Un-projectable Global Types for Multiparty Sessions

Franco Barbanera
University of Catania, Italy
franco.barbanera@unict.it

Mariangiola Dezani-Ciancaglini
University of Torino, Italy
dezani@di.unito.it

Ugo de'Liguoro
University of Torino, Italy
ugo.deliguoro@unito.it

ABSTRACT

A well-formed global type describes the interaction protocol of multiple end-points via the projection to local specifications. Typed sessions of processes enjoy good communication properties and their overall behaviour is the one described by the global type. We show that a projectable global type is bounded (also said "balanced" in the literature) but also that projectability is not necessary for a global type to be a sound description of well-behaved systems. By revising the semantics of global types via a coinductively defined LTS, we obtain a conservative extension of previous type systems in case of simple sessions without channels and local types, which we call Simple MultiParty Sessions, accommodating unbounded and hence un-projectable global types. Such a system is sound and encompasses infinite sessions that do not type-check for any bounded and/or projectable global type.

CCS CONCEPTS

• **Theory of computation → Process calculi**; **Type theory**.

KEYWORDS

Communication-centred programming, Process Calculi, Simple Multiparty Sessions, Global Types

ACM Reference Format:
Franco Barbanera, Mariangiola Dezani-Ciancaglini, and Ugo de'Liguoro. 2024. Un-projectable Global Types for Multiparty Sessions. In *26th International Symposium on Principles and Practice of Declarative Programming (PPDP 2024), September 09–11, 2024, Milano, Italy*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3678232.3678245

1 INTRODUCTION

MultiParty Session Types (MPST), introduced in [26, 27], is a body of choreographic formalisms where two distinct but related views of concurrent systems coexist: (*a*) the *global view*, a formal specification via *global types* of the overall behaviour of a system; (*b*) the *local view*, namely the specification of the behaviours of the single components, generalising binary local types [24, 25]. The relation among these two layers is established via the notion of *projection* which, given a global type, may produce a tuple of local types, one for each component. Indeed the projection is a partial mapping that is defined only if certain well-formedness conditions are met, including the mergeability of projections to local end-points. Such conditions are sufficient to guarantee that a network made of any tuple of processes (the third layer of the picture) communicating via channels that are typable by the so obtained local types will adhere to the protocol formalised by the projected global type and will interact safely.

As observed in [33], the original approach, called the "classic" MPST, suffers from some theoretical issues and limitations ruling out very natural and harmless protocols. More precisely, some projectable global types do not ensure a consistency condition (which generalises duality from binary local types) of the so-obtained local types. In this case typed processes can reduce to untyped ones. On the other hand, perfectly sound systems of local types (consisting of typing contexts assigning local types to variables and channels) are not the projection of any global type. In the same paper, it is shown that attempts to overcome the difficulties by defining more sophisticated merge functions and subtyping relations have led to unsound systems. The way out proposed in [33] is to abandon the idea of global types and of binary local type duality introducing a two-layer system consisting of local types and processes that are typable by them, where good behavioural properties are guaranteed by corresponding properties at type-level.

The standpoint of the present contribution is that the elimination of global types from the MPST is a loss in practice that might be unnecessary in theory. Indeed, without a global view of interaction, programmers will hardly understand whether processes conforming to local protocols realise the system they have in mind as a whole. From the theoretical point of view, the actual weakness of the classic MPST depends on the notion of projectability. To study the problem we then step back to simpler systems where global types are directly assigned to sessions, namely sets of named concurrent processes.

In [4] a two-layered MPST formalism with synchronous communications has been introduced, which we dub *Simple MultiParty Sessions* (SMPS). SMPS are based on the system for an asynchronous calculus in [13], and they have been further studied in the subsequent [2, 5, 6]. In particular, only the layers of global types and processes are considered, disregarding local types and projections. Remarkably, conditions ensuring the relevant properties of safety and liveness at the process level are embodied in the typing rules. These conditions are weaker than the original projectability conditions. We note that, albeit SMPS have no local types, processes are abstract and essentially coincide with local types of a single session, as e.g. in [22], where messages may have just ground types, that in the present study are immaterial, and hence omitted.

In the SMPS approach, systems of communicating processes are represented as *multiparty sessions*, i.e. parallel compositions of named processes (the *participants*). Then the type system derives global types for such sessions. To prove the soundness of the type system both sessions and types are given semantics using two LTS with the set of labels in common, each label representing an atomic synchronous communication among two of the participants of the

PPDP 2024, September 09–11, 2024, Milano, Italy

ACM ISBN 979-8-4007-0969-2/24/09.
https://doi.org/10.1145/3678232.3678245

session. The key properties are *Subject Reduction* and *Session Fidelity*. Subject Reduction states that, if a typable session evolves along a path of the session LTS, the same does its type in the respective LTS leading to a new type that can be derived for the so-obtained session. Session Fidelity states that, if a global type types a session and it evolves along a path of its LTS, then the same does the session in the respective LTS leading to a new session, which can be typed by the so obtained global type. Moreover, the system ensures the *Padovani Lock-freedom* property, in the words of [23]. Namely, any participant of a session reachable from a typable one which is the location of a non-terminated process will be involved in a transition in at least an execution path out of the reached session [31].

However, such good properties of the type system come at the price of a severe restriction of the global types that can occur in the typing judgements, dubbed *boundedness* (corresponding to *balancing* [21, Definition 3.3]). Roughly, a global type is bounded whenever in any point of its (possibly infinite) syntax tree, if a participant occurs in a branch out of this point, then it must occur in all the other branches. Since operational semantics, for both types and sessions, amounts to a set of *traces* in the corresponding LTS, the restriction to bounded types amounts to selecting a set of paths in the potential communications of a session and ruling out others. This corresponds to a *fairness notion* that is much stronger than Lock-freedom [23]. Worse than this, if the boundedness condition is removed, while keeping the standard operational semantics as e.g. in [4], Subject Reduction as formulated above fails.

In the present work we do not introduce new typing rules w.r.t. the system in [4], but we change the global type semantics. A crucial feature of the LTS of types is accounting for *internal communications*. Global types are hierarchical, as they are shaped as (possibly infinite) trees and such that, but for the type of the terminated sessions, there is a pair of participants associated with the root of the type which is allowed to communicate before any other in the type. Even if this does not necessarily implies that there is some causal dependency among the respective transitions. On the contrary, the structure of a session is flat, and several transitions may happen starting with the same session, involving disjoint pairs of participants in arbitrary order. To cope with this mismatch, in the standard LTS for global types [27], adopted in [4], also subexpressions of a type can be reduced, provided that such a reduction does not conflict with the transitions represented in the root of the type, and that this is the case for any immediate subexpression of the type.

The change we propose for the global type semantics consists in a coinductive formulation of the rule of internal communication, so allowing for a proper treatment of infinite global types, and of the unbounded ones in particular. More importantly, it suffices to establish all the relevant properties for the unrestricted system, including Subject Reduction, Session Fidelity and Lock-freedom as defined in [31]. Then we move on by adapting to global types the definition of projection, which in the present setting is a partial map that yields processes. We compare projectability to boundedness, showing that the former is equivalent to the latter when paired with *inhabitation* (i.e. when a global type is derivable for at least one session). As inhabited unbounded types do not break Subject Reduction, Session Fidelity, and Lock-freedom when the semantics of global types is coinductive, it follows that the new system is sound and it is a proper and conservative extension of the original one. Moreover, even protocols like the recursive two-buyer protocol from [33] are representable by a (simple) session that inhabits a global type in our system. As argued in [33], the local types formalising the latter protocol are not the projection of any global type, and indeed the corresponding session inhabits only an unbounded global type in our system.

Structure of the paper. In Section 2 we recall the calculus of multiparty sessions. Section 3 is devoted to the definitions of global types and of the type assignment system enabling to type sessions with global types. The key definition of the coinductive LTS on global types is in Section 4. In Section 5 we prove Subject Reduction, Session Fidelity, and Lock-freedom for typable sessions. In Section 6 we formally define projectability and boundedness and investigate the relationship between them. Simple examples are used throughout the paper to illustrate the main notions and ideas. Moreover, in Section 7 we study two further examples that are adopted from [33] to illustrate the expressiveness of the system. Finally, in Section 8 we provide some final remarks and discuss our results with respect to related works.

2 MULTIPARTY SESSIONS

We recall the (synchronous) calculus of SMPS, as defined in [4]. Such a calculus is a further simplified version of the one used in [22], which was first introduced in [17], as a synchronous and more abstract version of the calculus in [26, 27] eliminating both session channels and local types for communications inside sessions.

The following base sets and notation are used: *labels*, ranged over by λ, λ', ...; *participant names*, ranged over by p, q, r, s, ...; *processes*, ranged over by P, Q, R, S, ...; *sessions*, ranged over by $\mathbb{M}$, $\mathbb{M}'$, $\mathbb{N}$, $\mathbb{N}'$, ...; *integers*, ranged over by i, j, l, h, k, u, v, ...; *(finite) integer sets*, ranged over by I, J, L, H, K, U, V, ...

Definition 2.1 (Processes). Processes *are defined by:*

$$P \quad ::=_\rho \quad \mathbf{0} \mid \mathrm{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathrm{p}?\{\lambda_i.P_i\}_{i\in I}$$

where $I \neq \emptyset$ is finite and $\lambda_h \neq \lambda_k$ for $h, k \in I$ and $h \neq k$. We restrict the set of processes to the regular ones, i.e. terms having finitely many distinct subterms.

The productions in the above definition are interpreted *coinductively*, as indicated by the symbol $::=_\rho$. This means that the set of processes is the greatest fixed point of the (monotonic) functor over sets defined by the grammar, restricted however to the regular processes. So, processes are possibly infinite. The regularity condition entails that we only consider processes which are solutions of finite sets of equations, see [12]. The present formulation, however, allows to avoid explicitly handling variables, thus simplifying a lot the technical development. The same choice was first made in [9]. This enables us to adopt in proofs the coinduction style advocated in [28] which promotes readability and conciseness, without any loss of formal rigour.

Processes implement the behaviour of participants. The output process $\mathrm{p}!\{\lambda_i.P_i\}_{i\in I}$ non-deterministically chooses one message λ_k for some $k \in I$, and sends it to the participant p, thereafter continuing as P_k. Symmetrically, the input process $\mathrm{p}?\{\lambda_i.P_i\}_{i\in I}$ waits for one of the messages λ_i from the participant p, then continues as P_k after receiving, say, λ_k. The symbol $\mathbf{0}$ is used to denote the

terminated process. We shall omit writing trailing **0**'s in processes and denote $\mathsf{p}!\{\lambda.P\}$ and $\mathsf{p}?\{\lambda.P\}$ by $\mathsf{p}!\lambda.P$ and $\mathsf{p}?\lambda.P$, respectively.

In actual communicating systems, messages would carry values, that we avoid for the sake of simplicity. Hence no selection operation over values is included in the syntax.

Definition 2.2 (Multiparty Sessions). Multiparty sessions *(*sessions *for short) are defined by:*

$$\mathbb{M} = \mathsf{p}_1[\,P_1\,] \parallel \cdots \parallel \mathsf{p}_n[\,P_n\,]$$

with $\mathsf{p}_h \neq \mathsf{p}_k$ *for any* $h \neq k$*. The set of* participants *of a session* $\mathbb{M}$*,* $\mathsf{prt}(\mathbb{M})$*, is defined as*

$$\mathsf{prt}(\mathsf{p}_1[\,P_1\,] \parallel \cdots \parallel \mathsf{p}_n[\,P_n\,]) = \{\mathsf{p}_i \mid P_i \neq \mathbf{0} \ \ \&\ \ 1 \leq i \leq n\}$$

Because of the condition $\mathsf{p}_h \neq \mathsf{p}_k$ for any $h \neq k$, a session is essentially a finite set of (not necessarily distinct) processes P_i located at distinct participants p_i. To make this formal, over sessions we define the *structural congruence relation* $\mathbb{M} \equiv \mathbb{M}'$, which is the least congruence according to which the parallel composition is commutative and associative, and such that, for all $\mathbb{M}$ and fresh p, we have $\mathsf{p}[\,\mathbf{0}\,] \parallel \mathbb{M} \equiv \mathbb{M}$. This implies that $\mathsf{p}[\,\mathbf{0}\,] \equiv \mathsf{p}[\,\mathbf{0}\,] \parallel \mathsf{q}[\,\mathbf{0}\,] \equiv \mathsf{q}[\,\mathbf{0}\,]$ for all distinct p, q. In a sense, any session of the shape $\mathsf{p}[\,\mathbf{0}\,]$ represents the empty session.

The (synchronous) operational semantics of sessions is formally defined by the following labelled transition system.

Definition 2.3 (Session LTS). *A* communication action *is a triple* $\mathsf{p}\lambda\mathsf{q}$*, where* $\mathsf{p} \neq \mathsf{q}$*. The* labelled transition system (LTS) for multiparty sessions*, with communication actions as labels, is the closure under structural congruence of the reduction specified by the following axiom:*

$$\frac{k \in I \subseteq J}{\mathsf{p}[\,P\,] \parallel \mathsf{q}[\,Q\,] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\lambda_k\mathsf{q}} \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,] \parallel \mathbb{M}}\ [\text{S-Comm}]$$

where $P = \mathsf{q}!\{\lambda_i.P_i\}_{i\in I}$ *and* $Q = \mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}$*.*

Axiom [S-Comm] above is non-deterministic in the choice of messages and makes the communication possible: participant p sends message λ_k to participant q. The sender can freely choose the message, since the condition $I \subseteq J$ (borrowed from [3, 7]) ensures that the receiver must offer all sender messages and possibly more. This makes it possible to distinguish between internal and external choices in the operational semantics. This condition will always be true in well-typed sessions.

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

$$\sigma ::=_\rho \epsilon \mid \Lambda \cdot \sigma$$

where Λ ranges over communication actions and 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 use $\mathbb{M} \xrightarrow{\Lambda^*} \mathbb{M}$ to denote an arbitrary number of Λ-labelled transitions $\mathbb{M} \xrightarrow{\Lambda} \cdots \xrightarrow{\Lambda} \mathbb{M}$ producing the same session.

