Event structure semantics for multiparty sessions

Ilaria Castellani[1], Mariangiola Dezani-Ciancaglini[2]⋆, and Paola Giannini[3]⋆⋆

[1] INRIA, Université Côte d'Azur, Sophia Antipolis, France
[2] Dipartimento di Informatica, Università di Torino, Italy
[3] DiSIT, Università del Piemonte Orientale, Alessandria, Italy

Abstract. We propose an interpretation of multiparty sessions as *flow event structures*, which allows concurrency between communications within a session to be explicitly represented. We show that this interpretation is equivalent, when the multiparty sessions can be described by global types, to an interpretation of global types as *prime event structures*.

1 Introduction

Session types were proposed in the mid-nineties [46, 32], as a tool for specifying and analysing web services and communication protocols. They were first introduced in a variant of the π-calculus to describe binary interactions between processes. Such binary interactions may often be viewed as a client-server protocol. Subsequently, session types were extended to *multiparty sessions* [33, 34], where several participants may interact with each other. A multiparty session is an interaction among peers, and there is no need to distinguish one of the participants as representing the server. All one needs is an abstract specification of the protocol that guides the interaction. This is called the *global type* of the session. The global type describes the behaviour of the whole session, as opposed to the local types that describe the behaviours of single participants. In a multiparty session, local types may be retrieved as projections from the global type.

Typical safety properties ensured by session types are *communication safety* (absence of communication errors), *session fidelity* (agreement with the protocol) and, in the absence of session interleaving, *progress* (no participant gets stuck).

Some simple examples of sessions not satisfying the above properties are: 1) a sender emitting a message while the receiver expects a different message (communication error); 2) two participants both waiting to receive a message from the other one (deadlock due to a protocol violation); 3) a three-party session where the first participant waits to receive a message from the second participant, which keeps interacting forever with the third participant (starvation, although the session is not deadlocked).

What makes session types particularly attractive is that they offer several advantages at once: 1) static safety guarantees, 2) automatic check of protocol

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

implementation correctness, based on local types, and 3) a strong connection with automata [27], graphical models [37] and logics [13, 47, 49].

In this paper we further investigate the relationship between multiparty session types and other concurrency models, by focussing on Event Structures [52]. We consider a standard multiparty session calculus where sessions are described as networks of sequential processes [27]. Each process implements a participant in the session. We propose an interpretation of such networks as *Flow Event Structures* (FESs) [8, 10] (a subclass of Winskel's Stable Event Structures [52]), which allows concurrency between session communications to be explicitly represented. We then introduce global types for these networks, and define an interpretation of them as *Prime Event Structures* (PESs) [50, 42]. Since the syntax of global types does not allow all the concurrency among communications to be expressed, the events of the associated PES need to be defined as equivalence classes of communication sequences up to *permutation equivalence*. We show that when a network is typable by a global type, the FES semantics of the former is equivalent, in a precise technical sense, to the PES semantics of the latter.

The paper is organised as follows. Section 2 introduces our multiparty session calculus. In Section 3 we recall the definitions of PESs and FESs, which will be used in Section 4 to interpret processes and networks, respectively. PESs are also used in Section 6 to interpret global types, which are defined in Section 5. In Section 7 we prove the equivalence between the FES semantics of a network and the PES semantics of its global type. Section 8 discusses related work in some detail and sketches directions for future work. Last but not least, we conclude by expressing our gratitude to Rocco.

For space reasons, all the proofs except that of the main theorem (Theorem 4) are omitted. The missing proofs may be found in the research report [15].

2 A Core Calculus for Multiparty Sessions

We now formally introduce our calculus, where multiparty sessions are represented as networks of processes. We assume the following base sets: *session participants*, ranged over by $\mathsf{p}, \mathsf{q}, \mathsf{r}$ and forming the set Part, and *messages*, ranged over by $\lambda, \lambda', \ldots$ and forming the set Msg.

Let $\pi \in \{\mathsf{p}?\lambda, \mathsf{p}!\lambda \mid \mathsf{p} \in \mathsf{Part}, \lambda \in \mathsf{Msg}\}$ denote an *atomic action*. The action $\mathsf{p}?\lambda$ represents an input of message λ from participant p, while the action $\mathsf{p}!\lambda$ represents an output of message λ to participant p.

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

$$P ::= \Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i \quad | \quad \bigoplus_{i\in I} \mathsf{p}!\lambda_i; P_i \quad | \quad \mu X.P \quad | \quad X \quad | \quad \mathbf{0}$$

External choice ($\sum$) and internal choice ($\bigoplus$) are assumed to be associative, commutative, and non-empty. When I is a singleton, $\Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$ will be rendered as $\mathsf{p}?\lambda; P$ and $\bigoplus_{i\in I} \mathsf{p}!\lambda_i; P_i$ will be rendered as $\mathsf{p}!\lambda; P$.

A process prefixed by an atomic action is either an *input process* or an *output process*. Note that in an external choice all summands are input processes receiving

$$\mathsf{p}[\![\bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i]\!] \parallel \mathsf{q}[\![\sum_{j\in J} \mathsf{p}?\lambda_j; Q_j]\!] \parallel \mathbb{N} \xrightarrow{\mathsf{p}\lambda_h\mathsf{q}} \mathsf{p}[\![P_h]\!] \parallel \mathsf{q}[\![Q_h]\!] \parallel \mathbb{N} \quad h \in I \cap J \qquad [\textsc{Com}]$$

Fig. 1: LTS for networks.

from the same sender p, and in an internal choice all summands are output processes sending to the same receiver p. Trailing $\mathbf{0}$ processes will be omitted.

Recursion is required to be guarded and processes are treated equi-recursively, i.e. they are identified with their generated tree [45] (Chapter 21).

In a full-fledged calculus, messages would carry values, namely they would be of the form $\lambda(v)$. For simplicity, we consider only pure messages here. This will allow us to project global types directly to processes, without having to explicitly introduce local types, see Section 5.

Networks are comprised of at least two pairs of the form $\mathsf{p}[\![P]\!]$ composed in parallel, each with a different participant p.

Definition 2 (Networks). Networks *are defined by:*

$$\mathbb{N} = \mathsf{p}_1[\![P_1]\!] \parallel \cdots \parallel \mathsf{p}_n[\![P_n]\!] \qquad n \geq 2,\ \mathsf{p}_i \neq \mathsf{p}_j \ \textit{for any}\ i, j$$

We assume the standard structural congruence on networks, stating that parallel composition is associative and commutative and has neutral element $\mathsf{p}[\![\mathbf{0}]\!]$ for any fresh p. To express the operational semantics of networks, we use an LTS whose labels record the message exchanged during a communication together with its sender and receiver. The set of *atomic communications*, ranged over by α, α', is defined to be $\{\mathsf{p}\lambda\mathsf{q} \mid \mathsf{p}, \mathsf{q} \in \mathsf{Part}, \lambda \in \mathsf{Msg}\}$, where $\mathsf{p}\lambda\mathsf{q}$ represents the emission of a message λ from participant p to participant q. We write $\mathsf{part}(\mathsf{p}\lambda\mathsf{q}) = \{\mathsf{p}, \mathsf{q}\}$.

The LTS semantics of networks is specified by the unique rule [Com] given in Figure 1. Notice that rule [Com] is symmetric with respect to external and internal choices. In a well-typed network (see Section 5) it will always be the case that $I \subseteq J$, assuring that participant p can freely choose an output, since participant q offers all corresponding inputs. As usual, we write $\mathbb{N} \xrightarrow{\alpha_1 \cdots \alpha_n} \mathbb{N}'$ as short for $\mathbb{N} \xrightarrow{\alpha_1} \mathbb{N}_1 \cdots \mathbb{N}_{n-1} \xrightarrow{\alpha_n} \mathbb{N}'$.

3 Event Structures

We recall now the definitions of *Prime Event Structure* (PES) from [42] and *Flow Event Structure* (FES) from [8]. The class of FESs is more general than that of PESs: for a precise comparison of various classes of event structures, we refer the reader to [9]. As we shall see in Section 4, while PESs are sufficient to interpret processes, the generality of FESs is needed to interpret networks.

Definition 3 (Prime Event Structure). *A prime event structure* (PES) *is a tuple* $S = (E, \leq, \#)$ *where:*

1. *E is a denumerable set of events;*
2. *$\leq \subseteq (E \times E)$ is a partial order relation, called the* causality *relation;*

3. $\# \subseteq (E \times E)$ *is an irreflexive symmetric relation, called the* conflict *relation, satisfying the property:* $\forall e, e', e'' \in E : e \# e' \leq e'' \Rightarrow e \# e''$ *(*conflict hereditariness*).*

We say that two events are *concurrent* if they are neither causally related nor in conflict.

Definition 4 (Flow Event Structure). *A flow event structure* (FES) *is a tuple* $S = (E, \prec, \#)$ *where:*

1. *E is a denumerable set of events;*
2. $\prec \subseteq (E \times E)$ *is an irreflexive relation, called the* flow *relation;*
3. $\# \subseteq (E \times E)$ *is a symmetric relation, called the* conflict *relation.*

Note that the flow relation is not required to be transitive, nor acyclic (its reflexive and transitive closure is just a preorder, not necessarily a partial order). Intuitively, the flow relation represents a possible *direct causality* between two events. Observe also that in a FES the conflict relation is not required to be irreflexive nor hereditary; indeed, FESs may exhibit self-conflicting events, as well as disjunctive causality (an event may have conflicting causes).

Any PES $S = (E, \leq, \#)$ may be regarded as a FES, with $\prec$ given by $<$ (the strict ordering) or by the covering relation of $\leq$.

We now recall the definition of *configuration* for event structures. Intuitively, a configuration is a set of events having occurred at some stage of the computation. Thus, the semantics of an event structure S is given by its poset of configurations ordered by set inclusion, where $\mathcal{X}_1 \subset \mathcal{X}_2$ means that S may evolve from $\mathcal{X}_1$ to $\mathcal{X}_2$.

Definition 5 (PES Configuration). *Let* $S = (E, \leq, \#)$ *be a prime event structure. A configuration of S is a finite subset $\mathcal{X}$ of E such that:*

1. *$\mathcal{X}$ is left-closed:* $e' \leq e \in \mathcal{X} \;\Rightarrow\; e' \in \mathcal{X}$;
2. *$\mathcal{X}$ is conflict-free:* $\forall e, e' \in \mathcal{X}, \neg(e \# e')$.

The definition of configuration for FESs is slightly more elaborated. For a subset $\mathcal{X}$ of E, let $\prec_{\mathcal{X}}$ be the restriction of the flow relation to $\mathcal{X}$ and $\prec^*_{\mathcal{X}}$ be its transitive and reflexive closure.

