Modular Multiparty Sessions with Mixed Choice

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

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

MultiParty Session Types (MPST) provide a useful framework for safe concurrent systems. *Mixed choice* (enabling a participant to play at the same time the roles of sender and receiver) increases the expressive power of MPST as well as the difficulty in controlling safety of communications. Such a control is more viable when modular systems are considered and the power of mixed choice fully exploited only inside loosely coupled modules. We carry over such idea in a type assignment approach to multiparty sessions. Typability for modular sessions entails Subject Reductions, Session Fidelity and Lock Freedom.

1 Introduction

MultiParty Session Types (MPST) offer a structured approach to the development and formal verification of concurrent and distributed systems [18, 19]. As in the vast majority of choreographic formalisms, two distinct but related views of concurrent systems are taken into account: (a) the *global view*, a formal specification via *global types* of the overall behaviour of a system; (b) the *local view*, namely a description, at different levels of abstraction, of the behaviours of the single components. A key issue in MPST, and choreographies in general, is the relation between these two views. Among others, we can refer to the notion of *projection*, used till recently in most of the MPST formalisms. Given a (well-formed) global type, the projection operator produces a tuple of local types – one for each component – which generalises binary session types [16, 17]. Such local types can be looked at as an abstraction of finer grained descriptions of processes. Another approach to the MPST global-local relationship is the one embodied in the so called *Simple MultiParty Sessions* (SMPS) formalisms. Such an approach was first introduced in [12] and [4], and further investigated in a bunch of papers, among which [3, 7, 5, 6, 1, 2]. Whereas the MPST approach typically considers two-layered local views – a layer of processes and a layer of local types – SMPS are based on single-layered local views, where only a fairly abstract notion of process is considered. In SMPS, which is the general setting of the present paper, systems of communicating processes are represented as *multiparty sessions*, i.e. parallel compositions of named processes (the *participants*). Then, by means of type systems, global types are inferred for such sessions. Typability is such to ensure relevant communication properties – typically Lock Freedom – for sessions. Besides the above mentioned ones, it is worth recalling that also other approaches have been investigated, like the one introduced in [32], where the global view is only implicitly considered.

*Partially supported by Project "National Center for HPC, Big Data e Quantum Computing", Programma M4C2, Investimento 1.3 – Next Generation EU; and by the PIAno di inCEntivi per la RIcerca di Ateneo 2024-2026 UniCT (Linea di Intervento 1).

To appear in EPTCS.

A common feature of all the above mentioned approaches, till lately, has been the use of communication models where, before any interaction, a process can clearly be identified as a sender or a receiver. The intrinsic potentiality of nondeterministically choosing among both inputs and outputs inside a single process interaction (usually referred to in the literature as "mixed choice") has however recently intrigued session type researchers, both for the binary and the multiparty cases [10, 28, 29, 30, 31]. A thorough investigation of the expressivity of mixed choice in (synchronous) MPST formalisms has been carried on in [31]. For instance, mixed choice enables to implement protocols safely exploiting circular interactions, as shown in [31] through an example recalled in the present paper (see Example 2.4). This is not possible in usual MPST formalisms where typing [7] or, equivalently, the projectability condition on global types – as shown in [6] – consists essentially in checking the possibility of sequentialising the interactions in a protocol. Together with its expressive power, however, mixed choice brings subtly harmful features, as already exposed decades ago [15] in the setting of Communicating Finite State Machines [9] (an asynchronous formalism closely related to MPST). In the present paper we aim at exploiting such expressive power in a safe and controlled way using a SMPS setting. In order to do that we resort to the notion of *modularity*.

Modularity is a fairly general property of complex systems. Any complex system can be decomposed into smaller subsystems that are always going to be interdependent to some extent and independent to some other extent [33]. In fact, in many human activities, from business to biology, as well as to software engineering, modularisation offers a strategic approach enabling to cope with their complexity. Modularity in software engineering refers to the design approach that emphasises the separation of concerns: a complex software system is decomposed into smaller, loosely coupled modules, where coupling is the degree of interdependence between the modules. By means of project modularisation one manages to, among others, reduce complexity (breaking down a large system into smaller modules makes it more manageable and easier to understand [25]) as well as to improve testing and separation of concerns (SoC), a fundamental principle in software engineering. In particular, in modular programming, concerns are separated such that modules, performing logically coherent tasks, do interact through simple and manageable interfaces.

Our proposal is hence to restrict our attention to sessions corresponding to modularised systems. A type discipline is then proposed that profits, as in more rigid MPST formalisms, from a form of "sequentialisation" condition. Such a condition however, instead of being imposed on participants, is imposed on the modules forming a session, inside which the mixed choice can be freely used (at the cost of a thoroughly check, but limited inside the single modules, of all the possible interactions among participants). The inter-modules interactions are instead more controlled, so respecting the decoupling of modules characterising any sound decomposition of systems. It is then possible to prove the properties of Subject Reduction and Session Fidelity for typable modular sessions. Moreover, typability also entails the communication property of Lock Freedom. Typability is shown to be independent from the way a session can be modularised as well as from the order in which the typing of the modules is "sequentialised". We propose, as working example, a modular extension of the above mentioned leader election example of [31].

Overview. In Section 2 we introduce the calculus of multiparty sessions with mixed choice as an extension of the SMPS calculus of [4] and [7]. Modularisable multiparty sessions are then formally presented in Section 3. Global types equipped with a coinductive LTS are defined in Section 4 in the style of [6]. Properties of typable modular sessions are proven in Section 5, namely Subject Reduction, Session Fidelity and Lock Freedom. In that section we also show that typability does not depend on how a session is modularised or on the particular modules considered during typing. A summing-up section, also discussing related and future works, concludes the paper.

2 Multiparty Sessions with Mixed Choice

We present now a SMPS synchronous calculus of multiparty sessions with mixed choice, inspired mainly by [31] and partially by [4]. We assume to have the following denumerable base sets: *messages* (ranged over by $\lambda, \lambda', \ldots$); *session participants* (ranged over by $\mathsf{p}, \mathsf{q}, \mathsf{r}, \mathsf{s}, \ldots$); *indexes* (ranged over by $i, j, h, k, \ldots$); *finite sets of indexes* (ranged over by $I, J, H, K, \ldots$). We refer to the denumerable set of participant names as $\mathfrak{P}$.

Processes, ranged over by $P, Q, R, S, \ldots$, implement the behaviour of participants. In the following and in later definitions the symbol $::=^{coind}$ does express that the productions have to be interpreted *coinductively* and that only *regular* terms are allowed. Then we can adopt in proofs the coinduction style advocated in [21] which, without any loss of formal rigour, promotes readability and conciseness.

Definition 2.1 (Processes) *i)* Action prefixes *are defined by* $\pi ::= \mathsf{p}?\lambda \mid \mathsf{p}!\lambda$.

ii) Processes *are coinductively defined by*

$$P ::=^{coind} \mathbf{0} \mid \Sigma_{i\in I}\pi_i.P_i$$

where $I \neq \emptyset$ and finite, and $\pi_l = \mathsf{q}?\lambda_l$, $\pi_j = \mathsf{q}?\lambda_j$ (resp. $\pi_l = \mathsf{q}!\lambda_l$, $\pi_j = \mathsf{q}!\lambda_j$) imply $\lambda_l \neq \lambda_j$, for any $l, j \in I$ such that $l \neq j$.

In the above definition, $\Sigma_{i\in I}\pi_i.P_i$ stands, as usual, for the summand of processes $\pi_i.P_i$'s. A $\Sigma_{i\in I}\pi_i.P_i$ process represents the nondeterministic choice of one of the actions π_i, after which the process continues as P_i with $i \in I$. As usual, we assume the summand of processes to be commutative and associative. A prefix π can be any input (i.e. of the form $\mathsf{p}?\lambda$) or output (i.e. of the form $\mathsf{p}!\lambda$) action. We use $\mathbf{0}$ to denote the terminated process. For the sake of readability, we omit trailing $\mathbf{0}$'s in processes.

We define the participants of action prefixes by $\mathsf{prt}(\mathsf{p}?\lambda) = \mathsf{prt}(\mathsf{p}!\lambda) = \{\mathsf{p}\}$. Moreover we define the participants of processes by

$$\mathsf{prt}(\mathbf{0}) = \emptyset \qquad \mathsf{prt}(\Sigma_{i\in I}\pi_i.P_i) = \bigcup_{i\in I}\mathsf{prt}(\pi_i) \cup \bigcup_{i\in I}\mathsf{prt}(P_i)$$

By the regularity condition and the finiteness of indexes, $\mathsf{prt}(P)$ is finite for any P.

Processes correspond to inductively defined terms of the calculus MCMP (Mixed Choice Multiparty Sessions) as defined in [31], where the μ-operator is used to describe infinite behaviours. Our use of coinductively defined (possibly) infinite terms enables us to get simpler formalisations and proofs with respect to the use of μ-terms. The latter entail just technicalities that can be dealt with as done in the literature on session types and MPST [19], where such terms are usually considered.

Multiparty sessions are parallel compositions of participant-process pairs, where all participants are different.

Definition 2.2 (Multiparty sessions) Multiparty sessions *are defined by* $\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$ *where $\mathsf{p}_j \neq \mathsf{p}_l$ for $1 \leq j, l \leq n$ and $j \neq l$.*

We assume the standard structural congruence $\equiv$ on multiparty sessions, stating that parallel composition is commutative and associative and has neutral elements $\mathsf{p}[\mathbf{0}]$ for any fresh p. Such a congruence can be inductively defined, as multiparty sessions are. We then write $\mathsf{p}[P] \in \mathbb{M}$ if $\mathbb{M} \equiv \mathsf{p}[P] \parallel \mathbb{M}'$ and $P \neq \mathbf{0}$. Moreover, we define the participants of multiparty sessions by $\mathsf{prt}(\mathbb{M}) = \{\mathsf{p} \mid \mathsf{p}[P] \in \mathbb{M}\}$.

To define the *synchronous operational semantics* of sessions we use an LTS, whose transitions are decorated by *communication labels*, i.e. expressions of the shape $\mathsf{p}\lambda\mathsf{q}$. In the following we use Λ to range over communication labels.

Notation: We use $\pi.P \dotplus Q$ as short for either $\pi.P + Q$ or $\pi.P$. Such a notation enables us to present the

following LTS in a compact and yet formal way, since in our processes – as in [31] – we cannot have unprefixed **0**'s as summands.

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

$$[\textsc{Comm}]\ \frac{}{\mathsf{p}[\mathsf{q}!\lambda.P \dotplus P'] \parallel \mathsf{q}[\mathsf{p}?\lambda.Q \dotplus Q'] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}}$$

Rule [Comm] makes the communication possible: if participant p is enabled to send message λ to participant q which, in turn, is enabled to receive it, the message can be exchanged. This rule is non-deterministic in the choice of messages exchanged. The implementation issues raised by such operational semantics are similar to those for most of the calculi for concurrency and can be dealt with by resorting to suitable coordination protocols. Such issues are however outside the scope of the present paper.

Note that in the above semantics there is no difference between the behaviours of inputs and outputs, while usually a sender freely chooses among all its available messages. In actual communicating systems, messages would also carry values that are abstracted away in SMPS formalisms for the sake of simplicity. The present calculus (as well as its type system) could however be extended to messages with data.

We define *traces* as (possibly infinite) sequences of communication labels. Formally,

$$\sigma ::=^{coind} \varepsilon \mid \Lambda\cdot\sigma$$

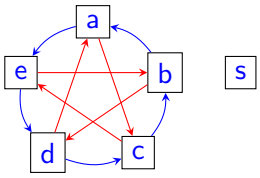
where ε is the empty sequence. When $\sigma = \Lambda_1 \cdot \ldots \cdot \Lambda_n$ $(n \geq 0)$ we write $\mathbb{M} \xrightarrow{\sigma} \mathbb{M}'$ as short for

$$\mathbb{M} \xrightarrow{\Lambda_1} \mathbb{M}_1 \cdots \xrightarrow{\Lambda_n} \mathbb{M}_n = \mathbb{M}'$$

We write $\mathbb{M} \xrightarrow{\sigma}$ with the standard meaning. Moreover, we define the participants of labels by $\mathsf{prt}(\mathsf{p}\lambda\mathsf{q}) = \{\mathsf{p},\mathsf{q}\}$ and the participants of traces – notation $\mathsf{prt}(\sigma)$ – as its obvious extension. We also denote by $\mathscr{L}(\mathbb{M})$ the *set of all labels the session* $\mathbb{M}$ *can emit,* i.e. $\mathscr{L}(\mathbb{M}) = \{\Lambda \mid \mathbb{M} \xrightarrow{\Lambda}\}$.