We write $\mathbb{M} \xrightarrow{\sigma}$ and $\mathbb{M} \not\xrightarrow{\sigma}$ with the standard meaning. Moreover, we define $\mathsf{prt}(\mathsf{p}\lambda\mathsf{q}) = \{\mathsf{p}, \mathsf{q}\}$ and $\mathsf{prt}(\sigma)$ as its obvious extension to traces.

Example 2.4 (Lovers and Greetings). Alice (a) and Bob (b) are steadily exchanging the message `love` until, possibly, deciding to issue `bye`. In the meantime, Carl (c) is willing to greet Daisy (d) by sending her the message `hello`. This session is formalised by

$$\mathbb{M}_{\mathsf{LG}} = \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,] \parallel \mathsf{c}[\,\mathsf{d}!\mathtt{hello}\,] \parallel \mathsf{d}[\,\mathsf{c}?\mathtt{hello}\,]$$

where $P_{\mathsf{LG}} = \mathsf{b}!\{\mathtt{love}.P_{\mathsf{LG}},\ \mathtt{bye}\}$, and $Q_{\mathsf{LG}} = \mathsf{a}?\{\mathtt{love}.Q_{\mathsf{LG}},\ \mathtt{bye}\}$. Since $\mathbb{M}_{\mathsf{LG}} \xrightarrow{\mathsf{a}\,\mathtt{love}\,\mathsf{b}} \mathbb{M}_{\mathsf{LG}}$ there is a (potentially infinite) sequence of transitions in which neither Carl nor Daisy are involved. However, Carl and Daisy are never prevented from greeting each other:

$$\mathbb{M}_{\mathsf{LG}} \xrightarrow{(\mathsf{a}\,\mathtt{love}\,\mathsf{b})^*} \mathbb{M}_{\mathsf{LG}} \xrightarrow{\mathsf{c}\,\mathtt{hello}\,\mathsf{d}} \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$$

Besides, $\mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$ can always terminate by a transition labelled by `bye`. ◇

Example 2.5 (Buyer, Seller and Carrier). In this classic example, a buyer (b) communicates to a seller (s) a list of arbitrarily many goodies he is willing to purchase. This is achieved by repeatedly sending the message `item` until he, possibly, decides to issue the message `buy`. In case this happens, the seller instructs the carrier (c) for the shipment of the goods by sending him the message `ship`:

$$\mathbb{M}_{\mathsf{bsc}} = \mathsf{b}[\,P_{\mathsf{bsc}}\,] \parallel \mathsf{s}[\,Q_{\mathsf{bsc}}\,] \parallel \mathsf{c}[\,\mathsf{s}?\mathtt{ship}\,]$$

where $P_{\mathsf{bsc}} = \mathsf{s}!\{\mathtt{item}.P_{\mathsf{bsc}}, \mathtt{buy}\}$, $Q_{\mathsf{bsc}} = \mathsf{b}?\{\mathtt{item}.Q_{\mathsf{bsc}}, \mathtt{buy}.\mathsf{c}!\mathtt{ship}\}$. Similarly to the previous example, we have $\mathbb{M}_{\mathsf{bsc}} \xrightarrow{(\mathsf{b}\,\mathtt{item}\,\mathsf{s})^*} \mathbb{M}_{\mathsf{bsc}}$, but the carrier will receive the message `ship` only after the seller has received `buy` from the buyer. ◇

A subsession of the shape $\mathsf{p}[\,\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}\,]$, where $I \subseteq J$, is called a *redex* and any $\mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,]$ for $k \in I$ is a *contractum* of the redex. Since I may contain more than one index k, a redex can have several contracta. However, in a transition labelled by $\mathsf{p}\lambda\mathsf{q}$ both the redex and the contractum are uniquely determined.

Lemma 2.6. *If* $\mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'$*, then there exists a unique redex* $\mathsf{p}[\,\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}\,]$ *such that*

$$\mathbb{M} \equiv \mathsf{p}[\,\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}\,] \parallel \mathbb{N}$$

and $\mathbb{M}' \equiv \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,] \parallel \mathbb{N}$ *for some* $\mathbb{N}$*, where* $k \in I \subseteq J$ *and* $\lambda = \lambda_k$*.*

Proof. Immediate by the definition of session LTS. □

The property of Lock-freedom for sessions is defined as in [31]. 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 2.7 (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* $\mathsf{p} \in \mathsf{prt}(\Lambda)$*.*

3 TYPE SYSTEM

Global types are usually represented via μ-expressions, as in [26, 27] and in almost all papers on MPST. Here instead we define coinductive global types as possibly infinite regular terms. An extensive discussion of formalisms for the representation of infinite (regular and non-regular) types can be found in [20].

Definition 3.1 (Global types). Global types *are defined by:*

$$\mathsf{G} ::=_\rho \mathsf{End} \mid \mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}$$

where $I \neq \emptyset$ and $\lambda_i \neq \lambda_j$ for $i, j \in I$ and $i \neq j$. We restrict the set of global types to the regular ones.
Given a global type G*, the set of its* participants*,* $\mathsf{prt}(\mathsf{G})$*, is defined as the smallest set satisfying the following equations:*

$$\mathsf{prt}(\mathsf{End}) = \emptyset \qquad \mathsf{prt}(\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p},\mathsf{q}\} \cup \bigcup_{i\in I} \mathsf{prt}(\mathsf{G}_i)$$

Since the definition of G is coinductive, the definition of $\mathsf{prt}(\mathsf{G})$ is such. As the syntactic tree of G is regular, $\mathsf{prt}(\mathsf{G})$ is well defined and finite. In the following, trailing End's will be omitted, and $\mathsf{p} \to \mathsf{q} : \{\lambda.\mathsf{G}\}$ will be written as $\mathsf{p} \to \mathsf{q} : \lambda.\mathsf{G}$.

Now we recall the SMPS type assignment system, as defined in [4]. The (synchronous) calculus of SMPS, because of its simplicity, enables to devise a type system where global types are directly inferred for sessions. Double lines in the rules recall that they are interpreted coinductively [32, Chapter 21].

Definition 3.2 (Type System). *Judgements of the form* $\mathsf{G} \vdash \mathbb{M}$ *are coinductively derived by the type system below, by considering sessions up to structural congruence:*

$$\mathsf{End} \vdash \mathsf{p}[\,\mathbf{0}\,] \qquad \text{[T-End]}$$

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[\,P_i\,] \parallel \mathsf{q}[\,Q_i\,] \parallel \mathbb{N} \quad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{N}) \quad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\,P\,] \parallel \mathsf{q}[\,Q\,] \parallel \mathbb{N}} \text{ [T-Comm]}$$

where $P = \mathsf{q}!\{\lambda_i.P_i\}_{i\in I}$ *and* $Q = \mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}$.

Rule [T-Comm] just adds simultaneous communications to global types and to corresponding processes inside sessions. More inputs than corresponding outputs are allowed by this rule, in agreement with the condition in Rule [S-Comm] (Definition 2.3). It also allows more branches in the input process than in the global type, so mimicking local type subtyping [14]. Instead, the number of branches in the output process and in the global type must be the same. This does not restrict typability, as shown in [7], while it improves Session Fidelity. In fact, by allowing more branches in the global type than in the output process (and less branches in the global type than in the input process), we could not prove Session Fidelity as formulated in Theorem 5.4. We could only prove its following weaker version:

If $\mathsf{G} \vdash \mathbb{M}$ and $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}}$, then there are λ' and G' such that $\mathbb{M} \xrightarrow{\mathsf{p}\lambda'\mathsf{q}} \mathbb{M}'$ and $\mathsf{G}' \vdash \mathbb{M}'$.

Unwanted effects of having possibly infinite derivations are avoided by requiring that the global type and the session have exactly the same set of participants, as expressed by the condition $\mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{N})$ for all $i \in I$. Otherwise, the following judgement would be derivable for any $R \neq \mathbf{0}$:

$$\mathsf{p} \to \mathsf{q} : \lambda.\mathsf{G} \vdash \mathsf{p}[\,\mathsf{q}!\lambda.P\,] \parallel \mathsf{q}[\,\mathsf{p}?\lambda.Q\,] \parallel \mathsf{r}[\,R\,]$$

where $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \lambda.\mathsf{G}$, $P = \mathsf{q}!\lambda.P$ and $Q = \mathsf{p}?\lambda.Q$.

The regularity of processes and global types ensures the decidability of type checking.
We now illustrate the type system by deriving global types for the sessions of Examples 2.4 and 2.5. In the derivations, we omit the axiom/rule names as they are clear from the context. Moreover, we do not write the conditions on the participants, which can be easily checked.

Example 3.3. Consider the session $\mathbb{M}_{\mathsf{LG}}$ from Example 2.4. Then, by setting

$$\mathsf{G}_{\mathsf{LG}} = \mathsf{a} \to \mathsf{b} : \{\mathtt{love}.\mathsf{G}_{\mathsf{LG}},\ \mathtt{bye}.\mathsf{c} \to \mathsf{d} : \mathtt{hello}\}$$

we have the derivation $\mathcal{D}_{\mathsf{LG}}$:

$$\dfrac{\mathcal{D}_{\mathsf{LG}} \quad \dfrac{\mathsf{End} \vdash \mathsf{c}[\,\mathbf{0}\,] \parallel \mathsf{d}[\,\mathbf{0}\,]}{\mathsf{c} \to \mathsf{d} : \mathtt{hello} \vdash \mathsf{c}[\,\mathsf{d}!\mathtt{hello}\,] \parallel \mathsf{d}[\,\mathsf{c}?\mathtt{hello}\,]}}{\mathsf{G}_{\mathsf{LG}} \vdash \mathbb{M}_{\mathsf{LG}}}$$

The non-recursive type $\mathsf{c} \to \mathsf{d} : \mathtt{hello}$ is a subexpression of the recursive type G_{LG}. Alternatively, let $\mathsf{G}'_{\mathsf{LG}} = \mathsf{a} \to \mathsf{b} : \{\mathtt{love}.\mathsf{G}'_{\mathsf{LG}},\ \mathtt{bye}\}$ and $\mathcal{D}'_{\mathsf{LG}}$ be the derivation:

$$\dfrac{\mathcal{D}'_{\mathsf{LG}} \quad \mathsf{End} \vdash \mathsf{a}[\,\mathbf{0}\,] \parallel \mathsf{b}[\,\mathbf{0}\,]}{\mathsf{G}'_{\mathsf{LG}} \vdash \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]}$$

Then, by "pushing" the recursion inside, we have that the type $\mathsf{c} \to \mathsf{d} : \mathtt{hello}.\mathsf{G}'_{\mathsf{LG}}$ can be derived for $\mathbb{M}_{\mathsf{LG}}$ as follows:

$$\dfrac{\mathcal{D}'_{\mathsf{LG}}}{\mathsf{c} \to \mathsf{d} : \mathtt{hello}.\mathsf{G}'_{\mathsf{LG}} \vdash \mathbb{M}_{\mathsf{LG}}}$$

as $\mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,] \parallel \mathsf{c}[\,\mathbf{0}\,] \parallel \mathsf{d}[\,\mathbf{0}\,] \equiv \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$. ◇

Example 3.4. Consider the session $\mathbb{M}_{\mathsf{bsc}}$ from Example 2.5, and set

$$\mathsf{G}_{\mathsf{bsc}} = \mathsf{b} \to \mathsf{s} : \{\mathtt{item}.\mathsf{G}_{\mathsf{bsc}},\ \mathtt{buy}.\mathsf{s} \to \mathsf{c} : \mathtt{ship}\}$$

Then we have the following derivation $\mathcal{D}_{\mathsf{bsc}}$:

$$\dfrac{\mathcal{D}_{\mathsf{bsc}} \quad \dfrac{\mathsf{End} \vdash \mathsf{b}[\,\mathbf{0}\,] \parallel \mathsf{s}[\,\mathbf{0}\,] \parallel \mathsf{c}[\,\mathbf{0}\,]}{\mathsf{s} \to \mathsf{c} : \mathtt{ship} \vdash \mathsf{b}[\,\mathbf{0}\,] \parallel \mathsf{s}[\,\mathsf{c}!\mathtt{ship}\,] \parallel \mathsf{c}[\,\mathsf{s}?\mathtt{ship}\,]}}{\mathsf{G}_{\mathsf{bsc}} \vdash \mathbb{M}_{\mathsf{bsc}}}$$

Differently from the previous example, in $\mathsf{G}_{\mathsf{bsc}}$ the recursion cannot be pushed into $\mathsf{s} \to \mathsf{c} : \mathtt{ship}$. ◇

We use *paths*(G) to denote the set of paths in the syntactic tree of G. More formally, *paths*(G) is the greatest set of traces (as defined before Example 2.4) such that

$$\mathit{paths}(\mathsf{End}) = \emptyset$$

$$\mathit{paths}(\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\lambda_i\mathsf{q} \cdot \sigma_i \mid \sigma_i \in \mathit{paths}(\mathsf{G}_i)\}_{i\in I}$$

It is useful to check that the participants of a session and of its global type are the same.

Lemma 3.5. *If* $\mathsf{G} \vdash \mathbb{M}$*, then* $\mathsf{prt}(\mathsf{G}) = \mathsf{prt}(\mathbb{M})$.

Proof. The proof is by cases on the last applied axiom/rule in the derivation of $\mathsf{G} \vdash \mathbb{M}$. The case of the axiom is trivial.
For Rule [T-Comm] we have $\mathbb{M} \equiv \mathsf{p}[\,P\,] \parallel \mathsf{q}[\,Q\,] \parallel \mathbb{N}$ and

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[\,P_i\,] \parallel \mathsf{q}[\,Q_i\,] \parallel \mathbb{N} \quad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{N}) \quad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\,P\,] \parallel \mathsf{q}[\,Q\,] \parallel \mathbb{N}}$$

where $P = \mathsf{q}!\{\lambda_i.P_i\}_{i\in I}$ and $Q = \mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}$. Then

$$\mathsf{prt}(\mathsf{G}) = \{\mathsf{p},\mathsf{q}\} \cup \bigcup_{i\in I} \mathsf{prt}(\mathsf{G}_i) = \{\mathsf{p},\mathsf{q}\} \cup \mathsf{prt}(\mathbb{N}) = \mathsf{prt}(\mathbb{M})$$

where the second equality is justified by the condition

$$\mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{N})$$

□

4 THE COINDUCTIVE LTS

In SMPS if a session $\mathbb{M}$ has a global type G, then to get a global type for a session obtained by reducing $\mathbb{M}$ we need to reduce G. As a matter of fact, such a type reduction is a way to define the type semantics. In [4], by adapting a similar notion from [27], we considered the following LTS for global types.

