Event structure semantics for multiparty sessions

Ilaria Castellani, Mariangiola Dezani-Ciancaglini, Paola Giannini

Event Structure Semantics for Multiparty Sessions

Ilaria Castellani [a,1,*], Mariangiola Dezani-Ciancaglini[b], Paola Giannini[c,2]

[a]INRIA, Université Côte d'Azur, France
[b]Dipartimento di Informatica, Università di Torino, Italy
[c]DiSSTE,Università del Piemonte Orientale, Italy

Abstract

We propose an interpretation of multiparty sessions as *Flow Event Structures*, which allows concurrency within sessions 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 such global types as *Prime Event Structures*.

Keywords: Communication-centric Systems, Communication-based Programming, Process Calculi, Event Structures, Multiparty Session Types.

1. Introduction

Session types were proposed in the mid-nineties [54, 38], 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 client-server protocols. Subsequently, session types were extended to *multiparty sessions* [39, 40], 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 *deadlock-freedom* [40]. When dealing with multiparty sessions, the type system is often enhanced so as to guarantee also the liveness property known as *progress* (no participant gets stuck) [41].

Some simple examples of sessions not satisfying the above properties are: 1) a

*Corresponding author. INRIA, 2004 Route des Lucioles, BP 93, 06902 Sophia Antipolis FRANCE

Email addresses: `ilaria.castellani@inria.fr` (Ilaria Castellani), `dezani@di.unito.it` (Mariangiola Dezani-Ciancaglini), `paola.giannini@uniupo.it` (Paola Giannini)

[1]This work was partially funded by the ANR17-CE25-0014-01 CISC project.

[2]This original research has the financial support of the Università del Piemonte Orientale. This work was partially funded by the MIUR project "T-LADIES" (PRIN 2020TL3X8X).

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).

What makes session types particularly attractive is that they offer several advantages at once: 1) static safety guarantees, 2) automatic check of protocol implementation correctness, based on local types, and 3) a strong connection with linear logics [13, 55, 59, 52, 14], and with concurrency models such as communicating automata [32], graphical choreographies [44, 56] and message-sequence charts [40].

In this paper we further investigate the relationship between multiparty session types and concurrency models, by focussing on Event Structures [62]. We consider a standard multiparty session calculus where sessions are described as networks of sequential processes [33]. 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 [62]), 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) [60, 49]. 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. To prove this equivalence, we exploit the bisimilarity of their Labelled Transition Systems, as expressed by the Subject Reduction and Session Fidelity theorems (Theorem 6.10 and Theorem 6.11). An alternative approach would have been to compare the two ESs directly, thus conducting the whole reasoning within the denotational model itself. However, while one side of the comparison (mapping the PES of the type to the FES of the network, which can be viewed as a synthesis problem) would be very direct, the other side (reconstructing the PES of the type from the FES of the network) would be more involved, as it would require a structural characterisation of the FESs that represent typable networks, which is far from obvious and therefore is left for future work. This issue will be discussed at length at the end of Section 7.

Event Structures have been used to give semantics to process calculi ever since their introduction at the beginning of the eighties [60, 49] (see Section 9 for an extensive historical discussion). A specific feature of our proposed FES semantics for networks is that we impose strong semantic constraints on the construction of the events themselves (like duality of the histories of their components) in order to reduce the number of events from the very beginning, and to enforce already at the syntactic level some of the expected semantic properties. This allows us to obtain more compact FESs, with fewer events, which is an advantage when displaying their graphical representations[3], as well as handling examples and carrying out

[3]Both FESs and PESs enjoy a graphical representation (see Figure 5 and Figure 6), as opposed to

proofs.

In a companion paper [16], we investigated a similar Event Structure semantics for a session calculus with asynchronous communication, which led to a quite different treatment as it made use of a new notion of asynchronous global type. A detailed comparison with [16] will be given in Section 9.

This paper is an expanded and amended version of [15]. The main novelty is that we use a coinductive definition for processes and global types, which simplifies several definitions and proofs, and a more stringent definition for network events. This definition relies on the new notion of causal set, which is crucial for the correctness of our ES semantics. Finally, the present paper includes the proofs of all results, some of which require ingenuity.

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 to interpret processes (Section 4) and networks (Section 5), respectively. PESs are also used to interpret global types (Section 7), which are defined in Section 6. In Section 8 we prove the equivalence between the FES semantics of a network and the PES semantics of its global type. Section 9 discusses related work and sketches directions for future work.

The proofs of all theorems and propositions are given in the main paper, except for those of Subject Reduction (Theorem 6.10) and Session Fidelity (Theorem 6.11), which are standard and thus deferred to Appendix B. The proofs of lemmas, when not trivial, are collected in Appendices A, B, C, D and E. To help the reader, Appendix F contains a glossary of the symbols used and a table of the notations with their meaning and a reference to where they are defined.

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 $\mathsf{p}, \mathsf{q}, \mathsf{r}, \ldots$ 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 *action*. The action $\mathsf{p}!\lambda$ represents an output of message λ to participant p, while the action $\mathsf{p}?\lambda$ represents an input of message λ from participant p. The *participant of an action*, $\mathsf{pt}(\pi)$, is defined by $\mathsf{pt}(\mathsf{p}!\lambda) = \mathsf{pt}(\mathsf{p}?\lambda) = \mathsf{p}$.

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

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

where I is non-empty and $\lambda_h \neq \lambda_k$ for all $h, k \in I$, $h \neq k$, i.e. messages in choices are all different.

Processes of the shape $\bigoplus_{i\in I} \mathsf{p}!\lambda_i; P_i$ and $\Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$ are called output *and* input processes, *respectively.*

other kinds of stable ESs.

The symbol $::=^{coind}$, in the definition above and in later definitions, indicates that the productions should be interpreted *coinductively*. Namely, they define possibly infinite processes. However, we assume such processes to be *regular*, that is, with finitely many distinct subprocesses. In this way, we only obtain processes which are solutions of finite sets of equations, see [20]. So, when writing processes, we shall use (mutually) recursive equations. When I is a singleton, $\bigoplus_{i\in I} \mathsf{p}!\lambda_i; P_i$ will be rendered as $\mathsf{p}!\lambda; P$ and $\Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$ will be rendered as $\mathsf{p}?\lambda; P$. When I contains only two elements, as it will be the case in most of our examples, we shall feel free to use the binary choices $\mathsf{p}!\lambda_1; P_1 \oplus \mathsf{p}!\lambda_2; P_2$ and $\mathsf{p}!\lambda_1; P_1 + \mathsf{p}!\lambda_2; P_2$, where the branches $\mathsf{p}!\lambda_i; P_i$ should be viewed as being parenthesised (since the connector ; is not an operator of our calculus, but an integral part of the guarded sum operators). Trailing $\mathbf{0}$ processes will be omitted.

Processes may be viewed as trees whose internal nodes are decorated by $\mathsf{p}!$ or $\mathsf{p}?$, leaves by $\mathbf{0}$, and edges by messages λ.

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 6.

Definition 2.2 (Networks). Networks *are defined by:*

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

We assume the standard structural congruence $\equiv$ on networks, stating that parallel composition is associative and commutative and has neutral element $\mathsf{p}[\![\, \mathbf{0} \,]\!]$ for any fresh p. Given the associativity of $\parallel$, we shall feel free to write networks in the form $\mathsf{N} = \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$ in the sequel.

If $P \neq \mathbf{0}$ we write $\mathsf{p}[\![\, P \,]\!] \in \mathsf{N}$ as short for $\mathsf{N} \equiv \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{N}'$ for some N'. We define the *set of participants of* N to be $\{\mathsf{p} \mid \exists P.\ \mathsf{p}[\![\, P \,]\!] \in \mathsf{N}\}$. We say that a network is *unary* if it has a unique participant[4] and *binary* if it has exactly two participants.

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 *communications*, ranged over by α, α', is defined to be $\{\mathsf{pq}\lambda \mid \mathsf{p}, \mathsf{q} \in \mathsf{Part}, \lambda \in \mathsf{Msg}\}$, where $\mathsf{pq}\lambda$ represents the transmission of a message λ from participant p to participant q.

$$\mathsf{p}[\![\, \textstyle\bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i \,]\!] \parallel \mathsf{q}[\![\, \Sigma_{j\in J}\mathsf{p}?\lambda_j; Q_j \,]\!] \parallel \mathsf{N} \xrightarrow{\mathsf{pq}\lambda_k} \mathsf{p}[\![\, P_k \,]\!] \parallel \mathsf{q}[\![\, Q_k \,]\!] \parallel \mathsf{N} \quad \text{where } k \in I \cap J \qquad [\textsc{Com}]$$

Figure 1: LTS for networks.

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 input and output

[4] Unary networks will not be typable, and therefore, by Subject Reduction, a typable network will never evolve to a unary network. On the other hand, this will be possible for non typable networks.

choices. In a well-typed network (see Section 6) it will always be the case that $I \subseteq J$, ensuring that participant p can freely choose an output, since participant q offers all corresponding inputs. Note also that a unary network has no transitions.

Note that we could have given first the (standard) LTS semantics for processes, and then derived the LTS for networks from it. However, the syntax of our calculus is so simple that the LTS for networks can be defined directly. Thus we chose to omit the LTS for processes, which would anyway be of no use in the sequel.

In the following we will make an extensive use of finite (and possibly empty) sequences of communications. As usual we define them as traces.

Definition 2.3 (Traces). (Finite) traces $\sigma \in$ *Traces are defined by:*

$$\sigma ::= \epsilon \mid \alpha \cdot \sigma$$

We use $|\sigma|$ *to denote the length of the trace* σ.
The set of participants of a trace, *notation* $\mathsf{part}(\sigma)$, *is defined by* $\mathsf{part}(\epsilon) = \emptyset$ *and* $\mathsf{part}(\mathsf{pq}\lambda \cdot \sigma) = \{\mathsf{p}, \mathsf{q}\} \cup \mathsf{part}(\sigma)$.

When $\sigma = \alpha_1 \cdot \ldots \cdot \alpha_n$ $(n \geq 1)$ we write $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$ as short for $\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n = \mathsf{N}'$.

3. Event Structures

We recall now the definitions of *Prime Event Structure* (PES) from [60, 49] 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 Sections 4 and 5, while PESs are sufficient to interpret processes, the greater generality of FESs is needed to interpret networks.

Definition 3.1 (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*).*

Definition 3.2 (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. Moreover, 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 3.3 (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 downward-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 3.4 (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 downward-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 configurations. Then, the *domain of configurations* of S is defined as follows:

Definition 3.5 (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 3.6 (Proving sequence). *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:*

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

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 3.7 (Representation of FES configurations as proving sequences [9]). *Given a flow event structure $S = (E, \prec, \#)$, a subset $\mathcal{X}$ 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 3.6 and Proposition 3.7 both for the FESs associated with networks (see Sections 5) and for the PESs associated with global types (see Section 7). Note that for a PES the condition of Definition 3.6 simplifies to

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

To conclude this section, we recall from [17] the definition of *downward surjectivity* (or *downward-onto*, as it was called there), a property that is required for partial functions between two FESs in order to ensure that they preserve configurations. We will make use of this property in Section 5.

Definition 3.8 (Downward surjectivity). *Let $S_i = (E_i, \prec_i, \#_i)$, be a flow event structure, $i = 0, 1$. Let e_i, e_i' range over E_i, $i = 0, 1$. A partial function $f : E_0 \rightarrow_* E_1$ is* downward surjective *if it satisfies the condition:*

$$e_1 \prec_1 f(e_0) \implies \exists e_0' \in E_0 \,.\, e_1 = f(e_0')$$

4. Event Structure Semantics of Processes

In this section, we define an event structure semantics for processes, and show that the obtained event structures are PESs. This semantics will be the basis for defining the ES semantics for networks in Section 5. We start by introducing process events, which are non-empty sequences of actions.

Definition 4.1 (Process event). Process events η, η', *also called* p-events, *are defined by:*

$$\eta \quad ::= \pi \quad | \quad \pi \cdot \eta \qquad\qquad \pi \in \{\mathsf{p}!\lambda, \mathsf{p}?\lambda \mid \mathsf{p} \in \mathsf{Part}, \lambda \in \mathsf{Msg}\}$$

We denote by $\mathcal{PE}$ the set of p-events, and by $|\eta|$ the length of the sequence of actions in the p-event η.

Let ζ denote a (possibly empty) sequence of actions, and $\sqsubseteq$ denote the prefix ordering on such sequences. Each p-event η may be written either in the form $\eta = \pi \cdot \zeta$ or in the form $\eta = \zeta \cdot \pi$. We shall feel free to use any of these forms. When a p-event is written as $\eta = \zeta \cdot \pi$, then ζ may be viewed as the *causal history* of η, namely the sequence of past actions that must have happened in the process for the last action π to be able to happen.

We define the *action* of a p-event to be its last action:

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

Definition 4.2 (Causality and conflict relations on process events). *The* causality *relation* $\leq$ *and the* conflict *relation* $\#$ *on the set of p-events* $\mathcal{PE}$ *are defined by:*

1. $\eta \sqsubseteq \eta' \;\Rightarrow\; \eta \leq \eta'$;

2. $\pi \neq \pi' \Rightarrow \zeta \cdot \pi \cdot \zeta' \;\#\; \zeta \cdot \pi' \cdot \zeta''$.

Definition 4.3 (Event structure of a process). *The* event structure of process P *is the triple*

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

where:

1. $\mathcal{PE}(P) \subseteq \mathcal{PE}$ *is the set of non-empty sequences of labels along the nodes and edges of a path from the root to an edge in the tree of* P*;*

2. $\leq_P$ *is the restriction of* $\leq$ *to the set* $\mathcal{PE}(P)$*;*

3. $\#_P$ *is the restriction of* $\#$ *to the set* $\mathcal{PE}(P)$*.*

It is easy to see that $\#_P = (\mathcal{PE}(P) \times \mathcal{PE}(P)) \setminus (\leq_P \cup \geq_P)$. In the following we shall feel free to drop the subscript in $\leq_P$ and $\#_P$.

Note that the set $\mathcal{PE}(P)$ may be denumerable, as shown by the following example.

Example 4.4. *If* $P = \mathsf{q}!\lambda; P \oplus \mathsf{q}!\lambda'$, *then*

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

Theorem 4.5. *Let* P *be a process. Then* $\mathcal{S}^{\mathcal{P}}(P)$ *is a prime event structure.*

Proof We show that $\leq$ and $\#$ satisfy Properties 2 and 3 of Definition 3.1. Reflexivity, transitivity and antisymmetry of $\leq$ follow from the corresponding properties of $\sqsubseteq$. As for irreflexivity and symmetry of $\#$, they follow from Clause 2 of Definition 4.2 and the corresponding properties of inequality. To show conflict hereditariness, suppose that $\eta \# \eta' \leq \eta''$. From Clause 2 of Definition 4.2 there are π, π', ζ, ζ' and ζ'' such that $\pi \neq \pi'$ and $\eta = \zeta \cdot \pi \cdot \zeta'$ and $\eta' = \zeta \cdot \pi' \cdot \zeta''$. From $\eta' \leq \eta''$ we derive that $\eta'' = \zeta \cdot \pi' \cdot \zeta'' \cdot \zeta_1$ for some ζ_1. Therefore $\eta \# \eta''$, again from Clause 2.

5. Event Structure Semantics of Networks

In this section we define the ES semantics of networks and show that the resulting ESs, which we call *network ESs*, are FESs. We also show that when the network is binary, then the obtained FES is a PES. The formal treatment involves defining the set of potential events of network ESs, which we call *network events*, as well as introducing the notion of *causal set* of a network event and the notion of *narrowing* of a set of network events. This will be the subject of Section 5.1.

In Section 5.2, we first prove some properties of the conflict relation in network ESs. Then, we come back to causal sets and we show that they are always finite and that each configuration includes a unique causal set for each of its network events. We also discuss the relationship between causal sets and prime configurations, which are specific configurations that are in 1-1 correspondence with network events in ESs. Finally, we define a notion of projection of network events on participants, yielding p-events, and prove that this projection (extended to sets of network events) is downward surjective and preserves configurations.

The proofs omitted in this section can be found in Appendix A.

5.1. Definitions and Main Properties

We start by defining network events, the potential events of network ESs. Since these events represent communications between two network participants p and q, they should be pairs of *dual p-events*, namely, of p-events emanating respectively from p and q, which have both dual actions and dual causal histories.

Formally, to define network events we need to specify the *location* of p-events, namely the participant to which they belong:

Definition 5.1 (Located event). *We call* located event *a p-event η pertaining to a participant p, written $\mathsf{p} :: \eta$.*

As hinted above, network events should be pairs of dual located events $\mathsf{p} :: \zeta \cdot \pi$ and $\mathsf{q} :: \zeta' \cdot \pi'$ with matching actions π and π' and matching histories ζ and ζ'. To formalise the matching condition, we first define the projections of p-events on participants, which yield sequences of *undirected actions* of the form $!\lambda$ and $?\lambda$, or the empty sequence ϵ. Then we introduce a notion of duality between located events, based on a notion of duality between undirected actions.

Let ϑ range over $!\lambda$ and $?\lambda$, and Θ range over (possibly empty) sequences of ϑ's.

Definition 5.2 (Projection of p-events on participants). *The projection of a p-event η on a participant p, written $\eta \upharpoonright \mathsf{p}$, is defined by:*

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

$$(\pi \cdot \eta) \upharpoonright \mathsf{p} = \pi \upharpoonright \mathsf{p} \cdot \eta \upharpoonright \mathsf{p}$$

Definition 5.3 (Duality of undirected action sequences). *The* duality of undirected action sequences*, written $\Theta \bowtie \Theta'$, is the symmetric relation induced by:*

$$\epsilon \bowtie \epsilon \qquad \Theta \bowtie \Theta' \Rightarrow\ !\lambda \cdot \Theta \bowtie ?\lambda \cdot \Theta'$$

Definition 5.4 (Duality of located events). *Two located events* $\mathsf{p} :: \eta, \mathsf{q} :: \eta'$ *are* dual, *written* $\mathsf{p} :: \eta \mathrel{\widehat{\bowtie}} \mathsf{q} :: \eta'$, *if* $\eta \Rsh \mathsf{q} \bowtie \eta' \Rsh \mathsf{p}$ *and* $\mathsf{pt}(\mathsf{act}(\eta)) = \mathsf{q}$ *and* $\mathsf{pt}(\mathsf{act}(\eta')) = \mathsf{p}$.

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

Definition 5.5 (Network event). Network events ν, ν', *also called* n-events, *are unordered pairs of dual located events, namely:*

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

We denote by $\mathcal{NE}$ *the set of n-events.*

We define *the communication of the event* ν, notation $\mathsf{cm}(\nu)$, by $\mathsf{cm}(\nu) = \mathsf{pq}\lambda$ if $\nu = \{\mathsf{p} :: \zeta \cdot \mathsf{q}!\lambda, \mathsf{q} :: \zeta' \cdot \mathsf{p}?\lambda\}$ and we say that the n-event ν *represents* the communication $\mathsf{pq}\lambda$. We also define the set of *locations* of an n-event to be $\mathsf{loc}(\{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}) = \{\mathsf{p}, \mathsf{q}\}$.

It is handy to have a notion of occurrence of a located event in a set of network events:

Definition 5.6. A located event $\mathsf{p} :: \eta$ occurs in a set E of n-events, *notation* $\mathsf{p} :: \eta \Subset E$, *if* $\mathsf{p} :: \eta \in \nu$ *and* $\nu \in E$ *for some* ν.

We define now the flow and conflict relations on network events. While the flow relation is the expected one (a network event inherits the causality from its constituent processes), the conflict relation is more subtle, as it can arise also between network events with disjoint sets of locations.

In the following definition we use $|\Theta|$ to denote the length of the sequence Θ.

Definition 5.7 (Flow and conflict relations on n-events). *The* flow *relation* $\prec$ *and the* conflict *relation* $\#$ *on the set of n-events* $\mathcal{NE}$ *are defined by:*

1. $\nu \prec \nu'$ *if* $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta < \eta'$;
2. $\nu \# \nu'$ *if*
   (a) *either* $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta \# \eta'$;
   (b) *or* $\mathsf{p} :: \eta \in \nu$ & $\mathsf{q} :: \eta' \in \nu'$ & $\mathsf{p} \neq \mathsf{q}$ & $|\eta \Rsh \mathsf{q}| = |\eta' \Rsh \mathsf{p}|$ & $\neg(\eta \Rsh \mathsf{q} \bowtie \eta' \Rsh \mathsf{p})$.

Two n-events are in conflict if they share a participant with conflicting p-events (Clause (2a)) or if some of their participants have communicated with each other in the past in incompatible ways (Clause (2b)), as illustrated by the n-events ν and ν' in Example 5.8 (Point 3). Observe that in Clause (2b) the condition $|\eta \Rsh \mathsf{q}| = |\eta' \Rsh \mathsf{p}|$ is needed if we want to check duality of the two projections. Without this condition we could get unwanted conflicts, for instance between $\nu = \{\mathsf{p} :: \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda\}$ and $\nu' = \{\mathsf{p} :: \mathsf{q}!\lambda \cdot \mathsf{q}!\lambda', \mathsf{q} :: \mathsf{p}?\lambda \cdot \mathsf{p}?\lambda'\}$. Removing this condition and checking duality only up to the length of the shortest projection would yield more conflicting events, as discussed in Example 5.8 (Point 3). Note also that the two clauses (2a) and (2b) are not exclusive, as shown in Example 5.8 (Point 4).

Example 5.8. *This example illustrates the use of Definition 5.7 in various cases. It also shows that the flow and conflict relations may be overlapping on n-events.*

1. *Let* $\nu = \{\mathsf{p} :: \mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\}$ *and* $\nu' = \{\mathsf{p} :: \mathsf{q}!\lambda_2, \mathsf{q} :: \mathsf{p}?\lambda_2\}$*. Then* $\nu \# \nu'$ *by Clause (2a) since* $\mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda \# \mathsf{q}!\lambda_2$*. Note that* $\nu \# \nu'$ *can be also deduced by Clause (2b), since* $(\mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda) \Rsh \mathsf{q} = !\lambda_1$ *and* $\mathsf{p}?\lambda_2 \Rsh \mathsf{p} = ?\lambda_2$ *and* $|!\lambda_1| = |?\lambda_2|$ *and* $\neg(!\lambda_1 \bowtie ?\lambda_2)$*.*

2. *Let* ν *be as in (1) and* $\nu' = \{\mathsf{p} :: \mathsf{q}!\lambda_2 \cdot \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda_2 \cdot \mathsf{p}?\lambda\}$*. Again, we can deduce* $\nu \# \nu'$ *using Clause (2a), since* $\mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda \# \mathsf{q}!\lambda_2 \cdot \mathsf{q}!\lambda$*. On the other hand, Clause (2b) does not apply in this case, since* $(\mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda) \Rsh \mathsf{q} = !\lambda_1$ *and* $(\mathsf{p}?\lambda_2 \cdot \mathsf{p}?\lambda) \Rsh \mathsf{p} = ?\lambda_2 \cdot ?\lambda$ *and thus* $|!\lambda_1| \neq |?\lambda_2 \cdot ?\lambda|$*.*

3. *Let* ν *be as in (1) and* $\nu' = \{\mathsf{q} :: \mathsf{p}?\lambda_2 \cdot \mathsf{s}!\lambda, \mathsf{s} :: \mathsf{q}?\lambda\}$*. Here* $\mathsf{loc}(\nu) \cap \mathsf{loc}(\nu') = \emptyset$*, so clearly Clause (2a) does not apply. On the other hand,* $\nu \# \nu'$ *can be deduced by Clause (2b), since* $(\mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda) \Rsh \mathsf{q} = !\lambda_1$ *and* $(\mathsf{p}?\lambda_2 \cdot \mathsf{s}!\lambda) \Rsh \mathsf{p} = ?\lambda_2$ *and* $|!\lambda_1| = |?\lambda_2|$ *and* $\neg(!\lambda_1 \bowtie ?\lambda_2)$*. Consider now* $\nu'' = \{\mathsf{q} :: \mathsf{p}?\lambda_2 \cdot \mathsf{p}?\lambda' \cdot \mathsf{s}!\lambda, \mathsf{s} :: \mathsf{q}?\lambda\}$*. Then we cannot deduce* $\nu \# \nu''$ *in the same way because the two projections do not have the same length. However, we can deduce* $\nu \# \nu''' \prec \nu''$*, where* $\nu''' = \{\mathsf{p} :: \mathsf{q}!\lambda_2, \mathsf{q} :: \mathsf{p}?\lambda_2\}$*. In other words,* ν *and* ν'' *are in* semantic conflict*, as Proposition 5.22 shows, but not in the syntactic conflict* $\#$ *(the fact that semantic conflict is in general larger than syntactic conflict is common to all classes of ESs except PESs). We could have chosen to make the syntactic conflict larger by replacing Clause (2b) by the following alternative clause, where* Θ, Θ' *are as in Definition 5.3 and* $\sqsubseteq$ *is the prefix ordering:*

   Clause (2b′) *or* $\mathsf{p} :: \eta \in \nu \;\&\; \mathsf{q} :: \eta' \in \nu' \;\&\; \mathsf{p} \neq \mathsf{q} \;\&$
   $$(\exists\, \Theta \sqsubseteq \eta \Rsh \mathsf{q}\,, \exists\, \Theta' \sqsubseteq \eta' \Rsh \mathsf{p}\;.\, |\Theta| = |\Theta'| \;\&\; \neg(\Theta \bowtie \Theta'))$$

   With this alternative clause, we could deduce the syntactic conflict $\nu \# \nu''$*. However, in Definition 5.7 we chose to keep our definition of* $\#$ *stricter in order to have fewer syntactic conflicts to handle in examples and proofs.*

4. *Let* ν *be as in (1) and* $\nu' = \{\mathsf{p} :: \mathsf{q}!\lambda_2 \cdot \mathsf{r}!\lambda \cdot \mathsf{r}!\lambda', \mathsf{r} :: \mathsf{p}?\lambda \cdot \mathsf{p}?\lambda'\}$*. In this case we have both* $\nu \prec \nu'$ *by Clause (1) and* $\nu \# \nu'$ *by Clause (2a), namely, causality is inherited from participant* r *and conflict from participant* p*.*

We introduce now the notion of *causal set* of an n-event ν in a given set of events Ev. Intuitively, a causal set of ν in Ev is a *complete set of non-conflicting direct causes* of ν which is included in Ev.

Definition 5.9 (Causal set). *Let* $\nu \in Ev \subseteq \mathcal{NE}$*. A set of n-events* E *is a* causal set *of* ν *in* Ev *if* E *is a minimal subset of* Ev *such that*

1. $E \cup \{\nu\}$ *is conflict-free and*

2. $\mathsf{p} :: \eta \in \nu$ *and* $\eta' < \eta$ *imply* $\mathsf{p} :: \eta' \in\!\in E$*.*

Note that in the above definition, the conjunction of minimality and Clause (2) implies that, if $\nu' \in E$, then $\nu' \prec \nu$. Thus E is a set of direct causes of ν. Moreover, a causal set of an n-event cannot be included in another causal set of the same n-event, as this would contradict the minimality of the larger set. Hence, Definition 5.9 indeed formalises the idea that causal sets should be complete sets of compatible direct causes of a given n-event.

Example 5.10. *Let $\nu_1 = \{\mathsf{p} :: \mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\}$ and $\nu_2 = \{\mathsf{p} :: \mathsf{q}!\lambda_2 \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\}$. Then both $\{\nu_1\}$ and $\{\nu_2\}$ are causal sets of $\nu = \{\mathsf{r} :: \mathsf{p}?\lambda \cdot \mathsf{s}!\lambda', \mathsf{s} :: \mathsf{r}?\lambda'\}$ in $Ev = \{\nu_1, \nu_2, \nu\}$. Note that $\nu_1 \# \nu_2$ and that neither ν_1 nor ν_2 has a causal set in Ev.*

Let us now consider also $\nu_1' = \{\mathsf{p} :: \mathsf{q}!\lambda_1, \mathsf{q} :: \mathsf{p}?\lambda_1\}$ and $\nu_2' = \{\mathsf{p} :: \mathsf{q}!\lambda_2, \mathsf{q} :: \mathsf{p}?\lambda_2\}$. Then ν still has the same causal sets $\{\nu_1\}$ and $\{\nu_2\}$ in $Ev' = \{\nu_1', \nu_2', \nu_1, \nu_2, \nu\}$, while each ν_i, $i = 1, 2$, has the unique causal set $\{\nu_i'\}$ in Ev', and each ν_i', $i = 1, 2$, has the empty causal set in Ev'.

Finally, ν has infinitely many causal sets in $\mathcal{NE}$. For instance, if for every natural number n we let $\nu_n = \{\mathsf{p} :: \mathsf{q}!\lambda_n \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\}$, then each $\{\nu_n\}$ is a causal set of ν in $\mathcal{NE}$. Symmetrically, a causal set may cause infinitely many events in $\mathcal{NE}$. For instance, the above causal sets $\{\nu_1\}$ and $\{\nu_2\}$ of ν could also act as causal sets for any n-event $\nu_n'' = \{\mathsf{r} :: \mathsf{p}?\lambda \cdot \mathsf{s}!\lambda_n, \mathsf{s} :: \mathsf{r}?\lambda_n\}$ or, assuming the set of participants to be denumerable, for any event $\nu_n''' = \{\mathsf{r} :: \mathsf{p}?\lambda \cdot \mathsf{s}_n!\lambda', \mathsf{s}_n :: \mathsf{r}?\lambda'\}$.

