



## Event structure semantics for multiparty sessions

Ilaria Castellani, Mariangiola Dezani-Ciancaglini, Paola Giannini

### ► To cite this version:

Ilaria Castellani, Mariangiola Dezani-Ciancaglini, Paola Giannini. Event structure semantics for multiparty sessions. *Journal of Logical and Algebraic Methods in Programming*, 2023, 131, 10.1016/j.jlamp.2022.100844 . hal-03940191

**HAL Id: hal-03940191**

**<https://inria.hal.science/hal-03940191>**

Submitted on 28 Jan 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

# Event Structure Semantics for Multiparty Sessions

Ilaria Castellani<sup>a,1,\*</sup>, Mariangiola Dezani-Ciancaglini<sup>b</sup>, Paola Giannini<sup>c,2</sup>

<sup>a</sup>INRIA, Université Côte d'Azur, France

<sup>b</sup>Dipartimento di Informatica, Università di Torino, Italy

<sup>c</sup>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  $\pi$ -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](mailto:ilaria.castellani@inria.fr) (Ilaria Castellani), [dezani@di.unito.it](mailto:dezani@di.unito.it) (Mariangiola Dezani-Ciancaglini), [paola.giannini@uniupo.it](mailto:paola.giannini@uniupo.it) (Paola Giannini)

<sup>1</sup>This work was partially funded by the ANR17-CE25-0014-01 CISC project.

<sup>2</sup>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<sup>3</sup>, as well as handling examples and carrying out

---

<sup>3</sup>Both FESs and PESs enjoy a graphical representation (see Figure 5 and Figure 6), as opposed to

63 proofs.

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

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

74 The paper is organised as follows. Section 2 introduces our multiparty session  
 75 calculus. In Section 3 we recall the definitions of PESs and FESs, which will be used  
 76 to interpret processes (Section 4) and networks (Section 5), respectively. PESs are  
 77 also used to interpret global types (Section 7), which are defined in Section 6. In  
 78 Section 8 we prove the equivalence between the FES semantics of a network and  
 79 the PES semantics of its global type. Section 9 discusses related work and sketches  
 80 directions for future work.

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

## 87 2. A Core Calculus for Multiparty Sessions

88 We now formally introduce our calculus, where multiparty sessions are rep-  
 89 resented as networks of processes. We assume the following base sets: *session*  
 90 *participants*, ranged over by  $p, q, r, \dots$  and forming the set  $\mathbf{Part}$ , and *messages*, ranged  
 91 over by  $\lambda, \lambda', \dots$  and forming the set  $\mathbf{Msg}$ .

92 Let  $\pi \in \{p!\lambda, p?\lambda \mid p \in \mathbf{Part}, \lambda \in \mathbf{Msg}\}$  denote an *action*. The action  $p!\lambda$  represents  
 93 an output of message  $\lambda$  to participant  $p$ , while the action  $p?\lambda$  represents an input  
 94 of message  $\lambda$  from participant  $p$ . The *participant of an action*,  $\mathbf{pt}(\pi)$ , is defined by  
 95  $\mathbf{pt}(p!\lambda) = \mathbf{pt}(p?\lambda) = p$ .

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

$$P ::=^{coind} \bigoplus_{i \in I} p!\lambda_i; P_i \mid \sum_{i \in I} p?\lambda_i; P_i \mid 0$$

96 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  
 97 different.

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

---

other kinds of stable ESs.

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

112 Processes may be viewed as trees whose internal nodes are decorated by  $p!$  or  
 113  $p?$ , leaves by  $0$ , and edges by messages  $\lambda$ .

114 In a full-fledged calculus, messages would carry values, namely they would be  
 115 of the form  $\lambda(v)$ . For simplicity, we consider only pure messages here. This will  
 116 allow us to project global types directly to processes, without having to explicitly  
 117 introduce local types, see Section 6.

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

$$N = p \llbracket P \rrbracket \mid p \llbracket P \rrbracket \mid N$$

118 We assume the standard structural congruence  $\equiv$  on networks, stating that  
 119 parallel composition is associative and commutative and has neutral element  $p \llbracket 0 \rrbracket$   
 120 for any fresh  $p$ . Given the associativity of  $\parallel$ , we shall feel free to write networks in  
 121 the form  $N = p_1 \llbracket P_1 \rrbracket \parallel \dots \parallel p_n \llbracket P_n \rrbracket$  in the sequel.

122 If  $P \neq 0$  we write  $p \llbracket P \rrbracket \in N$  as short for  $N \equiv p \llbracket P \rrbracket \parallel N'$  for some  $N'$ . We define  
 123 the *set of participants of  $N$*  to be  $\{p \mid \exists P. p \llbracket P \rrbracket \in N\}$ . We say that a network is *unary* if  
 124 it has a unique participant<sup>4</sup> and *binary* if it has exactly two participants.

125 To express the operational semantics of networks, we use an LTS whose labels  
 126 record the message exchanged during a communication together with its sender  
 127 and receiver. The set of *communications*, ranged over by  $\alpha, \alpha'$ , is defined to be  
 128  $\{pq\lambda \mid p, q \in \text{Part}, \lambda \in \text{Msg}\}$ , where  $pq\lambda$  represents the transmission of a message  $\lambda$   
 129 from participant  $p$  to participant  $q$ .

$$p \llbracket \bigoplus_{i \in I} q! \lambda_i; P_i \rrbracket \parallel q \llbracket \sum_{j \in J} p? \lambda_j; Q_j \rrbracket \parallel N \xrightarrow{pq\lambda_k} p \llbracket P_k \rrbracket \parallel q \llbracket Q_k \rrbracket \parallel N \quad \text{where } k \in I \cap J \quad [\text{Com}]$$

**Figure 1:** LTS for networks.

130 The LTS semantics of networks is specified by the unique rule [Com] given in  
 131 Figure 1. Notice that rule [Com] is symmetric with respect to input and output

<sup>4</sup>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.

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

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

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

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

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

141 We use  $|\sigma|$  to denote the length of the trace  $\sigma$ .

142 The set of participants of a trace, notation  $\text{part}(\sigma)$ , is defined by  $\text{part}(\epsilon) = \emptyset$  and  
 143  $\text{part}(pq\lambda \cdot \sigma) = \{p, q\} \cup \text{part}(\sigma)$ .

144 When  $\sigma = \alpha_1 \cdot \dots \cdot \alpha_n$  ( $n \geq 1$ ) we write  $N \xrightarrow{\sigma} N'$  as short for  $N \xrightarrow{\alpha_1} N_1 \cdot \dots \xrightarrow{\alpha_n} N_n = N'$ .

### 145 3. Event Structures

146 We recall now the definitions of *Prime Event Structure* (PES) from [60, 49] and  
 147 *Flow Event Structure* (FES) from [8]. The class of FESs is more general than that  
 148 of PESs: for a precise comparison of various classes of event structures, we refer  
 149 the reader to [9]. As we shall see in Sections 4 and 5, while PESs are sufficient to  
 150 interpret processes, the greater generality of FESs is needed to interpret networks.

151 **Definition 3.1 (Prime Event Structure).** A prime event structure (PES) is a tuple  $S =$   
 152  $(E, \leq, \#)$  where:

- 153 1.  $E$  is a denumerable set of events;
- 154 2.  $\leq \subseteq (E \times E)$  is a partial order relation, called the causality relation;
- 155 3.  $\# \subseteq (E \times E)$  is an irreflexive symmetric relation, called the conflict relation, satisfying  
 156 the property:  $\forall e, e', e'' \in E : e \# e' \leq e'' \Rightarrow e \# e''$  (conflict hereditariness).

157 **Definition 3.2 (Flow Event Structure).** A flow event structure (FES) is a tuple  $S =$   
 158  $(E, <, \#)$  where:

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

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

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

170 We now recall the definition of *configuration* for event structures. Intuitively, a  
 171 configuration is a set of events having occurred at some stage of the computation.  
 172 Thus, the semantics of an event structure  $S$  is given by its poset of configurations  
 173 ordered by set inclusion, where  $X_1 \subset X_2$  means that  $S$  may evolve from  $X_1$  to  $X_2$ .

174 **Definition 3.3 (PES configuration).** Let  $S = (E, \leq, \#)$  be a prime event structure. A  
 175 configuration of  $S$  is a finite subset  $X$  of  $E$  such that:

- 176 1.  $X$  is downward-closed:  $e' \leq e \in X \Rightarrow e' \in X$ ;
- 177 2.  $X$  is conflict-free:  $\forall e, e' \in X, \neg(e \# e')$ .

178 The definition of configuration for FESs is slightly more elaborated. For a subset  $X$   
 179 of  $E$ , let  $<_X$  be the restriction of the flow relation to  $X$  and  $<_X^*$  be its transitive and  
 180 reflexive closure.

181 **Definition 3.4 (FES configuration).** Let  $S = (E, <, \#)$  be a flow event structure. A  
 182 configuration of  $S$  is a finite subset  $X$  of  $E$  such that:

- 183 1.  $X$  is downward-closed up to conflicts:  $e' < e \in X, e' \notin X \Rightarrow \exists e'' \in X. e' \# e'' < e$ ;
- 184 2.  $X$  is conflict-free:  $\forall e, e' \in X, \neg(e \# e')$ ;
- 185 3.  $X$  has no causality cycles: the relation  $<_X^*$  is a partial order.

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

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

193 **Definition 3.5 (ES configuration domain).** Let  $S$  be a prime or flow event structure  
 194 with set of configurations  $C(S)$ . The domain of configurations of  $S$  is the partially ordered  
 195 set  $\mathcal{D}(S) =_{\text{def}} (C(S), \subseteq)$ .

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

**Definition 3.6 (Proving sequence).** Given a flow event structure  $S = (E, <, \#)$ , a proving sequence in  $S$  is a sequence  $e_1; \dots; e_n$  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:

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

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

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

201 **Proposition 3.7 (Representation of FES configurations as proving sequences [9]).**

202 Given a flow event structure  $S = (E, <, \#)$ , a subset  $X$  of  $E$  is a configuration of  $S$  if and  
203 only if it can be enumerated as a proving sequence  $e_1; \dots; e_n$ .

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

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

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

**Definition 3.8 (Downward surjectivity).** Let  $S_i = (E_i, <_i, \#_i)$ , 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:

$$e_1 <_1 f(e_0) \Rightarrow \exists e'_0 \in E_0. e_1 = f(e'_0)$$

## 208 4. Event Structure Semantics of Processes

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

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

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

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



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

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

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

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

- 223 1.  $\eta \sqsubseteq \eta' \Rightarrow \eta \leq \eta'$ ;
- 224 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

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

225 where:

- 226 1.  $\mathcal{PE}(P) \subseteq \mathcal{PE}$  is the set of non-empty sequences of labels along the nodes and edges of  
 227 a path from the root to an edge in the tree of  $P$ ;
- 228 2.  $\leq_P$  is the restriction of  $\leq$  to the set  $\mathcal{PE}(P)$ ;
- 229 3.  $\#_P$  is the restriction of  $\#$  to the set  $\mathcal{PE}(P)$ .

230 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  
 231 feel free to drop the subscript in  $\leq_P$  and  $\#_P$ .

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

233 **Example 4.4.** If  $P = q!\lambda; P \oplus q!\lambda'$ , then  $\mathcal{PE}(P) = \underbrace{\{q!\lambda \cdot \dots \cdot q!\lambda \mid n \geq 1\}}_n \cup$   
 $\underbrace{\{q!\lambda \cdot \dots \cdot q!\lambda \cdot q!\lambda' \mid n \geq 0\}}_n$

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

235 **Proof** We show that  $\leq$  and  $\#$  satisfy Properties 2 and 3 of Definition 3.1. Reflexivity,  
 236 transitivity and antisymmetry of  $\leq$  follow from the corresponding properties of  $\sqsubseteq$ .  
 237 As for irreflexivity and symmetry of  $\#$ , they follow from Clause 2 of Definition  
 238 4.2 and the corresponding properties of inequality. To show conflict hereditariness,  
 239 suppose that  $\eta \# \eta' \leq \eta''$ . From Clause 2 of Definition 4.2 there are  $\pi, \pi', \zeta, \zeta'$  and  
 240  $\zeta''$  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  
 241  $\eta'' = \zeta \cdot \pi' \cdot \zeta'' \cdot \zeta_1$  for some  $\zeta_1$ . 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  $\eta$  pertaining to a participant  $p$ , written  $p :: \eta$ .

As hinted above, network events should be pairs of dual located events  $p :: \zeta \cdot \pi$  and  $q :: \zeta' \cdot \pi'$  with matching actions  $\pi$  and  $\pi'$  and matching histories  $\zeta$  and  $\zeta'$ . 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  $\epsilon$ . Then we introduce a notion of duality between located events, based on a notion of duality between undirected actions.

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

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

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

$$(\pi \cdot \eta) \upharpoonright p = \pi \upharpoonright p \cdot \eta \upharpoonright p$$

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

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

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

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

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

$$v ::= \{p :: \eta, q :: \eta'\} \quad \text{where } p :: \eta \widehat{\bowtie} q :: \eta'$$

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

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

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

284 **Definition 5.6.** A located event  $p :: \eta$  occurs in a set  $E$  of n-events, notation  $p :: \eta \in E$ ,  
 285 if  $p :: \eta \in v$  and  $v \in E$  for some  $v$ .

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

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

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

293 1.  $v < v'$  if  $p :: \eta \in v$  &  $p :: \eta' \in v'$  &  $\eta < \eta'$ ;

294 2.  $v \# v'$  if

295 (a) either  $p :: \eta \in v$  &  $p :: \eta' \in v'$  &  $\eta \# \eta'$ ;

296 (b) or  $p :: \eta \in v$  &  $q :: \eta' \in v'$  &  $p \neq q$  &  $|\eta \dot{\bowtie} q| = |\eta' \dot{\bowtie} p|$  &  $\neg(\eta \dot{\bowtie} q \bowtie \eta' \dot{\bowtie} p)$ .

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

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

- 309 1. Let  $v = \{p :: q! \lambda_1 \cdot r! \lambda, r :: p? \lambda\}$  and  $v' = \{p :: q! \lambda_2, q :: p? \lambda_2\}$ . Then  $v \# v'$  by Clause  
 310 (2a) since  $q! \lambda_1 \cdot r! \lambda \# q! \lambda_2$ . Note that  $v \# v'$  can be also deduced by Clause (2b), since  
 311  $(q! \lambda_1 \cdot r! \lambda) \dot{p} q = ! \lambda_1$  and  $p? \lambda_2 \dot{p} p = ? \lambda_2$  and  $! \lambda_1 = |? \lambda_2|$  and  $\neg(! \lambda_1 \bowtie ? \lambda_2)$ .
- 312 2. Let  $v$  be as in (1) and  $v' = \{p :: q! \lambda_2 \cdot q! \lambda, q :: p? \lambda_2 \cdot p? \lambda\}$ . Again, we can deduce  
 313  $v \# v'$  using Clause (2a), since  $q! \lambda_1 \cdot r! \lambda \# q! \lambda_2 \cdot q! \lambda$ . On the other hand, Clause (2b)  
 314 does not apply in this case, since  $(q! \lambda_1 \cdot r! \lambda) \dot{p} q = ! \lambda_1$  and  $(p? \lambda_2 \cdot p? \lambda) \dot{p} p = ? \lambda_2 \cdot ? \lambda$   
 315 and thus  $! \lambda_1 \neq |? \lambda_2 \cdot ? \lambda|$ .
- 316 3. Let  $v$  be as in (1) and  $v' = \{q :: p? \lambda_2 \cdot s! \lambda, s :: q? \lambda\}$ . Here  $\text{loc}(v) \cap \text{loc}(v') = \emptyset$ ,  
 317 so clearly Clause (2a) does not apply. On the other hand,  $v \# v'$  can be deduced by  
 318 Clause (2b), since  $(q! \lambda_1 \cdot r! \lambda) \dot{p} q = ! \lambda_1$  and  $(p? \lambda_2 \cdot s! \lambda) \dot{p} p = ? \lambda_2$  and  $! \lambda_1 = |? \lambda_2|$   
 319 and  $\neg(! \lambda_1 \bowtie ? \lambda_2)$ . Consider now  $v'' = \{q :: p? \lambda_2 \cdot p? \lambda', s! \lambda, s :: q? \lambda\}$ . Then we  
 320 cannot deduce  $v \# v''$  in the same way because the two projections do not have the same  
 321 length. However, we can deduce  $v \# v''' < v''$ , where  $v''' = \{p :: q! \lambda_2, q :: p? \lambda_2\}$ .  
 322 In other words,  $v$  and  $v''$  are in semantic conflict, as Proposition 5.22 shows, but  
 323 not in the syntactic conflict  $\#$  (the fact that semantic conflict is in general larger  
 324 than syntactic conflict is common to all classes of ESs except PESs). We could have  
 325 chosen to make the syntactic conflict larger by replacing Clause (2b) by the following  
 326 alternative clause, where  $\Theta, \Theta'$  are as in Definition 5.3 and  $\sqsubseteq$  is the prefix ordering:  
 327 Clause (2b') or  $p :: \eta \in v \ \& \ q :: \eta' \in v' \ \& \ p \neq q \ \& \$   
 $(\exists \Theta \sqsubseteq \eta \dot{p} q, \exists \Theta' \sqsubseteq \eta' \dot{p} p \cdot |\Theta| = |\Theta'| \ \& \ \neg(\Theta \bowtie \Theta'))$
- 328 With this alternative clause, we could deduce the syntactic conflict  $v \# v''$ . However,  
 329 in Definition 5.7 we chose to keep our definition of  $\#$  stricter in order to have fewer  
 330 syntactic conflicts to handle in examples and proofs.
- 331 4. Let  $v$  be as in (1) and  $v' = \{p :: q! \lambda_2 \cdot r! \lambda \cdot r! \lambda', r :: p? \lambda \cdot p? \lambda'\}$ . In this case we have  
 332 both  $v < v'$  by Clause (1) and  $v \# v'$  by Clause (2a), namely, causality is inherited  
 333 from participant  $r$  and conflict from participant  $p$ .