Definition 4.1 (Inductive LTS for global types).

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}_i' \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i'\}_{i\in I}}$$

The axiom is called *external transition*, since the action $\mathsf{p}\lambda_j\mathsf{q}$ is among the communications between the participants p and q in the prefix $\mathsf{p} \to \mathsf{q}$. However, this is not sufficient to match all possible reductions in a session $\mathbb{M}$ when $\mathsf{G} \vdash \mathbb{M}$. Let $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}$. In $\mathbb{M}$ several redexes can occur, involving participants p and q, which can be distinct from r and s, and hence typed by some (subexpression of) G_i. Since such redexes can be independently reduced in $\mathbb{M}$, the second rule, called *internal transition*, allows the reduction of all the G_i's, provided that the same reduction is permitted for all of them. So that, for all $i \in I$, in any trace of G including $\mathsf{r}\lambda'$s either $\mathsf{p}\lambda\mathsf{q}$ will precede $\mathsf{r}\lambda'$s or vice versa. In fact, the set of traces of a global type G can be obtained from *paths*(G) by "swapping" (possibly infinite times) independent communication actions.

Now, if we take the judgement $\mathsf{G}_{\mathsf{LG}} \vdash \mathbb{M}_{\mathsf{LG}}$ from Example 3.3 we see that

$$\mathbb{M}_{\mathsf{LG}} \xrightarrow{\mathsf{c\ hello\ d}} \quad \text{but} \quad \mathsf{G}_{\mathsf{LG}} \not\xrightarrow{\mathsf{c\ hello\ d}}$$

as the internal reduction rule does not apply, hence breaking Subject Reduction in the formulation given in Theorem 5.2. For this reason, types like G_{LG} are ruled out in [4] by the *boundedness condition* (see Definition 6.5). In a nutshell, a type G is bounded whenever, if $\mathsf{p} \in \mathsf{prt}(\mathsf{G}')$ for a type G' which is a subexpression of G, then the search for an interaction of the shape $\mathsf{p}\lambda\mathsf{q}$ or $\mathsf{q}\lambda\mathsf{p}$ along all paths in G' terminates. As global types are finitary trees, this will happen within a number of steps which is bounded above by a single natural number, hence the name.

Still $\mathbb{M}_{\mathsf{LG}}$ is typable in the system in [4] by the type $\mathsf{G}'_{\mathsf{LG}}$, which is bounded and both $\mathsf{G}'_{\mathsf{LG}} \xrightarrow{\mathsf{c\ hello\ d}}$ (by an external reduction step) and $\mathsf{G}'_{\mathsf{LG}} \xrightarrow{\mathsf{a\ love\ b}}$ (by an internal one). However, also the type $\mathsf{G}_{\mathsf{bsc}}$ in Example 3.4 is unbounded, but $\mathbb{M}_{\mathsf{bsc}}$ cannot be typed with a bounded global type, as there exist infinitely many traces out of it deferring the transition labelled by `s ship c` an unbounded number of times. This motivates the search for a different LTS. In fact, we redefine the semantics of global types via a coinductive formal system.

In the next section we show in fact that – using the coinductive semantics of global types – Subject Reduction, Session Fidelity and Lock-freedom properties hold for sessions which are typable in the type system of Definition 3.2, without recurring to any further condition on global types, but regularity. This will enable to infer Lock-freedom for the Buyer-Seller-Carrier session of Example 2.5 from its typability (Example 3.3), a thing that would not be possible in systems allowing only bounded global types. Since boundedness and inhabitation imply projectability, as shown in Theorem 6.15, we get that the Buyer-Seller-Carrier session cannot be typed in systems which use projections.

It is handy to associate to global types sets of communication actions which could (but not necessarily can, see the example after Lemma 5.1) label their transitions. We dub them capabilities of global types.

Definition 4.2 (Capabilities). *Let* G *be a type. The set* $\mathsf{cap}(\mathsf{G})$ *of the* capabilities *of* G *is the smallest set satisfying the following equations:*

$$\mathsf{cap}(\mathsf{End}) = \emptyset$$

$$\mathsf{cap}(\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\lambda_i\mathsf{q} \mid i \in I\} \cup \textstyle\bigcup_{i\in I} \mathsf{cap}(\mathsf{G}_i)$$

By regularity, the set $\mathsf{cap}(\mathsf{G})$ is finite for all G.

Definition 4.3 (Coinductive LTS for global types).

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\text{E-Comm}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}_i' \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \quad \mathsf{p}\lambda\mathsf{q} \in \textstyle\bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i'\}_{i\in I}}\ [\text{I-Comm}]$$

The condition $\mathsf{p}\lambda\mathsf{q} \in \bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)$ in Rule [I-Comm] is needed, since otherwise we could get $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}$ for $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \lambda'.\mathsf{G}$.

The coinductive LTS allows more reductions of global types than the inductive one, as shown by the following example.

Example 4.4. Continuing Example 3.3, recall that

$$\mathsf{G}_{\mathsf{LG}} = \mathsf{a} \to \mathsf{b} : \{\mathtt{love}.\mathsf{G}_{\mathsf{LG}},\ \mathtt{bye}.\mathsf{c} \to \mathsf{d} : \mathtt{hello}\}$$

Since $\mathsf{cap}(\mathsf{c} \to \mathsf{d} : \mathtt{hello}) = \{\mathsf{c}\ \mathtt{hello}\ \mathsf{d}\}$, by definition we have

$$\begin{aligned}\mathsf{cap}(\mathsf{G}_{\mathsf{LG}}) &= \{\mathsf{a}\ \mathtt{love}\ \mathsf{b}, \mathsf{a}\ \mathtt{bye}\ \mathsf{b}\} \cup \{\mathsf{c}\ \mathtt{hello}\ \mathsf{d}\} \cup \mathsf{cap}(\mathsf{G}_{\mathsf{LG}}) \\ &= \{\mathsf{a}\ \mathtt{love}\ \mathsf{b}, \mathsf{a}\ \mathtt{bye}\ \mathsf{b}, \mathsf{c}\ \mathtt{hello}\ \mathsf{d}\}\end{aligned}$$

From this, we have both

$$\mathsf{G}_{\mathsf{LG}} \xrightarrow{\mathsf{a\ love\ b}} \mathsf{G}_{\mathsf{LG}} \quad \text{and} \quad \mathsf{G}_{\mathsf{LG}} \xrightarrow{\mathsf{a\ bye\ b}} (\mathsf{c} \to \mathsf{d} : \mathtt{hello})$$

by Rule [E-Comm] and, since $\mathsf{cap}(\mathsf{G}_{\mathsf{LG}}) \cap \mathsf{cap}(\mathsf{c} \to \mathsf{d} : \mathtt{hello}) = \{\mathsf{c}\ \mathtt{hello}\ \mathsf{d}\}$, by Rule [I-Comm] we have

$$\mathsf{G}_{\mathsf{LG}} \xrightarrow{\mathsf{c\ hello\ d}} \mathsf{G}'_{\mathsf{LG}}$$

where $\mathsf{G}'_{\mathsf{LG}} = \mathsf{a} \to \mathsf{b} : \{\mathtt{love}.\mathsf{G}'_{\mathsf{LG}}, \mathtt{bye}\}$ is the type of

$$\mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$$

Example 3.3 contains a derivation of $\mathsf{G}'_{\mathsf{LG}} \vdash \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$. ◇

It is interesting to notice that in the typing Rule [T-Comm] the session in the conclusion reduces to all the sessions in the premises using Rule [S-Comm] and the global type in the conclusion reduces to all the global types in the premises using Axiom [E-Comm].

5 SUBJECT REDUCTION, SESSION FIDELITY AND LOCK-FREEDOM

We begin this section with a technical lemma relating capabilities and possible reductions of a global type. It will be handy to prove the main results of the section.

Lemma 5.1. *If* $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'$, *then* $\mathsf{p}\lambda\mathsf{q} \in \mathsf{cap}(\mathsf{G})$.

Proof. By cases on the applied axiom/rule justifying $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'$. If this is [E-Comm], then $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j$ for some $j \in I$ such that $\lambda = \lambda_j$ and $\mathsf{p}\lambda\mathsf{q} \in \mathsf{cap}(\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I})$. Otherwise, $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}$ and $\mathsf{G}' = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}$ by Rule [I-Comm], where $\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i$ for all $i \in I$, $\{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset$ and $\mathsf{p}\lambda\mathsf{q} \in \bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)$. This implies $\mathsf{p}\lambda\mathsf{q} \in \mathsf{cap}(\mathsf{G})$, since $\bigcup_{i\in I} \mathsf{cap}(\mathsf{G}_i) \subseteq \mathsf{cap}(\mathsf{G})$ by Definition 4.2. □

The vice versa of the above lemma does not hold. It is immediate to check that, for $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\lambda_1.\mathsf{r} \to \mathsf{s} : \lambda_1,\ \lambda_2.\mathsf{r} \to \mathsf{s} : \lambda_2\}$, we have that $\mathsf{r}\lambda_1\mathsf{s} \in \mathsf{cap}(\mathsf{G})$, but $\mathsf{G} \xrightarrow{\mathsf{r}\lambda_1\mathsf{s}}\!\!\!\!\!/\ $.

Now, we establish the main features of our type system. We begin with Subject Reduction, the property ensuring that the transitions of typable sessions are mimicked by those of their global types.

Theorem 5.2 (Subject Reduction). *If* $\mathsf{G} \vdash \mathbb{M}$ *and* $\mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'$, *then* $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'$ *and* $\mathsf{G}' \vdash \mathbb{M}'$ *for some* G'.

Proof. By coinduction on the derivation of $\mathsf{G} \vdash \mathbb{M}$. By Lemma 2.6, if $\mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'$, then there exists a unique redex

$$\mathbb{R} = \mathsf{p}[\,\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}\,]$$

such that $\mathbb{M} \equiv \mathbb{R} \parallel \mathbb{N}$ and $\mathbb{M}' \equiv \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,] \parallel \mathbb{N}$ for some $\mathbb{N}$, where $k \in I \subseteq J$ and $\lambda = \lambda_k$. By the hypothesis that $\mathsf{G} \vdash \mathbb{M}$ we know that $\mathsf{G} \not\equiv \mathsf{End}$ and two cases of the last rule of this typing derivation are possible.

Case $\mathsf{G} \equiv \mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}$ and the derivation ends by

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[\,P_i\,] \parallel \mathsf{q}[\,Q_i\,] \parallel \mathbb{N} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{N}) \quad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\,\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}\,] \parallel \mathbb{N}}$$

It immediately follows that $\mathsf{G} \xrightarrow{\mathsf{p}\lambda_k\mathsf{q}} \mathsf{G}_k$ by Axiom [E-Comm], and $\mathsf{G}_k \vdash \mathbb{M}'$.

Case $\mathsf{G} \equiv \mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}_u\}_{u\in U}$ and $\{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset$, and the derivation ends by

$$\frac{\mathsf{G}_u \vdash \mathsf{r}[\,R_u\,] \parallel \mathsf{s}[\,S_u\,] \parallel \mathbb{N}' \parallel \mathbb{R} \qquad \mathsf{prt}(\mathsf{G}_u) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathbb{N}' \parallel \mathbb{R}) \qquad \forall u \in U \subseteq V}{\mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}_u\}_{u\in U} \vdash \mathbb{N} \parallel \mathbb{R}}$$

where $\mathbb{N} \equiv \mathsf{r}[\,\mathsf{s}!\{\lambda'_u.R_u\}_{u\in U}\,] \parallel \mathsf{s}[\,\mathsf{r}?\{\lambda'_v.S_v\}_{v\in V}\,] \parallel \mathbb{N}'$ and

$$\mathsf{prt}(\mathbb{N}' \parallel \mathbb{R}) = \mathsf{prt}(\mathbb{N}') \cup \{\mathsf{p},\mathsf{q}\}$$

Now, for all $u \in U$,

$$\mathbb{N}_u \parallel \mathbb{N}' \parallel \mathbb{R} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{N}_u \parallel \mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,]$$

where $\mathbb{N}_u = \mathsf{r}[\,R_u\,] \parallel \mathsf{s}[\,S_u\,]$, by contracting $\mathbb{R}$. By the coinduction hypothesis, we have $\mathsf{G}'_u \vdash \mathbb{N}_u \parallel \mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,]$ for some G'_u such that $\mathsf{G}_u \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_u$ and for all $u \in U$. Now by Lemma 5.1 we conclude that $\mathsf{p}\lambda\mathsf{q} \in \mathsf{cap}(\mathsf{G}_u)$ for all $u \in U$, hence taking

$$\mathsf{G}' = \mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}'_u\}_{u\in U}$$

we have that $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'$ by Rule [I-Comm]. By Lemma 3.5

$$\mathsf{G}'_u \vdash \mathbb{N}_u \parallel \mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,]$$

implies that $\mathsf{prt}(\mathsf{G}'_u) = \mathsf{prt}(\mathbb{N}_u \parallel \mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,])$, hence $\mathsf{prt}(\mathsf{G}'_u) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,])$. Therefore Rule [T-Comm] applies, namely

$$\frac{\mathsf{G}'_u \vdash \mathbb{N}_u \parallel \mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,] \qquad \mathsf{prt}(\mathsf{G}'_u) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathbb{N}' \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,]) \quad \forall u \in U \subseteq V}{\mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}'_u\}_{u\in U} \vdash \mathbb{N} \parallel \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,]}$$

We conclude by observing that the case $\mathsf{G} \equiv \mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}_u\}_{u\in U}$ with either $\mathsf{r} = \mathsf{p}$ and $\mathsf{s} \neq \mathsf{q}$ or $\mathsf{s} = \mathsf{q}$ and $\mathsf{r} \neq \mathsf{p}$ is impossible. □

Example 5.3. In Example 2.4 we have seen that

$$\mathbb{M}_{\mathsf{LG}} \xrightarrow{\mathsf{c}\,\mathtt{hello}\,\mathsf{d}} \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$$

and, from Example 3.3, we know that $\mathsf{G}_{\mathsf{LG}} \vdash \mathbb{M}_{\mathsf{LG}}$ where

$$\mathsf{G}_{\mathsf{LG}} = \mathsf{a} \to \mathsf{b} : \{\mathtt{love}.\mathsf{G}_{\mathsf{LG}},\ \mathtt{bye}.\mathsf{c} \to \mathsf{d} : \mathtt{hello}\}$$

Now, from Example 4.4 we have seen that

$$\mathsf{G}_{\mathsf{LG}} \xrightarrow{\mathsf{c}\,\mathtt{hello}\,\mathsf{d}} \mathsf{G}'_{\mathsf{LG}}$$

where $\mathsf{G}'_{\mathsf{LG}} = \mathsf{a} \to \mathsf{b} : \{\mathtt{love}.\mathsf{G}'_{\mathsf{LG}}, \mathtt{bye}\}$, and from Example 3.3 we know that $\mathsf{G}'_{\mathsf{LG}} \vdash \mathsf{a}[\,P_{\mathsf{LG}}\,] \parallel \mathsf{b}[\,Q_{\mathsf{LG}}\,]$, as expected by the theorem proved above. ◇

The property dubbed Session Fidelity implies that any reduction out of a global type G is also a reduction of any session typable with G.

