{p} = \bigvee_{i\in I} \pi_i;\mathsf{T}_i$, *then*
$$\mathsf{G} = \phi_1;\ldots;\phi_n;\boxplus_{i\in I}\alpha_i^{\mathsf{p}};\mathsf{G}_i$$
where ϕ_j *for* $1 \le j \le n$ *is an atomic interaction not involving* p *and* $(\alpha_i^{\mathsf{p}};\mathsf{G}_i) \upharpoonright \mathsf{p} = \pi_i;\mathsf{T}_i$ *for* $i \in I$.

2. *If either* $\mathsf{G} \upharpoonright \mathsf{p} = \circ\langle\!\langle\bullet\mathsf{q};\mathsf{T}$ *or* $\mathsf{G} \upharpoonright \mathsf{q} = \mathsf{p}\circ\langle\!\langle\bullet;\mathsf{S}$, *then* $\mathsf{G} = \phi_1;\ldots;\phi_n;\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q};\mathsf{G}'$, *where* ϕ_i *for* $1 \le i \le n$ *is an atomic interaction not involving* p *in the first case and* q *in the second case.*

3. *If* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q})$ *is defined, then* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q}) = \mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T}$ *for some* T.

4. *If* $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q})$ *is defined, then there exists* T *such that either* $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q}) = \bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{T}$ *or* $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q}) = \pi;\mathsf{T}$ *and*
$$\mathsf{G} = \phi_1;\ldots;\phi_n;\alpha;\mathsf{G}'$$
where ϕ_j *for* $1 \le j \le n$ *is an atomic interaction not involving* p, q *and* $\mathsf{p} \in \mathsf{part}(\alpha)$ *and* $(\alpha;\mathsf{G}') \upharpoonright_2 (\mathsf{p},\mathsf{q}) = \pi;\mathsf{T}$.

5. *If either* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q}) = \mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T}$ *or* $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q}) = \bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{S}$, *then*
$$\mathsf{G} = \phi_1;\ldots;\phi_n;\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{G}'$$
where ϕ_i *for* $1 \le i \le n$ *is an atomic interaction not involving* q.

Proof (1). By definition of direct projection, see Figure 5, $\mathsf{G} = \mathcal{G}[\boxplus_{i\in I}\alpha_i^{\mathsf{p}};\mathsf{G}_i]$ and $\mathcal{G}$ cannot contain choices, since otherwise the projection on p would start with an intersection of inputs. Moreover, $\mathcal{G}$ cannot contain forward/backward delegations involving p, since otherwise the projection on p would start with a forward/backward delegation action.

(2). By definition of direct projection, $\mathsf{G} = \mathcal{G}[\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}]$ where $\mathcal{G}$ is a flat context, since the direct projection of choices must be either a union of outputs or an intersection of inputs. In the first case p cannot occur in $\phi_1;\ldots;\phi_n$, since $\phi_1;\ldots;\phi_n$ are skipped in the projection. For the same reason q cannot occur in $\phi_1;\ldots;\phi_n$ in the second case.

(3). By definition of delegation projection, see Figure 6, $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q})$ skips all communications (including those involving p), as well as all delegations not involving p or q. Moreover, by Definition 4.8(2) and Lemma 4.9, we know that no delegation involving p may occur before the backward delegation. Notice that the presence of a backward delegation is ensured by the condition of well-delegation, which in turn follows from projectability (Lemma 4.9). Hence $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q})$ must have the form $\mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T}$.

(4). By definition of delegation projection, see Figure 6, if $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q})$ does not start with $\bullet\rangle\!\rangle\circ\mathsf{p}$, i.e., it is not the projection of a backward delegation from q to p, then it must start with an input or output type generated by the projection of a communication involving p (first 2 lines of the figure).

(5). By Definition 4.8(2), Definition 4.8(3) and Lemma 4.9, $\phi_1;\ldots;\phi_n$ is a sequence of atomic interactions not involving q.

6.2. Subject reduction, with respect to global types

The subject reduction proof requires some ingenuity, since the definition of projection is quite tricky. The following lemma connects the projections of a global type plugged into a flat context to its projections after a communication.

Lemma 6.5.

1. *Let* $\mathsf{G} = \phi_1;\ldots;\phi_n;\phi;\mathsf{G}''$ *and let* $\mathsf{G}' = \phi_1;\ldots;\phi_n;\mathsf{G}''$
   - *If* p *and* q *are such that* $\{\mathsf{p},\mathsf{q}\} \cap \mathsf{part}(\phi) = \emptyset$ *and* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q})$ *is defined, then* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q}) = \mathsf{G}' \upharpoonright_1 (\mathsf{p},\mathsf{q})$. *Similarly for* $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q})$.
   - *If* p *is such that* $\mathsf{p} \notin \mathsf{part}(\phi)$ *and* $\mathsf{G} \upharpoonright \mathsf{p}$ *is defined, then* $\mathsf{G} \upharpoonright \mathsf{p} = \mathsf{G}' \upharpoonright \mathsf{p}$.

2. *Let* $\mathsf{G} = \phi_1;\ldots;\phi_n;\boxplus_{i\in I}\alpha_i;\mathsf{G}_i$ *where* $|I| > 1$ *and let* $\mathsf{G}' = \phi_1;\ldots;\phi_n;\mathsf{G}_j$, $j \in I$.
   - *If* p *and* q *are such that* $\{\mathsf{p},\mathsf{q}\} \cap \mathsf{part}(\alpha_j) = \emptyset$ *and* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q})$ *is defined, then* $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q}) \le \mathsf{G}' \upharpoonright_1 (\mathsf{p},\mathsf{q})$. *Similarly for* $\mathsf{G} \upharpoonright_2 (\mathsf{p},\mathsf{q})$.

- *If* p *is such that* $\mathsf{p} \notin \mathsf{part}(\alpha_j)$ *and* $\mathsf{G} \upharpoonright \mathsf{p}$ *is defined, then* $\mathsf{G} \upharpoonright \mathsf{p} \leq \mathsf{G}' \upharpoonright \mathsf{p}$*.*

Proof (1). Let $\mathsf{G} = \phi_1; \ldots; \phi_n; \phi; \mathsf{G}''$ and p and q be such that $\{\mathsf{p},\mathsf{q}\} \cap \mathsf{part}(\phi) = \emptyset$. From the definition of delegation projection, Figure 6, we can see that ϕ is ignored both in $\mathsf{G} \upharpoonright_1(\mathsf{p},\mathsf{q})$ and $\mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{q})$. From the definition of direct projection, Figure 5, we can see that ϕ is ignored in $\mathsf{G} \upharpoonright \mathsf{p}$. Therefore $\mathsf{G} \upharpoonright_1(\mathsf{p},\mathsf{q}) = \mathsf{G}' \upharpoonright_1(\mathsf{p},\mathsf{q})$, $\mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{q}) = \mathsf{G}' \upharpoonright_2(\mathsf{p},\mathsf{q})$ and $\mathsf{G} \upharpoonright \mathsf{p} = \mathsf{G}' \upharpoonright \mathsf{p}$.

(2). Let $\mathsf{G} = \phi_1; \ldots; \phi_n; \boxplus_{i \in I} \alpha_i; \mathsf{G}_i$ where $|I| > 1$ and p and q be such that $\{\mathsf{p},\mathsf{q}\} \cap \mathsf{part}(\alpha_j) = \emptyset$. If $\mathsf{G} \upharpoonright_1 (\mathsf{p},\mathsf{q})$ is defined, by Lemma 6.4(3) we get $\phi_m = \mathsf{q}\bullet\!\rangle\!\rangle\circ\mathsf{p}$ for some m ($1 \leq m \leq n$). This implies $\mathsf{G} \upharpoonright_1(\mathsf{p},\mathsf{q}) = \mathsf{q}\bullet\!\rangle\!\rangle\circ; \gamma_1; \ldots; \gamma_h; (\boxplus_{i \in I} \alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p}$, where $0 \leq h \leq n - m$ (by convention $\gamma_1; \ldots; \gamma_h$ is the empty sequence if $h = 0$) and the γ_k with $1 \leq k \leq h$ are either atomic actions or delegation actions. Since $\{\mathsf{p},\mathsf{q}\} \cap \mathsf{part}(\alpha_j) = \emptyset$, $(\boxplus_{i \in I} \alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p} = (\mathsf{G}_j \upharpoonright \mathsf{p}) \wedge \mathsf{T}$ for some T. Therefore by definition of $\leq$, see Figure 8, we get $(\boxplus_{i \in I} \alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p} \leq \mathsf{G}_j \upharpoonright \mathsf{p}$ and thus also

$$\mathsf{G} \upharpoonright_1(\mathsf{p},\mathsf{q}) \leq \mathsf{q}\bullet\!\rangle\!\rangle\circ; \gamma_1; \ldots; \gamma_h; \mathsf{G}_j \upharpoonright \mathsf{p} = \mathsf{G}' \upharpoonright_1(\mathsf{p},\mathsf{q})$$

If $\mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{q})$ is defined, by Lemma 6.4(4) with a similar argument we get

$$\mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{q}) = \gamma'_1; \ldots; \gamma'_{h_1}; \bullet\!\rangle\!\rangle\circ\mathsf{p}; \gamma'_{h_1+1}; \ldots; \gamma'_{h_2}; (\boxplus_{i \in I} \alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{q}$$

where $0 \leq h_1 \leq h_2 < n$ (here the first sequence is empty if $h_1 = 0$ and the second sequence is empty if $h_1 = h_2$) and the γ'_k are atomic actions in the first sequence, while they are either atomic actions or delegation actions in the second sequence. We can then conclude as in the previous case.
The proof for the case where $\mathsf{G} \upharpoonright \mathsf{p}$ is defined is easier.

Theorem 6.6 (Subject Reduction). *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathbb{N} \xrightarrow{\phi} \mathbb{N}'$*, then* $\vdash \mathbb{N}' : \mathsf{G}'$ *for some* G'*.*

Proof Structurally equivalent networks can be typed by global types generating the same tree, since the projection of global types with the same tree yields session types with the same tree (as shown in Appendix A).