We present now, in our setting, the leader-election example used in [31] (inspired by [27], in turn inspired by [8]) to make evident the expressive power of mixed choice.

Example 2.4 (Leader election [31]) Five participants $(\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{e})$ interact with the aim of electing a leader. Each of them can send to the next participant, in a circular fashion, the message *leader* in order to ask it to become the leader. If such a communication succeeds, the sender terminates. Of course only two of this sort of communications can succeed. The protocol then allows two, among the remaining three participants, to be able to exchange the *leader* message. The receiver is hence considered as the *elect*ed leader and so it informs the station participant s which, in turn, provides to *del*ete the participant which remained inactive during the previous interactions.
The above behaviour is implemented by the following session:



$$\begin{aligned}\mathbb{E} &= \mathsf{a}[P_\mathsf{a}] \parallel \mathsf{b}[P_\mathsf{b}] \parallel \mathsf{c}[P_\mathsf{c}] \parallel \mathsf{d}[P_\mathsf{d}] \parallel \mathsf{e}[P_\mathsf{e}] \parallel\ \mathsf{s}[P_\mathsf{s}]\\ \text{where}\quad P_\mathsf{a} &= \mathsf{e}!\mathit{leader}\\ &\quad +\mathsf{b}?\mathit{leader}.(\mathsf{c}!\mathit{leader} + \mathsf{d}?\mathit{leader}.\mathsf{s}!\mathit{elect})\\ &\quad +\mathsf{s}?\mathit{del}\end{aligned}$$

and (with some abuse of notation)

$$P_\mathsf{s} = \Sigma_{\mathsf{x}\in\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{e}\}}\left(\mathsf{x}?\mathit{elect}.\Sigma_{\mathsf{x}\in\{\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{e}\}}\mathsf{x}!\mathit{del}\right)$$

Processes of participants b, c, d and e are obtained out of P_a by applying the name substitution $\nu = [\mathsf{a}\mapsto\mathsf{b},\mathsf{b}\mapsto\mathsf{c},\mathsf{c}\mapsto\mathsf{d},\mathsf{d}\mapsto\mathsf{e},\mathsf{e}\mapsto\mathsf{a}]$ as follows:

$$P_\mathsf{b} = P_\mathsf{a}\nu,\ P_\mathsf{c} = P_\mathsf{b}\nu,\ P_\mathsf{d} = P_\mathsf{c}\nu,\ P_\mathsf{e} = P_\mathsf{d}\nu$$

Session $\mathbb{E}$ can be graphically represented by the diagram above on the left, where blue arrows represent the initial possible exchanges of the message *leader* and the red ones the potential further exchanges of such message. The following sequence of reductions is, for instance, the one leading to the election of e:

$$\mathsf{a}\mathit{leader}\mathsf{e}\cdot\mathsf{d}\mathit{leader}\mathsf{c}\cdot\mathsf{c}\mathit{leader}\mathsf{e}\cdot\mathsf{e}\mathit{elect}\mathsf{s}\cdot\mathsf{s}\mathit{del}\mathsf{b} \qquad \diamond$$

Lock Freedom is a relevant property of concurrent systems. We define it in our setting following [26]: roughly, there is always a continuation enabling a participant to communicate whenever it is willing to do so. Lock Freedom entails Deadlock Freedom, since it ensures progress for each participant.

Definition 2.5 (Lock Freedom) *A session* $\mathbb{M}$ *is* lock free *if* $\mathbb{M} \xrightarrow{\sigma} \mathbb{M}'$ *with* σ *finite and* $\mathsf{p} \in \mathsf{prt}(\mathbb{M}')$ *imply* $\mathbb{M}' \xrightarrow{\sigma'\cdot\Lambda}$ *for some* σ' *and* Λ *such that* $\mathsf{p} \notin \mathsf{prt}(\sigma')$ *and* $\mathsf{p} \in \mathsf{prt}(\Lambda)$.

The above definition corresponds to the notion of liveness used in [20] and [22] in a channel-based synchronous communication setting.

3 Modular Multiparty Sessions

"With great [expressive] power comes great responsibility" (Spider-Man's Uncle Ben), since expressive power is often difficult to control and tame. Our aim is to provide a type system for multiparty sessions with mixed choice ensuring, like usual in SMPS, relevant communication properties, together with the guarantee that the overall behaviour of a session faithfully respects what the type assigned to the session, if any, describes. To do that, instead of restricting the expressive power of mixed choice, we decided to consider multiparty sessions that can be – as suggested by a well-known software engineering principle – modularised. A module, in our SMPS setting, is formalised in terms of a subsession inside which participants can freely interact by means of mixed choice. The communications among the modules are instead controlled by imposing them to be performed only by particular participants called "connectors". The processes of the connectors (dubbed connecting processes) must satisfy the restriction that each choice involving a participant not belonging to the module must be between communications with only that participant. Definition 3.1, where $\mathbf{P}$ is the set of module participants, formalises this condition.

Definition 3.1 (Connecting processes) *Given a set of participants* $\mathbf{P}$*, we say that a process* P *is* $\mathbf{P}$-connecting *if for any subprocess of* P*, say* $\Sigma_{i\in I}\pi_i.P_i$*, we have that* $\mathsf{prt}(\pi_j) \notin \mathbf{P}$ *for some* $j \in I$ *implies* $\mathsf{prt}(\pi_i) = \mathsf{prt}(\pi_j)$ *for all* $i, j \in I$.

A connector is hence a participant of a module (represented by the session $\mathbb{M}$ in the definition below) whose process is a connecting process which can interact with the outside of the module.

Definition 3.2 (Connectors) *Let* $\mathsf{p}[P] \in \mathbb{M}$*. We say that the participant* p *is a* connector *for* $\mathbb{M}$ *if* P *is* $\mathsf{prt}(\mathbb{M})$*-connecting and there is* $\mathsf{q} \in \mathsf{prt}(P)$ *such that* $\mathsf{q} \notin \mathsf{prt}(\mathbb{M})$.

The notions of subsession and session partition are at the basis of that of modular session. We say that $\mathbb{M}'$ is a *subsession* of $\mathbb{M}$ and write $\mathbb{M}' \subseteq \mathbb{M}$, whenever $\mathsf{p}[P] \in \mathbb{M}'$ implies $\mathsf{p}[P] \in \mathbb{M}$. A *partition* of a session $\mathbb{M}$ is, as expected, a set of subsessions $\{\mathbb{M}_h\}_{h\in H}$ of $\mathbb{M}$ such that $\mathsf{prt}(\mathbb{M}) = \bigcup_{h\in H}\mathsf{prt}(\mathbb{M}_h)$ and, for all $h,k \in H$, $\mathsf{prt}(\mathbb{M}_h) \cap \mathsf{prt}(\mathbb{M}_k) = \emptyset$. Therefore, $\mathsf{p}[P] \in \mathbb{M}$ implies that there is a unique $k \in H$ such that $\mathsf{p}[P] \in \mathbb{M}_k$.

Definition 3.3 ($\mathscr{P}$-partition) *Let* $\{\mathbb{M}_h\}_{h\in H}$ *be a partition of a session* $\mathbb{M}$*, and let* $\mathscr{P} = \{\mathbf{P}_k\}_{k\in K}$ *be a partition of a finite subset of the set* $\mathfrak{P}$*. We say that* $\{\mathbb{M}_h\}_{h\in H}$ *is a* $\mathscr{P}$-partition *of* $\mathbb{M}$ *if* $H \subseteq K$ *and* $\mathsf{prt}(\mathbb{M}_h) \subseteq \mathbf{P}_h$ *for all* $h \in H$.

It is not difficult to check that, given a session $\mathbb{M}$ and a partition $\mathscr{P} = \{\mathbf{P}_k\}_{k\in K}$ such that $\mathsf{prt}(\mathbb{M}) \subseteq \bigcup_{k\in K} \mathbf{P}_k$, there is a unique *$\mathscr{P}$-partition* of $\mathbb{M}$.

A session is modularisable (with respect to a partition of participants) when it can be partitioned into subsessions that interact only by means of connectors.

Definition 3.4 ($\mathscr{P}$-modularisation) *A* $\mathscr{P}$-modularisation of $\mathbb{M}$ *is a* $\mathscr{P}$-partition $\{\mathbb{M}_h\}_{h\in H}$ *of* $\mathbb{M}$ *such that, for all* $h \in H$*, the following conditions hold*

i) $\mathsf{p}[P] \in \mathbb{M}_h$ *implies that either* p *is a connector of* $\mathbb{M}_h$ *or, for each* $\mathsf{q} \in \mathsf{prt}(P)$, $\mathsf{q} \in \mathsf{prt}(\mathbb{M})$ *implies* $\mathsf{q} \in \mathsf{prt}(\mathbb{M}_h)$*;*

ii) for each connector $\mathsf{p}[P] \in \mathbb{M}_h$ *and each* $\mathsf{q} \in \mathsf{prt}(P) \setminus \mathsf{prt}(\mathbb{M}_h)$*:*
if $\mathsf{q} \in \mathsf{prt}(\mathbb{M}_k)$*, then* q *is a connector for* $\mathbb{M}_k$

In such a case, we say that $\mathbb{M}$ *is* $\mathscr{P}$*-modularisable.*

It is worth noticing that we impose no limit on the number of connectors present in a module, as well as on the number of external connectors a connector can interact with (see the following example). Besides, a session $\mathbb{M}$ is always $\{\mathsf{prt}(\mathbb{M})\}$-modularisable. For example $\mathbb{M} = \mathsf{p}[\mathsf{q}!\lambda + \mathsf{r}!\lambda']$ is $\{\{\mathsf{p}\}\}$-modularisable and its unique module does not contain any connector. Also, given a partition $\mathscr{P}$ and a session $\mathbb{M}$, there exists a unique $\mathscr{P}$-modularisation of $\mathbb{M}$, if any.

Example 3.5 (Modules with multiple connectors) Let us consider $\mathbb{M} = \mathbb{M}_1 \parallel \mathbb{M}_2$,
where $\mathbb{M}_1 = \mathsf{u}[\mathsf{p}?\lambda.\mathsf{q}!\lambda \;+\; \mathsf{q}!\lambda.\mathsf{p}?\lambda] \parallel \mathsf{p}[\mathsf{u}!\lambda.(\mathsf{r}!\lambda_1 + \mathsf{r}?\lambda_2)] \parallel \mathsf{q}[\mathsf{u}?\lambda.(\mathsf{s}!\lambda_1 + \mathsf{s}?\lambda_2)]$
and $\mathbb{M}_2 = \mathsf{v}[\mathsf{r}!\lambda.\mathsf{s}?\lambda \;+\; \mathsf{s}?\lambda.\mathsf{r}!\lambda] \parallel \mathsf{r}[\mathsf{v}?\lambda.(\mathsf{p}?\lambda_1 + \mathsf{p}!\lambda_2)] \parallel \mathsf{s}[\mathsf{v}!\lambda.(\mathsf{q}?\lambda_1 + \mathsf{q}!\lambda_2)]$.
This session is $\{\{\mathsf{u},\mathsf{p},\mathsf{q}\},\{\mathsf{v},\mathsf{r},\mathsf{s}\}\}$-modularisable and it has multiple connectors. In fact, p and q are the connectors for the module $\{\mathsf{u},\mathsf{p},\mathsf{q}\}$, whereas r and s are the connectors for the module $\{\mathsf{v},\mathsf{r},\mathsf{s}\}$. ◇

In actual programming, refining a modularisation enables to enhance parameters like scalability, maintenance, reusability and many more. In the present setting, it also allows for simpler typings. Any modularisation refinement corresponds to a partition refinement.

Definition 3.6 (Refinement) *A partition* $\mathscr{P}$ *refines a partition* $\mathscr{P}'$ *(notation* $\mathscr{P} \sqsubseteq \mathscr{P}'$*) if* $\mathscr{P} = \{\mathbf{P}_h\}_{h\in H}$*,* $\mathscr{P}' = \{\mathbf{P}'_k\}_{k\in K}$ *and for all* $k \in K$ *there is* $H_k \subseteq H$ *such that* $\mathbf{P}'_k = \bigcup_{h\in H_k} \mathbf{P}_h$*.*

As intuitively evident, it is possible to formally show that coarser partitions maintain modularisability.