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

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

- 339 1.  $E \cup \{v\}$  is conflict-free and
- 340 2.  $p :: \eta \in v$  and  $\eta' < \eta$  imply  $p :: \eta' \in E$ .

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

347 **Example 5.10.** Let  $v_1 = \{p :: q!\lambda_1 \cdot r!\lambda, r :: p?\lambda\}$  and  $v_2 = \{p :: q!\lambda_2 \cdot r!\lambda, r :: p?\lambda\}$ . Then  
 348 both  $\{v_1\}$  and  $\{v_2\}$  are causal sets of  $v = \{r :: p?\lambda \cdot s!\lambda', s :: r?\lambda'\}$  in  $Ev = \{v_1, v_2, v\}$ . Note  
 349 that  $v_1 \# v_2$  and that neither  $v_1$  nor  $v_2$  has a causal set in  $Ev$ .

350 Let us now consider also  $v'_1 = \{p :: q!\lambda_1, q :: p?\lambda_1\}$  and  $v'_2 = \{p :: q!\lambda_2, q :: p?\lambda_2\}$ .  
 351 Then  $v$  still has the same causal sets  $\{v_1\}$  and  $\{v_2\}$  in  $Ev' = \{v'_1, v'_2, v_1, v_2, v\}$ , while each  $v_i$ ,  
 352  $i = 1, 2$ , has the unique causal set  $\{v'_i\}$  in  $Ev'$ , and each  $v'_i$ ,  $i = 1, 2$ , has the empty causal set  
 353 in  $Ev'$ .

354 Finally,  $v$  has infinitely many causal sets in  $\mathcal{NE}$ . For instance, if for every natural  
 355 number  $n$  we let  $v_n = \{p :: q!\lambda_n \cdot r!\lambda, r :: p?\lambda\}$ , then each  $\{v_n\}$  is a causal set of  $v$  in  
 356  $\mathcal{NE}$ . Symmetrically, a causal set may cause infinitely many events in  $\mathcal{NE}$ . For instance,  
 357 the above causal sets  $\{v_1\}$  and  $\{v_2\}$  of  $v$  could also act as causal sets for any  $n$ -event  
 358  $v''_n = \{r :: p?\lambda \cdot s!\lambda_n, s :: r?\lambda_n\}$  or, assuming the set of participants to be denumerable, for  
 359 any event  $v'''_n = \{r :: p?\lambda \cdot s_n!\lambda', s_n :: r?\lambda'\}$ .

360 When defining the set of events of a network  $ES$ , we want to prune out all the  
 361  $n$ -events that do not have a causal set in the set itself. The reason is that such  
 362  $n$ -events should not happen in the event structure of a network, although, when  
 363 projected on their locations (see Definition 5.25), they would always give rise to  
 364  $p$ -events occurring in a configuration<sup>5</sup>. Example 5.14 should further clarify this  
 365 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  $n(E)$ , is the greatest fixpoint of the function  $f_E$  on sets of  $n$ -events defined by:

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

366 Note that we could not have taken  $n(E)$  to be the least fixpoint of  $f_E$  rather than  
 367 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 narrow-  
 ing. Let

$$\begin{aligned} v_1 &= \{r :: s?\lambda_1, s :: r!\lambda_1\} & v_2 &= \{r :: s?\lambda_2, s :: r!\lambda_2\} \\ v_3 &= \{p :: r?\lambda_1, r :: s?\lambda_1 \cdot p!\lambda_1\} & v_4 &= \{q :: s?\lambda_2, s :: r!\lambda_2 \cdot q!\lambda_2\} \\ v_5 &= \{p :: r?\lambda_1 \cdot q!\lambda, q :: s?\lambda_2 \cdot p?\lambda\} \end{aligned}$$

Then  $n(\{v_1, \dots, v_5\}) = \{v_1, \dots, v_4\}$ , because a causal set for  $v_5$  would need to contain both  
 $v_3$  and  $v_4$ , but this is not possible, since  $v_3 \# v_4$  by Clause (2b) of Definition 5.7. In fact  
 $(s?\lambda_1 \cdot p!\lambda_1) \dot{p} s = ?\lambda_1$  and  $(r!\lambda_2 \cdot q!\lambda_2) \dot{p} r = !\lambda_2$  and  $|\lambda_1| = |\lambda_2|$  and  $\neg(? \lambda_1 \bowtie ! \lambda_2)$ . Let

$$\begin{aligned} v_1 &= \{r :: s?\lambda_1, s :: r!\lambda_1\} & v_2 &= \{r :: s?\lambda_2, s :: r!\lambda_2\} \\ v_3 &= \{p :: r?\lambda_1, r :: s?\lambda_1 \cdot p!\lambda_1\} & v_4 &= \{p :: r?\lambda_1 \cdot s?\lambda_2, s :: r!\lambda_2 \cdot p!\lambda_2\} \\ v_5 &= \{p :: r?\lambda_1 \cdot s?\lambda_2 \cdot q!\lambda, q :: p?\lambda\} \end{aligned}$$

368 Here  $n(\{v_1, \dots, v_5\}) = \{v_1, v_2, v_3\}$ . Indeed, a causal set for  $v_4$  would need to contain both  
 369  $v_2$  and  $v_3$ , but this is not possible, since  $v_2 \# v_3$  by Clause (2a) of Definition 5.7. In fact  
 370  $s?\lambda_2 \# s?\lambda_1 \cdot p!\lambda_1$ . Then,  $v_5$  will also be pruned by the narrowing, since any causal set for  
 371  $v_5$  should contain  $v_4$ .

<sup>5</sup>In fact, every event of a PES occurs in a configuration.

372 We can now finally define the event structure associated with a network. The  
 373 intuition is that the events appearing in some configuration of the event structure  
 374 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*

$$\mathcal{S}^N(N) = (\mathcal{NE}(N), <_N, \#_N)$$

375 where:

- 376 1.  $\mathcal{NE}(N) = n(\mathcal{CE}(N))$  with  
 377  $\mathcal{CE}(N) = \{\{p :: \eta, q :: \eta'\} \mid p \ll P \ll N, q \ll Q \ll N, \eta \in \mathcal{PE}(P), \eta' \in \mathcal{PE}(Q), p :: \eta \widehat{\bowtie} q :: \eta'\}$
- 378 2.  $<_N$  is the restriction of  $<$  to the set  $\mathcal{NE}(N)$ ;
- 379 3.  $\#_N$  is the restriction of  $\#$  to the set  $\mathcal{NE}(N)$ .

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

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

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

**Example 5.15.** Let  $P$  be as in Example 4.4,  $Q = p? \lambda; Q + p? \lambda'$  and  $N = p \ll P \parallel q \ll Q \parallel$ .  
 Then

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

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

$$v_n = \{p :: \underbrace{q! \lambda \cdot \dots \cdot q! \lambda \cdot q! \lambda'}_n \cdot r! \lambda, r :: p? \lambda\}$$

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

397 **Theorem 5.16.** *Let  $N$  be a network. Then  $S^N(N)$  is a flow event structure with an*  
 398 *irreflexive conflict relation.*

399 **Proof** The relation  $<_N$  is irreflexive since  $\eta < \eta'$  implies  $v \neq v'$ , where  $\eta, \eta', v, v'$  are  
 400 as in Definition 5.7(1). As for the conflict relation, note first that a conflict between  
 401 an n-event and itself could not be derived by Clause (2b) of Definition 5.7, since  
 402 the two located events of an n-event are dual by construction. Lastly, symmetry  
 403 and irreflexivity of the conflict relation follow from the corresponding properties of  
 404 conflict between p-events.

405 The fact that the conflict relation is irreflexive in our network FESs means that  
 406 we do not exploit the possibility of self-conflicts offered by general FESs. This is due  
 407 to the way we defined the set of events of our network FESs, using the narrowing  
 408 function as discussed previously. We could have chosen an alternative definition,  
 409 introducing additional self-conflicting events of a more liberal form<sup>6</sup> which would  
 410 have disappeared when building configurations (together with their successors  
 411 having no other possible causes), as it was done for CCS in [10]. However, this  
 412 would have resulted in much larger sets of events for network FESs, leading to  
 413 more cumbersome examples and proofs. Our design choice here was to reduce the  
 414 set of events of network FESs by introducing already some semantic constraints on  
 415 their events (like duality and the existence of causal sets). It should be stressed,  
 416 however, that the narrowing function does not exclude all non executable events,  
 417 as shown by the FES in Example 5.20, which has three events, each of which has a  
 418 causal set but none of which is executable.

419 Although they have an irreflexive conflict relation like PESs, our network FESs  
 420 exhibit two important features which are not shared by PESs, namely non-hereditary  
 421 conflict (as shown by the FES given in Figure 5, where the two conflicting events  
 422  $v'_1$  and  $v'_2$  have a common successor  $v$ ) and causality cycles (as shown by the FES in  
 423 Example 5.20, where there is a circular dependency among the three events  $v_1, v_2$   
 424 and  $v_3$ ).

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

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

$$p \llbracket q! \lambda_1 \rrbracket \parallel q \llbracket p? \lambda_1; r! \lambda_2 \rrbracket \parallel r \llbracket q? \lambda_2; s! \lambda_3 \rrbracket \parallel s \llbracket r? \lambda_3 \rrbracket$$

---

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

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

$$v_1 = \{p :: q! \lambda_1, q :: p? \lambda_1\} \quad v_2 = \{q :: p? \lambda_1 \cdot r! \lambda_2, r :: q? \lambda_2\} \quad v_3 = \{r :: q? \lambda_2 \cdot s! \lambda_3, s :: r? \lambda_3\}$$

428 The flow relation obtained by Definition 5.13 is:  $v_1 < v_2$  and  $v_2 < v_3$ . These two flows are  
 429 inherited from the causality relations within the process ESs associated with participants  $q$   
 430 and  $r$ , respectively. The non-empty configurations are  $\{v_1\}$ ,  $\{v_1, v_2\}$  and  $\{v_1, v_2, v_3\}$ . Note  
 431 that  $\mathcal{S}^N(\mathbf{N})$  has only one proving sequence per configuration (which is the one given by the  
 432 numbering of events).

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

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

440 **Proposition 5.19.** *Let  $\mathbf{N} = p_1 \llbracket P_1 \rrbracket \parallel p_2 \llbracket P_2 \rrbracket$  and  $\mathcal{S}^N(\mathbf{N}) = (\mathcal{NE}(\mathbf{N}), <_N, \#)$ . Then*  
 441  *$n(\mathcal{CE}(\mathbf{N})) = \mathcal{CE}(\mathbf{N})$  and the structure  $\mathcal{S}_*^N(\mathbf{N}) =_{\text{def}} (\mathcal{NE}(\mathbf{N}), <_N^*, \#)$  is a prime event structure.*

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

$$\mathcal{CE}(\mathbf{N}) = \{ \{p_1 :: \eta_1, p_2 :: \eta_2\} \mid \eta_1 \in \mathcal{PE}(P_1), \eta_2 \in \mathcal{PE}(P_2), p_1 :: \eta_1 \widehat{\bowtie} p_2 :: \eta_2 \}$$

442 Let  $\{p_1 :: \eta_1, p_2 :: \eta_2\} \in \mathcal{CE}(\mathbf{N})$ . Since  $p_1 :: \eta_1 \widehat{\bowtie} p_2 :: \eta_2$  and all the actions in  $\eta_1$   
 443 involve  $p_2$  and all the actions in  $\eta_2$  involve  $p_1$ , we know that  $\eta_1$  and  $\eta_2$  have the  
 444 same length  $n \geq 1$  and for each  $i, 1 \leq i \leq n$ , the prefixes of length  $i$  of  $\eta_1$  and  $\eta_2$ ,  
 445 written  $\eta_1^i$  and  $\eta_2^i$ , must themselves be dual. Then  $\{p_1 :: \eta_1^i, p_2 :: \eta_2^i\} \in \mathcal{CE}(\mathbf{N})$  for each  
 446  $i, 1 \leq i \leq n$ , hence  $\{p_1 :: \eta_1, p_2 :: \eta_2\}$  has a causal set in  $\mathcal{CE}(\mathbf{N})$ .

447 We prove now that the reflexive and transitive closure  $<_N^*$  of  $<_N$  is a partial order.  
 448 Since by definition  $<_N^*$  is a preorder, we only need to show that it is antisymmetric.  
 449 Define the length of an n-event  $v = \{p_1 :: \eta_1, p_2 :: \eta_2\}$  to be  $\text{length}(v) =_{\text{def}} |\eta_1| + |\eta_2|$   
 450 (where  $|\eta|$  is the length of  $\eta$ ). Let now  $v, v' \in \mathcal{NE}(\mathbf{N})$ , with  $v = \{p_1 :: \eta_1, p_2 :: \eta_2\}$   
 451 and  $v' = \{p_1 :: \eta'_1, p_2 :: \eta'_2\}$ . By definition  $v <_N v'$  implies  $\eta_i < \eta'_i$  for some  
 452  $i = 1, 2$ , which in turn implies  $|\eta_i| < |\eta'_i|$ . As observed above,  $\eta_1$  and  $\eta_2$  must  
 453 have the same length, and so must  $\eta'_1$  and  $\eta'_2$ . This means that if  $v <_N v'$  then  
 454  $\text{length}(v) = |\eta_1| + |\eta_2| < |\eta'_1| + |\eta'_2| = \text{length}(v')$ . From this we can conclude that if  
 455  $v <_N^* v'$  and  $v' <_N^* v$ , then necessarily  $v = v'$ .

456 Finally we show that the relation  $\#$  satisfies the required properties. By Theo-  
 457 rem 5.16 we only need to prove that  $\#$  is hereditary. Let  $v$  and  $v'$  be as above. If  
 458  $v \# v'$ , then by Lemma 5.18  $\eta_1 \# \eta'_1$  and  $\eta_2 \# \eta'_2$ . Let now  $v'' = \{p_1 :: \eta''_1, p_2 :: \eta''_2\}$ . If  
 459  $v' <_N^* v''$ , this means that there exist  $v_1, \dots, v_n$  such that  $v' <_N v_1 \dots <_N v_n = v''$ . We  
 460 prove by induction on  $n$  that  $v \# v''$ . For  $n = 1$  we have  $v' <_N v''$ . Then by Clause  
 461 (1) of Definition 5.13 we have  $\eta'_j < \eta''_j$  for some  $j \in \{1, 2\}$ . Since  $\eta_i \# \eta'_i$  for all  $i \in \{1, 2\}$   
 462 and  $\#$  is hereditary on p-events, we deduce  $\eta_j \# \eta''_j$ , which implies  $v \# v''$ . Suppose  
 463 now  $n > 1$ . By induction  $v \# v_{n-1}$ . Since  $v_{n-1} <_N v_n = v''$  we then obtain  $v \# v''$  by the  
 464 same argument as in the base case.



465 If a network has more than two participants, then the duality requirement on  
 466 its n-events is not sufficient to ensure the absence of circular dependencies<sup>7</sup>. For  
 467 instance, in the following ternary network (which may be viewed as representing  
 468 the 3-philosopher deadlock) the relation  $<^*$  is not a partial order.

**Example 5.20.** *Let  $N$  be the network*

$$p \ll r? \lambda; q! \lambda' \parallel q \ll p? \lambda'; r! \lambda'' \parallel r \ll q? \lambda''; p! \lambda$$

*Then  $\mathcal{S}^N(N)$  has three n-events*

$$\begin{aligned} v_1 &= \{p :: r? \lambda, r :: q? \lambda'' \cdot p! \lambda\} & v_2 &= \{p :: r? \lambda \cdot q! \lambda', q :: p? \lambda'\} \\ v_3 &= \{q :: p? \lambda' \cdot r! \lambda'', r :: q? \lambda''\} \end{aligned}$$

469 By Definition 5.13(1) we have  $v_1 < v_2 < v_3$  and  $v_3 < v_1$ . The only configuration of  $\mathcal{S}^N(N)$   
 470 is the empty configuration, because the only set of n-events that satisfies downward-closure  
 471 up to conflicts is  $X = \{v_1, v_2, v_3\}$ , but this is not a configuration because  $<_X^*$  is not a partial  
 472 order (recall that  $<_X$  is the restriction of  $<$  to  $X$ ) and hence the condition (3) of Definition 3.4  
 473 is not satisfied.