When defining the set of events of a network ES, we want to prune out all the n-events that do not have a causal set in the set itself. The reason is that such n-events should not happen in the event structure of a network, although, when projected on their locations (see Definition 5.25), they would always give rise to p-events occurring in a configuration[5]. Example 5.14 should further clarify this point. This pruning is achieved by means of the following narrowing function.

Definition 5.11 (Narrowing of a set of n-events). *The* narrowing *of a set E of n-events, denoted by $\mathsf{n}(E)$, is the greatest fixpoint of the function f_E on sets of n-events defined by:*

$$f_E(X) \quad = \quad \{\nu \in E \mid \exists E' \subseteq X.\, E' \text{ is a causal set of } \nu \text{ in } X\}$$

Note that we could not have taken $\mathsf{n}(E)$ to be the least fixpoint of f_E rather than its greatest fixpoint. Indeed, the least fixpoint of f_E would be the empty set.

Example 5.12. *The following two examples illustrate the notions of causal set and narrowing. Let*

$$\begin{array}{lcllcl}
\nu_1 & = & \{\mathsf{r} :: \mathsf{s}?\lambda_1, \mathsf{s} :: \mathsf{r}!\lambda_1\} & \nu_2 & = & \{\mathsf{r} :: \mathsf{s}?\lambda_2, \mathsf{s} :: \mathsf{r}!\lambda_2\} \\
\nu_3 & = & \{\mathsf{p} :: \mathsf{r}?\lambda_1, \mathsf{r} :: \mathsf{s}?\lambda_1 \cdot \mathsf{p}!\lambda_1\} & \nu_4 & = & \{\mathsf{q} :: \mathsf{s}?\lambda_2, \mathsf{s} :: \mathsf{r}!\lambda_2 \cdot \mathsf{q}!\lambda_2\} \\
\nu_5 & = & \{\mathsf{p} :: \mathsf{r}?\lambda_1 \cdot \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{s}?\lambda_2 \cdot \mathsf{p}?\lambda\} & & &
\end{array}$$

Then $\mathsf{n}(\{\nu_1, \ldots, \nu_5\}) = \{\nu_1, \ldots, \nu_4\}$, because a causal set for ν_5 would need to contain both ν_3 and ν_4, but this is not possible, since $\nu_3 \# \nu_4$ by Clause (2b) of Definition 5.7. In fact $(\mathsf{s}?\lambda_1 \cdot \mathsf{p}!\lambda_1) \upharpoonright \mathsf{s} = ?\lambda_1$ and $(\mathsf{r}!\lambda_2 \cdot \mathsf{q}!\lambda_2) \upharpoonright \mathsf{r} = !\lambda_2$ and $|?\lambda_1| = |!\lambda_2|$ and $\neg(?\lambda_1 \bowtie !\lambda_2)$. Let

$$\begin{array}{lcllcl}
\nu_1 & = & \{\mathsf{r} :: \mathsf{s}?\lambda_1, \mathsf{s} :: \mathsf{r}!\lambda_1\} & \nu_2 & = & \{\mathsf{r} :: \mathsf{s}?\lambda_2, \mathsf{s} :: \mathsf{r}!\lambda_2\} \\
\nu_3 & = & \{\mathsf{p} :: \mathsf{r}?\lambda_1, \mathsf{r} :: \mathsf{s}?\lambda_1 \cdot \mathsf{p}!\lambda_1\} & \nu_4 & = & \{\mathsf{p} :: \mathsf{r}?\lambda_1 \cdot \mathsf{s}?\lambda_2, \mathsf{s} :: \mathsf{r}!\lambda_2 \cdot \mathsf{p}!\lambda_2\} \\
\nu_5 & = & \{\mathsf{p} :: \mathsf{r}?\lambda_1 \cdot \mathsf{s}?\lambda_2 \cdot \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda\} & & &
\end{array}$$

Here $\mathsf{n}(\{\nu_1, \ldots, \nu_5\}) = \{\nu_1, \nu_2, \nu_3\}$. Indeed, a causal set for ν_4 would need to contain both ν_2 and ν_3, but this is not possible, since $\nu_2 \# \nu_3$ by Clause (2a) of Definition 5.7. In fact $\mathsf{s}?\lambda_2 \# \mathsf{s}?\lambda_1 \cdot \mathsf{p}!\lambda_1$. Then, ν_5 will also be pruned by the narrowing, since any causal set for ν_5 should contain ν_4.

[5] In fact, every event of a PES occurs in a configuration.

We can now finally define the event structure associated with a network. The intuition is that the events appearing in some configuration of the event structure should correspond exactly to the transitions executable in some state of the network.

Definition 5.13 (Event structure of a network). *The* event structure of network N *is the triple*

$$\mathcal{S}^{\mathcal{N}}(\mathsf{N}) = (\mathcal{NE}(\mathsf{N}), \prec_{\mathsf{N}}, \#_{\mathsf{N}})$$

where:

1. $\mathcal{NE}(\mathsf{N}) = \mathsf{n}(\mathcal{CE}(\mathsf{N}))$ *with* $\mathcal{CE}(\mathsf{N}) = \{\{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\} \mid \mathsf{p}[\![\, P \,]\!] \in \mathsf{N}, \mathsf{q}[\![\, Q \,]\!] \in \mathsf{N}, \eta \in \mathcal{PE}(P), \eta' \in \mathcal{PE}(Q), \mathsf{p} :: \eta \;\widehat{\bowtie}\; \mathsf{q} :: \eta'\}$
2. $\prec_{\mathsf{N}}$ *is the restriction of* $\prec$ *to the set* $\mathcal{NE}(\mathsf{N})$*;*
3. $\#_{\mathsf{N}}$ *is the restriction of* $\#$ *to the set* $\mathcal{NE}(\mathsf{N})$.

The set of n-events of the ES associated with a network N is the narrowing of its set of *candidate* n-events, $\mathcal{CE}(\mathsf{N})$, which contains all pairs of dual located events that may be constructed from two different components of N. We give now a simple example that justifies the use of the narrowing function for building the set of events of a network ES.

Example 5.14. *Let* $\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}?\lambda \cdot \mathsf{r}!\lambda' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\lambda' \,]\!]$. *Then* $\mathcal{CE}(\mathsf{N})$ *contains the unique n-event* $\nu = \{\mathsf{p} :: \mathsf{q}?\lambda \cdot \mathsf{r}!\lambda', \mathsf{r} :: \mathsf{p}?\lambda'\}$. *If we did not apply the narrowing function to* $\mathcal{CE}(\mathsf{N})$*, namely if we took* $\mathcal{CE}(\mathsf{N})$ *as the set of n-events for* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$*, then* $\{\nu\}$ *would be a possible configuration of* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$*, which is clearly wrong, since the network* N *does not have a corresponding transition. Instead, by applying the narrowing function to* $\mathcal{CE}(\mathsf{N})$ *we obtain* $\mathcal{NE}(\mathsf{N}) = \mathsf{n}(\mathcal{CE}(\mathsf{N})) = \emptyset$*, since the n-event* ν *has no causal set in* $\mathcal{CE}(\mathsf{N})$*, which is what we expect.*

The set of n-events of a network ES can be infinite, as shown by the following example.

Example 5.15. *Let* P *be as in Example 4.4,* $Q = \mathsf{p}?\lambda; Q + \mathsf{p}?\lambda'$ *and* $\mathsf{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!]$. *Then*

$$\begin{aligned}\mathcal{NE}(\mathsf{N}) = \;& \{\{\mathsf{p} :: \underbrace{\mathsf{q}!\lambda \cdot \ldots \cdot \mathsf{q}!\lambda}_{n}, \mathsf{q} :: \underbrace{\mathsf{p}?\lambda \cdot \ldots \cdot \mathsf{p}?\lambda}_{n}\} \mid n \geq 1\} \quad \cup \\ & \{\{\mathsf{p} :: \underbrace{\mathsf{q}!\lambda \cdot \ldots \cdot \mathsf{q}!\lambda}_{n} \cdot \mathsf{q}!\lambda', \mathsf{q} :: \underbrace{\mathsf{p}?\lambda \cdot \ldots \cdot \mathsf{p}?\lambda}_{n} \cdot \mathsf{p}?\lambda'\} \mid n \geq 0\}\end{aligned}$$

A simple variation of this example shows that even within the events of a network ES, an n-event ν *may have an infinite number of causal sets. Let* $\nu = \{\mathsf{r} :: \mathsf{p}?\lambda \cdot \mathsf{s}!\lambda', \mathsf{s} :: \mathsf{r}?\lambda'\}$ *be as in Example 5.10. Consider the network* $\mathsf{N}' = \mathsf{p}[\![\, P' \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!] \parallel \mathsf{s}[\![\, S \,]\!]$*, where* $P' = \mathsf{q}!\lambda; P' \oplus \mathsf{q}!\lambda'; \mathsf{r}!\lambda$*,* Q *is as above,* $R = \mathsf{p}?\lambda; \mathsf{s}!\lambda'$ *and* $S = \mathsf{r}?\lambda'$*. Then* ν *has an infinite number of causal sets* $E_n = \{\nu_n\}$ *in* $\mathcal{NE}(\mathsf{N}')$*, where*

$$\nu_n = \{\mathsf{p} :: \underbrace{\mathsf{q}!\lambda \cdot \ldots \cdot \mathsf{q}!\lambda}_{n} \cdot \mathsf{q}!\lambda' \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\}$$

On the other hand, a causal set may only cause a finite number of events in a network ES, since the number of branches in any choice is finite, as well as the number of participants in the network.

Theorem 5.16. *Let* N *be a network. Then* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ *is a flow event structure with an irreflexive conflict relation.*

Proof The relation $\prec_{\mathsf{N}}$ is irreflexive since $\eta < \eta'$ implies $\nu \neq \nu'$, where η, η', ν, ν' are as in Definition 5.7(1). As for the conflict relation, note first that a conflict between an n-event and itself could not be derived by Clause (2b) of Definition 5.7, since the two located events of an n-event are dual by construction. Lastly, symmetry and irreflexivity of the conflict relation follow from the corresponding properties of conflict between p-events.

The fact that the conflict relation is irreflexive in our network FESs means that we do not exploit the possibility of self-conflicts offered by general FESs. This is due to the way we defined the set of events of our network FESs, using the narrowing function as discussed previously. We could have chosen an alternative definition, introducing additional self-conflicting events of a more liberal form[6] which would have disappeared when building configurations (together with their successors having no other possible causes), as it was done for CCS in [10]. However, this would have resulted in much larger sets of events for network FESs, leading to more cumbersome examples and proofs. Our design choice here was to reduce the set of events of network FESs by introducing already some semantic constraints on their events (like duality and the existence of causal sets). It should be stressed, however, that the narrowing function does not exclude all non executable events, as shown by the FES in Example 5.20, which has three events, each of which has a causal set but none of which is executable.
Although they have an irreflexive conflict relation like PESs, our network FESs exhibit two important features which are not shared by PESs, namely non-hereditary conflict (as shown by the FES given in Figure 5, where the two conflicting events ν'_1 and ν'_2 have a common successor ν) and causality cycles (as shown by the FES in Example 5.20, where there is a circular dependency among the three events ν_1, ν_2 and ν_3).

Note that n-events with disjoint sets of locations may be related by the transitive closure of the flow relation, as illustrated by the next example, which also shows how n-events inherit the flow relation from the causality relation of their p-events.

Example 5.17. *Let* N *be the network*

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

[6] For instance, we could have allowed events of the form $\{\mathsf{p} :: \eta, *\}$ to represent incomplete communications, and then prevented them from occurring by putting them in conflict with themselves. In this case, the event ν of Example 5.14 would have also been prevented from occurring because of its unique self-conflicting cause $\{\mathsf{p} :: \mathsf{q}?\lambda, *\}$, and we would not have needed the narrowing function.

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

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

The flow relation obtained by Definition 5.13 is: $\nu_1 \prec \nu_2$ *and* $\nu_2 \prec \nu_3$. *These two flows are inherited from the causality relations within the process ESs associated with participants* q *and* r*, respectively. The non-empty configurations are* $\{\nu_1\}, \{\nu_1, \nu_2\}$ *and* $\{\nu_1, \nu_2, \nu_3\}$. *Note that* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ *has only one proving sequence per configuration (which is the one given by the numbering of events).*

Clearly, if a network is unary, then the set of events of its FES is empty. If a network is binary, then its FES may be turned into a PES by replacing $\prec$ with its reflexive and transitive closure $\prec^*$. To prove this result, we first show a property of n-events of binary networks. We say that an n-event ν is *binary* if the participants occurring in the p-events of ν are contained in $\mathsf{loc}(\nu)$.

Lemma 5.18. *Let* ν *and* ν' *be binary n-events with* $\mathsf{loc}(\nu) = \mathsf{loc}(\nu')$. *Then* $\nu \# \nu'$ *iff* $\mathsf{p} :: \eta \in \nu$ *and* $\mathsf{p} :: \eta' \in \nu'$ *imply* $\eta \# \eta'$.

Proposition 5.19. *Let* $\mathsf{N} = \mathsf{p}_1[\![P_1]\!] \parallel \mathsf{p}_2[\![P_2]\!]$ *and* $\mathcal{S}^{\mathcal{N}}(\mathsf{N}) = (\mathcal{NE}(\mathsf{N}), \prec_{\mathsf{N}}, \#)$. *Then* $\mathsf{n}(\mathcal{CE}(\mathsf{N})) = \mathcal{CE}(\mathsf{N})$ *and the structure* $\mathcal{S}^{\mathcal{N}}_*(\mathsf{N}) =_{\mathsf{def}} (\mathcal{NE}(\mathsf{N}), \prec^*_{\mathsf{N}}, \#)$ *is a prime event structure.*

Proof We first show that $\mathsf{n}(\mathcal{CE}(\mathsf{N})) = \mathcal{CE}(\mathsf{N})$. By Definition 5.13(1)

$$\mathcal{CE}(\mathsf{N}) = \{\{\mathsf{p}_1 :: \eta_1, \mathsf{p}_2 :: \eta_2\} \mid \eta_1 \in \mathcal{PE}(P_1), \eta_2 \in \mathcal{PE}(P_2), \mathsf{p}_1 :: \eta_1 \mathrel{\widehat{\bowtie}} \mathsf{p}_2 :: \eta_2\}$$

Let $\{\mathsf{p}_1 :: \eta_1, \mathsf{p}_2 :: \eta_2\} \in \mathcal{CE}(\mathsf{N})$. Since $\mathsf{p}_1 :: \eta_1 \mathrel{\widehat{\bowtie}} \mathsf{p}_2 :: \eta_2$ and all the actions in η_1 involve p_2 and all the actions in η_2 involve p_1, we know that η_1 and η_2 have the same length $n \geq 1$ and for each $i, 1 \leq i \leq n$, the prefixes of length i of η_1 and η_2, written η_1^i and η_2^i, must themselves be dual. Then $\{\mathsf{p}_1 :: \eta_1^i, \mathsf{p}_2 :: \eta_2^i\} \in \mathcal{CE}(\mathsf{N})$ for each $i, 1 \leq i \leq n$, hence $\{\mathsf{p}_1 :: \eta_1, \mathsf{p}_2 :: \eta_2\}$ has a causal set in $\mathcal{CE}(\mathsf{N})$.
We prove now that the reflexive and transitive closure $\prec^*_{\mathsf{N}}$ of $\prec_{\mathsf{N}}$ is a partial order. Since by definition $\prec^*_{\mathsf{N}}$ is a preorder, we only need to show that it is antisymmetric. Define the length of an n-event $\nu = \{\mathsf{p}_1 :: \eta_1, \mathsf{p}_2 :: \eta_2\}$ to be $\mathsf{length}(\nu) =_{\mathsf{def}} |\eta_1| + |\eta_2|$ (where $|\eta|$ is the length of η). Let now $\nu, \nu' \in \mathcal{NE}(\mathsf{N})$, with $\nu = \{\mathsf{p}_1 :: \eta_1, \mathsf{p}_2 :: \eta_2\}$ and $\nu' = \{\mathsf{p}_1 :: \eta_1', \mathsf{p}_2 :: \eta_2'\}$. By definition $\nu \prec_{\mathsf{N}} \nu'$ implies $\eta_i < \eta_i'$ for some $i = 1, 2$, which in turn implies $|\eta_i| < |\eta_i'|$. As observed above, η_1 and η_2 must have the same length, and so must η_1' and η_2'. This means that if $\nu \prec_{\mathsf{N}} \nu'$ then $\mathsf{length}(\nu) = |\eta_1| + |\eta_2| < |\eta_1'| + |\eta_2'| = \mathsf{length}(\nu')$. From this we can conclude that if $\nu \prec^*_{\mathsf{N}} \nu'$ and $\nu' \prec^*_{\mathsf{N}} \nu$, then necessarily $\nu = \nu'$.
Finally we show that the relation $\#$ satisfies the required properties. By Theorem 5.16 we only need to prove that $\#$ is hereditary. Let ν and ν' be as above. If $\nu \# \nu'$, then by Lemma 5.18 $\eta_1 \# \eta_1'$ and $\eta_2 \# \eta_2'$. Let now $\nu'' = \{\mathsf{p}_1 :: \eta_1'', \mathsf{p}_2 :: \eta_2''\}$. If $\nu' \prec^*_{\mathsf{N}} \nu''$, this means that there exist $\nu_1, \ldots, \nu_n$ such that $\nu' \prec_{\mathsf{N}} \nu_1 \ldots \prec_{\mathsf{N}} \nu_n = \nu''$. We prove by induction on n that $\nu \# \nu''$. For $n = 1$ we have $\nu' \prec_{\mathsf{N}} \nu''$. Then by Clause (1) of Definition 5.13 we have $\eta_j' < \eta_j''$ for some $j \in \{1, 2\}$. Since $\eta_i \# \eta_i'$ for all $i \in \{1, 2\}$ and $\#$ is hereditary on p-events, we deduce $\eta_j \# \eta_j''$, which implies $\nu \# \nu''$. Suppose now $n > 1$. By induction $\nu \# \nu_{n-1}$. Since $\nu_{n-1} \prec_{\mathsf{N}} \nu_n = \nu''$ we then obtain $\nu \# \nu''$ by the same argument as in the base case.

If a network has more than two participants, then the duality requirement on its n-events is not sufficient to ensure the absence of circular dependencies[7]. 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 5.20. *Let* 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 $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ *has three n-events*

$$\nu_1 = \{\mathsf{p} :: \mathsf{r}?\lambda, \mathsf{r} :: \mathsf{q}?\lambda'' \cdot \mathsf{p}!\lambda\} \qquad \nu_2 = \{\mathsf{p} :: \mathsf{r}?\lambda \cdot \mathsf{q}!\lambda', \mathsf{q} :: \mathsf{p}?\lambda'\}$$
$$\nu_3 = \{\mathsf{q} :: \mathsf{p}?\lambda' \cdot \mathsf{r}!\lambda'', \mathsf{r} :: \mathsf{q}?\lambda''\}$$

By Definition 5.13(1) we have $\nu_1 \prec \nu_2 \prec \nu_3$ *and* $\nu_3 \prec \nu_1$. *The only configuration of* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ *is the empty configuration, because the only set of n-events that satisfies downward-closure up to conflicts 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 3.4 is not satisfied.*

5.2. Further Properties

In this subsection, we first prove two properties of the conflict relation in network ESs: non disjoint n-events are always in conflict, and conflict induced by Clause (2b) of Definition 5.7 is semantically inherited. We then discuss the relationship between causal sets and prime configurations and prove two further properties of causal sets, which are shared with prime configurations[8]: finiteness, and the existence of a causal set for each event in a configuration. Finally, observing that the FES of a network may be viewed as the product of the PESs of its processes, we proceed to prove a classical property for ES products, namely that their projections on their components preserve configurations. To this end, we define a projection function from n-events to participants, yielding p-events, and we show that configurations of a network ES project down to configurations of the PESs of its processes.

Let us start with the conflict properties. By definition, two n-events intersect each other if and only if they share a located event $\mathsf{p} :: \eta$. Otherwise, the two n-events are disjoint. Note that if $\mathsf{p} :: \eta \in (\nu \cap \nu')$, then $\mathsf{loc}(\nu) = \mathsf{loc}(\nu') = \{\mathsf{p}, \mathsf{q}\}$, where $\mathsf{q} = \mathsf{pt}(\mathsf{act}(\eta))$. The next proposition establishes that two distinct intersecting n-events in $\mathcal{NE}$ are in conflict.

Lemma 5.21 (Sharing of located events implies conflict). *If* $\nu, \nu' \in \mathcal{NE}$ *and* $\nu \neq \nu'$ *and* $(\nu \cap \nu') \neq \emptyset$, *then* $\nu \,\#\, \nu'$.

Although conflict is not hereditary in FESs, we prove that a conflict due to incompatible mutual projections (i.e., a conflict derived by Clause (2b) of Definition 5.7) is semantically inherited. Let $\vartheta \searrow n$ denote the prefix of length n of ϑ.

[7] This is a well-known issue in multiparty session types, which motivated the introduction of global types in [39], see Section 6.

[8] A prime configuration is a configuration with a unique maximal element, its *culminating* event.

Proposition 5.22 (Semantic conflict hereditariness). *Let* $\mathsf{p} :: \eta \in \nu$ *and* $\mathsf{q} :: \eta' \in \nu'$ *with* $\mathsf{p} \neq \mathsf{q}$*. Let* $n = min\{|\eta \upharpoonright \mathsf{q}|, |\eta' \upharpoonright \mathsf{p}|\}$*. If* $\neg((\eta \upharpoonright \mathsf{q}) \searrow n \bowtie (\eta' \upharpoonright \mathsf{p}) \searrow n)$*, then there exists no configuration* $\mathcal{X}$ *such that* $\nu, \nu' \in \mathcal{X}$*.*

Proof Suppose ad absurdum that $\mathcal{X}$ is a configuration such that $\nu, \nu' \in \mathcal{X}$. If $|\eta \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}|$ then $\nu \# \nu'$ by Definition 5.7(2b) and we reach immediately a contradiction. So, assume $|\eta \upharpoonright \mathsf{q}| > |\eta' \upharpoonright \mathsf{p}| = n$. This means that $|\eta| > 1$ and thus there exists a non-empty causal set E_ν of ν such that $E_\nu \subseteq \mathcal{X}$. Let $\eta_0 < \eta$ be such that $|\eta_0 \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}| = n$. By definition of causal set, there exists $\nu_0 \in E_\nu$ such that $\mathsf{p} :: \eta_0 \in \nu_0$. By Definition 5.7(2b) we have then $\nu_0 \# \nu'$, contradicting the fact that $\mathcal{X}$ is conflict-free.

We prove now two further properties of causal sets. For the reader familiar with ESs, the notion of causal set may be reminiscent of that of *prime configuration* [60], which similarly consists of a complete set of causes for a given event[9]. However, there are some important differences: the first is that a causal set does not include the event it causes, unlike a prime configuration. The second is that a causal set only contains direct causes of an event, and thus it is not downward-closed up to conflicts, as opposed to a prime configuration. The last difference is that, while a prime configuration uniquely identifies its caused event, a causal set may cause different events, as shown in Example 5.10.

A common feature of prime configurations and causal sets is that they are both finite. For causal sets, this is implied by minimality together with Clause (2) of Definition 5.9, as shown by the following proposition.

Proposition 5.23. *Let* $\nu \in Ev \subseteq \mathcal{NE}$*. If* E *is a causal set of* ν *in* Ev*, then* E *is finite.*

Proof Suppose $\nu = \{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}$. We show that $|E| \leq |\eta| + |\eta'| - 2$, where $|E|$ is the cardinality of E. By Condition (2) of Definition 5.9, for each $\eta_0 < \eta$ and $\eta'_0 < \eta'$ there must be $\nu_0, \nu'_0 \in E$ such that $\mathsf{p} :: \eta_0 \in \nu_0$ and $\mathsf{q} :: \eta'_0 \in \nu'_0$. Note that ν_0 and ν'_0 could possibly coincide. Moreover, there cannot be $\nu' \in E$ such that $\mathsf{p} :: \eta_0 \in \nu' \neq \nu_0$ or $\mathsf{q} :: \eta'_0 \in \nu' \neq \nu'_0$, since this would contradict the minimality of E (and also its conflict-freeness, since by Lemma 5.21 we would have either $\nu' \# \nu_0$ or $\nu' \# \nu'_0$). Hence the number of events in E is at most $(|\eta| - 1) + (|\eta'| - 1)$.

A key property of causal sets, which is again shared with prime configurations, is that each configuration includes a unique causal set for each n-event in the configuration.

Lemma 5.24. *If* $\mathcal{X}$ *is a configuration of* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ *and* $\nu \in \mathcal{X}$*, then there is a unique causal set* E *of* ν *such that* $E \subseteq \mathcal{X}$*.*

In the remainder of this section we show that projections of n-event configurations give p-event configurations. We start by formalising the projection function of n-events on participants, which yields p-events, and showing that it is downward surjective.

[9] In PESs, the prime configuration associated with an event is unique, while it is not unique in FESs and more generally in Stable ESs, just like a causal set.

Definition 5.25 (Projection of n-events on participants).

$$proj_{\mathsf{p}}(\nu) = \begin{cases} \eta & \text{if } \mathsf{p} :: \eta \in \nu, \\ undefined & otherwise. \end{cases}$$

The projection function $proj_{\mathsf{p}}(\cdot)$ is extended to sets of n-events in the obvious way:

$$proj_{\mathsf{p}}(X) = \{\eta \mid \exists \nu \in X \,.\, proj_{\mathsf{p}}(\nu) = \eta\}$$

Example 5.26. *Let $\{\nu_1, \nu_2, \nu_3\}$ be the configuration defined in Example 5.17. We get*

$$proj_{\mathsf{q}}(\{\nu_1, \nu_2, \nu_3\}) = \{\mathsf{p}?\lambda_1, \mathsf{p}?\lambda_1 \cdot \mathsf{r}!\lambda_2\}$$

Example 5.27. *Let N and ν be as in Example 5.14. As observed there, if we did not apply narrowing the set of events of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ would be the singleton $\{\nu\}$, which would also be a configuration of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$. However, $proj_{\mathsf{p}}(\nu) = \{\mathsf{q}?\lambda \cdot \mathsf{r}!\lambda'\}$ would not be a configuration in $\mathcal{S}^{\mathcal{P}}(P)$, since it would contain the event $\mathsf{q}?\lambda \cdot \mathsf{r}!\lambda'$ without its cause $\mathsf{q}?\lambda$.*

Narrowing ensures that each projection of the set of n-events of a network FES on one of its participants is downward surjective (according to Definition 3.8).

Proposition 5.28 (Downward surjectivity of projections).
Let $\mathcal{S}^{\mathcal{N}}(\mathsf{N}) = (\mathcal{NE}(\mathsf{N}), \prec_{\mathsf{N}}, \#_{\mathsf{N}})$ and $\mathcal{S}^{\mathcal{P}}(P) = (\mathcal{PE}(P), \leq_P, \#_P)$ and $\mathsf{p}[\![P]\!] \in \mathsf{N}$. Then the partial function $proj_{\mathsf{p}} : \mathcal{NE}(\mathsf{N}) \rightharpoonup \mathcal{PE}(P)$ is downward surjective.

Proof As mentioned already in Section 3, any PES $S = (E, \leq, \#)$ may be viewed as a FES, with $\prec$ given by $<$ (the strict ordering underlying $\leq$). Let $\eta \in \mathcal{PE}(P)$ and $\nu \in \mathcal{NE}(\mathsf{N})$. Then the property we need to show is:

$$\eta <_P proj_{\mathsf{p}}(\nu) \implies \exists \nu' \in \mathcal{NE}(\mathsf{N}) \,.\, \eta = proj_{\mathsf{p}}(\nu')$$

Note that $\eta <_P proj_{\mathsf{p}}(\nu)$ implies $proj_{\mathsf{p}}(\nu) = \eta \cdot \eta'$ for some η'. Recall that $\mathcal{NE}(\mathsf{N}) = \mathsf{n}(\mathcal{CE}(\mathsf{N}))$, where $\mathsf{n}(\cdot)$ is the narrowing function (Definition 5.11).
By definition of narrowing, $\mathsf{p} :: \eta \cdot \eta' \Subset \mathcal{NE}(\mathsf{N})$ implies that there is $E \subseteq \mathcal{NE}(\mathsf{N})$ such that E is a causal set of ν in $\mathcal{NE}(\mathsf{N})$. Therefore $\mathsf{p} :: \eta \cdot \eta' \in \nu$ implies $\mathsf{p} :: \eta \Subset E$ and so $\mathsf{p} :: \eta \Subset \mathcal{NE}(\mathsf{N})$, which is what we wanted to show.