Definition 6 (FES Configuration). *Let* $S = (E, \prec, \#)$ *be a flow event structure. A configuration of S is a finite subset $\mathcal{X}$ of E such that:*

1. *$\mathcal{X}$ is left-closed up to conflicts:* $e' \prec e \in \mathcal{X},\ e' \notin \mathcal{X} \;\Rightarrow\; \exists e'' \in \mathcal{X}.\ e' \# e'' \prec e$;
2. *$\mathcal{X}$ is conflict-free:* $\forall e, e' \in \mathcal{X}, \neg(e \# e')$;
3. *$\mathcal{X}$ has no causality cycles: the relation $\prec^*_{\mathcal{X}}$ is a partial order.*

Condition (2) is the same as for prime event structures. Condition (1) is adapted to account for the more general – non-hereditary – conflict relation. It states that any event appears in a configuration with a "complete set of causes". Condition (3) ensures that any event in a configuration is actually reachable at some stage of the computation.

If S is a prime or flow event structure, we denote by $\mathcal{C}(S)$ its set of finite configurations. Then, the *domain of configurations* of S is defined as follows:

Definition 7 (ES Configuration Domain). *Let S be a prime or flow event structure with set of configurations $\mathcal{C}(S)$. The* domain of configurations *of S is the partially ordered set $\mathcal{D}(S) =_{\mathsf{def}} (\mathcal{C}(S), \subseteq)$.*

We recall from [9] a useful characterisation for configurations of FESs, which is based on the notion of proving sequence, defined as follows:

Definition 8 (Proving Sequences). *Given a flow event structure $S = (E, \prec, \#)$, a* proving sequence *in S is a sequence $e_1; \cdots ; e_n$ of distinct non-conflicting events (i.e. $i \neq j \;\Rightarrow\; e_i \neq e_j$ and $\neg(e_i \# e_j)$ for all i, j) satisfying:*

$$\forall i \leq n \, \forall e \in E : \quad e \prec e_i \;\Rightarrow\; \exists j < i \,.\; \textit{either } e = e_j \textit{ or } e \# e_j \prec e_i$$

Note that any prefix of a proving sequence is itself a proving sequence.

We have the following characterisation of configurations of FESs in terms of proving sequences.

Proposition 1 (Representation of configurations as proving sequences [9]). *Given a flow event structure $S = (E, \prec, \#)$, a subset X of E is a configuration of S if and only if it can be enumerated as a proving sequence $e_1; \cdots ; e_n$.*

Since PESs may be viewed as particular FESs, we may use Definition 8 and Proposition 1 both for the FESs associated with networks (see Section 4) and for the PESs associated with global types (see Section 6). Note that for a PES the condition of Definition 8 simplifies to

$$\forall i \leq n \, \forall e \in E : \quad e \prec e_i \;\Rightarrow\; \exists j < i \,.\; e = e_j$$

4 Event Structure Semantics of Processes and Networks

We interpret both processes and networks as event structures. The event structures associated with processes will be PESs. On the other hand, the event structures associated with networks will be FESs that are not necessarily prime.

Process events, ranged over by η, η', are actions $\pi, \pi' \in \{\mathsf{p}?\lambda, \mathsf{p}!\lambda \mid \mathsf{p} \in \mathsf{Part}, \lambda \in \mathsf{Msg}\}$ preceded by their *causal history*, which is a sequence of past actions.

Definition 9 (Process event). Process events *η, η' are defined by:*

$$\eta \quad ::= \pi \;\mid\; \pi \cdot \eta$$

Let ζ denote a (possibly empty) sequence of actions, and $\sqsubseteq$ denote the prefix ordering on such sequences. Each process event η may be written either in the form $\eta = \zeta \cdot \pi$ or in the form $\eta = \pi \cdot \zeta$. We shall feel free to use any of these forms.

We define the action of a process event as follows:

$$\mathsf{act}(\zeta \cdot \pi) = \pi$$

Definition 10 (Event Structure of a Process). *The* event structure of process *P is the triple*

$$\mathcal{S}^{\mathcal{P}}(P) = (\mathcal{PE}(P), \leq, \#)$$

where:

1. *$\mathcal{PE}(P)$ is defined by induction on the structure of P as follows:*

   (a) $\mathcal{PE}(\Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i) = \bigcup_{i\in I}\{\mathsf{p}?\lambda_i\} \cup \bigcup_{i\in I}\{\mathsf{p}?\lambda_i \cdot \eta_i \mid \eta_i \in \mathcal{PE}(P_i)\}$;
   (b) $\mathcal{PE}(\bigoplus_{i\in I}\mathsf{p}!\lambda_i; P_i) = \bigcup_{i\in I}\{\mathsf{p}!\lambda_i\} \cup \bigcup_{i\in I}\{\mathsf{p}!\lambda_i \cdot \eta_i \mid \eta_i \in \mathcal{PE}(P_i)\}$;
   (c) $\mathcal{PE}(\mathbf{0}) = \emptyset$;
   (d) $\mathcal{PE}(\mu X.P) = \mathcal{PE}(P\{\mu X.P/X\})$;
2. *the* $\leq$ *relation on the set of events* $\mathcal{PE}(P)$ *is given by:*
   (a) $\zeta \sqsubseteq \zeta' \;\Rightarrow\; \pi\cdot\zeta \leq \pi\cdot\zeta'$;
3. *the* # *relation on the set of events* $\mathcal{PE}(P)$ *is given by:*
   (a) $\pi \neq \pi' \Rightarrow \pi\cdot\zeta \# \pi'\cdot\zeta'$;
   (b) $\eta \# \eta' \Rightarrow \pi\cdot\eta \# \pi\cdot\eta'$.

Note that, due to Clause 1d of the previous definition, the set $\mathcal{PE}(P)$ is denumerable.

Example 1. If $P = \mu X.\mathsf{q}!\lambda; X \oplus \mathsf{q}!\lambda'$, then

$$\mathcal{PE}(P) = \{\underbrace{\mathsf{q}!\lambda\cdot\ldots\cdot\mathsf{q}!\lambda}_{n} \mid n \geq 1\} \cup \{\underbrace{\mathsf{q}!\lambda\cdot\ldots\cdot\mathsf{q}!\lambda}_{n}\cdot\mathsf{q}!\lambda' \mid n \geq 0\}$$

Proposition 2. *Let P be a process. Then $\mathcal{S}^{\mathcal{P}}(P)$ is a prime event structure with an empty concurrency relation.*

The definition of network events requires some preliminary notions. We start by defining the projections of process events on participants, which yield sequences of *undirected actions* of the form $?\lambda$ and $!\lambda$, or the empty sequence ϵ. Let ϑ range over $?\lambda$ and $!\lambda$, and let Θ range over non empty sequences of ϑ's.

Definition 11 (Projection of process events).

$$\mathsf{q}?\lambda \upharpoonright \mathsf{p} = \begin{cases} ?\lambda & \text{if } \mathsf{p} = \mathsf{q}, \\ \epsilon & \text{otherwise.} \end{cases} \qquad \mathsf{q}!\lambda \upharpoonright \mathsf{p} = \begin{cases} !\lambda & \text{if } \mathsf{p} = \mathsf{q}, \\ \epsilon & \text{otherwise.} \end{cases}$$

$$\pi.\eta \upharpoonright \mathsf{p} = \begin{cases} \eta \upharpoonright \mathsf{p} & \text{if } \pi \upharpoonright \mathsf{p} = \epsilon, \\ \pi \upharpoonright \mathsf{p} \,.\, \eta \upharpoonright \mathsf{p} & \text{otherwise.} \end{cases}$$

Sequences of undirected actions are related by a standard notion of duality.

Definition 12 (Duality of projections of process events).

$$?\lambda \bowtie !\lambda \qquad \vartheta \bowtie \vartheta' \text{ and } \Theta \bowtie \Theta' \Rightarrow \vartheta.\Theta \bowtie \vartheta'.\Theta'$$

Network events are essentially pairs of matching process events. To formalise the matching condition, we need to specify the locations of process events, namely the participants to which they belong.

Definition 13 (Located event). *We call* located event *a process event η pertaining to a participant* p*, written* $\mathsf{p} :: \eta$.

The duality between projections of process events induces a duality between located events.

Definition 14 (Duality of located events). *Two located events* $\mathsf{p} :: \eta, \mathsf{q} :: \eta'$ *are* dual*, written* $\mathsf{p} :: \eta \bowtie \mathsf{q} :: \eta'$*, if* $\eta \upharpoonright \mathsf{q} \bowtie \eta' \upharpoonright \mathsf{p}$ *and either* $\mathsf{act}(\eta) = \mathsf{q}?\lambda$ *and* $\mathsf{act}(\eta') = \mathsf{p}!\lambda$ *or* $\mathsf{act}(\eta) = \mathsf{q}!\lambda$ *and* $\mathsf{act}(\eta') = \mathsf{p}?\lambda$*.*

Dual located events may be sequences of actions of different length. For instance $\mathsf{p} :: \mathsf{q}!\lambda \cdot \mathsf{r}!\lambda' \bowtie \mathsf{r} :: \mathsf{p}?\lambda'$ and $\mathsf{p} :: \mathsf{q}!\lambda \bowtie \mathsf{q} :: \mathsf{r}!\lambda' \cdot \mathsf{p}?\lambda$.

Definition 15 (Network event). Network events ν, ν' *are unordered pairs of dual located events, namely:*

$$\nu ::= \{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\} \qquad \textit{where} \quad \mathsf{p} :: \eta \bowtie \mathsf{q} :: \eta'$$

We can now define the event structure associated with a network.

Definition 16 (Event Structure of a Network). *The* event structure of network $\mathbb{N} = \mathsf{p}_1[\![P_1]\!] \parallel \cdots \parallel \mathsf{p}_n[\![P_n]\!]$ *is the triple*

$$\mathcal{S}^{\mathcal{N}}(\mathbb{N}) = (\mathcal{NE}(\mathbb{N}), \prec, \#)$$

where:

1. $\mathcal{NE}(\mathbb{N}) = \bigcup_{1 \le i \ne j \le n} \{\{\mathsf{p}_i :: \eta_i, \mathsf{p}_j :: \eta_j\} \mid \eta_i \in \mathcal{PE}(P_i), \eta_j \in \mathcal{PE}(P_j), \mathsf{p}_i :: \eta_i \bowtie \mathsf{p}_j :: \eta_j\}$
2. *the* $\prec$ *relation on the set of events* $\mathcal{NE}(\mathbb{N})$ *is given by:*
   $\eta < \eta'$ & $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu' \Rightarrow \nu \prec \nu'$*;*
3. *the* # *relation on the set of events* $\mathcal{NE}(\mathbb{N})$ *is given by:*
   $\eta \# \eta'$ & $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu' \Rightarrow \nu \# \nu'$*.*

We define $\mathsf{comm}(\nu) = \mathsf{p}\lambda\mathsf{q}$ *if* $\nu = \{\mathsf{p} :: \zeta \cdot \mathsf{q}!\lambda, \mathsf{q} :: \zeta' \cdot \mathsf{p}?\lambda\}$ *and we say that the network event* ν represents *the atomic communication* $\mathsf{p}\lambda\mathsf{q}$*.*
Two events ν *and* ν' *are* concurrent *if* $\mathsf{part}(\mathsf{comm}(\nu)) \cap \mathsf{part}(\mathsf{comm}(\nu')) = \emptyset$*.*

The set of network events can be infinite as in the following example.