Lemma 3.7 *If* $\mathbb{M}$ *is* $\mathscr{P}$*-modularisable and* $\mathscr{P}$ *refines* $\mathscr{P}'$*, then* $\mathbb{M}$ *is* $\mathscr{P}'$*-modularisable.*

Proof. It is not difficult to verify that if conditions (i) and (ii) of Definition 3.4 hold for $\mathscr{P}$, then they hold also for $\mathscr{P}'$. □

From the previous lemma, being $\mathbb{M}$ $\{\mathsf{prt}(\mathbb{M})\}$-modularisable, it immediately follows that $\mathbb{M}$ is $\mathscr{P}$-modularisable for all $\mathscr{P}$ such that $\mathsf{prt}(\mathbb{M}) \subseteq \mathbf{P}$ for some $\mathbf{P} \in \mathscr{P}$.

We can build a minimal refined partition, with respect to $\sqsubseteq$, among those modularising a session. We start with $\mathscr{P}_0 = \{\{\mathsf{p}\} \mid \mathsf{p} \in \mathsf{prt}(\mathbb{M})\}$ and we iteratively build $\mathscr{P}_{i+1}$ by replacing in $\mathscr{P}_i$ the two sets $\mathbf{P},\mathbf{P}'$ with the unique set $\mathbf{P}\cup\mathbf{P}'$ if there is $\mathsf{p}[P] \in \mathbb{M}$ such that $\mathsf{p} \in \mathbf{P}$, $\mathsf{q} \in \mathbf{P}' \cap \mathsf{prt}(P)$ and either P is not $\mathbf{P}$-connecting or q is not a connector for the subsession of $\mathbb{M}$ whose set of participants is $\mathbf{P}'$. It is possible to verify that $\mathbb{M}$ is $\mathscr{P}$-modularisable, where $\mathscr{P}$ is the fixed point of this procedure. We show now such a partition to be also the minimum among the partitions modularising a session.

Lemma 3.8 *The minimal partition modularising a session is unique, i.e. a minimum.*

Proof. Assume toward a contradiction that there are two different minimal (w.r.t $\sqsubseteq$) partitions $\mathscr{P} = \{\mathbf{P}_h\}_{h\in H}$ and $\mathscr{P}' = \{\mathbf{P}'_k\}_{k\in K}$ for modularising a session. This implies that there are $h_1, h_2 \in H$ and $k_0 \in K$ such that $\mathsf{p} \in \mathbf{P}_{h_1}$, $\mathsf{q} \in \mathbf{P}_{h_2}$ and $\{\mathsf{p},\mathsf{q}\} \subseteq \mathbf{P}'_{k_0}$. Then also the partition obtained from $\mathscr{P}'$ by replacing the set $\mathbf{P}'_{k_0}$ with the two sets $\mathbf{P}'_{k_0} \cap \mathbf{P}_{h_1}$ and $\mathbf{P}'_{k_0} \cap \mathbf{P}_{h_2}$ modularises the same session. This is clearly a contradiction. □

It is crucial that $\mathscr{P}$-modularisation is preserved by reduction.

Lemma 3.9 *Let $\mathbb{M}$ be $\mathscr{P}$-modularisable and let $\mathbb{M} \xrightarrow{\Lambda} \mathbb{M}'$. Then also $\mathbb{M}'$ is $\mathscr{P}$-modularisable.*

Proof. Notice that conditions (i) and (ii) of Definition 3.4 are invariant by reduction. In fact a participant p which is a connector in $\mathbb{M}$ is either a connector in $\mathbb{M}'$ too or it has a process whose participants all belong to the subsession containing p in the $\mathscr{P}$-modulation of $\mathbb{M}'$. Therefore a $\mathscr{P}$-modularisation of $\mathbb{M}$ is also a $\mathscr{P}$-modularisation of $\mathbb{M}'$. □

We can notice that, if $\mathscr{P}$ is the minimal partition for modularising $\mathbb{M}$ and $\mathbb{M} \xrightarrow{\Lambda} \mathbb{M}'$, in general $\mathscr{P}$ is not the minimal partition for modularising $\mathbb{M}'$. In fact $\mathsf{prt}(\mathbb{M}')$ can be a proper subset of $\mathsf{prt}(\mathbb{M})$ and a process in $\mathbb{M}$ which is not a connector can reduce to a process which is a connector in $\mathbb{M}'$. For example the minimal partition of $\mathbb{M} \equiv \mathsf{p}[\mathsf{q}!\lambda.\mathsf{r}!\lambda + \mathsf{r}?\lambda'.(\mathsf{q}!\lambda_1 + \mathsf{q}?\lambda_2)] \parallel \mathsf{q}[\mathsf{p}?\lambda + \mathsf{p}?\lambda_1 + \mathsf{p}!\lambda_2] \parallel \mathsf{r}[\mathsf{p}?\lambda + \mathsf{p}!\lambda']$ is $\{\{\mathsf{p},\mathsf{q},\mathsf{r}\}\}$, since p is not a connector. But $\mathbb{M} \xrightarrow{\mathsf{r}\lambda'\mathsf{p}} \mathsf{p}[\mathsf{q}!\lambda_1 + \mathsf{q}?\lambda_2] \parallel \mathsf{q}[\mathsf{p}?\lambda + \mathsf{p}?\lambda_1 + \mathsf{p}!\lambda_2]$ and the minimal partition of this last session is $\{\{\mathsf{p}\},\{\mathsf{q}\}\}$, since p and q are connectors and r disappeared.

Example 3.10 (Modular election) We consider three "local" elections, all managed like in the Example 2.4. The names of the participants of the three local election are like the ones in the Example 2.4, but indexed with indexes in $\{1,2,3\}$. We also consider a further "global election" with participants w_1, w_2 and w_3 and global station gs. Such an election follows a protocol similar to that of the local elections (but simpler, since only three participants do compete for leadership). It can be seen as an election among the winners of the local elections. In the present example the "local leaders", once they are elected, are informed whether they have been elected also "global leader" or not. The above sketched global behaviour is implemented by the following session $\mathbb{E}^{gl}$ made of four subsessions: three local elections $(\mathbb{E}_1, \mathbb{E}_2, \mathbb{E}_3)$ and one global election $\mathbb{G}$ among the "local" winners.

The processes of the participants, are fairly similar to the ones in the Example 2.4 to which a part is added implementing the communication to the local leaders of whether they are also global leader or not.

$$\mathbb{E}^{gl} \quad = \quad \mathbb{E}_1 \parallel \mathbb{E}_2 \parallel \mathbb{E}_3 \parallel \mathbb{G}$$

where, for $1 \le i \le 3$,

$$\mathbb{E}_i \quad = \quad \mathsf{a}_i[P_{\mathsf{a}_i}] \parallel \mathsf{b}_i[P_{\mathsf{b}_i}] \parallel \mathsf{c}_i[P_{\mathsf{c}_i}] \parallel \mathsf{d}_i[P_{\mathsf{d}_i}] \parallel \mathsf{e}_i[P_{\mathsf{e}_i}] \parallel \mathsf{s}_i[P_{\mathsf{s}_i}]$$

with $P_{\mathsf{a}_i} = \mathsf{e}_i!\mathit{leader} + \mathsf{b}_i?\mathit{leader}.\Big(\mathsf{c}_i!\mathit{leader} + \mathsf{d}_i?\mathit{leader}.\mathsf{s}_i!\mathit{elect}.\big(\mathsf{s}_i?\mathit{gleader} + \mathsf{s}_i?\mathit{no}\big)\Big) + \mathsf{s}_i?\mathit{del}$

and $P_{\mathsf{s}_i} = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}?\mathit{elect}.\big(\mathsf{gs}?\mathit{gleader}.\mathsf{x}!\mathit{gleader}.Q_i + \mathsf{gs}?\mathit{no}.\mathsf{x}!\mathit{no}.Q_i\big)$

with $Q_i = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}!\mathit{del}$

and with processes of participants b_i, c_i, d_i and e_i obtained out of P_{a_i} by applying the name substitution

$\nu_i = [\mathsf{a}_i \mapsto \mathsf{b}_i, \mathsf{b}_i \mapsto \mathsf{c}_i, \mathsf{c}_i \mapsto \mathsf{d}_i, \mathsf{d}_i \mapsto \mathsf{e}_i, \mathsf{e}_i \mapsto \mathsf{a}_i]$ as in Example 2.4

and

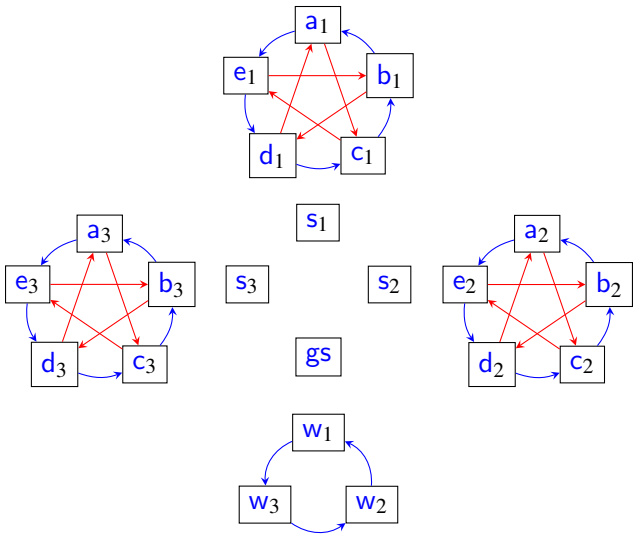


Figure 1: The three local elections and the global one of Example 3.10.

$$\mathbb{G} \;=\; \mathsf{w}_1[P_{\mathsf{w}_1}] \parallel \mathsf{w}_2[P_{\mathsf{w}_2}] \parallel \mathsf{w}_3[P_{\mathsf{w}_3}] \;\parallel\; \mathsf{gs}[P_{\mathsf{gs}}]$$

with $P_{\mathsf{w}_i} = \mathsf{w}_{i+2}!\mathit{leader} + \mathsf{w}_{i+1}?\mathit{leader}.\mathsf{gs}!\mathit{gleader} + \mathsf{gs}?\mathit{del}$

and $P_{\mathsf{gs}} = \Sigma_{i\in\{1,2,3\}}\mathsf{w}_i?\mathit{gleader}.\mathsf{s}_i!\mathit{gleader}.\mathsf{s}_{i+1}!\mathit{no}.\mathsf{s}_{i+2}!\mathit{no}.\big(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del}\big)$
where all the indexes above have to be considered modulo 3, plus 1.

It is not difficult to check that $\mathbb{E}^{gl}$ is $\mathscr{P}$-modularisable for $\mathscr{P} = \{\mathsf{prt}(\mathbb{E}_1), \mathsf{prt}(\mathbb{E}_2), \mathsf{prt}(\mathbb{E}_3), \mathsf{prt}(\mathbb{G})\}$, where $\mathsf{s}_1, \mathsf{s}_2, \mathsf{s}_3$ and gs are the connectors for, respectively, the modules $\mathbb{E}_1$, $\mathbb{E}_2$, $\mathbb{E}_3$ and $\mathbb{G}$. ◇

4 A Type System for Modular Sessions

Global types are used to represent the overall behaviour of multiparty sessions. Here we use a notion of global type similar to the one in MPST but, following SMPS, we define them coinductively, as possibly infinite regular terms.

Definition 4.1 (Global types) Global types *are coinductively defined by:*

$$\mathsf{G} ::=^{coind} \mathtt{End} \;\mid\; \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$$

where $I \neq \emptyset$ and, for any $j,l \in I$ such that $j \neq l$, $\Lambda_i = \mathsf{p}\lambda_i\mathsf{q}$ and $\Lambda_j = \mathsf{p}\lambda_j\mathsf{q}$ imply $\lambda_j \neq \lambda_l$.

We define the participants of global types by $\mathsf{prt}(\mathtt{End}) = \emptyset$ and $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \bigcup_{i\in I}\mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I}\mathsf{prt}(\mathsf{G}_i)$. By the regularity condition, $\mathsf{prt}(\mathsf{G})$ is finite for any G. As usual, trailing $\mathtt{End}$'s will be omitted.

As mentioned in the Introduction, in standard SMPS, typing rules take into account single interactions between pairs of participants. In our mixed-choice setting, that approach would allow to type only sessions whose independent parts were intrinsically sequential, so ruling out protocols like the leader election and the modular election. We hence consider a typing rule where all the interactions involving

the participants of a single module $\widehat{\mathbb{M}}$ (so including also the communications of the connectors of $\widehat{\mathbb{M}}$ with other modules) are taken into account. Such a set of interactions is formalised through the notion of coherent set of communication labels. A module $\widehat{\mathbb{M}}$ is therefore indirectly represented in the rule in terms of its corresponding coherent set and referred to, when necessary, as the *witness* of such a set.