The rest of the proof is done by case analysis on the reduction rules for networks. There are three rules to consider, Rule [Com], Rule [BDel] and Rule [EDel], all as premises of rule [Ct].

Let us consider rule [Com]. In this case $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0$ and

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

and $\mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathbb{N}_0$.
By Lemma 6.3(2) $P = \oplus_{i \in I} \pi_i; P_i$, where $\pi_j = \mathsf{q}!\Lambda$ and $P' = P_j$ for some $j \in I$. By Lemma 6.3(1) $Q = \Sigma_{h \in H} \pi'_h; Q_h$, where $\pi'_k = \mathsf{p}?\Lambda$ and $Q' = Q_k$ for some $k \in H$.
By Lemma 6.1(7) $\vdash P : \mathsf{T}$ and $\vdash Q : \mathsf{S}$ for some T and S.
By Lemma 6.1(2) $\mathsf{T} = \bigvee_{i \in I} \pi_i; \mathsf{T}_i$ and $\vdash P_i : \mathsf{T}_i$ for $i \in I$ and by Lemma 6.1(1) $\mathsf{S} = \bigwedge_{h \in H} \pi'_h; \mathsf{S}_h$ and $\vdash Q_h : \mathsf{S}_h$ for $h \in H$. Therefore neither p nor q may be the deputy of some open delegation, by Definition 4.8(2).
By Lemma 6.1(7) and the definition of projection there are four cases:

1. $\mathsf{T} \leq \mathsf{G} \upharpoonright \mathsf{p}$ and $\mathsf{S} \leq \mathsf{G} \upharpoonright \mathsf{q}$, i.e. both p and q are not principals;
2. $\mathsf{T} \leq \mathsf{G} \upharpoonright \mathsf{p}$ and $\mathsf{S} \leq \mathsf{G} \upharpoonright_2(\mathsf{q},\mathsf{r})$, i.e. p is not a principal but q is the principal of r;
3. $\mathsf{T} \leq \mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{s})$ and $\mathsf{S} \leq \mathsf{G} \upharpoonright \mathsf{q}$, i.e. p is the principal of s and q is not a principal;
4. $\mathsf{T} \leq \mathsf{G} \upharpoonright_2(\mathsf{p},\mathsf{s})$ and $\mathsf{S} \leq \mathsf{G} \upharpoonright_2(\mathsf{q},\mathsf{r})$, i.e. both p and q are principals, of s and r, respectively.

In the first case, by definition of $\leq$, $\mathsf{G} \upharpoonright \mathsf{p} = \bigvee_{i \in I'} \pi_i; \mathsf{T}'_i$ with $I \subseteq I'$ and $\mathsf{T}_i \leq \mathsf{T}'_i$ for $i \in I$. By Lemma 6.4(1) $\mathsf{G} = \phi_1; \ldots; \phi_n; \boxplus_{i \in I'} \alpha^{\mathsf{p}}_i; \mathsf{G}_i$, where ϕ_l for $1 \leq l \leq n$ is an atomic interaction not involving p and $(\alpha^{\mathsf{p}}_i; \mathsf{G}_i) \upharpoonright \mathsf{p} = \pi_i; \mathsf{T}'_i$ for $i \in I'$. From $\pi_j = \mathsf{q}!\Lambda$ we get $\alpha^{\mathsf{p}}_j = \mathsf{p}\Lambda\mathsf{q}$. By definition of $\leq$, $\mathsf{G} \upharpoonright \mathsf{q} = \bigwedge_{h \in H'} \pi'_h; \mathsf{S}'_h$ with $H' \subseteq H$ and $\mathsf{S}_h \leq \mathsf{S}'_h$ for $h \in H'$. From $\pi'_k = \mathsf{p}?\Lambda$ and $\alpha^{\mathsf{p}}_j = \mathsf{p}\Lambda\mathsf{q}$ it follows that $(\alpha^{\mathsf{p}}_j; \mathsf{G}_j) \upharpoonright \mathsf{q} = \pi'_k; \mathsf{S}'_k$ and that ϕ_l for $1 \leq l \leq n$ is an atomic interaction not involving q. We can then choose $\mathsf{G}' = \phi_1; \ldots; \phi_n; \mathsf{G}_j$.

We prove that $\vdash \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \mathbb{N}_0 : \mathsf{G}$ implies $\vdash \mathsf{p}[\![P']\!] \parallel \mathsf{q}[\![Q']\!] \parallel \mathbb{N}_0 : \mathsf{G}'$.
By definition of projection $\mathsf{G}' \upharpoonright \mathsf{p} = \mathsf{T}'_j$ and $\mathsf{G}' \upharpoonright \mathsf{q} = \mathsf{S}'_k$. So $\vdash P' : \mathsf{T}_j$ and $\vdash Q' : \mathsf{S}_k$, where $\mathsf{T}_j \leq \mathsf{G}' \upharpoonright \mathsf{p}$ and $\mathsf{S}_k \leq \mathsf{G}' \upharpoonright \mathsf{q}$. From Lemma 6.5(2) we get that for all r, s such that $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$, if $\mathsf{G} \upharpoonright_1 (\mathsf{r}, \mathsf{s})$ is defined then $\mathsf{G} \upharpoonright_1 (\mathsf{r}, \mathsf{s}) \leq \mathsf{G}' \upharpoonright_1 (\mathsf{r}, \mathsf{s})$. So, letting R be the process associated with participant r, if $\vdash R : \mathsf{T}_\mathsf{r}$ and $\mathsf{T}_\mathsf{r} \leq \mathsf{G} \upharpoonright_1 (\mathsf{r}, \mathsf{s})$, then $\mathsf{T}_\mathsf{r} \leq \mathsf{G}' \upharpoonright_1 (\mathsf{r}, \mathsf{s})$. A similar argument may be used for $\mathsf{G} \upharpoonright_2 (\mathsf{r}, \mathsf{s})$ and $\mathsf{G} \upharpoonright \mathsf{r}$. Therefore applying rule [T-NET] we can derive $\vdash \mathsf{p}[\![P']\!] \parallel \mathsf{q}[\![Q']\!] \parallel \mathbb{N}_0 : \mathsf{G}'$.

In the second case $\mathbb{N}_0 = \overset{*}{\mathsf{q}}[\![Q'']\!] \parallel \mathbb{N}_1$ and $\vdash Q'' : \mathsf{S}_\mathsf{q}$ where $\mathsf{S}_\mathsf{q} \leq \mathsf{G} \upharpoonright_1 (\mathsf{q}, \mathsf{r})$. Moreover, $\mathsf{S} \leq \mathsf{G} \upharpoonright_2 (\mathsf{q}, \mathsf{r})$ implies that $\mathsf{G} \upharpoonright_2 (\mathsf{q}, \mathsf{r})$ cannot start with $\bullet\rangle\!\rangle\circ\mathsf{q}$. By Lemma 6.4(4) $\mathsf{G} = \phi_1; \ldots; \phi_n; \alpha; \mathsf{G}_0$, where ϕ_l for $1 \leq l \leq n$ is an atomic interaction not involving q, r and $(\alpha; \mathsf{G}_0) \upharpoonright_2 (\mathsf{q}, \mathsf{r}) = \pi; \mathsf{S}'$ and $\mathsf{q} \in \mathsf{part}(\alpha)$. Then $\mathsf{S} \leq \mathsf{G} \upharpoonright_2 (\mathsf{q}, \mathsf{r})$ implies that $\pi = \mathsf{p}?\Lambda$, and $\alpha = \mathsf{p}\Lambda\mathsf{q}$, and $\mathsf{S} = \mathsf{p}?\Lambda; \mathsf{S}'' \wedge \mathsf{S}_0$ for some S_0 with $\mathsf{S}'' \leq \mathsf{S}'$, by definition of $\leq$. Moreover ϕ_l for $1 \leq l \leq n$ is an atomic interaction not involving p. From $\mathsf{T} \leq \mathsf{G} \upharpoonright \mathsf{p}$ we get $\mathsf{G} \upharpoonright \mathsf{p} = \mathsf{q}!\Lambda; \mathsf{T}'$ and $\mathsf{T} = \mathsf{q}!\Lambda; \mathsf{T}''$ with $\mathsf{T}'' \leq \mathsf{T}'$. We can then choose $\mathsf{G}' = \phi_1; \ldots; \phi_n; \mathsf{G}_0$. By construction $\vdash P' : \mathsf{T}''$ and $\vdash Q' : \mathsf{S}''$. By definition of projection $\mathsf{G}' \upharpoonright \mathsf{p} = \mathsf{T}'$ and $\mathsf{G} \upharpoonright_1 (\mathsf{q}, \mathsf{r}) = \mathsf{G}' \upharpoonright_1 (\mathsf{q}, \mathsf{r})$ and $\mathsf{G}' \upharpoonright_2 (\mathsf{q}, \mathsf{r}) = \mathsf{S}'$. Then $\mathsf{T}'' \leq \mathsf{G}' \upharpoonright \mathsf{p}$, $\mathsf{S}_\mathsf{q} = \mathsf{G}' \upharpoonright_1 (\mathsf{q}, \mathsf{r})$ and $\mathsf{S}'' \leq \mathsf{G}' \upharpoonright_2 (\mathsf{q}, \mathsf{r})$. Moreover by Lemma 6.5(1) $\mathsf{G} \upharpoonright \mathsf{s} = \mathsf{G}' \upharpoonright \mathsf{s}$, $\mathsf{G} \upharpoonright_1 (\mathsf{s}, \mathsf{s}') = \mathsf{G}' \upharpoonright_1 (\mathsf{s}, \mathsf{s}')$, $\mathsf{G} \upharpoonright_2 (\mathsf{s}', \mathsf{s}) = \mathsf{G}' \upharpoonright_2 (\mathsf{s}', \mathsf{s})$ for all s, s' such that $\{\mathsf{p}, \mathsf{q}, \mathsf{r}\} \cap \{\mathsf{s}, \mathsf{s}'\} = \emptyset$. Therefore applying rule [T-NET] we can derive $\vdash \mathsf{p}[\![P']\!] \parallel \mathsf{q}[\![Q']\!] \parallel \mathbb{N}_0 : \mathsf{G}'$.

The proof for the remaining two cases is similar to that of the second case.