Example 2. Let P be as in Example 1, $Q = \mu Y.\mathsf{p}?\lambda; Y \oplus \mathsf{p}?\lambda'$ and $\mathbb{N} = \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!]$. Then

$$\mathcal{NE}(\mathbb{N}) = \{\{\mathsf{p} :: \underbrace{\mathsf{q}!\lambda \cdot \ldots \cdot \mathsf{q}!\lambda}_{n}, \mathsf{q} :: \underbrace{\mathsf{p}?\lambda \cdot \ldots \cdot \mathsf{p}?\lambda}_{n}\} \mid n \ge 1\} \quad \cup$$
$$\{\{\mathsf{p} :: \underbrace{\mathsf{q}!\lambda \cdot \ldots \cdot \mathsf{q}!\lambda}_{n} \cdot \mathsf{q}!\lambda', \mathsf{q} :: \underbrace{\mathsf{p}?\lambda \cdot \ldots \cdot \mathsf{p}?\lambda}_{n} \cdot \mathsf{p}?\lambda'\} \mid n \ge 0\}$$

Notably, concurrent events may also be related by the transitive closure of the flow relation, as shown in Example 3.

Proposition 3. *Let* $\mathbb{N}$ *be a network. Then* $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ *is a flow event structure with an irreflexive conflict relation.*

The following example shows how communications inherit the flow relation from the causality relation of their components.

Example 3. Let $\mathbb{N}$ be the network

$$\mathsf{p}[\![\mathsf{q}!\lambda_1]\!] \parallel \mathsf{q}[\![\mathsf{p}?\lambda_1; \mathsf{r}!\lambda_2]\!] \parallel \mathsf{r}[\![\mathsf{q}?\lambda_2; \mathsf{s}!\lambda_3]\!] \parallel \mathsf{s}[\![\mathsf{r}?\lambda_3]\!]$$

Then $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ has three network events

$$\nu_1 = \{\mathsf{p} :: \mathsf{q}!\lambda_1, \mathsf{q} :: \mathsf{p}?\lambda_1\} \qquad \nu_2 = \{\mathsf{q} :: \mathsf{p}?\lambda_1; \mathsf{r}!\lambda_2, \mathsf{r} :: \mathsf{q}?\lambda_2\}$$
$$\nu_3 = \{\mathsf{r} :: \mathsf{q}?\lambda_2; \mathsf{s}!\lambda_3, \mathsf{s} :: \mathsf{r}?\lambda_3\}$$

The flow relation obtained by Definition 16 is: $\nu_1 \prec \nu_2$ and $\nu_2 \prec \nu_3$. Note that each time the flow relation is inherited from the causality within a different participant, q in the first case and r in the second case. By the same definition the events ν_1 and ν_3 are concurrent. However, since $\nu_1 \prec^* \nu_3$, the events ν_1 and ν_3 cannot occur in any order. Indeed, the nonempty configurations are $\{\nu_1\}, \{\nu_1, \nu_2\}$ and $\{\nu_1, \nu_2, \nu_3\}$. Note that $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ has only one proving sequence per configuration (which is that given by the numbering of events in the configuration).

If $\mathbb{N}$ is a binary network, then its flow event structure may be turned into a prime event structure simply by replacing $\prec$ by $\prec^*$:

Theorem 1. *Let* $\mathbb{N} = \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \mathsf{p}_2[\![\, P_2 \,]\!]$ *and* $\mathcal{S}^{\mathcal{N}}(\mathbb{N}) = (\mathcal{NE}(\mathbb{N}), \prec, \#)$. *Then the structure* $\mathcal{S}^{\mathcal{N}}_*(\mathbb{N}) =_{\mathsf{def}} (\mathcal{NE}(\mathbb{N}), \prec^*, \#)$ *is a prime event structure.*

If $\mathbb{N}$ has more than two participants, then the duality requirement on its events is not sufficient to ensure the absence of circular dependencies[4]. For instance, in the following ternary network (which may be viewed as representing the 3-philosopher deadlock) the relation $\prec^*$ is not a partial order.

Example 4. Let $\mathbb{N}$ be the network
$$\mathsf{p}[\![\, \mathsf{r}?\lambda; \mathsf{q}!\lambda' \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda'; \mathsf{r}!\lambda'' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{q}?\lambda''; \mathsf{p}!\lambda \,]\!].$$
Then $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ has three network events
$$\nu_1 = \{\mathsf{p} :: \mathsf{r}?\lambda, \mathsf{r} :: \mathsf{q}?\lambda''; \mathsf{p}!\lambda\} \qquad \nu_2 = \{\mathsf{p} :: \mathsf{r}?\lambda; \mathsf{q}!\lambda', \mathsf{q} :: \mathsf{p}?\lambda'\}$$
$$\nu_3 = \{\mathsf{q} :: \mathsf{p}?\lambda'; \mathsf{r}!\lambda'', \mathsf{r} :: \mathsf{q}?\lambda''\}$$
By Definition 16(2) we have $\nu_1 \prec \nu_2 \prec \nu_3$ and $\nu_3 \prec \nu_1$. The only configuration of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ is the empty configuration, because the only set of events that satisfies left-closure is $X = \{\nu_1, \nu_2, \nu_3\}$, but this is not a configuration because $\prec^*_X$ is not a partial order (recall that $\prec_X$ is the restriction of $\prec$ to X) and hence the condition (3) of Definition 6 is not satisfied.

The next example illustrates Proposition 3 and shows that a network event may have both conflicting and concurrent causes.

Example 5. Let $\mathbb{N}$ be the network
$$\mathsf{p}[\![\, \mathsf{q}!\lambda; \mathsf{r}!\lambda_1 \oplus \mathsf{q}!\lambda'; \mathsf{r}!\lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda_2 + \mathsf{p}?\lambda'; \mathsf{s}!\lambda_2 \,]\!] \parallel$$
$$\mathsf{r}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda_3 \,]\!] \parallel \mathsf{s}[\![\, \mathsf{q}?\lambda_2; \mathsf{r}?\lambda_3 \,]\!]$$
Then $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ has seven network events:
$$\nu_1 = \{\mathsf{p} :: \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda\} \qquad \nu'_1 = \{\mathsf{p} :: \mathsf{q}!\lambda', \mathsf{q} :: \mathsf{p}?\lambda'\}$$
$$\nu_2 = \{\mathsf{p} :: \mathsf{q}!\lambda; \mathsf{r}!\lambda_1, \mathsf{r} :: \mathsf{p}?\lambda_1\} \qquad \nu'_2 = \{\mathsf{p} :: \mathsf{q}!\lambda'; \mathsf{r}!\lambda_1, \mathsf{r} :: \mathsf{p}?\lambda_1\}$$
$$\nu_3 = \{\mathsf{q} :: \mathsf{p}?\lambda; \mathsf{s}!\lambda_2, \mathsf{s} :: \mathsf{q}?\lambda_2\} \qquad \nu'_3 = \{\mathsf{q} :: \mathsf{p}?\lambda'; \mathsf{s}!\lambda_2, \mathsf{s} :: \mathsf{q}?\lambda_2\}$$
$$\nu_4 = \{\mathsf{r} :: \mathsf{p}?\lambda_1; \mathsf{s}!\lambda_3, \mathsf{s} :: \mathsf{q}?\lambda_2; \mathsf{r}?\lambda_3\}$$
We have $\nu_1 \prec \nu_i$ for $i = 2, 3$ and $\nu_j \prec \nu_4$ for $j = 2, 3$. Similarly, we have $\nu'_1 \prec \nu'_i$ for $i = 2, 3$ and $\nu'_j \prec \nu_4$ for $j = 2, 3$. The events ν_2 and ν'_2 share $\mathsf{r} :: \mathsf{p}?\lambda_1$, the

[4] This is a well-known issue in multiparty session types, which motivated the introduction of global types in [33], see Section 6.

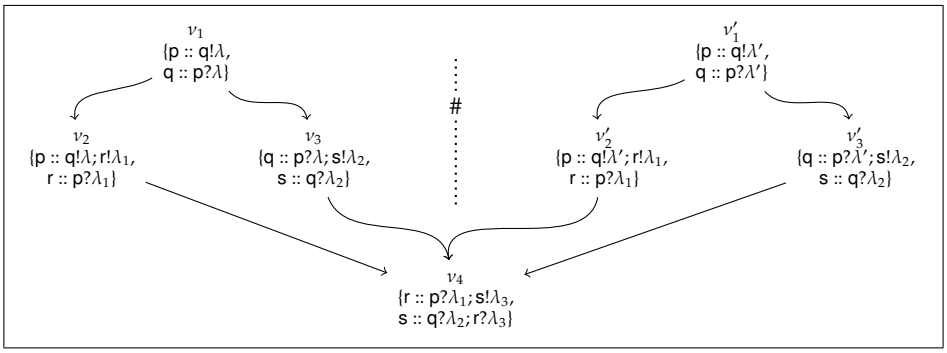

Fig. 2: Flow relation between events of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ in Example 5.

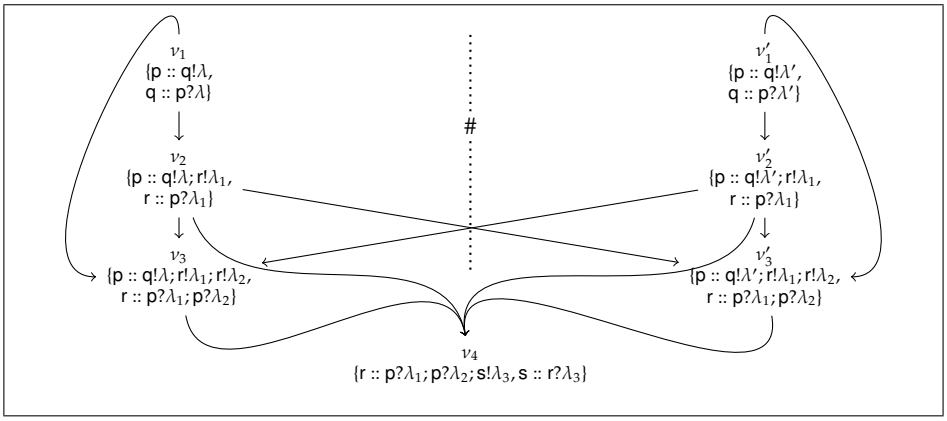

Fig. 3: Flow relation between events of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ in Example 6.

events ν_3 and ν'_3 share $\mathsf{s} :: \mathsf{q}?\lambda_2$. Moreover $\nu_i \# \nu'_j$ for each $i, j = 1, 2, 3$, whereas ν_2 and ν_3 are concurrent, and so are ν'_2 and ν'_3. The event ν_4 has two conflicting sets of causes $\{\nu_1, \nu_2, \nu_3\}$ and $\{\nu'_1, \nu'_2, \nu'_3\}$, and the nonempty configurations are $\{\nu_1\}, \{\nu_1, \nu_2\}, \{\nu_1, \nu_3\}$, $\{\nu_1, \nu_2, \nu_3\}$ and $\{\nu_1, \nu_2, \nu_3, \nu_4\}$, as well as $\{\nu'_1\}, \{\nu'_1, \nu'_2\}, \{\nu'_1, \nu'_3\}$, $\{\nu'_1, \nu'_2, \nu'_3\}$ and $\{\nu'_1, \nu'_2, \nu'_3, \nu_4\}$. Let $\mathcal{X} = \{\nu_1, \nu_2, \nu_3, \nu_4\}$ and $\mathcal{X}' = \{\nu'_1, \nu'_2, \nu'_3, \nu_4\}$. Note that the event ν_4 has two concurrent causes in both $\mathcal{X}$ and $\mathcal{X}'$. The proving sequences are:

$$\begin{array}{lllllll} \nu_1, & \nu_1;\nu_2, & \nu_1;\nu_3, & \nu_1;\nu_2;\nu_3, & \nu_1;\nu_3;\nu_2, & \nu_1;\nu_2;\nu_3;\nu_4, & \nu_1;\nu_3;\nu_2;\nu_4 \\ \nu'_1, & \nu'_1;\nu'_2, & \nu'_1;\nu'_3, & \nu'_1;\nu'_2;\nu'_3, & \nu'_1;\nu'_3;\nu'_2, & \nu'_1;\nu'_2;\nu'_3;\nu_4, & \nu'_1;\nu'_3;\nu'_2;\nu_4 \end{array}$$

Note that there are two proving sequences corresponding to the configuration $\mathcal{X}$ (and similarly for $\mathcal{X}'$ and each of the configurations $\{\nu_1, \nu_2, \nu_3\}$ and $\{\nu'_1, \nu'_2, \nu'_3\}$).