## 474 5.2. Further Properties

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

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

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

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

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

<sup>8</sup>A prime configuration is a configuration with a unique maximal element, its *culminating* event.

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

499 **Proof** Suppose ad absurdum that  $X$  is a configuration such that  $v, v' \in X$ . If  
 500  $|\eta \dot{p} q| = |\eta' \dot{p} p|$  then  $v \# v'$  by Definition 5.7(2b) and we reach immediately a  
 501 contradiction. So, assume  $|\eta \dot{p} q| > |\eta' \dot{p} p| = n$ . This means that  $|\eta| > 1$  and thus  
 502 there exists a non-empty causal set  $E_v$  of  $v$  such that  $E_v \subseteq X$ . Let  $\eta_0 < \eta$  be such that  
 503  $|\eta_0 \dot{p} q| = |\eta' \dot{p} p| = n$ . By definition of causal set, there exists  $v_0 \in E_v$  such that  
 504  $p :: \eta_0 \in v_0$ . By Definition 5.7(2b) we have then  $v_0 \# v'$ , contradicting the fact that  $X$   
 505 is conflict-free.

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

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

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

519 **Proof** Suppose  $v = \{p :: \eta, q :: \eta'\}$ . We show that  $|E| \leq |\eta| + |\eta'| - 2$ , where  
 520  $|E|$  is the cardinality of  $E$ . By Condition (2) of Definition 5.9, for each  $\eta_0 < \eta$  and  
 521  $\eta'_0 < \eta'$  there must be  $v_0, v'_0 \in E$  such that  $p :: \eta_0 \in v_0$  and  $q :: \eta'_0 \in v'_0$ . Note  
 522 that  $v_0$  and  $v'_0$  could possibly coincide. Moreover, there cannot be  $v' \in E$  such that  
 523  $p :: \eta_0 \in v' \neq v_0$  or  $q :: \eta'_0 \in v' \neq v'_0$ , since this would contradict the minimality of  $E$   
 524 (and also its conflict-freeness, since by Lemma 5.21 we would have either  $v' \# v_0$  or  
 525  $v' \# v'_0$ ). Hence the number of events in  $E$  is at most  $(|\eta| - 1) + (|\eta'| - 1)$ .

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

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

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

---

<sup>9</sup>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).**

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

The projection function  $proj_p(\cdot)$  is extended to sets of n-events in the obvious way:

$$proj_p(X) = \{\eta \mid \exists v \in X. proj_p(v) = \eta\}$$

**Example 5.26.** Let  $\{v_1, v_2, v_3\}$  be the configuration defined in Example 5.17. We get

$$proj_q(\{v_1, v_2, v_3\}) = \{p?\lambda_1, p?\lambda_1 \cdot r!\lambda_2\}$$

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

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  $\mathcal{S}^N(N) = (\mathcal{NE}(N), <_N, \#_N)$  and  $\mathcal{S}^P(P) = (\mathcal{PE}(P), \leq_P, \#_P)$  and  $p \ll P \in N$ . Then the partial function  $proj_p : \mathcal{NE}(N) \rightarrow \mathcal{PE}(P)$  is downward surjective.

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

$$\eta <_P proj_p(v) \implies \exists v' \in \mathcal{NE}(N). \eta = proj_p(v')$$

Note that  $\eta <_P proj_p(v)$  implies  $proj_p(v) = \eta \cdot \eta'$  for some  $\eta'$ . Recall that  $\mathcal{NE}(N) = n(\mathcal{CE}(N))$ , where  $n(\cdot)$  is the narrowing function (Definition 5.11).

By definition of narrowing,  $p :: \eta \cdot \eta' \in \mathcal{NE}(N)$  implies that there is  $E \subseteq \mathcal{NE}(N)$  such that  $E$  is a causal set of  $v$  in  $\mathcal{NE}(N)$ . Therefore  $p :: \eta \cdot \eta' \in v$  implies  $p :: \eta \in E$  and so  $p :: \eta \in \mathcal{NE}(N)$ , which is what we wanted to show.

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

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

Notice that the reverse of Theorem 5.29 is not true, namely  $p \ll P \in N$  does not imply that each configuration of  $C(\mathcal{S}^P(P))$  can be obtained by projecting some

557 configuration of  $C(S^N(N))$  on  $p$ . Consider for instance the network  $N = p \llbracket q? \lambda \rrbracket$ .  
 558 Then  $\{q? \lambda\} \in C(S^P(P))$ , while  $C(S^N(N)) = \emptyset$ .

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

## 569 6. Global Types

570 This section is devoted to our type system for multiparty sessions. Global types  
 571 describe the communication protocols involving all session participants. Usually,  
 572 global types are projected into local types and typing rules are used to derive local  
 573 types for processes [39, 19, 40]. The simplicity of our calculus allows us to project  
 574 directly global types into processes and to have exactly one typing rule, see Figures 2  
 575 and 3. This section is split in two subsections.

576 The first subsection presents the projection of global types onto processes, together  
 577 with the proof of its soundness. Moreover it introduces a *boundedness* condition on  
 578 global types, which is crucial for our type system to ensure progress.

579 The second subsection presents the type system, as well as an LTS for global types.  
 580 Lastly, the properties of Subject Reduction, Session Fidelity and Progress are shown.  
 581 The omitted proofs can be found in Appendix B.

### 582 6.1. Well-formed Global Types

583 Global types are built from choices among communications.

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

$$G ::=^{coind} p \rightarrow q : \boxplus_{i \in I} \lambda_i ; G_i \mid \text{End}$$

584 *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.*

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

588 The type  $p \rightarrow q : \boxplus_{i \in I} \lambda_i ; G_i$  formalises a protocol which starts with the commu-  
 589 nication of a message  $\lambda_k$  from  $p$  to  $q$ , for some  $k \in I$ , and then, depending on which  
 590  $\lambda_k$  was chosen by  $p$ , continues as  $G_k$ .

591 When  $I$  is a singleton, a choice  $p \rightarrow q : \boxplus_{i \in I} \lambda_i ; G_i$  will be rendered simply as  
 592  $p \xrightarrow{\lambda} q ; G$ . When  $I$  contains only two elements, as for processes we will use the  
 593 binary choice notation  $p \rightarrow q : (\lambda_1 ; G_1 \boxplus \lambda_2 ; G_2)$ . Trailing End types will be omitted.

$$\begin{aligned}
& G \upharpoonright r = 0 \text{ if } r \notin \text{part}(G) \\
& (p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i) \upharpoonright r = \begin{cases} \sum_{i \in I} p? \lambda_i; G_i \upharpoonright r & \text{if } r = q, \\ \oplus_{i \in I} q! \lambda_i; G_i \upharpoonright r & \text{if } r = p, \\ G_1 \upharpoonright r & \text{if } r \notin \{p, q\} \text{ and } r \in \text{part}(G_1) \text{ and} \\ & G_i \upharpoonright r = G_1 \upharpoonright r \text{ for all } i \in I \end{cases}
\end{aligned}$$

**Figure 2:** Projection of global types onto participants.

Global types may be viewed as trees whose internal nodes are decorated by  $pq$ , leaves by  $\text{End}$ , and edges by messages  $\lambda$ . 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  $\text{Tr}^+(G)$  the set of traces of  $G$ . By definition,  $\text{Tr}^+(\text{End}) = \emptyset$  and each trace in  $\text{Tr}^+(G)$  is non-empty.

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

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

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  $\mu$ -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  $G = p \rightarrow q : (\lambda_1; q \xrightarrow{\lambda_3} r \boxplus \lambda_2; G)$  is projectable and

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

On the other hand, the corresponding global type based on the  $\mu$ -notation

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

622 is not projectable because  $G' \upharpoonright r$  is not defined.

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

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

**Definition 6.3 (Depth and boundedness).**

Let the two functions  $\text{depth}(\sigma, p)$  and  $\text{depth}(G, p)$  be defined by:

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

639 Then

$$640 \quad \text{depth}(G, p) = \sup\{\text{depth}(\sigma, p) \mid \sigma \in \text{Tr}^+(G)\}$$

641

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

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

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

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

whereas

$$\text{depth}(G, p) = 1 \quad \text{depth}(G, q) = 1 \quad \text{depth}(G, r) = \infty$$

647 since

$$648 \quad \text{Tr}^+(G) = \underbrace{\{pq\lambda_2 \cdots pq\lambda_2 \cdot pq\lambda_1 \cdot qr\lambda_3 \mid n \geq 0\}}_n \cup \{pq\lambda_2 \cdots pq\lambda_2 \cdots\}$$

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

650 The depths of the participants in  $G$  which are not participants of its root com-  
 651 munication decrease in the immediate subtrees of  $G$ . The proof is trivial since, if  
 652  $G = p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i$ , then  $\sigma \in \text{Tr}^+(G)$  implies  $\sigma = pq\lambda_i \cdot \sigma'$  and  $\sigma' \in \text{Tr}^+(G_i)$  for  
 653 some  $i \in I$ .

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

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

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

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

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

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

## 665 6.2. Type System

$$\begin{array}{c}
 \mathbf{0} \leq \mathbf{0} [\leq -\mathbf{0}] \quad \frac{P_i \leq Q_i \quad i \in I}{\sum_{i \in I \cup \mathbf{p}} \mathbf{p} ? \lambda_i; P_i \leq \sum_{i \in I \cup \mathbf{p}} \mathbf{p} ? \lambda_i; Q_i} [\leq -\text{IN}] \quad \frac{P_i \leq Q_i \quad i \in I}{\bigoplus_{i \in I} \mathbf{p} ! \lambda_i; P_i \leq \bigoplus_{i \in I} \mathbf{p} ! \lambda_i; Q_i} [\leq -\text{OUT}] \\
 \\
 \frac{P_i \leq G \upharpoonright \mathbf{p}_i \quad i \in I \quad \text{part}(G) \subseteq \{\mathbf{p}_i \mid i \in I\}}{\vdash \prod_{i \in I} \mathbf{p}_i \llbracket P_i \rrbracket : G} [\text{NET}]
 \end{array}$$

**Figure 3:** Preorder on processes and network typing rule.

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

676 A network is well typed if all its participants have associated processes that  
 677 behave as specified by the projections of a global type. In Rule [NET], the condition  
 678  $\text{part}(G) \subseteq \{\mathbf{p}_i \mid i \in I\}$  ensures that all participants of the global type appear in the

$$\begin{array}{c}
p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i \xrightarrow{pq\lambda_j} G_j \quad j \in I \quad [\text{Ecomm}] \\
\\
\frac{G_i \xrightarrow{\alpha} G'_i \quad \text{for all } i \in I \quad \text{part}(\alpha) \cap \{p, q\} = \emptyset}{p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i \xrightarrow{\alpha} p \rightarrow q : \boxplus_{i \in I} \lambda_i; G'_i} \quad [\text{Icomm}]
\end{array}$$

**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  $p \llbracket 0 \rrbracket$  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*

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

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  $p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i$ , 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  $G \xrightarrow{pq\lambda} G'$ , 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  $\vdash N : G$  and  $N \xrightarrow{\alpha} N'$ , then  $G \xrightarrow{\alpha} G'$  and  $\vdash N' : G'$ .*

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

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  $\vdash N : G$  and  $p \llbracket P \rrbracket \in N$ , then  $N \xrightarrow{\sigma \cdot \alpha} N'$  and  $p \in \text{part}(\alpha)$ .*



704 **Proof** We prove by induction on  $d = \text{depth}(G, p)$  that: if  $\vdash N : G$  and  $p \llbracket P \rrbracket \in N$ ,  
 705 then  $G \xrightarrow{\sigma \cdot \alpha} G'$  with  $p \in \text{part}(\alpha)$ . This will imply  $N \xrightarrow{\sigma \cdot \alpha} N'$  by Session Fidelity  
 706 (Theorem 6.11).

707 *Case  $d = 1$ .* In this case  $G = q \rightarrow r : \boxplus_{i \in I} \lambda_i; G_i$  and  $p \in \{q, r\}$  and  $G \xrightarrow{qr\lambda_h} G_h$  for some  
 708  $h \in I$  by Rule [EComm].

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

714 The proof of the progress theorem shows that the execution strategy which uses  
 715 only Rule [EComm] is fair, since there are no infinite transition sequences where  
 716 some participant is stuck. This is due to the boundedness condition on global  
 717 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:*

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

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

## 721 7. Event Structure Semantics of Global Types

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

724 The unique omitted proof can be found in Appendix C.

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

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

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

732 We start by defining an equivalence relation on  $\text{Traces}$  which allows swapping  
 733 of communications with disjoint participants.

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

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

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

The events associated with a global type, called *g-events* and denoted by  $\gamma, \gamma'$ , 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  $\sigma = \sigma[1 \dots n]$  is said to be pointed if

$$\text{for all } i, 1 \leq i < n, \text{ part}(\sigma[i]) \cap \text{part}(\sigma[(i+1) \dots n]) \neq \emptyset$$

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  $\alpha_1 = \text{pq}\lambda_1$ ,  $\alpha_2 = \text{rs}\lambda_2$  and  $\alpha_3 = \text{rp}\lambda_3$ . 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  $\text{last}(\sigma)$  to denote the last communication of  $\sigma$ .

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

**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  $\alpha$ , notation  $\text{cm}(\gamma) = \alpha$ . We denote by  $\mathcal{G}$  the set of g-events.

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

We now introduce an operator of prefixing of a g-event  $\gamma$  by a communication  $\alpha$ , which acts as follows: if  $\alpha$  is a cause of some communication in the trace of  $\gamma$ , then  $\alpha$  is added at the beginning of the trace, otherwise  $\gamma$  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  $\gamma$  if we were to execute the communication  $\alpha$  before  $\gamma$ . This operator is the counterpart of the “residual of a g-event after a communication”, which yields the g-event obtained from  $\gamma$  after executing the communication  $\alpha$  from  $\gamma$ , 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:

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

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

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

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

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

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

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

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

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

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  $\alpha$  onto  $r$ ,  $\alpha@r$ , is defined by:

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

2. The projection of a trace  $\sigma$  onto  $r$ ,  $\sigma@r$ , is defined by:

$$\epsilon@r = \epsilon \quad (\alpha \cdot \sigma)@r = \alpha@r \cdot \sigma@r$$

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

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

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

If  $\gamma = [\sigma \cdot \alpha \cdot \sigma' \cdot \alpha']_{\sim}$ , then the communication  $\alpha$  must be done before the communication  $\alpha'$ . This is expressed by the causality  $[\sigma \cdot \alpha]_{\sim} \leq \gamma$ . An example is  $[pq\lambda]_{\sim} \leq [rs\lambda' \cdot pq\lambda \cdot sq\lambda']_{\sim}$ .

As regards conflict, note that if  $\sigma \sim \sigma'$  then  $\sigma@p = \sigma'@p$  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  $\sigma@p \# \sigma'@p$  states that participant  $p$  does the same actions in both traces up to some point, after which it performs two different actions in  $\sigma$  and  $\sigma'$ . For example  $[pq\lambda \cdot rp\lambda_1 \cdot qp\lambda']_{\sim} \# [pq\lambda \cdot rp\lambda_2]_{\sim}$ , since  $(pq\lambda \cdot rp\lambda_1 \cdot qp\lambda')@p = q!\lambda \cdot r?\lambda_1 \cdot q?\lambda' \# q!\lambda \cdot r?\lambda_2 = (pq\lambda \cdot rp\lambda_2)@p$ .

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

$$\mathcal{S}^G(G) = (\mathcal{E}(G), \leq_G, \#_G)$$

788 *where:*

- 789 1.  $\mathcal{E}(G) = \{\text{ev}(\sigma) \mid \sigma \in \text{Tr}^+(G)\}$
- 790 2.  $\leq_G$  is the restriction of  $\leq$  to the set  $\mathcal{E}(G)$ ;
- 791 3.  $\#_G$  is the restriction of  $\#$  to the set  $\mathcal{E}(G)$ .

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

**Example 7.12.** *Let  $G_1 = p \xrightarrow{\lambda_1} q; r \xrightarrow{\lambda_2} s; r \xrightarrow{\lambda_3} p$  and  $G_2 = r \xrightarrow{\lambda_2} s; p \xrightarrow{\lambda_1} q; r \xrightarrow{\lambda_3} p$ . Then  $\mathcal{E}(G_1) = \mathcal{E}(G_2) = \{\gamma_1, \gamma_2, \gamma_3\}$  where*

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

*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*

$$\gamma_1 \quad \gamma_2 \quad \gamma_1; \gamma_2 \quad \gamma_2; \gamma_1 \quad \gamma_1; \gamma_2; \gamma_3 \quad \gamma_2; \gamma_1; \gamma_3$$

*If  $G'$  is as in Example 6.8, then  $\mathcal{E}(G') = \{\gamma_1, \gamma_2, \gamma_3\}$  where*

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

793 *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*  
 794 *unique proving sequence corresponding to each configuration.*

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

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

806 Observe that, while our interpretation of networks as FESs exactly reflects the  
 807 concurrency expressed by the syntax of networks, our interpretation of global types  
 808 as PESs exhibits more concurrency than that given by the syntax of global types.