Definition 4.2 (Coherent set of communication labels) *A set of labels $\{\Lambda_i\}_{i\in I}$ is* $\mathscr{P}$-coherent for $\mathbb{M}$ *if $\mathbb{M}$ is $\mathscr{P}$-modularisable and there exists an element $\widehat{\mathbb{M}}$ of the (unique) $\mathscr{P}$-modularisation of $\mathbb{M}$ such that*

$$\{\Lambda_i\}_{i\in I} = \{\Lambda \in \mathscr{L}(\mathbb{M}) \mid \mathsf{prt}(\Lambda) \cap \mathsf{prt}(\widehat{\mathbb{M}}) \neq \emptyset\}$$

The $\widehat{\mathbb{M}}$ above is called witness *for the $\mathscr{P}$-coherence of $\{\Lambda_i\}_{i\in I}$.*

Example 4.3 (Witness) Let us consider $\mathbb{M}'_1 = \mathsf{u}[\mathsf{q}!\lambda] \parallel \mathsf{p}[\mathsf{r}!\lambda_1 + \mathsf{r}?\lambda_2] \parallel \mathsf{q}[\mathsf{u}?\lambda.(\mathsf{s}!\lambda_1 + \mathsf{s}?\lambda_2)]$ and $\mathbb{M}_2 = \mathsf{v}[\mathsf{r}!\lambda.\mathsf{s}?\lambda \ + \ \mathsf{s}?\lambda.\mathsf{r}!\lambda] \parallel \mathsf{r}[\mathsf{v}?\lambda.(\mathsf{p}?\lambda_1 + \mathsf{p}!\lambda_2)] \parallel \mathsf{s}[\mathsf{v}!\lambda.(\mathsf{q}?\lambda_1 + \mathsf{q}!\lambda_2)]$. The session $\mathbb{M}'_1 \parallel \mathbb{M}_2$ can be obtained by reducing the session of Example 3.5. It is still $\{\{\mathsf{u},\mathsf{p},\mathsf{q}\},\{\mathsf{v},\mathsf{r},\mathsf{s}\}\}$-modularisable and $\mathbb{M}'_1$ is the witness for the $\{\{\mathsf{u},\mathsf{p},\mathsf{q}\},\{\mathsf{v},\mathsf{r},\mathsf{s}\}\}$-coherent set of labels $\{\mathsf{u}\lambda\mathsf{q},\ \mathsf{p}\lambda_1\mathsf{r},\ \mathsf{r}\lambda_2\mathsf{p}\}$. ◇

Here and in the following, the double line indicates that the rules are interpreted coinductively.

Definition 4.4 (Type system) *The type system $\vdash^{\mathscr{P}}$ is defined by the following axiom and rule, where sessions are considered modulo structural congruence:*

$$[\textsc{End}]\ \frac{}{\mathsf{End} \vdash^{\mathscr{P}} \mathsf{p}[\mathbf{0}]}$$

$$[\textsc{TComm}]\ \frac{\begin{array}{c}\mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i \qquad \mathsf{G}_i \vdash^{\mathscr{P}} \mathbb{M}_i \qquad \forall i \in I \neq \emptyset \\ \{\Lambda_i\}_{i\in I} \textit{ is } \mathscr{P}\textit{-coherent for } \mathbb{M} \qquad \mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \mathsf{prt}(\mathbb{M})\end{array}}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \vdash^{\mathscr{P}} \mathbb{M}}$$

It is not difficult to check that we can derive the global type $\mathsf{u}\lambda\mathsf{q}.\mathsf{p}\lambda\mathsf{u}.\mathsf{G}_1 + \mathsf{p}\lambda\mathsf{u}.\mathsf{u}\lambda\mathsf{q}.\mathsf{G}_1$, where $\mathsf{G}_1 = \mathsf{v}\lambda\mathsf{r}.\mathsf{s}\lambda\mathsf{v}.\mathsf{G}_2 + \mathsf{s}\lambda\mathsf{v}.\mathsf{v}\lambda\mathsf{r}.\mathsf{G}_2$, $\mathsf{G}_2 = \mathsf{p}\lambda_1\mathsf{r}.\mathsf{G}_3 + \mathsf{r}\lambda_2\mathsf{p}.\mathsf{G}_3$, and $\mathsf{G}_3 = \mathsf{q}\lambda_1\mathsf{s} + \mathsf{s}\lambda_2\mathsf{q}$, for the session of Example 3.5.

The condition "$\{\Lambda_i\}_{i\in I}$ is $\mathscr{P}$-coherent for $\mathbb{M}$" is essential to get Subject Reduction. In fact, by allowing any subset of $\mathscr{L}(\mathbb{M})$ as $\{\Lambda_i\}_{i\in I}$ in [TComm], we could derive $\mathsf{p}\lambda'\mathsf{r}.\mathsf{p}\lambda\mathsf{q} \vdash^{\mathscr{P}} \mathbb{M}_0$ for

$$\mathbb{M}_0 \equiv \mathsf{p}[\mathsf{q}!\lambda + \mathsf{r}!\lambda'.\mathsf{q}!\lambda] \parallel \mathsf{q}[\mathsf{p}?\lambda] \parallel \mathsf{r}[\mathsf{p}?\lambda']$$

regardless of $\mathscr{P}$. However, we would also have $\mathbb{M}_0 \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r}[\mathsf{p}?\lambda']$, with $\mathsf{r}[\mathsf{p}?\lambda']$ untypable. The above example also shows that, in order to get Subject Reduction it is necessary that, at any moment, a connector can interact with one other connector only. Let us assume to relax Definition 3.1 as follows

> *Given a set of participants* $\mathbf{P}$*, we say that a process P is* $\mathbf{P}$-connecting *if for any subprocess of P, say $\Sigma_{i\in I}\pi_i.P_i$, we have that $\mathsf{prt}(\pi_j) \notin \mathbf{P}$ for some $j \in I$ implies $\mathsf{prt}(\pi_i) \notin \mathbf{P}$ for all $i \in I$.*

This would imply $\mathbb{M}_0$ above to be $\{\{\mathsf{p}\},\{\mathsf{q}\},\{\mathsf{r}\}\}$-modularisable and all the participants would turn out to be connectors in their respective modules. Hence $\mathbb{M}_0$ would be $\vdash^{\{\{\mathsf{p}\},\{\mathsf{q}\},\{\mathsf{r}\}\}}$ typable, whereas $\mathsf{r}[\mathsf{p}?\lambda']$ would not.

The condition "$\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \mathsf{prt}(\mathbb{M})$" is necessary to get Lock Freedom. For example without this condition we could derive $\mathsf{G} \vdash^{\{\{\mathsf{p}\},\{\mathsf{q}\},\{\mathsf{r}\}\}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathsf{r}[\mathsf{s}!\lambda]$ with $P = \mathsf{q}!\lambda.P$, $Q = \mathsf{p}?\lambda.Q$ and $\mathsf{G} = \mathsf{p}\lambda\mathsf{q}.\mathsf{G}$. For what concerns the condition $I \neq \emptyset$, let us consider one of our previous examples, namely $\mathbb{M} = \mathsf{p}[\mathsf{q}!\lambda + \mathsf{r}!\lambda']$. Such a session can be uniquely modularised with itself as possible module and it is not lock free. In fact it is not typable, since its unique coherent set is empty.

It can be proved that $\mathscr{P}$-coherence of a label set for a session is preserved by reducing the session with a label not belonging to the $\mathscr{P}$-coherent set, see Lemma 5.6.

The type system is decidable, since processes and global types are regular, and there is only a finite number of partitions for the participants of a session. More interesting, typability of a session does depend on the choice neither of the $\mathscr{P}$-coherent sets nor of the partition $\mathscr{P}$ (see, respectively, Theorems 5.11 and 5.12).

We define the semantics of global types via a coinductive formal system, as done first in [6]. Such a coinductive definition enables to take into account global types containing branches, where some communications can be indefinitely procrastinated, see Example 4.7. In order to do that, it is handy to associate to a global type the set of communication labels which might (not necessarily) decorate its transitions. We dub them capabilities of the global type.

Definition 4.5 (Capabilities) *Capabilities of global types are defined by:*

$$cap(\mathtt{End}) = \emptyset \quad cap(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \{\Lambda_i\}_{i\in I} \cup \bigcup_{i\in I} cap(\mathsf{G}_i)$$

Definition 4.6 (LTS for global types) *The* labelled transition system (LTS) for global types *is specified by the following axiom and rule:*

$$[\text{E-COMM}]\ \frac{}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \xrightarrow{\Lambda_j} \mathsf{G}_j} \qquad j \in I$$

$$[\text{I-COMM}]\ \frac{\mathsf{G}_i \xrightarrow{\Lambda} \mathsf{G}_i' \quad \Lambda \in cap(\mathsf{G}_i) \quad \mathsf{prt}(\Lambda) \cap \mathsf{prt}(\Lambda_i) = \emptyset \quad \forall i \in I}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \xrightarrow{\Lambda} \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i'}$$

Axiom [E-COMM] formalises the fact that, in a session exposing the behaviour $\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$, the communication labelled Λ_j for any $j \in I$ can happen. When such a communication is actually performed, the resulting session will expose the behaviour G_j.

Rule [I-COMM] enables to describe independent and concurrent communications, even if global types apparently look like sequential descriptions of sessions' overall behaviours. In fact, behaviours involving participants ready to interact with each other uniformly in all branches of a global type, can do that if neither of them is involved in an interaction appearing at top level in the global type. The condition $\Lambda \in cap(\mathsf{G}_i)$ in Rule [I-COMM] is needed because such a rule is coinductive. In fact, without such a condition, we could get the following infinite derivation for $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}$ with $\mathsf{G} = \mathsf{r}\lambda'\mathsf{s}.\mathsf{G}$:

$$\mathscr{D} = \frac{\mathscr{D}}{\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}}\ [\text{I-COMM}]$$

Example 4.7 (Use of coinduction in Rule [I-COMM]) As shown in [6] the coinductive formulation of Rule [I-COMM] allows to get $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'$, where $\mathsf{G} = \mathsf{r}\lambda_1\mathsf{s}.\mathsf{G} + \mathsf{r}\lambda_2\mathsf{s}.\mathsf{p}\lambda\mathsf{q}$ and $\mathsf{G}' = \mathsf{r}\lambda_1\mathsf{s}.\mathsf{G}' + \mathsf{r}\lambda_2\mathsf{s}$. The inductive definition of this rule does not allow the shown transition. ◇

Example 4.8 (Typing the modular election session) In Figure 2 we provide a typing for the modular session $\mathbb{E}^{gl}$ of Example 3.10 in the type system $\vdash^{\mathscr{P}}$ with

$$\mathscr{P} = \bigcup_{i\in\{1,2,3\}}\{\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i,\mathsf{s}_i\}\} \cup \{\{\mathsf{w}_1,\mathsf{w}_2,\mathsf{w}_3,\mathsf{gs}\}\}$$

We use colours for representing reduced participants. In particular, a participant has the colour
□ in its initial state; ■ after one interaction; ■ after two interactions;
■ after three interactions; ■ after four interactions; ■ when terminated.