Theorem 5.29 (Projection of n-events preserves configurations). *If $\mathsf{p}[\![P]\!] \in \mathsf{N}$, then $\mathcal{X} \in \mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathsf{N}))$ implies $proj_{\mathsf{p}}(\mathcal{X}) \in \mathcal{C}(\mathcal{S}^{\mathcal{P}}(P))$.*

Proof Clearly, $proj_{\mathsf{p}}(\mathcal{X})$ is conflict-free. We show that it is also downward-closed. If $\nu \in \mathcal{X}$, by Lemma 5.24 there is a causal set E of ν such that $E \subseteq \mathcal{X}$. If $\mathsf{p} :: \eta \in \nu$ and $\eta' < \eta$, by Definition 5.9 there is $\nu' \in E$ such that $\mathsf{p} :: \eta' \in \nu'$. We conclude that $\nu' \in \mathcal{X}$, and therefore $\eta' \in proj_{\mathsf{p}}(\mathcal{X})$.

Notice that the reverse of Theorem 5.29 is not true, namely $\mathsf{p}[\![P]\!] \in \mathsf{N}$ does not imply that each configuration of $\mathcal{C}(\mathcal{S}^{\mathcal{P}}(P))$ can be obtained by projecting some configuration of $\mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathsf{N}))$ on p. Consider for instance the network $\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}?\lambda \,]\!]$. Then $\{\mathsf{q}?\lambda\} \in \mathcal{C}(\mathcal{S}^{\mathcal{P}}(P))$, while $\mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathsf{N})) = \emptyset$.

The reader may wonder why our ES semantics for sessions is not cast in categorical terms, like classical ES semantics for process calculi [60, 17], where process constructions arise as categorical constructions (e.g., parallel composition arises as a categorical product). In fact, a categorical formulation of our semantics would not be possible, due to our two-level syntax for processes and networks, which does not allow networks to be further composed in parallel. However, it should be clear that our construction of a network FES from the process PESs of its components is a form of parallel composition, and the properties expressed by Proposition 5.28 and Theorem 5.29 give some evidence that this construction satisfies the conditions usually required for a categorical product of ESs.

6. Global Types

This section is devoted to our type system for multiparty sessions. Global types describe the communication protocols involving all session participants. Usually, global types are projected into local types and typing rules are used to derive local types for processes [39, 19, 40]. The simplicity of our calculus allows us to project directly global types into processes and to have exactly one typing rule, see Figures 2 and 3. This section is split in two subsections.
The first subsection presents the projection of global types onto processes, together with the proof of its soundness. Moreover it introduces a *boundedness* condition on global types, which is crucial for our type system to ensure progress.
The second subsection presents the type system, as well as an LTS for global types. Lastly, the properties of Subject Reduction, Session Fidelity and Progress are shown. The omitted proofs can be found in Appendix B.

6.1. Well-formed Global Types

Global types are built from choices among communications.

Definition 6.1 (Global types). *Global types* G *are defined by:*

$$\mathsf{G} \quad ::=^{coind} \quad \mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i \;\mid\; \mathsf{End}$$

where I is not empty, $\lambda_h \neq \lambda_k$ for all $h, k \in I$, $h \neq k$, i.e. messages in choices are all different.

As for processes, $::=^{coind}$ indicates that global types are defined coinductively. Again, we focus on *regular* terms. Since also processes are defined coinductively this allows for a simpler definition of projection, see Figure 2.

The type $\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i$ formalises a protocol which starts with the communication of a message λ_k from p to q, for some $k \in I$, and then, depending on which λ_k was chosen by p, continues as G_k.

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 $\mathsf{p} \xrightarrow{\lambda} \mathsf{q}; \mathsf{G}$. When I contains only two elements, as for processes we will use the binary choice notation $\mathsf{p} \to \mathsf{q} : (\lambda_1; \mathsf{G}_1 \boxplus \lambda_2; \mathsf{G}_2)$. Trailing End types will be omitted.

$$\mathsf{G} \upharpoonright \mathsf{r} \; = \mathbf{0} \text{ if } \mathsf{r} \notin \mathsf{part}(\mathsf{G})$$

$$(\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i) \upharpoonright \mathsf{r} \; = \begin{cases} \Sigma_{i \in I} \mathsf{p}?\lambda_i; \mathsf{G}_i \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{q}, \\ \bigoplus_{i \in I} \mathsf{q}!\lambda_i; \mathsf{G}_i \upharpoonright \mathsf{r} & \text{if } \mathsf{r} = \mathsf{p}, \\ \mathsf{G}_1 \upharpoonright \mathsf{r} & \text{if } \mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\} \text{ and } \mathsf{r} \in \mathsf{part}(\mathsf{G}_1) \text{ and} \\ & \mathsf{G}_i \upharpoonright \mathsf{r} \; = \mathsf{G}_1 \upharpoonright \mathsf{r} \text{ for all } i \in I \end{cases}$$

Figure 2: Projection of global types onto participants.

Global types may be viewed as trees whose internal nodes are decorated by pq, leaves by End, and edges by messages λ. Given a global type, the sequences of decorations of nodes and edges on the path from the root to an edge in the tree of the global type are traces, in the sense of Definition 2.3. We denote by $\mathsf{Tr}^+(\mathsf{G})$ the set of traces of G. By definition, $\mathsf{Tr}^+(\mathsf{End}) = \emptyset$ and each trace in $\mathsf{Tr}^+(\mathsf{G})$ is non-empty.

The set of *participants of a global type* G, $\mathsf{part}(\mathsf{G})$, is defined to be the union of the sets of participants of all its traces, namely

$$\mathsf{part}(\mathsf{G}) = \bigcup_{\sigma \in \mathsf{Tr}^+(\mathsf{G})} \mathsf{part}(\sigma)$$

Note that the regularity assumption ensures that the set of participants is finite.

The projection of a global type onto participants is given in Figure 2. 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 definition is coinductive, so a global type with an infinite (regular) tree produces a process with a regular tree. The projection of a choice type on the sender produces an output process, i.e. a 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 [39].

Our coinductive definition of global types is more permissive than that based on the standard μ-notation used in [39], because it allows more global types to be projected, as shown by the following example.

Example 6.2. *The global type* $\mathsf{G} = \mathsf{p} \to \mathsf{q} : (\lambda_1; \mathsf{q} \xrightarrow{\lambda_3} \mathsf{r} \boxplus \lambda_2; \mathsf{G})$ *is projectable and*

- $\mathsf{G} \upharpoonright \mathsf{p} \; = P = \mathsf{q}!\lambda_1 \oplus \mathsf{q}!\lambda_2; P$
- $\mathsf{G} \upharpoonright \mathsf{q} \; = Q = \mathsf{p}?\lambda_1; \mathsf{r}!\lambda_3 + \mathsf{p}?\lambda_2; Q$
- $\mathsf{G} \upharpoonright \mathsf{r} \; = \mathsf{q}?\lambda_3$

On the other hand, the corresponding global type based on the μ-notation

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

is not projectable because $\mathsf{G}' \restriction \mathsf{r}$ *is not defined.*

However, this additional permissiveness will not be exploited in the present paper. Indeed, the global type G of Example 6.2 will be ruled out by the condition of boundedness, introduced next, which aims at forbidding starvation. On the other hand, such permissiveness could be of interest whenever starvation is not a concern.

To achieve progress, we need to ensure that each network participant occurs in every computation, whether finite or infinite. This means that each type participant must occur in every path of the tree of the type. Projectability already ensures that each participant of a choice type occurs in all its branches. This implies that if one branch of the choice gives rise to an infinite path, either the participant occurs at some finite depth in this path, or this path crosses infinitely many branching points in which the participant occurs in all branches. In the latter case, since the depth of the participant increases when crossing each branching point, there is no bound on the depth of the participant over all paths of the type. Hence, to ensure that all type participants occur in all paths, it is enough to require the existence of such bounds. This motivates the following definition of depth and boundedness.

Definition 6.3 (Depth and boundedness).
Let the two functions $\mathsf{depth}(\sigma, \mathsf{p})$ *and* $\mathsf{depth}(\mathsf{G}, \mathsf{p})$ *be defined by:*

$$\mathsf{depth}(\sigma, \mathsf{p}) = \begin{cases} n & \text{if } \sigma = \sigma_1 \cdot \alpha \cdot \sigma_2 \text{ and } |\sigma_1| = n-1 \text{ and } \mathsf{p} \notin \mathsf{part}(\sigma_1) \text{ and } \mathsf{p} \in \mathsf{part}(\alpha) \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\mathsf{depth}(\mathsf{G}, \mathsf{p}) = \sup\{\mathsf{depth}(\sigma, \mathsf{p}) \mid \sigma \in \mathsf{Tr}^{+}(\mathsf{G})\}$$

We say that a global type G *is* bounded *if* $\mathsf{depth}(\mathsf{G}', \mathsf{p})$ *is finite for all subtrees* G' *of* G *and for all participants* p.

If $\mathsf{depth}(\mathsf{G}, \mathsf{p})$ is finite, then there are no paths in the tree of G in which p is delayed indefinitely. Note that if $\mathsf{depth}(\mathsf{G}, \mathsf{p})$ is finite, G may have subtrees G' for which $\mathsf{depth}(\mathsf{G}', \mathsf{p})$ is infinite as the following example shows.

Example 6.4. *Consider* $\mathsf{G}' = \mathsf{q} \xrightarrow{\lambda} \mathsf{r}; \mathsf{G}$ *where* G *is as defined in Example 6.2. Then we have:*

$$\mathsf{depth}(\mathsf{G}', \mathsf{p}) = 2 \qquad \mathsf{depth}(\mathsf{G}', \mathsf{q}) = 1 \qquad \mathsf{depth}(\mathsf{G}', \mathsf{r}) = 1$$

whereas

$$\mathsf{depth}(\mathsf{G}, \mathsf{p}) = 1 \qquad \mathsf{depth}(\mathsf{G}, \mathsf{q}) = 1 \qquad \mathsf{depth}(\mathsf{G}, \mathsf{r}) = \infty$$

since

$$\mathsf{Tr}^{+}(\mathsf{G}) = \{\underbrace{\mathsf{pq}\lambda_2 \cdots \mathsf{pq}\lambda_2}_{n} \cdot \mathsf{pq}\lambda_1 \cdot \mathsf{qr}\lambda_3 \mid n \geq 0\} \cup \{\mathsf{pq}\lambda_2 \cdots \mathsf{pq}\lambda_2 \cdots\}$$

and $\sup\{2, 3, \ldots\} = \infty$.

The depths of the participants in G which are not participants of its root communication decrease in the immediate subtrees of G. The proof is trivial since, if $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$, then $\sigma \in \mathsf{Tr}^+(\mathsf{G})$ implies $\sigma = \mathsf{pq}\lambda_i \cdot \sigma'$ and $\sigma' \in \mathsf{Tr}^+(\mathsf{G}_i)$ for some $i \in I$.

Lemma 6.5. *If* $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ *and* $\mathsf{r} \in \mathsf{part}(\mathsf{G})\setminus\{\mathsf{p}, \mathsf{q}\}$*, then* $\mathsf{depth}(\mathsf{G}, \mathsf{r}) > \mathsf{depth}(\mathsf{G}_i, \mathsf{r})$ *for all* $i \in I$.

We can now show that the definition of projection given in Figure 2 is sound for bounded global types.

Lemma 6.6. *If* G *is bounded, then* $\mathsf{G} \upharpoonright \mathsf{r}$ *is a partial function for all* r.

Boundedness and projectability single out the global types we want to use in our type system.

Definition 6.7 (Well-formed global types). *We say that the global type* G *is* well formed *if* G *is bounded and* $\mathsf{G} \upharpoonright \mathsf{p}$ *is defined for all* p.

Clearly it is sufficient to check that $\mathsf{G} \upharpoonright \mathsf{p}$ is defined for all $\mathsf{p} \in \mathsf{part}(\mathsf{G})$, since otherwise $\mathsf{G} \upharpoonright \mathsf{p} = \mathbf{0}$.

6.2. Type System

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

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

Figure 3: Preorder on processes and network typing rule.

The definition of well-typed networks is given in Figure 3. We first define a preorder on processes, $P \leq Q$, meaning that *process P can be used where we expect process Q*. More precisely, $P \leq Q$ if either P is equal to Q, or we are in one of two situations: either both P and Q are output processes with the same receiver and choice of messages, and their continuations after the send are two processes P' and Q' such that $P' \leq Q'$; or they are both input processes with the same sender and choice of messages, and P may receive more messages than Q (and thus have more behaviours) but whenever it receives the same message as Q their continuations are two processes P' and Q' such that $P' \leq Q'$. The rules are interpreted coinductively, since the processes may have infinite (regular) trees.

A network is well typed 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

$$\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i \xrightarrow{\mathsf{pq}\lambda_j} \mathsf{G}_j \qquad j \in I \quad [\textsc{Ecomm}]$$

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

Figure 4: LTS for global types.

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 6.8. *The first network of Example 5.15 and the network of Example 5.17 can be typed respectively by*

$$\begin{array}{rcl} \mathsf{G} & = & \mathsf{p} \to \mathsf{q} : (\lambda; \mathsf{G} \boxplus \lambda') \\ \mathsf{G}' & = & \mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; \mathsf{q} \xrightarrow{\lambda_2} \mathsf{r}; \mathsf{r} \xrightarrow{\lambda_3} \mathsf{s} \end{array}$$

It is handy to define the LTS for global types given in Figure 4. 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 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. This LTS respects well-formedness of global types, as shown by Lemma 6.9.

Lemma 6.9. *If* G *is a well-formed global type and* $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda} \mathsf{G}'$, *then* G' *is a well-formed global type.*

Given this lemma, we will focus on **well-formed global types from now on.**

We end this section with the expected results of Subject Reduction, Session Fidelity [39, 40] and Progress [19, 51]. The proof of Progress relies on Session Fidelity. Both Subject Reduction and Session Fidelity will be used in Section 8 to show the isomorphism between the configuration domains of the FES of a typable network and the PES of its global type (Theorem 8.18).

Theorem 6.10 (Subject Reduction). *If* $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$, *then* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

Theorem 6.11 (Session Fidelity). *If* $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$, *then* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

We are now able to prove that in a typable network, every participant whose process is not terminated may eventually perform a communication. This property is generally referred to as progress.

Theorem 6.12 (Progress). *If* $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{p}[\![\, P \,]\!] \in \mathsf{N}$, *then* $\mathsf{N} \xrightarrow{\sigma\cdot\alpha} \mathsf{N}'$ *and* $\mathsf{p} \in \mathsf{part}(\alpha)$.

Proof We prove by induction on $d = \mathsf{depth}(\mathsf{G},\mathsf{p})$ that: if $\vdash \mathsf{N} : \mathsf{G}$ and $\mathsf{p}[\![P]\!] \in \mathsf{N}$, then $\mathsf{G} \xrightarrow{\sigma\cdot\alpha} \mathsf{G}'$ with $\mathsf{p} \in \mathsf{part}(\alpha)$. This will imply $\mathsf{N} \xrightarrow{\sigma\cdot\alpha} \mathsf{N}'$ by Session Fidelity (Theorem 6.11).

Case $d = 1$. In this case $\mathsf{G} = \mathsf{q} \to \mathsf{r} : \boxplus_{i\in I}\lambda_i;\mathsf{G}_i$ and $\mathsf{p} \in \{\mathsf{q},\mathsf{r}\}$ and $\mathsf{G} \xrightarrow{\mathsf{qr}\lambda_h} \mathsf{G}_h$ for some $h \in I$ by Rule [EComm].

Case $d > 1$. In this case $\mathsf{G} = \mathsf{q} \to \mathsf{r} : \boxplus_{i\in I}\lambda_i;\mathsf{G}_i$ and $\mathsf{p} \notin \{\mathsf{q},\mathsf{r}\}$. By Lemma 6.5 this implies $\mathsf{depth}(\mathsf{G}_i,\mathsf{p}) < d$ for all $i \in I$. Using Rule [EComm] we get $\mathsf{G} \xrightarrow{\mathsf{qr}\lambda_i} \mathsf{G}_i$ for all $i \in I$. By Session Fidelity, $\mathsf{N} \xrightarrow{\mathsf{qr}\lambda_i} \mathsf{N}_i$ and $\vdash \mathsf{N}_i : \mathsf{G}_i$ for all $i \in I$. Moreover, since $\mathsf{p} \notin \{\mathsf{q},\mathsf{r}\}$ we also have $\mathsf{p}[\![P]\!] \in \mathsf{N}_i$ for all $i \in I$. By induction $\mathsf{G}_i \xrightarrow{\sigma_i\cdot\alpha_i} \mathsf{G}'_i$ with $\mathsf{p} \in \mathsf{part}(\alpha_i)$ for all $i \in I$. We conclude $\mathsf{G} \xrightarrow{\mathsf{qr}\lambda_i\cdot\sigma_i\cdot\alpha_i} \mathsf{G}'_i$ for all $i \in I$.

The proof of the progress theorem shows that the execution strategy which uses only Rule [EComm] is fair, since there are no infinite transition sequences where some participant is stuck. This is due to the boundedness condition on global types.

Example 6.13. *The second network of Example 5.15 and the network of Example 5.20 cannot be typed because they do not enjoy progress. Notice that the candidate global type for the second network of Example 5.15:*

$$\mathsf{G}'' = \mathsf{p} \to \mathsf{q} : (\lambda;\mathsf{G}'' \boxplus \lambda';\mathsf{p} \xrightarrow{\lambda} \mathsf{r};\mathsf{r} \xrightarrow{\lambda'} \mathsf{s})$$

is not bounded, given that $\mathsf{depth}(\mathsf{G}'',\mathsf{r})$ *and* $\mathsf{depth}(\mathsf{G}'',\mathsf{s})$ *are not finite.*
Moreover we cannot define a global type whose projections are greater than or equal to the processes associated with the network of Example 5.20.

7. Event Structure Semantics of Global Types

We define now the event structure associated with a global type, whose events are equivalence classes of particular traces, and we show that it is a PES.

The unique omitted proof can be found in Appendix C.

We recall that a trace $\sigma \in$ *Traces* is a finite sequence of communications (see Definition 2.3). We will use the following notational conventions:

- We denote by $\sigma[i]$ the i-th element of σ, $i > 0$.
- If $i \leq j$, we define $\sigma[i\,...\,j] = \sigma[i]\cdots\sigma[j]$ to be the subtrace of σ consisting of the $(j - i + 1)$ elements starting from the i-th one and ending with the j-th one. If $i > j$, we convene $\sigma[i\,...\,j]$ to be the empty trace ϵ.

If not otherwise stated we assume that σ has n elements, so $\sigma = \sigma[1\,...\,n]$.

We start by defining an equivalence relation on *Traces* which allows swapping of communications with disjoint participants.

Definition 7.1 (Permutation equivalence). *The permutation equivalence on Traces is the least equivalence $\sim$ such that*

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

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

The events associated with a global type, called *g-events* and denoted by γ, γ', are equivalence classes of particular traces that we call *pointed*. Intuitively, in a pointed trace all communications but the last one are causes of some subsequent communication. Formally:

Definition 7.2 (Pointed trace). *A trace $\sigma = \sigma[1 \ldots n]$ is said to be* pointed *if*

$$\textit{for all } i,\, 1 \leq i < n,\; \mathsf{part}(\sigma[i]) \cap \mathsf{part}(\sigma[(i+1) \ldots n]) \neq \emptyset$$

Note that the condition of Definition 7.2 must be satisfied only by the $\sigma[i]$ with $i < n$, thus it is vacuously satisfied by any trace of length 1.

Example 7.3. *Let $\alpha_1 = \mathsf{pq}\lambda_1$, $\alpha_2 = \mathsf{rs}\lambda_2$ and $\alpha_3 = \mathsf{rp}\lambda_3$. Then $\sigma_1 = \alpha_1$ and $\sigma_3 = \alpha_1 \cdot \alpha_2 \cdot \alpha_3$ are pointed traces, while $\sigma_2 = \alpha_1 \cdot \alpha_2$ is not a pointed trace.*

We use $\mathsf{last}(\sigma)$ to denote the last communication of σ.

Lemma 7.4. *Let σ be a pointed trace. If $\sigma \sim \sigma'$, then σ' is a pointed trace and $\mathsf{last}(\sigma) = \mathsf{last}(\sigma')$.*

Definition 7.5 (Global event). *Let $\sigma = \sigma' \cdot \alpha$ be a pointed trace. Then $\gamma = [\sigma]_\sim$ is a* global event*, also called* g-event*, with communication α, notation $\mathsf{cm}(\gamma) = \alpha$. We denote by $\mathcal{GE}$ the set of g-events.*

Notice that $\mathsf{cm}(\gamma)$ is well defined due to Lemma 7.4.

We now introduce an operator of prefixing of a g-event γ by a communication α, which acts as follows: if α is a cause of some communication in the trace of γ, then α is added at the beginning of the trace, otherwise γ is left unchanged. This ensures that the operator always transforms a g-event into another g-event. We call this operator "retrieval of a g-event before a communication", because it yields the g-event obtained from γ if we were to execute the communication α before γ. This operator is the counterpart of the "residual of a g-event after a communication", which yields the g-event obtained from γ after executing the communication α from γ, see Definition 8.9.

Definition 7.6 (Retrieval of g-events before communications).

1. *The* retrieval operator $\circ$ *applied to a communication and a g-event is defined by:*

$$\alpha \circ [\sigma]_\sim = \begin{cases} [\alpha \cdot \sigma]_\sim & \textit{if } \mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset \\ [\sigma]_\sim & \textit{otherwise} \end{cases}$$

2. *The operator* $\circ$ *naturally extends to traces:*

$$\epsilon \circ \gamma = \gamma \qquad (\alpha \cdot \sigma) \circ \gamma = \alpha \circ (\sigma \circ \gamma)$$

Using the retrieval, we can define the mapping $\mathsf{ev}(\cdot)$ which, applied to a trace σ, gives the g-event representing the communication $\mathsf{last}(\sigma)$ prefixed by its causes occurring in σ.

Definition 7.7. *The* g-event generated by a non-empty trace *is defined by:*

$$\mathsf{ev}(\sigma \cdot \alpha) = \sigma \circ [\alpha]_\sim$$

Clearly $\mathsf{cm}(\mathsf{ev}(\sigma)) = \mathsf{last}(\sigma)$.

Example 7.8. *A trace of the global type* $\mathsf{p} \xrightarrow{\lambda_1} \mathsf{q}; \mathsf{q} \xrightarrow{\lambda_2} \mathsf{r}; \mathsf{s} \xrightarrow{\lambda_3} \mathsf{p}$ *is* $\mathsf{pq}\lambda_1 \cdot \mathsf{qr}\lambda_2 \cdot \mathsf{sp}\lambda_3$*, and*

$$\mathsf{ev}(\mathsf{pq}\lambda_1 \cdot \mathsf{qr}\lambda_2 \cdot \mathsf{sp}\lambda_3) = \mathsf{pq}\lambda_1 \cdot \mathsf{qr}\lambda_2 \circ \{\mathsf{sp}\lambda_3\} = \mathsf{pq}\lambda_1 \circ \{\mathsf{sp}\lambda_3\} = \{\mathsf{pq}\lambda_1 \cdot \mathsf{sp}\lambda_3\}$$

We proceed now to define the causality and conflict relations on g-events. To define the conflict relation, it is handy to define the projection of a trace on a participant, which gives the sequence of the participant's actions in the trace. The result is a p-event. In this way we can define the conflict between g-events using the conflict between p-events.

Definition 7.9 (Projection of traces on participants).

1. *The* projection *of* α *onto* r, $\alpha@\mathsf{r}$*, is defined by:*

$$\mathsf{pq}\lambda@\mathsf{r} = \begin{cases} \mathsf{q}!\lambda & \textit{if } \mathsf{r} = \mathsf{p} \\ \mathsf{p}?\lambda & \textit{if } \mathsf{r} = \mathsf{q} \\ \epsilon & \textit{if } \mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\} \end{cases}$$

2. *The projection of a trace* σ *onto* r, $\sigma@\mathsf{r}$*, is defined by:*

$$\epsilon@\mathsf{r} = \epsilon \qquad (\alpha \cdot \sigma)@\mathsf{r} = \alpha@\mathsf{r} \cdot \sigma@\mathsf{r}$$

Definition 7.10 (Causality and conflict relations on g-events). *The* causality *relation* $\leq$ *and the* conflict *relation* $\#$ *on the set of g-events* $\mathcal{GE}$ *are defined by:*

1. $\gamma \leq \gamma'$ *if* $\gamma = [\sigma]_\sim$ *and* $\gamma' = [\sigma \cdot \sigma']_\sim$ *for some* σ, σ'*;*

2. $[\sigma]_\sim \# [\sigma']_\sim$ *if* $\sigma@\mathsf{p} \# \sigma'@\mathsf{p}$ *for some* p.

If $\gamma = [\sigma \cdot \alpha \cdot \sigma' \cdot \alpha']_\sim$, then the communication α must be done before the communication α'. This is expressed by the causality $[\sigma \cdot \alpha]_\sim \leq \gamma$. An example is $[\mathsf{pq}\lambda]_\sim \leq [\mathsf{rs}\lambda' \cdot \mathsf{pq}\lambda \cdot \mathsf{sq}\lambda'']_\sim$.
As regards conflict, note that if $\sigma \sim \sigma'$ then $\sigma@\mathsf{p} = \sigma'@\mathsf{p}$ for all p, because $\sim$ does not swap communications which share some participant. Hence, conflict is well defined, since it does not depend on the trace chosen in the equivalence class. The condition $\sigma@\mathsf{p} \# \sigma'@\mathsf{p}$ states that participant p does the same actions in both traces up to some point, after which it performs two different actions in σ and σ'. For example $[\mathsf{pq}\lambda \cdot \mathsf{rp}\lambda_1 \cdot \mathsf{qp}\lambda']_\sim \# [\mathsf{pq}\lambda \cdot \mathsf{rp}\lambda_2]_\sim$, since $(\mathsf{pq}\lambda \cdot \mathsf{rp}\lambda_1 \cdot \mathsf{qp}\lambda')@\mathsf{p} = \mathsf{q}!\lambda \cdot \mathsf{r}?\lambda_1 \cdot \mathsf{q}?\lambda' \# \mathsf{q}!\lambda \cdot \mathsf{r}?\lambda_2 = (\mathsf{pq}\lambda \cdot \mathsf{rp}\lambda_2)@\mathsf{p}$.

Definition 7.11 (Event structure of a global type). *The* event structure of the global type G *is the triple*

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

where:

1. $\mathcal{GE}(\mathsf{G}) = \{\mathsf{ev}(\sigma) \mid \sigma \in \mathsf{Tr}^{+}(\mathsf{G})\}$
2. $\leq_{\mathsf{G}}$ *is the restriction of* $\leq$ *to the set* $\mathcal{GE}(\mathsf{G})$*;*
3. $\#_{\mathsf{G}}$ *is the restriction of* $\#$ *to the set* $\mathcal{GE}(\mathsf{G})$.

Note that, in case the tree of G is infinite, the set $\mathcal{GE}(\mathsf{G})$ is denumerable.

Example 7.12. *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*

$$\gamma_1 = \{\mathsf{pq}\lambda_1\} \qquad \gamma_2 = \{\mathsf{rs}\lambda_2\} \qquad \gamma_3 = \{\mathsf{pq}\lambda_1 \cdot \mathsf{rs}\lambda_2 \cdot \mathsf{rp}\lambda_3, \mathsf{rs}\lambda_2 \cdot \mathsf{pq}\lambda_1 \cdot \mathsf{rp}\lambda_3\}$$

with $\gamma_1 \leq \gamma_3$ *and* $\gamma_2 \leq \gamma_3$. *The configurations are* $\{\gamma_1\}$*,* $\{\gamma_2\}$*,* $\{\gamma_1, \gamma_2\}$ *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 6.8, then* $\mathcal{GE}(\mathsf{G}') = \{\gamma_1, \gamma_2, \gamma_3\}$ *where*

$$\gamma_1 = \{\mathsf{pq}\lambda_1\} \qquad \gamma_2 = \{\mathsf{pq}\lambda_1 \cdot \mathsf{qr}\lambda_2\} \qquad \gamma_3 = \{\mathsf{pq}\lambda_1 \cdot \mathsf{qr}\lambda_2 \cdot \mathsf{rs}\lambda_3\}$$

with $\gamma_1 \leq \gamma_2 \leq \gamma_3$. *The configurations are* $\{\gamma_1\}$*,* $\{\gamma_1, \gamma_2\}$ *and* $\{\gamma_1, \gamma_2, \gamma_3\}$*, and there is a unique proving sequence corresponding to each configuration.*

Theorem 7.13. *Let* G *be a global type. Then* $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$ *is a prime event structure.*

Proof We show that $\leq$ and $\#$ satisfy Properties (2) and (3) of Definition 3.1. Reflexivity and transitivity of $\leq$ follow from the properties of concatenation and of permutation equivalence. As for antisymmetry, by Definition 7.10(1) $[\sigma]_\sim \leq [\sigma']_\sim$ implies $\sigma' \sim \sigma \cdot \sigma_1$ for some σ_1 and $[\sigma']_\sim \leq [\sigma]_\sim$ implies $\sigma \sim \sigma' \cdot \sigma_2$ for some σ_2. Hence $\sigma \sim \sigma \cdot \sigma_1 \cdot \sigma_2$, which implies $\sigma_1 = \sigma_2 = \epsilon$. Irreflexivity and symmetry of $\#$ follow from the corresponding properties of $\#$ on p-events.
As for conflict hereditariness, suppose that $[\sigma]_\sim \# [\sigma']_\sim \leq [\sigma'']_\sim$. By Definition 7.10(1) and (2) we have respectively that $\sigma' \cdot \sigma_1 \sim \sigma''$ for some σ_1 and $\sigma @ \mathsf{p} \# \sigma' @ \mathsf{p}$ for some p. Hence also $\sigma @ \mathsf{p} \# (\sigma' \cdot \sigma_1) @ \mathsf{p}$, whence by Definition 7.10(2) we conclude that $[\sigma]_\sim \# [\sigma'']_\sim$.