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

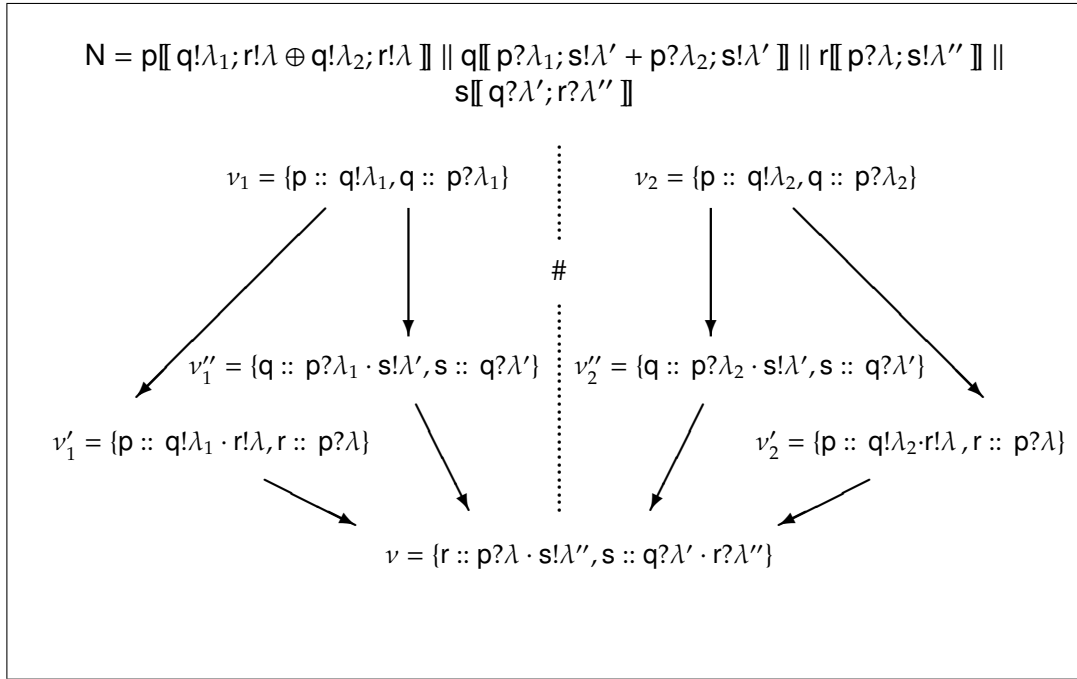


Figure 5: FES of the network N.

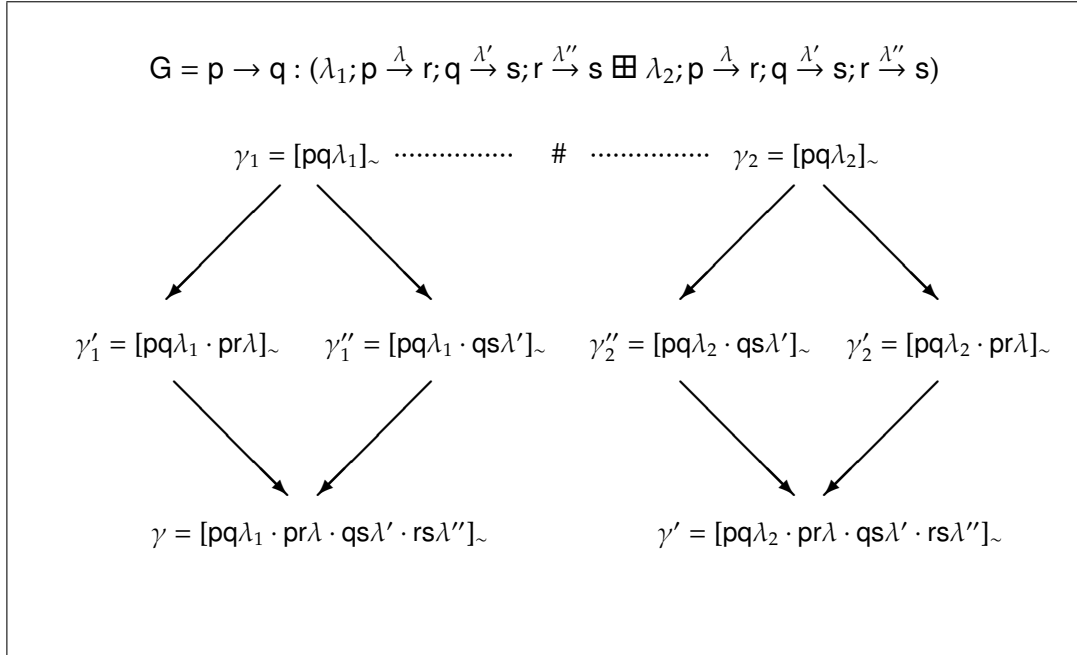


Figure 6: PES of the type G.

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

813 Consider the network FES pictured in Figure 5, where the arrows represent the  
814 flow relation and all the n-events on the left of the dotted line are in conflict with all  
815 the n-events on the right of the line. In particular, notice that the conflicts between

816 n-events with a common location are deduced by Clause (2a) of Definition 5.7, while  
 817 the conflicts between n-events with disjoint sets of locations, such as  $v'_1$  and  $v''_2$ , are  
 818 deduced by Clause (2b) of Definition 5.7. Observe also that the n-event  $v$  has two  
 819 different causal sets in  $\mathcal{NE}(\mathcal{N})$ , namely  $\{v'_1, v''_1\}$  and  $\{v'_2, v''_2\}$ . The reader familiar with  
 820 ESs will have noticed that there are also two prime configurations whose maximal  
 821 element is  $v$ , namely  $\{v_1, v'_1, v''_1, v\}$  and  $\{v_2, v'_2, v''_2, v\}$ . It is easy to see that the network  
 822  $\mathcal{N}$  can be typed with the global type  $G$  shown in Figure 6.

823 Consider now the PES of the type  $G$  pictured in Figure 6, where the arrows  
 824 represent the covering relation of the partial order of causality and inherited conflicts  
 825 are not shown. Note that while the FES of  $\mathcal{N}$  has a unique maximal n-event  $v$ , the  
 826 PES of its type  $G$  has two maximal g-events  $\gamma$  and  $\gamma'$ . This is because an n-event  
 827 only records the computations that occurred at its locations, while a g-event records  
 828 the global computation and keeps a record of each choice, including those involving  
 829 locations that are disjoint from those of its last communication. Indeed, g-events  
 830 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  $\text{gn}(\cdot)$  that maps g-events to n-events:

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

831 On the other hand, the inverse construction is not as direct. First of all, an  
 832 n-event in the network FES may give rise to several g-events in the type PES, as  
 833 shown by the n-event  $v$  in Figure 5, which gives rise to the pair of g-events  $\gamma$  and  $\gamma'$   
 834 in Figure 6. Moreover, the local information contained in an n-event is not sufficient  
 835 to reconstruct the corresponding g-events: for each n-event, we need to consider all  
 836 the prime configurations that culminate with that event, and then map each of these  
 837 configurations to a g-event. Hence, we need a function  $\text{ng}(\cdot)$  that maps n-events to  
 838 sets of prime configurations of the FES, and then maps each such configuration to  
 839 a g-event. We will not explicitly define this function here, since we miss another  
 840 important ingredient to compare the FES of a network and the PES of its type,  
 841 namely a structural characterisation of the FESs that represent typable networks.  
 842 Indeed, if we started from the FES of a non typable network, this construction  
 843 would not be correct. Consider for instance the network  $\mathcal{N}'$  obtained from  $\mathcal{N}$  by  
 844 omitting the output  $r!\lambda$  from the second branch of the process of  $p$ . Then the FES  
 845 of  $\mathcal{N}'$  would not contain the n-event  $v'_2$  and the event  $v$  would have the unique  
 846 causal set  $\{v'_1, v''_1\}$ , and the unique prime configuration culminating with  $v$  would  
 847 be  $\{v_1, v'_1, v''_1, v\}$ . Then our construction would give a PES that differs from that of  
 848 type  $G$  only for the absence of the g-events  $\gamma'_2$  and  $\gamma'$ . However, the network  $\mathcal{N}'$   
 849 is not typable and thus we would expect the construction to fail. Note that in the  
 850 FES of  $\mathcal{N}'$ , the n-event  $v''_2$  is a cause of  $v$  but does not belong to any causal set of  
 851  $v$ . Thus a possible well-formedness property to require for FESs to be images of a  
 852 typable network would be that each cause of each n-event belong to some causal  
 853 set of that event. However, this would still not be enough to exclude the FES of  
 854 the non typable network  $\mathcal{N}''$  obtained from  $\mathcal{N}'$  by omitting the output  $s!\lambda'$  from the  
 855 second branch of the process of  $q$ .

856 To conclude, in the absence of a semantic counterpart for the well-formedness  
 857 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

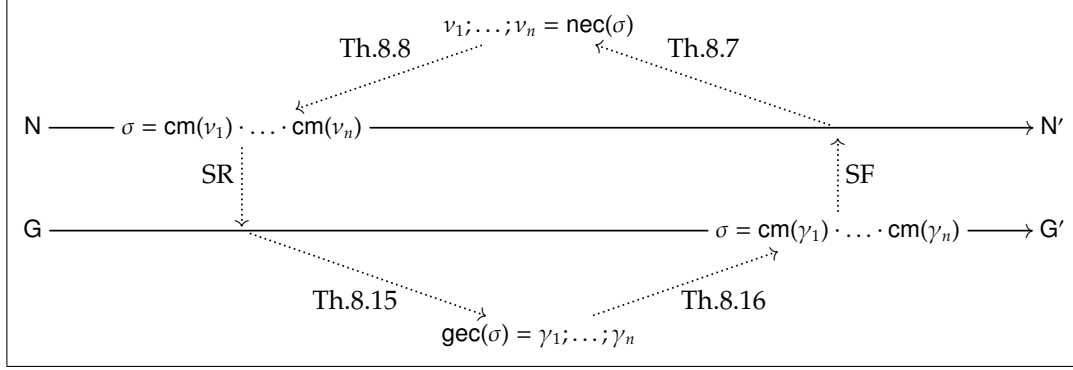


Figure 7: Isomorphism proof in a nutshell.

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  $v_1; \dots; v_n$  and  $\gamma_1; \dots; \gamma_n$  are proving sequences of  $\mathcal{S}^N(N)$  and  $\mathcal{S}^G(G)$ , respectively. Finally,  $\text{nec}(\sigma)$  and  $\text{gec}(\sigma)$  denote the proving sequences of n-events and g-events which correspond to the trace  $\sigma$  (as given by Definition 8.3 and Definition 8.13). Theorem 8.8 says that, if  $v_1; \dots; v_n$  is a proving sequence of  $\mathcal{S}^N(N)$ , then  $N \xrightarrow{\sigma} N'$ , where  $\sigma = \text{cm}(v_1) \cdot \dots \cdot \text{cm}(v_n)$ . By Subject Reduction (Theorem 6.10)  $G \xrightarrow{\sigma} G'$ . This implies that  $\text{gec}(\sigma)$  is a proving sequence of  $\mathcal{S}^G(G)$  by Theorem 8.15. Dually, Theorem 8.16 says that, if  $\gamma_1; \dots; \gamma_n$  is a proving sequence of  $\mathcal{S}^G(G)$ , then  $G \xrightarrow{\sigma} G'$ , where  $\sigma = \text{cm}(\gamma_1) \cdot \dots \cdot \text{cm}(\gamma_n)$ . By Session Fidelity (Theorem 6.11)  $N \xrightarrow{\sigma} N'$ . Lastly,  $\text{nec}(\sigma)$  is a proving sequence of  $\mathcal{S}^N(N)$  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.

### 887 8.1. Relating Transition Sequences of Networks and Proving Sequences of their ESs

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

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

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

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

$$\alpha \diamond (p :: \eta) = p :: (\alpha @ p) \cdot \eta$$

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):

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

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

$$\begin{aligned} \alpha \diamond (\{p :: \eta, q :: \eta'\}) &= \{\alpha \diamond (p :: \eta), \alpha \diamond (q :: \eta')\} \\ \alpha \blacklozenge (\{p :: \eta, q :: \eta'\}) &= \{\alpha \blacklozenge (p :: \eta), \alpha \blacklozenge (q :: \eta')\} \\ \epsilon \diamond v &= v & (\alpha \cdot \sigma) \diamond v &= \alpha \diamond (\sigma \diamond v) \\ \epsilon \blacklozenge v &= v & (\alpha \cdot \sigma) \blacklozenge v &= \sigma \blacklozenge (\alpha \blacklozenge v) \end{aligned}$$

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

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

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

- 907 1. If  $\alpha \blacklozenge v$  is defined, then  $\alpha \diamond (\alpha \blacklozenge v) = v$ ;  
 908 2.  $\alpha \blacklozenge (\alpha \diamond v) = v$ ;  
 909 3. If  $v < v'$ , then  $\alpha \diamond v < \alpha \diamond v'$ ;  
 910 4. If  $v < v'$  and both  $\alpha \blacklozenge v$  and  $\alpha \blacklozenge v'$  are defined, then  $\alpha \blacklozenge v < \alpha \blacklozenge v'$ ;  
 911 5. If  $v \# v'$ , then  $\alpha \diamond v \# \alpha \diamond v'$ ;  
 912 6. If  $v \# v'$  and both  $\alpha \blacklozenge v$  and  $\alpha \blacklozenge v'$  are defined, then  $\alpha \blacklozenge v \# \alpha \blacklozenge v'$ ;



913 7. If  $\alpha \diamond v \# \alpha \diamond v'$ , then  $v \# v'$ .

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

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

$$\text{nec}(\sigma) = v_1; \dots; v_n$$

917 where  $v_i = \sigma[1 \dots i-1] \diamond \{p_i :: q_i! \lambda_i, q_i :: p_i? \lambda_i\}$  for  $1 \leq i \leq n$ .

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

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

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

926 **Lemma 8.4 (Properties of  $\text{nec}(\cdot)$ ).**

928 1. Let  $\text{nec}(\sigma) = v_1; \dots; v_n$ . Then

929 (a)  $\text{cm}(v_i) = \sigma[i]$  for all  $i$ ,  $1 \leq i \leq n$ ;

930 (b) If  $1 \leq h, k \leq n$ , then  $\neg(v_h \# v_k)$ .

931 2.  $\neg(\text{nec}(\alpha) \# \alpha \diamond v)$  for all  $v$ .

932 3. Let  $\sigma = \alpha \cdot \sigma'$  and  $\sigma' \neq \epsilon$ . If  $\text{nec}(\sigma) = v_1; \dots; v_n$  and  $\text{nec}(\sigma') = v'_2; \dots; v'_n$ , then  
 933  $\alpha \diamond v'_i = v_i$  and  $\alpha \diamond v_i = v'_i$  for all  $i$ ,  $2 \leq i \leq n$ .

934 Notice that if  $\alpha \diamond v$  is undefined and  $v$  is an n-event of a network with commu-  
 935 nication  $\alpha$ , then either  $v = \text{nec}(\alpha)$  or  $v \# \text{nec}(\alpha)$ .

936 **Lemma 8.5.** If  $N \xrightarrow{\alpha} N'$  and  $v \in \mathcal{NE}(N)$ , then  $v = \text{nec}(\alpha)$  or  $v \# \text{nec}(\alpha)$  or  $\alpha \diamond v$  is defined.

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

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

943 1.  $\{\text{nec}(\alpha)\} \cup \{\alpha \diamond v \mid v \in \mathcal{NE}(N')\} \subseteq \mathcal{NE}(N)$ ;

944 2.  $\{\alpha \diamond v \mid v \in \mathcal{NE}(\mathbf{N}) \text{ and } \alpha \diamond v \text{ defined}\} \subseteq \mathcal{NE}(\mathbf{N}')$ .

945 We may now prove the correspondence between the traces labelling the transi-  
946 tion sequences of a network and the proving sequences of its FES.

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

948 **Proof** The proof is by induction on  $\sigma$ .

949 *Base case.* Let  $\sigma = \alpha$ . From  $\mathbf{N} \xrightarrow{\alpha} \mathbf{N}'$  and Lemma 8.6(1)  $\text{nec}(\alpha) \in \mathcal{NE}(\mathbf{N})$ . Since  $\text{nec}(\alpha)$   
950 has no causes, by Definition 3.6 we conclude that  $\text{nec}(\alpha)$  is a proving sequence in  
951  $\mathcal{S}^{\mathcal{N}}(\mathbf{N})$ .

952 *Inductive case.* Let  $\sigma = \alpha \cdot \sigma'$ . From  $\mathbf{N} \xrightarrow{\sigma} \mathbf{N}'$  we get  $\mathbf{N} \xrightarrow{\alpha} \mathbf{N}'' \xrightarrow{\sigma'} \mathbf{N}'$  for some  $\mathbf{N}''$ .  
953 Let  $\text{nec}(\sigma) = v_1; \dots; v_n$  and  $\text{nec}(\sigma') = v'_2; \dots; v'_n$ . By induction  $\text{nec}(\sigma')$  is a proving  
954 sequence in  $\mathcal{S}^{\mathcal{N}}(\mathbf{N}'')$ .