Theorem 5.4 (Session Fidelity). *If* $\mathsf{G} \vdash \mathbb{M}$ *and* $\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'$, *then* $\mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'$ *and* $\mathsf{G}' \vdash \mathbb{M}'$ *for some* $\mathbb{M}'$.

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

Axiom [E-Comm]: then $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}_k$ and $\lambda = \lambda_k$ for some $k \in I$. Since $\mathsf{G} \neq \mathsf{End}$, the last rule in the derivation of $\mathsf{G} \vdash \mathbb{M}$ must be [T-Comm], which implies that

$$\mathbb{M} \equiv \mathsf{p}[\,\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}\,] \parallel \mathbb{N}$$

for some $\mathbb{N}$, and

$$\mathsf{G}_i \vdash \mathsf{p}[\,P_i\,] \parallel \mathsf{q}[\,Q_i\,] \parallel \mathbb{N}$$

for all $i \in I \subseteq J$. In this case we have $\mathbb{M} \xrightarrow{\mathsf{p}\lambda_k\mathsf{q}} \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,] \parallel \mathbb{N}$ by Rule [S-Comm] and $\mathsf{G}_k \vdash \mathsf{p}[\,P_k\,] \parallel \mathsf{q}[\,Q_k\,] \parallel \mathbb{N}$, since $k \in I \subseteq J$.

Rule [I-Comm]: then

$$\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}_u\}_{u\in U} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda'_u.\mathsf{G}'_u\}_{u\in U}$$

with $\{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset$, and $\mathsf{p}\lambda\mathsf{q} \in \mathsf{cap}(\mathsf{G}_u)$ and $\mathsf{G}_u \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_u$ for all $u \in U$. Since the last rule in the derivation of $\mathsf{G} \vdash \mathbb{M}$ must be [T-Comm], it follows that

- $\mathbb{M} \equiv \mathsf{r}[\,\mathsf{s}!\{\lambda'_u.R_u\}_{u\in U}\,] \parallel \mathsf{s}[\,\mathsf{r}?\{\lambda'_v.S_v\}_{v\in V}\,] \parallel \mathbb{N}$ for some $\mathbb{N}$ with $U \subseteq V$

- $G_u \vdash \mathsf{r}[\, R_u \,] \parallel \mathsf{s}[\, S_u \,] \parallel \mathbb{N}$ with $\mathsf{prt}(G_u) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathbb{N})$ for all $u \in U$.

By the coinduction hypothesis, we know that there exists $\mathbb{M}_u$ such that

$$\mathsf{r}[\, R_u \,] \parallel \mathsf{s}[\, S_u \,] \parallel \mathbb{N} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}_u \qquad \text{and} \qquad G'_u \vdash \mathbb{M}_u$$

for all $u \in U$. Notice that, being the label $\mathsf{p}\lambda\mathsf{q}$ the same for all these reductions, by Lemma 2.6 there exists a unique redex

$$\mathsf{p}[\, \mathsf{q}!\{\lambda_i.P_i\}_{i\in I} \,] \parallel \mathsf{q}[\, \mathsf{p}?\{\lambda_j.Q_j\}_{j\in J} \,]$$

with contractum $\mathsf{p}[\, P_k \,] \parallel \mathsf{q}[\, Q_k \,]$ (assuming $\lambda = \lambda_k$) in all the $\mathbb{M}_u$.

On the other hand, since we know that p, q are distinct from r, s, it must be the case that $\mathbb{M}_u \equiv \mathsf{r}[\, R_u \,] \parallel \mathsf{s}[\, S_u \,] \parallel \mathbb{N}'$ for some $\mathbb{N}'$ such that $\mathbb{N} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{N}'$ so that

$$\mathsf{r}[\, \mathsf{s}!\{\lambda'_u.R_u\}_{u\in U} \,] \parallel \mathsf{s}[\, \mathsf{r}?\{\lambda'_v.S_v\}_{v\in V} \,] \parallel \mathbb{N} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r}[\, \mathsf{s}!\{\lambda'_u.R_u\}_{u\in U} \,] \parallel \mathsf{s}[\, \mathsf{r}?\{\lambda'_v.S_v\}_{v\in V} \,] \parallel \mathbb{N}'$$

Then we can set $\mathbb{M}' = \mathsf{r}[\, \mathsf{s}!\{\lambda'_u.R_u\}_{u\in U} \,] \parallel \mathsf{s}[\, \mathsf{r}?\{\lambda'_v.S_v\}_{v\in V} \,] \parallel \mathbb{N}'$. By Lemma 3.5 $G'_u \vdash \mathsf{r}[\, R_u \,] \parallel \mathsf{s}[\, S_u \,] \parallel \mathbb{N}'$ implies

$$\mathsf{prt}(G'_u) = \mathsf{prt}(\mathsf{r}[\, R_u \,] \parallel \mathsf{s}[\, S_u \,] \parallel \mathbb{N}')$$

and then $\mathsf{prt}(G'_u) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathbb{N}')$ for all $u \in U$. We conclude that there exists a derivation ending by the following application of Rule [T-Comm]

$$\frac{G'_u \vdash \mathsf{r}[\, R_u \,] \parallel \mathsf{s}[\, S_u \,] \parallel \mathbb{N}' \qquad \mathsf{prt}(G'_u) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathbb{N}') \qquad \forall u \in U \subseteq V}{\mathsf{r} \to \mathsf{s} : \{\lambda_u.G'_u\}_{u\in U} \vdash \mathbb{M}'}$$

□

Remark 5.5. Since transitions of sessions and global types are both labelled by communication actions, they can be merged into a unique LTS. Then an immediate consequence of Theorems 5.2 and 5.4 is that, if $G \vdash \mathbb{M}$, then G and $\mathbb{M}$ are bisimilar, roughly meaning that they have the same computational content. The above implies that if both G and G′ type the same $\mathbb{M}$, then they are bisimilar to each other. ◇

Toward establishing the property that typable sessions are lock free, we first prove the following lemma. In words, if $\mathsf{p} \in \mathsf{prt}(G)$, 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 action Λ involving p.

Lemma 5.6. *If* $\mathsf{p} \in \mathsf{prt}(G)$, *then there exist* σ, Λ *and* G' *such that*

$$G \xrightarrow{\sigma\cdot\Lambda} G'$$

where $\mathsf{p} \notin \mathsf{prt}(\sigma)$ *and* $\mathsf{p} \in \mathsf{prt}(\Lambda)$.

Proof. The proof is by coinduction on G. Since $\mathsf{p} \in \mathsf{prt}(G)$ we have that $G \neq \mathsf{End}$, so that $G = \mathsf{r} \to \mathsf{s} : \{\lambda_i.G_i\}_{i\in I}$. Now, if $\mathsf{p} \in \{\mathsf{r},\mathsf{s}\}$ then, taking $\Lambda = \mathsf{r}\lambda_i\mathsf{s}$ for any $i \in I$, we immediately have that $G \xrightarrow{\Lambda} G'$ by Axiom [E-Comm], and the thesis trivially follows by taking $\sigma = \varepsilon$. Otherwise, since $\mathsf{p} \in \mathsf{prt}(G) = \{\mathsf{r},\mathsf{s}\} \cup \bigcup_{i\in I} \mathsf{prt}(G_i)$, we have that $\mathsf{p} \notin \{\mathsf{r},\mathsf{s}\}$ implies $\mathsf{p} \in \mathsf{prt}(G_k)$ for some $k \in I$. By coinduction hypothesis, we have that there are a σ' and a Λ such that $G_k \xrightarrow{\sigma'\cdot\Lambda} G'$, $\mathsf{p} \notin \mathsf{prt}(\sigma')$ and $\mathsf{p} \in \mathsf{prt}(\Lambda)$. Then the thesis follows by setting $\sigma = \mathsf{r}\lambda_k\mathsf{s} \cdot \sigma'$, since $G \xrightarrow{\mathsf{r}\lambda_k\mathsf{s}} G_k$ by Axiom [E-Comm] and $G_k \xrightarrow{\sigma'\cdot\Lambda} G'$. □

Observe that the last lemma is a sort of inverse implication w.r.t. Lemma 5.1, 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 $\mathsf{prt}(G)$.

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

Theorem 5.7. *If* $\mathbb{M}$ *is typable, then* $\mathbb{M}$ *is lock free.*

Proof. By Theorem 5.2 and an easy induction over the length of σ we have that $G \vdash \mathbb{M}$ and $\mathbb{M} \xrightarrow{\sigma} \mathbb{M}'$ imply that $G \xrightarrow{\sigma} G'$ for some G′ such that $G' \vdash \mathbb{M}'$. Hence it suffices to show Lock-freedom (Definition 2.7) simply for the case when $\sigma = \varepsilon$ and $\mathbb{M}' = \mathbb{M}$ and $G' = G$.

Let $\mathsf{p} \in \mathsf{prt}(\mathbb{M})$, then $\mathsf{p} \in \mathsf{prt}(G)$ by Lemma 3.5. On the other hand, from the fact that $\mathsf{p} \in \mathsf{prt}(G)$ and by Lemma 5.6 it follows that $G \xrightarrow{\sigma\cdot\Lambda} G'$ 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 5.4). □

Example 5.8. From Example 3.4 we know that

$$G_{\mathsf{bsc}} \vdash \mathbb{M}_{\mathsf{bsc}}$$

From the previous result, we get that $\mathbb{M}_{\mathsf{bsc}}$ is lock free. ◇

6 PROJECTABILITY AND BOUNDEDNESS

We focus now on two important properties of global types that have been investigated in the literature. The "classical" one is *projectability*. Its relevance is based on the fact that, by projecting a global type G, we get local types that must be derivable for the processes implementing the protocol represented by G, see [26, 27, 35]. Projectability is not required when the global types are directly derived for the sessions, as in [4, 6, 13] and in the present paper. In all the works using the last mentioned approach to global types for sessions, global types are required to be *bounded* (a notion equivalent to that of *balancing* used in [21]).

The type system of the previous section is the first one which requires neither projectability nor boundedness of global types.

The main results of the present section concern the relationship between the above properties. In particular we show that

- projectability implies boundedness and inhabitation (Theorem 6.14 below);
- boundedness and inhabitation imply projectability (Theorem 6.15 below),

where G is *inhabited* if there exists a session $\mathbb{M}$ such that $G \vdash \mathbb{M}$.

We start by straightforwardly adapting the classical projection to the present case where we consider directly processes instead of local types.

Definition 6.1 (Projections). *Given a global type* G *and a participant* p, *the process* $G\upharpoonright\mathsf{p}$, *called* the projection of G at p, *is coinductively defined by:*

$$\mathrm{G}\upharpoonright\mathsf{p} = \mathbf{0} \qquad \textit{if } \mathsf{p} \notin \mathrm{prt}(\mathrm{G})$$

$$(\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathrm{G}_i\}_{i\in I})\upharpoonright\mathsf{p} = \begin{cases} \mathsf{s}!\{\lambda_i.(\mathrm{G}_i\upharpoonright\mathsf{p})\}_{i\in I} & \textit{if } \mathsf{p} = \mathsf{r} \\ \mathsf{r}?\{\lambda_i.(\mathrm{G}_i\upharpoonright\mathsf{p})\}_{i\in I} & \textit{if } \mathsf{p} = \mathsf{s} \\ \bigsqcap_{i\in I}(\mathrm{G}_i\upharpoonright\mathsf{p}) & \textit{if } \mathsf{p} \in \mathrm{prt}(\mathrm{G}) \textit{ and } \mathsf{p} \notin \{\mathsf{r},\mathsf{s}\} \end{cases}$$

where the (full) merge $P \sqcap Q$ *is the partial operation corecursively defined by*

$$\mathbf{0} \sqcap \mathbf{0} = \mathbf{0}$$

$$\mathsf{q}!\{\lambda_i.P_i\}_{i\in I} \sqcap \mathsf{q}!\{\lambda_i.Q_i\}_{i\in I} = \mathsf{q}!\{\lambda_i.P_i \sqcap Q_i\}_{i\in I}$$

$$\mathsf{q}?\{\lambda_i.P_i\}_{i\in I} \sqcap \mathsf{q}?\{\lambda_i.Q_j\}_{i\in J} = \mathsf{q}?\{\lambda_k.R_k\}_{k\in I\cup J}$$

and the R_k *are defined by*

$$R_k = \begin{cases} P_k \sqcap Q_k & \textit{if } k \in I \cap J \\ P_k & \textit{if } k \in I \setminus J \\ Q_k & \textit{if } k \in J \setminus I \end{cases}$$

We say that G *is* projectable *if* $\mathrm{G}\upharpoonright\mathsf{p}$ *is defined for all* $\mathsf{p} \in \mathrm{prt}(\mathrm{G})$.