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.

We conclude this section with two pictures that summarise the features of our ES semantics and illustrate the difference between the FES of a network and the

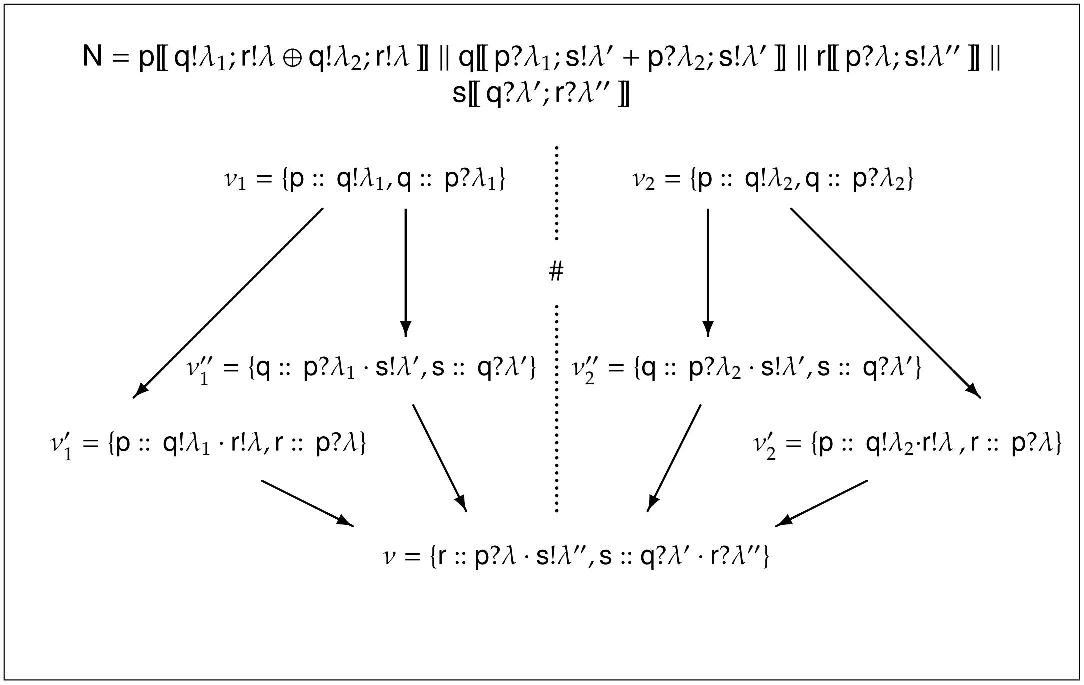

Figure 5: FES of the network N.

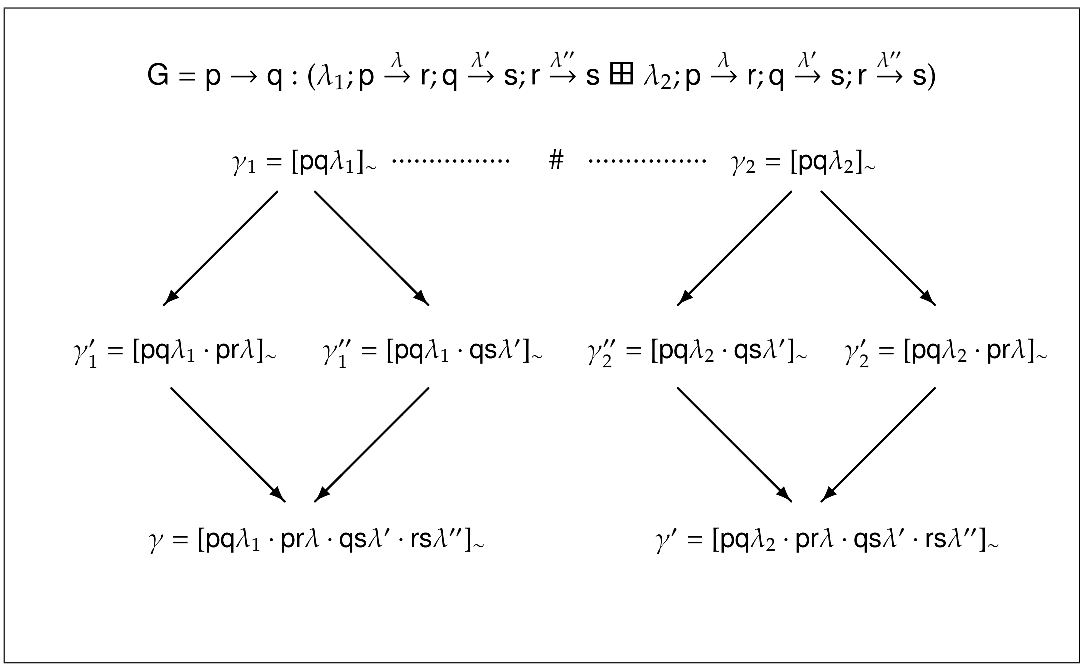

Figure 6: PES of the type G.

PES of its type. In general these two ESs are not isomorphic, unless the FES of the network is itself a PES.

Consider the network FES pictured in Figure 5, where the arrows represent the flow relation and all the n-events on the left of the dotted line are in conflict with all the n-events on the right of the line. In particular, notice that the conflicts between n-events with a common location are deduced by Clause (2a) of Definition 5.7, while the conflicts between n-events with disjoint sets of locations, such as ν_1' and ν_2'', are deduced by Clause (2b) of Definition 5.7. Observe also that the n-event ν has two different causal sets in $\mathcal{NE}(\mathsf{N})$, namely $\{\nu_1', \nu_1''\}$ and $\{\nu_2', \nu_2''\}$. The reader familiar with ESs will have noticed that there are also two prime configurations whose maximal element is ν, namely $\{\nu_1, \nu_1', \nu_1'', \nu\}$ and $\{\nu_2, \nu_2', \nu_2'', \nu\}$. It is easy to see that the network N can be typed with the global type G shown in Figure 6.

Consider now the PES of the type G pictured in Figure 6, where the arrows represent the covering relation of the partial order of causality and inherited conflicts are not shown. Note that while the FES of N has a unique maximal n-event ν, the PES of its type G has two maximal g-events γ and γ'. This is because an n-event only records the computations that occurred at its locations, while a g-event records the global computation and keeps a record of each choice, including those involving locations that are disjoint from those of its last communication. Indeed, g-events correspond exactly to prime configurations.

Note that the FES of a network may be easily recovered from the PES of its global type by using the following function $\mathsf{gn}(\cdot)$ that maps g-events to n-events:

$$\mathsf{gn}(\gamma) = \{\mathsf{p} :: \sigma@\mathsf{p}\,, \mathsf{q} :: \sigma@\mathsf{q}\,\} \quad \text{if } \gamma = [\sigma]_\sim \text{ with } \mathsf{part}(\mathsf{cm}(\gamma)) = \{\mathsf{p}, \mathsf{q}\}$$

On the other hand, the inverse construction is not as direct. First of all, an n-event in the network FES may give rise to several g-events in the type PES, as shown by the n-event ν in Figure 5, which gives rise to the pair of g-events γ and γ' in Figure 6. Moreover, the local information contained in an n-event is not sufficient to reconstruct the corresponding g-events: for each n-event, we need to consider all the prime configurations that culminate with that event, and then map each of these configurations to a g-event. Hence, we need a function $\mathsf{ng}(\cdot)$ that maps n-events to sets of prime configurations of the FES, and then maps each such configuration to a g-event. We will not explicitly define this function here, since we miss another important ingredient to compare the FES of a network and the PES of its type, namely a structural characterisation of the FESs that represent typable networks. Indeed, if we started from the FES of a non typable network, this construction would not be correct. Consider for instance the network N' obtained from N by omitting the output $\mathsf{r}!\lambda$ from the second branch of the process of p. Then the FES of N' would not contain the n-event ν_2' and the event ν would have the unique causal set $\{\nu_1', \nu_1''\}$, and the unique prime configuration culminating with ν would be $\{\nu_1, \nu_1', \nu_1'', \nu\}$. Then our construction would give a PES that differs from that of type G only for the absence of the g-events γ_2' and γ'. However, the network N' is not typable and thus we would expect the construction to fail. Note that in the FES of N', the n-event ν_2'' is a cause of ν but does not belong to any causal set of ν. Thus a possible well-formedness property to require for FESs to be images of a typable network would be that each cause of each n-event belong to some causal set of that event. However, this would still not be enough to exclude the FES of the non typable network N'' obtained from N' by omitting the output $\mathsf{s}!\lambda'$ from the second branch of the process of q.

To conclude, in the absence of a semantic counterpart for the well-formedness properties of global types, which eludes us for the time being, we will follow another approach here, namely we will compare the FESs of networks and the PESs of their types at a more operational level, by looking at their configuration domains and by relating their configurations to the transition sequences of the underlying networks and types.

8. Equivalence of the two Event Structure Semantics

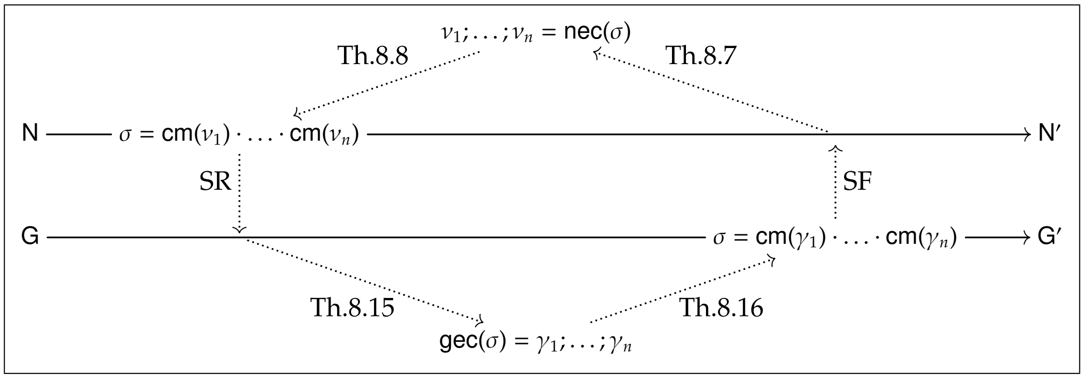

Figure 7: Isomorphism proof in a nutshell.

In this section we establish our main result for typable networks (Theorem 8.18), namely the isomorphism between the domain of configurations of the FES of a typable network and the domain of configurations of the PES of its global type. To do so, we first relate the transition sequences of networks and global types to the configurations of their respective ESs. Then, we exploit our results of Subject Reduction (Theorem 6.10) and Session Fidelity (Theorem 6.11), which relate the transition sequences of networks and their global types, to derive a similar relation between the configurations of their respective ESs. The schema of our proof is described by the diagram in Figure 7. Here, SR stands for Subject Reduction and SF for Session Fidelity, and $\nu_1; \ldots ; \nu_n$ and $\gamma_1; \ldots ; \gamma_n$ are proving sequences of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ and $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$, respectively. Finally, $\mathsf{nec}(\sigma)$ and $\mathsf{gec}(\sigma)$ denote the proving sequences of n-events and g-events which correspond to the trace σ (as given by Definition 8.3 and Definition 8.13). Theorem 8.8 says that, if $\nu_1; \cdots ; \nu_n$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$, then $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$, where $\sigma = \mathsf{cm}(\nu_1) \cdot \ldots \cdot \mathsf{cm}(\nu_n)$. By Subject Reduction (Theorem 6.10) $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$. This implies that $\mathsf{gec}(\sigma)$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$ by Theorem 8.15. Dually, Theorem 8.16 says that, 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 $\sigma = \mathsf{cm}(\gamma_1) \cdot \ldots \cdot \mathsf{cm}(\gamma_n)$. By Session Fidelity (Theorem 6.11) $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$. Lastly, $\mathsf{nec}(\sigma)$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ by Theorem 8.7. The equalities in the top and bottom lines are proved in Lemmas 8.4(1a) and 8.14(1).

This section is divided in two subsections: Section 8.1, which handles the upper part of the above diagram, and Section 8.2, which handles the lower part of the diagram and then connects the two parts using both SR and SF within Theorem 8.18, our closing result. The omitted proofs of Sections 8.1 and 8.2 can be found in Appendices D and E, respectively.

8.1. Relating Transition Sequences of Networks and Proving Sequences of their ESs

The aim of this subsection is to relate the traces that label the transition sequences of networks with the configurations of their FESs. We start by showing how network communications affect n-events in the associated FES. To this end we define two partial operators $\lozenge$ and $\blacklozenge$, which applied to a communication α and an n-event ν yield another n-event ν' (when defined), which represents the event ν before the communication α or after the communication α, respectively. We call "retrieval" the $\lozenge$ operator (in agreement with Definition 7.6) and "residual" the $\blacklozenge$ operator.

Formally, the operators $\lozenge$ and $\blacklozenge$ are defined as follows.

Definition 8.1 (Retrieval and residual of n-events with respect to communications).

1. *The* retrieval operator $\lozenge$ *applied to a communication and a located event returns the located event obtained by prefixing the p-event by the projection of the communication:*

$$\alpha \lozenge (\mathsf{p} :: \eta) = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta$$

2. *The* residual operator $\blacklozenge$ *applied to a communication and a located event returns the located event obtained by erasing from the p-event the projection of the communication (if possible):*

$$\alpha \blacklozenge (\mathsf{p} :: \eta) = \mathsf{p} :: \eta' \quad \text{if } \eta = (\alpha @ \mathsf{p}) \cdot \eta'$$

3. *The operators $\lozenge$ and $\blacklozenge$ naturally extend to n-events and to traces:*

$$\begin{array}{l}\alpha \lozenge (\{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}) = \{\alpha \lozenge (\mathsf{p} :: \eta), \alpha \lozenge (\mathsf{q} :: \eta')\} \\ \alpha \blacklozenge (\{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}) = \{\alpha \blacklozenge (\mathsf{p} :: \eta), \alpha \blacklozenge (\mathsf{q} :: \eta')\}\end{array}$$

$$\begin{array}{ll}\epsilon \lozenge \nu = \nu & (\alpha \cdot \sigma) \lozenge \nu = \alpha \lozenge (\sigma \lozenge \nu) \\ \epsilon \blacklozenge \nu = \nu & (\alpha \cdot \sigma) \blacklozenge \nu = \sigma \blacklozenge (\alpha \blacklozenge \nu)\end{array}$$

Note that the operator $\lozenge$ is always defined. Instead $\mathsf{pq}\lambda \blacklozenge \mathsf{r} :: \eta$ is undefined if $\mathsf{r} \in \{\mathsf{p}, \mathsf{q}\}$ and either η is just one atomic action or $\mathsf{pq}\lambda @ \mathsf{r}$ is not the first atomic action of η. For example $\mathsf{pq}\lambda \blacklozenge \mathsf{p} :: \mathsf{q}!\lambda$ and $\mathsf{pq}\lambda \blacklozenge \mathsf{p} :: \mathsf{q}!\lambda' \cdot \eta$ with $\lambda \neq \lambda'$ are undefined for any η.

The retrieval and residual operators are inverse of each other. Moreover they preserve the flow and conflict relations.

Lemma 8.2 (Properties of retrieval and residual for n-events).

1. *If $\alpha \blacklozenge \nu$ is defined, then $\alpha \lozenge (\alpha \blacklozenge \nu) = \nu$;*
2. *$\alpha \blacklozenge (\alpha \lozenge \nu) = \nu$;*
3. *If $\nu \prec \nu'$, then $\alpha \lozenge \nu \prec \alpha \lozenge \nu'$;*
4. *If $\nu \prec \nu'$ and both $\alpha \blacklozenge \nu$ and $\alpha \blacklozenge \nu'$ are defined, then $\alpha \blacklozenge \nu \prec \alpha \blacklozenge \nu'$;*
5. *If $\nu \# \nu'$, then $\alpha \lozenge \nu \# \alpha \lozenge \nu'$;*
6. *If $\nu \# \nu'$ and both $\alpha \blacklozenge \nu$ and $\alpha \blacklozenge \nu'$ are defined, then $\alpha \blacklozenge \nu \# \alpha \blacklozenge \nu'$;*

7. *If* $\alpha \lozenge \nu \,\#\, \alpha \lozenge \nu'$, *then* $\nu \,\#\, \nu'$.

Starting from the trace $\sigma \neq \epsilon$ that labels a transition sequence in a network, one can reconstruct the corresponding sequence of n-events in its FES. Recall that $\sigma[1 \ldots i]$ is the prefix of length i of σ and $\sigma[i \ldots j]$ is the empty trace if $i > j$.

Definition 8.3 (Building sequences of n-events from traces). *If* σ *is a non-empty trace with* $\sigma[i] = \mathsf{p}_i\mathsf{q}_i\lambda_i$, $1 \leq i \leq n$, *we define the* sequence of n-events corresponding to σ *by*

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

where $\nu_i = \sigma[1 \ldots i-1] \lozenge \{\mathsf{p}_i :: \mathsf{q}_i!\lambda_i, \mathsf{q}_i :: \mathsf{p}_i?\lambda_i\}$ *for* $1 \leq i \leq n$.

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

We show now that σ can be recovered from $\mathsf{nec}(\sigma)$, and that two n-events occurring in $\mathsf{nec}(\sigma)$ cannot be in conflict. Moreover, the n-event obtained by applying nec to a communication cannot be in conflict with the n-event obtained by applying the retrieval to the same communication and an arbitrary n-event.

Lastly, we relate the sequences of n-events generated by two traces one of which is a suffix of the other. Given that the mapping nec is based on the retrieval operator, this relation is naturally expressed using the retrieval and residual operators.

Lemma 8.4 (Properties of $\mathsf{nec}(\cdot)$).

1. *Let* $\mathsf{nec}(\sigma) = \nu_1; \cdots ; \nu_n$. *Then*
   (a) $\mathsf{cm}(\nu_i) = \sigma[i]$ *for all* i, $1 \leq i \leq n$;
   (b) *If* $1 \leq h, k \leq n$, *then* $\neg(\nu_h \,\#\, \nu_k)$.
2. $\neg(\mathsf{nec}(\alpha) \,\#\, \alpha \lozenge \nu)$ *for all* ν.
3. *Let* $\sigma = \alpha \cdot \sigma'$ *and* $\sigma' \neq \epsilon$. *If* $\mathsf{nec}(\sigma) = \nu_1; \cdots ; \nu_n$ *and* $\mathsf{nec}(\sigma') = \nu'_2; \cdots ; \nu'_n$, *then* $\alpha \lozenge \nu'_i = \nu_i$ *and* $\alpha \blacklozenge \nu_i = \nu'_i$ *for all* i, $2 \leq i \leq n$.

Notice that if $\alpha \blacklozenge \nu$ is undefined and ν is an n-event of a network with communication α, then either $\nu = \mathsf{nec}(\alpha)$ or $\nu \,\#\, \mathsf{nec}(\alpha)$.

Lemma 8.5. *If* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ *and* $\nu \in \mathcal{NE}(\mathsf{N})$, *then* $\nu = \mathsf{nec}(\alpha)$ *or* $\nu \,\#\, \mathsf{nec}(\alpha)$ *or* $\alpha \blacklozenge \nu$ *is defined.*

The following lemma, which is technically quite challenging as it involves reasoning about the fixpoint properties of the set of n-events of a network FES (as defined by the narrowing function), relates the sets of n-events of two network FESs, where one network is a one-step derivative of the other, by means of the retrieval and residual operators.

Lemma 8.6. *Let* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$. *Then*

1. $\{\mathsf{nec}(\alpha)\} \cup \{\alpha \lozenge \nu \mid \nu \in \mathcal{NE}(\mathsf{N}')\} \subseteq \mathcal{NE}(\mathsf{N})$;

2. $\{\alpha \blacklozenge \nu \mid \nu \in \mathcal{NE}(\mathsf{N})$ *and* $\alpha \blacklozenge \nu$ *defined*$\} \subseteq \mathcal{NE}(\mathsf{N}')$.

We may now prove the correspondence between the traces labelling the transition sequences of a network and the proving sequences of its FES.

Theorem 8.7. *If* $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$, *then* $\mathsf{nec}(\sigma)$ *is a proving sequence in* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$.

Proof The proof is by induction on σ.
Base case. Let $\sigma = \alpha$. From $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ and Lemma 8.6(1) $\mathsf{nec}(\alpha) \in \mathcal{NE}(\mathsf{N})$. Since $\mathsf{nec}(\alpha)$ has no causes, by Definition 3.6 we conclude that $\mathsf{nec}(\alpha)$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$.
Inductive case. Let $\sigma = \alpha \cdot \sigma'$. From $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$ we get $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'' \xrightarrow{\sigma'} \mathsf{N}'$ for some N''. Let $\mathsf{nec}(\sigma) = \nu_1; \cdots ; \nu_n$ and $\mathsf{nec}(\sigma') = \nu_2'; \cdots ; \nu_n'$. By induction $\mathsf{nec}(\sigma')$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathsf{N}'')$.
We show that $\mathsf{nec}(\sigma)$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$. By Lemma 8.4(1b) $\mathsf{nec}(\sigma')$ is conflict free. By Lemma 8.4(3) $\nu_i = \alpha \lozenge \nu_i'$ for all i, $2 \leq i \leq n$. This implies $\nu_i \in \mathcal{NE}(\mathsf{N})$ for all i, $2 \leq i \leq n$ by Lemma 8.6(1) and $\neg(\nu_1 \,\#\, \nu_j)$ for all i, j, $2 \leq i, j \leq n$ by Lemma 8.2(7). Finally, since $\nu_1 = \mathsf{nec}(\alpha)$, by Lemma 8.4(2) we obtain $\neg(\nu_1 \,\#\, \nu_i)$ for all i, $2 \leq i \leq n$. We conclude that $\mathsf{nec}(\sigma)$ is conflict-free and included in $\mathcal{NE}(\mathsf{N})$.
Let $\nu \in \mathcal{NE}(\mathsf{N})$ and $\nu \prec \nu_k$ for some k, $1 \leq k \leq n$. This implies $k > 1$ since $\mathsf{nec}(\alpha)$ has no causes. Hence $\nu_k = \alpha \lozenge \nu_k'$. By Lemma 8.5, we know that $\nu = \mathsf{nec}(\alpha)$ or $\nu \,\#\, \mathsf{nec}(\alpha)$ or $\alpha \blacklozenge \nu$ is defined. We consider the three cases. Let $\mathsf{part}(\alpha) = \{\mathsf{p}, \mathsf{q}\}$.
Case $\nu = \mathsf{nec}(\alpha)$. In this case we conclude immediately since $\mathsf{nec}(\alpha) = \nu_1$ and $1 < k$.
Case $\nu \,\#\, \mathsf{nec}(\alpha)$. Since $\mathsf{nec}(\alpha) = \nu_1$, if $\nu_1 \prec \nu_k$ we are done. If $\nu_1 \not\prec \nu_k$, then $\mathsf{loc}(\nu_k) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$ otherwise $\nu_1 \,\#\, \nu_k$. We get $\nu_k = \alpha \lozenge \nu_k' = \nu_k'$. Since $\nu \prec \nu_k$, there exists $\mathsf{r} :: \eta \in \nu$ and $\mathsf{r} :: \eta' \in \nu_k = \nu_k'$ such that $\eta < \eta'$, where $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$ because $\mathsf{r} \in \mathsf{loc}(\nu_k)$. Since $\mathsf{nec}(\sigma')$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathsf{N}'')$, by Lemma 5.24 there is $\nu_h' \in \mathcal{NE}(\mathsf{N}'')$ such that $\mathsf{r} :: \eta \in \nu_h'$. Since $\alpha \lozenge \mathsf{r} :: \eta = \mathsf{r} :: \eta$ we get $\mathsf{r} :: \eta \in \nu_h$. This implies $\nu_h \prec \nu_k$, where $\nu_h \,\#\, \nu$ by Lemma 5.21.
Case $\alpha \blacklozenge \nu$ *defined*. We get $\alpha \blacklozenge \nu \prec \nu_k'$ by Lemma 8.2(4). Since $\mathsf{nec}(\sigma')$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathsf{N}'')$, there is $h < k$ such that either $\alpha \blacklozenge \nu = \nu_h'$ or $\alpha \blacklozenge \nu \,\#\, \nu_h' \prec \nu_k'$. In the first case $\nu = \alpha \lozenge (\alpha \blacklozenge \nu) = \alpha \lozenge \nu_h' = \nu_h$ by Lemma 8.2(1). In the second case:

- from $\alpha \blacklozenge \nu \,\#\, \nu_h'$ we get $(\alpha \lozenge (\alpha \blacklozenge \nu)) \,\#\, (\alpha \lozenge \nu_h')$ by Lemma 8.2(5), which implies $\nu \,\#\, \nu_h$ by Lemma 8.2(1), and
- from $\nu_h' \prec \nu_k'$ we get $(\alpha \lozenge \nu_h') \prec (\alpha \lozenge \nu_k')$ by Lemma 8.2(3), namely $\nu_h \prec \nu_k$.

Theorem 8.8. *If* $\nu_1; \cdots ; \nu_n$ *is a proving sequence in* $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$, *then* $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$, *where* $\sigma = \mathsf{cm}(\nu_1) \cdots \mathsf{cm}(\nu_n)$.

Proof The proof is by induction on n.
Case $n = 1$. Let $\nu_1 = \{\mathsf{p} :: \zeta \cdot \mathsf{q}!\lambda, \mathsf{q} :: \zeta' \cdot \mathsf{p}?\lambda\}$. Then $\mathsf{cm}(\nu_1) = \mathsf{pq}\lambda$. We first show that $\zeta = \zeta' = \epsilon$. Assume ad absurdum that $\zeta \neq \epsilon$ or $\zeta' \neq \epsilon$. By narrowing, this implies that there is $\nu \in \mathcal{NE}(\mathsf{N})$ such that $\nu \prec \nu_1$, contradicting the fact that ν_1 is a proving sequence.

By Definition 5.13(1) we have $\mathsf{N} = \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{N}_0$ with $\mathsf{q}!\lambda \in \mathcal{PE}(P)$ and $\mathsf{p}?\lambda \in \mathcal{PE}(Q)$. Whence by Definition 4.3(1) we get $P = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$ and $Q = \sum_{j\in J} \mathsf{p}?\lambda_j; Q_j$ where $\lambda = \lambda_k$ for some $k \in I \cap J$. Therefore

$$\mathsf{N} \xrightarrow{\mathsf{pq}\lambda} \mathsf{p}[\![\, P_k \,]\!] \parallel \mathsf{q}[\![\, Q_k \,]\!] \parallel \mathsf{N}_0$$

Case $n > 1$. Let ν_1 and N be as in the basic case, $\mathsf{N}'' = \mathsf{p}[\![\, P_k \,]\!] \parallel \mathsf{q}[\![\, Q_k \,]\!] \parallel \mathsf{N}_0$ and $\alpha = \mathsf{pq}\lambda$. Since $\nu_1; \cdots ; \nu_n$ is a proving sequence, we have $\neg(\nu_l \,\#\, \nu_{l'})$ for all l, l' such that $1 \le l, l' \le n$. Moreover, for all l, $2 \le l \le n$ we have $\nu_l \neq \nu_1 = \mathsf{nec}(\alpha)$, thus $\alpha \blacklozenge \nu_l$ is defined by Lemma 8.5. Let $\nu'_l = \alpha \blacklozenge \nu_l$ for all l, $2 \le l \le n$, then $\nu'_l \in \mathcal{NE}(\mathsf{N}'')$ by Lemma 8.6(2).
We show that $\nu'_2; \cdots ; \nu'_n$ is a proving sequence in $\mathcal{S}^{\mathcal{N}}(\mathsf{N}'')$. First notice that for all l, $2 \le l \le n$, $\neg(\nu_l \,\#\, \nu_{l'})$ implies $\neg(\nu'_l \,\#\, \nu'_{l'})$ by Lemma 8.2(5) and (1). Let now $\nu < \nu'_h$ for some h, $2 \le h \le n$. By Lemma 8.2(3) and (1) $\alpha \lozenge \nu < \alpha \lozenge (\alpha \blacklozenge \nu_h) = \nu_h$. This implies by Definition 3.6 that there is $h' < h$ such that either $\alpha \lozenge \nu = \nu_{h'}$ or $\alpha \lozenge \nu \,\#\, \nu_{h'} < \nu_h$. Therefore, since ν'_l is defined for all l, $2 \le l \le n$, we get either $\nu = \nu'_{h'}$ by Lemma 8.2(2) or $\nu \,\#\, \nu'_{h'} < \nu'_h$ by Lemma 8.2(6) and (4).
By induction $\mathsf{N}'' \xrightarrow{\sigma'} \mathsf{N}'$ where $\sigma' = \mathsf{cm}(\nu'_2) \cdots \mathsf{cm}(\nu'_n)$. Since $\mathsf{cm}(\nu_l) = \mathsf{cm}(\nu'_l)$ for all l, $2 \le l \le n$ we get $\sigma = \alpha \cdot \sigma'$. Hence $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'' \xrightarrow{\sigma'} \mathsf{N}'$ is the required transition sequence.