A graphical representation of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ is given in Figure 2, where the arrows represent the flow relation $\lessdot$ and the vertical dotted line for $\#$ indicates that all the events on the left of the line are in conflict with all the events on the right.

The next example shows that the relations of flow and conflict on network events are not necessarily disjoint.

Example 6. Let $\mathbb{N}$ be the network
$\mathsf{p}[\![\, \mathsf{q}!\lambda; \mathsf{r}!\lambda_1; \mathsf{r}!\lambda_2 \oplus \mathsf{q}!\lambda'; \mathsf{r}!\lambda_1; \mathsf{r}!\lambda_2 \,]\!] \,\|\, \mathsf{q}[\![\, \mathsf{p}?\lambda + \mathsf{p}?\lambda' \,]\!] \,\|\, \mathsf{r}[\![\, \mathsf{p}?\lambda_1; \mathsf{p}?\lambda_2; \mathsf{s}!\lambda_3 \,]\!] \,\|\, \mathsf{s}[\![\, \mathsf{r}?\lambda_3 \,]\!]$.
Then $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ has seven network events:

$$\nu_1 = \{\mathsf{p} :: \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda\} \qquad \nu_1' = \{\mathsf{p} :: \mathsf{q}!\lambda', \mathsf{q} :: \mathsf{p}?\lambda'\}$$
$$\nu_2 = \{\mathsf{p} :: \mathsf{q}!\lambda; \mathsf{r}!\lambda_1, \mathsf{r} :: \mathsf{p}?\lambda_1\} \qquad \nu_2' = \{\mathsf{p} :: \mathsf{q}!\lambda'; \mathsf{r}!\lambda_1, \mathsf{r} :: \mathsf{p}?\lambda_1\}$$
$$\nu_3 = \{\mathsf{p} :: \mathsf{q}!\lambda; \mathsf{r}!\lambda_1; \mathsf{r}!\lambda_2, \mathsf{r} :: \mathsf{p}?\lambda_1; \mathsf{p}?\lambda_2\} \qquad \nu_3' = \{\mathsf{p} :: \mathsf{q}!\lambda'; \mathsf{r}!\lambda_1; \mathsf{r}!\lambda_2, \mathsf{r} :: \mathsf{p}?\lambda_1; \mathsf{p}?\lambda_2\}$$
$$\nu_4 = \{\mathsf{r} :: \mathsf{p}?\lambda_1; \mathsf{p}?\lambda_2; \mathsf{s}!\lambda_3, \mathsf{s} :: \mathsf{r}?\lambda_3\}$$

We have $\nu_1 \prec \nu_i$ for $i = 2, 3$ and $\nu_j \prec \nu_4$ for $j = 2, 3$. Similarly, we have $\nu_1' \prec \nu_i'$ for $i = 2, 3$ and $\nu_j' \prec \nu_4$ for $j = 2, 3$. Moreover $\nu_i \# \nu_j'$ for each $i, j = 1, 2, 3$. Finally, we have $\nu_2 \prec \nu_3$ and $\nu_2' \prec \nu_3'$, and also the cross flows $\nu_2 \prec \nu_3'$ and $\nu_2' \prec \nu_3$. Since we have also $\nu_2 \# \nu_3'$ and $\nu_2' \# \nu_3$, this shows that the two relations $\prec$ and $\#$ are not disjoint. The nonempty configurations are $\{\nu_1\}, \{\nu_1, \nu_2\}, \{\nu_1, \nu_2, \nu_3\}$ and $\{\nu_1, \nu_2, \nu_3, \nu_4\}$, as well as $\{\nu_1'\}, \{\nu_1', \nu_2'\}, \{\nu_1', \nu_2', \nu_3'\}$ and $\{\nu_1', \nu_2', \nu_3', \nu_4\}$. The proving sequences are:

$$\nu_1, \quad \nu_1; \nu_2, \quad \nu_1; \nu_2; \nu_3, \quad \nu_1; \nu_2; \nu_3; \nu_4$$
$$\nu_1', \quad \nu_1'; \nu_2', \quad \nu_1'; \nu_2'; \nu_3', \quad \nu_1'; \nu_2'; \nu_3'; \nu_4$$

A graphical representation of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$ is given in Figure 3, where we use the same conventions as for Example 5.

5 Global Types

Global types are built from choices among atomic communications.

Definition 17 (Global types). Global types G *are defined by:*

$$\mathsf{G} \;::=\; \mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i \;\mid \mathsf{G} \,\|\, \mathsf{G} \;\mid\; \mu \mathbf{t}.\mathsf{G} \;\mid\; \mathbf{t} \;\mid\; \mathsf{End}$$

where $\lambda_j \neq \lambda_h$ for all $j, h \in I$, $j \neq h$, i.e. messages in choices are all different.

Sequential composition ($;$) has higher precedence than choice ($\boxplus$). Recursion must be guarded by atomic communications and it is treated equi-recursively. While there is no syntactic restriction on parallel composition of global types, our definition of projection will enforce that the component types have disjoint sets of participants. When I is a singleton, a choice $\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i$ will be rendered simply as $\mathsf{p} \xrightarrow{\lambda} \mathsf{q}; \mathsf{G}$. In writing global types, we omit the final End.

Participants of global types are defined inductively as follows:

$$\mathsf{part}(\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i) = \{\mathsf{p}, \mathsf{q}\} \cup \bigcup_{i \in I} \mathsf{part}(\mathsf{G}_i)$$
$$\mathsf{part}(\mu \mathbf{t}.\mathsf{G}) = \mathsf{part}(\mathsf{G}) \qquad \mathsf{part}(\mathbf{t}) = \mathsf{part}(\mathsf{End}) = \emptyset$$

The projection of a global type onto participants is given in Figure 4. As usual, projection is defined only when it is defined on all participants. Because of the simplicity of our calculus, the projection of a global type, when defined, is simply a process. The projection of a choice type on the sender produces an output process sending one of its possible messages to the receiver and then acting according to the projection of the corresponding branch. Similarly for the projection on the receiver, which produces an input process. Projection of a choice type on the other participants is defined only if it produces the same process

for all the branches of the choice. This is a standard condition for multiparty session types. The projection of a parallel global type $\mathsf{G}_1 \parallel \mathsf{G}_2$ on a participant p is undefined if p appears in both G_1 and G_2. Otherwise there are two possibilities: 1) if p appears in G_i but not in G_j, for $i \neq j$, then $(\mathsf{G}_1 \parallel \mathsf{G}_2) \upharpoonright \mathsf{p}$ yields the projection of G_i on p; 2) if p appears in neither G_1 nor G_2, then $(\mathsf{G}_1 \parallel \mathsf{G}_2) \upharpoonright \mathsf{p}$ yields $\mathbf{0}$.

From now on we will only consider projectable global types.

The definition of well-typed network is given in Figure 5. We first define a preorder on processes, $P \leq P'$, saying when a process P *can be used where we expect process* P'. In particular, $P \leq P'$, if either P is equal to P' or they are both input processes receiving messages from the same participant, P may receive more messages than P' and after receiving the same message the process P continues with a process that can be used when we expect the corresponding one in P'. The double line indicates that the rule is interpreted coinductively [45] (Chapter 21). A network is well typed with global type G, if all its participants have associated processes that behave as specified by the projections of a global type. In Rule [Net], the condition $\mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}$ ensures that all participants of the global type appear in the network. Moreover it permits additional participants that do not appear in the global type, allowing the typing of sessions containing $\mathsf{p}[\![\, \mathbf{0} \,]\!]$ for a fresh p — a property required to guarantee invariance of types under structural congruence of networks.

Example 7. The networks of Examples 2, 3, 5 and 6 can be typed respectively by

$$\begin{aligned}
\mathsf{G} &= \mu \mathbf{t}.\mathsf{p} \to \mathsf{q} : (\lambda; \mathbf{t} \boxplus \lambda') \\
\mathsf{G}' &= \mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; \mathsf{q} \xrightarrow{\lambda_2} \mathsf{r}; \mathsf{r} \xrightarrow{\lambda_3} \mathsf{s} \\
\mathsf{G}'' &= \mathsf{p} \to \mathsf{q} : (\lambda; \mathsf{p} \xrightarrow{\lambda_1} \mathsf{r}; \mathsf{q} \xrightarrow{\lambda_2} \mathsf{s}; \mathsf{r} \xrightarrow{\lambda_3} \mathsf{s} \boxplus \lambda'; \mathsf{p} \xrightarrow{\lambda_1} \mathsf{r}; \mathsf{q} \xrightarrow{\lambda_2} \mathsf{s}; \mathsf{r} \xrightarrow{\lambda_3} \mathsf{s}) \\
\mathsf{G}''' &= \mathsf{p} \to \mathsf{q} : (\lambda; \mathsf{p} \xrightarrow{\lambda_1} \mathsf{r}; \mathsf{p} \xrightarrow{\lambda_2} \mathsf{r}; \mathsf{r} \xrightarrow{\lambda_3} \mathsf{s} \boxplus \lambda'; \mathsf{p} \xrightarrow{\lambda_1} \mathsf{r}; \mathsf{p} \xrightarrow{\lambda_2} \mathsf{r}; \mathsf{r} \xrightarrow{\lambda_3} \mathsf{s})
\end{aligned}$$

The network of Example 4 instead cannot be typed.