In the literature there are various definitions of merge. We use the more permissive one as defined in [16] and called full merge in [33].

Example 6.2. By taking

$$\mathrm{G}_{\mathrm{bsc}} = \mathsf{b} \to \mathsf{s} : \{\mathtt{item}.\mathrm{G}_{\mathrm{bsc}},\ \mathtt{buy}.\mathsf{s} \to \mathsf{c} : \mathtt{ship}\}$$

as defined in Example 3.4, we get $\mathrm{G}_{\mathrm{bsc}}\upharpoonright\mathsf{b} = \mathsf{s}!\{\mathtt{item}.P_{\mathrm{bsc}}, \mathtt{buy}\}$ and $\mathrm{G}_{\mathrm{bsc}}\upharpoonright\mathsf{s} = \mathsf{b}?\{\mathtt{item}.Q_{\mathrm{bsc}}, \mathtt{buy}.\mathsf{c}!\mathtt{ship}\}$, i.e. $\mathrm{G}_{\mathrm{bsc}}\upharpoonright\mathsf{b} = P_{\mathrm{bsc}}$ and $\mathrm{G}_{\mathrm{bsc}}\upharpoonright\mathsf{s} = Q_{\mathrm{bsc}}$ as defined in Example 2.5. Instead $\mathrm{G}_{\mathrm{bsc}}\upharpoonright\mathsf{c}$ is undefined, as shown after Theorem 6.15. ◇

The definition of projection implies that, if $\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathrm{G}_i\}_{i\in I}$ is projectable, then $\mathrm{prt}(\mathrm{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathrm{prt}(\mathrm{G}_j) \setminus \{\mathsf{p},\mathsf{q}\}$ for all $i, j \in I$. The same condition is required by the typing Rule [T-Comm]. The vice versa does not hold, i.e. if $\mathrm{G} = \mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathrm{G}_i\}_{i\in I}$, then $\mathrm{prt}(\mathrm{G}_i)\setminus\{\mathsf{p},\mathsf{q}\} = \mathrm{prt}(\mathrm{G}_j)\setminus\{\mathsf{p},\mathsf{q}\}$ for all $i, j \in I$ does not imply projectability of G. This is illustrated in the next example.

Example 6.3. In

$$\mathrm{G} = \mathsf{p} \to \mathsf{q} : \{\lambda.\mathsf{r} \to \mathsf{p} : \lambda,\ \lambda'\}$$

the immediate subexpressions are $\mathsf{r} \to \mathsf{p} : \lambda$ and End. Then we get $\mathrm{prt}(\mathsf{r} \to \mathsf{p} : \lambda) \setminus \{\mathsf{p},\mathsf{q}\} = \{\mathsf{r}\} \neq \emptyset = \mathrm{prt}(\mathrm{End}) \setminus \{\mathsf{p},\mathsf{q}\}$. Hence G is un-projectable. Instead in

$$\mathrm{G}' = \mathsf{p} \to \mathsf{q} : \{\lambda_1.\mathsf{r} \to \mathsf{s} : \lambda,\ \lambda_2.\mathsf{s} \to \mathsf{r} : \lambda\}$$

the immediate subexpressions have the very same participants, but G' is un-projectable. In fact we get $\mathrm{G}'\upharpoonright\mathsf{r} = \mathsf{s}!\lambda \sqcap \mathsf{s}?\lambda$, which is undefined. ◇

We now proceed by recalling the boundedness property (Definitions 6.1 and 6.5 below). To define boundedness it is handy to take $\mathbf{N} \cup \{\infty\}$ as the set ordered by $n \leq m$ for $n, m \in \mathbf{N}$ in the ordinary sense, plus $n < \infty$ for all $n \in \mathbf{N}$. For any path σ we define $|\sigma|$ as its length if it is finite, and $|\sigma| = \infty$ otherwise. Also, if $n \in \mathbf{N}$ and $1 \leq n \leq |\sigma|$, then $\sigma[n]$ denotes the n-th communication action in σ.

Definition 6.4 (Depth). *Let* G *be a global type. For* $\sigma \in \mathit{paths}(\mathrm{G})$ *we define*

$$\mathit{depth}(\sigma, \mathsf{p}) = \inf\{n \in \mathbf{N} \mid 1 \leq n \leq |\sigma| \ \ \&\ \ \mathsf{p} \in \mathrm{prt}(\sigma[n])\}$$

where $\inf\ \emptyset = \infty$.

We also define $\mathit{depth}(\mathrm{G}, \mathsf{p})$, *the* depth *of* p *in* G, *as follows:*

$$\mathit{depth}(\mathrm{G}, \mathsf{p}) = \begin{cases} \sup\{\mathit{depth}(\sigma, \mathsf{p}) \mid \sigma \in \mathit{paths}(\mathrm{G})\} & \textit{if } \mathsf{p} \in \mathrm{prt}(\mathrm{G}) \\ 0 & \textit{otherwise} \end{cases}$$

Definition 6.5 (Boundedness). *A global type* G *is* bounded *if* $\mathit{depth}(\mathrm{G}', \mathsf{p})$ *is finite for all participants* $\mathsf{p} \in \mathrm{prt}(\mathrm{G}')$ *and all subexpressions* G' *of* G *which are global types.*

Example 6.6. The type End is trivially bounded, since $\mathrm{prt}(\mathrm{End}) = \emptyset$. Types $\mathrm{G}_0 = \mathsf{p} \to \mathsf{q} : \lambda$ and $\mathrm{G}_1 = \mathsf{p} \to \mathsf{q} : \lambda.\mathrm{G}_1$ are immediately seen to be bounded. The type

$$\mathrm{G}_{\mathrm{sca}} = \mathsf{s} \to \mathsf{c} : \begin{cases} \mathtt{login}.\mathsf{c} \to \mathsf{a} : \mathtt{pwd}.\ \mathsf{a} \to \mathsf{s} : \mathtt{auth} \\ \mathtt{cancel}.\mathsf{c} \to \mathsf{a} : \mathtt{quit} \end{cases}$$

of the authentication protocol in Section 7 is finite and bounded. However, not any finite type is bounded: take

$$\mathrm{G}_2 = \mathsf{r} \to \mathsf{s} : \{\lambda_1,\ \lambda_2 : \mathsf{r} \to \mathsf{p} : \lambda_3\}.$$

Then $\mathsf{p} \in \mathrm{prt}(\mathrm{G}_2)$, but $\mathit{depth}(\mathsf{r}\lambda_1\mathsf{s}, \mathsf{p}) = \infty$.

Type $\mathrm{G}'_{\mathsf{LG}}$ in Example 3.3 is infinite and bounded, while G_{LG} in the same example and $\mathrm{G}_{\mathrm{bsc}}$ in Example 3.4 are unbounded. ◇

As a first step towards the proof of the main results of this section we show now two simple lemmas. The first lemma deals with projections. The second lemma relates the depth of a global type to the depths of its branches.

Lemma 6.7.

(1) *If* $\mathrm{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathrm{G}_i\}_{i\in I}$ *is projectable, then* G_i *is projectable for each* $i \in I$.
(2) *If* $\mathsf{p} \in \mathrm{prt}(\mathrm{G})$ *and* $\mathrm{G}\upharpoonright\mathsf{p}$ *is defined, then* $\mathrm{G}\upharpoonright\mathsf{p} \neq \mathbf{0}$.

Proof. (1). Easy, since $\mathrm{G}\upharpoonright\mathsf{p}$ is defined using $\mathrm{G}_i\upharpoonright\mathsf{p}$ for each $i \in I$.
(2). Immediate from the definition of $\mathrm{G}\upharpoonright\mathsf{p}$ and the fact that $\mathbf{0}$ is mergeable only with itself. □

Lemma 6.8. *If* $\mathrm{G} = \mathsf{q} \to \mathsf{r} : \{\lambda_i.\mathrm{G}_i\}_{i\in I}$, $\mathsf{p} \in \mathrm{prt}(\mathrm{G})$ *and* $\mathsf{p} \notin \{\mathsf{q},\mathsf{r}\}$, *then* $\mathit{depth}(\mathrm{G}, \mathsf{p}) = 1 + \sup\{\mathit{depth}(\mathrm{G}_i, \mathsf{p}) \mid i \in I\}$.

Proof. It is enough to observe that $\sigma \in \mathit{paths}(\mathrm{G})$ implies $\sigma = \mathsf{p}\lambda_i\mathsf{q} \cdot \sigma'$ for some $\sigma' \in \mathit{paths}(\mathrm{G}_i)$ and $i \in I$. □

It is handy to define a preorder on processes, $P \leqslant Q$, meaning, roughly, that *process P can be used where we expect process Q*, More precisely, $P \leqslant Q$ if either P is equal to Q, or we are in one of two situations: either both P and Q are output processes, sending the same labels to the same participant, and after the send P continues with a process that can be used when we expect the corresponding one in Q; or they are both input processes receiving labels from the same participant, and P may receive more labels than Q (and thus has more behaviours) but, whenever it receives the same label as Q, it continues with a process that can be used when we expect the corresponding one in Q. The rules in the definition below are interpreted coinductively, since the processes may be infinite. However, this preorder is decidable, since processes have finitely many distinct subprocesses.

Definition 6.9. *The preorder* $\leqslant$ *between processes is coinductively defined by*

$$\mathbf{0} \leqslant \mathbf{0} \quad [\text{sub-}\mathbf{0}] \qquad \frac{P_i \leqslant Q_i \qquad \forall i \in I}{\mathsf{q}!\{\lambda_i.P_i\}_{i\in I} \leqslant \mathsf{q}!\{\lambda_i.Q_i\}_{i\in I}} \ [\text{sub-Out}]$$

$$\frac{P_i \leqslant Q_i \qquad \forall i \in I}{\mathsf{q}?\{\lambda_i.P_i\}_{i\in I\cup J} \leqslant \mathsf{q}?\{\lambda_i.Q_i\}_{i\in I}} \ [\text{sub-In}]$$

This relation can be seen as a restricted version of the subtyping relation defined in [14]. Such a restriction does pair with the requirement in Rule [T-Comm] of the type system, where the set of indexes for the branches of the output process is the same as that for the branches in the global type, whereas the input process can have more branches.

The meaning of the preorder on processes is exploited in the following lemma.

Lemma 6.10. *If* $\mathsf{G} \vdash \mathsf{p}[\,Q\,] \parallel \mathbb{M}$ *and* $P \leqslant Q$, *then* $\mathsf{G} \vdash \mathsf{p}[\,P\,] \parallel \mathbb{M}$.

Proof. We proceed by coinduction over the derivation of

$$\mathsf{G} \vdash \mathsf{p}[\,Q\,] \parallel \mathbb{M}$$

We distinguish two cases according to the last applied axiom/rule.

Axiom [T-End]. Immediate.

Rule [T-Comm]. In this case the last applied rule has the shape

$$\frac{\mathsf{G}_i \vdash \mathsf{r}[\,R_i\,] \parallel \mathsf{s}[\,S_i\,] \parallel \mathsf{p}[\,Q\,] \parallel \mathbb{N} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathsf{p}[\,Q\,] \parallel \mathbb{N}) \qquad \forall i \in I \subseteq J}{\mathsf{G} \vdash \mathsf{r}[\,\mathsf{s}!\{\lambda_i.R_i\}_{i\in I}\,] \parallel \mathsf{s}[\,\mathsf{r}?\{\lambda_j.S_j\}_{j\in J}\,] \parallel \mathsf{p}[\,Q\,] \parallel \mathbb{N}}$$

where $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}$. Notice that $\mathsf{prt}(\mathsf{p}[\,Q\,] \parallel \mathbb{N}') = \mathsf{prt}(\mathsf{p}[\,P\,] \parallel \mathbb{N}')$ for any $\mathbb{N}'$ by definition of $\leqslant$, since $\mathbf{0}$ is related only with itself. We have that $\mathsf{G}_i \vdash \mathsf{r}[\,R_i\,] \parallel \mathsf{s}[\,S_i\,] \parallel \mathsf{p}[\,P\,] \parallel \mathbb{N}$ for each $i \in I$ by coinduction. For the cases $\mathsf{p} \notin \{\mathsf{r},\mathsf{s}\}$ and $\mathsf{p} = \mathsf{r}$, we get the thesis easily by applying Rule [T-Comm]. In case $\mathsf{p} = \mathsf{s}$ we have that $Q = \mathsf{r}?\{\lambda_j.Q_j\}_{j\in J}$ and, by definition of $\leqslant$, $P = \mathsf{r}?\{\lambda_h.P_h\}_{h\in H}$ with $J \subseteq H$ and $P_j \leqslant Q_j$ for each $j \in J$. By coinduction we get that $\mathsf{G}_i \vdash \mathsf{r}[\,R_i\,] \parallel \mathsf{s}[\,Q_i\,] \parallel \mathbb{N}$ implies $\mathsf{G}_i \vdash \mathsf{r}[\,R_i\,] \parallel \mathsf{s}[\,P_i\,] \parallel \mathbb{N}$ for each $i \in I$. Being $I \subseteq H$ we can get the thesis by applying Rule [T-Comm]. □

The following two lemmas establish connections between the preorder on processes and process merge.

Lemma 6.11. *Let* P, V *and* W *be such that* $P \leqslant V$ *and* $P \leqslant W$. *Then* $V \sqcap W$ *is defined and* $P \leqslant V \sqcap W$.

Proof. By coinduction on P. We distinguish among the different shapes of P.