Consider now rule [BDEL]. Then $\mathbb{N} = \mathsf{p}[\![\circ\langle\!\langle\bullet\mathsf{q}; P]\!] \parallel \mathsf{q}[\![\mathsf{p}\circ\langle\!\langle\bullet; Q]\!] \parallel \mathbb{N}_0$ and

$$\mathsf{p}[\![\circ\langle\!\langle\bullet\mathsf{q}; P]\!] \parallel \mathsf{q}[\![\mathsf{p}\circ\langle\!\langle\bullet; Q]\!] \xrightarrow{\mathsf{p}\circ\langle\bullet\mathsf{q}} \overset{*}{\mathsf{p}}[\![P]\!] \parallel \mathsf{p}[\![Q]\!]$$

and $\mathbb{N}' = \overset{*}{\mathsf{p}}[\![P]\!] \parallel \mathsf{p}[\![Q]\!] \parallel \mathbb{N}_0$.
By Lemma 6.1(7c) we have $\vdash \circ\langle\!\langle\bullet\mathsf{q}; P : \mathsf{T}$ and $\vdash \mathsf{p}\circ\langle\!\langle\bullet; Q : \mathsf{S}$ for some T and S such that $\mathsf{T} \leq \mathsf{G} \upharpoonright \mathsf{p}$ and $\mathsf{S} \leq \mathsf{G} \upharpoonright \mathsf{q}$. Then by Lemma 6.1(3) we get $\mathsf{T} = \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{T}'$ where $\vdash P : \mathsf{T}'$ and $\mathsf{S} = \mathsf{p}\circ\langle\!\langle\bullet; \mathsf{S}'$ where $\vdash Q : \mathsf{S}'$. By definition of $\leq$ it must be $\mathsf{G} \upharpoonright \mathsf{p} = \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{T}''$ and $\mathsf{G} \upharpoonright \mathsf{q} = \mathsf{p}\circ\langle\!\langle\bullet; \mathsf{S}''$ where $\mathsf{T}' \leq \mathsf{T}''$ and $\mathsf{S}' \leq \mathsf{S}''$. By Lemma 6.4(2) $\mathsf{G} = \phi_1; \ldots; \phi_n; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}; \mathsf{G}_0$, where ϕ_l for $1 \leq l \leq n$ is an atomic interaction not involving p and q. Note that $\mathsf{T}'' = \mathsf{G}_0 \upharpoonright_1 (\mathsf{p}, \mathsf{q})$ and $\mathsf{S}'' = \mathsf{G}_0 \upharpoonright_2 (\mathsf{p}, \mathsf{q})$. We can then choose $\mathsf{G}' = \phi_1; \ldots; \phi_n; \mathsf{G}_0$. In fact we get by definition of projection $\mathsf{G}' \upharpoonright_1 (\mathsf{p}, \mathsf{q}) = \mathsf{T}''$ and $\mathsf{G}' \upharpoonright_2 (\mathsf{p}, \mathsf{q}) = \mathsf{S}''$. Moreover, by Lemma 6.5(1) we have $\mathsf{G} \upharpoonright \mathsf{r} = \mathsf{G}' \upharpoonright \mathsf{r}$, $\mathsf{G} \upharpoonright_1 (\mathsf{r}, \mathsf{s}) = \mathsf{G}' \upharpoonright_1 (\mathsf{r}, \mathsf{s})$, $\mathsf{G} \upharpoonright_2 (\mathsf{r}, \mathsf{s}) = \mathsf{G}' \upharpoonright_2 (\mathsf{r}, \mathsf{s})$ for all r, s such that $\{\mathsf{p}, \mathsf{q}\} \cap \{\mathsf{r}, \mathsf{s}\} = \emptyset$. Therefore we may apply rule [T-NET] to derive $\vdash \overset{*}{\mathsf{p}}[\![P]\!] \parallel \mathsf{p}[\![Q]\!] \parallel \mathbb{N}_0 : \mathsf{G}'$.

Let us consider rule [EDEL], then $\mathbb{N} = \overset{*}{\mathsf{p}}[\![\mathsf{q}\bullet\rangle\!\rangle\circ; P]\!] \parallel \mathsf{p}[\![\bullet\rangle\!\rangle\circ\mathsf{p}; Q]\!] \parallel \mathbb{N}_0$ and

$$\overset{*}{\mathsf{p}}[\![\mathsf{q}\bullet\rangle\!\rangle\circ; P]\!] \parallel \mathsf{p}[\![\bullet\rangle\!\rangle\circ\mathsf{p}; Q]\!] \xrightarrow{\mathsf{q}\bullet\rangle\circ\mathsf{p}} \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!]$$

and $\mathbb{N}' = \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \mathbb{N}_0$.
By Lemma 6.1(7) $\vdash \mathsf{q}\bullet\rangle\!\rangle\circ; P : \mathsf{T}$ and $\vdash \bullet\rangle\!\rangle\circ\mathsf{p}; Q : \mathsf{S}$ for some T and S.
By Lemma 6.1(3) $\mathsf{T} = \mathsf{q}\bullet\rangle\!\rangle\circ; \mathsf{T}'$ and $\vdash P : \mathsf{T}'$ and $\mathsf{S} = \bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{S}'$ and $\vdash Q : \mathsf{S}'$. By Lemma 6.1(7b) and the fact that only delegation projection $\mathsf{G} \upharpoonright_1 (\mathsf{p}, \mathsf{q})$ and $\mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q})$ generate types starting with $\mathsf{q}\bullet\rangle\!\rangle\circ$ and $\bullet\rangle\!\rangle\circ\mathsf{p}$, respectively, we derive that $\mathsf{T} \leq \mathsf{G} \upharpoonright_1 (\mathsf{p}, \mathsf{q})$ and $\mathsf{S} \leq \mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q})$. This implies $\mathsf{G} \upharpoonright_1 (\mathsf{p}, \mathsf{q}) = \mathsf{q}\bullet\rangle\!\rangle\circ; \mathsf{T}''$ and $\mathsf{G} \upharpoonright_2 (\mathsf{p}, \mathsf{q}) = \bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{S}''$, where $\mathsf{T}' \leq \mathsf{T}''$ and $\mathsf{S}' \leq \mathsf{S}''$, by definition of $\leq$. By Lemma 6.4(5) $\mathsf{G} = \phi_1; \ldots; \phi_n; \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{G}_0$, where ϕ_l for $1 \leq l \leq n$ is an atomic interaction not involving p and q. We can then choose $\mathsf{G}' = \phi_1; \ldots; \phi_n; \mathsf{G}_0$. In fact we get by definition of projection $\mathsf{G}' \upharpoonright \mathsf{p} = \mathsf{T}''$ and $\mathsf{G}' \upharpoonright \mathsf{q} = \mathsf{S}''$. Moreover, by Lemma 6.5(1) $\mathsf{G} \upharpoonright \mathsf{r} = \mathsf{G}' \upharpoonright \mathsf{r}$, $\mathsf{G} \upharpoonright_1 (\mathsf{r}, \mathsf{s}) = \mathsf{G}' \upharpoonright_1 (\mathsf{r}, \mathsf{s})$, $\mathsf{G} \upharpoonright_2 (\mathsf{r}, \mathsf{s}) = \mathsf{G}' \upharpoonright_2 (\mathsf{r}, \mathsf{s})$ for all r, s such that $\{\mathsf{p}, \mathsf{q}\} \cap \{\mathsf{r}, \mathsf{s}\} = \emptyset$. Therefore applying rule [T-NET] we can derive $\vdash \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \mathbb{N}_0 : \mathsf{G}'$.

6.3. Session fidelity with internal delegation

In our setting, session fidelity means that the communications and the delegations between participants are performed as prescribed by the global type.

Theorem 6.7 (Session Fidelity). *The following hold:*

1. *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$, *then* $\mathsf{G} = \phi_1; \ldots; \phi_n; (\boxplus_{i\in I}\, \alpha_i^{\mathsf{p}}; \mathsf{G}_i \boxplus \mathsf{p}\Lambda\mathsf{q}; \mathsf{G}')$, *where* ϕ_j *for* $1 \leq j \leq n$ *is an atomic interaction not involving* p *and* q.

2. *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathbb{N} \xrightarrow{\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}} \mathbb{N}'$, *then* $\mathsf{G} = \phi_1; \ldots; \phi_n; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}; \mathsf{G}'$, *where* ϕ_i *for* $1 \leq i \leq n$ *is an atomic interaction not involving* p *and* q.

3. *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathbb{N} \xrightarrow{\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$, *then* $\mathsf{G} = \phi_1; \ldots; \phi_n; \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{G}'$, *where* ϕ_i *for* $1 \leq i \leq n$ *is an atomic interaction not involving* p *and* q.

4. *If* $\vdash \mathbb{N} : \boxplus_{i\in I}\mathsf{p}\Lambda_i\mathsf{q}_i; \mathsf{G}_i$, *then* $\mathbb{N} = \mathsf{p}[\![\, \oplus_{i\in I'}\mathsf{q}_i!\Lambda_i; P_i \,]\!] \parallel \mathbb{N}_0$ *with* $I' \subseteq I$ *and* $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda_i\mathsf{q}_i} \mathbb{N}_i$ *and* $\vdash \mathbb{N}_i : \mathsf{G}_i$ *for all* $i \in I'$.

5. *If* $\vdash \mathbb{N} : \phi; \mathsf{G}$, *then* $\mathbb{N} \xrightarrow{\phi} \mathbb{N}'$ *and* $\vdash \mathbb{N}' : \mathsf{G}$.

Proof (1). By Lemma 6.3(3) $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda\mathsf{q}} \mathbb{N}'$ implies $\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0$ and $P \xrightarrow{\mathsf{q}!\Lambda} P'$ and $Q \xrightarrow{\mathsf{p}?\Lambda} Q'$ and $\mathbb{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q' \,]\!] \parallel \mathbb{N}_0$. As in the proof of Theorem 6.6 we obtain $\mathsf{G} = \phi_1; \ldots; \phi_n; (\boxplus_{i\in I}\, \alpha_i^{\mathsf{p}}; \mathsf{G}_i \boxplus \mathsf{p}\Lambda\mathsf{q}; \mathsf{G}')$ where ϕ_j for $1 \leq j \leq n$ is an atomic interaction not involving p and q.