955 We show that  $\text{nec}(\sigma)$  is a proving sequence in  $\mathcal{S}^{\mathcal{N}}(\mathbf{N})$ . By Lemma 8.4(1b)  $\text{nec}(\sigma')$   
956 is conflict free. By Lemma 8.4(3)  $v_i = \alpha \diamond v'_i$  for all  $i$ ,  $2 \leq i \leq n$ . This implies  
957  $v_i \in \mathcal{NE}(\mathbf{N})$  for all  $i$ ,  $2 \leq i \leq n$  by Lemma 8.6(1) and  $\neg(v_1 \# v_j)$  for all  $i, j$ ,  $2 \leq i, j \leq n$   
958 by Lemma 8.2(7). Finally, since  $v_1 = \text{nec}(\alpha)$ , by Lemma 8.4(2) we obtain  $\neg(v_1 \# v_i)$   
959 for all  $i$ ,  $2 \leq i \leq n$ . We conclude that  $\text{nec}(\sigma)$  is conflict-free and included in  $\mathcal{NE}(\mathbf{N})$ .  
960 Let  $v \in \mathcal{NE}(\mathbf{N})$  and  $v < v_k$  for some  $k$ ,  $1 \leq k \leq n$ . This implies  $k > 1$  since  $\text{nec}(\alpha)$  has  
961 no causes. Hence  $v_k = \alpha \diamond v'_k$ . By Lemma 8.5, we know that  $v = \text{nec}(\alpha)$  or  $v \# \text{nec}(\alpha)$   
962 or  $\alpha \diamond v$  is defined. We consider the three cases. Let  $\text{part}(\alpha) = \{p, q\}$ .

963 *Case  $v = \text{nec}(\alpha)$ .* In this case we conclude immediately since  $\text{nec}(\alpha) = v_1$  and  $1 < k$ .

964 *Case  $v \# \text{nec}(\alpha)$ .* Since  $\text{nec}(\alpha) = v_1$ , if  $v_1 < v_k$  we are done. If  $v_1 \not< v_k$ , then  
965  $\text{loc}(v_k) \cap \{p, q\} = \emptyset$  otherwise  $v_1 \# v_k$ . We get  $v_k = \alpha \diamond v'_k = v'_k$ . Since  $v < v_k$ , there exists  
966  $r :: \eta \in v$  and  $r :: \eta' \in v_k = v'_k$  such that  $\eta < \eta'$ , where  $r \notin \{p, q\}$  because  $r \in \text{loc}(v_k)$ .  
967 Since  $\text{nec}(\sigma')$  is a proving sequence in  $\mathcal{S}^{\mathcal{N}}(\mathbf{N}'')$ , by Lemma 5.24 there is  $v'_h \in \mathcal{NE}(\mathbf{N}'')$   
968 such that  $r :: \eta \in v'_h$ . Since  $\alpha \diamond r :: \eta = r :: \eta$  we get  $r :: \eta \in v_h$ . This implies  $v_h < v_k$ ,  
969 where  $v_h \# v$  by Lemma 5.21.

970 *Case  $\alpha \diamond v$  defined.* We get  $\alpha \diamond v < v'_k$  by Lemma 8.2(4). Since  $\text{nec}(\sigma')$  is a proving  
971 sequence in  $\mathcal{S}^{\mathcal{N}}(\mathbf{N}'')$ , there is  $h < k$  such that either  $\alpha \diamond v = v'_h$  or  $\alpha \diamond v \# v'_h < v'_k$ . In  
972 the first case  $v = \alpha \diamond (\alpha \diamond v) = \alpha \diamond v'_h = v_h$  by Lemma 8.2(1). In the second case:

- 973 • from  $\alpha \diamond v \# v'_h$  we get  $(\alpha \diamond (\alpha \diamond v)) \# (\alpha \diamond v'_h)$  by Lemma 8.2(5), which implies  
974  $v \# v_h$  by Lemma 8.2(1), and
- 975 • from  $v'_h < v'_k$  we get  $(\alpha \diamond v'_h) < (\alpha \diamond v'_k)$  by Lemma 8.2(3), namely  $v_h < v_k$ .

976 **Theorem 8.8.** *If  $v_1; \dots; v_n$  is a proving sequence in  $\mathcal{S}^{\mathcal{N}}(\mathbf{N})$ , then  $\mathbf{N} \xrightarrow{\sigma} \mathbf{N}'$ , where  $\sigma =$   
977  $\text{cm}(v_1) \dots \text{cm}(v_n)$ .*

**Proof** The proof is by induction on  $n$ .

Case  $n = 1$ . Let  $v_1 = \{p :: \zeta \cdot q! \lambda, q :: \zeta' \cdot p? \lambda\}$ . Then  $\text{cm}(v_1) = \text{pq}\lambda$ . 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  $v \in \mathcal{NE}(\mathbf{N})$  such that  $v < v_1$ , contradicting the fact that  $v_1$  is a proving  
sequence.

By Definition 5.13(1) we have  $N = p \llbracket P \rrbracket \parallel q \llbracket Q \rrbracket \parallel N_0$  with  $q! \lambda \in \mathcal{PE}(P)$  and  $p? \lambda \in \mathcal{PE}(Q)$ . Whence by Definition 4.3(1) we get  $P = \bigoplus_{i \in I} q! \lambda_i; P_i$  and  $Q = \sum_{j \in J} p? \lambda_j; Q_j$  where  $\lambda = \lambda_k$  for some  $k \in I \cap J$ . Therefore

$$N \xrightarrow{pq\lambda} p \llbracket P_k \rrbracket \parallel q \llbracket Q_k \rrbracket \parallel N_0$$

978 Case  $n > 1$ . Let  $v_1$  and  $N$  be as in the basic case,  $N'' = p \llbracket P_k \rrbracket \parallel q \llbracket Q_k \rrbracket \parallel N_0$  and  
 979  $\alpha = pq\lambda$ . Since  $v_1; \dots; v_n$  is a proving sequence, we have  $\neg(v_l \# v_{l'})$  for all  $l, l'$  such  
 980 that  $1 \leq l, l' \leq n$ . Moreover, for all  $l, 2 \leq l \leq n$  we have  $v_l \neq v_1 = \text{nec}(\alpha)$ , thus  $\alpha \diamond v_l$   
 981 is defined by Lemma 8.5. Let  $v'_l = \alpha \diamond v_l$  for all  $l, 2 \leq l \leq n$ , then  $v'_l \in \mathcal{NE}(N'')$  by  
 982 Lemma 8.6(2).

983 We show that  $v'_2; \dots; v'_n$  is a proving sequence in  $\mathcal{S}^N(N'')$ . First notice that for all  $l$ ,  
 984  $2 \leq l \leq n$ ,  $\neg(v_l \# v_{l'})$  implies  $\neg(v'_l \# v'_{l'})$  by Lemma 8.2(5) and (1). Let now  $v < v'_h$  for  
 985 some  $h, 2 \leq h \leq n$ . By Lemma 8.2(3) and (1)  $\alpha \diamond v < \alpha \diamond (\alpha \diamond v_h) = v_h$ . This implies  
 986 by Definition 3.6 that there is  $h' < h$  such that either  $\alpha \diamond v = v_{h'}$  or  $\alpha \diamond v \# v_{h'} < v_h$ .  
 987 Therefore, since  $v'_l$  is defined for all  $l, 2 \leq l \leq n$ , we get either  $v = v'_{h'}$  by Lemma 8.2(2)  
 988 or  $v \# v'_{h'} < v'_h$  by Lemma 8.2(6) and (4).

989 By induction  $N'' \xrightarrow{\sigma'} N'$  where  $\sigma' = \text{cm}(v'_2) \cdots \text{cm}(v'_n)$ . Since  $\text{cm}(v_l) = \text{cm}(v'_l)$  for all  $l$ ,  
 990  $2 \leq l \leq n$  we get  $\sigma = \alpha \cdot \sigma'$ . Hence  $N \xrightarrow{\alpha} N'' \xrightarrow{\sigma'} N'$  is the required transition sequence.

## 991 8.2. Relating Transition Sequences of Global Types and Proving Sequences of their ESs

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

### 996 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:

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

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

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

998 The operator  $\bullet$ , applied to a communication and a g-event, gives the g-event  
 999 obtained by erasing the communication, if it occurs in head position (modulo  $\sim$ ) in  
 1000 the given g-event, and leaves the g-event unchanged if its participants are disjoint  
 1001 from those of the communication. Note that the operator  $\alpha \bullet [\sigma]_{\sim}$  is undefined  
 1002 whenever either  $[\sigma]_{\sim} = \{\alpha\}$  or one of the participants of  $\alpha$  occurs in  $\sigma$  but the  
 1003 first communication of  $\sigma$  is different from  $\alpha$ . For example  $pq\lambda \bullet [pq\lambda]_{\sim}$  and  $pq\lambda \bullet$   
 1004  $[pq\lambda' \cdot \sigma]_{\sim}$  with  $\lambda \neq \lambda'$  are undefined for any  $\sigma$ .

1005 The following lemma gives some simple properties of the retrieval and residual  
 1006 operators for g-events. The first five statements correspond to those of Lemma 8.2  
 1007 for n-events. The last three statements give properties that are relevant only for the  
 1008 operators  $\circ$  and  $\bullet$ .

1009 **Lemma 8.10 (Properties of retrieval and residual for g-events).**

- 1011 1. If  $\alpha \bullet \gamma$  is defined, then  $\alpha \circ (\alpha \bullet \gamma) = \gamma$ ;
- 1012 2.  $\alpha \bullet (\alpha \circ \gamma) = \gamma$ ;
- 1013 3. If  $\gamma_1 < \gamma_2$ , then  $\alpha \circ \gamma_1 < \alpha \circ \gamma_2$ ;
- 1014 4. If  $\gamma_1 < \gamma_2$  and both  $\alpha \bullet \gamma_1$  and  $\alpha \bullet \gamma_2$  are defined, then  $\alpha \bullet \gamma_1 < \alpha \bullet \gamma_2$ ;
- 1015 5. If  $\gamma_1 \# \gamma_2$ , then  $\alpha \circ \gamma_1 \# \alpha \circ \gamma_2$ ;
- 1016 6. If  $\gamma < \alpha \circ \gamma'$ , then either  $\gamma = [\alpha]_{\sim}$  or  $\alpha \bullet \gamma < \gamma'$ ;
- 1017 7. If  $\text{part}(\alpha_1) \cap \text{part}(\alpha_2) = \emptyset$ , then  $\alpha_1 \circ (\alpha_2 \circ \gamma) = \alpha_2 \circ (\alpha_1 \circ \gamma)$ ;
- 1018 8. If  $\text{part}(\alpha_1) \cap \text{part}(\alpha_2) = \emptyset$  and both  $\alpha_2 \bullet (\alpha_1 \circ \gamma)$ ,  $\alpha_2 \bullet \gamma$  are defined, then  $\alpha_1 \circ (\alpha_2 \bullet \gamma) =$   
1019  $\alpha_2 \bullet (\alpha_1 \circ \gamma)$ .

1020 The next lemma relates the retrieval and residual operator with the global types  
1021 in the branches of choices.

1022 **Lemma 8.11.** *The following hold:*

- 1024 1. If  $\gamma \in \mathcal{GE}(\mathbf{G})$ , then  $\text{pq}\lambda \circ \gamma \in \mathcal{GE}(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$ , where  $\lambda = \lambda_k$  and  $\mathbf{G} = \mathbf{G}_k$   
1025 for some  $k \in I$ ;
- 1026 2. If  $\gamma \in \mathcal{GE}(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$  and  $\text{pq}\lambda_k \bullet \gamma$  is defined, then  $\text{pq}\lambda_k \bullet \gamma \in \mathcal{GE}(\mathbf{G}_k)$ ,  
1027 where  $k \in I$ .

1028 The following lemma plays the role of Lemma 8.6 for n-events.

1029 **Lemma 8.12.** *Let  $\mathbf{G} \xrightarrow{\alpha} \mathbf{G}'$ .*

- 1030 1. If  $\gamma \in \mathcal{GE}(\mathbf{G}')$ , then  $\alpha \circ \gamma \in \mathcal{GE}(\mathbf{G})$ ;
- 1031 2. If  $\gamma \in \mathcal{GE}(\mathbf{G})$  and  $\alpha \bullet \gamma$  is defined, then  $\alpha \bullet \gamma \in \mathcal{GE}(\mathbf{G}')$ .

1032 Each non-empty trace gives rise to a sequence of g-events, compare with Defi-  
1033 nition 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  $\sigma$  by*

$$\text{gec}(\sigma) = \gamma_1; \dots; \gamma_n$$

1034 where  $\gamma_i = \text{ev}(\sigma[1 \dots i])$  for all  $i$ ,  $1 \leq i \leq n$ .

1035 We show that  $\text{gec}(\cdot)$  has similar properties as  $\text{nec}(\cdot)$ , see Lemma 8.4(1). The  
1036 proof is straightforward.

1037 **Lemma 8.14.** *Let  $\text{gec}(\sigma) = \gamma_1; \dots; \gamma_n$ .*

- 1038 1.  $\text{cm}(\gamma_i) = \sigma[i]$  for all  $i$ ,  $1 \leq i \leq n$ .
- 1039 2. If  $1 \leq h, k \leq n$ , then  $\neg(\gamma_h \# \gamma_k)$ ;

1040 We may now prove the correspondence between the traces labelling the transi-  
 1041 tion sequences of a global type and the proving sequences of its PES. Let us stress  
 1042 the difference between the set of traces  $\text{Tr}^+(\mathbf{G})$  of a global type  $\mathbf{G}$  as defined at page  
 1043 20 and the set of traces that label the transition sequences of  $\mathbf{G}$ , which is a larger set  
 1044 due to the internal Rule [ICOMM] of the LTS for global types given in Figure 4.

1045 **Theorem 8.15.** *If  $\mathbf{G} \xrightarrow{\sigma} \mathbf{G}'$ , then  $\text{gec}(\sigma)$  is a proving sequence in  $\mathcal{S}^{\mathcal{G}}(\mathbf{G})$ .*

1046 **Proof** By induction on  $\sigma$ .

1047 *Base case.* Let  $\sigma = \alpha$ , then  $\text{gec}(\alpha) = [\alpha]_{\sim}$ . We use a further induction on the inference  
 1048 of the transition  $\mathbf{G} \xrightarrow{\alpha} \mathbf{G}'$ .

1049 Let  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$ ,  $\mathbf{G}' = \mathbf{G}_h$  and  $\alpha = \text{pq}\lambda_h$  for some  $h \in I$ . By Defini-  
 1050 tion 7.11(1)  $[\text{pq}\lambda_h]_{\sim} \in \mathcal{GE}(\mathbf{G})$ .

1051 Let  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\mathbf{G}' = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}'_i$  and  $\mathbf{G}_i \xrightarrow{\alpha} \mathbf{G}'_i$  for all  $i \in I$   
 1052 and  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . By induction  $[\alpha]_{\sim} \in \mathcal{GE}(\mathbf{G}_i)$  for all  $i \in I$ . By Lemma 8.11(1)  
 1053  $\text{pq}\lambda_i \circ [\alpha]_{\sim} \in \mathcal{GE}(\mathbf{G})$  for all  $i \in I$ . By Definition 7.11(1)  $\text{pq}\lambda_i \circ [\alpha]_{\sim} = [\alpha]_{\sim}$ , since  
 1054  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . We conclude  $[\alpha]_{\sim} \in \mathcal{GE}(\mathbf{G})$ .

1055 *Inductive case.* Let  $\sigma = \alpha \cdot \sigma'$  with  $\sigma' \neq \epsilon$ . From  $\mathbf{G} \xrightarrow{\sigma} \mathbf{G}'$  we get  $\mathbf{G} \xrightarrow{\alpha} \mathbf{G}'' \xrightarrow{\sigma'} \mathbf{G}'$  for  
 1056 some  $\mathbf{G}''$ . Let  $\text{gec}(\sigma) = \gamma_1; \dots; \gamma_n$  and  $\text{gec}(\sigma') = \gamma'_2; \dots; \gamma'_n$ . By induction  $\text{gec}(\sigma')$  is a  
 1057 proving sequence in  $\mathcal{S}^{\mathcal{G}}(\mathbf{G}'')$ . By Definitions 8.13 and 7.6  $\gamma_i = \alpha \circ \gamma'_i$ , which implies  
 1058  $\alpha \bullet \gamma_i = \gamma'_i$  by Lemma 8.10(2) for all  $i$ ,  $2 \leq i \leq n$ .