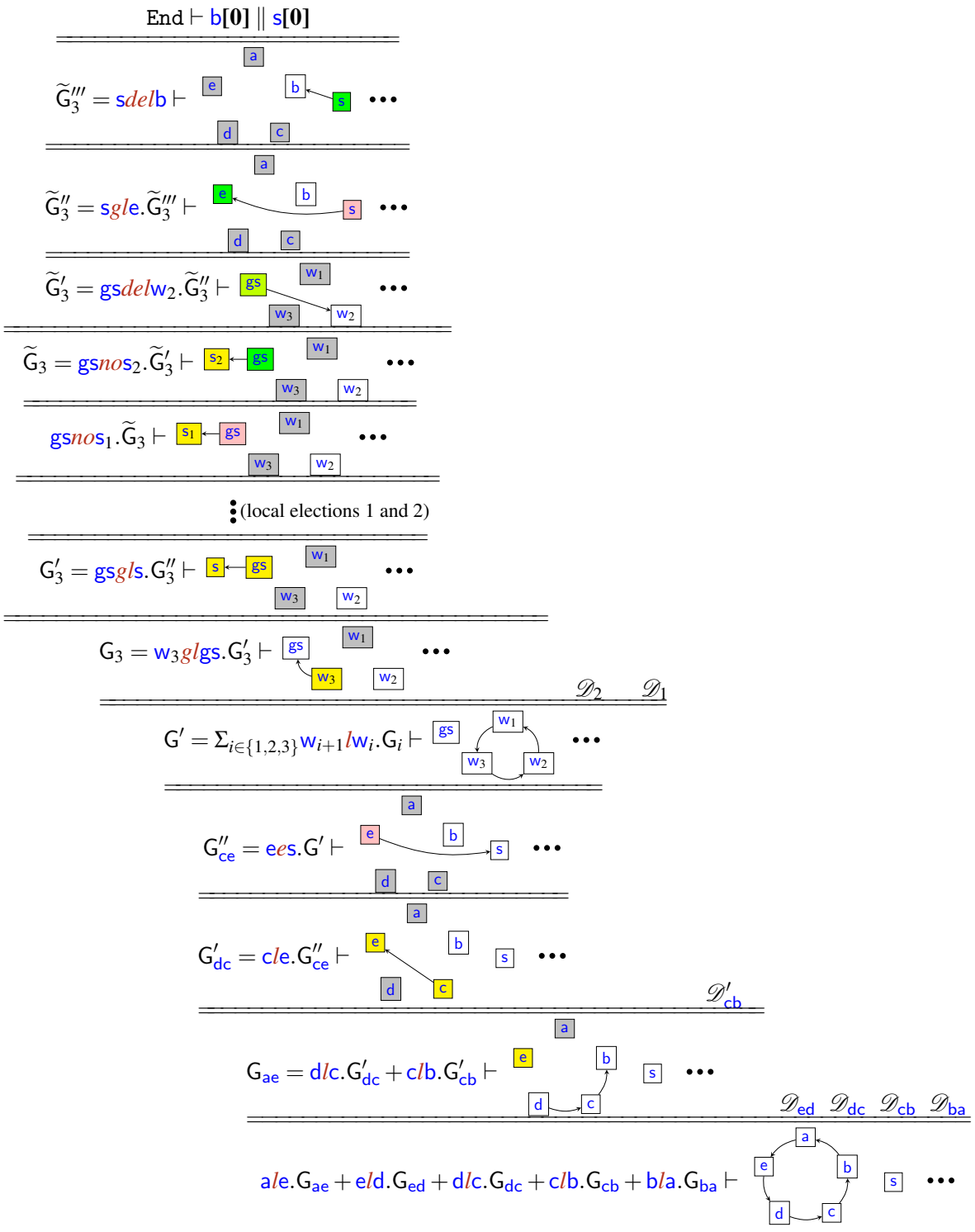


Figure 2: A type derivation for Example 3.10.

For instance:
$\boxed{\mathsf{gs}} = \mathsf{gs}[\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i?\mathit{gleader}.\mathsf{s}_i!\mathit{gleader}.\mathsf{s}_{i+1}!\mathit{no}.\mathsf{s}_{i+2}!\mathit{no}.(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del})]$
$\boxed{\mathsf{gs}} = \mathsf{gs}[\mathsf{s}_3!\mathit{gleader}.\mathsf{s}_1!\mathit{no}.\mathsf{s}_2!\mathit{no}.(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del})]$ $\boxed{\mathsf{gs}} = \mathsf{gs}[\mathsf{s}_1!\mathit{no}.\mathsf{s}_2!\mathit{no}.(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del})]$
$\boxed{\mathsf{gs}} = \mathsf{gs}[\mathsf{s}_2!\mathit{no}.(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del})]$ $\boxed{\mathsf{gs}} = \mathsf{gs}[\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del}]$ $\boxed{\mathsf{gs}} = \mathsf{gs}[\mathbf{0}]$

In Figure 2, arrows do connect pair of participants forming a redex. Moreover, in the conclusion of the rules we show only the participants of the module containing the coherent set of reductions and, in case, the "external" connectors. The rest of the session will be denoted by " $\bullet\bullet\bullet$". Also, the figure shows only one branch of the typing derivation tree: the one concerning the global election of e_3.

For the sake of readability, l and gl are abbreviation for, respectively, *leader* and *gleader*. Moreover, $\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{e}$ and s stand for $\mathsf{a}_3,\mathsf{b}_3,\mathsf{c}_3,\mathsf{d}_3,\mathsf{e}_3$ and s_3. ◇

5 Properties

A subsession of the shape $\mathsf{p}[\mathsf{q}!\lambda.P \dotplus P'] \parallel \mathsf{q}[\mathsf{p}?\lambda.Q \dotplus Q']$ is called a *redex* and $\mathsf{p}[P] \parallel \mathsf{q}[Q]$ is the *contractum* of the redex. In a transition labelled by $\mathsf{p}\lambda\mathsf{q}$ both the redex and the contractum are uniquely determined.

Lemma 5.1 *If* $\mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'$, *then there exists a unique redex* $\mathsf{p}[\mathsf{q}!\lambda.P \dotplus P'] \parallel \mathsf{q}[\mathsf{p}?\lambda.Q \dotplus Q']$ *such that*

$$\mathbb{M} \equiv \mathsf{p}[\mathsf{q}!\lambda.P \dotplus P'] \parallel \mathsf{q}[\mathsf{p}?\lambda.Q \dotplus Q'] \parallel \mathbb{M}''$$

and $\mathbb{M}' \equiv \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}''$.

Proof. Immediate by the definition of session LTS. □

Rule [Comm] in Definition 2.3 entails an easy relation between the participants connected by reductions in a session.

Lemma 5.2 *If* $\mathbb{M} \xrightarrow{\Lambda} \mathbb{M}'$, *then* $\mathsf{prt}(\mathbb{M}) = \mathsf{prt}(\Lambda) \cup \mathsf{prt}(\mathbb{M}')$.

It is not difficult to check that the participants of a session and of its global type are the same.

Lemma 5.3 *If* $\mathsf{G} \vdash^{\mathscr{P}} \mathbb{M}$, *then* $\mathsf{prt}(\mathsf{G}) = \mathsf{prt}(\mathbb{M})$.

The following technical lemma relating capabilities and possible reductions of a global type will be handy later on.

Lemma 5.4 *If* $\mathsf{G} \xrightarrow{\Lambda} \mathsf{G}'$, *then* $\Lambda \in \mathit{cap}(\mathsf{G})$.

Proof. By cases on the applied axiom/rule justifying $\mathsf{G} \xrightarrow{\Lambda} \mathsf{G}'$. If this is [E-Comm], then $\mathsf{G} = \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$ and $\Lambda = \Lambda_j$ for some $j \in I$ and $\Lambda_j \in \mathit{cap}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i)$ by Definition 4.5.

Otherwise, $\mathsf{G} = \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$ and $\mathsf{G}' = \Sigma_{i\in I}\Lambda_i.\mathsf{G}'_i$ by Rule [I-Comm], where $\mathsf{G}_i \xrightarrow{\Lambda} \mathsf{G}'_i$, $\Lambda \in \mathit{cap}(\mathsf{G}_i)$ and $\mathsf{prt}(\Lambda) \cap \mathsf{prt}(\Lambda_i) = \emptyset$ for all $i \in I$. This implies $\Lambda \in \mathit{cap}(\mathsf{G})$, since $\bigcup_{i\in I} \mathit{cap}(\mathsf{G}_i) \subseteq \mathit{cap}(\mathsf{G})$ by Definition 4.5. □

For showing Subject Reduction it is crucial to ensure that the $\mathscr{P}$-coherence of a set of labels is preserved by reducing a label not belonging to this set, see Lemma 5.6 whose proof uses Lemma 5.5 below.

Lemma 5.5 *Let* $\{\Lambda_i\}_{i\in I}$ *be* $\mathscr{P}$*-coherent for* $\mathbb{M}$ *and let* $\Lambda \in \mathscr{L}(\mathbb{M})$. *Moreover, let* $\Lambda \neq \Lambda_i$ *for all* $i \in I$. *Then* $\mathsf{prt}(\Lambda) \cap \mathsf{prt}(\Lambda_i) = \emptyset$ *for all* $i \in I$.

Proof. By definition of coherence (Definition 4.2), we have a subsession $\widehat{\mathbb{M}}$ of $\mathbb{M}$ witnessing the $\mathscr{P}$-coherence of $\{\Lambda_i\}_{i\in I}$. By contradiction, let us assume $\mathsf{prt}(\Lambda)\cap\mathsf{prt}(\Lambda_j)\neq\emptyset$ for some $j\in I$. By definition of $\mathscr{P}$-modularisation, this implies that $\mathsf{prt}(\Lambda_j)$ contains a connector p of $\widehat{\mathbb{M}}$. Let $\mathsf{prt}(\Lambda_j)=\{\mathsf{p},\mathsf{q}\}$ with $\mathsf{q}\notin\mathsf{prt}(\widehat{\mathbb{M}})$ and $\mathsf{prt}(\Lambda)=\{\mathsf{q},\mathsf{r}\}$. Then $\mathsf{q}\in\mathsf{prt}(\mathbb{M})$ and the process Q of q must have a choice between a communication with p and a communication with r. But this is impossible, since q must be a connector for some subsession $\widehat{\mathbb{M}}'$ of $\mathbb{M}$ by condition (ii) of Definition 3.4 and then the process Q must be $\mathsf{prt}(\widehat{\mathbb{M}}')$-connecting by Definition 3.2. □

Lemma 5.6 (Coherence preservation) *Let $\{\Lambda_i\}_{i\in I}$ be $\mathscr{P}$-coherent for $\mathbb{M}$ and let $\mathbb{M}\xrightarrow{\Lambda}\mathbb{M}'$. Then $\Lambda\notin\{\Lambda_i\}_{i\in I}$ implies that $\{\Lambda_i\}_{i\in I}$ is $\mathscr{P}$-coherent for $\mathbb{M}'$ as well.*

Proof. Let $\widehat{\mathbb{M}}$ be the subsession of $\mathbb{M}$ witnessing the $\mathscr{P}$-coherence of $\{\Lambda_i\}_{i\in I}$ for $\mathbb{M}$. From $\Lambda\notin\{\Lambda_i\}_{i\in I}$ and Lemma 5.5 we get that $\mathsf{prt}(\Lambda)\cap\mathsf{prt}(\Lambda_i)=\emptyset$ for all $i\in I$. This implies that the reduction $\xrightarrow{\Lambda}$ cannot affect any reduction with label in $\{\Lambda_i\}_{i\in I}$. Hence $\widehat{\mathbb{M}}$ is a witness of the $\mathscr{P}$-coherence of $\{\Lambda_i\}_{i\in I}$ also for $\mathbb{M}'$. □

Notice how the conditions on connectors (Definitions 3.1 and 3.2) are crucial in getting the property that $\mathsf{prt}(\Lambda)\cap\mathsf{prt}(\Lambda_i)=\emptyset$ for all $i\in I$ in the above result of coherence preservation. If we allowed connectors to communicate with external partners having unrestricted processes we could consider the $\{\{\mathsf{p}\},\{\mathsf{r},\mathsf{s}\}\}$-modularisation of $\mathbb{M}=\mathsf{p}[\mathsf{r}!\lambda]\parallel\mathsf{r}[\mathsf{p}?\lambda+\mathsf{s}?\lambda]\parallel\mathsf{s}[\mathsf{r}!\lambda]$. In such a case, the set $\{\mathsf{p}\lambda\mathsf{r}\}$ would be $\{\{\mathsf{p}\},\{\mathsf{r},\mathsf{s}\}\}$-coherent with witness $\mathbb{M}'=\mathsf{p}[\mathsf{r}!\lambda]$. However, we would also have that $\mathbb{M}\xrightarrow{\mathsf{s}\lambda\mathsf{r}}\mathbb{M}'$, but $\{\mathsf{p}\lambda\mathsf{r}\}$ would not be $\{\{\mathsf{p}\},\{\mathsf{r},\mathsf{s}\}\}$-coherent for $\mathbb{M}'$, since $\mathsf{p}\lambda\mathsf{r}\notin\mathscr{L}(\mathbb{M}')$.