(2). By Lemma 6.3(4) $\mathbb{N} \xrightarrow{\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}} \mathbb{N}'$ implies $\mathbb{N} = \mathsf{p}[\![\, \circ\langle\!\langle\bullet\mathsf{q}; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\circ\langle\!\langle\bullet; Q \,]\!] \parallel \mathbb{N}_0$ and $\mathbb{N}' = \overset{*}{\mathsf{p}}[\![\, P \,]\!] \parallel \mathsf{p}[\![\, Q \,]\!] \parallel \mathbb{N}_0$. As in the proof of Theorem 6.6 we obtain $\mathsf{G} = \phi_1; \ldots; \phi_n; \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}; \mathsf{G}'$, where ϕ_i for $1 \leq i \leq n$ is an atomic interaction not involving p and q.

(3). By Lemma 6.3(5) $\mathbb{N} \xrightarrow{\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$ implies $\mathbb{N} = \overset{*}{\mathsf{p}}[\![\, \mathsf{q}\bullet\rangle\!\rangle\circ; P \,]\!] \parallel \mathsf{p}[\![\, \bullet\rangle\!\rangle\circ\mathsf{p}; Q \,]\!] \parallel \mathbb{N}_0$ and $\mathbb{N}' = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0$. As in the proof of Theorem 6.6 we obtain $\mathsf{G} = \phi_1; \ldots; \phi_n; \mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{G}'$, where ϕ_i for $1 \leq i \leq n$ is an atomic interaction not involving p and q.

(4). Let $\mathsf{G} = \boxplus_{i\in I}\mathsf{p}\Lambda_i\mathsf{q}_i; \mathsf{G}_i$. By Lemma 6.2(7b) or (7c)

$$\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \Pi_{i\in I}\mathsf{q}_i[\![\, Q_i \,]\!] \parallel \mathbb{N}_0$$

and $\vdash P : \mathsf{T}$ with either $\mathsf{T} \leq \mathsf{G}\upharpoonright\mathsf{p}$ or $\mathsf{T} \leq \mathsf{G}\upharpoonright_2(\mathsf{p},\mathsf{r})$ and $\vdash Q_i : \mathsf{S}_i$ with either $\mathsf{S}_i \leq \mathsf{G}\upharpoonright\mathsf{q}_i$ or $\mathsf{S}_i \leq \mathsf{G}\upharpoonright_2(\mathsf{q}_i, \mathsf{s}_i)$ for $i \in I$. In both cases by definition of $\leq$ and projection $\mathsf{T} = \bigvee_{i\in I'} \mathsf{q}_i!\Lambda_i; \mathsf{T}_i$ with $I' \subseteq I$ and $\mathsf{S}_i = \mathsf{p}?\Lambda_i; \mathsf{S}_i' \wedge \mathsf{S}_i''$ for $i \in I$. Lemma 6.2(2) gives $P = \oplus_{i\in I'}\mathsf{q}_i!\Lambda_i; P_i$. Lemma 6.2(1) gives $Q_i = \mathsf{p}?\Lambda_i; Q_i' + Q_i''$ for $i \in I$. By the reduction rules [Com] and [Ct] $\mathbb{N} \xrightarrow{\mathsf{p}\Lambda_i\mathsf{q}_i} \mathsf{p}[\![\, P_i \,]\!] \parallel \mathsf{q}_i[\![\, Q_i' \,]\!] \parallel \Pi_{j\in I\setminus\{i\}}\mathsf{q}_j[\![\, Q_j \,]\!] \parallel \mathbb{N}_0$ for all $i \in I'$. As in the proof of Subject Reduction we can show that

$$\vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathsf{q}_i[\![\, Q_i' \,]\!] \parallel \Pi_{j\in I\setminus\{i\}}\mathsf{q}_j[\![\, Q_j \,]\!] \parallel \mathbb{N}_0 : \mathsf{G}_i$$

for all $i \in I'$.

(5). If ϕ is a communication the statement is a particular case of the previous point.
Let $\phi = \mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}$. By Lemma 6.2(7c) and the definition of projection

$$\mathbb{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathbb{N}_0$$

and $\vdash P : \mathsf{T}$ with $\mathsf{T} \leq \mathsf{G}\upharpoonright\mathsf{p}$ and $\vdash Q : \mathsf{S}$ with $\mathsf{S} \leq \mathsf{G}\upharpoonright\mathsf{q}$ by definition of projection. By definition of $\leq$ we get $\mathsf{T} = \circ\langle\!\langle\bullet\mathsf{q}; \mathsf{T}'$ and $\mathsf{S} = \mathsf{p}\circ\langle\!\langle\bullet; \mathsf{S}'$. Lemma 6.2(3) implies $P = \circ\langle\!\langle\bullet\mathsf{q}; P'$ and $Q = \mathsf{p}\circ\langle\!\langle\bullet; Q'$. By applying the reduction rules [BDel] and [Ct] $\mathbb{N} \xrightarrow{\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}} \overset{*}{\mathsf{p}}[\![\, P' \,]\!] \parallel \mathsf{p}[\![\, Q' \,]\!] \parallel \mathbb{N}_0$. As in the proof of Subject Reduction we can show that $\vdash \overset{*}{\mathsf{p}}[\![\, P' \,]\!] \parallel \mathsf{p}[\![\, Q' \,]\!] \parallel \mathbb{N}_0 : \mathsf{G}$.
If $\phi = \mathsf{p}\bullet\rangle\!\rangle\circ\mathsf{q}$ the proof is similar.

6.4. Proof of strong progress

To prove the progress property it is handy to represent global types as trees. In these trees the internal nodes are decorated by $\boxplus$ or nothing, the branches are decorated by atomic interactions, and the leaves are decorated by End. In case the global type has some recursive subtype, the tree is an infinite (regular) tree.

Lemma 6.8. *If* $\mathsf{G}\upharpoonright\mathsf{p} = \bigwedge_{i\in I} \mathsf{q}_i?\lambda_i; \mathsf{T}_i$, *then* p *occurs on each path from the root of* G *and the first atomic interaction involving* p *is* $\mathsf{q}_i\lambda_i\mathsf{p}$ *for some* $i \in I$.

Proof First we show that p occurs on each path from the root of G. Assume there is a path without occurrences of p, then the projection of the corresponding global type on p would be End. This is impossible, since the meet between an intersection of simple inputs and End is undefined.

We show now that on each path from the root of G the first atomic interaction involving p is $\mathsf{q}_i\lambda_i\mathsf{p}$ for some $i \in I$. The proof is by induction on the maximal length of the paths from the root of G to the first atomic interaction involving p.
Let $\mathsf{G} = \boxplus_{j\in J}\,\alpha^{\mathsf{r}}_j;\mathsf{G}_j$: by definition of projection $\mathsf{p} \neq \mathsf{r}$ and $\mathsf{G}\restriction\mathsf{p} = \bigsqcap_{j\in J}(\,\alpha^{\mathsf{r}}_j;\mathsf{G}_j\,)\restriction\mathsf{p}$. Consider $\ell \in J$, then either $\alpha^{\mathsf{r}}_\ell = \mathsf{q}_{i_\ell}\lambda_{i_\ell}\mathsf{p}$ for some $i_\ell \in I$ or not. In the second case from the definition of meet $\mathsf{G}_l\restriction\mathsf{p} = \bigwedge_{k\in K}\mathsf{q}'_k?\lambda'_k;\mathsf{T}'_k$ and for all $k \in K$ we have $\mathsf{q}'_k?\lambda'_k = \mathsf{q}_i?\lambda_i$ for some $i \in I$. By induction hypothesis on each path from the root of G_ℓ the first atomic interaction involving p is $\mathsf{q}_k\lambda_k\mathsf{p}$ for some $k \in K$.
Let $\mathsf{G} = \delta;\mathsf{G}_0$. By definition of projection δ cannot involve p. By induction hypothesis on each path from the root of G_0 the first atomic interaction involving p is $\mathsf{q}_i\lambda_i\mathsf{p}$ for some $i \in I$.

Note the previous lemma does not hold if simple inputs are replaced by connecting inputs, since the meet between connecting inputs and End is defined.

By employing the above lemma and Theorem 6.7 we can establish the main result of this paper: progress — Theorem 5.2. To this end, we prove the following four lemmas, already mentioned in the previous section.

Lemma 6.9 (Progress scenario 1). *If* $\mathbb{N} = \mathsf{p}[\![\,\oplus_{i\in I}\mathsf{q}_i!\Lambda_i;P_i\,]\!] \parallel \mathbb{N}_0$ *is a well typed network, then* $\mathbb{N} \xrightarrow{\overrightarrow{\phi}\mathsf{p}\Lambda_i\mathsf{q}_i} \mathbb{N}'$ *for some* $\overrightarrow{\phi}$ *and for all* $i \in I$.

Proof By Lemma 6.1(7b) or (7c) and the definition of projection $\vdash \oplus_{i\in I}\mathsf{q}_i!\Lambda_i;P_i : \mathsf{T}$ with either $\mathsf{T} \leq \mathsf{G}\restriction\mathsf{p}$ or $\mathsf{T} \leq \mathsf{G}\restriction_2(\mathsf{p},\mathsf{r})$ when $|I| = 1$. By Lemma 6.1(2) $\mathsf{T} = \bigvee_{i\in I}\mathsf{q}_i!\Lambda_i;\mathsf{T}_i$. Then either $\mathsf{G}\restriction\mathsf{p} = \bigvee_{i\in I'}\mathsf{q}_i!\Lambda_i;\mathsf{T}_i$ with $I \subseteq I'$ or $\mathsf{G}\restriction_2(\mathsf{p},\mathsf{r}) = \mathsf{q}!\Lambda;\mathsf{T}'$ by definition of $\leq$ and projection. In the first case Lemma 6.4(1) gives $\mathsf{G} = \phi_1;\ldots;\phi_n;\boxplus_{i\in I'}\mathsf{p}\Lambda_i\mathsf{q}_i;\mathsf{G}_i$, where ϕ_j for $1 \leq j \leq n$ is an atomic interaction not involving p. In the second case Lemma 6.4(4) gives $\mathsf{G} = \phi_1;\ldots;\phi_n;\mathsf{p}\Lambda\mathsf{q};\mathsf{G}'$, where ϕ_j for $1 \leq j \leq n$ is an atomic interaction not involving p. In both cases we obtain the desired reduction by repeated applications of Theorem 6.7(5) and in the first case by one application of Theorem 6.7(4) as last reduction step.