To formalise the classical properties of Subject Reduction and Session Fidelity [33, 34], we use the standard LTS for global types given in Figure 6. Rule [Icomm] is justified by the fact that in a projectable global type $\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i$, the

$$(\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i) \upharpoonright \mathsf{r} = \begin{cases} \Sigma_{i \in I} \mathsf{p}?\lambda_i; \mathsf{G}_i \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{q}, \\ \bigoplus_{i \in I} \mathsf{q}!\lambda_i; \mathsf{G}_i \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{p}, \\ \mathsf{G}_i \upharpoonright \mathsf{r} & \text{if } \mathsf{G}_i \upharpoonright \mathsf{r} = \mathsf{G}_j \upharpoonright \mathsf{r} \text{ for all } i, j \in I \end{cases}$$

$$(\mathsf{G}_1 \parallel \mathsf{G}_2) \upharpoonright \mathsf{p} = \mathsf{G}_i \upharpoonright \mathsf{p} \text{ if } \mathsf{p} \notin \mathsf{part}(\mathsf{G}_j) \text{ for } \{i, j\} = \{1, 2\}$$

$$(\mu \mathbf{t}.\mathsf{G}) \upharpoonright \mathsf{p} = \begin{cases} \mu X_{\mathbf{t}}.\mathsf{G} \upharpoonright \mathsf{p} & \text{if } \mathsf{p} \in \mathsf{part}(\mathsf{G}) \\ \mathbf{0} & \text{otherwise} \end{cases} \qquad \mathbf{t} \upharpoonright \mathsf{p} = X_{\mathbf{t}} \qquad \mathsf{End} \upharpoonright \mathsf{p} = \mathbf{0}$$

Fig. 4: Projection of global types onto participants.

$$\mathbf{0} \leq \mathbf{0}\ [\text{S-}\mathbf{0}] \qquad \frac{P_i \leq Q_i \quad i \in I}{\sum_{i\in I\cup J} \mathsf{p}?\lambda_i; P_i \leq \sum_{i\in I} \mathsf{p}?\lambda_i; Q_i}\ [\text{S-In}] \qquad \frac{P_i \leq Q_i \quad i \in I}{\bigoplus_{i\in I} \mathsf{p}!\lambda_i; P_i \leq \bigoplus_{i\in I} \mathsf{p}!\lambda_i; Q_i}\ [\text{S-out}]$$

$$\frac{P_i \leq \mathsf{G} \upharpoonright \mathsf{p}_i \quad i \in I \quad \mathsf{part}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \prod_{i\in I} \mathsf{p}_i[\![\, P_i \,]\!] : \mathsf{G}}\ [\text{Net}]$$

Fig. 5: Preorder on processes and network typing rule.

$$\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I} \lambda_i; \mathsf{G}_i \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j \quad j \in I\ [\text{Ecomm}] \qquad \frac{\mathsf{G}_1 \xrightarrow{\alpha} \mathsf{G}_1'}{\mathsf{G}_1 \parallel \mathsf{G}_2 \xrightarrow{\alpha} \mathsf{G}_1' \parallel \mathsf{G}_2}\ [\text{Pcomm}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\alpha} \mathsf{G}_i' \quad i \in I \quad \mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset}{\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I} \lambda_i; \mathsf{G}_i \xrightarrow{\alpha} \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I} \lambda_i; \mathsf{G}_i'}\ [\text{Icomm}]$$

Fig. 6: LTS for global types.

behaviours of the participants different from p and q are the same in all branches, and hence they are independent from the choice and may be executed before it.

Theorem 2 (Subject Reduction). *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathbb{N} \xrightarrow{\alpha} \mathbb{N}'$, *then* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$ *and* $\vdash \mathbb{N}' : \mathsf{G}'$.

Theorem 3 (Session Fidelity). *If* $\vdash \mathbb{N} : \mathsf{G}$ *and* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$, *then* $\mathbb{N} \xrightarrow{\alpha} \mathbb{N}'$ *and* $\vdash \mathbb{N}' : \mathsf{G}'$.

6 Event Structure Semantics of Global Types

We define now the event structure associated with a global type. The events of this PES will be equivalence classes of particular sequences of communications.

Let σ denote a finite (and possibly empty) sequence of atomic communications, and *Seq* denote the set of these sequences.

Definition 18 (Permutation equivalence). *The permutation equivalence on Seq is the least equivalence* $\sim$ *such that*

$$\sigma \cdot \alpha_1 \cdot \alpha_2 \cdot \sigma' \ \sim\ \sigma \cdot \alpha_2 \cdot \alpha_1 \cdot \sigma' \quad \textit{if} \quad \mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$$

We denote by $[\sigma]_\sim$ *the equivalence class of the sequence* σ, *and by* $Seq/\sim$ *the set of equivalence classes on Seq. Note that* $[\epsilon]_\sim = \{\epsilon\} \in Seq/\sim$, *and* $[\alpha]_\sim = \{\alpha\} \in Seq/\sim$ *for any* α. *Moreover* $|\sigma'| = |\sigma|$ *for all* $\sigma' \in [\sigma]_\sim$, *where* $|\cdot|$ *yields the length of the sequence.*

The events associated with a global type, called *global events* and denoted by γ, γ', are equivalence classes of particular communication sequences that we call *pointed*. Intuitively, all communications in a pointed sequence are causes of some subsequent communication. Formally:

Definition 19 (Pointed communication sequence). *A communication sequence* $\sigma = \alpha_1 \cdots \alpha_n$, $n > 0$, *is said to be* pointed *if*

$$\textit{for all } i,\ 1 \leq i < n,\ \mathsf{part}(\alpha_i) \cap \bigcup_{i+1 \leq j \leq n} \mathsf{part}(\alpha_j) \neq \emptyset$$

Note that the condition of Definition 19 must be satisfied only by the α_i with $i < n$, thus it is vacuously satisfied by any communication sequence of length 1.

Example 8. Let $\alpha_1 = \mathsf{p}\lambda_1\mathsf{q}$, $\alpha_2 = \mathsf{r}\lambda_2\mathsf{s}$ and $\alpha_3 = \mathsf{r}\lambda_3\mathsf{p}$. Then $\sigma_1 = \alpha_1$ and $\sigma_3 = \alpha_1 \cdot \alpha_2 \cdot \alpha_3$ are pointed sequences, while $\sigma_2 = \alpha_1 \cdot \alpha_2$ is *not* a pointed sequence.

Definition 20 (Global event). *Let* $\sigma = \sigma' \cdot \alpha$ *be a pointed communication sequence. Then* $\gamma = [\sigma]_\sim$ *is a* global event *with communication* α*, notation* $\mathsf{comm}(\gamma) = \alpha$.

Notice that $\mathsf{comm}(\cdot)$ is well defined due to the following proposition, where $\mathsf{last}(\sigma)$ denotes the last communication of σ.

Proposition 4. *Let* σ *be pointed communication sequence. If* $\sigma \sim \sigma'$*, then* σ' *is a pointed communication sequence and* $\mathsf{last}(\sigma) = \mathsf{last}(\sigma')$.

In order to interpret global types as ESs, we define a form of prefixing of a global event by a communication, in such a way that the result is again a global event.

Definition 21 (Causal prefixing of a global event by communications). *The causal prefixing of a global event by a nonempty sequence of communications is defined as follows:*

1. *The causal prefixing of a global event by a communication is defined by*
$$\mathsf{p}\lambda\mathsf{q} \circ \gamma = \begin{cases} [\mathsf{p}\lambda\mathsf{q} \cdot \sigma]_\sim & \textit{if } \gamma = [\sigma]_\sim \textit{ and } \mathsf{p}\lambda\mathsf{q} \cdot \sigma \textit{ is a pointed sequence} \\ \gamma & \textit{otherwise} \end{cases}$$
2. *The mapping* $\circ$ *naturally extends to communication sequences*
$$(\alpha \cdot \sigma) \circ \gamma = \alpha \circ (\sigma \circ \gamma) \qquad \sigma \neq \epsilon$$

Definition 22 (Event Structure of a Global Type). *The* event structure of global type G *is the triple*

$$\mathcal{S}^{\mathcal{G}}(\mathsf{G}) = (\mathcal{GE}(\mathsf{G}), \leq, \#)$$

where:

1. $\mathcal{GE}(\mathsf{G})$ *is defined by induction on the structure of* G *as follows:*
   (a) $\mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i) = \bigcup_{i \in I} \{\{\mathsf{p}\lambda_i\mathsf{q}\}\} \cup \bigcup_{i \in I} \{\mathsf{p}\lambda_i\mathsf{q} \circ \gamma_i \mid \gamma_i \in \mathcal{GE}(\mathsf{G}_i)\}$;
   (b) $\mathcal{GE}(\mathsf{G}_1 \parallel \mathsf{G}_2) = \mathcal{GE}(\mathsf{G}_1) \cup \mathcal{GE}(\mathsf{G}_2)$;
   (c) $\mathcal{GE}(\mathsf{End}) = \mathcal{GE}(\mathbf{t}) = \emptyset$;
   (d) $\mathcal{GE}(\mu\mathbf{t}.\mathsf{G}) = \mathcal{GE}(\mathsf{G}\{\mu\mathbf{t}.\mathsf{G}/\mathbf{t}\})$;
2. *the* $\leq$ *relation on the set of events* $\mathcal{GE}(\mathsf{G})$ *is given by:*
   $[\sigma]_\sim \leq [\sigma']_\sim$ *if* $\sigma \cdot \sigma'' \sim \sigma'$ *for some* σ'';
3. *the* $\#$ *relation on the set of events* $\mathcal{GE}(\mathsf{G})$ *is given by:*
   $[\sigma]_\sim \# [\sigma']_\sim$ *if* $\sigma \sim \sigma_1 \cdot \mathsf{p}\lambda\mathsf{q} \cdot \sigma_2$ *and* $\sigma' \sim \sigma_1 \cdot \mathsf{p}\lambda'\mathsf{q} \cdot \sigma_2'$ *for some* $\sigma_1, \sigma_2, \sigma_2', \mathsf{p}, \mathsf{q}, \lambda, \lambda'$ *such that* $\lambda \neq \lambda'$.

Note that, due to Clause 1d of Definition 22, the set $\mathcal{GE}(\mathsf{G})$ is denumerable.

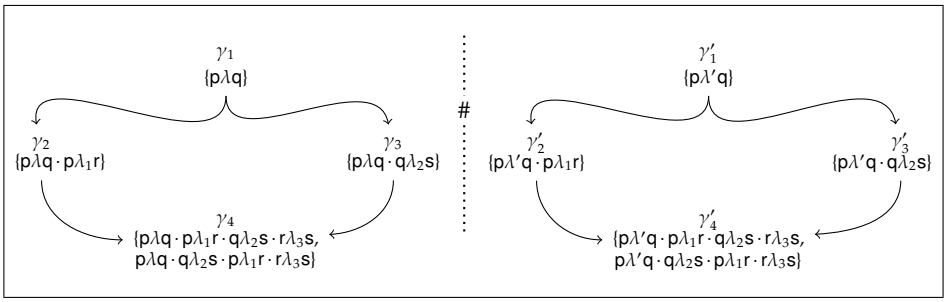

Fig. 7: Relation between events of $\mathcal{S}^{\mathcal{G}}(\mathsf{G}'')$ in Example 9.