Theorem 5.7 (Subject Reduction) *If $\mathsf{G}\vdash^{\mathscr{P}}\mathbb{M}$ and $\mathbb{M}\xrightarrow{\Lambda}\mathbb{M}'$, then $\mathsf{G}'\vdash^{\mathscr{P}}\mathbb{M}'$ and $\mathsf{G}\xrightarrow{\Lambda}\mathsf{G}'$ for some G'.*

Proof. By coinduction on the derivation of $\mathsf{G}\vdash^{\mathscr{P}}\mathbb{M}$. Let $\Lambda=\mathsf{p}\lambda\mathsf{q}$. By Lemma 5.1, if $\mathbb{M}\xrightarrow{\Lambda}\mathbb{M}'$, then there exists a unique redex

$$\mathscr{R}=\mathsf{p}[\mathsf{q}!\lambda.P\,\vdots\,P']\parallel\mathsf{q}[\mathsf{p}?\lambda.Q\,\vdots\,Q']$$

such that $\mathbb{M}\equiv\mathscr{R}\parallel\mathbb{M}''$ and $\mathbb{M}'\equiv\mathsf{p}[P]\parallel\mathsf{q}[Q]\parallel\mathbb{M}''$ for some $\mathbb{M}''$. By the hypothesis that $\mathsf{G}\vdash^{\mathscr{P}}\mathbb{M}$ we know that G is of the form $\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$ and the derivation ends by

$$[\text{TComm}]\ \frac{\mathbb{M}\xrightarrow{\Lambda_i}\mathbb{M}_i\qquad\mathsf{G}_i\vdash^{\mathscr{P}}\mathbb{M}_i\qquad\forall i\in I\neq\emptyset\qquad\{\Lambda_i\}_{i\in I}\text{ is }\mathscr{P}\text{-coherent for }\mathbb{M}\qquad\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i)=\mathsf{prt}(\mathbb{M})}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i\vdash^{\mathscr{P}}\mathbb{M}}$$

We proceed by distinguishing the two possible following cases.

Case $\mathsf{p}\lambda\mathsf{q}=\Lambda_j$ for some $j\in I$. By the premises of the rule, we have $\mathbb{M}\xrightarrow{\Lambda}\mathbb{M}_j$ and $\mathsf{G}_j\vdash^{\mathscr{P}}\mathbb{M}_j$, where $\mathbb{M}'=\mathbb{M}_j$. Moreover, it immediately follows that $\mathsf{G}\xrightarrow{\Lambda}\mathsf{G}_j$ by Axiom [E-Comm].

Case $\mathsf{p}\lambda\mathsf{q}\neq\Lambda_i$ for all $i\in I$. We have that, for all $i\in I$, $\mathbb{M}_i\equiv\mathscr{R}\parallel\mathbb{M}'_i$ for some $\mathbb{M}'_i$. Hence we get that $\mathbb{M}_i\xrightarrow{\Lambda}\mathsf{p}[P]\parallel\mathsf{q}[Q]\parallel\mathbb{M}'_i$ for all $i\in I$. Moreover, for all $i\in I$, $\mathbb{M}''\xrightarrow{\Lambda_i}\mathbb{M}'_i$. By the coinduction hypothesis on the premises of the rule, we have that, for all $i\in I$, $\mathsf{G}'_i\vdash^{\mathscr{P}}\mathsf{p}[P]\parallel\mathsf{q}[Q]\parallel\mathbb{M}'_i$ and $\mathsf{G}_i\xrightarrow{\Lambda}\mathsf{G}'_i$ for some G'_i. Now, by Lemma 5.4, we get that $\Lambda\in cap(\mathsf{G}_i)$ for all $i\in I$, hence we have that $\mathsf{G}\xrightarrow{\Lambda}\Sigma_{i\in I}\Lambda_i.\mathsf{G}'_i$ by Rule [I-Comm]. From $\mathbb{M}\xrightarrow{\Lambda}\mathbb{M}'\equiv\mathsf{p}[P]\parallel\mathsf{q}[Q]\parallel\mathbb{M}''$ and $\mathbb{M}''\xrightarrow{\Lambda_i}\mathbb{M}'_i$ we get $\mathbb{M}'\xrightarrow{\Lambda_i}\mathsf{p}[P]\parallel\mathsf{q}[Q]\parallel\mathbb{M}'_i$ for all $i\in I$, which imply, by Lemma 5.2,

$$\mathsf{prt}(\mathbb{M}')=\bigcup_{i\in I}\mathsf{prt}(\Lambda_i)\cup\bigcup_{i\in I}\mathsf{prt}(\mathsf{p}[P]\parallel\mathsf{q}[Q]\parallel\mathbb{M}'_i)$$

By Lemma 5.3, $\mathsf{G}'_i \vdash^{\mathscr{P}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}'_i$ gives $\mathsf{prt}(\mathsf{G}'_i) = \mathsf{prt}(\mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}'_i)$ for all $i \in I$. Hence $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}'_i) = \bigcup_{i\in I}\mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I}\mathsf{prt}(\mathsf{G}'_i) = \mathsf{prt}(\mathbb{M}')$. Moreover, from Lemma 5.6 and $\mathsf{p}\lambda\mathsf{q} \neq \Lambda_i$ for all $i \in I$, it follows that $\{\Lambda_i\}_{i\in I}$ is $\mathscr{P}$-coherent for $\mathbb{M}'$ as well. Therefore Rule [TComm] applies, namely

$$\frac{\mathbb{M}' \xrightarrow{\Lambda_i} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}'_i \quad \mathsf{G}'_i \vdash^{\mathscr{P}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}'_i \quad \forall i \in I \neq \emptyset \qquad \{\Lambda_i\}_{i\in I} \text{ is } \mathscr{P}\text{-coherent for } \mathbb{M}' \quad \mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}'_i) = \mathsf{prt}(\mathbb{M}')}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}'_i \vdash^{\mathscr{P}} \mathbb{M}'}$$

□

Theorem 5.8 (Session Fidelity) *If* $\mathsf{G} \vdash \mathbb{M}$ *and* $\mathsf{G} \xrightarrow{\Lambda} \mathsf{G}'$, *then* $\mathbb{M} \xrightarrow{\Lambda} \mathbb{M}'$ *and* $\mathsf{G}' \vdash \mathbb{M}'$ *for some* $\mathbb{M}'$.

Proof. By coinduction on the derivation of $\mathsf{G} \xrightarrow{\Lambda} \mathsf{G}'$. We distinguish two cases according to the axiom/rule justifying $\mathsf{G} \xrightarrow{\Lambda} \mathsf{G}'$.

Axiom [E-Comm]: then $\mathsf{G} = \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$, $\Lambda = \Lambda_j$ and $\mathsf{G}' = \mathsf{G}_j$ for some $j \in I$. Since $\mathsf{G} \neq \mathtt{End}$, the last rule in the derivation of $\mathsf{G} \vdash \mathbb{M}$ must be [TComm], which implies that $\Lambda = \mathsf{p}\lambda\mathsf{q}$ for some p, λ and q such that

$$\mathbb{M} \equiv \mathsf{p}[\mathsf{q}!\lambda.P \dotplus P'] \parallel \mathsf{q}[\mathsf{p}?\lambda.Q \dotplus Q'] \parallel \mathbb{M}_0 \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}_0 \equiv \mathbb{M}'$$

for some $\mathbb{M}_0$, and $\mathsf{G}' \vdash^{\mathscr{P}} \mathbb{M}'$.

Rule [I-Comm]: then $\mathsf{G} = \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$ and $\mathsf{G}' = \Sigma_{i\in I}\Lambda_i.\mathsf{G}'_i$ with $\mathsf{G}_i \xrightarrow{\Lambda} \mathsf{G}'_i$ and $\Lambda \in cap(\mathsf{G}_i)$ and $\mathsf{prt}(\Lambda) \cap \mathsf{prt}(\Lambda_i) = \emptyset$ for all $i \in I$.
Since the last rule in the derivation of $\mathsf{G} \vdash \mathbb{M}$ must be [TComm], it follows that

- $\{\Lambda_i\}_{i\in I}$ is $\mathscr{P}$-coherent for $\mathbb{M}$;
- $\mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i$ and $\mathsf{G}_i \vdash^{\mathscr{P}} \mathbb{M}_i$, for all $i \in I \neq \emptyset$;
- $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \mathsf{prt}(\mathbb{M})$.

By the coinduction hypothesis, we know that, for each $i \in I$, there exists $\mathbb{M}'_i$ such that

$$\mathbb{M}_i \xrightarrow{\Lambda} \mathbb{M}'_i \qquad \text{and} \qquad \mathsf{G}'_i \vdash \mathbb{M}'_i$$

Notice that, being the label Λ the same for all these reductions, by Lemma 5.1 there exists a unique redex

$$\mathsf{p}[\mathsf{q}!\lambda.P \dotplus P'] \parallel \mathsf{q}[\mathsf{p}?\lambda.Q \dotplus Q']$$

with contractum $\mathsf{p}[P] \parallel \mathsf{q}[Q]$ in all the $\mathbb{M}_i$, such that $\Lambda = \mathsf{p}\lambda\mathsf{q}$. On the other hand, since we know that $\mathsf{prt}(\Lambda) \cap \mathsf{prt}(\Lambda_i) = \emptyset$ for all $i \in I$, it must be the case that $\Lambda_i = \mathsf{r}_i\lambda_i\mathsf{s}_i$ and

$$\mathbb{M}'_i \equiv \mathsf{r}_i[R_i] \parallel \mathsf{s}_i[S_i] \parallel \mathbb{M}''_i$$

for some r_i, s_i, R_i, S_i, $\mathbb{M}''_i$ and for all $i \in I$. Hence, since $\mathbb{M}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'_i$, we have that, for each $i \in I$,

$$\mathbb{M} \equiv \mathsf{r}_i[\mathsf{s}_i!\lambda_i.R_i \dotplus R'_i] \parallel \mathsf{s}_i[\mathsf{r}?\lambda_i.S_i \dotplus S'_i] \parallel \mathbb{M}''_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r}_i[\mathsf{s}_i!\lambda_i.R_i \dotplus R'_i] \parallel \mathsf{s}_i[\mathsf{r}?\lambda_i.S_i \dotplus S'_i] \parallel \mathbb{M}'''_i \equiv \mathbb{M}'$$

for some R'_i, S'_i (if any), $\mathbb{M}'''_i$ and for all $i \in I$.

By Lemma 5.3, $\mathsf{G}'_i \vdash \mathsf{r}_i[R_i] \parallel \mathsf{s}_i[S_i] \parallel \mathbb{M}''_i$ implies $\mathsf{prt}(\mathsf{G}'_i) = \mathsf{prt}(\mathsf{r}_i[R_i] \parallel \mathsf{s}_i[S_i] \parallel \mathbb{M}''_i)$ and then $\mathsf{prt}(\mathsf{G}') = \mathsf{prt}(\mathbb{M}')$. Moreover, from $\mathsf{prt}(\Lambda) \cap \mathsf{prt}(\Lambda_i) = \emptyset$ for all $i \in I$, we immediately get that $\Lambda \notin \{\Lambda_i\}_{i\in I}$. So, by Lemma 5.6 we get that $\{\Lambda_i\}_{i\in I}$ is $\mathscr{P}$-coherent for $\mathbb{M}'$. We conclude that there exists a derivation ending by the following application of Rule [TComm]

$$[\text{TComm}]\ \frac{\mathbb{M}' \xrightarrow{\Lambda_i} \mathbb{M}'_i \quad \mathsf{G}'_i \vdash \mathbb{M}'_i \quad \forall i \in I \neq \emptyset \qquad \{\Lambda_i\}_{i\in I} \text{ is } \mathscr{P}\text{-coherent for } \mathbb{M}' \quad \mathsf{prt}(\mathsf{G}') = \mathsf{prt}(\mathbb{M}')}{\mathsf{G}' \vdash^{\mathscr{P}} \mathbb{M}'}$$

□

Toward establishing the property that typable sessions are lock free, we first prove the following lemma. In words, if $\mathsf{p} \in \mathsf{prt}(\mathsf{G})$, then it must occur somewhere in its syntactic tree, hence there is a trace $\sigma \cdot \Lambda$ out of G, consisting just of external communications, which corresponds to a path in the tree ending by the first communication label Λ involving p.