8.2. Relating Transition Sequences of Global Types and Proving Sequences of their ESs

In this subsection, we relate the traces that label the transition sequences of global types with the configurations of their PESs. As for n-events, we need retrieval and residual operators for g-events. The first operator was already introduced in Definition 7.6, so we only need to define the second one, which is given next.

Definition 8.9 (Residual of g-events after communications).

1. *The* residual operator $\bullet$ *applied to a communication and a g-event is defined by:*

$$\alpha \bullet [\sigma]_\sim = \begin{cases} [\sigma']_\sim & \text{if } \sigma \sim \alpha \cdot \sigma' \text{ and } \sigma' \neq \epsilon \\ [\sigma]_\sim & \text{if } \mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \emptyset \end{cases}$$

2. *The operator* $\bullet$ *naturally extends to traces:*

$$\epsilon \bullet \gamma = \gamma \qquad\qquad (\alpha \cdot \sigma) \bullet \gamma = \sigma \bullet (\alpha \bullet \gamma)$$

The operator $\bullet$, applied to a communication and a g-event, gives the g-event obtained by erasing the communication, if it occurs in head position (modulo $\sim$) in the given g-event, and leaves the g-event unchanged if its participants are disjoint from those of the communication. Note that the operator $\alpha \bullet [\sigma]_\sim$ is undefined whenever either $[\sigma]_\sim = \{\alpha\}$ or one of the participants of α occurs in σ but the first communication of σ is different from α. For example $\mathsf{pq}\lambda \bullet [\mathsf{pq}\lambda]_\sim$ and $\mathsf{pq}\lambda \bullet [\mathsf{pq}\lambda' \cdot \sigma]_\sim$ with $\lambda \neq \lambda'$ are undefined for any σ.

The following lemma gives some simple properties of the retrieval and residual operators for g-events. The first five statements correspond to those of Lemma 8.2 for n-events. The last three statements give properties that are relevant only for the operators $\circ$ and $\bullet$.

Lemma 8.10 (Properties of retrieval and residual for g-events).

1. *If $\alpha \bullet \gamma$ is defined, then $\alpha \circ (\alpha \bullet \gamma) = \gamma$;*
2. *$\alpha \bullet (\alpha \circ \gamma) = \gamma$;*
3. *If $\gamma_1 < \gamma_2$, then $\alpha \circ \gamma_1 < \alpha \circ \gamma_2$;*
4. *If $\gamma_1 < \gamma_2$ and both $\alpha \bullet \gamma_1$ and $\alpha \bullet \gamma_2$ are defined, then $\alpha \bullet \gamma_1 < \alpha \bullet \gamma_2$;*
5. *If $\gamma_1 \# \gamma_2$, then $\alpha \circ \gamma_1 \# \alpha \circ \gamma_2$;*
6. *If $\gamma < \alpha \circ \gamma'$, then either $\gamma = [\alpha]_\sim$ or $\alpha \bullet \gamma < \gamma'$;*
7. *If $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$, then $\alpha_1 \circ (\alpha_2 \circ \gamma) = \alpha_2 \circ (\alpha_1 \circ \gamma)$;*
8. *If $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$ and both $\alpha_2 \bullet (\alpha_1 \circ \gamma)$, $\alpha_2 \bullet \gamma$ are defined, then $\alpha_1 \circ (\alpha_2 \bullet \gamma) = \alpha_2 \bullet (\alpha_1 \circ \gamma)$.*

The next lemma relates the retrieval and residual operator with the global types in the branches of choices.

Lemma 8.11. *The following hold:*

1. *If $\gamma \in \mathcal{GE}(\mathsf{G})$, then $\mathsf{pq}\lambda \circ \gamma \in \mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i)$, where $\lambda = \lambda_k$ and $\mathsf{G} = \mathsf{G}_k$ for some $k \in I$;*
2. *If $\gamma \in \mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i; \mathsf{G}_i)$ and $\mathsf{pq}\lambda_k \bullet \gamma$ is defined, then $\mathsf{pq}\lambda_k \bullet \gamma \in \mathcal{GE}(\mathsf{G}_k)$, where $k \in I$.*

The following lemma plays the role of Lemma 8.6 for n-events.

Lemma 8.12. *Let $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$.*

1. *If $\gamma \in \mathcal{GE}(\mathsf{G}')$, then $\alpha \circ \gamma \in \mathcal{GE}(\mathsf{G})$;*
2. *If $\gamma \in \mathcal{GE}(\mathsf{G})$ and $\alpha \bullet \gamma$ is defined, then $\alpha \bullet \gamma \in \mathcal{GE}(\mathsf{G}')$.*

Each non-empty trace gives rise to a sequence of g-events, compare with Definition 8.3.

Definition 8.13 (Building sequences of g-events from traces). *We define the* sequence of g-events corresponding to a non-empty trace σ *by*

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

where $\gamma_i = \mathsf{ev}(\sigma[1 \dots i])$ for all i, $1 \le i \le n$.

We show that $\mathsf{gec}(\cdot)$ has similar properties as $\mathsf{nec}(\cdot)$, see Lemma 8.4(1). The proof is straightforward.

Lemma 8.14. *Let $\mathsf{gec}(\sigma) = \gamma_1; \cdots; \gamma_n$.*

1. $\mathsf{cm}(\gamma_i) = \sigma[i]$ *for all* i*,* $1 \le i \le n$*.*

2. *If* $1 \le h, k \le n$*, then* $\neg(\gamma_h \ \#\ \gamma_k)$*;*

We may now prove the correspondence between the traces labelling the transition sequences of a global type and the proving sequences of its PES. Let us stress the difference between the set of traces $\mathsf{Tr}^+(\mathsf{G})$ of a global type G as defined at page 20 and the set of traces that label the transition sequences of G, which is a larger set due to the internal Rule [ICOMM] of the LTS for global types given in Figure 4.

Theorem 8.15. *If* $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$*, then* $\mathsf{gec}(\sigma)$ *is a proving sequence in* $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$*.*

Proof By induction on σ.
Base case. Let $\sigma = \alpha$, then $\mathsf{gec}(\alpha) = [\alpha]_\sim$. We use a further induction on the inference of the transition $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$.
Let $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$, $\mathsf{G}' = \mathsf{G}_h$ and $\alpha = \mathsf{pq}\lambda_h$ for some $h \in I$. By Definition 7.11(1) $[\mathsf{pq}\lambda_h]_\sim \in \mathcal{GE}(\mathsf{G})$.
Let $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}'_i$ and $\mathsf{G}_i \xrightarrow{\alpha} \mathsf{G}'_i$ for all $i \in I$ and $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. By induction $[\alpha]_\sim \in \mathcal{GE}(\mathsf{G}_i)$ for all $i \in I$. By Lemma 8.11(1) $\mathsf{pq}\lambda_i \circ [\alpha]_\sim \in \mathcal{GE}(\mathsf{G})$ for all $i \in I$. By Definition 7.11(1) $\mathsf{pq}\lambda_i \circ [\alpha]_\sim = [\alpha]_\sim$, since $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. We conclude $[\alpha]_\sim \in \mathcal{GE}(\mathsf{G})$.
Inductive case. Let $\sigma = \alpha \cdot \sigma'$ with $\sigma' \neq \epsilon$. From $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$ we get $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'' \xrightarrow{\sigma'} \mathsf{G}'$ for some G''. Let $\mathsf{gec}(\sigma) = \gamma_1; \cdots ; \gamma_n$ and $\mathsf{gec}(\sigma') = \gamma'_2; \cdots ; \gamma'_n$. By induction $\mathsf{gec}(\sigma')$ is a proving sequence in $\mathcal{S}^{\mathcal{G}}(\mathsf{G}'')$. By Definitions 8.13 and 7.6 $\gamma_i = \alpha \circ \gamma'_i$, which implies $\alpha \bullet \gamma_i = \gamma'_i$ by Lemma 8.10(2) for all i, $2 \le i \le n$.
We can show that $\gamma_1 = [\alpha]_\sim \in \mathcal{GE}(\mathsf{G})$ as in the proof of the base case. By Lemma 8.12(1) $\gamma_i \in \mathcal{GE}(\mathsf{G})$ since $\gamma'_i \in \mathcal{GE}(\mathsf{G}'')$ and $\alpha \bullet \gamma_i = \gamma'_i$ for all i, $2 \le i \le n$. We prove that $\mathsf{gec}(\sigma)$ is a proving sequence in $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$. Let $\gamma < \gamma_k$ for some k, $1 \le k \le n$. Note that this implies $k > 1$. Since $\gamma_k = \alpha \circ \gamma'_k$ by Lemma 8.10(6) either $\gamma = [\alpha]_\sim$ or $\alpha \bullet \gamma < \gamma'_h$. If $\gamma = [\alpha]_\sim = \gamma_1$ we are done. Otherwise $\alpha \bullet \gamma \in \mathcal{GE}(\mathsf{G}'')$ by Lemma 8.11(2). Since $\mathsf{gec}(\sigma')$ is a proving sequence in $\mathcal{S}^{\mathcal{G}}(\mathsf{G}'')$, there is $h < k$ such that $\alpha \bullet \gamma = \gamma'_h$ and this implies $\gamma = \alpha \circ (\alpha \bullet \gamma) = \alpha \circ \gamma'_h = \gamma_h$ by Lemma 8.10(1).

Theorem 8.16. *If* $\gamma_1; \cdots ; \gamma_n$ *is a proving sequence in* $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$*, then* $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$*, where* $\sigma = \mathsf{cm}(\gamma_1) \cdot \cdots \cdot \mathsf{cm}(\gamma_n)$*.*

Proof The proof is by induction on the length n of the proving sequence. Let $\mathsf{cm}(\gamma_1) = \alpha$ and $\{\mathsf{p}, \mathsf{q}\} = \mathsf{part}(\alpha)$.
Case $n = 1$. Since γ_1 is the first event of a proving sequence, we have $\gamma_1 = [\alpha]_\sim$. We show this case by induction on $d = \mathsf{depth}(\mathsf{G}, \mathsf{p}) = \mathsf{depth}(\mathsf{G}, \mathsf{q})$.
Case $d = 1$. Let $\alpha = \mathsf{pq}\lambda$ and $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\lambda = \lambda_h$ for some $h \in I$. Then $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}_h$ by rule [ECOMM].
Case $d > 1$. Let $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. By Definition 8.9(1) $\mathsf{rs}\lambda_i \bullet \gamma_1$ is defined for all $i \in I$ since $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. This implies $\mathsf{rs}\lambda_i \bullet \gamma_1 \in \mathcal{GE}(\mathsf{G}_i)$ for all $i \in I$ by Lemma 8.11(2). By induction hypothesis $\mathsf{G}_i \xrightarrow{\alpha} \mathsf{G}'_i$ for all $i \in I$. Then we can apply rule [ICOMM] to derive $\mathsf{G} \xrightarrow{\alpha} \mathsf{r} \to \mathsf{s} : \boxplus_{i\in I}\lambda_i; \mathsf{G}'_i$.

Case $n > 1$. Let $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}''$ be the transition as obtained from the base case. We show that $\alpha \bullet \gamma_j$ is defined for all j, $2 \leq j \leq n$. If $\alpha \bullet \gamma_k$ were undefined for some k, $2 \leq k \leq n$, then by Definition 8.9(1) either $\gamma_k = \gamma_1$ or $\gamma_k = [\sigma]_\sim$ with $\sigma \nsim \alpha \cdot \sigma'$ and $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$. In the second case $\alpha @ \mathsf{p} \# \sigma @ \mathsf{p}$ or $\alpha @ \mathsf{q} \# \sigma @ \mathsf{q}$, which implies $\gamma_k \# \gamma_1$. So both cases are impossible. If $\alpha \bullet \gamma_j$ is defined, by Lemma 8.12(2) we get $\alpha \bullet \gamma_j \in \mathcal{GE}(\mathsf{G}'')$ for all j, $2 \leq j \leq n$.
We show that $\gamma'_2; \cdots ; \gamma'_n$ is a proving sequence in $\mathcal{S}^{\mathcal{G}}(\mathsf{G}'')$ where $\gamma'_j = \alpha \bullet \gamma_j$ for all j, $2 \leq j \leq n$. By Lemma 8.10(1) $\gamma_j = \alpha \circ \gamma'_j$ for all j, $2 \leq j \leq n$. Then by Lemma 8.10(5) no two events in the sequence $\gamma'_2; \cdots ; \gamma'_n$ can be in conflict. Let $\gamma \in \mathcal{GE}(\mathsf{G}'')$ and $\gamma < \gamma'_h$ for some h, $2 \leq h \leq n$. By Lemma 8.12(1) $\alpha \circ \gamma$ and $\alpha \circ \gamma'_h$ belong to $\mathcal{GE}(\mathsf{G})$. By Lemma 8.10(3) $\alpha \circ \gamma < \alpha \circ \gamma'_h$. By Lemma 8.10(1) $\alpha \circ \gamma'_h = \gamma_h$. Let $\gamma' = \alpha \circ \gamma$. Then $\gamma' < \gamma_h$ implies, by Definition 3.6 and the fact that $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$ is a PES, that there is $k < h$ such that $\gamma' = \gamma_k$. By Lemma 8.10(1) we get $\gamma = \alpha \bullet \gamma' = \alpha \bullet \gamma_k = \gamma'_k$.
Since $\gamma'_2; \cdots ; \gamma'_n$ is a proving sequence in $\mathcal{S}^{\mathcal{G}}(\mathsf{G}'')$, by induction $\mathsf{G}'' \xrightarrow{\sigma'} \mathsf{G}'$ where $\sigma' = \mathsf{cm}(\gamma'_2) \cdot \ldots \cdot \mathsf{cm}(\gamma'_n)$. Let $\sigma = \mathsf{cm}(\gamma_1) \cdot \ldots \cdot \mathsf{cm}(\gamma_n)$. Since $\mathsf{cm}(\gamma'_j) = \mathsf{cm}(\gamma_j)$ for all $j, 2 \leq j \leq n$, we have $\sigma = \alpha \cdot \sigma'$. Hence $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'' \xrightarrow{\sigma'} \mathsf{G}'$ is the required transition sequence.

The last ingredient required to prove our main theorem is the following separation result from [9] (Lemma 2.8 p. 12):

Lemma 8.17 (Separation [9]). *Let $S = (E, \prec, \#)$ be a flow event structure and $\mathcal{X}, \mathcal{X}' \in \mathcal{C}(S)$ be such that $\mathcal{X} \subset \mathcal{X}'$. Then there exist $e \in \mathcal{X}' \backslash \mathcal{X}$ such that $\mathcal{X} \cup \{e\} \in \mathcal{C}(S)$.*

We may now finally 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 8.18 (Isomorphism). *If* $\vdash \mathsf{N} : \mathsf{G}$, *then* $\mathcal{D}(\mathcal{S}^{\mathcal{N}}(\mathsf{N})) \simeq \mathcal{D}(\mathcal{S}^{\mathcal{G}}(\mathsf{G}))$.

Proof By Theorem 8.8 if $\nu_1; \cdots ; \nu_n$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$, then $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$ where $\sigma = \mathsf{cm}(\nu_1) \cdots \mathsf{cm}(\nu_n)$. By applying iteratively Subject Reduction (Theorem 6.10) $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}'$ and $\vdash \mathsf{N}' : \mathsf{G}'$. By Theorem 8.15 $\mathsf{gec}(\sigma)$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$.

By Theorem 8.16 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 $\sigma = \mathsf{cm}(\gamma_1) \cdots \mathsf{cm}(\gamma_n)$. By applying iteratively Session Fidelity (Theorem 6.11) $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}'$ and $\vdash \mathsf{N}' : \mathsf{G}'$. By Theorem 8.7 $\mathsf{nec}(\sigma)$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$.

Therefore we have a bijection between $\mathcal{D}(\mathcal{S}^{\mathcal{N}}(\mathsf{N}))$ 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 N and G.

We show now that this bijection preserves inclusion of configurations. By Lemma 8.17 it is enough to prove that if $\nu_1; \cdots ; \nu_n \in \mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathsf{N}))$ is mapped to $\gamma_1; \cdots ; \gamma_n \in \mathcal{C}(\mathcal{S}^{\mathcal{G}}(\mathsf{G}))$, then $\nu_1; \cdots ; \nu_n; \nu \in \mathcal{C}(\mathcal{S}^{\mathcal{N}}(\mathsf{N}))$ 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. I.e. let $\mathsf{nec}(\sigma \cdot \alpha) = \nu_1; \cdots ; \nu_n; \nu$ and $\mathsf{gec}(\sigma \cdot \alpha) = \gamma_1; \cdots ; \gamma_n; \gamma$. This implies $\sigma = \mathsf{cm}(\nu_1) \cdots \mathsf{cm}(\nu_n) = \mathsf{cm}(\gamma_1) \cdots \mathsf{cm}(\gamma_n)$ and $\alpha = \mathsf{cm}(\nu) = \mathsf{cm}(\gamma)$ by Lemmas 8.4 and 8.14.

By Theorem 8.8, if $\nu_1; \cdots; \nu_n; \nu$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$, then $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}_0 \xrightarrow{\alpha} \mathsf{N}'$. By applying iteratively Subject Reduction (Theorem 6.10) $\mathsf{G} \xrightarrow{\sigma} \mathsf{G}_0 \xrightarrow{\alpha} \mathsf{G}'$ and $\vdash \mathsf{N}' : \mathsf{G}'$. By Theorem 8.15 $\mathsf{gec}(\sigma \cdot \alpha)$ is a proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$.

By Theorem 8.16, 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 applying iteratively Session Fidelity (Theorem 6.11) $\mathsf{N} \xrightarrow{\sigma} \mathsf{N}_0 \xrightarrow{\alpha} \mathsf{N}'$ and $\vdash \mathsf{N}' : \mathsf{G}'$. By Theorem 8.7 $\mathsf{nec}(\sigma \cdot \alpha)$ is a proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$.

9. Related Work and Conclusions

Event Structures (ESs) were introduced in Winskel's PhD Thesis [60] and in the seminal paper by Nielsen, Plotkin and Winskel [49], roughly in the same frame of time as Milner's calculus CCS [47]. 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 [61]. 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 these forking computations 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 [29, 27, 28], were both shown to agree with an interpretation of CCS processes into some class of ESs (PESs for [27, 28], 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, 50]), proposed by Goltz and Loogen [46] and later generalised by Baier and Majster-Cederbaum [2], and the Bundle ES semantics for LOTOS, proposed by Langerak [45] and extended by Katoen [43]. 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, 26, 34], were subsequently proposed by Goltz and van Glabbeek [57]. 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 [21, 58, 22] and by Cristescu, Krivine and Varacca [23, 24, 25]. Previously, other causal models for the π-calculus had already been put forward by Jategaonkar and Jagadeesan [42], by Montanari and Pistore [48], by Cattani and Sewell [18] 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 early work on noninterleaving operational semantics for the π-calculus, respectively by Boreale and Sangiorgi [6] and by Degano and Priami [30]. 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 [30] 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 [22] 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.

- The calculus considered in the present paper uses synchronous communication - rather than asynchronous, buffered communication - because this is how communication is classically modelled in ESs, when they are used to give semantics to process calculi. We should mention however that after first proposing the present study in [15], we also considered a calculus with asynchronous communication in the companion paper [16]. In that work too, networks are interpreted as FESs, and their associated global types, which we called *asynchronous types* as they split communications into outputs and inputs, are interpreted as PESs. The key result is again an isomorphism between the configuration domain of the FES of a typed network and that of the PES of its type.

- 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 [39]. This is essentially a simplifying assumption, and we do not foresee any difficulty in extending our results to a more general choice operator, where the projection is rendered more flexible through the use of a merge operator [31].

- As regards the preorder on processes, which is akin to a subtyping relation, we envisaged to use the standard subtyping, in which a process with fewer outputs can be used in place of a process with more outputs. However, in that case Session Fidelity would become weaker, since a transition in the LTS of a global type would only ensure a transition in the LTS of the corresponding network, but not necessarily with the same labelling communication. The main drawback would be that Theorem 8.18 would no longer hold: more precisely, the domains of network configurations would only be embedded in (and not isomorphic to) the domains of their global type configurations. Notably, typability is independent from the use of our preorder or of the standard one, as proved in [4].

As regards future work, we plan to define an asynchronous transition system (ATS) [5] 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 [53, 25, 36, 37, 35].

As mentioned at the end of Section 7, the quest for a semantic counterpart of our well-formedness conditions on global types – namely, for properties that characterise the FESs obtained from typable networks – is still open. By way of comparison, such semantic well-formedness conditions have been proposed in [56] for *graphical choreographies*, a truly concurrent graphical model for global specifications with two kinds of forking nodes, representing respectively choice and parallel composition. In [56], those well-formedness conditions, called *well-sequencing* and *well-branchedness*, were shown to be sufficient to ensure projectability on local specifications. In our case, the property corresponding to well-sequencing is automatically ensured by our ES semantics, and we conjecture that the well-branchedness condition for choice nodes (corresponding to projectability) could amount in our simpler setting[10] to the following semantic condition:

Let $\nu_1, \nu_2 \in \mathcal{NE}(N)$ and $\mathsf{p} :: \zeta \cdot \pi \in \nu_1$ and $\mathsf{p} :: \zeta \cdot \pi' \in \nu_2$ with $\pi \neq \pi'$ and $\mathsf{q} = \mathsf{pt}(\pi) = \mathsf{pt}(\pi')$. If $\nu_1 \prec^* \nu'_1$ for some $\nu'_1 \in \mathcal{NE}(N)$ such that $\mathsf{r} \in \mathsf{loc}(\nu'_1)$ with $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$, then $\nu_2 \prec^* \nu'_2$ for some $\nu'_2 \in \mathcal{NE}(N)$ such that $\mathsf{r} \in \mathsf{loc}(\nu'_2)$.

This condition would allow us to rule out the FESs of both networks N' and N'' discussed at the end of Section 7. However, it should be completed with a condition corresponding to boundedness, and the conjunction of these two conditions might

[10]Our choice operator for global types is less general than that of [56].

still not be sufficient in general to ensure typability. We plan to further investigate this question in the near future.

Acknowledgment. We are strongly indebted to the anonymous referees for their constructive remarks, which helped us improve the submitted version of the paper.

References

[1] Luca Aceto and Matthew Hennessy. Towards action-refinement in process algebras. In Albert R. Meyer, editor, *LICS*, pages 138–145, Washington, 1989. IEEE Computer Society Press.

[2] Christel Baier and Mila E. Majster-Cederbaum. The connection between an event structure semantics and an operational semantics for TCSP. *Acta Informatica*, 31(1):81–104, 1994.

[3] Paolo Baldan, Andrea Corradini, and Fabio Gadducci. Domains and event structures for fusions. In Joel Ouaknine, editor, *LICS*, pages 1–12, Washington, 2017. IEEE Computer Society Press.

[4] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Emilio Tuosto. Composition and decomposition of multiparty sessions. *Journal of Logical and Algebraic Methods in Programming*, 119:100620, 2021.

[5] Marek Bednarczyk. *Categories of Asynchronous Systems*. PhD thesis, University of Sussex, 1988.

[6] Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Informatica*, 35(5):353–400, 1998.

[7] Gérard Boudol and Ilaria Castellani. On the semantics of concurrency: partial orders and transition systems. In Hartmut Ehrig, Robert A. Kowalski, Giorgio Levi, and Ugo Montanari, editors, *TAPSOFT*, volume 249 of *LNCS*, pages 123–137, Heidelberg, 1987. Springer.

[8] Gérard Boudol and Ilaria Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In Jaco W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 411–427, Heidelberg, 1988. Springer.

[9] Gérard Boudol and Ilaria Castellani. Flow models of distributed computations: event structures and nets. Research Report 1482, INRIA, 1991.

[10] Gérard Boudol and Ilaria Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Information and Computation*, 114(2):247–314, 1994.

[11] Stephen Brookes, Charles A.R. Hoare, and Andrew Roscoe. A theory of communicating sequential processes. *Journal of ACM*, 31(3):560–599, 1984.

[12] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Event structure semantics for nominal calculi. In Christel Baier and Holger Hermanns, editors, *CONCUR*, volume 4137 of *LNCS*, pages 295–309, Heidelberg, 2006. Springer.

[13] Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *LNCS*, pages 222–236, Heidelberg, 2010. Springer.

[14] Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *Mathematical Structures in Computer Science*, 26(3):367–423, 2016.

[15] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. Event structure semantics for multiparty sessions. In Michele Boreale, Flavio Corradini, Michele Loreti, and Rosario Pugliese, editors, *Models, Languages, and Tools for Concurrent and Distributed Programming - Essays Dedicated to Rocco De Nicola on the Occasion of His 65th Birthday*, volume 11665 of *LNCS*, pages 340–363, Heidelberg, 2019. Springer.

[16] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. Global types and event structure semantics for asynchronous multiparty sessions. *CoRR*, abs/2102.00865, 2021.

[17] Ilaria Castellani and Guo Qiang Zhang. Parallel product of event structures. *Theoretical Computer Science*, 179(1-2):203–215, 1997.

[18] Gian Luca Cattani and Peter Sewell. Models for name-passing processes: interleaving and causal. *Information and Computation*, 190(2):136–178, 2004.

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

[20] Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25:95–169, 1983.

[21] Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Compositional event structure semantics for the internal π-calculus. In Luís Caires and Vasco T. Vasconcelos, editors, *CONCUR*, volume 4703 of *LNCS*, pages 317–332, Heidelberg, 2007. Springer.

[22] Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Event structure semantics of parallel extrusion in the π-calculus. In Lars Birkedal, editor, *FOSSACS*, volume 7213 of *LNCS*, pages 225–239, Heidelberg, 2012. Springer.

[23] Ioana Cristescu. *Operational and denotational semantics for the reversible π-calculus*. PhD thesis, University Paris Diderot - Paris 7, 2015.

[24] Ioana Cristescu, Jean Krivine, and Daniele Varacca. Rigid families for CCS and the π-calculus. In Martin Leucker, Camilo Rueda, and Frank D. Valencia, editors, *ICTAC*, volume 9399 of *LNCS*, pages 223–240, Heidelberg, 2015. Springer.

[25] Ioana Cristescu, Jean Krivine, and Daniele Varacca. Rigid families for the reversible π-calculus. In Simon J. Devitt and Ivan Lanese, editors, *Reversible Computation*, volume 9720 of *LNCS*, pages 3–19, Heidelberg, 2016. Springer.

[26] Philippe Darondeau and Pierpaolo Degano. Refinement of actions in event structures and causal trees. *Theoretical Computer Science*, 118(1):21–48, 1993.

[27] Pierpaolo Degano, Rocco De Nicola, and Ugo Montanari. On the consistency of truly concurrent operational and denotational semantics. In Ashok K. Chandra, editor, *LICS*, Washington, 1988. IEEE Computer Society Press Press.

[28] Pierpaolo Degano, Rocco De Nicola, and Ugo Montanari. A partial ordering semantics for CCS. *Theoretical Computer Science*, 75(3):223–262, 1990.

[29] Pierpaolo Degano and Ugo Montanari. Concurrent histories: A basis for observing distributed systems. *Journal of Computer and System Sciences*, 34(2/3):422–461, 1987.

[30] Pierpaolo Degano and Corrado Priami. Non-interleaving semantics for mobile processes. *Theoretical Computer Science*, 216(1-2):237–270, 1999.

[31] Pierre-Malo Deniélou and Nobuko Yoshida. Dynamic multirole session types. In Mooly Sagiv, editor, *POPL*, pages 435–446, New York, 2011. ACM Press.

[32] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata. In Helmut Seidl, editor, *ESOP*, volume 7211 of *LNCS*, pages 194–213, Heidelberg, 2012. Springer.

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

[34] Ursula Goltz, Roberto Gorrieri, and Arend Rensink. Comparing syntactic and semantic action refinement. *Information and Computation*, 125(2):118–143, 1996.

[35] Eva Graversen. *Event Structure Semantics of Reversible Process Calculi*. PhD thesis, Imperial College London, 2021.

[36] Eva Graversen, Iain Phillips, and Nobuko Yoshida. Towards a categorical representation of reversible event structures. *Journal of Logical and Algebraic Methods in Programming*, 104:16–59, 2019.