Example 9. Let $\mathsf{G}_1 = \mathsf{p} \xrightarrow{\lambda_1} \mathsf{q};\mathsf{r} \xrightarrow{\lambda_2} \mathsf{s};\mathsf{r} \xrightarrow{\lambda_3} \mathsf{p}$ and $\mathsf{G}_2 = \mathsf{r} \xrightarrow{\lambda_2} \mathsf{s};\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q};\mathsf{r} \xrightarrow{\lambda_3} \mathsf{p}$. Then $\mathcal{GE}(\mathsf{G}_1) = \mathcal{GE}(\mathsf{G}_2) = \{\gamma_1, \gamma_2, \gamma_3\}$ where

$$\gamma_1 = \{\mathsf{p}\lambda_1\mathsf{q}\} \qquad \gamma_2 = \{\mathsf{r}\lambda_2\mathsf{s}\} \qquad \gamma_3 = \{\mathsf{p}\lambda_1\mathsf{q} \cdot \mathsf{r}\lambda_2\mathsf{s} \cdot \mathsf{r}\lambda_3\mathsf{p}, \mathsf{r}\lambda_2\mathsf{s} \cdot \mathsf{p}\lambda_1\mathsf{q} \cdot \mathsf{r}\lambda_3\mathsf{p}\}$$

with $\gamma_1 \leq \gamma_3$ and $\gamma_2 \leq \gamma_3$. The configurations are $\{\gamma_1\}$, $\{\gamma_2\}$ and $\{\gamma_1, \gamma_2, \gamma_3\}$ and the proving sequences are

$$\gamma_1 \qquad \gamma_2 \qquad \gamma_1;\gamma_2 \qquad \gamma_2;\gamma_1 \qquad \gamma_1;\gamma_2;\gamma_3 \qquad \gamma_2;\gamma_1;\gamma_3$$

If G' is as in Example 7, then $\mathcal{GE}(\mathsf{G}') = \{\gamma_1, \gamma_2, \gamma_3\}$ where

$$\gamma_1 = \{\mathsf{p}\lambda_1\mathsf{q}\} \qquad \gamma_2 = \{\mathsf{p}\lambda_1\mathsf{q} \cdot \mathsf{q}\lambda_2\mathsf{r}\} \qquad \gamma_3 = \{\mathsf{p}\lambda_1\mathsf{q} \cdot \mathsf{q}\lambda_2\mathsf{r} \cdot \mathsf{r}\lambda_3\mathsf{s}\}$$

with $\gamma_1 \leq \gamma_2 \leq \gamma_3$. The configurations are $\{\gamma_1\}$, $\{\gamma_1, \gamma_2\}$ and $\{\gamma_1, \gamma_2, \gamma_3\}$. There is a proving sequence corresponding to each configuration. Notice that G' types the network of Example 3.

If G'' is as in Example 7, then $\mathcal{GE}(\mathsf{G}'') = \{\gamma_1, \gamma'_1, \gamma_2, \gamma'_2, \gamma_3, \gamma'_3, \gamma_4, \gamma'_4\}$ where

$$\gamma_1 = \{\mathsf{p}\lambda\mathsf{q}\} \qquad \gamma'_1 = \{\mathsf{p}\lambda'\mathsf{q}\} \qquad \gamma_2 = \{\mathsf{p}\lambda\mathsf{q} \cdot \mathsf{p}\lambda_1\mathsf{r}\} \qquad \gamma'_2 = \{\mathsf{p}\lambda'\mathsf{q} \cdot \mathsf{p}\lambda_1\mathsf{r}\}$$
$$\gamma_3 = \{\mathsf{p}\lambda\mathsf{q} \cdot \mathsf{q}\lambda_2\mathsf{s}\} \qquad \gamma'_3 = \{\mathsf{p}\lambda'\mathsf{q} \cdot \mathsf{q}\lambda_2\mathsf{s}\}$$
$$\gamma_4 = \{\mathsf{p}\lambda\mathsf{q} \cdot \mathsf{p}\lambda_1\mathsf{r} \cdot \mathsf{q}\lambda_2\mathsf{s} \cdot \mathsf{r}\lambda_3\mathsf{s}, \mathsf{p}\lambda\mathsf{q} \cdot \mathsf{q}\lambda_2\mathsf{s} \cdot \mathsf{p}\lambda_1\mathsf{r} \cdot \mathsf{r}\lambda_3\mathsf{s}\}$$
$$\gamma'_4 = \{\mathsf{p}\lambda'\mathsf{q} \cdot \mathsf{p}\lambda_1\mathsf{r} \cdot \mathsf{q}\lambda_2\mathsf{s} \cdot \mathsf{r}\lambda_3\mathsf{s}, \mathsf{p}\lambda'\mathsf{q} \cdot \mathsf{q}\lambda_2\mathsf{s} \cdot \mathsf{p}\lambda_1\mathsf{r} \cdot \mathsf{r}\lambda_3\mathsf{s}\}$$

with $\gamma_1 \leq \gamma_2 \leq \gamma_4$, $\gamma_1 \leq \gamma_3 \leq \gamma_4$ and $\gamma'_1 \leq \gamma'_2 \leq \gamma'_4$, $\gamma'_1 \leq \gamma'_3 \leq \gamma'_4$. The configurations are $\{\gamma_1\}$, $\{\gamma'_1\}$, $\{\gamma_1, \gamma_2\}$, $\{\gamma'_1, \gamma'_2\}$, $\{\gamma_1, \gamma_3\}$, $\{\gamma'_1, \gamma'_3\}$, $\{\gamma_1, \gamma_2, \gamma_3\}$, $\{\gamma'_1, \gamma'_2, \gamma'_3\}$, and $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$, $\{\gamma'_1, \gamma'_2, \gamma'_3, \gamma'_4\}$. The configurations with less than three elements correspond to only one proving sequence, while the others correspond to two proving sequences each. Notice that G'' types the network of Example 5. A graphical representation of $\mathcal{S}^{\mathcal{G}}(\mathsf{G}'')$ is given in Figure 7, where the arrows represent the covering relation of $\leq$. Note that the event structure is prime and so conflict is hereditary. Indeed, since the events maintain their complete history the events γ_4 and γ'_4 are in conflict.

Proposition 5. *Let* G *be a global type. Then* $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$ *is a prime event structure.*

Observe that while our interpretation of networks as FESs exactly reflects the concurrency expressed by the syntax of networks, our interpretation of global types as PESs exhibits more concurrency than that given by the syntax of global types. This is because the parallel composition of global types is only defined when its arguments have disjoint participants, and thus it cannot be

used to specify concurrency between two forking paths that may join again, e.g., two concurrent events that are both causes of a third event, as γ_1 and γ_2 in $\mathcal{GE}(\mathsf{G}_1) = \mathcal{GE}(\mathsf{G}_2)$ in the above Example 9.

7 Equivalence of the two Event Structure Semantics

We establish now our main result for typed networks, namely the isomorphism between the domain of configurations of the FES of the network and the domain of configurations of the PES of its global type. We start by stating the correspondence between the communication sequences of networks and the proving sequences of their event structures. To this end, we introduce some auxiliary definitions.

Definition 23 (Truncation of a communication sequence). *Let $\sigma = \alpha_1 \cdots \alpha_n$ be a communication sequence with $n > 0$. For each $i = 1, \ldots, n+1$, we define $\sigma\rfloor_i =_{\mathsf{def}} \alpha_1 \cdots \alpha_{i-1}$ to be the ith truncation of σ, where by convention $\alpha_1 \cdots \alpha_{i-1} = \epsilon$ if $i = 1$. Note that $\sigma\rfloor_{n+1} = \sigma$.*

Definition 24 (Projection). *The* projection of the communication sequence σ on participant p, *notation $\sigma \upharpoonright \mathsf{p}$, is the process event defined by:*

1. $(\mathsf{p}\lambda\mathsf{q} \cdot \sigma) \upharpoonright \mathsf{p} = \mathsf{q}!\lambda \cdot \sigma \upharpoonright \mathsf{p}$;
2. $(\mathsf{q}\lambda\mathsf{p} \cdot \sigma) \upharpoonright \mathsf{p} = \mathsf{q}?\lambda \cdot \sigma \upharpoonright \mathsf{p}$;
3. $(\mathsf{r}\lambda\mathsf{s} \cdot \sigma) \upharpoonright \mathsf{p} = \sigma \upharpoonright \mathsf{p}$ *if* $\mathsf{p} \neq \mathsf{r}, \mathsf{s}$;
4. $\epsilon \upharpoonright \mathsf{p} = \epsilon$.

It is easy to verify that if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$, then $(\alpha_1 \cdot \alpha_2) \upharpoonright \mathsf{p} = (\alpha_2 \cdot \alpha_1) \upharpoonright \mathsf{p}$ for all p. Therefore $\sigma \sim \sigma'$ implies $\sigma \upharpoonright \mathsf{p} = \sigma' \upharpoonright \mathsf{p}$.

Definition 25 (Network events from communications). *If $\sigma = \alpha_1 \cdots \alpha_n$ is a communication sequence with $\mathsf{part}(\alpha_i) = \{\mathsf{p}_i, \mathsf{q}_i\}$, we define the* sequence of network events corresponding to σ *by*

$$\mathsf{nec}(\sigma) = \nu_1; \cdots; \nu_n$$

where $\nu_i = \{\mathsf{p}_i :: \sigma\rfloor_{i+1} \upharpoonright \mathsf{p}_i, \mathsf{q}_i :: \sigma\rfloor_{i+1} \upharpoonright \mathsf{q}_i\}$ for $1 \leq i \leq n$.

It is immediate to see that, if $\sigma = \mathsf{p}\lambda\mathsf{q}$, then $\mathsf{nec}(\sigma)$ is the event $\{\mathsf{p} :: \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda\}$.

Lemma 1. *Let $\mathbb{N} \xrightarrow{\sigma} \mathbb{N}'$.*

1. *If $\{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\} \in \mathcal{NE}(\mathbb{N}')$, then $\{\mathsf{p} :: \sigma \upharpoonright \mathsf{r} \cdot \eta, \mathsf{q} :: \sigma \upharpoonright \mathsf{q} \cdot \eta'\} \in \mathcal{NE}(\mathbb{N})$;*
2. *$\mathsf{nec}(\sigma)$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$.*

Lemma 2. *If $\nu_1; \cdots; \nu_n$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$, then $\mathbb{N} \xrightarrow{\sigma} \mathbb{N}'$ where $\sigma = \mathsf{comm}(\nu_1) \cdots \mathsf{comm}(\nu_n)$.*

Similar relations hold between reductions of global types and their events.

Definition 26 (Global events from communications). *If $\sigma = \alpha_1 \cdots \alpha_n$ is a communication sequence, we define the* the sequence of global events corresponding to σ *by*

$$\mathsf{gec}(\sigma) = \gamma_1; \cdots; \gamma_n$$

where $\gamma_i = \sigma\rfloor_i \circ [\alpha_i]_\sim$ for $1 \leq i \leq n$.

Lemma 3. *Let* $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$.

1. *If* $\gamma \in \mathcal{GE}(\mathsf{G}')$, *then* $\sigma \circ \gamma \in \mathcal{GE}(\mathsf{G})$;
2. $\mathsf{gec}(\sigma)$ *is a proving sequence in* $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$.