- $P = \mathbf{0}$. In such a case the thesis follows immediately.
- $P = \mathsf{q}!\{\lambda_i.P_i\}_{i\in I}$. By definition of $\leqslant$ we have necessarily that
$$V = \mathsf{q}!\{\lambda_i.V_i\}_{i\in I} \quad W = \mathsf{q}!\{\lambda_i.W_i\}_{i\in I}$$
and, for all $i \in I$,
$$P_i \leqslant V_i \qquad P_i \leqslant W_i$$
Now, we can recur to coinduction to get that, for all $i \in I$,
$$V_i \sqcap W_i \text{ is defined and } P_i \leqslant V_i \sqcap W_i$$
By definition of $\sqcap$, we have that
$$V \sqcap W = \mathsf{q}!\{\lambda_i.V_i \sqcap W_i\}_{i\in I}$$
So, $V \sqcap W$ is defined and, by definition of $\leqslant$, we can infer $P \leqslant V \sqcap W$.
- $P = \mathsf{q}?\{\lambda_i.P_i\}_{i\in I}$. By definition of $\leqslant$ we have necessarily that
$$V = \mathsf{q}?\{\lambda_h.V_h\}_{h\in H} \quad W = \mathsf{q}?\{\lambda_k.W_k\}_{k\in K} \quad H \subseteq I \quad K \subseteq I$$
and, for each $i \in H \cap K$,
$$P_i \leqslant V_i \qquad P_i \leqslant W_i$$
Now, we can recur to coinduction to get that, for each $i \in H \cap K$,
$$V_i \sqcap W_i \text{ is defined and } P_i \leqslant V_i \sqcap W_i$$
By definition of $\sqcap$, we have that
$$V \sqcap W = \mathsf{q}?\{\lambda_j.Z_j\}_{j\in H\cup K}$$
where $Z_j = V_j \sqcap W_j$ if $j \in H \cap K$, $Z_j = V_j$ if $j \in H \setminus K$ and $Z_j = W_j$ if $j \in K \setminus H$. So, $V \sqcap W$ is defined and, since $H \cup K \subseteq I$, we have also $P \leqslant V \sqcap W$ by definition of $\leqslant$. □

Lemma 6.12. *Let* P *and* Q *be such that* $P \sqcap Q$ *is defined. Then* $P \sqcap Q \leqslant P$.

Proof. By coinduction on P. We distinguish among the different shapes of P.

- $P = \mathbf{0}$. In such a case the thesis follows immediately.
- $P = \mathsf{p}!\{\lambda_i.P_i\}_{i\in I}$. In such a case, necessarily $Q = \mathsf{p}!\{\lambda_i.Q_i\}_{i\in I}$ and $P \sqcap Q = \mathsf{p}!\{\lambda_i.P_i \sqcap Q_i\}_{i\in I}$. The thesis hence follows by coinduction and definition of $\leqslant$.
- $P = \mathsf{p}?\{\lambda_i.P_i\}_{i\in I}$. In such a case, necessarily $Q = \mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}$ and $P \sqcap Q = \mathsf{p}?\{\lambda_h.Z_h\}_{h\in I\cup J}$ where $Z_h = P_h \sqcap Q_h$ if $h \in H \cap J$, $Z_h = P_h$ if $h \in I \setminus J$ and $Z_h = Q_h$ if $h \in J \setminus I$. The thesis hence follows by coinduction and definition of $\leqslant$. □

By Lemmas 6.11 and 6.12, the merge $P \sqcap Q$, when defined, is the meet of P and Q w.r.t. $\leqslant$.

The relation $\leqslant$ between processes is useful to relate the processes in a typable session with the projection of the global type on the corresponding participants, see the following lemma.

Lemma 6.13. *If* $\mathsf{G} \vdash \mathbb{M}$ *and* $\mathsf{G}\upharpoonright\mathsf{p}$ *is defined, then* $\mathbb{M} \equiv \mathsf{p}[\,P\,] \parallel \mathbb{N}$ *and* $P \leqslant \mathsf{G}\upharpoonright\mathsf{p}$.

Proof. By coinduction on the derivation of $\mathsf{G} \vdash \mathbb{M}$. We distinguish two case according to the possible last applied axiom/rule.

Axiom [T-End]. Immediate.

Rule [T-Comm]. In this case the last applied rule has the shape

$$\frac{\mathsf{G}_i \vdash \mathsf{r}[\,R_i\,] \parallel \mathsf{s}[\,S_i\,] \parallel \mathsf{p}[\,P\,] \parallel \mathbb{N} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{r},\mathsf{s}\} = \mathsf{prt}(\mathsf{p}[\,P\,] \parallel \mathbb{N}) \qquad \forall i \in I \subseteq J}{\mathsf{G} \vdash \mathsf{r}[\,\mathsf{s}!\{\lambda_i.R_i\}_{i\in I}\,] \parallel \mathsf{s}[\,\mathsf{r}?\{\lambda_j.S_j\}_{j\in J}\,] \parallel \mathsf{p}[\,P\,] \parallel \mathbb{N}}$$

where $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}$. By definition of projection $\mathsf{G}_i\upharpoonright\mathsf{p}$ is defined for all $i \in I$. By coinduction we have that $P_i \leqslant \mathsf{G}_i\upharpoonright\mathsf{p}$ for all $i \in I$. If $\mathsf{p} = \mathsf{r}$, by definition of projection, $\mathsf{G}\upharpoonright\mathsf{p} = \mathsf{s}!\{\lambda_i.\mathsf{G}_i\upharpoonright\mathsf{p}\}_{i\in I}$. If $\mathsf{p} = \mathsf{s}$, by definition of projection, $\mathsf{G}\upharpoonright\mathsf{p} = \mathsf{r}?\{\lambda_i.\mathsf{G}_i\upharpoonright\mathsf{p}\}_{i\in I}$. In both cases the thesis follows by definition of $\leqslant$. If $\mathsf{p} \notin \{\mathsf{r},\mathsf{s}\}$, by definition of projection $\mathsf{G}\upharpoonright\mathsf{p} = \bigsqcap_{i\in I} \mathsf{G}_i\upharpoonright\mathsf{p}$. We hence can get the thesis by Lemma 6.11. □

We are ready now for proving the main results of this section.

Theorem 6.14. *Projectability implies boundedness and inhabitation.*

Proof. Let G be a projectable global type. We prove boundedness and inhabitation separately.

G *is bounded*. The proof is by coinduction over G. If $\mathsf{G} = \mathsf{End}$, then

the thesis is trivial. If $\mathrm{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathrm{G}_i\}_{i\in I}$, then from the hypothesis that G is projectable, it follows that all the G_i's are projectable by Lemma 6.7(1). Then, by coinduction we have that all the G_i's are bounded.

If $\mathsf{p} \notin \mathrm{prt}(\mathrm{G})$, then $depth(\mathrm{G}, \mathsf{p}) = 0$. Let $\mathsf{p} \in \mathrm{prt}(\mathrm{G})$. If $\mathsf{p} \in \{\mathsf{r}, \mathsf{s}\}$, then $depth(\mathrm{G}, \mathsf{p}) = 1$. Otherwise

$$depth(\mathrm{G}, \mathsf{p}) = 1 + \sup\{depth(\mathrm{G}_i, \mathsf{p}) \mid i \in I\}$$

by Lemma 6.8. So, in both cases $depth(\mathrm{G}, \mathsf{p})$ is finite. Moreover, a proper subexpression G' of G is a subexpression of G_j for some $j \in I$. Then the boundedness of G_j implies that $depth(\mathrm{G}', \mathsf{p})$ is finite. We hence conclude that G is bounded.

G *is inhabited.* If $\mathrm{G} = \mathrm{End}$, then $\mathrm{G} \vdash \mathbb{M}$ for all $\mathbb{M} \equiv \mathsf{p}[\,\mathbf{0}\,]$. Otherwise, we first define $\mathrm{G}\!\upharpoonright$ as the session with $\mathrm{prt}(\mathrm{G})$ as set of participants and with the projections of G on the participants in $\mathrm{prt}(\mathrm{G})$ as processes. That is

$$\mathrm{G}\!\upharpoonright = \Big\|_{\mathsf{p}\in\mathrm{prt}(\mathrm{G})} \mathsf{p}[\,\mathrm{G}\!\upharpoonright\mathsf{p}\,]$$

Hence we proceed to show that $\mathrm{G} \vdash \mathrm{G}\!\upharpoonright$ by coinduction on G. We observe that

$$(\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathrm{G}_i\}_{i\in I})\!\upharpoonright = \mathsf{p}[\,\mathsf{q}!\{\lambda_i.\mathrm{G}_i\!\upharpoonright\mathsf{p}\}_{i\in I}\,] \parallel \mathsf{q}[\,\mathsf{p}?\{\lambda_i.\mathrm{G}_i\!\upharpoonright\mathsf{q}\}_{i\in I}\,] \Big\|_{\mathsf{r}\in\mathrm{prt}(\mathrm{G})\setminus\{\mathsf{p},\mathsf{q}\}} \mathsf{r}[\,\mathrm{G}\!\upharpoonright\mathsf{r}\,]$$

where, by the assumption that G is projectable and by definition of projection, for each $\mathsf{r} \in \mathrm{prt}(\mathrm{G}) \setminus \{\mathsf{p}, \mathsf{q}\}$,

$$\mathrm{G}\!\upharpoonright\mathsf{r} = \bigsqcap_{i\in I} \mathrm{G}_i\!\upharpoonright\mathsf{r}$$

By coinduction we have that, for all $i \in I$, $\mathrm{G}_i \vdash \mathrm{G}_i\!\upharpoonright$, where by definition,

$$\mathrm{G}_i\!\upharpoonright \equiv \mathsf{p}[\,\mathrm{G}_i\!\upharpoonright\mathsf{p}\,] \parallel \mathsf{q}[\,\mathrm{G}_i\!\upharpoonright\mathsf{q}\,] \Big\|_{\mathsf{r}\in\mathrm{prt}(\mathrm{G}_i)\setminus\{\mathsf{p},\mathsf{q}\}} \mathsf{r}[\,\mathrm{G}_i\!\upharpoonright\mathsf{r}\,]$$

Now, by Lemma 6.12 we have that, for all $i \in I$,

$$\mathrm{G}\!\upharpoonright\mathsf{r} = \bigsqcap_{i\in I} \mathrm{G}_i\!\upharpoonright\mathsf{r} \leqslant \mathrm{G}_i\!\upharpoonright\mathsf{r}$$

We can hence recur to Lemma 6.10 in order to obtain that, for all $i \in I$,

$$\mathrm{G}_i \vdash \mathsf{p}[\,\mathrm{G}_i\!\upharpoonright\mathsf{p}\,] \parallel \mathsf{q}[\,\mathrm{G}_i\!\upharpoonright\mathsf{q}\,] \Big\|_{\mathsf{r}\in\mathrm{prt}(\mathrm{G}_i)\setminus\{\mathsf{p},\mathsf{q}\}} \mathsf{r}[\,\mathrm{G}\!\upharpoonright\mathsf{r}\,]$$

By Lemma 3.5

$$\mathrm{prt}(\mathrm{G}_i) = \mathrm{prt}(\mathsf{p}[\,\mathrm{G}_i\!\upharpoonright\mathsf{p}\,] \parallel \mathsf{q}[\,\mathrm{G}_i\!\upharpoonright\mathsf{q}\,] \Big\|_{\mathsf{r}\in\mathrm{prt}(\mathrm{G}_i)\setminus\{\mathsf{p},\mathsf{q}\}} \mathsf{r}[\,\mathrm{G}\!\upharpoonright\mathsf{r}\,])$$

which implies $\mathrm{prt}(\mathrm{G}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathrm{prt}(\Big\|_{\mathsf{r}\in\mathrm{prt}(\mathrm{G}_i)\setminus\{\mathsf{p},\mathsf{q}\}} \mathsf{r}[\,\mathrm{G}\!\upharpoonright\mathsf{r}\,])$ for all $i \in I$. It is now possible to use Rule [T-Comm] in order to get $\mathrm{G} \vdash \mathrm{G}\!\upharpoonright$. □

The proof of this theorem justifies our choice of taking (in all our examples) as processes the projections of the global types shown for the corresponding sessions, whenever these projections are defined.

Theorem 6.15. *Boundedness and inhabitation imply projectability.*

Proof. Let G be a bounded global type such that $\mathrm{G} \vdash \mathbb{M}$. To show that G is projectable, let consider $\mathsf{p} \in \mathrm{prt}(\mathrm{G})$ in order to get $\mathrm{G}\!\upharpoonright\mathsf{p}$ defined. We proceed by coinduction on the derivation of $\mathrm{G} \vdash \mathbb{M}$.

Axiom [T-End]. Immediate.

Rule [T-Comm]. In this case the last applied rule has the shape

$$\frac{\mathrm{G}_i \vdash \mathsf{r}[\,R_i\,] \parallel \mathsf{s}[\,S_i\,] \parallel \mathsf{p}[\,P\,] \parallel \mathbb{N} \quad \mathrm{prt}(\mathrm{G}_i) \setminus \{\mathsf{r}, \mathsf{s}\} = \mathrm{prt}(\mathsf{p}[\,P\,] \parallel \mathbb{N}) \quad \forall i \in I \subseteq J}{\mathrm{G} \vdash \mathsf{r}[\,\mathsf{s}!\{\lambda_i.R_i\}_{i\in I}\,] \parallel \mathsf{s}[\,\mathsf{r}?\{\lambda_j.S_j\}_{j\in J}\,] \parallel \mathsf{p}[\,P\,] \parallel \mathbb{N}}$$

where $\mathrm{G} = \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathrm{G}_i\}_{i\in I}$. By definition of boundedness we have that, for each $i \in I$, G_i is bounded. Moreover, by coinduction, $\mathrm{G}_i\!\upharpoonright\mathsf{p}$ is defined for each $i \in I$. If $\mathsf{p} = \mathsf{r}$, by definition of projection we have that $\mathrm{G}\!\upharpoonright\mathsf{p} = \mathsf{s}!\{\lambda_i.\mathrm{G}_i\!\upharpoonright\mathsf{p}\}_{i\in I}$. If $\mathsf{p} = \mathsf{s}$, by definition of projection we have that $\mathrm{G}\!\upharpoonright\mathsf{p} = \mathsf{r}?\{\lambda_i.\mathrm{G}_i\!\upharpoonright\mathsf{p}\}_{i\in I}$. So in both cases $\mathrm{G}\!\upharpoonright\mathsf{p}$ is defined. If $\mathsf{p} \notin \{\mathsf{r}, \mathsf{s}\}$, by Lemma 6.13, $P \leqslant \mathrm{G}_i\!\upharpoonright\mathsf{p}$ for each $i \in I$. We can hence recur to Lemma 6.11 to conclude that $\bigsqcap_{i\in I} \mathrm{G}_i\!\upharpoonright\mathsf{p}$ is defined. The thesis now follows by definition of projection, since in the present case $\mathrm{G}\!\upharpoonright\mathsf{p} = \bigsqcap_{i\in I} \mathrm{G}_i\!\upharpoonright\mathsf{p}$. □

The global type $\mathrm{G}_{\mathrm{bsc}}$ defined in Example 3.4 types the session $\mathbb{M}_{\mathrm{bsc}}$ defined in Example 2.5. It is unbounded, since $depth(\mathrm{G}_{\mathrm{bsc}}, \mathsf{c}) = \infty$. By previous theorem $\mathrm{G}_{\mathrm{bsc}}$ is un-projectable, and, since the projections on b and s are defined (see Example 6.2), the projection on c is undefined.