Lemma 6.10 (Progress scenario 2). *If* $\mathbb{N} = \mathsf{p}[\![\,\Sigma_{i\in I}\mathsf{q}_i?\lambda_i;P_i\,]\!] \parallel \mathbb{N}_0$ *is a well typed network, then* $\mathbb{N} \xrightarrow{\overrightarrow{\phi}\mathsf{q}_i\lambda_i\mathsf{p}} \mathbb{N}'$ *for some* $\overrightarrow{\phi}$ *and for some* $i \in I$.

Proof By Lemma 6.1(7b) or (7c) and the definition of projection $\vdash \Sigma_{i\in I}\mathsf{q}_i?\lambda_i;P_i : \mathsf{T}$ with either $\mathsf{T} \leq \mathsf{G}\restriction\mathsf{p}$ or $\mathsf{T} \leq \mathsf{G}\restriction_2(\mathsf{p},\mathsf{r})$. By Lemma 6.1(2) $\mathsf{T} = \bigwedge_{i\in I}\mathsf{q}_i?\lambda_i;\mathsf{T}_i$. By definition of $\leq$ and projection either $\mathsf{G}\restriction\mathsf{p} = \bigwedge_{i\in I'}\mathsf{q}_i?\lambda_i;\mathsf{T}_i$ with $I \supseteq I'$ or $\mathsf{G}\restriction_2(\mathsf{p},\mathsf{r}) = \mathsf{q}?\lambda;\mathsf{T}'$. In the first case Lemma 6.8 assures that p occurs on each path from the root of G and the first atomic interaction involving p is $\mathsf{q}_i\lambda_i\mathsf{p}$ for some $i \in I$. In the second case Lemma 6.4(4) gives $\mathsf{G} = \phi_1;\ldots;\phi_n;\mathsf{q}\lambda\mathsf{p};\mathsf{G}'$, where ϕ_j for $1 \leq j \leq n$ is an atomic interaction not involving p. In both cases we obtain the desired reduction by repeated applications of Theorems 6.7(4) and (5).

Lemma 6.11 (Progress scenario 3). *If* $\mathbb{N} = \mathsf{p}[\![\,\circ\langle\!\langle\bullet\mathsf{q};P\,]\!] \parallel \mathbb{N}_0$ *is a well typed network, then* $\mathbb{N} \xrightarrow{\overrightarrow{\phi}\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}\overrightarrow{\phi'}\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$ *for some* $\overrightarrow{\phi}$ *and* $\overrightarrow{\phi'}$.

Proof By Lemma 6.1(7c) $\vdash \circ\langle\!\langle\bullet\mathsf{q};P : \mathsf{T}$ with $\mathsf{T} \leq \mathsf{G}\restriction\mathsf{p}$. By Lemma 6.1(3) $\mathsf{T} = \circ\langle\!\langle\bullet\mathsf{q};\mathsf{T}'$. The definition of $\leq$ gives $\mathsf{G}\restriction\mathsf{p} = \circ\langle\!\langle\bullet\mathsf{q};\mathsf{S}$. By Lemma 6.4(2) $\mathsf{G} = \phi_1;\ldots;\phi_n;\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q};\mathsf{G}'$, where ϕ_i for $1 \leq i \leq n$ is an atomic interaction not involving p. The definition of projection implies $\mathsf{S} = \mathsf{G}'\restriction_1(\mathsf{p},\mathsf{q})$. By Lemma 6.4(3) we have $\mathsf{G}'\restriction_1(\mathsf{p},\mathsf{q}) = \mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{S}'$. By Lemma 6.4(5) $\mathsf{G}' = \phi'_1;\ldots;\phi'_m;\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{G}''$, where ϕ'_j for $1 \leq j \leq m$ is an atomic interaction not involving q. We obtain the desired reduction by repeated applications of Theorem 6.7(5).

Lemma 6.12 (Progress scenario 4). *If* $\mathbb{N} = \mathsf{q}[\![\,\mathsf{p}\circ\langle\!\langle\bullet;Q\,]\!] \parallel \mathbb{N}_0$*, is a well typed network, then* $\mathbb{N} \xrightarrow{\overrightarrow{\phi}\mathsf{p}\circ\langle\!\langle\bullet\mathsf{q}\overrightarrow{\phi'}\mathsf{q}\bullet\rangle\!\rangle\circ\mathsf{p}} \mathbb{N}'$ *for some* $\overrightarrow{\phi}$ *and* $\overrightarrow{\phi'}$.

Proof The proof is similar to the proof of the previous lemma.

The above four lemmas cover all cases in Theorem 5.2.

$$\frac{}{\mathsf{p}[\![\,\mathsf{End}\,]\!]} \qquad \frac{\mathcal{T}}{\mathsf{p}[\![\,\mathsf{End}\,]\!]\,,\mathcal{T}} \qquad \frac{\mathsf{p}[\![\,\mathsf{T}\,]\!]\,,\mathsf{q}[\![\,\mathsf{U}\,]\!]\,,\mathcal{T}}{\mathsf{p}[\![\,\mathsf{q}!\Lambda;\mathsf{T}\,]\!]\,,\mathsf{q}[\![\,\mathsf{p}?\Lambda;\mathsf{U}\,]\!]\,,\mathcal{T}}$$

$$\frac{\mathsf{p}[\![\,\pi_j;\mathsf{T}_j\,]\!]\,,\mathcal{T} \quad \text{for } \textbf{all } j \in I}{\mathsf{p}[\![\,\bigvee_{i\in I}\pi_i;\mathsf{T}_i\,]\!]\,,\mathcal{T}} \qquad \frac{\mathsf{p}[\![\,\pi_j;\mathsf{T}_j\,]\!]\,,\mathcal{T} \quad \text{for } \textbf{some } j \in I}{\mathsf{p}[\![\,\bigwedge_{i\in I}\pi_i;\mathsf{T}_i\,]\!]\,,\mathcal{T}} \qquad \frac{\mathcal{T}}{\mathsf{p}[\![\,\mathsf{q}?\stackrel{\lambda}{\leftrightarrow};\mathsf{T}\,]\!]\,,\mathcal{T}}$$

$$\frac{\overset{*}{\mathsf{p}}[\![\,\mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T}\,]\!]\,,\mathsf{p}[\![\,\mathsf{U}\,]\!]\,,\mathcal{T}}{\mathsf{p}[\![\,\circ\langle\!\langle\bullet\mathsf{q};\mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T}\,]\!]\,,\mathsf{q}[\![\,\mathsf{p}\circ\langle\!\langle\bullet;\mathsf{U}\,]\!]\,,\mathcal{T}} \qquad \frac{\mathsf{p}[\![\,\mathsf{T}\,]\!]\,,\mathsf{q}[\![\,\mathsf{U}\,]\!]\,,\mathcal{T}}{\overset{*}{\mathsf{p}}[\![\,\mathsf{q}\bullet\rangle\!\rangle\circ;\mathsf{T}\,]\!]\,,\mathsf{p}[\![\,\bullet\rangle\!\rangle\circ\mathsf{p};\mathsf{U}\,]\!]\,,\mathcal{T}}$$

Figure 9: Rules determining whether located types are multiparty compatible.

7. Future Work: Multiparty Compatibility and Reversible Sessions

The classic notion of duality, ensuring two session types are compatible, does not guarantee progress or the existence of a global type in the multiparty setting. An example mentioned in the introduction is the classic 3-philosopher problem, where each party is waiting for an input from the other party. Consequently, in the multiparty setting of this work it is necessary to consider more powerful notions of compatibility that take into account a more global picture of the dependencies between messages.

In the following definition we propose a notion of multiparty compatibility that is sensitive to delegation. In the presence of fixed points there are additional subtleties, such as building a suitable notion of fairness or justness into the definition [33]. We push infinite sessions to future work and restrict ourselves to finite processes in what follows.

Located types are defined similarly to networks, by the syntax:

$$\mathcal{T} ::= \mathsf{p}[\![\,\mathsf{T}\,]\!] \mid \mathcal{T}\,,\mathcal{T}$$

where the comma is an associative, commutative operator. Multiparty compatibility of located types is defined according to the rules and axiom in Fig. 9. Observe that only the rule for union types has multiple premises.

We say that a global type is *initial* if all backward delegations have corresponding forward delegations. For finite and initial global types the following holds by induction over the structure of the global type.

Proposition 7.1. *Let* G *be a finite and initial global type and* $\{\mathsf{p}_1, \ldots, \mathsf{p}_n\} = \mathsf{part}(\mathsf{G})$*, then* $\mathsf{p}_1[\![\,\mathsf{G} \upharpoonright \mathsf{p}_1\,]\!], \ldots, \mathsf{p}_n[\![\,\mathsf{G} \upharpoonright \mathsf{p}_n\,]\!]$ *are multiparty compatible.*

The condition of being initial is needed in previous proposition, since after some delegation started we need to use the delegation projection.

The converse direction would establish that for multiparty-compatible located types, there exists a global type such that projection for each participant of the global type is a super type of the pre-type of each participant. This property is sometimes referred to as checking whether types are *coherent* [27] with respect to a global type. However, care must be taken to establish the assumption under which this holds; for example, we must ensure there are no choices made by a delegated process. A detailed study of such conditions is pushed to future work.

A separate line of future work is to study the hypothesis that internal delegation facilitates a theory of reversible sessions. In the literature, there are various proposals for formalising reversibility in sessions, but few [34, 25] consider session interleaving and delegation, since these features highly complicate the matter. Thanks to its simplicity, internal delegation could be more easily amenable to a smooth integration with reversibility, leading to a flexible and expressive calculus of reversible multiparty sessions. We leave the investigation of this question for future work.