Lemma 5.9 *If $\mathsf{p} \in \mathsf{prt}(\mathsf{G})$, then there are σ, Λ and G' such that $\mathsf{G} \xrightarrow{\sigma\cdot\Lambda} \mathsf{G}'$, $\mathsf{p} \notin \mathsf{prt}(\sigma)$ and $\mathsf{p} \in \mathsf{prt}(\Lambda)$.*

Proof. The proof is by coinduction on G. Since $\mathsf{p} \in \mathsf{prt}(\mathsf{G})$ we have that $\mathsf{G} = \Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$. Now, let us assume $\mathsf{p} \in \bigcup_{i\in I} \mathsf{prt}(\Lambda_i)$. Without loss of generality, we can also assume that $\mathsf{p} \in \mathsf{prt}(\Lambda_j)$ for some $j \in I$. Then we immediately have that $\mathsf{G} \xrightarrow{\Lambda_j} \mathsf{G}_j$ by Axiom [E-Comm], and the thesis trivially follows by taking $\sigma = \varepsilon$. Otherwise, since $\mathsf{p} \in \mathsf{prt}(\mathsf{G}) = \bigcup_{i\in I} \mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I} \mathsf{prt}(\mathsf{G}_i)$, we have that $\mathsf{p} \notin \bigcup_{i\in I} \mathsf{prt}(\Lambda_i)$ implies $\mathsf{p} \in \mathsf{prt}(\mathsf{G}_j)$ for some $j \in I$. By the coinduction hypothesis, we have that there are a σ' and a Λ such that $\mathsf{G}_j \xrightarrow{\sigma'\cdot\Lambda} \mathsf{G}'$, $\mathsf{p} \notin \mathsf{prt}(\sigma')$ and $\mathsf{p} \in \mathsf{prt}(\Lambda)$. Then the thesis follows by setting $\sigma = \Lambda_j \cdot \sigma'$, since $\mathsf{G} \xrightarrow{\Lambda_j} \mathsf{G}_j$ by Axiom [E-Comm] and $\mathsf{G}_j \xrightarrow{\sigma'\cdot\Lambda} \mathsf{G}'$. □

Observe that the last lemma is a sort of inverse implication w.r.t. Lemma 5.4, since it shows that the existence of a capability which is an actual communication of a global type G follows by the fact that one of the involved participants is in $\mathsf{prt}(\mathsf{G})$.

We are now in place to prove that typable sessions are lock free.

Theorem 5.10 (Lock Freedom) *If $\mathbb{M}$ is typable, then $\mathbb{M}$ is lock free.*

Proof. Let $\mathsf{G} \vdash^{\mathscr{P}} \mathbb{M}$. Following Definition 2.5, in order to prove Lock Freedom for $\mathbb{M}$, let $\mathbb{M} \xrightarrow{\sigma} \mathbb{M}'$ for a finite σ and let $\mathsf{p} \in \mathsf{prt}(\mathbb{M}')$. By Subject Reduction (Theorem 5.7) we get $\mathsf{G}' \vdash^{\mathscr{P}} \mathbb{M}'$. We can now recur to Lemma 5.3 and get $\mathsf{p} \in \mathsf{prt}(\mathsf{G}')$. From the fact that $\mathsf{p} \in \mathsf{prt}(\mathsf{G}')$ and by Lemma 5.9 it follows that $\mathsf{G}' \xrightarrow{\sigma'\cdot\Lambda} \mathsf{G}''$ for some σ' and Λ with $\mathsf{p} \notin \mathsf{prt}(\sigma')$ and $\mathsf{p} \in \mathsf{prt}(\Lambda)$. Now the thesis follows by Session Fidelity (Theorem 5.8). □

We conclude this section by showing that typability of a session does depend on the choice neither of the $\mathscr{P}$-coherent sets nor of the partition $\mathscr{P}$.

Theorem 5.11 *If $\mathbb{M}$ is typable in $\vdash^{\mathscr{P}}$ and $\{\Lambda_i\}_{i\in I}$ is $\mathscr{P}$-coherent for $\mathbb{M}$, then $\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \vdash^{\mathscr{P}} \mathbb{M}$ for some G_i and all $i \in I$.*

Proof. The $\mathscr{P}$-coherence of $\{\Lambda_i\}_{i\in I}$ for $\mathbb{M}$ gives $\mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i$, which implies by Subject Reduction (Theorem 5.7) $\mathsf{G}_i \vdash^{\mathscr{P}} \mathbb{M}_i$ for some G_i and all $i \in I$. By Lemma 5.3 $\mathsf{prt}(\mathsf{G}_i) = \mathsf{prt}(\mathbb{M}_i)$ for all $i \in I$. From $\mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i$ for all $i \in I$ we get $\mathsf{prt}(\mathbb{M}) = \bigcup_{i\in I} \mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I} \mathsf{prt}(\mathbb{M}_i)$ by Lemma 5.2. By definition, $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \bigcup_{i\in I} \mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I} \mathsf{prt}(\mathsf{G}_i)$. We conclude $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i) = \mathsf{prt}(\mathbb{M})$, so we can derive $\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \vdash^{\mathscr{P}} \mathbb{M}$ using Rule [TComm]. □

Theorem 5.12 *If $\mathbb{M}$ is typable in $\vdash^{\mathscr{P}}$ and it is $\mathscr{P}'$-modularisable, then $\mathbb{M}$ is typable in $\vdash^{\mathscr{P}'}$ too.*

Proof. Since $\mathbb{M}$ is typable in $\vdash^{\mathscr{P}}$, then $\mathbb{M}$ is $\mathscr{P}$-modularisable by Definition 4.2. Let $\mathscr{P} = \{\mathbf{P}_k\}_{k\in K}$ and $\{\Lambda^k_h\}_{h\in H_k} = \{\Lambda \in \mathscr{L}(\mathbb{M}) \mid \mathsf{prt}(\Lambda) \cap \mathbf{P}_k \neq \emptyset\}$. By definition of partition we observe that $\bigcup_{k\in K}\bigcup_{h\in H_k} \Lambda^k_h = \mathscr{L}(\mathbb{M})$. By Definitions 4.2 and 3.4 for all $k \in K$ $\{\Lambda^k_h\}_{h\in H_k}$ is $\mathscr{P}$-coherent for $\mathbb{M}$. By Lemma 5.11 for each $k \in K$ there is a global type $\mathsf{G}_k = \Sigma_{h\in H_k}\Lambda^k_h.\widehat{\mathsf{G}}_h$ which can be assigned to $\mathbb{M}$ in $\vdash^{\mathscr{P}}$, that is $\Sigma_{h\in H_k}\Lambda^k_h.\widehat{\mathsf{G}}_h \vdash^{\mathscr{P}} \mathbb{M}$.
Let $\{\Lambda_i\}_{i\in I}$ be $\mathscr{P}'$-coherent for $\mathbb{M}$. We proceed now by coinduction simultaneously on the derivations $\Sigma_{h\in H_k}\Lambda^k_h.\widehat{\mathsf{G}}_h \vdash^{\mathscr{P}} \mathbb{M}$ for all $k \in K$. From the above, for each $i \in I$ there is $k_i \in K$ and $l_i \in H_{k_i}$ such that one

of the premises of the conclusion $\Sigma_{h\in H_{k_i}}\Lambda_h^{k_i}.\widehat{\mathsf{G}}_h \vdash^{\mathscr{P}} \mathbb{M}$ is $\widehat{\mathsf{G}}_{l_i} \vdash^{\mathscr{P}} \mathbb{M}_i$ where $\mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i$. By coinduction we get $\mathsf{G}_i'' \vdash^{\mathscr{P}'} \mathbb{M}_i$ for some G_i'' and for all $i \in I$. By Lemma 5.3 $\mathsf{prt}(\mathsf{G}_i'') = \mathsf{prt}(\mathbb{M}_i)$ for all $i \in I$. We have $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i'') = \bigcup_{i\in I}\mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I}\mathsf{prt}(\mathsf{G}_i'')$ and $\mathsf{prt}(\mathbb{M}) = \bigcup_{i\in I}\mathsf{prt}(\Lambda_i) \cup \bigcup_{i\in I}\mathsf{prt}(\mathbb{M}_i)$ by Lemma 5.2, so we get $\mathsf{prt}(\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i'') = \mathsf{prt}(\mathbb{M})$. Then we can use Rule [TCOMM] to derive $\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i'' \vdash^{\mathscr{P}'} \mathbb{M}$. □

6 Concluding Remarks, Related and Future Works

In the setting of the *message-passing* communication model, it is possible to envisage mechanisms of interaction where a process can be, at the very same time, both a potential sender and a potential receiver. In various frameworks for concurrent systems, like session types and communicating finite state machines, as well as the π-calculus, such mechanism is referred to as *mixed choice*. The flexibility and expressive power of mixed choice is understandably counterbalanced by a difficult control of the behaviour of systems. That was arguably the motivation that mostly restrained the session type community from pursuing a thorough investigation of this sort of interactions. A stimulus in that direction has been instead recently given by some papers like [10, 28, 29, 30, 31]. In particular, [10, 30] investigate mixed choice for binary session types, whereas [31] considers mixed choice in a MPST setting [18, 19] following the approach of [32] (global types are in fact not taken into account in [31]). Even if the main concern of [31] is the expressivity of multiparty calculi (according to the full range of possible restrictions of mixed choice), type systems assigning local types to processes are provided, where various predicates on contexts of local types are investigated. In [28, 29] binary sessions with timeout and mixed choice are enriched with a semantics guaranteeing Progress and a type system enjoying Subject Reduction.

Inspired by [31], we carry on an investigation on the use of mixed choice for synchronous communications in the setting of SMPS [12, 4]. In SMPS, global types are inferred for sessions, i.e. parallel compositions of named processes, the latter being an abstraction for both processes and local types usually considered in MPST. Our processes can use now mixed choice. Subject Reduction, Session Fidelity, as well as Lock Freedom are ensured for typable sessions. The most relevant aspect of our type system is that we look at sessions as implicitly composed by modules whose participants freely interact via unrestricted mixed choice, whereas the inter-module communications can be more easily controlled by allowing communications with only one participant. Such an approach does not discard a priori any session. In fact all sessions – even those developed without any specific modularisation in mind – can be modularised in a less or more refined way: from a single large module comprising the whole session, to a set of modules made by single participants. In the former case the typing is less effective, since the global type would result in a complete interaction tree, whereas in the latter case the typing coincides with the standard SMPS (with no mixed choice). In particular, our type system is conservative since, for processes without mixed choice, it coincides with the type system of [6], which is at present one of the most expressive. For mixed choice there is only the type system of [31], which is modularised by predicates on local types. An interesting property ensured by that type system is *safety*. Safety in [31] entails that the protocol for process interactions is such that, when a participant intends to perform an interaction, it nondeterministically chooses among all the participants that can interact with it. Then there is a nondeterministic decision concerning who has to play the role of the sender and who of the receiver. At that point, the sender performs an internal choice among the available outputs. Typing then guarantees that the possibility of interaction does not depend on the chosen output. We consider, instead, a simplified synchronisation protocol where there exists a nondeterministic choice among all the participants that can actually interact and all the possible communication interactions. As mentioned in Section 2, this approach makes "!" and "?" just two complementary synchronisation actions. It is however possible to

modify our type system in order to guarantee safety of sessions as defined in [31] by requiring that

$$\mathsf{p}[\mathsf{q}!\lambda.P \dddot{+} P'] \in \mathbb{M} \text{ and } \mathsf{q}[\mathsf{p}?\lambda'.Q \dddot{+} Q'] \in \mathbb{M} \text{ imply } \mathsf{p}\lambda\mathsf{q} \in \mathscr{L}(\mathbb{M})$$

The safety condition only ensures that an output finds the corresponding input if the receiver offers some inputs for the sender. An alternative condition is

$$\mathsf{p}[\mathsf{q}!\lambda.P \dddot{+} P'] \in \mathbb{M} \text{ implies } \mathbb{M} \xrightarrow{\sigma\cdot\mathsf{p}\lambda\mathsf{q}} \text{ for some } \sigma$$

With this last condition we would type less sessions, for example we would not type

$$\mathsf{p}[\mathsf{q}!\lambda + \mathsf{r}!\lambda'] \parallel \mathsf{r}[\mathsf{p}?\lambda']$$

and the election example. The feature of this alternative condition is that the choice between outputs is internal, in agreement with an asynchronous implementation. In such a case it is worth remarking that a restriction of the subtyping relation used in [31] would be implicitly entailed by our typing.

As first pointed out in [14], the naive extension of the original type system [19] to sessions where input choices have different senders is unsound. In fact one can type sessions which reduce to untypable and stuck sessions. Suitable conditions ensuring a sound extension have been proposed both for the synchronous [14] and asynchronous [24, 11] communications. Notably the type system proposed here does not have this problem.