1059 We can show that  $\gamma_1 = [\alpha]_{\sim} \in \mathcal{GE}(\mathbf{G})$  as in the proof of the base case. By  
 1060 Lemma 8.12(1)  $\gamma_i \in \mathcal{GE}(\mathbf{G})$  since  $\gamma'_i \in \mathcal{GE}(\mathbf{G}'')$  and  $\alpha \bullet \gamma_i = \gamma'_i$  for all  $i$ ,  $2 \leq i \leq n$ . We  
 1061 prove that  $\text{gec}(\sigma)$  is a proving sequence in  $\mathcal{S}^{\mathcal{G}}(\mathbf{G})$ . Let  $\gamma < \gamma_k$  for some  $k$ ,  $1 \leq k \leq n$ .  
 1062 Note that this implies  $k > 1$ . Since  $\gamma_k = \alpha \circ \gamma'_k$  by Lemma 8.10(6) either  $\gamma = [\alpha]_{\sim}$  or  
 1063  $\alpha \bullet \gamma < \gamma'_k$ . If  $\gamma = [\alpha]_{\sim} = \gamma_1$  we are done. Otherwise  $\alpha \bullet \gamma \in \mathcal{GE}(\mathbf{G}'')$  by Lemma 8.11(2).  
 1064 Since  $\text{gec}(\sigma')$  is a proving sequence in  $\mathcal{S}^{\mathcal{G}}(\mathbf{G}'')$ , there is  $h < k$  such that  $\alpha \bullet \gamma = \gamma'_h$   
 1065 and this implies  $\gamma = \alpha \circ (\alpha \bullet \gamma) = \alpha \circ \gamma'_h = \gamma_h$  by Lemma 8.10(1).

1066 **Theorem 8.16.** *If  $\gamma_1; \dots; \gamma_n$  is a proving sequence in  $\mathcal{S}^{\mathcal{G}}(\mathbf{G})$ , then  $\mathbf{G} \xrightarrow{\sigma} \mathbf{G}'$ , where  $\sigma =$   
 1067  $\text{cm}(\gamma_1) \cdot \dots \cdot \text{cm}(\gamma_n)$ .*

1068 **Proof** The proof is by induction on the length  $n$  of the proving sequence. Let  
 1069  $\text{cm}(\gamma_1) = \alpha$  and  $\{\mathbf{p}, \mathbf{q}\} = \text{part}(\alpha)$ .

1070 *Case  $n = 1$ .* Since  $\gamma_1$  is the first event of a proving sequence, we have  $\gamma_1 = [\alpha]_{\sim}$ . We  
 1071 show this case by induction on  $d = \text{depth}(\mathbf{G}, \mathbf{p}) = \text{depth}(\mathbf{G}, \mathbf{q})$ .

1072 *Case  $d = 1$ .* Let  $\alpha = \text{pq}\lambda$  and  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\lambda = \lambda_h$  for some  $h \in I$ . Then  
 1073  $\mathbf{G} \xrightarrow{\alpha} \mathbf{G}_h$  by rule [ECOMM].

1074 *Case  $d > 1$ .* Let  $\mathbf{G} = \mathbf{r} \rightarrow \mathbf{s} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\{\mathbf{r}, \mathbf{s}\} \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . By Definition 8.9(1)  
 1075  $\text{rs}\lambda_i \bullet \gamma_1$  is defined for all  $i \in I$  since  $\{\mathbf{r}, \mathbf{s}\} \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . This implies  $\text{rs}\lambda_i \bullet \gamma_1 \in \mathcal{GE}(\mathbf{G}_i)$   
 1076 for all  $i \in I$  by Lemma 8.11(2). By induction hypothesis  $\mathbf{G}_i \xrightarrow{\alpha} \mathbf{G}'_i$  for all  $i \in I$ . Then  
 1077 we can apply rule [ICOMM] to derive  $\mathbf{G} \xrightarrow{\alpha} \mathbf{r} \rightarrow \mathbf{s} : \boxplus_{i \in I} \lambda_i; \mathbf{G}'_i$ .

1078 *Case  $n > 1$ .* Let  $G \xrightarrow{\alpha} G''$  be the transition as obtained from the base case. We  
 1079 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  $k$ ,  
 1080  $2 \leq k \leq n$ , then by Definition 8.9(1) either  $\gamma_k = \gamma_1$  or  $\gamma_k = [\sigma]_{\sim}$  with  $\sigma \not\sim \alpha \cdot \sigma'$  and  
 1081  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$ . In the second case  $\alpha @ p \# \sigma @ p$  or  $\alpha @ q \# \sigma @ q$ , which implies  
 1082  $\gamma_k \# \gamma_1$ . So both cases are impossible. If  $\alpha \bullet \gamma_j$  is defined, by Lemma 8.12(2) we get  
 1083  $\alpha \bullet \gamma_j \in \mathcal{GE}(G'')$  for all  $j, 2 \leq j \leq n$ .  
 1084 We show that  $\gamma'_2; \dots; \gamma'_n$  is a proving sequence in  $\mathcal{S}^{\mathcal{G}}(G'')$  where  $\gamma'_j = \alpha \bullet \gamma_j$  for all  $j$ ,  
 1085  $2 \leq j \leq n$ . 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)  
 1086 no two events in the sequence  $\gamma'_2; \dots; \gamma'_n$  can be in conflict. Let  $\gamma \in \mathcal{GE}(G'')$  and  
 1087  $\gamma < \gamma'_h$  for some  $h, 2 \leq h \leq n$ . By Lemma 8.12(1)  $\alpha \circ \gamma$  and  $\alpha \circ \gamma'_h$  belong to  $\mathcal{GE}(G)$ . By  
 1088 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  
 1089  $\gamma' < \gamma_h$  implies, by Definition 3.6 and the fact that  $\mathcal{S}^{\mathcal{G}}(G)$  is a PES, that there is  $k < h$   
 1090 such that  $\gamma' = \gamma_k$ . By Lemma 8.10(1) we get  $\gamma = \alpha \bullet \gamma' = \alpha \bullet \gamma_k = \gamma'_k$ .  
 1091 Since  $\gamma'_2; \dots; \gamma'_n$  is a proving sequence in  $\mathcal{S}^{\mathcal{G}}(G'')$ , by induction  $G'' \xrightarrow{\sigma'} G'$  where  
 1092  $\sigma' = \text{cm}(\gamma'_2) \cdot \dots \cdot \text{cm}(\gamma'_n)$ . Let  $\sigma = \text{cm}(\gamma_1) \cdot \dots \cdot \text{cm}(\gamma_n)$ . Since  $\text{cm}(\gamma'_j) = \text{cm}(\gamma_j)$  for  
 1093 all  $j, 2 \leq j \leq n$ , we have  $\sigma = \alpha \cdot \sigma'$ . Hence  $G \xrightarrow{\alpha} G'' \xrightarrow{\sigma'} G'$  is the required transition  
 1094 sequence.

1095 The last ingredient required to prove our main theorem is the following separa-  
 1096 tion result from [9] (Lemma 2.8 p. 12):

1097 **Lemma 8.17 (Separation [9]).** *Let  $S = (E, <, \#)$  be a flow event structure and  $X, X' \in$   
 1098  $C(S)$  be such that  $X \subset X'$ . Then there exist  $e \in X' \setminus X$  such that  $X \cup \{e\} \in C(S)$ .*

1099 We may now finally show the correspondence between the configurations of the  
 1100 FES of a network and the configurations of the PES of its global type. Let  $\simeq$  denote  
 1101 isomorphism on domains of configurations.

1102 **Theorem 8.18 (Isomorphism).** *If  $\vdash N : G$ , then  $\mathcal{D}(\mathcal{S}^{\mathcal{N}}(N)) \simeq \mathcal{D}(\mathcal{S}^{\mathcal{G}}(G))$ .*

1103 **Proof** By Theorem 8.8 if  $v_1; \dots; v_n$  is a proving sequence of  $\mathcal{S}^{\mathcal{N}}(N)$ , then  $N \xrightarrow{\sigma} N'$   
 1104 where  $\sigma = \text{cm}(v_1) \cdot \dots \cdot \text{cm}(v_n)$ . By applying iteratively Subject Reduction (Theo-  
 1105 rem 6.10)  $G \xrightarrow{\sigma} G'$  and  $\vdash N' : G'$ . By Theorem 8.15  $\text{gec}(\sigma)$  is a proving sequence of  
 1106  $\mathcal{S}^{\mathcal{G}}(G)$ .

1107 By Theorem 8.16 if  $\gamma_1; \dots; \gamma_n$  is a proving sequence of  $\mathcal{S}^{\mathcal{G}}(G)$ , then  $G \xrightarrow{\sigma} G'$   
 1108 where  $\sigma = \text{cm}(\gamma_1) \cdot \dots \cdot \text{cm}(\gamma_n)$ . By applying iteratively Session Fidelity (Theorem 6.11)  
 1109  $N \xrightarrow{\sigma} N'$  and  $\vdash N' : G'$ . By Theorem 8.7  $\text{nec}(\sigma)$  is a proving sequence of  $\mathcal{S}^{\mathcal{N}}(N)$ .

1110 Therefore we have a bijection between  $\mathcal{D}(\mathcal{S}^{\mathcal{N}}(N))$  and  $\mathcal{D}(\mathcal{S}^{\mathcal{G}}(G))$ , given by  
 1111  $\text{nec}(\sigma) \leftrightarrow \text{gec}(\sigma)$  for any  $\sigma$  generated by the (bisimilar) LTSs of  $N$  and  $G$ .

1112 We show now that this bijection preserves inclusion of configurations. By  
 1113 Lemma 8.17 it is enough to prove that if  $v_1; \dots; v_n \in C(\mathcal{S}^{\mathcal{N}}(N))$  is mapped to  
 1114  $\gamma_1; \dots; \gamma_n \in C(\mathcal{S}^{\mathcal{G}}(G))$ , then  $v_1; \dots; v_n; v \in C(\mathcal{S}^{\mathcal{N}}(N))$  iff  $\gamma_1; \dots; \gamma_n; \gamma \in C(\mathcal{S}^{\mathcal{G}}(G))$ ,  
 1115 where  $\gamma_1; \dots; \gamma_n; \gamma$  is the image of  $v_1; \dots; v_n; v$  under the bijection. I.e. let  $\text{nec}(\sigma \cdot \alpha) =$   
 1116  $v_1; \dots; v_n; v$  and  $\text{gec}(\sigma \cdot \alpha) = \gamma_1; \dots; \gamma_n; \gamma$ . This implies  $\sigma = \text{cm}(v_1) \cdot \dots \cdot \text{cm}(v_n) =$   
 1117  $\text{cm}(\gamma_1) \cdot \dots \cdot \text{cm}(\gamma_n)$  and  $\alpha = \text{cm}(v) = \text{cm}(\gamma)$  by Lemmas 8.4 and 8.14.

1118 By Theorem 8.8, if  $\nu_1; \dots; \nu_n; \nu$  is a proving sequence of  $\mathcal{S}^N(N)$ , then  $N \xrightarrow{\sigma} N_0 \xrightarrow{\alpha}$   
 1119  $N'$ . By applying iteratively Subject Reduction (Theorem 6.10)  $G \xrightarrow{\sigma} G_0 \xrightarrow{\alpha} G'$  and  
 1120  $\vdash N' : G'$ . By Theorem 8.15  $\text{gec}(\sigma \cdot \alpha)$  is a proving sequence of  $\mathcal{S}^G(G)$ .

1121 By Theorem 8.16, if  $\gamma_1; \dots; \gamma_n; \gamma$  is a proving sequence of  $\mathcal{S}^G(G)$ , then  $G \xrightarrow{\sigma}$   
 1122  $G_0 \xrightarrow{\alpha} G'$ . By applying iteratively Session Fidelity (Theorem 6.11)  $N \xrightarrow{\sigma} N_0 \xrightarrow{\alpha} N'$   
 1123 and  $\vdash N' : G'$ . By Theorem 8.7  $\text{nec}(\sigma \cdot \alpha)$  is a proving sequence of  $\mathcal{S}^N(N)$ .

## 1124 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):

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

1125 More sophisticated ES semantics for CCS, based on FESs and designed to be  
 1126 robust under action refinement [1, 26, 34], were subsequently proposed by Goltz  
 1127 and van Glabbeek [57]. Importantly, all the above-mentioned classes of ESs, except  
 1128 General ESs, give rise to the same *prime algebraic domains* of configurations, from  
 1129 which one can recover a PES by selecting the complete prime elements.

1130 More recently, ES semantics have been investigated for the  $\pi$ -calculus by Crafa,  
 1131 Varacca and Yoshida [21, 58, 22] and by Cristescu, Krivine and Varacca [23, 24, 25].  
 1132 Previously, other causal models for the  $\pi$ -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  $\pi$ -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  $\pi$ -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  $P = \nu a (\bar{b}\langle a \rangle \mid \bar{c}\langle a \rangle \mid a)$ , 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  $\pi$ -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  $\bar{b}\langle a \rangle$ ,  $\bar{c}\langle a \rangle$  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  $\pi$ -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.



- 1178 • Concerning the choice operator, we adopted here the basic (and most restric-  
 1179 tive) variant for it, as it was originally proposed for multiparty session calculi  
 1180 in [39]. This is essentially a simplifying assumption, and we do not foresee any  
 1181 difficulty in extending our results to a more general choice operator, where the  
 1182 projection is rendered more flexible through the use of a merge operator [31].
- 1183 • As regards the preorder on processes, which is akin to a subtyping relation,  
 1184 we envisaged to use the standard subtyping, in which a process with fewer  
 1185 outputs can be used in place of a process with more outputs. However, in that  
 1186 case Session Fidelity would become weaker, since a transition in the LTS of  
 1187 a global type would only ensure a transition in the LTS of the corresponding  
 1188 network, but not necessarily with the same labelling communication. The  
 1189 main drawback would be that Theorem 8.18 would no longer hold: more  
 1190 precisely, the domains of network configurations would only be embedded  
 1191 in (and not isomorphic to) the domains of their global type configurations.  
 1192 Notably, typability is independent from the use of our preorder or of the  
 1193 standard one, as proved in [4].

1194 As regards future work, we plan to define an asynchronous transition system  
 1195 (ATS) [5] for our calculus, along the lines of [10], and show that it provides a  
 1196 noninterleaving operational semantics for networks that is equivalent to their FES  
 1197 semantics. This would enable us also to investigate the issue of reversibility, jointly  
 1198 on our networks and on their FES representations, since the ATS semantics would  
 1199 give us the handle to unwind networks, while the corresponding FESs could be  
 1200 unrolled following one of the methods proposed in existing work on reversible  
 1201 event structures [53, 25, 36, 37, 35].

1202 As mentioned at the end of Section 7, the quest for a semantic counterpart  
 1203 of our well-formedness conditions on global types – namely, for properties that  
 1204 characterise the FESs obtained from typable networks – is still open. By way  
 1205 of comparison, such semantic well-formedness conditions have been proposed  
 1206 in [56] for *graphical choreographies*, a truly concurrent graphical model for global  
 1207 specifications with two kinds of forking nodes, representing respectively choice  
 1208 and parallel composition. In [56], those well-formedness conditions, called *well-*  
 1209 *sequencing* and *well-branchedness*, were shown to be sufficient to ensure projectability  
 1210 on local specifications. In our case, the property corresponding to well-sequencing  
 1211 is automatically ensured by our ES semantics, and we conjecture that the well-  
 1212 branchedness condition for choice nodes (corresponding to projectability) could  
 1213 amount in our simpler setting<sup>10</sup> to the following semantic condition:

1214 Let  $v_1, v_2 \in \mathcal{NE}(N)$  and  $p :: \zeta \cdot \pi \in v_1$  and  $p :: \zeta \cdot \pi' \in v_2$  with  $\pi \neq \pi'$  and  
 1215  $q = \text{pt}(\pi) = \text{pt}(\pi')$ . If  $v_1 <^* v'_1$  for some  $v'_1 \in \mathcal{NE}(N)$  such that  $r \in \text{loc}(v'_1)$  with  
 1216  $r \notin \{p, q\}$ , then  $v_2 <^* v'_2$  for some  $v'_2 \in \mathcal{NE}(N)$  such that  $r \in \text{loc}(v'_2)$ .

1217 This condition would allow us to rule out the FESs of both networks  $N'$  and  $N''$   
 1218 discussed at the end of Section 7. However, it should be completed with a condition  
 1219 corresponding to boundedness, and the conjunction of these two conditions might

---

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

1220 still not be sufficient in general to ensure typability. We plan to further investigate  
1221 this question in the near future.

1222 *Acknowledgment.* We are strongly indebted to the anonymous referees for their  
1223 constructive remarks, which helped us improve the submitted version of the paper.