We observe that the finite global type G' of Example 6.3 is bounded, but it is un-projectable, as shown in that example. Hence, by the previous theorem, it is not inhabited. The global type G in the same example cannot be inhabited, since Rule [T-Comm] requires that the participants different from p, q of the two branches are the same. So, by Theorem 6.14, G is un-projectable. Another reason for the non projectability of G is given in Example 6.3. Moreover, by Theorem 6.15 G is not bounded (in fact participant r occurs only in one of the two paths from the root to the leaves).

7 FURTHER EXAMPLES

In this section we show how two relevant examples from [33] can be rephrased with our approach into typable sessions. This is not possible in general since the process language we are working with lacks channels, so that we cannot represent interleaved sessions and delegation. Also, there is a basic mismatch among the two systems, since the one in [33] gets rid of global types, which are kept only for the sake of comparison with the "classical" MPST and its extensions. The authors of [33] recover the desired properties of typable processes by parameterising to *safety properties* of contexts of local types that can be seen as systems of local specifications. These properties include Lock-freedom. A crucial property is *consistency*, which in our framework means that each pair of session participants have dual input/output communications.

Nonetheless, we can still formulate in our system at least two relevant examples from [33] by proceeding as follows. First, we observe that, when there is a unique session (which is the case for all the examples in Figure 4 of [33]), local types are akin to our processes. We just lose the typing of messages, which is irrelevant since in a unique session only expressions of ground types can be exchanged. Then, we reconstruct the global type of the so-obtained session (in our terms).

Service, client and authentication protocol. In the example in the Introduction of [33], further detailed in Figure 4(1) of the same paper, the service s sends to the client c either a request to `login` or `cancel`. If `login` is issued, then c sends a password `pwd` to the authorisation server a and then s receives from a the authorisation `auth`. In case c receives `cancel` from s the interaction with a is

$$\dfrac{\dfrac{\dfrac{\mathsf{End} \vdash \mathsf{s}[\,\mathbf{0}\,] \parallel \mathsf{c}[\,\mathbf{0}\,] \parallel \mathsf{a}[\,\mathbf{0}\,] \quad \mathsf{prt}(\mathsf{End}) \setminus \{\mathsf{a},\mathsf{s}\} = \emptyset = \mathsf{prt}(\mathsf{c}[\,\mathbf{0}\,])}{\mathsf{a} \to \mathsf{s} : \mathsf{auth} \vdash \mathsf{s}[\,\mathsf{a}?\mathsf{auth}\,] \parallel \mathsf{c}[\,\mathbf{0}\,] \parallel \mathsf{a}[\,\mathsf{s}!\mathsf{auth}\,]} \quad \mathsf{prt}(\mathsf{a} \to \mathsf{s} : \mathsf{auth}) \setminus \{\mathsf{c},\mathsf{a}\} = \{\mathsf{s}\} = \mathsf{prt}(\mathsf{s}[\,\mathsf{a}?\mathsf{auth}\,])}{\mathrm{G}'_{\mathsf{sca}} \vdash \mathsf{s}[\,\mathsf{a}?\mathsf{auth}\,] \parallel \mathsf{c}[\,\mathsf{a}!\mathsf{pwd}\,] \parallel \mathsf{a}[\,R_{\mathsf{sca}}\,]} \quad \dfrac{\dfrac{\mathsf{End} \vdash \mathsf{s}[\,\mathbf{0}\,] \parallel \mathsf{c}[\,\mathbf{0}\,] \parallel \mathsf{a}[\,\mathbf{0}\,] \quad \mathsf{prt}(\mathsf{End}) \setminus \{\mathsf{a},\mathsf{s}\} = \emptyset = \mathsf{prt}(\mathsf{c}[\,\mathbf{0}\,])}{\mathsf{c} \to \mathsf{a} : \mathtt{quit} \vdash \mathsf{s}[\,\mathbf{0}\,] \parallel \mathsf{c}[\,\mathsf{a}!\mathtt{quit}\,] \parallel \mathsf{a}[\,R_{\mathsf{sca}}\,]}}{\mathsf{prt}(\mathrm{G}'_{\mathsf{sca}}) \setminus \{\mathsf{s},\mathsf{c}\} = \mathsf{prt}(\mathsf{c} \to \mathsf{a} : \mathtt{quit}) \setminus \{\mathsf{s},\mathsf{c}\} = \{\mathsf{a}\} = \mathsf{prt}(\mathsf{a}[\,R_{\mathsf{sca}}\,])}}{\mathrm{G}_{\mathsf{sca}} \vdash \mathbb{M}_{\mathsf{sca}}}$$

Figure 1: A derivation of $\mathrm{G}_{\mathsf{sca}} \vdash \mathbb{M}_{\mathsf{sca}}$.

$$\dfrac{\dfrac{\mathcal{D}}{\mathsf{s} \to \mathsf{a} : \mathsf{pri}.\mathrm{G}'_{\mathsf{asb}} \vdash \mathsf{a}[\,\mathsf{s}?\mathsf{pri}.\,P'_{\mathsf{asb}}\,] \parallel \mathsf{s}[\,\mathsf{a}!\mathsf{pri}.\,\mathsf{a}?\{\mathsf{buy},\ \mathsf{no}\}\,] \parallel \mathsf{b}[\,R_{\mathsf{asb}}\,]}}{\mathrm{G}_{\mathsf{asb}} \vdash \mathbb{M}_{\mathsf{asb}}}$$

where $\mathcal{D}$ is the derivation

$$\dfrac{\dfrac{\dfrac{\mathsf{End} \vdash \mathsf{a}[\,\mathbf{0}\,] \parallel \mathsf{s}[\,\mathbf{0}\,] \parallel \mathsf{b}[\,\mathbf{0}\,]}{\mathrm{G}_{\mathsf{yes}} \vdash \mathsf{a}[\,\mathsf{s}!\mathsf{buy}\,] \parallel \mathsf{s}[\,\mathsf{a}?\{\mathsf{buy},\ \mathsf{no}\}\,] \parallel \mathsf{b}[\,\mathbf{0}\,]} \quad \mathcal{D}}{\mathsf{b} \to \mathsf{a} : \{\mathsf{yes}.\mathrm{G}_{\mathsf{yes}},\ \mathsf{no}.\mathrm{G}'_{\mathsf{asb}}\} \vdash \mathbb{M}'_{\mathsf{asb}}} \quad \dfrac{\mathsf{End} \vdash \mathsf{a}[\,\mathbf{0}\,] \parallel \mathsf{s}[\,\mathbf{0}\,] \parallel \mathsf{b}[\,\mathbf{0}\,]}{\mathsf{a} \to \mathsf{s} : \mathsf{no} \vdash \mathsf{a}[\,\mathsf{s}!\mathsf{no}\,] \parallel \mathsf{s}[\,\mathsf{a}?\{\mathsf{buy},\ \mathsf{no}\}\,] \parallel \mathsf{b}[\,\mathbf{0}\,]}}{\mathrm{G}'_{\mathsf{asb}} \vdash \mathsf{a}[\,P'_{\mathsf{asb}}\,] \parallel \mathsf{s}[\,\mathsf{a}?\{\mathsf{buy},\ \mathsf{no}\}\,] \parallel \mathsf{b}[\,R_{\mathsf{asb}}\,]}$$

and $\mathbb{M}'_{\mathsf{asb}} = \mathsf{a}[\,\mathsf{b}?\{\mathsf{yes}.\ \mathsf{s}!\mathsf{buy},\ \mathsf{no}.\ P'_{\mathsf{asb}}\}\,] \parallel \mathsf{s}[\,\mathsf{a}?\{\mathsf{buy},\ \mathsf{no}\}\,] \parallel \mathsf{b}[\,\mathsf{a}!\{\mathsf{yes},\ \mathsf{no}.\ R_{\mathsf{asb}}\}\,]$

Figure 2: A derivation of $\mathrm{G}_{\mathsf{asb}} \vdash \mathbb{M}_{\mathsf{asb}}$.

aborted by c sending `quit` to a. This session is represented by

$$\mathbb{M}_{\mathsf{sca}} = \mathsf{s}[\,P_{\mathsf{sca}}\,] \parallel \mathsf{c}[\,Q_{\mathsf{sca}}\,] \parallel \mathsf{a}[\,R_{\mathsf{sca}}\,]$$

where the processes are

$$\begin{array}{lcl} P_{\mathsf{sca}} & = & \mathsf{c}!\{\mathtt{login}.\mathsf{a}?\mathsf{auth}, \mathtt{cancel}\} \\ Q_{\mathsf{sca}} & = & \mathsf{s}?\{\mathtt{login}.\mathsf{a}!\mathsf{pwd},\ \mathtt{cancel}.\mathsf{a}!\mathtt{quit}\} \\ R_{\mathsf{sca}} & = & \mathsf{c}?\{\mathsf{pwd}.\mathsf{s}!\mathsf{auth},\ \mathtt{quit}\} \end{array}$$

Then we derive $\mathrm{G}_{\mathsf{sca}} \vdash \mathbb{M}_{\mathsf{sca}}$ where

$$\begin{array}{lcl} \mathrm{G}_{\mathsf{sca}} & = & \mathsf{s} \to \mathsf{c} : \{\mathtt{login}.\mathrm{G}'_{\mathsf{sca}},\ \mathtt{cancel}.\mathsf{c} \to \mathsf{a} : \mathtt{quit}\} \\ \mathrm{G}'_{\mathsf{sca}} & = & \mathsf{c} \to \mathsf{a} : \mathsf{pwd}.\ \mathsf{a} \to \mathsf{s} : \mathsf{auth} \end{array}$$

as shown in Figure 1.

We observe that $\mathrm{G}_{\mathsf{sca}}$ is exactly the global type derived for this session in [33]. In that paper it is shown that $\mathrm{G}_{\mathsf{sca}}$ is projectable, but the set of processes $\mathrm{G}_{\mathsf{sca}}\upharpoonright\mathsf{s} = P_{\mathsf{sca}}$, $\mathrm{G}_{\mathsf{sca}}\upharpoonright\mathsf{c} = Q_{\mathsf{sca}}$, $\mathrm{G}_{\mathsf{sca}}\upharpoonright\mathsf{a} = R_{\mathsf{sca}}$ is inconsistent since P_{sca} and R_{sca} are inconsistent. The problem is that the input/output behaviours between s and a depend on inputs/outputs of them with c. More precisely, s and a exchange the message `auth` only when c sends `login` to s. Hence this example is ruled out by the classical MPST system in [15, 26, 27]. In this case, it seems that our system surrogates this inconsistency by directly embodying in Rule [T-Comm] what guarantees that any implementation of the protocol $\mathbb{M}_{\mathsf{sca}}$ will be well behaved and lock free.

Recursive two-buyer protocol. This is the example in Figure 4(2) of [33]. After querying the store s for the `price` of some good, Alice, represented by role a, asks Bob, represented by b, to `split` the price. If Bob replies by issuing `yes`, then Alice sends `buy` to s; otherwise she insists by asking Bob for a different splitting (the fact that the subsequent proposals by Alice are actually different is not explicitly represented neither in the protocol below nor in the type of a in [33]). At any time, Alice might quit the protocol, sending `esc` to Bob and `no` to the store.

The protocol is encoded by the session

$$\mathbb{M}_{\mathsf{asb}} = \mathsf{a}[\,P_{\mathsf{asb}}\,] \parallel \mathsf{s}[\,Q_{\mathsf{asb}}\,] \parallel \mathsf{b}[\,R_{\mathsf{asb}}\,]$$

where the processes are

$$\begin{array}{l} P_{\mathsf{asb}} = \mathsf{s}!\mathsf{que}.\ \mathsf{s}?\mathsf{pri}.\ P'_{\mathsf{asb}} \\ P'_{\mathsf{asb}} = \mathsf{b}!\begin{cases} \mathtt{spl}.\ \mathsf{b}?\{\mathsf{yes}.\ \mathsf{s}!\mathsf{buy},\ \mathsf{no}.\ P'_{\mathsf{asb}}\} \\ \mathtt{esc}.\ \mathsf{s}!\mathsf{no} \end{cases} \\ Q_{\mathsf{asb}} = \mathsf{a}?\mathsf{que}.\ \mathsf{a}!\mathsf{pri}.\ \mathsf{a}?\{\mathsf{buy},\ \mathsf{no}\} \\ R_{\mathsf{asb}} = \mathsf{a}?\begin{cases} \mathtt{spl}.\ \mathsf{a}!\{\mathsf{yes},\ \mathsf{no}.\ R_{\mathsf{asb}}\} \\ \mathtt{esc} \end{cases} \end{array}$$

The session $\mathbb{M}_{\mathsf{asb}}$ can be typed by

$$\mathrm{G}_{\mathsf{asb}} = \mathsf{a} \to \mathsf{s} : \mathsf{que}.\mathsf{s} \to \mathsf{a} : \mathsf{pri}.\mathrm{G}'_{\mathsf{asb}}$$

where

$$\begin{array}{l} \mathrm{G}'_{\mathsf{asb}} = \mathsf{a} \to \mathsf{b} : \begin{cases} \mathtt{spl}.\mathsf{b} \to \mathsf{a} : \{\mathsf{yes}.\mathrm{G}_{\mathsf{yes}}, \mathsf{no}.\mathrm{G}'_{\mathsf{asb}}\}, \\ \mathtt{esc}.\mathsf{a} \to \mathsf{s} : \mathsf{no} \end{cases} \\ \mathrm{G}_{\mathsf{yes}} = \mathsf{a} \to \mathsf{s} : \mathsf{buy} \end{array}$$

as shown in Figure 2. In that figure we omit the conditions on participants, which can be easily verified.