Lemma 4. *If* $\gamma_1; \cdots ; \gamma_n$ *is a proving sequence in* $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$, *then* $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$ *and* $\sigma = \mathsf{comm}(\gamma_1) \cdots \mathsf{comm}(\gamma_n)$.

To prove our main theorem we will also use the following separation result from [9] (Lemma 2.8 p. 12):

Lemma 5 (Separation [9]). *Let* $S = (E, \prec, \#)$ *be a flow event structure and* $X, X' \in \mathcal{C}(S)$ *be such that* $X \subset X'$. *Then there exist* $e \in X' \backslash X$ *such that* $X \cup \{e\} \in \mathcal{C}(S)$.

We may now show the correspondence between the configurations of the FES of a network and the configurations of the PES of its global type.

Let $\simeq$ denote isomorphism on domains of configurations.

Theorem 4. *If* $\vdash \mathbb{N} : \mathsf{G}$, *then* $\mathcal{D}(\mathcal{S}^{\mathcal{N}}(\mathbb{N})) \simeq \mathcal{D}(\mathcal{S}^{\mathcal{G}}(\mathsf{G}))$.

Proof. By Lemma 2 if $\nu_1; \cdots ; \nu_n$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$, then $\mathbb{N} \xrightarrow{\sigma} \mathbb{N}'$ where $\sigma = \mathsf{comm}(\nu_1) \cdots \mathsf{comm}(\nu_n)$. By applying iteratively Subject Reduction (Theorem 2) $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$ and $\vdash \mathbb{N}' : \mathsf{G}'$. By Lemma 3(2) $\mathsf{gec}(\sigma)$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$.

By Lemma 4 if $\gamma_1; \cdots ; \gamma_n$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$, then $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$ where $\sigma = \mathsf{comm}(\gamma_1) \cdots \mathsf{comm}(\gamma_n)$. By applying iteratively Session Fidelity (Theorem 3) $\mathbb{N} \xrightarrow{\sigma} \mathbb{N}'$ and $\vdash \mathbb{N}' : \mathsf{G}'$. By Lemma 1(2) $\mathsf{nec}(\sigma)$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$.

Therefore we have a bijection between $\mathcal{D}(\mathcal{S}^{\mathcal{N}}(\mathbb{N}))$ and $\mathcal{D}(\mathcal{S}^{\mathcal{G}}(\mathsf{G}))$, given by $\mathsf{nec}(\sigma) \leftrightarrow \mathsf{gec}(\sigma)$ for any σ generated by the (bisimilar) LTSs of $\mathbb{N}$ and G.

We show now that this bijection preserves inclusion of configurations. By Lemma 5 it is enough to prove that if $\nu_1; \cdots ; \nu_n \in \mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathbb{N}))$ is mapped to $\gamma_1; \cdots ; \gamma_n \in \mathcal{C}(\mathcal{S}^{\mathcal{G}}(\mathsf{G}))$, then $\nu_1; \cdots ; \nu_n; \nu \in \mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathbb{N}))$ iff $\gamma_1; \cdots ; \gamma_n; \gamma \in \mathcal{C}(\mathcal{S}^{\mathcal{G}}(\mathsf{G}))$, where $\gamma_1; \cdots ; \gamma_n; \gamma$ is the image of $\nu_1; \cdots ; \nu_n; \nu$ under the bijection.

Suppose $\sigma = \mathsf{comm}(\nu_1) \cdots \mathsf{comm}(\nu_n) = \mathsf{comm}(\gamma_1) \cdots \mathsf{comm}(\gamma_n)$.

Let $\mathsf{comm}(\nu) = \alpha$. By Lemma 2, if $\nu_1; \cdots ; \nu_n; \nu$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$, then $\mathbb{N} \xrightarrow{\sigma} \mathbb{N}_0 \xrightarrow{\alpha} \mathbb{N}'$. Then we get $\nu = \{\mathsf{p} :: \sigma \cdot \alpha \twoheadrightarrow \mathsf{p}, \mathsf{q} :: \sigma \cdot \alpha \twoheadrightarrow \mathsf{q}\}$ by Lemma 1(1). By Definition 25 $\mathsf{nec}(\sigma \cdot \alpha) = \nu_1; \cdots ; \nu_n; \nu$. By applying iteratively Subject Reduction (Theorem 2) $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}_0 \xrightarrow{\alpha} \mathsf{G}'$ and $\vdash \mathbb{N}' : \mathsf{G}'$. By Definition 26 $\mathsf{gec}(\sigma \cdot \alpha) = \gamma_1; \cdots ; \gamma_n; \gamma$. By Lemma 3(2) $\mathsf{gec}(\sigma \cdot \alpha)$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$.

Let now $\mathsf{comm}(\gamma) = \alpha$. By Lemma 4, if $\gamma_1; \cdots ; \gamma_n; \gamma$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$, then $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}_0 \xrightarrow{\alpha} \mathsf{G}'$. By Lemma 3(1) we have $\gamma = [\sigma \circ \alpha]_{\sim}$. By Definition 26 $\mathsf{gec}(\sigma \cdot \alpha) = \gamma_1; \cdots ; \gamma_n; \gamma$. By applying iteratively Session Fidelity (Theorem 3) $\mathbb{N} \xrightarrow{\sigma} \mathbb{N}_0 \xrightarrow{\alpha} \mathbb{N}'$ and $\vdash \mathbb{N}' : \mathsf{G}'$. By Definition 25 $\mathsf{nec}(\sigma \cdot \alpha) = \nu_1; \cdots ; \nu_n; \nu$. By Lemma 1(2) $\mathsf{nec}(\sigma \cdot \alpha)$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathbb{N})$.

8 Related Work and Conclusions

Event Structures (ESs) were introduced in Winskel's PhD Thesis [50] and in the seminal paper by Nielsen, Plotkin and Winskel [42], roughly in the same frame of time as Milner's calculus CCS [40]. It is therefore not surprising that the relationship between these two approaches for modelling concurrent computations started to be investigated very soon afterwards. The first interpretation of CCS into ESs was proposed by Winskel in [51]. This interpretation made use of Stable ESs, because PESs, the simplest form of ESs, appeared not to be flexible enough to account for CCS parallel composition. Indeed, since CCS parallel composition allows for two concurrent complementary actions to either synchronise or occur independently in any order, each pair of such actions gives rise to two forking computations: this requires duplication of the same continuation process for each computation in PESs, while the continuation process may be shared by the forking computations in Stable ESs, which allow for disjunctive causality. Subsequently, ESs (as well as other nonsequential "denotational models" for concurrency such as Petri Nets) have been used as the touchstone for assessing noninterleaving operational semantics for CCS: for instance, the pomset semantics for CCS by Boudol and Castellani [7, 8] and the semantics based on "concurrent histories" proposed by Degano, De Nicola and Montanari [25, 23, 24], were both shown to agree with an interpretation of CCS processes into some class of ESs (PESs for [23, 24], PESs with non-hereditary conflict for [7] and FESs for [8]). Among the early interpretations of process calculi into ESs, we should also mention the PES semantics for TCSP (Theoretical CSP [11, 43]), proposed by Goltz and Loogen [39] and generalised by Baier and Majster-Cederbaum [2], and the Bundle ES semantics for LOTOS, proposed by Langerak [38] and extended by Katoen [36]. Like FESs, Bundle ESs are a subclass of Stable ESs. We recall the relationships between the above classes of ESs (the reader is referred to [10] for separating examples):

$$\textit{Prime ESs} \subset \textit{Bundle ESs} \subset \textit{Flow ESs} \subset \textit{Stable ESs} \subset \textit{General ESs}$$

More sophisticated ES semantics for CCS, based on FESs and designed to be robust under action refinement [1, 22, 29], were later proposed by Goltz and van Glabbeek [28]. Importantly, all the above-mentioned classes of ESs, except General ESs, give rise to the same *prime algebraic domains* of configurations, from which one can recover a PES by selecting the complete prime elements.

More recently, ES semantics have been investigated for the π-calculus by Crafa, Varacca and Yoshida [17, 48, 18] and by Cristescu, Krivine and Varacca [19–21]. Other causal models for the π-calculus had already been put forward by Jategaonkar and Jagadeesan [35], by Montanari and Pistore [41], by Cattani and Sewell [16] and by Bruni, Melgratti and Montanari [12]. The main new issue, when addressing causality-based semantics for the π-calculus, is the implicit causality induced by scope extrusion. Two alternative views of such implicit causality had been proposed in previous work on noninterleaving operational semantics for the π-calculus, respectively by Boreale and Sangiorgi [6] and by Degano and Priami [26]. Essentially, in [6] an *extruder* (that is, an output of a private name) is considered to cause any action that uses the extruded name,

whether in subject or object position, while in [26] it is considered to cause only the actions that use the extruded name in subject position. Thus, for instance, in the process $P = \nu a\,(\overline{b}\langle a\rangle \mid \overline{c}\langle a\rangle \mid a)$, the two parallel extruders are considered to be causally dependent in the former approach, and independent in the latter. All the causal models for the π-calculus mentioned above, including the ES-based ones, take one or the other of these two stands. Note that opting for the second one leads necessarily to a non-stable ES model, where there may be causal ambiguity within the configurations themselves: for instance, in the above example the maximal configuration contains three events, the extruders $\overline{b}\langle a\rangle$, $\overline{c}\langle a\rangle$ and the input on a, and one does not know which of the two extruders enabled the input. Indeed, the paper [18] uses non-stable ESs. The use of non-stable ESs (General ESs) to express situations where a computational step can merge parts of the state is advocated for instance by Baldan, Corradini and Gadducci in [3]. These ESs give rise to configuration domains that are not prime algebraic, hence the classical representation theorems have to be adjusted.

In our simple setting, where we deal only with single sessions and do not consider session interleaving nor delegation, we can dispense with channels altogether, and therefore the question of parallel extrusion does not arise. In this sense, our notion of causality is closer to that of CCS than to the more complex one of the π-calculus. However, even in a more general setting, where participants would be paired with the channel name of the session they pertain to, the issue of parallel extrusion would not arise: indeed, in the above example b and c should be equal, because participants can only delegate their own channel, but then they could not be in parallel because of linearity, one of the distinguishing features enforced by session types. Hence we believe that in a session-based framework the two above views of implicit causality should collapse into just one.

We now briefly discuss our design choices. Our calculus uses synchronous communication - rather than asynchronous, buffered communication - because this is how communication is modelled in ESs, when they are used to give semantics to process calculi. Concerning the choice operator, we adopted here the basic (and most restrictive) variant for it, as it was originally proposed for multiparty session calculi in [33]. This is essentially a simplifying assumption, and we do not foresee any difficulty in extending our results to a more general choice operator allowing for different receivers, where the projection is more flexible thanks to a merge operator [34]. Finally, concerning subtyping, we envisaged to use the standard preorder on processes, in which a process with fewer outputs is smaller than a process with more outputs. Session Fidelity becomes weaker, since the reduction of global types only assures the reduction of networks, possibly with a different atomic communication. The main drawback is that Theorem 4 would no longer hold, and the domains of network configurations would only be embedded in the domains of their global type configurations.