## 1224 References

- 1225 [1] Luca Aceto and Matthew Hennessy. Towards action-refinement in process  
1226 algebras. In Albert R. Meyer, editor, *LICS*, pages 138–145, Washington, 1989.  
1227 IEEE Computer Society Press.
- 1228 [2] Christel Baier and Mila E. Majster-Cederbaum. The connection between an  
1229 event structure semantics and an operational semantics for TCSP. *Acta Informatica*, 31(1):81–104, 1994.
- 1231 [3] Paolo Baldan, Andrea Corradini, and Fabio Gadducci. Domains and event  
1232 structures for fusions. In Joel Ouaknine, editor, *LICS*, pages 1–12, Washington,  
1233 2017. IEEE Computer Society Press.
- 1234 [4] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Emilio  
1235 Tuosto. Composition and decomposition of multiparty sessions. *Journal of*  
1236 *Logical and Algebraic Methods in Programming*, 119:100620, 2021.
- 1237 [5] Marek Bednarczyk. *Categories of Asynchronous Systems*. PhD thesis, University  
1238 of Sussex, 1988.
- 1239 [6] Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality  
1240 in the  $\pi$ -calculus. *Acta Informatica*, 35(5):353–400, 1998.
- 1241 [7] Gérard Boudol and Ilaria Castellani. On the semantics of concurrency: partial  
1242 orders and transition systems. In Hartmut Ehrig, Robert A. Kowalski, Giorgio  
1243 Levi, and Ugo Montanari, editors, *TAPSOFT*, volume 249 of *LNCS*, pages 123–  
1244 137, Heidelberg, 1987. Springer.
- 1245 [8] Gérard Boudol and Ilaria Castellani. Permutation of transitions: an event struc-  
1246 ture semantics for CCS and SCCS. In Jaco W. de Bakker, Willem P. de Roever,  
1247 and Grzegorz Rozenberg, editors, *REX: Linear Time, Branching Time and Partial*  
1248 *Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 411–427,  
1249 Heidelberg, 1988. Springer.
- 1250 [9] Gérard Boudol and Ilaria Castellani. Flow models of distributed computations:  
1251 event structures and nets. Research Report 1482, INRIA, 1991.
- 1252 [10] Gérard Boudol and Ilaria Castellani. Flow models of distributed computations:  
1253 three equivalent semantics for CCS. *Information and Computation*, 114(2):247–  
1254 314, 1994.
- 1255 [11] Stephen Brookes, Charles A.R. Hoare, and Andrew Roscoe. A theory of com-  
1256 municating sequential processes. *Journal of ACM*, 31(3):560–599, 1984.

- 1257 [12] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Event structure  
1258 semantics for nominal calculi. In Christel Baier and Holger Hermanns, editors,  
1259 *CONCUR*, volume 4137 of *LNCS*, pages 295–309, Heidelberg, 2006. Springer.
- 1260 [13] Luís Caires and Frank Pfenning. Session types as intuitionistic linear propo-  
1261 sitions. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume  
1262 6269 of *LNCS*, pages 222–236, Heidelberg, 2010. Springer.
- 1263 [14] Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions  
1264 as session types. *Mathematical Structures in Computer Science*, 26(3):367–423,  
1265 2016.
- 1266 [15] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. Event  
1267 structure semantics for multiparty sessions. In Michele Boreale, Flavio Corra-  
1268 dini, Michele Loreti, and Rosario Pugliese, editors, *Models, Languages, and Tools  
1269 for Concurrent and Distributed Programming - Essays Dedicated to Rocco De Nicola  
1270 on the Occasion of His 65th Birthday*, volume 11665 of *LNCS*, pages 340–363,  
1271 Heidelberg, 2019. Springer.
- 1272 [16] Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Paola Giannini. Global  
1273 types and event structure semantics for asynchronous multiparty sessions.  
1274 *CoRR*, abs/2102.00865, 2021.
- 1275 [17] Ilaria Castellani and Guo Qiang Zhang. Parallel product of event structures.  
1276 *Theoretical Computer Science*, 179(1-2):203–215, 1997.
- 1277 [18] Gian Luca Cattani and Peter Sewell. Models for name-passing processes:  
1278 interleaving and causal. *Information and Computation*, 190(2):136–178, 2004.
- 1279 [19] Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida, and Luca  
1280 Padovani. Global progress for dynamically interleaved multiparty sessions.  
1281 *Mathematical Structures in Computer Science*, 26(2):238–302, 2016.
- 1282 [20] Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer  
1283 Science*, 25:95–169, 1983.
- 1284 [21] Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Compositional event  
1285 structure semantics for the internal  $\pi$ -calculus. In Luís Caires and Vasco T.  
1286 Vasconcelos, editors, *CONCUR*, volume 4703 of *LNCS*, pages 317–332, Heidel-  
1287 berg, 2007. Springer.
- 1288 [22] Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Event structure semantics  
1289 of parallel extrusion in the  $\pi$ -calculus. In Lars Birkedal, editor, *FOSSACS*,  
1290 volume 7213 of *LNCS*, pages 225–239, Heidelberg, 2012. Springer.
- 1291 [23] Ioana Cristescu. *Operational and denotational semantics for the reversible  $\pi$ -  
1292 calculus*. PhD thesis, University Paris Diderot - Paris 7, 2015.

- [24] Ioana Cristescu, Jean Krivine, and Daniele Varacca. Rigid families for CCS and the  $\pi$ -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  $\pi$ -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.
- [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 Deniérou and Nobuko Yoshida. Dynamic multirole session types. In Mooly Sagiv, editor, *POPL*, pages 435–446, New York, 2011. ACM Press.
- [32] Pierre-Malo Deniérou 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.

- 1331 [38] Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language primitives  
1332 and type discipline for structured communication-based programming. In  
1333 Chris Hankin, editor, *ESOP*, volume 1381 of *LNCS*, pages 122–138, Heidelberg,  
1334 1998. Springer.
- 1335 [39] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous  
1336 session types. In George C. Necula and Philip Wadler, editors, *POPL*, pages  
1337 273–284, New York, 2008. ACM Press.
- 1338 [40] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous  
1339 session types. *Journal of ACM*, 63(1):9:1–9:67, 2016.
- 1340 [41] Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone,  
1341 Pierre-Malo Deniérou, Dimitris Mostros, Luca Padovani, António Ravara,  
1342 Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of  
1343 session types and behavioural contracts. *ACM Computing Surveys*, 49(1):3:1–  
1344 3:36, 2016.
- 1345 [42] Lalita Jategaonkar Jagadeesan and Radha Jagadeesan. Causality and true  
1346 concurrency: A data-flow analysis of the  $\pi$ -calculus (extended abstract). In  
1347 Vangalur S. Alagar and Maurice Nivat, editors, *AMAST*, volume 936 of *LNCS*,  
1348 pages 277–291, Heidelberg, 1995. Springer.
- 1349 [43] Joost-Pieter Katoen. *Quantitative and qualitative extensions of event structures*.  
1350 PhD thesis, University of Twente, 1996.
- 1351 [44] Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From communicating  
1352 machines to graphical choreographies. In Sriram K. Rajamani and David  
1353 Walker, editors, *POPL*, pages 221–232, New York, 2015. ACM Press.
- 1354 [45] Rom Langerak. Bundle event structures: a non-interleaving semantics for LO-  
1355 TOS. In Michael Diaz and Roland Groz, editors, *Formal Description Techniques*  
1356 *for Distributed Systems and Communication Protocols*, pages 331–346, Amster-  
1357 dam, 1993. North-Holland.
- 1358 [46] Rita Loogen and Ursula Goltz. Modelling nondeterministic concurrent pro-  
1359 cesses with event structures. *Fundamenta Informaticae*, 14(1):39–74, 1991.
- 1360 [47] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*.  
1361 Springer, Heidelberg, 1980.
- 1362 [48] Ugo Montanari and Marco Pistore. Concurrent semantics for the  $\pi$ -calculus. In  
1363 Stephen Brookes, Michael Main, Austin Melton, and Michael Mislove, editors,  
1364 *MFPS*, volume 1 of *ENTCS*, pages 411–429, Oxford, 1995. Elsevier.
- 1365 [49] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event struc-  
1366 tures and domains, part I. *Theoretical Computer Science*, 13(1):85–108, 1981.

- 1367 [50] Ernst-Rüdiger Olderog. TCSP: theory of communicating sequential processes.  
1368 In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Ad-  
1369 vances in Petri Nets*, volume 255 of *LNCS*, pages 441–465, Heidelberg, 1986.  
1370 Springer.
- 1371 [51] Luca Padovani. Type reconstruction for the linear  $\pi$ -calculus with composite  
1372 regular types. *Logical Methods in Computer Science*, 11(4), 2015.
- 1373 [52] Jorge A. Pérez, Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear  
1374 logical relations and observational equivalences for session-based concurrency.  
1375 *Information and Computation*, 239:254–302, 2014.
- 1376 [53] Iain Phillips and Irek Ulidowski. Reversibility and asymmetric conflict in event  
1377 structures. *Journal of Logical and Algebraic Methods in Programming*, 84(6):781 –  
1378 805, 2015.
- 1379 [54] Kaku Takeuchi, Kohei Honda, and Makoto Kubo. An interaction-based lan-  
1380 guage and its typing system. In Chris Hankin, editor, *PARLE*, volume 817 of  
1381 *LNCS*, pages 122–138, Heidelberg, 1994. Springer.
- 1382 [55] Bernardo Toninho, Luís Caires, and Frank Pfenning. Dependent session types  
1383 via intuitionistic linear type theory. In Peter Schneider-Kamp and Michael  
1384 Hanus, editors, *PPDP*, pages 161–172, New York, 2011. ACM Press.
- 1385 [56] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreogra-  
1386 phies. *Journal of Logic and Algebraic Methods in Programming*, 95:17–40, 2018.
- 1387 [57] Rob J. van Glabbeek and Ursula Goltz. Well-behaved flow event structures  
1388 for parallel composition and action refinement. *Theoretical Computer Science*,  
1389 311(1-3):463–478, 2004.
- 1390 [58] Daniele Varacca and Nobuko Yoshida. Typed event structures and the linear  
1391  $\pi$ -calculus. *Theoretical Computer Science*, 411(19):1949–1973, 2010.
- 1392 [59] Philip Wadler. Propositions as sessions. *Journal of Functional Programming*,  
1393 24(2-3):384–418, 2014.
- 1394 [60] Glynn Winskel. *Events in Computation*. PhD thesis, University of Edinburgh,  
1395 1980.
- 1396 [61] Glynn Winskel. Event structure semantics for CCS and related languages. In  
1397 Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of  
1398 *LNCS*, pages 561–576, Heidelberg, 1982. Springer.
- 1399 [62] Glynn Winskel. An introduction to event structures. In Jaco W. de Bakker,  
1400 Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX: Linear Time,  
1401 Branching Time and Partial Order in Logics and Models for Concurrency*, volume  
1402 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  $v$  and  $v'$  be binary  $n$ -events with  $\text{loc}(v) = \text{loc}(v')$ . Then  $v \# v'$  iff  $\mathbf{p} :: \eta \in v$  and  $\mathbf{p} :: \eta' \in v'$  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  $v = \{\mathbf{p} :: \eta_1, \mathbf{q} :: \eta_2\}$  the condition  $\mathbf{p} :: \eta_1 \widehat{\bowtie} \mathbf{q} :: \eta_2$  of Definition 5.5 implies  $\eta_1 \dot{\bowtie} \mathbf{q} \bowtie \eta_2 \dot{\bowtie} \mathbf{p}$  by Definition 5.4, which in turn implies  $|\eta_1 \dot{\bowtie} \mathbf{q}| = |\eta_2 \dot{\bowtie} \mathbf{p}|$  by Definition 5.3. If  $v$  is a binary event, we also have  $|\eta_1| = |\eta_1 \dot{\bowtie} \mathbf{q}|$  and  $|\eta_2| = |\eta_2 \dot{\bowtie} \mathbf{p}|$  by Definition 5.2, since all the actions of  $\eta_1$  involve  $\mathbf{q}$  and all the actions of  $\eta_2$  involve  $\mathbf{p}$ , and thus the projections do not erase actions. Assume now  $v' = \{\mathbf{p} :: \eta'_1, \mathbf{q} :: \eta'_2\}$ . We consider two cases (the others being symmetric):

- $v \# v'$  because  $\eta_1 \# \eta'_1$ . Then  $\eta_1 \dot{\bowtie} \mathbf{q} \bowtie \eta_2 \dot{\bowtie} \mathbf{p}$  and  $\eta'_1 \dot{\bowtie} \mathbf{q} \bowtie \eta'_2 \dot{\bowtie} \mathbf{p}$  imply  $\eta_2 \# \eta'_2$ ;
- $v \# v'$  because  $|\eta_1 \dot{\bowtie} \mathbf{q}| = |\eta'_2 \dot{\bowtie} \mathbf{p}|$  and  $\neg(\eta_1 \dot{\bowtie} \mathbf{q} \bowtie \eta'_2 \dot{\bowtie} \mathbf{p})$ . As argued before, we have  $|\eta_2 \dot{\bowtie} \mathbf{p}| = |\eta_1 \dot{\bowtie} \mathbf{q}|$  and  $|\eta'_2 \dot{\bowtie} \mathbf{p}| = |\eta'_1 \dot{\bowtie} \mathbf{q}|$ . Then, from  $|\eta_1 \dot{\bowtie} \mathbf{q}| = |\eta'_2 \dot{\bowtie} \mathbf{p}|$  and the above remark about binary events, we get  $|\eta_2| = |\eta_1| = |\eta'_2| = |\eta'_1|$ . From  $\neg(\eta_1 \dot{\bowtie} \mathbf{q} \bowtie \eta'_2 \dot{\bowtie} \mathbf{p})$  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  $v, v' \in \mathcal{NE}$  and  $v \neq v'$  and  $(v \cap v') \neq \emptyset$ , then  $v \# v'$ .*

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

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

**Proof** By Definition 5.11, if  $v \in \mathcal{NE}(\mathcal{N})$ , then  $v$  has at least one causal set included in  $\mathcal{NE}(\mathcal{N})$ . Let  $E' = \{v' \in \mathcal{X} \mid v' < v\}$ . By Definition 3.4,  $E' \cup \{v\}$  is conflict-free. Moreover, if  $\mathbf{p} :: \eta \in v$  and  $\eta' < \eta$ , then by Lemma 5.21 there is at most one  $v'' \in E'$  such that  $\mathbf{p} :: \eta' \in v''$ . Therefore,  $E' \subseteq E$  for some causal set  $E$  of  $v$  by Definition 5.9. We show that  $E \subseteq E'$ . Assume ad absurdum that  $v_0 \in E \setminus E'$ . By definition of causal set,  $v_0 < v$ . By definition of  $E'$ ,  $v_0 \notin E'$  implies  $v_0 \notin \mathcal{X}$ . By Definition 3.4 this implies  $v_0 \# v_1 < v$  for some  $v_1 \in \mathcal{X}$ . Then  $v_1 \in E'$  by definition of  $E'$ , and thus  $v_1 \in E$ . Hence  $v_0, v_1 \in E$  and  $v_0 \# v_1$ , contradicting Definition 5.9.

## 1439 B. Proofs of Section 6

1440 This section contains the proofs of Lemmas 6.6, 6.9, Theorems 6.10, 6.11 and of  
1441 the auxiliary Lemmas B.1, B.2, B.3.

1442 **Lemma 6.6** *If  $G$  is bounded, then  $G \upharpoonright r$  is a partial function for all  $r$ .*

1443 **Proof** We redefine the projection  $\downarrow_r$  as the largest relation between global types and  
1444 processes such that  $(G, P) \in \downarrow_r$  implies:

- 1445 i) if  $r \notin \text{part}(G)$ , then  $P = 0$ ;
- 1446 ii) if  $G = r \rightarrow p : \boxplus_{i \in I} \lambda_i; G_i$ , then  $P = \bigoplus_{i \in I} q! \lambda_i; P_i$  and  $(G_i, P_i) \in \downarrow_r$  for all  $i \in I$ ;
- 1447 iii) if  $G = p \rightarrow r : \boxplus_{i \in I} \lambda_i; G_i$ , then  $P = \sum_{i \in I} p? \lambda_i; P_i$  and  $(G_i, P_i) \in \downarrow_r$  for all  $i \in I$ ;
- 1448 iv) if  $G = p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i$  and  $r \notin \{p, q\}$  and  $r \in \text{part}(G_i)$ , then  $(G_i, P) \in \downarrow_r$  for all  
1449  $i \in I$ .

1450 The equality  $\mathcal{E}$  of processes is the largest symmetric binary relation  $\mathcal{R}$  on processes  
1451 such that  $(P, Q) \in \mathcal{R}$  implies:

- 1452 (a) if  $P = \bigoplus_{i \in I} p! \lambda_i; P_i$ , then  $Q = \bigoplus_{i \in I} p! \lambda_i; Q_i$  and  $(P_i, Q_i) \in \mathcal{R}$  for all  $i \in I$ ;
- 1453 (b) if  $P = \sum_{i \in I} p? \lambda_i; P_i$ , then  $Q = \sum_{i \in I} p? \lambda_i; Q_i$  and  $(P_i, Q_i) \in \mathcal{R}$  for all  $i \in I$ .

It is then enough to show that the relation

$$\mathcal{R}_r = \{(P, Q) \mid \exists G. (G, P) \in \downarrow_r \text{ and } (G, Q) \in \downarrow_r\}$$

1454 satisfies Clauses (a) and (b) (with  $\mathcal{R}$  replaced by  $\mathcal{R}_r$ ), since this will imply  $\mathcal{R}_r \subseteq \mathcal{E}$ .  
1455 Note first that  $(0, 0) \in \mathcal{R}_r$  because  $(\text{End}, 0) \in \downarrow_r$ , and that  $(0, 0) \in \mathcal{E}$  because Clauses (a)  
1456 and (b) are vacuously satisfied by the pair  $(0, 0)$ . The proof is by induction on  
1457  $d = \text{depth}(G, r)$ . We only consider Clause (b), the proof for Clause (a) being similar.  
1458 So, assume  $(P, Q) \in \mathcal{R}_r$  and  $P = \sum_{i \in I} p? \lambda_i; P_i$ .