[37] Eva Graversen, Iain C. C. Phillips, and Nobuko Yoshida. Event structure semantics of (controlled) reversible CCS. *Journal of Logical and Algebraic Methods in Programming*, 121:100686, 2021.

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

[39] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *POPL*, pages 273–284, New York, 2008. ACM Press.

[40] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. *Journal of ACM*, 63(1):9:1–9:67, 2016.

[41] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of session types and behavioural contracts. *ACM Computing Surveys*, 49(1):3:1–3:36, 2016.

[42] Lalita Jategaonkar Jagadeesan and Radha Jagadeesan. Causality and true concurrency: A data-flow analysis of the π-calculus (extended abstract). In Vangalur S. Alagar and Maurice Nivat, editors, *AMAST*, volume 936 of *LNCS*, pages 277–291, Heidelberg, 1995. Springer.

[43] Joost-Pieter Katoen. *Quantitative and qualitative extensions of event structures*. PhD thesis, University of Twente, 1996.

[44] Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From communicating machines to graphical choreographies. In Sriram K. Rajamani and David Walker, editors, *POPL*, pages 221–232, New York, 2015. ACM Press.

[45] Rom Langerak. Bundle event structures: a non-interleaving semantics for LOTOS. In Michael Diaz and Roland Groz, editors, *Formal Description Techniques for Distributed Systems and Communication Protocols*, pages 331–346, Amsterdam, 1993. North-Holland.

[46] Rita Loogen and Ursula Goltz. Modelling nondeterministic concurrent processes with event structures. *Fundamenta Informaticae*, 14(1):39–74, 1991.

[47] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, Heidelberg, 1980.

[48] Ugo Montanari and Marco Pistore. Concurrent semantics for the π-calculus. In Stephen Brookes, Michael Main, Austin Melton, and Michael Mislove, editors, *MFPS*, volume 1 of *ENTCS*, pages 411–429, Oxford, 1995. Elsevier.

[49] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

[50] Ernst-Rüdiger Olderog. TCSP: theory of communicating sequential processes. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *LNCS*, pages 441–465, Heidelberg, 1986. Springer.

[51] Luca Padovani. Type reconstruction for the linear π-calculus with composite regular types. *Logical Methods in Computer Science*, 11(4), 2015.

[52] Jorge A. Pérez, Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logical relations and observational equivalences for session-based concurrency. *Information and Computation*, 239:254–302, 2014.

[53] Iain Phillips and Irek Ulidowski. Reversibility and asymmetric conflict in event structures. *Journal of Logical and Algebraic Methods in Programming*, 84(6):781 – 805, 2015.

[54] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. An interaction-based language and its typing system. In Chris Hankin, editor, *PARLE*, volume 817 of *LNCS*, pages 122–138, Heidelberg, 1994. Springer.

[55] Bernardo Toninho, Luís Caires, and Frank Pfenning. Dependent session types via intuitionistic linear type theory. In Peter Schneider-Kamp and Michael Hanus, editors, *PPDP*, pages 161–172, New York, 2011. ACM Press.

[56] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreographies. *Journal of Logic and Algebraic Methods in Programming*, 95:17–40, 2018.

[57] Rob J. van Glabbeek and Ursula Goltz. Well-behaved flow event structures for parallel composition and action refinement. *Theoretical Computer Science*, 311(1-3):463–478, 2004.

[58] Daniele Varacca and Nobuko Yoshida. Typed event structures and the linear π-calculus. *Theoretical Computer Science*, 411(19):1949–1973, 2010.

[59] Philip Wadler. Propositions as sessions. *Journal of Functional Programming*, 24(2-3):384–418, 2014.

[60] Glynn Winskel. *Events in Computation*. PhD thesis, University of Edinburgh, 1980.

[61] Glynn Winskel. Event structure semantics for CCS and related languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of *LNCS*, pages 561–576, Heidelberg, 1982. Springer.

[62] Glynn Winskel. An introduction to event structures. In Jaco W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 364–397, Heidelberg, 1988. Springer.

Appendices

A. Proofs of Section 5

This section contains the proofs of Lemmas 5.18, 5.21 and 5.24.

Lemma 5.18 *Let ν and ν' be binary n-events with $\mathsf{loc}(\nu) = \mathsf{loc}(\nu')$. Then $\nu \# \nu'$ iff $\mathsf{p} :: \eta \in \nu$ and $\mathsf{p} :: \eta' \in \nu'$ imply $\eta \# \eta'$.*

Proof The "if" direction holds by Definition 5.7(2a). We show the "only-if" direction. First observe that for any n-event $\nu = \{\mathsf{p} :: \eta_1, \mathsf{q} :: \eta_2\}$ the condition $\mathsf{p} :: \eta_1 \ \widehat{\bowtie}\ \mathsf{q} :: \eta_2$ of Definition 5.5 implies $\eta_1 \Rsh \mathsf{q} \bowtie \eta_2 \Rsh \mathsf{p}$ by Definition 5.4, which in turn implies $|\eta_1 \Rsh \mathsf{q}\,| = |\eta_2 \Rsh \mathsf{p}\,|$ by Definition 5.3. If ν is a binary event, we also have $|\eta_1| = |\eta_1 \Rsh \mathsf{q}\,|$ and $|\eta_2| = |\eta_2 \Rsh \mathsf{p}\,|$ by Definition 5.2, since all the actions of η_1 involve q and all the actions of η_2 involve p, and thus the projections do not erase actions. Assume now $\nu' = \{\mathsf{p} :: \eta'_1, \mathsf{q} :: \eta'_2\}$. We consider two cases (the others being symmetric):

- $\nu \# \nu'$ because $\eta_1 \# \eta'_1$. Then $\eta_1 \Rsh \mathsf{q} \bowtie \eta_2 \Rsh \mathsf{p}$ and $\eta'_1 \Rsh \mathsf{q} \bowtie \eta'_2 \Rsh \mathsf{p}$ imply $\eta_2 \# \eta'_2$;
- $\nu \# \nu'$ because $|\eta_1 \Rsh \mathsf{q}\,| = |\eta'_2 \Rsh \mathsf{p}\,|$ and $\neg(\eta_1 \Rsh \mathsf{q} \bowtie \eta'_2 \Rsh \mathsf{p}\,)$. As argued before, we have $|\eta_2 \Rsh \mathsf{p}\,| = |\eta_1 \Rsh \mathsf{q}\,|$ and $|\eta'_2 \Rsh \mathsf{p}\,| = |\eta'_1 \Rsh \mathsf{q}\,|$. Then, from $|\eta_1 \Rsh \mathsf{q}\,| = |\eta'_2 \Rsh \mathsf{p}\,|$ and the above remark about binary events, we get $|\eta_2| = |\eta_1| = |\eta'_2| = |\eta'_1|$. From $\neg(\eta_1 \Rsh \mathsf{q} \bowtie \eta'_2 \Rsh \mathsf{p}\,)$ it follows that $\eta_1 \neq \eta'_1$ and $\eta_2 \neq \eta'_2$. Then we may conclude, since $|\eta_i| = |\eta'_i|$ and $\eta_i \neq \eta'_i$ imply $\eta_i \# \eta'_i$ for $i = 1, 2$.

Lemma 5.21 (Sharing of located events implies conflict) *If $\nu, \nu' \in \mathcal{NE}$ and $\nu \neq \nu'$ and $(\nu \cap \nu') \neq \emptyset$, then $\nu \# \nu'$.*

Proof Let $\mathsf{p} :: \eta \in (\nu \cap \nu')$ and $\mathsf{loc}(\nu) = \mathsf{loc}(\nu') = \{\mathsf{p}, \mathsf{q}\}$. Then there must exist η_0, η'_0 such that $\mathsf{q} :: \eta_0 \in \nu$ and $\mathsf{q} :: \eta'_0 \in \nu'$. From $\mathsf{p} :: \eta \ \widehat{\bowtie}\ \mathsf{q} :: \eta_0$ and $\mathsf{p} :: \eta \ \widehat{\bowtie}\ \mathsf{q} :: \eta'_0$ it follows that $\eta_0 \Rsh \mathsf{p} \ = \eta'_0 \Rsh \mathsf{p}$. This, in conjunction with the fact that $\mathsf{pt}(\mathsf{act}(\eta_0)) = \mathsf{pt}(\mathsf{act}(\eta'_0)) = \mathsf{p}$, implies that neither $\eta_0 < \eta'_0$ nor $\eta'_0 < \eta_0$. Thus $\eta_0 \# \eta'_0$ and therefore $\nu \# \nu'$ by Definition 5.7.

Lemma 5.24 *If $\mathcal{X}$ is a configuration of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ and $\nu \in \mathcal{X}$, then there is a unique causal set E of ν such that $E \subseteq \mathcal{X}$.*

Proof By Definition 5.11, if $\nu \in \mathcal{NE}(\mathsf{N})$, then ν has at least one causal set included in $\mathcal{NE}(\mathsf{N})$. Let $E' = \{\nu' \in \mathcal{X} \mid \nu' \prec \nu\}$. By Definition 3.4, $E' \cup \{\nu\}$ is conflict-free. Moreover, if $\mathsf{p} :: \eta \in \nu$ and $\eta' < \eta$, then by Lemma 5.21 there is at most one $\nu'' \in E'$ such that $\mathsf{p} :: \eta' \in \nu''$. Therefore, $E' \subseteq E$ for some causal set E of ν by Definition 5.9. We show that $E \subseteq E'$. Assume ad absurdum that $\nu_0 \in E \backslash E'$. By definition of causal set, $\nu_0 \prec \nu$. By definition of E', $\nu_0 \notin E'$ implies $\nu_0 \notin \mathcal{X}$. By Definition 3.4 this implies $\nu_0 \# \nu_1 \prec \nu$ for some $\nu_1 \in \mathcal{X}$. Then $\nu_1 \in E'$ by definition of E', and thus $\nu_1 \in E$. Hence $\nu_0, \nu_1 \in E$ and $\nu_0 \# \nu_1$, contradicting Definition 5.9.

B. Proofs of Section 6

This section contains the proofs of Lemmas 6.6, 6.9, Theorems 6.10, 6.11 and of the auxiliary Lemmas B.1, B.2, B.3.

Lemma 6.6 *If* G *is bounded, then* $\mathsf{G}\upharpoonright\mathsf{r}$ *is a partial function for all* r.

Proof We redefine the projection $\downarrow_\mathsf{r}$ as the largest relation between global types and processes such that $(\mathsf{G}, P) \in\downarrow_\mathsf{r}$ implies:

i) if $\mathsf{r} \notin \mathsf{part}(\mathsf{G})$, then $P = \mathbf{0}$;

ii) if $\mathsf{G} = \mathsf{r} \to \mathsf{p} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$, then $P = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$ and $(\mathsf{G}_i, P_i) \in\downarrow_\mathsf{r}$ for all $i \in I$;

iii) if $\mathsf{G} = \mathsf{p} \to \mathsf{r} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$, then $P = \Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$ and $(\mathsf{G}_i, P_i) \in\downarrow_\mathsf{r}$ for all $i \in I$;

iv) if $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$ and $\mathsf{r} \in \mathsf{part}(\mathsf{G}_i)$, then $(\mathsf{G}_i, P) \in\downarrow_\mathsf{r}$ for all $i \in I$.

The equality $\mathcal{E}$ of processes is the largest symmetric binary relation $\mathcal{R}$ on processes such that $(P, Q) \in \mathcal{R}$ implies:

(a) if $P = \bigoplus_{i\in I} \mathsf{p}!\lambda_i; P_i$, then $Q = \bigoplus_{i\in I} \mathsf{p}!\lambda_i; Q_i$ and $(P_i, Q_i) \in \mathcal{R}$ for all $i \in I$;

(b) if $P = \Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$, then $Q = \Sigma_{i\in I}\mathsf{p}?\lambda_i; Q_i$ and $(P_i, Q_i) \in \mathcal{R}$ for all $i \in I$.

It is then enough to show that the relation

$$\mathcal{R}_\mathsf{r} = \{(P, Q) \mid \exists\, \mathsf{G}\,.\, (\mathsf{G}, P) \in\downarrow_\mathsf{r} \text{ and } (\mathsf{G}, Q) \in\downarrow_\mathsf{r}\}$$

satisfies Clauses (a) and (b) (with $\mathcal{R}$ replaced by $\mathcal{R}_\mathsf{r}$), since this will imply $\mathcal{R}_\mathsf{r} \subseteq \mathcal{E}$. Note first that $(\mathbf{0}, \mathbf{0}) \in \mathcal{R}_\mathsf{r}$ because $(\mathsf{End}, \mathbf{0}) \in\downarrow_\mathsf{r}$, and that $(\mathbf{0}, \mathbf{0}) \in \mathcal{E}$ because Clauses (a) and (b) are vacuously satisfied by the pair $(\mathbf{0}, \mathbf{0})$. The proof is by induction on $d = \mathsf{depth}(\mathsf{G}, \mathsf{r})$. We only consider Clause (b), the proof for Clause (a) being similar. So, assume $(P, Q) \in \mathcal{R}_\mathsf{r}$ and $P = \Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$.
Case $d = 1$. In this case $\mathsf{G} = \mathsf{p} \to \mathsf{r} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $P = \Sigma_{i\in I}\mathsf{p}?\lambda_i; P_i$ and $(\mathsf{G}_i, P_i) \in\downarrow_\mathsf{r}$ for all $i \in I$. From $(\mathsf{G}, Q) \in\downarrow_\mathsf{r}$ we get $Q = \Sigma_{i\in I}\mathsf{p}?\lambda_i; Q_i$ and $(\mathsf{G}_i, Q_i) \in\downarrow_\mathsf{r}$ for all $i \in I$. Hence Q has the required form and $(P_i, Q_i) \in \mathcal{R}_\mathsf{r}$ for all $i \in I$.
Case $d > 1$. In this case $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{j\in J}\lambda'_j; \mathsf{G}_j$ and $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$ and $(\mathsf{G}_j, P) \in\downarrow_\mathsf{r}$ for all $j \in J$. From $(\mathsf{G}, Q) \in\downarrow_\mathsf{r}$ we get $(\mathsf{G}_j, Q) \in\downarrow_\mathsf{r}$ for all $j \in J$. Then $(P, Q) \in \mathcal{R}_\mathsf{r}$.

We need a lemma relating the projections of a well-formed global type with its transitions.

Lemma B.1. *Let* G *be a well-formed global type.*

1. *If* $\mathsf{G}\upharpoonright\mathsf{p} = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$ *and* $\mathsf{G}\upharpoonright\mathsf{q} = \Sigma_{j\in J}\mathsf{p}?\lambda'_j; Q_j$, *then* $I = J$, $\lambda_i = \lambda'_i$, $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda_i} \mathsf{G}_i$, $\mathsf{G}_i\upharpoonright\mathsf{p} = P_i$ *and* $\mathsf{G}_i\upharpoonright\mathsf{q} = Q_i$ *for all* $i \in I$.

2. *If* $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda} \mathsf{G}'$, *then* $\mathsf{G}\upharpoonright\mathsf{p} = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$, $\mathsf{G}\upharpoonright\mathsf{q} = \Sigma_{i\in I}\mathsf{p}?\lambda_i; Q_i$, *where* $\lambda_k = \lambda$ *for some* $k \in I$, *and* $\mathsf{G}'\upharpoonright\mathsf{r} = \mathsf{G}\upharpoonright\mathsf{r}$ *for all* $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$.

Proof (1). The proof is by induction on $d = \mathsf{depth}(\mathsf{G}, \mathsf{p})$.
If $d = 1$, then by definition of projection (see Figure 2) $\mathsf{G} \upharpoonright \mathsf{p} = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$ implies $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ with $\mathsf{G}_i \upharpoonright \mathsf{p} = P_i$. By the same definition $\mathsf{G} \upharpoonright \mathsf{q} = \sum_{j\in J} \mathsf{p}?\lambda'_j; Q_j$ implies $J = I$ and $\lambda'_j = \lambda_j$ and $Q_j = \mathsf{G}_j \upharpoonright \mathsf{q}$ for all $j \in J$. Moreover $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda_i} \mathsf{G}_i$ by Rule [Ecomm] for all $i \in I$.
If $d > 1$, then $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \boxplus_{h\in H}\lambda''_h; \mathsf{G}'_h$ with $\{\mathsf{p}, \mathsf{q}\} \cap \{\mathsf{r}, \mathsf{s}\} = \emptyset$. By definition of projection $\mathsf{G} \upharpoonright \mathsf{p} = \mathsf{G}'_h \upharpoonright \mathsf{p}$ and $\mathsf{G} \upharpoonright \mathsf{q} = \mathsf{G}'_h \upharpoonright \mathsf{q}$ for all $h \in H$. By Lemma 6.5 $\mathsf{depth}(\mathsf{G}, \mathsf{p}) > \mathsf{depth}(\mathsf{G}'_h, \mathsf{p})$ for all $h \in H$. Then by induction $I = J$, $\lambda_i = \lambda'_i$, $\mathsf{G}'_h \xrightarrow{\mathsf{pq}\lambda_i} \mathsf{G}^i_h$, $\mathsf{G}^i_h \upharpoonright \mathsf{p} = P_i$ and $\mathsf{G}^i_h \upharpoonright \mathsf{q} = Q_i$ for all $i \in I$ and all $h \in H$. Let $\mathsf{G}_i = \mathsf{r} \to \mathsf{s} : \boxplus_{h\in H}\lambda''_h; \mathsf{G}^i_h$. By Rule [Icomm] $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda_i} \mathsf{G}_i$ for all $i \in I$. By definition of projection $\mathsf{G}_i \upharpoonright \mathsf{p} = P_i$ and $\mathsf{G}_i \upharpoonright \mathsf{q} = Q_i$ for all $i \in I$.
(2). The proof is by induction on the transition rules of Figure 4.

The interesting case is:
$$\frac{\mathsf{G}_h \xrightarrow{\mathsf{pq}\lambda} \mathsf{G}'_h \quad h \in H \quad \{\mathsf{p}, \mathsf{q}\} \cap \{\mathsf{s}, \mathsf{t}\} = \emptyset}{\mathsf{s} \to \mathsf{t} : \boxplus_{h\in H}\lambda'_h; \mathsf{G}_h \xrightarrow{\mathsf{pq}\lambda} \mathsf{s} \to \mathsf{t} : \boxplus_{h\in H}\lambda'_h; \mathsf{G}'_h}\ [\textsc{Icomm}]$$
with $\mathsf{G} = \mathsf{s} \to \mathsf{t} : \boxplus_{h\in H}\lambda'_h; \mathsf{G}_h$ and $\mathsf{G}' = \mathsf{s} \to \mathsf{t} : \boxplus_{h\in H}\lambda'_h; \mathsf{G}'_h$. By induction $\mathsf{G}_h \upharpoonright \mathsf{p} = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$, $\mathsf{G}_h \upharpoonright \mathsf{q} = \sum_{i\in I} \mathsf{p}?\lambda_i; Q_i$, $\lambda = \lambda_k$ for some $k \in I$ and $\mathsf{G}'_h \upharpoonright \mathsf{r} = \mathsf{G}_h \upharpoonright \mathsf{r}$ for all $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$ and all $h \in H$. By definition of projection $\mathsf{G} \upharpoonright \mathsf{p} = \mathsf{G}_h \upharpoonright \mathsf{p}$ and $\mathsf{G} \upharpoonright \mathsf{q} = \mathsf{G}_h \upharpoonright \mathsf{q}$ for all $h \in H$. For $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}, \mathsf{s}, \mathsf{t}\}$ we get $\mathsf{G}' \upharpoonright \mathsf{r} = \mathsf{G}'_h \upharpoonright \mathsf{r} = \mathsf{G}_h \upharpoonright \mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r}$. Moreover $\mathsf{G}' \upharpoonright \mathsf{s} = \bigoplus_{h\in H} \mathsf{t}!\lambda'_h; \mathsf{G}'_h \upharpoonright \mathsf{s} = \bigoplus_{h\in H} \mathsf{t}!\lambda'_h; \mathsf{G}_h \upharpoonright \mathsf{s} = \mathsf{G} \upharpoonright \mathsf{s}$ and $\mathsf{G}' \upharpoonright \mathsf{t} = \sum_{h\in H} \mathsf{t}?\lambda'_h; \mathsf{G}'_h \upharpoonright \mathsf{t} = \sum_{h\in H} \mathsf{s}?\lambda'_h; \mathsf{G}_h \upharpoonright \mathsf{s} = \mathsf{G} \upharpoonright \mathsf{t}$.

Lemma 6.9 *If* G *is a well-formed global type and* $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda} \mathsf{G}'$, *then* G' *is a well-formed global type.*

Proof If $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda} \mathsf{G}'$, by Lemma B.1(1) and (2) $\mathsf{G}' \upharpoonright \mathsf{r}$ is defined for all r. The proof that $\mathsf{depth}(\mathsf{G}'', \mathsf{r})$ is finite for all r and G'' subtree of G' is easy by induction on the transition rules of Figure 4.

The proofs of Subject Reduction and Session Fidelity rely on the Inversion and Canonical Form lemmas whose proofs are immediate.

Lemma B.2 (Inversion). *If* $\vdash \mathsf{N} : \mathsf{G}$, *then* $P \leq \mathsf{G} \upharpoonright \mathsf{p}$ *for all* $\mathsf{p}[\![P]\!] \in \mathsf{N}$.

Lemma B.3 (Canonical Form). *If* $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{p} \in \mathsf{part}(\mathsf{G})$, *then* $\mathsf{p}[\![P]\!] \in \mathsf{N}$ *and* $P \leq \mathsf{G} \upharpoonright \mathsf{p}$.

Theorem 6.10 (Subject Reduction) *If* $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$, *then* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

Proof Let $\alpha = \mathsf{pq}\lambda$. By Rule [Com] of Figure 1, $\mathsf{N} \equiv \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \mathsf{N}''$ where $P = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i$ and $Q = \sum_{j\in J} \mathsf{p}?\lambda_j; Q_j$ and $\mathsf{N}' \equiv \mathsf{p}[\![P_h]\!] \parallel \mathsf{q}[\![Q_h]\!] \parallel \mathsf{N}''$ and $\lambda = \lambda_h$ for some $h \in I \cap J$. From Lemma B.2 we get

1. $\mathsf{G} \upharpoonright \mathsf{p} = \bigoplus_{i\in I} \mathsf{q}!\lambda_i; P'_i$ with $P_i \leq P'_i$ for all $i \in I$, from Rule [$\leq$ -Out] of Figure 3, and

2. $\mathsf{G} \upharpoonright \mathsf{q} = \Sigma_{j \in J'} \mathsf{p}?\lambda_j; Q'_j$ with $Q_j \leq Q'_j$ for all $j \in J' \subseteq J$, from Rule [$\leq$ -In] of Figure 3, and

3. $R \leq \mathsf{G} \upharpoonright \mathsf{r}$ for all $\mathsf{r}[\![R]\!] \in \mathsf{N}''$.

By Lemma B.1(1) $\mathsf{G} \xrightarrow{\mathsf{pq}\lambda_h} \mathsf{G}_h$ and $\mathsf{G}_h \upharpoonright \mathsf{p} = P'_h$ and $\mathsf{G}_h \upharpoonright \mathsf{q} = Q'_h$. By Lemma B.1(2) $\mathsf{G}_h \upharpoonright \mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r}$ for all $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$. We can then choose $\mathsf{G}' = \mathsf{G}_h$.

Theorem 6.11 (Session Fidelity) *If* $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$, *then* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

Proof Let $\alpha = \mathsf{pq}\lambda$. By Lemma B.1(2) $\mathsf{G} \upharpoonright \mathsf{p} = \bigoplus_{i \in I} \mathsf{p}!\lambda_i; P_i$ and $\mathsf{G} \upharpoonright \mathsf{q} = \Sigma_{i \in I} \mathsf{p}?\lambda_i; Q_i$ and $\lambda = \lambda_i$ for some $i \in I$ and $\mathsf{G}' \upharpoonright \mathsf{r} = \mathsf{G} \upharpoonright \mathsf{r}$ for all $\mathsf{r} \notin \{\mathsf{p}, \mathsf{q}\}$. By Lemma B.1(1) $\mathsf{G}' \upharpoonright \mathsf{p} = P_i$ and $\mathsf{G}' \upharpoonright \mathsf{q} = Q_i$. From Lemma B.3 and Lemma B.2 we get $\mathsf{N} \equiv \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \mathsf{N}''$ and

1. $P = \bigoplus_{i \in I} \mathsf{q}!\lambda_i; P'_i$ with $P'_i \leq P_i$ for $i \in I$, from Rule [$\leq$ -Out] of Figure 3, and

2. $Q = \Sigma_{j \in J} \mathsf{p}?\lambda_j; Q'_j$ with $Q'_j \leq Q_j$ for $j \in I \subseteq J$, from Rule [$\leq$ -In] of Figure 3, and

3. $R \leq \mathsf{G} \upharpoonright \mathsf{r}$ for all $\mathsf{r}[\![R]\!] \in \mathsf{N}''$.

We can then choose $\mathsf{N}' = \mathsf{p}[\![P'_i]\!] \parallel \mathsf{q}[\![Q'_i]\!] \parallel \mathsf{N}''$.

C. Proofs of Section 7

Lemma 7.4 *Let σ be a pointed trace. If $\sigma \sim \sigma'$, then σ' is a pointed trace and* $\mathsf{last}(\sigma) = \mathsf{last}(\sigma')$.

Proof Let $\sigma \sim \sigma'$. By Definition 7.1 σ' is obtained from σ by m swaps of adjacent communications. The proof is by induction on such a number m.
If $m = 0$ the result is obvious.
If $m > 0$, then there exists σ_0 obtained from σ by $m - 1$ swaps of adjacent communications and there are σ_1, σ_2, α and α' such that

$$\sigma_0 = \sigma_1 \cdot \alpha \cdot \alpha' \cdot \sigma_2 \ \sim \ \sigma_1 \cdot \alpha' \cdot \alpha \cdot \sigma_2 = \sigma' \ \text{ and } \ \mathsf{part}(\alpha) \cap \mathsf{part}(\alpha') = \emptyset$$

By induction hypothesis σ_0 is a pointed trace and $\mathsf{last}(\sigma) = \mathsf{last}(\sigma_0)$. Therefore $\sigma_2 \neq \epsilon$ since otherwise α' would be the last communication of σ_0 and it cannot be $\mathsf{part}(\alpha) \cap \mathsf{part}(\alpha') = \emptyset$. This implies $\mathsf{last}(\sigma) = \mathsf{last}(\sigma')$.
To show that σ' is pointed, since all the communications in σ_1 and σ_2 have the same successors in σ_0 and σ', all we have to prove is that the required property holds for the two swapped communications α' and α in σ', namely:

$$\mathsf{part}(\alpha') \cap (\mathsf{part}(\alpha) \cup \mathsf{part}(\sigma_2)) \neq \emptyset$$
$$\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma_2) \neq \emptyset$$

Since $\mathsf{part}(\alpha) \cap \mathsf{part}(\alpha') = \emptyset$, these two statements are respectively equivalent to:

$$\mathsf{part}(\alpha') \cap \mathsf{part}(\sigma_2) \neq \emptyset$$
$$\mathsf{part}(\alpha) \cap (\mathsf{part}(\alpha') \cup \mathsf{part}(\sigma_2)) \neq \emptyset$$

The last two statements are known to hold since σ_0 is pointed by induction hypothesis.

D. Proofs of Subsection 8.1

This section contains the proofs of Lemmas 8.2, 8.4, 8.5 and 8.6.

Lemma 8.2 (Properties of retrieval and residual for n-events).

1. *If* $\alpha \blacklozenge \nu$ *is defined, then* $\alpha \lozenge (\alpha \blacklozenge \nu) = \nu$*;*
2. $\alpha \blacklozenge (\alpha \lozenge \nu) = \nu$*;*
3. *If* $\nu \prec \nu'$*, then* $\alpha \lozenge \nu \prec \alpha \lozenge \nu'$*;*
4. *If* $\nu \prec \nu'$ *and both* $\alpha \blacklozenge \nu$ *and* $\alpha \blacklozenge \nu'$ *are defined, then* $\alpha \blacklozenge \nu \prec \alpha \blacklozenge \nu'$*;*
5. *If* $\nu \# \nu'$*, then* $\alpha \lozenge \nu \# \alpha \lozenge \nu'$*;*
6. *If* $\nu \# \nu'$ *and both* $\alpha \blacklozenge \nu$ *and* $\alpha \blacklozenge \nu'$ *are defined, then* $\alpha \blacklozenge \nu \# \alpha \blacklozenge \nu'$*;*
7. *If* $\alpha \lozenge \nu \# \alpha \lozenge \nu'$*, then* $\nu \# \nu'$*.*

Proof For (1) and (2) it is enough to show the corresponding properties for located events.

(1) Since $\alpha \blacklozenge (\mathsf{p} :: \eta)$ is defined, we have $\eta = (\alpha @ \mathsf{p}) \cdot \eta'$ and $\alpha \blacklozenge (\mathsf{p} :: \eta) = \mathsf{p} :: \eta'$ for some η'. Then $\alpha \lozenge (\alpha \blacklozenge (\mathsf{p} :: \eta)) = \alpha \lozenge (\mathsf{p} :: \eta') = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta' = \mathsf{p} :: \eta$.