8. Related Work

Session types to govern communication protocols have been first proposed for dyadic communications [15]. Delegation for binary sessions was introduced as early as the end of the 90's, in [16], as a means of appointing part of the conversation in a session to a participant acting in another session. In a binary session framework, delegation is also the only way to allow more than two participants to communicate in order to achieve a common task.

The first papers on multiparty sessions are [4, 17]. In their foundational work [17], Honda et al. introduced the keystone notion of global type, from which the session types of participants could be retrieved by means of a projection operation. The calculus of [17] included delegation, which was a simple generalisation of that of [16] to the multiparty case. In the last ten years the literature on this subject has grown with very high speed. In the majority of papers session calculi are channel based, as in [17] or in the simplified setting of [3], and delegation is implemented by means of channel passing, thus exploiting an essential feature of the pi-calculus. Here instead we consider multiparty conversations confined to a single session, and delegation takes place within the session itself (delegation is an intra-session rather than a inter-session mechanism), hence we can omit channels, as done for example in [28, 12].

The paper [21] is a recent overview on the genesis and evolution of session types, produced by the IC Cost Action Betty (`http://www.dcs.gla.ac.uk/research/betty/www.behavioural-types.eu/index.html`). An account of several implementations of session types, both as languages with native session types or as extensions of mainstream languages, may be found at `http://groups.inf.ed.ac.uk/abcd/session-implementations.html`.

The papers that are most closely related to our work are [10, 19, 7, 31]. Our notion of strong progress coincides with that of [10], when restricted to the intersection of the two calculi. The same paper also uses a notion of meet for defining projection, which however is incomparable with ours since it is based on a different syntax for global types.

Connecting communications have been first discussed in [19]. Here we take the simplified and more flexible version proposed in [7]. The introductory example of [31] was our inspiration for defining delegation within a single global type. Notably, delegation in [31] is realised by sending channels as usual, while we avoid channels. The drawback is that our delegation is less expressive than usual, because some behaviours cannot be delegated (as specified by our well-delegation conditions). On the positive side, we are able to achieve both deadlock-freedom and lock-freedom without additional machinery.

Deadlock-freedom ensures there are no states where there are hanging actions and yet no atomic interactions can be performed. Lock-freedom [29] strengthens deadlock-freedom by guaranteeing that certain atomic interactions will eventually succeed. Some definitions of lock-freedom [23] are subject to assumptions guaranteeing fairness of schedulers. In the current work, the type system is sufficiently restrictive for recursive processes with choices that we are not required to make fairness assumptions. However, it would be interesting to investigate in future work whether, by introducing explicit fairness assumptions, the type system can be liberated to permit more processes to be typed while maintaining lock-freedom. Recent work [14, 13] suggest that weak notions of fairness are sufficient assumptions to ensure lock-freedom. In particular, it appears that *justness* is sufficient; where justness assumes that one party cannot prevent another party from eventually acting; thereby avoiding scenarios where one party schedules her actions in such a way that an action critical for another party to progress cannot occur.

Another approach to multiparty sessions that yields deadlock-freedom "for free" in the presence of session interleaving and delegation is that of *Choreographic Programming* [6, 26]. A choreography is a global description of a network engaged in one or more sessions, possibly interleaved with each other. In fact, a choreography may be viewed as the "abstract skeleton" of a concrete network execution. Another way to put it is that a choreography is an interleaving of global types. Therefore, although it adopts channel-based delegation which requires session interleaving, the choreographic approach guarantees progress because it provides complete information about the interleaving of different sessions and about the points where delegation actions are performed.

9. Conclusion

In this paper we have provided an approach to delegation that alleviates the problem that it can be challenging to find conditions on session types guaranteeing progress in the presence of delegation. The approach we adopt does guarantee progress for (non-ambiguous and projectable) global types (Theorem 5.2), ensuring participants connected to the session do not have hanging actions that are never triggered. Progress is established from *session fidelity* (Theorem 6.7), which ensures the processes adhere to their global type. In turn, session fidelity follows from *subject reduction* (Theorem 6.6), which requires some ingenuity in cases where at least one of the principals involved in an atomic interaction is acting as a deputy in an internal delegation.

As expected of a type system, not all networks exhibiting strong progress can be typed. However, we argue that we cover a sufficiently large class of networks. Notably, by including connecting inputs, we are able to increase the class of networks we can type that feature internal delegation. For example, without connecting inputs there would be no type for the network in Example 5.1(3) and it would not be possible to express the motivating example in Section 2. We note that with additional effort the expressive power could still be increased, for example, to allow the deputy in a delegation to make choices; or to permit nested delegations. However, we argue that the current constraints on the considered fragment are desirable, since they ensure that all choices are made by the principal, including who is the current deputy.

Acknowledgments. As theoretical computer scientists, we are all strongly indebted to Maurice Nivat, since the recognition and flourishing of our discipline owe very much to his farsighted view of computer science and to his inexhaustible energy. In particular Maurice Nivat initiated the ICALP series of conferences, which offered a unique opportunity for theoretical computer scientists to get together and share results: this conference certainly played a crucial role in Mariangiola's initial research activity. Finally, as a scientist Maurice made fundamental contributions to the theory of computing and specifically to the semantics of parallel processes, which not only shaped the whole French research community in this area but also deeply influenced the scientific community at large. Maurice was also for many years an external advisor of the team MEIJE at INRIA Sophia Antipolis, in which Ilaria Castellani was working.

[1] Martín Abadi, Michael Burrows, Charlie Kaufman, and Butler Lampson. Authentication and delegation with smart-cards. *Science of Computer Programming*, 21(2):93 – 113, 1993.

[2] Gul Agha. Concurrent object-oriented programming. *Communications of ACM*, 33(9):125–141, 1990.

[3] Lorenzo Bettini, Mario Coppo, Loris D'Antoni, Marco De Luca, Mariangiola Dezani-Ciancaglini, and Nobuko Yoshida. Global progress in dynamically interleaved multiparty sessions. In *CONCUR'08*, volume 5201 of *LNCS*, pages 418–433. Springer, 2008.

[4] Eduardo Bonelli and Adriana B. Compagnoni. Multipoint session types for a distributed calculus. In *TGC*, volume 4912 of *LNCS*, pages 240–256. Springer, 2007.

[5] Marco Carbone and Søren Debois. A graphical approach to progress for structured communication in web services. In *ICE*, volume 38 of *EPTCS*, pages 13–27, 2010.

[6] Marco Carbone and Fabrizio Montesi. Deadlock-freedom-by-design: multiparty asynchronous global programming. In *POPL*, pages 263–274. ACM, 2013.

[7] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. Reversible sessions with flexible choices. `http://www.di.unito.it/~dezani/papers/cdg18.pdf`, 2018.

[8] Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida, and Luca Padovani. Global progress for dynamically interleaved multiparty sessions. *Mathematical Structures in Computer Science*, 26(2):238–302, 2016.

[9] Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet, Karthikeyan Bhargavan, and James J. Leifer. Secure implementations for typed session abstractions. In *CSF*, pages 170–186. IEEE Computer Society, 2007.

[10] Pierre-Malo Deniélou and Nobuko Yoshida. Dynamic multirole session types. In *POPL*, pages 435–446. ACM Press, 2011.

[11] Mariangiola Dezani-Ciancaglini, Sophia Drossopoulou, Dimitris Mostrous, and Nobuko Yoshida. Objects and session types. *Information and Computation*, 207(5):595 – 641, 2009.

[12] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, and Nobuko Yoshida. Precise subtyping for synchronous multiparty sessions. In *PLACES*, volume 203 of *EPTCS*, pages 29 – 44, 2016.

[13] Rob van Glabbeek. Structure preserving bisimilarity, supporting an operational Petri net semantics of CCSP. In *Correct System Design*, pages 99–130. Springer, 2015.

[14] Rob van Glabbeek and Peter Höfner. Progress, fairness and justness in process algebra. *arXiv*, 1501.03268, 2015.

[15] Kohei Honda. Types for dyadic interaction. In *CONCUR*, volume 715 of *LNCS*, pages 509–523. Springer, 1993.

[16] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP*, volume 1381 of *LNCS*, pages 122–138. Springer, 1998.

[17] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In *POPL*, pages 273–284. ACM Press, 2008.

[18] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of the ACM*, 63(1):9, 2016.

[19] Raymond Hu and Nobuko Yoshida. Explicit connection actions in multiparty session types. In *FASE*, volume 10202 of *LNCS*, pages 116–133. Springer, 2017.
[20] Raymond Hu, Nobuko Yoshida, and Kohei Honda. Session-based distributed programming in Java. In *ECOOP*, volume 5142 of *LNCS*, pages 516–541. Springer, 2008.
[21] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of session types and behavioural contracts. *ACM Computing Surveys*, 49(1):3:1–3:36, 2016.
[22] Message Sequence Chart (MSC). Annex B: Formal semantics of Message Sequence Charts. ITU-T Recommendation Z.120, ITU-T, Geneva, 1998.
[23] Naoki Kobayashi. A type system for lock-free processes. *Information and Computation*, 177(2):122–159, 2002.
[24] Sjouke Mauw and Michel Reniers. Operational semantics for MSC'96. *Computer Networks and ISDN Systems*, 31(17):1785–1799, 1999.
[25] Claudio Antares Mezzina and Jorge A. Pérez. Causally consistent reversible choreographies: a monitors-as-memories approach. In *PPDP*, pages 127–138. ACM Press, 2017.
[26] Fabrizio Montesi. *Choreographic Programming*. Ph.D. thesis, IT University of Copenhagen, 2013.
[27] Dimitris Mostrous, Nobuko Yoshida, and Kohei Honda. Global principal typing in partially commutative asynchronous sessions. In *ESOP*, volume 5502 of *LNCS*, pages 316–332. Springer, 2009.
[28] Luca Padovani. Session types = intersection types + union types. In *ITRS*, volume 45 of *EPTCS*, pages 71–89. Open Publishing Association, 2011.
[29] Luca Padovani. Deadlock and lock freedom in the linear pi-calculus. In *CSL-LICS*, pages 72:1–72:10. ACM Press, 2014.
[30] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002.
[31] Alceste Scalas, Ornela Dardha, Raymond Hu, and Nobuko Yoshida. A linear decomposition of multiparty sessions for safe distributed programming. In *ECOOP*, volume 74 of *LIPIcs*, pages 24:1–24:31. Schloss Dagstuhl, 2017.
[32] Alceste Scalas and Nobuko Yoshida. Lightweight session programming in Scala. In *ECOOP*, volume 56 of *LIPIcs*, pages 21:1–21:28. Schloss Dagstuhl, 2016.
[33] Alceste Scalas and Nobuko Yoshida. Multiparty session types, beyond duality. *Journal of Logic and Algebraic Methods in Programming*, 97:55–84, 2018.
[34] Francesco Tiezzi and Nobuko Yoshida. Reversible session-based pi-calculus. *Journal of Logical and Algebraic Methods in Programming*, 84(5):684–707, 2015.
[35] Akinori Yonezawa and Mario Tokoro, editors. *Object-oriented Concurrent Programming*. MIT Press, Cambridge, MA, USA, 1987.