This example is interesting for two reasons. In [33] it is argued that the typing context corresponding to $\mathbb{M}_{\mathsf{asb}}$ cannot be obtained by projecting any global type, no matter whether one uses plain or full merging (see Definition 3.3 in Figure 3 of [33]). Instead in our type system we can type $\mathbb{M}_{\mathsf{asb}}$ with the un-projectable global type $\mathrm{G}_{\mathsf{asb}}$. The other reason is that $\mathrm{G}_{\mathsf{asb}}$ is a good example of an unbounded global type which is inhabited.

8 RELATED WORKS AND CONCLUDING REMARKS

Local types have been initially conceived for describing binary protocols in the π-calculus [25, 34]. Later, local types have been extended to multiparty protocols [26, 27], and embedded into a range of functional, concurrent, and object-oriented programming languages [1].

When coming to multiparty sessions, central in this approach is the concept of global type, modelling a choreography coordinating several protocols of local end-points, which in classical MPST are recovered by projecting the global type to a context of (multiparty) local types. As reported in the Introduction, some weaknesses of the approach has led in [33] to a system which, although more general, abandons global types. To reply to subsequent criticisms, in [36] the notion of *association* of a global type with a typing context has been proposed. This association is proved to be sound and complete w.r.t. the respective LTS semantics. However, the LTS of global types is the inductive one (see Figure 5 of [36]), and the definition of association (*ibidem*, Definition 9) makes essential use of projections. Therefore, although the authentication protocol from [33] is associated with its global type and hence provably sound in the system, the recursive two-buyer protocol from [33], as well as the buyer, seller and carrier protocol of Example 2.5, cannot be associated with any (projectable and hence bounded) global type. With the present type system, instead, both the recursive two-buyer protocol and the buyer, seller and carrier protocol have global types, as shown in Section 7 and in Example 3.4.

With respect to [4, 6], the essential novelty of the present paper is the operational semantics of global types, in particular the coinductive rule for internal communication in Definition 4.2. As shown in Section 5, the adoption of the coinductive LTS (more precisely Rule [I-Comm] in Definition 4.3) makes our type system safe even when using unbounded global types, and more powerful, as shown by many examples.

A (slightly different) coinductive rule for internal communication appeared in [35, Definition 7] and in [22, Definition 3.18]. The typing rules in [35, Table 5] require projectability of global types, and the projection is defined recursively (*ibidem*, Definition 5), so that the coinductive semantics of global types does not allow to type more sessions than with the simpler inductive semantics. Moreover, in [22, Definition 3.3] global types are "balanced", namely *bounded* in our terminology. Then both systems in [35] and in [22] do not allow sessions in which the starvation of some participants is due to the choices of other participants. So these systems are less expressive than ours. For instance, the session of Example 2.5 (which is typed by the global type in Example 3.4) has no type in the systems of [22, 35]. Anyway, the goals of those papers are not to enlarge typability: in [22] the subtyping relation of [14] is shown to be sound and complete, whereas [35] is a simple introduction to MPST.

A more permissive coinductive rule for internal communication is considered in [11, Table 8] and projections are coinductively defined (*ibidem*, Table 7). In [11, Definition 4.9] there is a notion of "boundedness" of local types, which ensures that well-typed sessions terminate when the participants choose in a fair way. For instance, the carrier can receive `ship` and the session ends if the buyer fairly chooses between `item` and `buy` in the session of Example 2.5. A global type like G_1 in Example 6.6 (inhabited in our system), even if coinductively projectable, is not "bounded" in [11]. The boundedness condition of [11] is different from the one recalled in the present paper, since there the focus is on sessions in which the progress of one participant depends on the choices performed by other participants.

A criticism against adopting the SMPS approach [4] we have taken here is that the process syntax is poor, and hence insufficiently expressive. Indeed, as observed e.g. in [33], such a *non-classic* type system is single-session and first-order, namely without channel passing. To make this apparent we observe that our *processes* are essentially local types, akin to those used e.g. in [22]. Let us observe in passing that all the examples in Figure 4 of [33] are single-session, and the types of messages are ground, that is immaterial w.r.t. the issue of deriving local from global types. Now, while it is true that the original formulation of the typing systems for SMPS is subsumed by the system in [33], this is not anymore the case with the present one, which is a proper (and conservative) extension of the original system, as shown by Examples 2.5 and 3.4. Moreover, since we can derive a global type for sessions like the recursive two-buyer protocol, but we do not model delegation, our system is essentially incomparable with the ones in [33, 36].

The preorder on processes we define in Section 6 plays the same role as the subtyping relation on local types in other works. In the original standard subtyping of [19] a better type has more outputs and less inputs, while in the subtyping of [14] a better type has fewer outputs and more inputs. The subtyping of [19] allows channel substitution, while the subtyping of [14] allows process substitution, as observed in [18]. This justifies our structural preorder on processes, which is akin to a restriction of the subtyping of [14]. The advantage of this restriction is a strong version of Session Fidelity, see Theorem 5.4. On the other hand, as shown in [7], such a restriction does not change the class of sessions that can be typed by standard global types (but may change the types assigned to them). The proof in [7] can be easily adapted to the present type system.

From another perspective, our processes – equipped with their synchronous operational semantics – could be interpreted as a representation of synchronous versions of Communicating Finite State Machines (CFSM) [8]. The properties of Subject Reduction and Session Fidelity do correspond (in our synchronous setting) to implementability in [30]. In [29, 30], the projection of a global type on all of its participants produces a system of (asynchronous) CFSMs. Such projection has an automata-theoretic basis and works on a general set of global types allowing for multiple receiver choices. However, because of the automata-theoretic approach to projections in [29, 30] – as well as the asynchronous model of communication adopted there – the comparison with our work is not immediate and deserves further investigation, which we leave as future work.

More expressive global types are proposed in [10, 30], the main novelty being the possibility of having multiple senders and receivers. This is realised in [30] by generalising projections, while in [10] by using a richer syntax of global types. We plan to study a coinductive semantics for global types allowing multiple senders

and receivers for both synchronous and asynchronous communications.

ACKNOWLEDGMENTS

We are grateful to Luca Padovani for enlightening discussions on the coinductive semantics of global types, and to the referees for their careful reading and the useful suggestions to improve our paper.

The first author was partially supported by Project "National Center for HPC, Big Data e Quantum Computing", Programma M4C2, Investimento 1.3 – Next Generation EU. The third author has been partially supported by INdAM – GNCS 2024 project "Fondamenti di Informatica e Sistemi Informatici".

REFERENCES

[1] Davide Ancona, Viviana Bono, Mario Bravetti, Joana Campos, Giuseppe Castagna, Pierre-Malo Deniélou, Simon J. Gay, Nils Gesbert, Elena Giachino, Raymond Hu, Einar Broch Johnsen, Francisco Martins, Viviana Mascardi, Fabrizio Montesi, Rumyana Neykova, Nicholas Ng, Luca Padovani, Vasco T. Vasconcelos, and Nobuko Yoshida. 2016. Behavioral types in programming languages. *Foundations and Trends in Programming Languages* 3, 2-3 (2016), 95–230. https://doi.org/10.1561/2500000031

[2] Franco Barbanera and Mariangiola Dezani-Ciancaglini. 2023. Partially typed multiparty sessions. In *ICE (EPTCS, Vol. 383)*, Clément Aubert, Cinzia Di Giusto, Simon Fowler, and Larisa Safina (Eds.). Open Publishing Association, Waterloo, 15–34. https://doi.org/10.4204/EPTCS.383.2

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

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

[5] Franco Barbanera, Mariangiola Dezani-Ciancaglini, and Ugo de'Liguoro. 2023. Partial typing for asynchronous multiparty sessions. In *DCM*. Open Publishing Association, Waterloo, 1–19. invited paper, to appear, http://www.di.unito.it/~dezani/papers/bdl23.pdf.

[6] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Lorenzo Gheri, and Nobuko Yoshida. 2023. Multicompatibility for multiparty-session composition. In *PPDP*, Santiago Escobar and Vasco Vasconcelos (Eds.). ACM Press, New York, 2:1–2:15. https://doi.org/10.1145/3610612.3610614

[7] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Emilio Tuosto. 2021. Composition and decomposition of multiparty sessions. *Journal of Logic and Algebraic Methods in Programming* 119 (2021), 100620. https://doi.org/10.1016/j.jlamp.2020.100620

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

[9] Giuseppe Castagna, Nils Gesbert, and Luca Padovani. 2009. A theory of contracts for Web services. *ACM Transaction on Programming Languages and Systems* 31, 5 (2009), 19:1–19:61. https://doi.org/10.1145/1538917.1538920

[10] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. 2021. Global types and event structure semantics for asynchronous multiparty sessions. *CoRR* abs/2102.00865 (2021), 1–72. arXiv:2102.00865 https://arxiv.org/abs/2102.00865

[11] Luca Ciccone, Francesco Dagnino, and Luca Padovani. 2024. Fair termination of multiparty sessions. *Journal of Logical and Algebraic Methods in Programming* 139 (2024), 100964. https://doi.org/10.1016/j.jlamp.2024.100964

[12] Bruno Courcelle. 1983. Fundamental properties of infinite trees. *Theoretical Computer Science* 25 (1983), 95–169. https://doi.org/10.1016/0304-3975(83)90059-2

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

[14] Romain Demangeon and Kohei Honda. 2011. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR (LNCS, Vol. 6901)*, Joost-Pieter Katoen and Barbara König (Eds.). Springer, Berlin, 280–296. https://doi.org/10.1007/978-3-642-23217-6_19

[15] Pierre-Malo Deniélou and Nobuko Yoshida. 2011. Dynamic multirole session types. In *POPL*, Thomas Ball and Mooly Sagiv (Eds.). ACM Press, New York, 435–446.

[16] Pierre-Malo Deniélou, Nobuko Yoshida, Andi Bejleri, and Raymond Hu. 2012. Parameterised multiparty session types. *Logical Methods in Computer Science* 8, 4 (2012), 1–46. https://doi.org/10.2168/LMCS-8(4:6)2012

[17] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, and Nobuko Yoshida. 2015. Precise subtyping for synchronous multiparty sessions. In *PLACES (EPTCS, Vol. 203)*, Simon Gay and Jade Alglave (Eds.). Open Publishing Association, Waterloo, 29–43. https://doi.org/10.4204/EPTCS.203.3

[18] Simon Gay. 2016. Subtyping supports safe session substitution. In *A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday (LNCS, Vol. 9600)*, Sam Lindley, Conor McBride, Philip W. Trinder, and Donald Sannella (Eds.). Springer, Berlin, 95–108. https://doi.org/10.1007/978-3-319-30936-1_5

[19] Simon Gay and Malcolm Hole. 2005. Subtyping for session types in the pi-calculus. *Acta Informatica* 42, 2/3 (2005), 191–225. https://doi.org/10.1007/s00236-005-0177-z

[20] Simon J. Gay, Diogo Poças, and Vasco T. Vasconcelos. 2022. The different shades of infinite session types. In *FOSSACS (LNCS, Vol. 13242)*, Patricia Bouyer and Lutz Schröder (Eds.). Springer, Berlin, 347–367. https://doi.org/10.1007/978-3-030-99253-8_18

[21] Silvia Ghilezan, Svetlana Jaksic, Jovanka Pantovic, Alceste Scalas, and Nobuko Yoshida. 2019. Precise subtyping for synchronous multiparty sessions. *Journal of Logic and Algebraic Methods in Programamming* 104 (2019), 127–173. https://doi.org/10.1016/J.JLAMP.2018.12.002

[22] Silvia Ghilezan, Jovanka Pantovic, Ivan Prokic, Alceste Scalas, and Nobuko Yoshida. 2021. Precise subtyping for asynchronous multiparty sessions. *PACMPL* 5, POPL (2021), 1–28. https://doi.org/10.1145/3434297

[23] Rob van Glabbeek, Peter Höfner, and Ross Horne. 2021. Assuming just enough fairness to make session types complete for lock-freedom. In *LICS*, Leonid Libkin (Ed.). ACM Press, New York, 1–13. https://doi.org/10.1109/LICS52264.2021.9470531

[24] Kohei Honda. 1993. Types for Dyadic Interaction. In *CONCUR (LNCS, Vol. 715)*, Eike Best (Ed.). Springer, Berlin, 509–523.

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

[26] Kohei Honda, Nobuko Yoshida, and Marco Carbone. 2008. Multiparty asynchronous session types. In *POPL*, George C. Necula and Philip Wadler (Eds.). ACM Press, New York, 273–284. https://doi.org/10.1145/1328897.1328472

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

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

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

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

[31] Luca Padovani. 2014. Deadlock and lock freedom in the linear π-calculus. In *CSL-LICS*, Thomas A. Henzinger and Dale Miller (Eds.). ACM Press, New York, 72:1–72:10. https://doi.org/10.1007/978-3-662-43376-8_10

[32] Benjamin C. Pierce. 2002. *Types and Programming Languages*. MIT Press, Cambridge, Massachusetts. I–XXI, 1–623 pages.

[33] Alceste Scalas and Nobuko Yoshida. 2019. Less is more: multiparty session types revisited. *PACMPL* 3, POPL (2019), 30:1–30:29.

[34] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. 1994. An interaction-based language and its typing system. In *PARLE (LNCS, Vol. 817)*, Chris Hankin (Ed.). Springer, Berlin, 122–138. https://doi.org/10.1007/BFb0053567

[35] Nobuko Yoshida and Lorenzo Gheri. 2020. A very gentle introduction to multiparty session types. In *ICDCIT (LNCS, Vol. 11969)*, Dang Van Hung and Meenakshi D'Souza (Eds.). Springer, Berlin, 73–93. https://doi.org/10.1007/978-3-030-36987-3_5

[36] Nobuko Yoshida and Ping Hou. 2024. Less is more revisited: association with global multiparty session types. *CoRR* abs/2402.16741 (2024), 1–41. https://doi.org/10.48550/ARXIV.2402.16741