(2) Since $\alpha \lozenge (\mathsf{p} :: \eta) = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta$ is always defined, we immediately get $\alpha \blacklozenge (\alpha \lozenge (\mathsf{p} :: \eta)) = \alpha \blacklozenge (\mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta) = \mathsf{p} :: \eta$.

(3) Let $\nu \prec \nu'$. By Definition 5.7(1), there are $\mathsf{p} :: \eta \in \nu$ and $\mathsf{p} :: \eta' \in \nu'$ such that $\eta < \eta'$. Then $\alpha \lozenge (\mathsf{p} :: \eta) = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta \in \alpha \lozenge \nu$ and $\alpha \lozenge (\mathsf{p} :: \eta') = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta' \in \alpha \lozenge \nu'$. Since $\eta < \eta'$ implies $(\alpha @ \mathsf{p}) \cdot \eta < (\alpha @ \mathsf{p}) \cdot \eta'$, we conclude that $\alpha \lozenge \nu \prec \alpha \lozenge \nu'$.

(4) As in the previous case, there are $\mathsf{p} :: \eta \in \nu$ and $\mathsf{p} :: \eta' \in \nu'$ such that $\eta < \eta'$. Since both $\alpha \blacklozenge \nu$ and $\alpha \blacklozenge \nu'$ are defined, there exist η_0 and η'_0 such that $\eta = (\alpha @ \mathsf{p}) \cdot \eta_0$ and $\eta' = (\alpha @ \mathsf{p}) \cdot \eta'_0$ and $\alpha \blacklozenge (\mathsf{p} :: \eta) = \mathsf{p} :: \eta_0$ and $\alpha \blacklozenge (\mathsf{p} :: \eta') = \mathsf{p} :: \eta'_0$. Since $\eta < \eta'$ implies $\eta_0 < \eta'_0$, we conclude that $\alpha \blacklozenge \nu \prec \alpha \blacklozenge \nu'$.

(5) Let $\nu \# \nu'$. If Clause (2a) of Definition 5.7 applies, then there are $\mathsf{p} :: \eta \in \nu$ and $\mathsf{p} :: \eta' \in \nu'$ such that $\eta \# \eta'$. From $\alpha \lozenge (\mathsf{p} :: \eta) = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta$ and $\alpha \lozenge (\mathsf{p} :: \eta') = \mathsf{p} :: (\alpha @ \mathsf{p}) \cdot \eta'$ we get $(\alpha @ \mathsf{p}) \cdot \eta \# (\alpha @ \mathsf{p}) \cdot \eta'$. If Clause (2b) of Definition 5.7 applies, then there are $\mathsf{p} :: \eta \in \nu$ and $\mathsf{q} :: \eta' \in \nu'$ with $\mathsf{p} \neq \mathsf{q}$ such that $|\eta \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}|$ and $\neg(\eta \upharpoonright \mathsf{q} \bowtie \eta' \upharpoonright \mathsf{p})$. Let $\eta_0 = (\alpha @ \mathsf{p}) \cdot \eta$ and $\eta'_0 = (\alpha @ \mathsf{q}) \cdot \eta'$. If $\mathsf{part}(\alpha) \neq \{\mathsf{p}, \mathsf{q}\}$, then $(\alpha @ \mathsf{p}) \upharpoonright \mathsf{q} = \epsilon = (\alpha @ \mathsf{q}) \upharpoonright \mathsf{p}$ and thus $\eta_0 \upharpoonright \mathsf{q} = \eta \upharpoonright \mathsf{q}$ and $\eta'_0 \upharpoonright \mathsf{p} = \eta' \upharpoonright \mathsf{p}$. If $\mathsf{part}(\alpha) = \{\mathsf{p}, \mathsf{q}\}$, say $\alpha = \mathsf{pq}\lambda$, then $\eta_0 = \mathsf{q}!\lambda \cdot \eta$ and $\eta'_0 = \mathsf{p}?\lambda \cdot \eta'$, which implies $|\eta_0 \upharpoonright \mathsf{q}| = |\eta \upharpoonright \mathsf{q}| + 1 = |\eta' \upharpoonright \mathsf{p}| + 1 = |\eta'_0 \upharpoonright \mathsf{p}|$ and $\neg(\eta_0 \upharpoonright \mathsf{q} \bowtie \eta'_0 \upharpoonright \mathsf{p})$. In both cases we conclude that $\alpha \lozenge \nu \# \alpha \lozenge \nu'$.

(6) The proof is similar to that of Point (5), considering that $\alpha \blacklozenge \nu$ and $\alpha \blacklozenge \nu'$ are defined.

(7) Let $\alpha \lozenge \nu \# \alpha \lozenge \nu'$. If Clause (2a) of Definition 5.7 applies, then there are $\mathsf{p} :: \eta \in \nu$ and $\mathsf{p} :: \eta' \in \nu'$ such that $(\alpha @ \mathsf{p}) \cdot \eta \# (\alpha @ \mathsf{p}) \cdot \eta'$. Therefore $\eta \# \eta'$ and thus $\nu \# \nu'$. If Clause (2b) of Definition 5.7 applies, then there are $\mathsf{p} :: \eta_0 = \alpha \lozenge (\mathsf{p} :: \eta) \in \alpha \lozenge \nu$ and $\mathsf{q} :: \eta'_0 = \alpha \lozenge (\mathsf{q} :: \eta') \in \alpha \lozenge \nu'$ with $\mathsf{p} \neq \mathsf{q}$ such that $|\eta_0 \upharpoonright \mathsf{q}| = |\eta'_0 \upharpoonright \mathsf{p}|$ and $\neg(\eta_0 \upharpoonright \mathsf{q} \bowtie \eta'_0 \upharpoonright \mathsf{p})$. It follows that $\eta_0 = (\alpha @ \mathsf{p}) \cdot \eta$ and $\eta'_0 = (\alpha @ \mathsf{q}) \cdot \eta'$ and $\mathsf{p} :: \eta \in \nu$

and $\mathsf{q} :: \eta' \in \nu'$. If $\mathsf{part}(\alpha) \neq \{\mathsf{p},\mathsf{q}\}$, then $(\alpha@\mathsf{p}) \upharpoonright \mathsf{q} = \epsilon = (\alpha@\mathsf{q}) \upharpoonright \mathsf{p}$ and thus $\eta \upharpoonright \mathsf{q} = \eta_0 \upharpoonright \mathsf{q}$ and $\eta' \upharpoonright \mathsf{p} = \eta'_0 \upharpoonright \mathsf{p}$. If $\mathsf{part}(\alpha) = \{\mathsf{p},\mathsf{q}\}$, say $\alpha = \mathsf{pq}\lambda$, then $\eta_0 = \mathsf{q}!\lambda \cdot \eta$ and $\eta'_0 = \mathsf{p}?\lambda \cdot \eta'$, and thus $|\eta \upharpoonright \mathsf{q}| = |\eta_0 \upharpoonright \mathsf{q}| - 1 = |\eta'_0 \upharpoonright \mathsf{p}| - 1 = |\eta' \upharpoonright \mathsf{p}|$ and $\neg(\eta \upharpoonright \mathsf{q} \bowtie \eta' \upharpoonright \mathsf{p})$. In both cases we conclude that $\nu \,\#\, \nu'$.

Lemma 8.4 (Properties of $\mathsf{nec}(\cdot)$)

1. *Let $\mathsf{nec}(\sigma) = \nu_1; \cdots; \nu_n$. Then*
   (a) *$\mathsf{cm}(\nu_i) = \sigma[i]$ for all i, $1 \leq i \leq n$;*
   (b) *If $1 \leq h, k \leq n$, then $\neg(\nu_h \,\#\, \nu_k)$.*
2. *$\neg(\mathsf{nec}(\alpha) \,\#\, \alpha \lozenge \nu)$ for all ν.*
3. *Let $\sigma = \alpha \cdot \sigma'$ and $\sigma' \neq \epsilon$. If $\mathsf{nec}(\sigma) = \nu_1; \cdots; \nu_n$ and $\mathsf{nec}(\sigma') = \nu'_2; \cdots; \nu'_n$, then $\alpha \lozenge \nu'_i = \nu_i$ and $\alpha \blacklozenge \nu_i = \nu'_i$ for all i, $2 \leq i \leq n$.*

Proof (1a) Immediate from Definition 8.3, since $\mathsf{cm}(\sigma \lozenge \nu) = \mathsf{cm}(\nu)$ for any event ν.

(1b) We show that neither Clause (2a) nor Clause (2b) of Definition 5.7 can be used to derive $\nu_h \,\#\, \nu_k$. Notice that $\nu_i = \{\mathsf{p}_i :: \sigma[1 \ldots i]@\mathsf{p}_i, \mathsf{q}_i :: \sigma[1 \ldots i]@\mathsf{q}_i\}$. So if $\mathsf{p} :: \eta \in \nu_h$ and $\mathsf{p} :: \eta' \in \nu_k$ with $h < k$, then either $\eta < \eta'$ or $\eta = \eta'$. Therefore Clause (2a) does not apply. If $\mathsf{p} :: \eta \in \nu_h$ and $\mathsf{q} :: \eta' \in \nu_k$ and $\mathsf{p} \neq \mathsf{q}$ and $|\eta \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}|$, then it must be $\eta \upharpoonright \mathsf{q} = (\sigma[1 \ldots h]@\mathsf{p}) \upharpoonright \mathsf{q} \bowtie (\sigma[1 \ldots k]@\mathsf{q}) \upharpoonright \mathsf{p} = \eta' \upharpoonright \mathsf{p}$. Therefore Clause (2b) cannot be used.

(2) We show that neither Clause (2a) nor Clause (2b) of Definition 5.7 can be used to derive $\mathsf{nec}(\alpha) \,\#\, \alpha \lozenge \nu$. Let $\mathsf{part}(\alpha) = \{\mathsf{p},\mathsf{q}\}$. Then $\mathsf{nec}(\alpha) = \{\mathsf{p} :: \alpha@\mathsf{p}, \mathsf{q} :: \alpha@\mathsf{q}\}$. Note that $\mathsf{p} :: \eta \in \alpha \lozenge \nu$ iff $\eta = (\alpha@\mathsf{p}) \cdot \eta'$ and $\mathsf{p} :: \eta' \in \nu$. Since $\alpha@\mathsf{p} < (\alpha@\mathsf{p}) \cdot \eta'$, Clause (2a) of Definition 5.7 cannot be used. Now suppose $\mathsf{r} :: \eta \in \alpha \lozenge \nu$ for some $\mathsf{r} \notin \{\mathsf{p},\mathsf{q}\}$. In this case $(\alpha@\mathsf{p}) \upharpoonright \mathsf{r} = (\alpha@\mathsf{q}) \upharpoonright \mathsf{r} = \epsilon$. Therefore, since $\epsilon \bowtie \epsilon$, Clause (2b) of Definition 5.7 does not apply.

(3) Notice that $\sigma[i] = \sigma'[i-1]$ for all i, $2 \leq i \leq n$. Then, by Definition 8.3

$$\begin{aligned}\nu_i \;=\;& \sigma[1 \ldots i-1] \lozenge \mathsf{nec}(\sigma[i]) = \alpha \lozenge (\sigma[2 \ldots i-1] \lozenge \mathsf{nec}(\sigma[i])) = \\ & \alpha \lozenge (\sigma'[1 \ldots i-2] \lozenge \mathsf{nec}(\sigma'[i-1])) = \alpha \lozenge \nu'_i\end{aligned}$$

for all i, $2 \leq i \leq n$.
By Lemma 8.2(2) $\alpha \lozenge \nu'_i = \nu_i$ implies $\alpha \blacklozenge \nu_i = \nu'_i$ for all i, $2 \leq i \leq n$.

Lemma 8.5 *If $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ and $\nu \in \mathcal{NE}(\mathsf{N})$, then $\nu = \mathsf{nec}(\alpha)$ or $\nu \,\#\, \mathsf{nec}(\alpha)$ or $\alpha \blacklozenge \nu$ is defined.*

Proof Let $\mathsf{nec}(\alpha) = \{\mathsf{p} :: \alpha@\mathsf{p}, \mathsf{q} :: \alpha@\mathsf{q}\}$ and $\nu = \{\mathsf{r} :: \eta, \mathsf{s} :: \eta'\}$. By Definition 8.1(3) $\alpha \blacklozenge \nu$ is defined iff $\eta = (\alpha@\mathsf{r}) \cdot \eta_0$ and $\eta' = (\alpha@\mathsf{s}) \cdot \eta'_0$ for some η_0, η'_0.
There are 2 possibilities:

- $\{\mathsf{r},\mathsf{s}\} \cap \{\mathsf{p},\mathsf{q}\} = \emptyset$. Then $\alpha@\mathsf{r} = \alpha@\mathsf{s} = \epsilon$ and $\alpha \blacklozenge \nu = \nu$;
- $\{\mathsf{r},\mathsf{s}\} \cap \{\mathsf{p},\mathsf{q}\} \neq \emptyset$. Suppose $\mathsf{r} = \mathsf{p}$. There are three possible subcases:
  1. $\eta = \pi \cdot \zeta$ with $\pi \neq \alpha@\mathsf{p}$. Then $\mathsf{r} :: \eta \,\#\, \mathsf{p} :: \alpha@\mathsf{p}$ and thus $\nu \,\#\, \mathsf{nec}(\alpha)$;

2. $\eta = \alpha@\mathsf{p}$. Then either $\eta' = \alpha@\mathsf{q}$ and $\nu = \mathsf{nec}(\alpha)$, or $\eta' \neq \alpha@\mathsf{q}$ and $\nu \# \mathsf{nec}(\alpha)$ by Lemma 5.21;

3. $\eta = (\alpha@\mathsf{p}) \cdot \eta_0$. Then $\alpha \blacklozenge \mathsf{p} :: \eta = \mathsf{p} :: \eta_0$. Now, if $\mathsf{s} \neq \mathsf{q}$ we have $\alpha \blacklozenge \mathsf{s} :: \eta' = \mathsf{s} :: \eta'$, and thus $\alpha \blacklozenge \nu = \{\mathsf{p} :: \eta_0, \mathsf{s} :: \eta'\}$. Otherwise, $\nu = \{\mathsf{p} :: (\alpha@\mathsf{p}) \cdot \eta_0, \mathsf{q} :: \eta'\}$. By Definition 5.5 $\mathsf{p} :: (\alpha@\mathsf{p}) \cdot \eta_0 \;\widehat{\bowtie}\; \mathsf{q} :: \eta'$, which implies $\eta' = (\alpha@\mathsf{q}) \cdot \eta'_0$ for some η'_0.

Lemma 8.6 *Let* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$. *Then*

1. $\{\mathsf{nec}(\alpha)\} \cup \{\alpha \lozenge \nu \mid \nu \in \mathcal{NE}(\mathsf{N}')\} \subseteq \mathcal{NE}(\mathsf{N})$;

2. $\{\alpha \blacklozenge \nu \mid \nu \in \mathcal{NE}(\mathsf{N})$ *and* $\alpha \blacklozenge \nu$ *defined*$\} \subseteq \mathcal{NE}(\mathsf{N}')$.

Proof Let $\alpha = \mathsf{pq}\lambda$. From $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ we get

$$\mathsf{N} = \mathsf{p}[\![\textstyle\bigoplus_{i\in I} \mathsf{q}!\lambda_i; P]\!] \parallel \mathsf{q}[\![\sum_{j\in J} \mathsf{p}?\lambda_j; Q_j]\!] \parallel \mathsf{N}_0$$

where for some $k \in (I \cap J)$ we have $\lambda_k = \lambda$ and

$$\mathsf{N}' = \mathsf{p}[\![P_k]\!] \parallel \mathsf{q}[\![Q_k]\!] \parallel \mathsf{N}_0$$

(1) Let $\mathcal{RT} = \{\mathsf{nec}(\alpha)\} \cup \{\alpha \lozenge \nu \mid \nu \in \mathcal{NE}(\mathsf{N}')\}$. We first show that $\mathcal{RT} \subseteq \mathcal{CE}(\mathsf{N})$. By Definition 5.13(1) $\mathsf{nec}(\alpha) \in \mathcal{CE}(\mathsf{N})$. Let $\nu = \{\mathsf{r} :: \eta, \mathsf{s} :: \eta'\} \in \mathcal{NE}(\mathsf{N}')$. We want to prove that $\alpha \lozenge \nu \in \mathcal{CE}(\mathsf{N})$. By Definition 5.13(1) there are R, S such that $\mathsf{r}[\![R]\!] \in \mathsf{N}'$ and $\mathsf{s}[\![S]\!] \in \mathsf{N}'$ and $\eta \in \mathcal{PE}(R)$ and $\eta' \in \mathcal{PE}(S)$. There are two possible cases:

- $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. Then $\mathsf{r}[\![R]\!] \in \mathsf{N}$ and $\mathsf{s}[\![S]\!] \in \mathsf{N}$ and thus $\alpha \lozenge \nu = \nu \in \mathcal{CE}(\mathsf{N})$;
- $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} \neq \emptyset$. Suppose $\mathsf{r} = \mathsf{p}$. Then $\eta \in \mathcal{PE}(P_k)$ and $\mathsf{p} :: \mathsf{q}!\lambda_k \cdot \eta \in \alpha \lozenge \nu$ and $\mathsf{q}!\lambda_k \cdot \eta \in \mathcal{PE}(\bigoplus_{i\in I} \mathsf{q}!\lambda_i; P_i)$. There are two subcases:
  - $\mathsf{s} = \mathsf{q}$. Then $\eta' \in \mathcal{PE}(Q_k)$ and $\mathsf{q} :: \mathsf{p}?\lambda_k \cdot \eta' \in \alpha \lozenge \nu$ and $\mathsf{q}!\lambda_k \cdot \eta' \in \mathcal{PE}(\sum_{j\in J} \mathsf{p}?\lambda_j; Q_j)$. We have $\alpha \lozenge \nu = \{\mathsf{p} :: \mathsf{q}!\lambda_k \cdot \eta, \mathsf{q} :: \mathsf{p}?\lambda_k \cdot \eta'\} \in \mathcal{CE}(\mathsf{N})$;
  - $\mathsf{s} \neq \mathsf{q}$. Then $\alpha \lozenge \mathsf{s} :: \eta' = \mathsf{s} :: \eta'$, and thus $\alpha \lozenge \nu = \{\mathsf{p} :: \mathsf{q}!\lambda_k \cdot \eta, \mathsf{s} :: \eta'\} \in \mathcal{CE}(\mathsf{N})$.

Therefore in all cases $\mathcal{RT} \subseteq \mathcal{CE}(\mathsf{N})$. We want now to show that $\mathcal{RT} \subseteq \mathcal{NE}(\mathsf{N})$.

Recall from Section 5 that $\mathcal{NE}(\mathsf{N})$ is the greatest fixed point of the function

$$f_{\mathcal{CE}(\mathsf{N})}(X) = \{\nu_0 \in \mathcal{CE}(\mathsf{N}) \mid \exists E_0 \subseteq X.\, E_0 \text{ is a causal set of } \nu_0 \text{ in } X\}$$

Then $\mathcal{NE}(\mathsf{N})$ is also the greatest post-fixed point of $f_{\mathcal{CE}(\mathsf{N})}(X)$, namely the greatest X such that $X \subseteq f_{\mathcal{CE}(\mathsf{N})}(X)$. Therefore, to show that $\mathcal{RT} \subseteq \mathcal{NE}(\mathsf{N})$, it is enough to show that $\mathcal{RT}$ is also a post-fixed point of $f_{\mathcal{CE}(\mathsf{N})}(X)$, namely that $\mathcal{RT} \subseteq f_{\mathcal{CE}(\mathsf{N})}(\mathcal{RT})$.

Consider first the event $\mathsf{nec}(\alpha)$. Since the only causal set of $\mathsf{nec}(\alpha)$ in any set is $\emptyset$, it is immediate that $\mathsf{nec}(\alpha) \in f_{\mathcal{CE}(\mathsf{N})}(\mathcal{RT})$. Consider now $\alpha \lozenge \nu \in \mathcal{RT}$ for some $\nu \in \mathcal{NE}(\mathsf{N}')$ with $\mathsf{loc}(\nu) = \{\mathsf{r}, \mathsf{s}\}$. Define

$$\mathsf{pre}(\alpha, E, \nu) = \begin{cases} \Xi & \text{if } \{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset \\ \{\mathsf{nec}(\alpha)\} \cup \Xi & \text{otherwise} \end{cases}$$

where $\Xi = \{\alpha \lozenge \nu' \mid \nu' \in E$ and E is a causal set of ν in $\mathcal{NE}(\mathsf{N}')\}$.

We show that $\mathsf{pre}(\alpha, E, \nu)$ is a causal set of $\alpha \lozenge \nu$ in $\mathcal{RT}$, namely that it is a minimal subset of $\mathcal{RT}$ satisfying Conditions (1) and (2) of Definition 5.9.

Condition (1) If $\mathsf{nec}(\alpha) \in \mathsf{pre}(\alpha, E, \nu)$, then $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} \neq \emptyset$. A conflict between $\mathsf{nec}(\alpha)$ and any other event of $\mathsf{pre}(\alpha, E, \nu) \cup \{\alpha \lozenge \nu\}$ can only be derived by Clause (2a) of Definition 5.7, since $\mathsf{nec}(\alpha) = \{\mathsf{p} :: \mathsf{q}!\lambda, \mathsf{q} :: \mathsf{p}?\lambda\}$ and $(\alpha @ \mathsf{p}) \upharpoonright \mathsf{t} = (\alpha @ \mathsf{q}) \upharpoonright \mathsf{t} = \epsilon$ for all $\mathsf{t} \notin \{\mathsf{p}, \mathsf{q}\}$. Suppose $\mathsf{r} = \mathsf{p}$. Then $\mathsf{p} :: \mathsf{q}!\lambda \cdot \eta \in \alpha \lozenge \nu$. Since $\mathsf{q}!\lambda < \mathsf{q}!\lambda \cdot \eta$, Clause (2a) cannot be used to derive a conflict $\mathsf{nec}(\alpha) \# \alpha \lozenge \nu$. Similarly, if $\alpha \lozenge \nu_1 \in \mathsf{pre}(\alpha, E, \nu)$ and $\mathsf{p} :: \eta_1 \in \nu_1$, then $\mathsf{p} :: \mathsf{q}!\lambda \cdot \eta_1 \in \alpha \lozenge \nu_1$. Then $\mathsf{q}!\lambda < \mathsf{q}!\lambda \cdot \eta_1$, hence Clause (2a) cannot be used to derive $\mathsf{nec}(\alpha) \# \alpha \lozenge \nu_1$.

Suppose now $\alpha \lozenge \nu_1 \in \mathsf{pre}(\alpha, E, \nu)$ and $\alpha \lozenge \nu_2 \in \mathsf{pre}(\alpha, E, \nu)$. Since E is a causal set, we have $\neg(\nu_1 \# \nu_2)$. Thus $\neg(\alpha \lozenge \nu_1 \# \alpha \lozenge \nu_2)$ by Lemma 8.2(7).

Condition (2) Let $\nu = \{\mathsf{r} :: \eta, \mathsf{s} :: \eta'\}$, we have $\alpha \lozenge \nu = \{\mathsf{r} :: (\alpha @ \mathsf{r}) \cdot \eta, \mathsf{s} :: (\alpha @ \mathsf{s}) \cdot \eta'\}$. We show that if $\eta_0 < (\alpha @ \mathsf{r}) \cdot \eta$, then $\mathsf{r} :: \eta_0 \in \nu_0$ for some $\nu_0 \in \mathsf{pre}(\alpha, E, \nu)$. From $\eta_0 < (\alpha @ \mathsf{r}) \cdot \eta$ we derive $\eta_0 = (\alpha @ \mathsf{r}) \cdot \zeta$ for some ζ such that $\zeta < \eta$. If $\zeta \neq \epsilon$, then $\zeta = \eta_0' < \eta$. Since E is a causal set, $\eta_0' < \eta_0$ implies $\mathsf{r} :: \eta_0' \Subset E$. Hence $\mathsf{r} :: \eta_0 \Subset \mathsf{pre}(\alpha, E, \nu)$. If instead $\zeta = \epsilon$, then it must be $\eta_0 = \alpha @ \mathsf{r} \neq \epsilon$ and thus $\mathsf{r} \in \{\mathsf{p}, \mathsf{q}\}$. In this case $\{\mathsf{nec}(\alpha)\} \in \mathsf{pre}(\alpha, E, \nu)$ and thus $\mathsf{r} :: \eta_0 \Subset \mathsf{pre}(\alpha, E, \nu)$.

As for *minimality* , we first show that $\nu' \prec \alpha \lozenge \nu$ for all $\nu' \in \mathsf{pre}(\alpha, E, \nu)$. If $\mathsf{nec}(\alpha) \in \mathsf{pre}(\alpha, E, \nu)$, then $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} \neq \emptyset$. Then $\mathsf{nec}(\alpha) \prec \alpha \lozenge \nu$. If $\nu_1 \in \mathsf{pre}(\alpha, E, \nu)$ and $\nu_1 \neq \mathsf{nec}(\alpha)$, then there exists $\nu_1' \in E$ such that $\nu_1 = \alpha \lozenge \nu_1'$. Since E is a causal set for ν, we have $\nu_1' \prec \nu$. Therefore $\nu_1 = \alpha \lozenge \nu_1' \prec \alpha \lozenge \nu$ by Lemma 8.2(3). Assume now that $\mathsf{pre}(\alpha, E, \nu)$ is not minimal. Then there is $E' \subset \mathsf{pre}(\alpha, E, \nu)$ that verifies Condition (2) of Definition 5.9 for $\alpha \lozenge \nu$. Let $\nu' \in \mathsf{pre}(\alpha, E, \nu) \setminus E'$. Then $\nu' \prec \alpha \lozenge \nu = \{\mathsf{r} :: \eta_{\mathsf{r}}, \mathsf{s} :: \eta_{\mathsf{s}}\}$. Assume that $\mathsf{r} :: \eta_{\mathsf{r}}' \in \nu'$ with $\eta_{\mathsf{r}}' < \eta_{\mathsf{r}}$ (the proof is similar for s). By Condition (2), there is $\nu'' \in E'$ such that $\mathsf{r} :: \eta_{\mathsf{r}}' \in \nu''$. But then $\nu' \# \nu''$ by Lemma 5.21, contradicting the fact that $\mathsf{pre}(\alpha, E, \nu)$ verifies Condition (1). Therefore $\mathsf{pre}(\alpha, E, \nu)$ is minimal.

(2) Let $\mathcal{RS} = \{\alpha \blacklozenge \nu \mid \nu \in \mathcal{NE}(\mathsf{N})$ and $\alpha \blacklozenge \nu$ defined$\}$. We first show that $\mathcal{RS} \subseteq \mathcal{CE}(\mathsf{N}')$. Let $\nu = \{\mathsf{r} :: \eta, \mathsf{s} :: \eta'\} \in \mathcal{NE}(\mathsf{N})$ be such that $\alpha \blacklozenge \nu$ is defined. We want to prove that $\alpha \blacklozenge \nu \in \mathcal{CE}(\mathsf{N}')$. By Definition 5.13(1) there are R, S such that $\mathsf{r}[\![R]\!] \in \mathsf{N}$ and $\mathsf{s}[\![S]\!] \in \mathsf{N}$ and $\eta \in \mathcal{PE}(R)$ and $\eta' \in \mathcal{PE}(S)$. There are two possible cases:

- $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. Then $\mathsf{r}[\![R]\!] \in \mathsf{N}'$ and $\mathsf{s}[\![S]\!] \in \mathsf{N}'$ and thus $\alpha \blacklozenge \nu = \nu \in \mathcal{CE}(\mathsf{N}')$;
- $\{\mathsf{r}, \mathsf{s}\} \cap \{\mathsf{p}, \mathsf{q}\} \neq \emptyset$. Suppose $\mathsf{r} = \mathsf{p}$. Then $\eta \in \mathcal{PE}(\bigoplus_{i \in I} \mathsf{q}!\lambda_i; P_i)$ and since $\alpha \blacklozenge \nu$ is defined we have that $\eta = \mathsf{q}!\lambda_k \cdot \eta_k$ where $\eta_k \in \mathcal{PE}(P_k)$. There are two subcases:
  - $\mathsf{s} = \mathsf{q}$. Then $\eta' \in \mathcal{PE}(\Sigma_{j \in J} \mathsf{p}?\lambda_j; Q_j)$ and since $\alpha \blacklozenge \nu$ is defined $\eta' = \mathsf{p}?\lambda_k \cdot \eta_k'$ where $\eta_k' \in \mathcal{PE}(Q_k)$. In this case we have $\alpha \blacklozenge \nu = \{\mathsf{p} :: \eta_k, \mathsf{q} :: \eta_k'\} \in \mathcal{CE}(\mathsf{N}')$;
  - $\mathsf{s} \neq \mathsf{q}$. Then $\alpha \blacklozenge \mathsf{s} :: \eta' = \mathsf{s} :: \eta'$, and thus $\alpha \blacklozenge \nu = \{\mathsf{p} :: \eta_k, \mathsf{s} :: \eta'\} \in \mathcal{CE}(\mathsf{N}')$.