As regards future work, we plan to define an asynchronous transition system (ATS) [4] for our calculus, along the lines of [10], and show that it provides a noninterleaving operational semantics for networks that is equivalent to their FES semantics. This would enable us also to investigate the issue of reversibility,

jointly on our networks and on their FES representations, since the ATS semantics would give us the handle to unwind networks, while the corresponding FESs could be unrolled following one of the methods proposed in existing work on reversible event structures [44, 21, 30, 31].

Acknowledgments It is a great pleasure for us to contribute to this volume in honour of Rocco, who has been a long-time friend and colleague for all of us. For some of us, this friendship dates back to the early years when Rocco was a Master student at Pisa University. The human qualities that made him become so widely appreciated in the community, namely his friendliness, sense of humour and warmth, as well as his sharpness, dynamism and animating skills, were already quite visible at the time. Since then, Rocco has built up a highly successful career and has acted as an inspiring mentor for several students and young researchers. This is not the place for an exhaustive tribute to Rocco, whose scientific contributions span a wide spectrum of topics and are too numerous to recall. Suffice it to remember, besides his highly influential work on testing in collaboration with Matthew Hennessy, his frontline work on non-interleaving models of computation together with Pierpaolo Degano and Ugo Montanari, and his more recent work on models for service-oriented computing with a number of co-authors from the SENSORIA [5, 14] and ASCENS projects. Our paper is a wink to Rocco's achievements in the last two areas of research.

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

References

1. Aceto, L., Hennessy, M.: Towards action-refinement in process algebras. In: Meyer, A.R. (ed.) LICS. pp. 138–145. IEEE Computer Society Press, Washington (1989)
2. Baier, C., Majster-Cederbaum, M.E.: The connection between an event structure semantics and an operational semantics for TCSP. Acta Informatica **31**(1), 81–104 (1994)
3. Baldan, P., Corradini, A., Gadducci, F.: Domains and event structures for fusions. In: Ouaknine, J. (ed.) LICS. pp. 1–12. IEEE Computer Society Press, Washington (2017)
4. Bednarczyk, M.: Categories of Asynchronous Systems. Ph.D. thesis, University of Sussex (1988)
5. Boreale, M., Bruni, R., De Nicola, R., Loreti, M.: Caspis: a calculus of sessions, pipelines and services. Mathematical Structures in Computer Science **25**(3), 666–709 (2015)
6. Boreale, M., Sangiorgi, D.: A fully abstract semantics for causality in the π-calculus. Acta Informatica **35**(5), 353–400 (1998)
7. Boudol, G., Castellani, I.: On the semantics of concurrency: partial orders and transition systems. In: Ehrig, H., Kowalski, R.A., Levi, G., Montanari, U. (eds.) TAPSOFT. LNCS, vol. 249, pp. 123–137. Springer, Heidelberg (1987)
8. Boudol, G., Castellani, I.: Permutation of transitions: an event structure semantics for CCS and SCCS. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.) REX: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. LNCS, vol. 354, pp. 411–427. Springer, Heidelberg (1988)

9. Boudol, G., Castellani, I.: Flow models of distributed computations: event structures and nets. Research Report 1482, INRIA (1991)
10. Boudol, G., Castellani, I.: Flow models of distributed computations: three equivalent semantics for CCS. Information and Computation **114**(2), 247–314 (1994)
11. Brookes, S., Hoare, C., Roscoe, A.: A theory of communicating sequential processes. Journal of ACM **31**(3), 560–599 (1984)
12. Bruni, R., Melgratti, H.C., Montanari, U.: Event structure semantics for nominal calculi. In: Baier, C., Hermanns, H. (eds.) CONCUR. LNCS, vol. 4137, pp. 295–309. Springer, Heidelberg (2006)
13. Caires, L., Pfenning, F.: Session types as intuitionistic linear propositions. In: Gastin, P., Laroussinie, F. (eds.) CONCUR. LNCS, vol. 6269, pp. 222–236. Springer, Heidelberg (2010)
14. Caires, L., De Nicola, R., Pugliese, R., Vasconcelos, V.T., Zavattaro, G.: Core calculi for service-oriented computing. In: Wirsing, M., Hölzl, M.M. (eds.) Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA Project, LNCS, vol. 6582, pp. 153–188. Springer, Heidelberg (2011)
15. Castellani, I., Dezani-Ciancaglini, M., Giannini, P.: Event structure semantics for multiparty sessions (Extended Version). Research Report 9266, INRIA (2019)
16. Cattani, G.L., Sewell, P.: Models for name-passing processes: interleaving and causal. Information and Computation **190**(2), 136–178 (2004)
17. Crafa, S., Varacca, D., Yoshida, N.: Compositional event structure semantics for the internal π-calculus. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR. LNCS, vol. 4703, pp. 317–332. Springer, Heidelberg (2007)
18. Crafa, S., Varacca, D., Yoshida, N.: Event structure semantics of parallel extrusion in the π-calculus. In: Birkedal, L. (ed.) FOSSACS. LNCS, vol. 7213, pp. 225–239. Springer, Heidelberg (2012)
19. Cristescu, I.: Operational and denotational semantics for the reversible π-calculus. Ph.D. thesis, University Paris Diderot - Paris 7 (2015)
20. Cristescu, I., Krivine, J., Varacca, D.: Rigid families for CCS and the π-calculus. In: Leucker, M., Rueda, C., Valencia, F.D. (eds.) ICTAC. LNCS, vol. 9399, pp. 223–240. Springer, Heidelberg (2015)
21. Cristescu, I., Krivine, J., Varacca, D.: Rigid families for the reversible π-calculus. In: Devitt, S.J., Lanese, I. (eds.) Reversible Computation. LNCS, vol. 9720, pp. 3–19. Springer, Heidelberg (2016)
22. Darondeau, P., Degano, P.: Refinement of actions in event structures and causal trees. Theoretical Computer Science **118**(1), 21–48 (1993)
23. Degano, P., De Nicola, R., Montanari, U.: On the consistency of truly concurrent operational and denotational semantics. In: Chandra, A.K. (ed.) LICS. IEEE Computer Society Press Press, Washington (1988)
24. Degano, P., De Nicola, R., Montanari, U.: A partial ordering semantics for CCS. Theoretical Computer Science **75**(3), 223–262 (1990)
25. Degano, P., Montanari, U.: Concurrent histories: A basis for observing distributed systems. Journal of Computer and System Sciences **34**(2/3), 422–461 (1987)
26. Degano, P., Priami, C.: Non-interleaving semantics for mobile processes. Theoretical Computer Science **216**(1-2), 237–270 (1999)
27. Deniélou, P., Yoshida, N.: Multiparty session types meet communicating automata. In: Seidl, H. (ed.) ESOP. LNCS, vol. 7211, pp. 194–213. Springer, Heidelberg (2012)
28. van Glabbeek, R.J., Goltz, U.: Well-behaved flow event structures for parallel composition and action refinement. Theoretical Computer Science **311**(1-3), 463–478 (2004)

29. Goltz, U., Gorrieri, R., Rensink, A.: Comparing syntactic and semantic action refinement. Information and Computation **125**(2), 118–143 (1996)
30. Graversen, E., Phillips, I., Yoshida, N.: Towards a categorical representation of reversible event structures. In: Vasconcelos, V.T., Haller, P. (eds.) PLACES. EPTCS, vol. 246, pp. 49–60. Open Publishing Association, Waterloo (2017)
31. Graversen, E., Phillips, I., Yoshida, N.: Event structure semantics of (controlled) reversible CCS. In: Kari, J., Ulidowski, I. (eds.) Reversible Computation. LNCS, vol. 11106, pp. 122–102. Springer, Heidelberg (2018)
32. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type discipline for structured communication-based programming. In: Hankin, C. (ed.) ESOP. LNCS, vol. 1381, pp. 122–138. Springer, Heidelberg (1998)
33. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: Necula, G.C., Wadler, P. (eds.) POPL. pp. 273–284. ACM Press, New York (2008)
34. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. Journal of ACM **63**(1), 9:1–9:67 (2016)
35. Jagadeesan, L.J., Jagadeesan, R.: Causality and true concurrency: A data-flow analysis of the π-calculus (extended abstract). In: Alagar, V.S., Nivat, M. (eds.) AMAST. LNCS, vol. 936, pp. 277–291. Springer, Heidelberg (1995)
36. Katoen, J.: Quantitative and qualitative extensions of event structures. Ph.D. thesis, University of Twente (1996)
37. Lange, J., Tuosto, E., Yoshida, N.: From communicating machines to graphical choreographies. In: Rajamani, S.K., Walker, D. (eds.) POPL. pp. 221–232. ACM Press, New York (2015)
38. Langerak, R.: Bundle event structures: a non-interleaving semantics for LOTOS. In: Diaz, M., Groz, R. (eds.) Formal Description Techniques for Distributed Systems and Communication Protocols. pp. 331–346. North-Holland, Amsterdam (1993)
39. Loogen, R., Goltz, U.: Modelling nondeterministic concurrent processes with event structures. Fundamenta Informaticae **14**(1), 39–74 (1991)
40. Milner, R.: A Calculus of Communicating Systems, LNCS, vol. 92. Springer, Heidelberg (1980)
41. Montanari, U., Pistore, M.: Concurrent semantics for the π-calculus. In: Brookes, S., Main, M., Melton, A., Mislove, M. (eds.) MFPS. ENTCS, vol. 1, pp. 411–429. Elsevier, Oxford (1995)
42. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains, part I. Theoretical Computer Science **13**(1), 85–108 (1981)
43. Olderog, E.: TCSP: theory of communicating sequential processes. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) Advances in Petri Nets. LNCS, vol. 255, pp. 441–465. Springer, Heidelberg (1986)
44. Phillips, I., Ulidowski, I.: Reversibility and asymmetric conflict in event structures. Journal of Logical and Algebraic Methods in Programming **84**(6), 781 – 805 (2015)
45. Pierce, B.C.: Types and Programming Languages. MIT Press (2002)
46. Takeuchi, K., Honda, K., Kubo, M.: An interaction-based language and its typing system. In: Hankin, C. (ed.) PARLE. LNCS, vol. 817, pp. 122–138. Springer, Heidelberg (1994)
47. Toninho, B., Caires, L., Pfenning, F.: Dependent session types via intuitionistic linear type theory. In: Schneider-Kamp, P., Hanus, M. (eds.) PPDP. pp. 161–172. ACM Press, New York (2011)
48. Varacca, D., Yoshida, N.: Typed event structures and the linear π-calculus. Theoretical Computer Science **411**(19), 1949–1973 (2010)
49. Wadler, P.: Propositions as sessions. Journal of Functional Programming **24**(2-3), 384–418 (2014)

50. Winskel, G.: Events in Computation. Ph.D. thesis, University of Edinburgh (1980)
51. Winskel, G.: Event structure semantics for CCS and related languages. In: Nielsen, M., Schmidt, E.M. (eds.) ICALP. LNCS, vol. 140, pp. 561–576. Springer, Heidelberg (1982)
52. Winskel, G.: An introduction to event structures. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.) REX: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. LNCS, vol. 354, pp. 364–397. Springer, Heidelberg (1988)