Appendix A. Projection respects Equirecursion

$$\Theta \vdash^g \mathsf{End} \sim \mathsf{End} \qquad \Theta, (\mathsf{G}, \mathsf{G}') \vdash^g \mathsf{G} \sim \mathsf{G}' \qquad \frac{\Theta \vdash^g \mathsf{G} \sim \mathsf{G}'}{\Theta \vdash^g \mathsf{G}' \sim \mathsf{G}}$$

$$\frac{\Theta \vdash^g \mathsf{G} \sim \mathsf{G}'}{\Theta \vdash^g \phi; \mathsf{G} \sim \phi; \mathsf{G}'} \qquad \frac{\Theta \vdash^g \mathsf{G}_i \sim \mathsf{G}'_i \quad \forall i \in I}{\Theta \vdash^g \boxplus_{i\in I}\, \alpha_i; \mathsf{G}_i \sim \boxplus_{i\in I}\, \alpha_i; \mathsf{G}'_i}$$

$$\frac{\Theta, (\mu \mathbf{t}.\mathsf{G}, \mathsf{G}') \vdash^g \mathsf{G}\{\mu \mathbf{t}.\mathsf{G}/\mathbf{t}\} \sim \mathsf{G}'}{\Theta \vdash^g \mu \mathbf{t}.\mathsf{G} \sim \mathsf{G}'}$$

Figure A.10: Equirecursive global types.

Let Θ denote a set of global type pairs. Figure A.10 gives a proof system for showing that two global types generate the same tree. Let Δ denote a set of session type pairs. Figure A.11 gives a proof system for showing that two session types generate the same tree.

Lemma Appendix A.1 (Inversion Lemma). *1. If $\Delta \vdash^s \bigwedge_{i\in I} \pi_i; \mathsf{T}_i \sim \bigwedge_{i\in I'} \pi'_i; \mathsf{T}'_i$, then $I = I'$, $\pi_i = \pi'_i$ and $\Delta \vdash^s \mathsf{T}_i \sim \mathsf{T}'_i$ for all $i \in I$.*

2. If $\Delta \vdash^s \bigvee_{i\in I} \pi_i; \mathsf{T}_i \sim \bigvee_{i\in I'} \pi'_i; \mathsf{T}'_i$, then $I = I'$, $\pi_i = \pi'_i$ and $\Delta \vdash^s \mathsf{T}_i \sim \mathsf{T}'_i$ for all $i \in I$.

Lemma Appendix A.2. *If $\mathsf{G}\{\mathsf{G}'/\mathbf{t}\}$ is a well formed global type and $\mathbf{t}'$ occurs in G, then $\mathsf{part}(\mathsf{G}) = \mathsf{part}(\mathsf{G}\{\mathsf{G}'/\mathbf{t}\})$.*

Proof If $\mathbf{t}$ does not occur in G this is obvious. So we can assume that both $\mathbf{t}$ and $\mathbf{t}'$ occur in G. Since all type variables can only occur as rightmost operands of sequential compositions, there must be a subtype G'' of G, such that all the following conditions hold:

$$\Delta \vdash^s \mathsf{End} \sim \mathsf{End} \qquad \Delta, (\mathsf{T}, \mathsf{T}') \vdash^s \mathsf{T} \sim \mathsf{T}' \qquad \frac{\Delta \vdash^s \mathsf{T} \sim \mathsf{T}'}{\Delta \vdash^s \mathsf{T}' \sim \mathsf{T}}$$

$$\frac{\Delta \vdash^s \mathsf{T} \sim \mathsf{T}'}{\Delta \vdash^s \delta; \mathsf{T} \sim \delta; \mathsf{T}'} \qquad \frac{\Delta \vdash^s \mathsf{T}_i \sim \mathsf{T}'_i \quad \forall i \in I}{\Delta \vdash^s \bigwedge_{i \in I} \pi_i; \mathsf{T}_i \sim \bigwedge_{i \in I} \pi_i; \mathsf{T}'_i}$$

$$\frac{\Delta \vdash^s \mathsf{T}_i \sim \mathsf{T}'_i \quad \forall i \in I}{\Delta \vdash^s \bigvee_{i \in I} \pi_i; \mathsf{T}_i \sim \bigvee_{i \in I} \pi_i; \mathsf{T}'_i} \qquad \frac{\Delta, (\mu\mathbf{t}.\mathsf{T}, \mathsf{T}') \vdash^s \mathsf{T}\{\mu\mathbf{t}.\mathsf{T}/\mathbf{t}\} \sim \mathsf{T}'}{\Delta \vdash^s \mu\mathbf{t}.\mathsf{T} \sim \mathsf{T}'}$$

Figure A.11: Equirecursive session types.

- $\mathsf{G}'' = \boxplus_{i \in I} \alpha_i; \mathsf{G}_i$,
- $\mathbf{t}$ occurs in G_j and $\mathbf{t}'$ does not occur in G_j for some $j \in I$,
- $\mathbf{t}'$ occurs in G_k and $\mathbf{t}$ does not occur in G_k for some $k \in I$.

Assume ad absurdum that there is $\mathsf{q} \in \mathsf{part}(\mathsf{G}\{\mathsf{G}'/\mathbf{t}\})$ such that $\mathsf{q} \notin \mathsf{part}(\mathsf{G})$. Then by construction $\mathsf{q} \in \mathsf{part}(\mathsf{G}_j\{\mathsf{G}'/\mathbf{t}\})$ and $\mathsf{q} \notin \mathsf{part}(\mathsf{G}_k\{\mathsf{G}'/\mathbf{t}\})$, which gives $\mathsf{G}_k\{\mathsf{G}'/\mathbf{t}\} \upharpoonright \mathsf{q} = \mathbf{t}'$. This implies that $(\mathsf{G}\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{q}$ is undefined, since the meet between a session type and $\mathbf{t}'$ is undefined. This contradicts our hypothesis.

Lemma Appendix A.3 (Projection commutes with substitution).

$$(\mathsf{G}\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \mathsf{G} \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\}$$

Proof By structural induction on G.

Let $\mathsf{G} = \mathbf{t}'$, by definition of projection $\mathsf{G} \upharpoonright \mathsf{p} = \mathbf{t}'$. The result is immediate in both cases $\mathbf{t}' = \mathbf{t}$ or $\mathbf{t}' \neq \mathbf{t}$.

Let $\mathsf{G} = \boxplus_{i \in I} \alpha_i; \mathsf{G}_i$. We get $\mathsf{G}\{\mathsf{G}'/\mathbf{t}\} = \boxplus_{i \in I} \alpha_i; \mathsf{G}_i\{\mathsf{G}'/\mathbf{t}\}$. By induction hypothesis

$$(\mathsf{G}_i\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \mathsf{G}_i \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\} \text{ for all } i \in I$$

If $(\boxplus_{i \in I} \alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p} = \bigsqcap_{i \in I}(\alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p}$ and $J = \{i \in I \mid \alpha_i = \mathsf{q}_i \Lambda_i \mathsf{p}\}$, then

$$\begin{aligned} \textstyle\bigsqcap_{i \in I}(\alpha_i; \mathsf{G}_i\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} &= (\textstyle\bigsqcap_{i \in J} \mathsf{q}_i?\Lambda_i; \mathsf{G}_i\{\mathsf{G}'/\mathbf{t}\} \upharpoonright \mathsf{p}) \sqcap (\textstyle\bigsqcap \\ &= (\textstyle\bigsqcap_{i \in J} \mathsf{q}_i?\Lambda_i; \mathsf{G}_i \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\}) \sqcap \\ &= \textstyle\bigsqcap_{i \in I}(\alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\} \end{aligned}$$

The proof in the case $\boxplus_{i \in I} \alpha_i; \mathsf{G}_i \upharpoonright \mathsf{p} = \bigvee_{i \in I}(\alpha_i; \mathsf{G}_i) \upharpoonright \mathsf{p}$ is similar and simpler.

Let $\mathsf{G} = \phi; \mathsf{G}_1$. If ϕ is a communication, then it is a particular case of choice with $|I|=1$. If $\phi = \mathsf{r}\circ\langle\!\langle\bullet\mathsf{s}$ and neither $\mathsf{r} = \mathsf{p}$ nor $\mathsf{s} = \mathsf{p}$ the result follows easily by induction on G_1.
If $\phi = \mathsf{p}\circ\langle\!\langle\bullet\mathsf{s}$, since, by Lemma 4.9, G is well delegated, we have that $\mathsf{G} = \mathsf{p}\circ\langle\!\langle\bullet\mathsf{s}; \phi_1; \ldots; \phi_n; \mathsf{s}\bullet\rangle\!\rangle\circ\mathsf{p}; \mathsf{G}_2$ for some G_2 and p does not occur in $\phi_1; \ldots; \phi_n$. By definition of projection $(\mathsf{G}\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \circ\langle\!\langle\bullet\mathsf{s}; \mathsf{s}\bullet\rangle\!\rangle\circ; (\mathsf{G}_2\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p}$. By induction hypothesis $\circ\langle\!\langle\bullet\mathsf{s}; \mathsf{s}\bullet\rangle\!\rangle\circ; (\mathsf{G}_2\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \circ\langle\!\langle\bullet\mathsf{s}; \mathsf{s}\bullet\rangle\!\rangle\circ; (\mathsf{G}_2 \upharpoonright \mathsf{p})\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\}$ and $\circ\langle\!\langle\bullet\mathsf{s}; \mathsf{s}\bullet\rangle\!\rangle\circ; (\mathsf{G}_2 \upharpoonright \mathsf{p})\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\} = \mathsf{G} \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\}$.
If $\phi = \mathsf{r}\circ\langle\!\langle\bullet\mathsf{p}$, as before $\mathsf{G} = \mathsf{r}\circ\langle\!\langle\bullet\mathsf{p}; \phi_1; \ldots; \phi_n; \mathsf{p}\bullet\rangle\!\rangle\circ\mathsf{r}; \mathsf{G}_2$ for some G_2. By definition of projection $(\mathsf{G}\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \mathsf{r}\circ\langle\!\langle\bullet; \pi_1; \ldots; \pi_m; \bullet\rangle\!\rangle\circ\mathsf{r}; (\mathsf{G}_2\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p}$. As before the result follows by induction hypothesis on G_2.