Therefore in all cases $\mathcal{RS} \subseteq \mathcal{CE}(\mathsf{N}')$. We want now to show that $\mathcal{RS} \subseteq \mathcal{NE}(\mathsf{N}')$.

We proceed as in the proof of Statement (1). We know that $\mathcal{NE}(\mathsf{N}')$ is the greatest post-fixed point of the function

$$f_{\mathcal{CE}(\mathsf{N}')}(X) = \{\nu_0 \in \mathcal{CE}(\mathsf{N}') \mid \exists E_0 \subseteq X.\, E_0 \text{ is a causal set of } \nu_0 \text{ in } X\}$$

Then, in order to obtain $\mathcal{RS} \subseteq \mathcal{NE}(\mathsf{N}')$ it is enough to show that $\mathcal{RS}$ is a post-fixed point of $f_{\mathcal{CE}(\mathsf{N}')}(X)$, namely that $\mathcal{RS} \subseteq f_{\mathcal{CE}(\mathsf{N}')}(\mathcal{RS})$.

Let $\alpha \blacklozenge \nu \in \mathcal{RS}$ for some $\nu \in \mathcal{NE}(\mathsf{N})$. Define

$$\mathsf{post}(\alpha, E, \nu) = \{\alpha \blacklozenge \nu' \mid \nu' \in E \text{ and } E \text{ is a causal set of } \nu \text{ in } \mathcal{NE}(\mathsf{N})\}$$

We show that $\mathsf{post}(\alpha, E, \nu)$ is a causal set of $\alpha \blacklozenge \nu$ in $\mathcal{RS}$, namely that it is a minimal subset of $\mathcal{RS}$ satisfying Conditions (1) and (2) of Definition 5.9.
Condition (1) Suppose $\alpha \blacklozenge \nu_1 \in \mathsf{post}(\alpha, E, \nu)$ and $\alpha \blacklozenge \nu_2 \in \mathsf{post}(\alpha, E, \nu)$. Since E is a causal set and $\nu_1, \nu_2 \in E$, we have $\neg(\nu_1 \# \nu_2)$. Thus $\neg(\alpha \blacklozenge \nu_1 \# \alpha \blacklozenge \nu_2)$ by Lemma 8.2(5) and (1).
Condition (2) Since $\nu = \{\mathsf{r} :: \eta, \mathsf{s} :: \eta'\}$ and $\alpha \blacklozenge \nu$ is defined, we have $\eta = (\alpha@\mathsf{r}) \cdot \eta_\mathsf{r}$ and $\eta' = (\alpha@\mathsf{s}) \cdot \eta_\mathsf{s}$ and $\alpha \blacklozenge \nu = \{\mathsf{r} :: \eta_\mathsf{r}, \mathsf{s} :: \eta_\mathsf{s}\}$. Let $\eta_0 < \eta_\mathsf{r}$. Then $(\alpha@\mathsf{r}) \cdot \eta_0 < (\alpha@\mathsf{r}) \cdot \eta_\mathsf{r} = \eta$. Since E is a causal set for ν in $\mathcal{NE}(\mathsf{N})$, this implies $\mathsf{r} :: (\alpha@\mathsf{r}) \cdot \eta_0 \Subset E$. Hence $\mathsf{r} :: \eta_0 \Subset \mathsf{post}(\alpha, E, \nu)$.
As for *minimality*, we first show that $\nu' \prec \alpha \blacklozenge \nu$ for all $\nu' \in \mathsf{post}(\alpha, E, \nu)$. If $\nu_1 \in \mathsf{post}(\alpha, E, \nu)$, then there exists $\nu_1' \in E$ such that $\nu_1 = \alpha \blacklozenge \nu_1'$. Since E is a causal set for ν, we have $\nu_1' \prec \nu$. Therefore $\nu_1 = \alpha \lozenge \nu_1' \prec \alpha \lozenge \nu$ by Lemma 8.2(3). Assume now that $\mathsf{post}(\alpha, E, \nu)$ is not minimal. Then there is $E' \subset \mathsf{post}(\alpha, E, \nu)$ that verifies Condition (2) of Definition 5.9 for $\alpha \blacklozenge \nu$. Let $\nu' \in \mathsf{post}(\alpha, E, \nu) \setminus E'$. Then $\nu' \prec \alpha \blacklozenge \nu = \{\mathsf{r} :: \eta_\mathsf{r}, \mathsf{s} :: \eta_\mathsf{s}\}$. Assume that $\mathsf{r} :: \eta_\mathsf{r}' \in \nu'$ with $\eta_\mathsf{r}' < \eta_\mathsf{r}$ (the proof is similar for s). By Condition (2), there is $\nu'' \in E'$ such that $\mathsf{r} :: \eta_\mathsf{r}' \in \nu''$. But then $\nu' \# \nu''$ by Lemma 5.21, contradicting the fact that $\mathsf{post}(\alpha, E, \nu)$ verifies Condition (1). Therefore $\mathsf{post}(\alpha, E, \nu)$ is minimal.

E. Proofs of Subsection 8.2

This section contains the proofs of Lemmas 8.10, 8.11 and 8.12.

Lemma 8.10 (Properties of retrieval and residual for g-events).

1. *If $\alpha \bullet \gamma$ is defined, then $\alpha \circ (\alpha \bullet \gamma) = \gamma$;*
2. *$\alpha \bullet (\alpha \circ \gamma) = \gamma$;*
3. *If $\gamma_1 < \gamma_2$, then $\alpha \circ \gamma_1 < \alpha \circ \gamma_2$;*
4. *If $\gamma_1 < \gamma_2$ and both $\alpha \bullet \gamma_1$ and $\alpha \bullet \gamma_2$ are defined, then $\alpha \bullet \gamma_1 < \alpha \bullet \gamma_2$;*
5. *If $\gamma_1 \# \gamma_2$, then $\alpha \circ \gamma_1 \# \alpha \circ \gamma_2$;*
6. *If $\gamma < \alpha \circ \gamma'$, then either $\gamma = [\alpha]_\sim$ or $\alpha \bullet \gamma < \gamma'$;*
7. *If $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$, then $\alpha_1 \circ (\alpha_2 \circ \gamma) = \alpha_2 \circ (\alpha_1 \circ \gamma)$;*
8. *If $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$ and both $\alpha_2 \bullet (\alpha_1 \circ \gamma)$, $\alpha_2 \bullet \gamma$ are defined, then $\alpha_1 \circ (\alpha_2 \bullet \gamma) = \alpha_2 \bullet (\alpha_1 \circ \gamma)$.*

Proof (1) If $\alpha \bullet [\sigma]_\sim$ is defined, then in case $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \emptyset$ we get $\alpha \bullet [\sigma]_\sim = [\sigma]_\sim$ and also $\alpha \circ [\sigma]_\sim = [\sigma]_\sim$, so $\alpha \circ (\alpha \bullet [\sigma]_\sim) = [\sigma]_\sim$. Instead if $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$, then $\alpha \bullet [\sigma]_\sim = [\sigma']_\sim$ where $\sigma \sim \alpha \cdot \sigma'$ and $\sigma' \neq \epsilon$. From $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$ we get $\alpha \circ [\sigma']_\sim = [\alpha \cdot \sigma']_\sim$ by Definition 7.6. This implies $\alpha \circ (\alpha \bullet [\sigma]_\sim) = [\sigma]_\sim$.

(2) By Definition 7.6 either $\alpha \circ [\sigma]_\sim = [\alpha \cdot \sigma]_\sim$ if $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$, or $\alpha \circ \sigma = [\sigma]_\sim$. In the first case $\alpha \bullet [\alpha \cdot \sigma]_\sim = [\sigma]_\sim$ and in the second $\alpha \bullet [\sigma]_\sim = [\sigma]_\sim$, which proves the result.

(3) Let $\gamma_1 = [\sigma]_\sim$ and $\gamma_2 = [\sigma \cdot \sigma']_\sim$. If $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$, then $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma \cdot \sigma') \neq \emptyset$, and we have $\alpha \circ \gamma_1 = [\alpha \cdot \sigma]_\sim$ and $\alpha \circ \gamma_2 = [\alpha \cdot \sigma \cdot \sigma']_\sim$. Whence $\alpha \circ \gamma_1 \leq \alpha \circ \gamma_2$. Suppose now $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \emptyset$. Then $\alpha \circ \gamma_1 = [\sigma]_\sim = \gamma_1$. If also $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma') = \emptyset$, then $\alpha \circ \gamma_2 = [\sigma \cdot \sigma]_\sim = \gamma_2$ and we are done. If instead $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma') \neq \emptyset$, then $\alpha \circ \gamma_2 = [\alpha \cdot \sigma \cdot \sigma']_\sim = [\sigma \cdot \alpha \cdot \sigma']_\sim$, whence $\gamma_1 \leq \alpha \circ \gamma_2$.

(4) Let $\gamma_1 = [\sigma]_\sim$ and $\gamma_2 = [\sigma \cdot \sigma']_\sim$. If $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \mathsf{part}(\alpha) \cap \mathsf{part}(\sigma \cdot \sigma') = \emptyset$, then $\alpha \bullet \gamma_1 = \gamma_1$ and $\alpha \bullet \gamma_2 = \gamma_2$. If $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$, then $\sigma \sim \alpha \cdot \sigma_0$, which implies $\alpha \bullet \gamma_1 = [\sigma_0]_\sim$ and $\alpha \bullet \gamma_2 = [\sigma_0 \cdot \sigma']_\sim$. If $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \emptyset$ and $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma \cdot \sigma') \neq \emptyset$, then $\alpha \bullet \gamma_1 = [\sigma]_\sim$ and $\sigma' \sim \alpha \cdot \sigma_0$, which implies $\alpha \bullet \gamma_2 = [\sigma \cdot \sigma_0]_\sim$.

(5) Let $\gamma_1 = [\sigma]_\sim$ and $\gamma_2 = [\sigma']_\sim$ and $\sigma @ \mathsf{p} \# \sigma' @ \mathsf{p}$ for some p. The only interesting case is $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \emptyset$ and $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma') \neq \emptyset$. This implies $\alpha \circ \gamma_1 = [\sigma]_\sim$ and $\alpha \circ \gamma_2 = [\alpha \cdot \sigma']_\sim$. We get $(\alpha \cdot \sigma') @ \mathsf{p} = \sigma' @ \mathsf{p}$ since $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) = \emptyset$ implies $\mathsf{p} \notin \mathsf{part}(\alpha)$. We conclude $\alpha \circ \gamma_1 \# \alpha \circ \gamma_2$.

(6) The case $\gamma = [\alpha]_\sim$ is immediate. If $\alpha \bullet \gamma$ is defined, we get $\alpha \bullet \gamma < \alpha \bullet (\alpha \circ \gamma')$ by Point 4 and $\alpha \bullet (\alpha \circ \gamma') = \gamma'$ by Point 2. Otherwise let $\gamma = [\sigma]_\sim$ and $\alpha \circ \gamma' = [\sigma \cdot \sigma']_\sim$. From $\alpha \bullet \gamma$ undefined we get $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$ and $\sigma \nsim \alpha \cdot \sigma_0$. Since $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma) \neq \emptyset$ implies $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma \cdot \sigma') \neq \emptyset$ we get $\sigma \cdot \sigma' \sim \alpha \cdot \sigma_1$ for some σ_1 by Definition 7.6(1). Then this case is impossible.

(7) Let $\gamma = [\sigma]_\sim$. By Definition 7.6(1) we have four cases:

(a) $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\alpha_1 \cdot (\alpha_2 \cdot \sigma)]_\sim = [\alpha_2 \cdot (\alpha_1 \cdot \sigma)]_\sim = \alpha_2 \circ (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) \neq \emptyset$ and $\mathsf{part}(\alpha_2) \cap \mathsf{part}(\sigma) \neq \emptyset$, since $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\alpha_2) = \emptyset$;

(b) $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\alpha_1 \cdot \sigma]_\sim = \alpha_2 \circ (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) \neq \emptyset$ and $\mathsf{part}(\alpha_2) \cap \mathsf{part}(\sigma) = \emptyset$;

(c) $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\alpha_2 \cdot \sigma]_\sim = \alpha_2 \circ (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) = \emptyset$ and $\mathsf{part}(\alpha_2) \cap \mathsf{part}(\sigma) \neq \emptyset$;

(d) $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\sigma]_\sim = \alpha_2 \circ (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) = \emptyset$ and $\mathsf{part}(\alpha_2) \cap \mathsf{part}(\sigma) = \emptyset$.

(8) Let $\gamma = [\sigma]_\sim$. By Definitions 7.6(1) and 8.9(1) we have four cases:

(a) $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\alpha_1 \cdot \sigma']_\sim = \alpha_2 \bullet (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) \neq \emptyset$ and $\sigma \sim \alpha_2 \cdot \sigma'$;

(b) $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\alpha_1 \cdot \sigma]_\sim = \alpha_2 \bullet (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) \neq \emptyset$ and $\mathsf{part}(\alpha_2) \cap \mathsf{part}(\sigma) = \emptyset$;

(c) $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\sigma']_\sim = \alpha_2 \bullet (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) = \emptyset$ and $\sigma \sim \alpha_2 \cdot \sigma'$;

(d) $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\sigma]_\sim = \alpha_2 \bullet (\alpha_1 \circ \sigma)$ if $\mathsf{part}(\alpha_1) \cap \mathsf{part}(\sigma) = \emptyset$ and $\mathsf{part}(\alpha_2) \cap \mathsf{part}(\sigma) = \emptyset$.

Lemma 8.11 *The following hold:*

1. *If $\gamma \in \mathcal{GE}(\mathsf{G})$, then $\mathsf{pq}\lambda \circ \gamma \in \mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i)$, where $\lambda = \lambda_k$ and $\mathsf{G} = \mathsf{G}_k$ for some $k \in I$;*

2. *If $\gamma \in \mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i)$ and $\mathsf{pq}\lambda_k \bullet \gamma$ is defined, then $\mathsf{pq}\lambda_k \bullet \gamma \in \mathcal{GE}(\mathsf{G}_k)$, where $k \in I$.*

Proof (1) By Definition 7.11(1) $\gamma \in \mathcal{GE}(\mathsf{G})$ implies $\gamma = \mathsf{ev}(\sigma)$ for some $\sigma \in \mathsf{Tr}^+(\mathsf{G})$. Since $\mathsf{pq}\lambda\circ\gamma = \mathsf{ev}(\mathsf{pq}\lambda\cdot\sigma)$ by Definitions 7.6, 7.7 and $\mathsf{pq}\lambda\cdot\sigma \in \mathsf{Tr}^+(\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i)$ we conclude $\mathsf{pq}\lambda \circ \gamma \in \mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i)$ by Definition 7.11(1).

(2) By Definition 7.11(1) $\gamma \in \mathcal{GE}(\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i)$ implies $\gamma = \mathsf{ev}(\sigma)$ for some $\sigma \in \mathsf{Tr}^+(\mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i)$. We get $\sigma = \mathsf{pq}\lambda_h\cdot\sigma'$ with $\sigma' \in \mathsf{Tr}^+(\mathsf{G}_h)$ or $\sigma' = \epsilon$ for some $h \in I$. The hypothesis $\mathsf{pq}\lambda_k \bullet \gamma$ defined implies either $h = k$ and $\sigma' \neq \epsilon$ or $\mathsf{part}(\sigma') \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$ and $\mathsf{pq}\lambda_k \bullet \gamma = \mathsf{ev}(\sigma')$ by Definition 8.9(1). In the first case $\sigma' \in \mathsf{Tr}^+(\mathsf{G}_k)$. In the second case $\sigma'' \in \mathsf{Tr}^+(\mathsf{G}_k)$ for some $\sigma'' \sim \sigma'$ by definition of projection, which prescribes the same behaviours to all participants different from p, q, see Figure 2. We conclude $\mathsf{pq}\lambda_k \bullet \gamma \in \mathcal{GE}(\mathsf{G}_k)$ by Definition 7.11(1).

Lemma 8.12 *Let $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$.*

1. *If $\gamma \in \mathcal{GE}(\mathsf{G}')$, then $\alpha \circ \gamma \in \mathcal{GE}(\mathsf{G})$;*

2. *If $\gamma \in \mathcal{GE}(\mathsf{G})$ and $\alpha \bullet \gamma$ is defined, then $\alpha \bullet \gamma \in \mathcal{GE}(\mathsf{G}')$.*

Proof Both proofs are by induction on the inference of the transition $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$, see Figure 4.

(1) For rule [Ecomm] we get $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\mathsf{G}' = \mathsf{G}_k$ and $\alpha = \mathsf{pq}\lambda_k$ for some $k \in I$. We conclude $\alpha \circ \gamma \in \mathcal{GE}(\mathsf{G})$ by Lemma 8.11(1).
For rule [Icomm] we get $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\mathsf{G}' = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}'_i$ and $\mathsf{G}_i \xrightarrow{\alpha} \mathsf{G}'_i$ for all $i \in I$ and $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. By Definition 7.11(1) $\gamma \in \mathcal{GE}(\mathsf{G}')$ implies $\gamma = \mathsf{ev}(\sigma)$ for some $\sigma \in \mathsf{Tr}^+(\mathsf{G}')$. This implies $\sigma = \mathsf{pq}\lambda_k\cdot\sigma'$ and $\gamma = [\sigma_0]_\sim$ with either $\sigma_0 \sim \mathsf{pq}\lambda_k\cdot\sigma'_0$ for some $k \in I$ or $\mathsf{part}(\sigma_0) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$ by Definition 7.6. Then $\mathsf{pq}\lambda_k \bullet \gamma$ is defined unless $\sigma_0 = \mathsf{pq}\lambda_k$ by Definition 8.9(1). We consider two cases.
If $\sigma_0 = \mathsf{pq}\lambda_k$, then $\alpha\circ\gamma = [\mathsf{pq}\lambda_k]_\sim$ since $\mathsf{part}(\alpha)\cap\{\mathsf{p}, \mathsf{q}\} = \emptyset$. We conclude $\alpha\circ\gamma \in \mathcal{GE}(\mathsf{G})$ by Definition 7.11(1). Otherwise let $\gamma' = \mathsf{pq}\lambda_k \bullet \gamma$. By Lemma 8.11(2) $\gamma' \in \mathcal{GE}(\mathsf{G}'_k)$. By induction $\alpha \circ \gamma' \in \mathcal{GE}(\mathsf{G}_k)$. By Lemma 8.11(1) $\mathsf{pq}\lambda_k \circ (\alpha \circ \gamma') \in \mathcal{GE}(\mathsf{G})$. We now show that $\mathsf{pq}\lambda_k \circ (\alpha \circ \gamma') = \alpha \circ \gamma$. By Lemma 8.10(7) and $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$ we get $\mathsf{pq}\lambda_k \circ (\alpha \circ \gamma') = \alpha \circ (\mathsf{pq}\lambda_k \circ \gamma')$ and by Lemma 8.10(1) we have $\mathsf{pq}\lambda_k \circ \gamma' = \mathsf{pq}\lambda_k \circ (\mathsf{pq}\lambda_k \bullet \gamma) = \gamma$. Therefore $\mathsf{pq}\lambda_k \circ (\alpha \circ \gamma') = \alpha \circ \gamma \in \mathcal{GE}(\mathsf{G})$.

(2) For rule [Ecomm] we get $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\mathsf{G}' = \mathsf{G}_k$ and $\alpha = \mathsf{pq}\lambda_k$ for some $k \in I$. We conclude $\alpha \bullet \gamma \in \mathcal{GE}(\mathsf{G}')$ by Lemma 8.11(2).
For rule [Icomm] we get $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}_i$ and $\mathsf{G} = \mathsf{p} \to \mathsf{q} : \boxplus_{i\in I}\lambda_i; \mathsf{G}'_i$ and $\mathsf{G}_i \xrightarrow{\alpha} \mathsf{G}'_i$ for all $i \in I$ and $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. By Definition 7.11(1) $\gamma \in \mathcal{GE}(\mathsf{G})$ implies $\gamma = \mathsf{ev}(\sigma)$ for some $\sigma \in \mathsf{Tr}^+(\mathsf{G})$. This implies $\sigma = \mathsf{pq}\lambda_k\cdot\sigma'$ and $\gamma = [\sigma_0]_\sim$ with either $\sigma_0 \sim \mathsf{pq}\lambda_k\cdot\sigma'_0$ for some $k \in I$ or $\mathsf{part}(\sigma_0) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$ by Definition 7.6. Then $\mathsf{pq}\lambda_k \bullet \gamma$ is defined unless $\sigma_0 = \mathsf{pq}\lambda_k$ by Definition 8.9(1). We consider two cases.
If $\sigma_0 = \mathsf{pq}\lambda_k$, then $\alpha\bullet\gamma = [\mathsf{pq}\lambda_k]_\sim$ since $\mathsf{part}(\alpha)\cap\{\mathsf{p}, \mathsf{q}\} = \emptyset$. We conclude $\alpha\bullet\gamma \in \mathcal{GE}(\mathsf{G}')$ by Definition 7.11(1). Otherwise let $\gamma' = \mathsf{pq}\lambda_k \bullet \gamma$. By Lemma 8.11(2) $\gamma' \in \mathcal{GE}(\mathsf{G}_k)$.

We first show that $\alpha \bullet \gamma'$ is defined. Since $\alpha \bullet \gamma$ and $\mathsf{pq}\lambda_k \bullet \gamma$ are defined, by Definition 8.9(1) we have four cases:

(a) $\sigma_0 \sim \alpha \cdot \sigma_1$ for some σ_1 and $\sigma_0 \sim \mathsf{pq}\lambda_k \cdot \sigma_0'$;

(b) $\sigma_0 \sim \alpha \cdot \sigma_1$ and $\mathsf{part}(\sigma_0) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$;

(c) $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma_0) = \emptyset$ and $\sigma_0 \sim \mathsf{pq}\lambda_k \cdot \sigma_0'$;

(d) $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma_0) = \emptyset$ and $\mathsf{part}(\sigma_0) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$.

In case (a) $\sigma_0 \sim \alpha \cdot \mathsf{pq}\lambda_k \cdot \sigma_1' \sim \mathsf{pq}\lambda_k \cdot \alpha \cdot \sigma_1'$ for some σ_1' since $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. Notice that $\sigma_1' \neq \epsilon$ since σ_0 is pointed and $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$. We get $\gamma' = \mathsf{pq}\lambda_k \bullet \gamma = [\alpha \cdot \sigma_1']_\sim$ and $\alpha \bullet \gamma' = [\sigma_1']_\sim$.
In case (b) $\gamma' = \gamma$ and $\alpha \bullet \gamma' = [\sigma_1]_\sim$.
In case (c) $\gamma' = [\sigma_0']_\sim$ and $\alpha \bullet \gamma' = [\sigma_0']_\sim$, since $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma_0) = \emptyset$ implies $\mathsf{part}(\alpha) \cap \mathsf{part}(\sigma_0') = \emptyset$.
In case (d) $\gamma' = \gamma$ and $\alpha \bullet \gamma' = \gamma$.
By induction $\alpha \bullet \gamma' \in \mathcal{GE}(\mathsf{G}_k')$. By Lemma 8.11(1) $\mathsf{pq}\lambda_k \circ (\alpha \bullet \gamma') \in \mathcal{GE}(\mathsf{G}')$.

We now show that $\mathsf{pq}\lambda_k \circ (\alpha \bullet \gamma') = \alpha \bullet \gamma$. From $\gamma' = \mathsf{pq}\lambda_k \bullet \gamma$ and Lemma 8.10(1) $\mathsf{pq}\lambda_k \circ \gamma' = \gamma$. Therefore from $\alpha \bullet \gamma$ defined we have $\alpha \bullet (\mathsf{pq}\lambda_k \circ \gamma')$ defined. Since $\alpha \bullet \gamma'$ is also defined and $\mathsf{part}(\alpha) \cap \{\mathsf{p}, \mathsf{q}\} = \emptyset$, by Lemma 8.10(8) we get $\mathsf{pq}\lambda_k \circ (\alpha \bullet \gamma') = \alpha \bullet (\mathsf{pq}\lambda_k \circ \gamma')$. Therefore $\mathsf{pq}\lambda_k \circ (\alpha \bullet \gamma') = \alpha \bullet \gamma \in \mathcal{GE}(\mathsf{G}')$.

F. Glossary of Symbols and Table of Notations

Symbol	Meaning
π	input/output action: $\mathsf{p}!\lambda$, $\mathsf{p}?\lambda$
α	communication $\mathsf{pq}\lambda$
σ	trace, finite sequence of communications
S	event structure
$\mathcal{X}$	configuration of an event structure
η	p-event, non-empty finite sequence of input/output actions
$\mathcal{PE}$	set of p-events
ζ	(possibly empty) finite sequence of input/output actions
ϑ	undirected action: $!\lambda$, $?\lambda$
Θ	(possibly empty) finite sequence of undirected actions
ν	n-event, unordered pair of dual located p-events
$\mathcal{NE}$	set of n-events
γ	g-event, equivalence class $[\sigma]_\sim$ with σ pointed
$\mathcal{GE}$	set of g-events

Notation	Meaning	Where Defined
$\mathsf{pt}(\pi)$	participant of action π	before Def. 2.1
$\mathsf{part}(\sigma)$	participants of trace σ	Def. 2.3
$\mathcal{D}(S)$	domain of configurations of ES S	Def. 3.5
$\mathsf{act}(\eta)$	action of p-event η	after Def. 4.1
$\mathcal{S}^{\mathcal{P}}(P)$	event structure of process P	Def. 4.3
$\mathcal{PE}(P)$	set of p-events of $\mathcal{S}^{\mathcal{P}}(P)$	Def. 4.3
$\mathsf{p} :: \eta$	located event, p-event η located at participant p	Def. 5.1
$\eta \upharpoonright \mathsf{p}$	projection of p-event η on participant p	Def. 5.2
$\Theta \bowtie \Theta'$	duality of undirected action sequences Θ and Θ'	Def. 5.3
$\mathsf{p} :: \eta \ \widehat{\bowtie}\ \mathsf{q} :: \eta'$	duality of located events $\mathsf{p} :: \eta$ and $\mathsf{q} :: \eta'$	Def. 5.4
$\mathsf{cm}(\nu)$	communication of n-event ν	after Def. 5.5
$\mathsf{loc}(\nu)$	set of locations of n-event ν	after Def. 5.5
$\mathsf{p} :: \eta \in\!\in E$	occurrence of located event $\mathsf{p} :: \eta$ in some n-event of E	Def. 5.6
$\mathsf{n}(E)$	narrowing of the n-event set E	Def. 5.11
$\mathcal{S}^{\mathcal{N}}(\mathsf{N})$	event structure of network N	Def. 5.13
$\mathcal{CE}(\mathsf{N})$	set of candidate n-events of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$	Def. 5.13
$\mathcal{NE}(\mathsf{N})$	set of n-events of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$	Def. 5.13
$\vartheta \searrow n$	prefix of length n of ϑ	before Prop. 5.22
$proj_{\mathsf{p}}(\nu)$	projection of n-event ν on participant p	Def. 5.25
$\mathsf{part}(\mathsf{G})$	participants of global type G	after Def. 6.1
$\mathsf{G} \upharpoonright \mathsf{p}$	projection of global type G on participant p	Figure 2
$\sigma[i]$	i-th element of trace σ	before Def. 7.1
$\sigma[i \dots j]$	subtrace $\sigma[i] \cdots \sigma[j]$ of trace σ	before Def. 7.1
$\sigma \sim \sigma'$	permutation equivalence of traces	Def. 7.1
$[\sigma]_{\sim}$	equivalence class of trace σ w.r.t $\sim$	Def. 7.1
$\mathsf{last}(\sigma)$	last communication of trace σ	before Lemma 7.4
$\mathsf{cm}(\gamma)$	communication of g-event γ	Def. 7.5
$\sigma \circ \gamma$	retrieval of g-event γ before trace σ	Def. 7.6(1) and (2)
$\mathsf{ev}(\sigma)$	g-event generated by trace σ	Def. 7.7
$\sigma @ \mathsf{p}$	projection of trace σ on participant p	Def. 7.9(1) and (2)
$\mathcal{S}^{\mathcal{G}}(\mathsf{G})$	event structure of global type G	Def. 7.11
$\mathcal{GE}(\mathsf{G})$	set of g-events of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$	Def. 7.11
$\sigma \lozenge \nu$	retrieval of n-event ν before trace σ	Def. 8.1(1) and (3)
$\sigma \blacklozenge \nu$	residual of n-event ν after trace σ	Def. 8.1(2) and (3)
$\mathsf{nec}(\sigma)$	sequence of n-events corresponding to trace σ	Def. 8.3
$\sigma \bullet \gamma$	residual of g-event γ after trace σ	Def. 8.9(1) and (2)
$\mathsf{gec}(\sigma)$	sequence of g-events corresponding to trace σ	Def. 8.13