We notice that modular sessions can be obtained by connecting independent sessions via gateways, according to the PaI approach to system composition. When composing several typable SMPS systems (through *compatible* interfaces) one gets a typable system [7]. Although the presence of mixed choice does not seem to be a major obstacle, it is unlikely a result like the one of [7] could be easily translated in the present context. In fact, as shown in Example 3.5 we allow the presence of multiple connectors per module. This possibility is actually a severe impediment to the safeness of PaI composition. In fact, let us take the following two typable sessions:

$$\mathsf{p}[\mathsf{q}!\lambda] \parallel \mathsf{q}[\mathsf{p}?\lambda] \qquad\qquad \mathsf{r}[\mathsf{s}?\lambda] \parallel \mathsf{s}[\mathsf{r}!\lambda]$$

Taking all the participants as interfaces and considering that there exists a typable connection policy among them [7], the PaI composition would result in the following untypable $\{\{\mathsf{p},\mathsf{q}\},\{\mathsf{r},\mathsf{s}\}\}$-modular session

$$\mathsf{p}[\mathsf{r}?\lambda.\mathsf{q}!\lambda] \parallel \mathsf{q}[\mathsf{p}?\lambda.\mathsf{s}!\lambda] \parallel \mathsf{r}[\mathsf{s}?\lambda.\mathsf{p}!\lambda] \parallel \mathsf{s}[\mathsf{q}?\lambda.\mathsf{r}!\lambda]$$

where all the processes begin with an input, so forming a deadlock. This problem was overcome in [13], in a setting using projections and without mixed choice, by means of a suitable extension of the syntax of global types.

It is worth noticing that mixed choice make PaI composition problematic even by allowing one connector only. For example, by composing using p and t the following typable sessions

$$\mathsf{p}[\mathsf{q}!\lambda_1 + \mathsf{r}?\lambda_2] \parallel \mathsf{q}[\mathsf{p}?\lambda_1 + \mathsf{s}?\lambda_3] \parallel \mathsf{r}[\mathsf{p}!\lambda_2 + \mathsf{s}!\lambda_4] \parallel \mathsf{s}[\mathsf{q}!\lambda_3 + \mathsf{r}?\lambda_4]$$
$$\mathsf{t}[\mathsf{u}?\lambda_1 + \mathsf{v}!\lambda_2] \parallel \mathsf{u}[\mathsf{t}!\lambda_1 + \mathsf{w}!\lambda_3] \parallel \mathsf{v}[\mathsf{t}?\lambda_2 + \mathsf{w}?\lambda_4] \parallel \mathsf{w}[\mathsf{u}?\lambda_3 + \mathsf{v}!\lambda_4]$$

we get the session

$$\mathsf{p}[\mathsf{t}?\lambda_1.\mathsf{q}!\lambda_1 + \mathsf{r}?\lambda_2.\mathsf{t}!\lambda_2] \parallel \mathsf{q}[\mathsf{p}?\lambda_1 + \mathsf{s}?\lambda_3] \parallel \mathsf{r}[\mathsf{p}!\lambda_2 + \mathsf{s}!\lambda_4] \parallel \mathsf{s}[\mathsf{q}!\lambda_3 + \mathsf{r}?\lambda_4] \parallel$$
$$\mathsf{t}[\mathsf{u}?\lambda_1.\mathsf{p}!\lambda_1 + \mathsf{p}?\lambda_2.\mathsf{v}!\lambda_2] \parallel \mathsf{u}[\mathsf{t}!\lambda_1 + \mathsf{w}!\lambda_3] \parallel \mathsf{v}[\mathsf{t}?\lambda_2 + \mathsf{w}?\lambda_4] \parallel \mathsf{w}[\mathsf{u}?\lambda_3 + \mathsf{v}!\lambda_4]$$

which reduces to the stuck session

$$\mathsf{p}[\mathsf{t}!\lambda_2] \parallel \mathsf{t}[\mathsf{p}!\lambda_1]$$

PaI composition provides also an intuitive justification for the shape of connectors in our modularisation. In fact, by composing systems via interfaces with unrestricted mixed choice, most of the communication properties, if any, are not preserved, as shown by the previous example.

As future work we plan to investigate PaI composition for sessions with mixed choice and asyn-

chronous communication, taking inspiration from [28, 29], where asynchronous communication for sessions with mixed choice is first modelled. We deem worth investigating the modular approach to session types also for standard MPST, as well as for the recent approaches to global types and projections devised in [24, 23].

References

[1] Franco Barbanera, Viviana Bono & Mariangiola Dezani-Ciancaglini (2025): *Open compliance in multiparty sessions with partial typing*. Journal of Logical and Algebraic Methods in Programming 144, p. 101046, doi:10.1016/j.jlamp.2025.101046.

[2] Franco Barbanera, Viviana Bono & Mariangiola Dezani-Ciancaglini (2025): *Partially typed multiparty sessions with internal delegation*. Journal of Logical and Algebraic Methods in Programming 142, p. 101018, doi:10.1016/J.JLAMP.2024.101018.

[3] Franco Barbanera & Mariangiola Dezani-Ciancaglini (2023): *Partially typed multiparty sessions*. In Clément Aubert, Cinzia Di Giusto, Simon Fowler & Larisa Safina, editors: *ICE*, *EPTCS* 383, Open Publishing Association, pp. 15–34, doi:10.4204/EPTCS.383.2.

[4] Franco Barbanera, Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2022): *Open compliance in multiparty sessions*. In S. Lizeth Tapia Tarifa & José Proença, editors: *FACS*, *LNCS* 13712, Springer, pp. 222–243, doi:10.1007/978-3-031-20872-0_13.

[5] Franco Barbanera, Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2024): *Partial typing for asynchronous multiparty sessions*. In Sandra Alves & Ian Mackie, editors: *DCM*, *EPTCS* 408, Open Publishing Association, pp. 1–20, doi:10.4204/EPTCS.408.1.

[6] Franco Barbanera, Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (2024): *Un-projectable global types for multiparty sessions*. In Alessandro Bruni, Alberto Momigliano, Matteo Pradella & Matteo Rossi, editors: *PPDP*, ACM Press, pp. 15:1–15:13, doi:10.1145/3678232.3678245.

[7] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Lorenzo Gheri & Nobuko Yoshida (2023): *Multicompatibility for multiparty-session composition*. In Santiago Escobar & Vasco T. Vasconcelos, editors: *PPDP*, ACM Press, pp. 2:1–2:15, doi:10.1145/3610612.3610614.

[8] Luc Bougé (1988): *On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes*. Acta Informaticae 25(2), p. 179–201, doi:10.1007/BF00263584.

[9] Daniel Brand & Pitro Zafiropulo (1983): *On communicating finite-state machines*. Journal of ACM 30(2), pp. 323–342, doi:10.1145/322374.322380.

[10] Filipe Casal, Andreia Mordido & Vasco T. Vasconcelos (2022): *Mixed sessions*. Theoretical Computer Science 897, pp. 23–48, doi:10.1016/J.TCS.2021.08.005.

[11] Ilaria Castellani, Mariangiola Dezani-Ciancaglini & Paola Giannini (2022): *Asynchronous Sessions with Input Races*. In Marco Carbone & Rumyana Neykova, editors: *PLACES*, *EPTCS* 356, Open Publishing Association, pp. 12–23, doi:10.4204/EPTCS.356.2.

[12] Francesco Dagnino, Paola Giannini & Mariangiola Dezani-Ciancaglini (2023): *Deconfined global types for asynchronous sessions*. Logical Methods in Computer Science 19(1), pp. 1–41, doi:10.46298/lmcs-19(1:3)2023.

[13] Lorenzo Gheri & Nobuko Yoshida (2023): *Hybrid multiparty session types: compositionality for protocol specification through endpoint projection*. PACMPL 7 (OOPSLA1), pp. 112–142, doi:10.1145/3586031.

[14] Rob van Glabbeek, Peter Höfner & Ross Horne (2021): *Assuming just enough fairness to make session types complete for lock-freedom*. In Leonid Libkin, editor: *LICS*, IEEE, pp. 1–13, doi:10.1109/LICS52264.2021.9470531.

[15] Mohamed G. Gouda, Eric G. Manning & Yao-Tin Yu (1984): *On the progress of communication between two machines*. Information and Control 63(3), pp. 200–2016, doi:10.1016/S0019-9958(84)80014-5.

[16] Kohei Honda (1993): *Types for dyadic Interaction*. In Eike Best, editor: *CONCUR*, *LNCS* 715, Springer, pp. 509–523, doi:10.1007/3-540-57208-2_35.

[17] Kohei Honda, Vasco T. Vasconcelos & Makoto Kubo (1998): *Language primitives and type discipline for structured communication-based programming*. In Chris Hankin, editor: *ESOP*, *LNCS* 1381, Springer, pp. 122–138, doi:10.1007/BFb0053567.

[18] Kohei Honda, Nobuko Yoshida & Marco Carbone (2008): *Multiparty asynchronous session types*. In George C. Necula & Philip Wadler, editors: *POPL*, ACM Press, pp. 273–284, doi:10.1145/1328897.1328472.

[19] Kohei Honda, Nobuko Yoshida & Marco Carbone (2016): *Multiparty asynchronous session types*. *Journal of the ACM* 63(1), pp. 9:1–9:67, doi:10.1145/2827695.

[20] Naoki Kobayashi & Davide Sangiorgi (2010): *A hybrid type system for lock-freedom of mobile processes*. *ACM Transactions on Programming Languages and Systems* 32(5), pp. 16:1–16:49, doi:10.1016/j.jlamp.2024.101018.

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

[22] Julien Lange, Nicholas Ng, Bernardo Toninho & Nobuko Yoshida (2017): *Fencing off go: liveness and safety for channel-based programming*. In Giuseppe Castagna & Andrew D. Gordon, editors: *POPL*, ACM Press, pp. 748–761, doi:10.1145/3009837.3009847.

[23] Elaine Li, Felix Stutz, Thomas Wies & Damien Zufferey (2023): *Complete multiparty session type projection with automata*. In Constantin Enea & Akash Lal, editors: *CAV*, *LNCS* 13966, Springer, pp. 350–373, doi:10.1007/978-3-031-37709-9_17.

[24] Rupak Majumdar, Madhavan Mukund, Felix Stutz & Damien Zufferey (2021): *Generalising projection in asynchronous multiparty session types*. In Serge Haddad & Daniele Varacca, editors: *CONCUR*, *LIPIcs* 203, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 35:1–35:24, doi:10.4230/LIPIcs.CONCUR.2021.35.

[25] `https://www.ottia.com/en/post/project-modularization`.

[26] Luca Padovani (2014): *Deadlock and lock freedom in the linear π-calculus*. In Thomas A. Henzinger & Dale Miller, editors: *CSL-LICS*, ACM Press, pp. 72:1–72:10, doi:10.1145/2603088.2603116.

[27] Catuscia Palamidessi (2003): *Comparing the expressive power of the synchronous and asynchronous pi-calculi*. *Mathematical Structures in Computer Science* 13(5), pp. 685–719, doi:10.1017/S0960129503004043.

[28] Jonah Pears, Laura Bocchi & Andy King (2023): *Safe Asynchronous Mixed-Choice for Timed Interactions*. In Sung-Shik Jongmans & Antónia Lopes, editors: *COORDINATION*, *LNCS* 13908, Springer, pp. 214–231, doi:10.1007/978-3-031-35361-1_12.

[29] Jonah Pears, Laura Bocchi, Maurizio Murgia & Andy King (2024): *Introducing TOAST: Safe Asynchronous Mixed-Choice For Timed Interactions*. *CoRR* abs/2401.11197, doi:10.48550/ARXIV.2401.11197.

[30] Kirstin Peters & Nobuko Yoshida (2024): *Mixed choice in session types*. *Information and Computation* 298, p. 105164, doi:10.1016/J.IC.2024.105164.

[31] Kirstin Peters & Nobuko Yoshida (2024): *Separation and encodability in mixed choice multiparty sessions*. In Pawel Sobocinski, Ugo Dal Lago & Javier Esparza, editors: *LICS*, ACM Press, pp. 62:1–62:15, doi:10.1145/3661814.3662085.

[32] Alceste Scalas & Nobuko Yoshida (2019): *Less is more: multiparty session types revisited*. *PACMPL* 3 (POPL), pp. 30:1–30:29, doi:10.1145/3290343.

[33] Herbert A. Simon (1991): *The architecture of complexity*. In: *Facets of Systems Science*, Springer, pp. 457–476, doi:10.1007/978-1-4899-0718-9_31.