Let $\mathsf{G} = \mu\mathbf{t}'.\mathsf{G}_1$. We may assume that $\mathbf{t} \neq \mathbf{t}'$ and $\mathbf{t}'$ does not occur in G'.

$$(\mu\mathbf{t}'.\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \begin{cases} (\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} & \text{if } \mathbf{t}' \text{ does not occur in } \mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\} \\ \mu\mathbf{t}'.(\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} & \text{if } \mathsf{p} \in \mathsf{part}(\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \\ \mathsf{End} & \text{otherwise} \end{cases}$$

By induction hypothesis we have that $(\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \mathsf{G}_1 \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\}$. Moreover, $\mathbf{t}'$ occurs in $\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}$ if and only if $\mathbf{t}'$ occurs in G_1. So the result is immediate if $\mathbf{t}'$ does not occur in G_1. Otherwise by Lemma Appendix A.2 $\mathsf{part}(\mathsf{G}_1) = \mathsf{part}(\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\})$. This implies $(\mu\mathbf{t}'.\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \mu\mathbf{t}'.(\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = \mu\mathbf{t}'.\mathsf{G}_1 \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\} = (\mu\mathbf{t}'.\mathsf{G}_1) \upharpoonright \mathsf{p}\,\{\mathsf{G}' \upharpoonright \mathsf{p}\,/\mathbf{t}\}$ if $\mathsf{p} \in \mathsf{part}(\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\})$ and $(\mu\mathbf{t}'.\mathsf{G}_1\{\mathsf{G}'/\mathbf{t}\}) \upharpoonright \mathsf{p} = (\mu\mathbf{t}'.\mathsf{G}_1) \upharpoonright \mathsf{p} = \mathsf{End}$ otherwise.

Theorem Appendix A.4. *Let* $\Theta \upharpoonright \mathsf{p} = \{(\mathsf{G} \upharpoonright \mathsf{p}, \mathsf{G}' \upharpoonright \mathsf{p}) \mid (\mathsf{G}, \mathsf{G}') \in \Theta\}$.
If $\Theta \vdash^g \mathsf{G} \sim \mathsf{G}'$, *then* $\Theta \upharpoonright \mathsf{p} \vdash^s \mathsf{G} \upharpoonright \mathsf{p} \sim \mathsf{G}' \upharpoonright \mathsf{p}$.

Proof By induction on the derivation of $\Theta \vdash^g \mathsf{G} \sim \mathsf{G}'$.

Let the last applied rule be $\dfrac{\Theta \vdash^g \mathsf{G}_i \sim \mathsf{G}'_i \quad \forall i \in I}{\Theta \vdash^g \boxplus_{i\in I} \alpha_i; \mathsf{G}_i \sim \boxplus_{i\in I} \alpha_i; \mathsf{G}'_i}$ By induction hypothesis $\Theta \restriction \mathsf{p} \vdash^s \mathsf{G}_i \restriction \mathsf{p} \sim \mathsf{G}'_i \restriction \mathsf{p}$ for all $i \in I$. Let $(\boxplus_{i\in I} \alpha_i; \mathsf{G}_i) \restriction \mathsf{p} = \prod_{i\in I}(\alpha_i; \mathsf{G}_i) \restriction \mathsf{p}$, then $(\boxplus_{i\in I} \alpha_i; \mathsf{G}'_i) \restriction \mathsf{p} = \prod_{i\in I}(\alpha_i; \mathsf{G}'_i) \restriction \mathsf{p}$. We define $J = \{i \in I \mid \alpha_i = \mathsf{q}\Lambda_i\mathsf{p}\}$, $H = \{i \in I \setminus J \mid \mathsf{G}_i \restriction \mathsf{p} = \mathsf{G}'_i \restriction \mathsf{p} = \mathsf{End}\}$ and $K = I \setminus (J \cup H)$. Notice that H includes all $\mathsf{G}_i \restriction \mathsf{p}$ and $\mathsf{G}'_i \restriction \mathsf{p}$ which are End, since $\Theta \restriction \mathsf{p} \vdash^s \mathsf{G}_i \restriction \mathsf{p} \sim \mathsf{G}'_i \restriction \mathsf{p}$ and $\Delta \vdash^s \mathsf{End} \sim \mathsf{T}$ implies $\mathsf{T} = \mathsf{End}$. Let $\mathsf{G}_i \restriction \mathsf{p} = \bigwedge_{\ell\in L_i} \pi^{(i)}_\ell; \mathsf{T}^{(i)}_\ell$ for $i \in K$, then $\Theta \restriction \mathsf{p} \vdash^s \mathsf{G}_i \restriction \mathsf{p} \sim \mathsf{G}'_i \restriction \mathsf{p}$ and Lemma Appendix A.1(1) imply $\mathsf{G}'_i \restriction \mathsf{p} = \bigwedge_{\ell\in L_i} \pi^{(i)}_\ell; \mathsf{S}^{(i)}_\ell$ and $\Theta \restriction \mathsf{p} \vdash^s \mathsf{T}^{(i)}_\ell \sim \mathsf{S}^{(i)}_\ell$ for $\ell \in L_i$ and $i \in K$. Then by definition of projection $\boxplus_{i\in I} \alpha_i; \mathsf{G}_i \restriction \mathsf{p} = (\bigwedge_{i\in J} \mathsf{q}?\Lambda_i; \mathsf{G}_i \restriction \mathsf{p}) \wedge (\bigwedge_{i\in K} \bigwedge_{\ell\in L_i} \pi^{(i)}_\ell; \mathsf{T}^{(i)}_\ell)$ and $\boxplus_{i\in I} \alpha_i; \mathsf{G}'_i \restriction \mathsf{p} = (\bigwedge_{i\in J} \mathsf{q}?\Lambda_i; \mathsf{G}'_i \restriction \mathsf{p}) \wedge (\bigwedge_{i\in K} \bigwedge_{\ell\in L_i} \pi^{(i)}_\ell; \mathsf{S}^{(i)}_\ell)$. We can then derive:

$$\frac{\Theta \restriction \mathsf{p} \vdash^s \mathsf{G}_i \restriction \mathsf{p} \sim \mathsf{G}'_i \restriction \mathsf{p} \quad \forall i \in J \qquad\qquad \Theta \restriction \mathsf{p} \vdash^s \mathsf{T}^{(i)}_\ell \sim \mathsf{S}^{(i)}_\ell \quad \forall \ell \in L_i\ \forall i \in K}{\Theta \restriction \mathsf{p} \vdash^s (\bigwedge_{i\in J} \mathsf{q}?\Lambda_i; \mathsf{G}_i \restriction \mathsf{p}) \wedge (\bigwedge_{i\in K} \bigwedge_{\ell\in L_i} \pi^{(i)}_\ell; \mathsf{T}^{(i)}_\ell) \sim (\bigwedge_{i\in J} \mathsf{q}?\Lambda_i; \mathsf{G}'_i \restriction \mathsf{p}) \wedge (\bigwedge_{i\in K} \bigwedge_{\ell\in L_i} \pi^{(i)}_\ell; \mathsf{S}^{(i)}_\ell)}$$

The proof in the case $\boxplus_{i\in I} \alpha_i; \mathsf{G}_i \restriction \mathsf{p} = \bigvee_{i\in I}(\alpha_i; \mathsf{G}_i) \restriction \mathsf{p}$ is similar and simpler.

Let the last applied rule be $\dfrac{\Theta, (\mu\mathbf{t}.\mathsf{G}, \mathsf{G}') \vdash^g \mathsf{G}\{\mu\mathbf{t}.\mathsf{G}/\mathbf{t}\} \sim \mathsf{G}'}{\Theta \vdash^g \mu\mathbf{t}.\mathsf{G} \sim \mathsf{G}'}$ By induction hypothesis $\Theta \restriction \mathsf{p}, (\mu\mathbf{t}.\mathsf{G} \restriction \mathsf{p}, \mathsf{G}' \restriction \mathsf{p}) \vdash^s (\mathsf{G}\{\mu\mathbf{t}.\mathsf{G}/\mathbf{t}\}) \restriction \mathsf{p} \sim \mathsf{G}' \restriction \mathsf{p}$. By Lemma Appendix A.3 $(\mathsf{G}\{\mu\mathbf{t}.\mathsf{G}/\mathbf{t}\}) \restriction \mathsf{p} = \mathsf{G} \restriction \mathsf{p}\{\mu\mathbf{t}.\mathsf{G} \restriction \mathsf{p}/\mathbf{t}\}$. We can then derive:

$$\frac{\Theta \restriction \mathsf{p}, (\mu\mathbf{t}.\mathsf{G} \restriction \mathsf{p}, \mathsf{G}' \restriction \mathsf{p}) \vdash^s \mathsf{G} \restriction \mathsf{p}\{\mu\mathbf{t}.\mathsf{G} \restriction \mathsf{p}/\mathbf{t}\} \sim \mathsf{G}' \restriction \mathsf{p}}{\Theta \restriction \mathsf{p} \vdash^s \mu\mathbf{t}.\mathsf{G} \restriction \mathsf{p} \sim \mathsf{G}' \restriction \mathsf{p}}$$