1459 *Case  $d = 1$ .* In this case  $G = p \rightarrow r : \boxplus_{i \in I} \lambda_i; G_i$  and  $P = \sum_{i \in I} p? \lambda_i; P_i$  and  $(G_i, P_i) \in \downarrow_r$   
1460 for all  $i \in I$ . From  $(G, Q) \in \downarrow_r$  we get  $Q = \sum_{i \in I} p? \lambda_i; Q_i$  and  $(G_i, Q_i) \in \downarrow_r$  for all  $i \in I$ .  
1461 Hence  $Q$  has the required form and  $(P_i, Q_i) \in \mathcal{R}_r$  for all  $i \in I$ .

1462 *Case  $d > 1$ .* In this case  $G = p \rightarrow q : \boxplus_{j \in J} \lambda'_j; G_j$  and  $r \notin \{p, q\}$  and  $(G_j, P) \in \downarrow_r$  for all  
1463  $j \in J$ . From  $(G, Q) \in \downarrow_r$  we get  $(G_j, Q) \in \downarrow_r$  for all  $j \in J$ . Then  $(P, Q) \in \mathcal{R}_r$ .

1464 We need a lemma relating the projections of a well-formed global type with its  
1465 transitions.

1466 **Lemma B.1.** *Let  $G$  be a well-formed global type.*

- 1467 1. If  $G \upharpoonright p = \bigoplus_{i \in I} q! \lambda_i; P_i$  and  $G \upharpoonright q = \sum_{j \in J} p? \lambda'_j; Q_j$ , then  $I = J$ ,  $\lambda_i = \lambda'_i$ ,  $G \xrightarrow{pq\lambda_i} G_i$ ,  
1468  $G_i \upharpoonright p = P_i$  and  $G_i \upharpoonright q = Q_i$  for all  $i \in I$ .
- 1469 2. If  $G \xrightarrow{pq\lambda} G'$ , then  $G \upharpoonright p = \bigoplus_{i \in I} q! \lambda_i; P_i$ ,  $G \upharpoonright q = \sum_{i \in I} p? \lambda_i; Q_i$ , where  $\lambda_k = \lambda$  for  
1470 some  $k \in I$ , and  $G' \upharpoonright r = G \upharpoonright r$  for all  $r \notin \{p, q\}$ .



1471 **Proof** (1). The proof is by induction on  $d = \text{depth}(G, p)$ .  
1472 If  $d = 1$ , then by definition of projection (see Figure 2)  $G \upharpoonright p = \bigoplus_{i \in I} q! \lambda_i; P_i$  implies  
1473  $G = p \rightarrow q : \boxplus_{i \in I} \lambda_i; G_i$  with  $G_i \upharpoonright p = P_i$ . By the same definition  $G \upharpoonright q = \sum_{j \in J} p? \lambda'_j; Q_j$   
1474 implies  $J = I$  and  $\lambda'_j = \lambda_j$  and  $Q_j = G_j \upharpoonright q$  for all  $j \in J$ . Moreover  $G \xrightarrow{\text{pq}\lambda_i} G_i$  by Rule  
1475 [Ecomm] for all  $i \in I$ .  
1476 If  $d > 1$ , then  $G = r \rightarrow s : \boxplus_{h \in H} \lambda''_h; G'_h$  with  $\{p, q\} \cap \{r, s\} = \emptyset$ . By definition of  
1477 projection  $G \upharpoonright p = G'_h \upharpoonright p$  and  $G \upharpoonright q = G'_h \upharpoonright q$  for all  $h \in H$ . By Lemma 6.5  
1478  $\text{depth}(G, p) > \text{depth}(G'_h, p)$  for all  $h \in H$ . Then by induction  $I = J$ ,  $\lambda_i = \lambda'_i$ ,  
1479  $G'_h \xrightarrow{\text{pq}\lambda_i} G^i_h$ ,  $G^i_h \upharpoonright p = P_i$  and  $G^i_h \upharpoonright q = Q_i$  for all  $i \in I$  and all  $h \in H$ . Let  
1480  $G_i = r \rightarrow s : \boxplus_{h \in H} \lambda''_h; G^i_h$ . By Rule [Icomm]  $G \xrightarrow{\text{pq}\lambda_i} G_i$  for all  $i \in I$ . By defini-  
1481 tion of projection  $G_i \upharpoonright p = P_i$  and  $G_i \upharpoonright q = Q_i$  for all  $i \in I$ .  
1482 (2). The proof is by induction on the transition rules of Figure 4.

The interesting case is: 
$$\frac{G_h \xrightarrow{\text{pq}\lambda} G'_h \quad h \in H \quad \{p, q\} \cap \{s, t\} = \emptyset}{s \rightarrow t : \boxplus_{h \in H} \lambda'_h; G_h \xrightarrow{\text{pq}\lambda} s \rightarrow t : \boxplus_{h \in H} \lambda'_h; G'_h} \quad [\text{Icomm}]$$
  
1483 with  $G = s \rightarrow t : \boxplus_{h \in H} \lambda'_h; G_h$  and  $G' = s \rightarrow t : \boxplus_{h \in H} \lambda'_h; G'_h$ . By induction  
1484  $G_h \upharpoonright p = \bigoplus_{i \in I} q! \lambda_i; P_i$ ,  $G_h \upharpoonright q = \sum_{i \in I} p? \lambda_i; Q_i$ ,  $\lambda = \lambda_k$  for some  $k \in I$  and  $G'_h \upharpoonright r = G_h \upharpoonright r$   
1485 for all  $r \notin \{p, q\}$  and all  $h \in H$ . By definition of projection  $G \upharpoonright p = G_h \upharpoonright p$  and  
1486  $G \upharpoonright q = G_h \upharpoonright q$  for all  $h \in H$ . For  $r \notin \{p, q, s, t\}$  we get  $G' \upharpoonright r = G'_h \upharpoonright r = G_h \upharpoonright r = G \upharpoonright r$ .  
1487 Moreover  $G' \upharpoonright s = \bigoplus_{h \in H} t! \lambda'_h; G'_h \upharpoonright s = \bigoplus_{h \in H} t! \lambda'_h; G_h \upharpoonright s = G \upharpoonright s$  and  $G' \upharpoonright t =$   
1488  $\sum_{h \in H} t? \lambda'_h; G'_h \upharpoonright t = \sum_{h \in H} t? \lambda'_h; G_h \upharpoonright t = G \upharpoonright t$ .

1490 **Lemma 6.9** If  $G$  is a well-formed global type and  $G \xrightarrow{\text{pq}\lambda} G'$ , then  $G'$  is a well-formed  
1491 global type.

1492 **Proof** If  $G \xrightarrow{\text{pq}\lambda} G'$ , by Lemma B.1(1) and (2)  $G' \upharpoonright r$  is defined for all  $r$ . The proof  
1493 that  $\text{depth}(G'', r)$  is finite for all  $r$  and  $G''$  subtree of  $G'$  is easy by induction on the  
1494 transition rules of Figure 4.

1495 The proofs of Subject Reduction and Session Fidelity rely on the Inversion and  
1496 Canonical Form lemmas whose proofs are immediate.

1497 **Lemma B.2 (Inversion).** If  $\vdash N : G$ , then  $P \leq G \upharpoonright p$  for all  $p \llbracket P \rrbracket \in N$ .

1498 **Lemma B.3 (Canonical Form).** If  $\vdash N : G$  and  $p \in \text{part}(G)$ , then  $p \llbracket P \rrbracket \in N$  and  
1499  $P \leq G \upharpoonright p$ .

1500 **Theorem 6.10 (Subject Reduction)** If  $\vdash N : G$  and  $N \xrightarrow{\alpha} N'$ , then  $G \xrightarrow{\alpha} G'$  and  $\vdash N' : G'$ .

1501 **Proof** Let  $\alpha = \text{pq}\lambda$ . By Rule [Com] of Figure 1,  $N \equiv p \llbracket P \rrbracket \parallel q \llbracket Q \rrbracket \parallel N''$  where  
1502  $P = \bigoplus_{i \in I} q! \lambda_i; P_i$  and  $Q = \sum_{j \in J} p? \lambda_j; Q_j$  and  $N' \equiv p \llbracket P_h \rrbracket \parallel q \llbracket Q_h \rrbracket \parallel N''$  and  $\lambda = \lambda_h$   
1503 for some  $h \in I \cap J$ . From Lemma B.2 we get

1504 1.  $G \upharpoonright p = \bigoplus_{i \in I} q! \lambda_i; P'_i$  with  $P_i \leq P'_i$  for all  $i \in I$ , from Rule [ $\leq$ -Out] of Figure 3,  
1505 and

1506 2.  $G \vdash q = \sum_{j \in J'} p? \lambda_j; Q'_j$  with  $Q_j \leq Q'_j$  for all  $j \in J' \subseteq J$ , from Rule  $[\leq -\text{IN}]$  of  
 1507 Figure 3, and

1508 3.  $R \leq G \vdash r$  for all  $r \ll R \gg \in N''$ .

1509 By Lemma B.1(1)  $G \xrightarrow{pq\lambda_h} G_h$  and  $G_h \vdash p = P'_h$  and  $G_h \vdash q = Q'_h$ . By Lemma B.1(2)  
 1510  $G_h \vdash r = G \vdash r$  for all  $r \notin \{p, q\}$ . We can then choose  $G' = G_h$ .

1511 **Theorem 6.11 (Session Fidelity)** *If  $\vdash N : G$  and  $G \xrightarrow{\alpha} G'$ , then  $N \xrightarrow{\alpha} N'$  and  $\vdash N' : G'$ .*

1512 **Proof** Let  $\alpha = pq\lambda$ . By Lemma B.1(2)  $G \vdash p = \bigoplus_{i \in I} p! \lambda_i; P_i$  and  $G \vdash q = \sum_{i \in I} p? \lambda_i; Q_i$   
 1513 and  $\lambda = \lambda_i$  for some  $i \in I$  and  $G' \vdash r = G \vdash r$  for all  $r \notin \{p, q\}$ . By Lemma B.1(1)  $G' \vdash p =$   
 1514  $P_i$  and  $G' \vdash q = Q_i$ . From Lemma B.3 and Lemma B.2 we get  $N \equiv p \ll P \gg \parallel q \ll Q \gg \parallel N''$   
 1515 and

1516 1.  $P = \bigoplus_{i \in I} q! \lambda_i; P'_i$  with  $P'_i \leq P_i$  for  $i \in I$ , from Rule  $[\leq -\text{OUT}]$  of Figure 3, and

1517 2.  $Q = \sum_{j \in J} p? \lambda_j; Q'_j$  with  $Q'_j \leq Q_j$  for  $j \in I \subseteq J$ , from Rule  $[\leq -\text{IN}]$  of Figure 3, and

1518 3.  $R \leq G \vdash r$  for all  $r \ll R \gg \in N''$ .

1519 We can then choose  $N' = p \ll P'_i \gg \parallel q \ll Q'_i \gg \parallel N''$ .

## 1520 C. Proofs of Section 7

1521 **Lemma 7.4** *Let  $\sigma$  be a pointed trace. If  $\sigma \sim \sigma'$ , then  $\sigma'$  is a pointed trace and  $\text{last}(\sigma) =$   
 1522  $\text{last}(\sigma')$ .*

**Proof** Let  $\sigma \sim \sigma'$ . By Definition 7.1  $\sigma'$  is obtained from  $\sigma$  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  $\sigma_0$  obtained from  $\sigma$  by  $m - 1$  swaps of adjacent communi-  
 cations and there are  $\sigma_1, \sigma_2, \alpha$  and  $\alpha'$  such that

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

By induction hypothesis  $\sigma_0$  is a pointed trace and  $\text{last}(\sigma) = \text{last}(\sigma_0)$ . Therefore  
 $\sigma_2 \neq \epsilon$  since otherwise  $\alpha'$  would be the last communication of  $\sigma_0$  and it cannot be  
 $\text{part}(\alpha) \cap \text{part}(\alpha') = \emptyset$ . This implies  $\text{last}(\sigma) = \text{last}(\sigma')$ .

To show that  $\sigma'$  is pointed, since all the communications in  $\sigma_1$  and  $\sigma_2$  have the same  
 successors in  $\sigma_0$  and  $\sigma'$ , all we have to prove is that the required property holds for  
 the two swapped communications  $\alpha'$  and  $\alpha$  in  $\sigma'$ , namely:

$$\begin{aligned} \text{part}(\alpha') \cap (\text{part}(\alpha) \cup \text{part}(\sigma_2)) &\neq \emptyset \\ \text{part}(\alpha) \cap \text{part}(\sigma_2) &\neq \emptyset \end{aligned}$$

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

$$\begin{aligned} \text{part}(\alpha') \cap \text{part}(\sigma_2) &\neq \emptyset \\ \text{part}(\alpha) \cap (\text{part}(\alpha') \cup \text{part}(\sigma_2)) &\neq \emptyset \end{aligned}$$

1523 The last two statements are known to hold since  $\sigma_0$  is pointed by induction hypoth-  
 1524 esis.

## 1525 D. Proofs of Subsection 8.1

1526 This section contains the proofs of Lemmas 8.2, 8.4, 8.5 and 8.6.

### 1527 Lemma 8.2 (Properties of retrieval and residual for n-events).

- 1528 1. If  $\alpha \blacklozenge v$  is defined, then  $\alpha \blacklozenge (\alpha \blacklozenge v) = v$ ;
- 1529 2.  $\alpha \blacklozenge (\alpha \blacklozenge v) = v$ ;
- 1530 3. If  $v < v'$ , then  $\alpha \blacklozenge v < \alpha \blacklozenge v'$ ;
- 1531 4. If  $v < v'$  and both  $\alpha \blacklozenge v$  and  $\alpha \blacklozenge v'$  are defined, then  $\alpha \blacklozenge v < \alpha \blacklozenge v'$ ;
- 1532 5. If  $v \# v'$ , then  $\alpha \blacklozenge v \# \alpha \blacklozenge v'$ ;
- 1533 6. If  $v \# v'$  and both  $\alpha \blacklozenge v$  and  $\alpha \blacklozenge v'$  are defined, then  $\alpha \blacklozenge v \# \alpha \blacklozenge v'$ ;
- 1534 7. If  $\alpha \blacklozenge v \# \alpha \blacklozenge v'$ , then  $v \# v'$ .

1535 **Proof** For (1) and (2) it is enough to show the corresponding properties for located  
1536 events.

1537 (1) Since  $\alpha \blacklozenge (p :: \eta)$  is defined, we have  $\eta = (\alpha @ p) \cdot \eta'$  and  $\alpha \blacklozenge (p :: \eta) = p :: \eta'$  for  
1538 some  $\eta'$ . Then  $\alpha \blacklozenge (\alpha \blacklozenge (p :: \eta)) = \alpha \blacklozenge (p :: \eta') = p :: (\alpha @ p) \cdot \eta' = p :: \eta$ .

1539 (2) Since  $\alpha \blacklozenge (p :: \eta) = p :: (\alpha @ p) \cdot \eta$  is always defined, we immediately get  
1540  $\alpha \blacklozenge (\alpha \blacklozenge (p :: \eta)) = \alpha \blacklozenge (p :: (\alpha @ p) \cdot \eta) = p :: \eta$ .

1541 (3) Let  $v < v'$ . By Definition 5.7(1), there are  $p :: \eta \in v$  and  $p :: \eta' \in v'$  such that  
1542  $\eta < \eta'$ . Then  $\alpha \blacklozenge (p :: \eta) = p :: (\alpha @ p) \cdot \eta \in \alpha \blacklozenge v$  and  $\alpha \blacklozenge (p :: \eta') = p :: (\alpha @ p) \cdot \eta' \in$   
1543  $\alpha \blacklozenge v'$ . Since  $\eta < \eta'$  implies  $(\alpha @ p) \cdot \eta < (\alpha @ p) \cdot \eta'$ , we conclude that  $\alpha \blacklozenge v < \alpha \blacklozenge v'$ .

1544 (4) As in the previous case, there are  $p :: \eta \in v$  and  $p :: \eta' \in v'$  such that  $\eta < \eta'$ .  
1545 Since both  $\alpha \blacklozenge v$  and  $\alpha \blacklozenge v'$  are defined, there exist  $\eta_0$  and  $\eta'_0$  such that  $\eta = (\alpha @ p) \cdot \eta_0$   
1546 and  $\eta' = (\alpha @ p) \cdot \eta'_0$  and  $\alpha \blacklozenge (p :: \eta) = p :: \eta_0$  and  $\alpha \blacklozenge (p :: \eta') = p :: \eta'_0$ . Since  $\eta < \eta'$   
1547 implies  $\eta_0 < \eta'_0$ , we conclude that  $\alpha \blacklozenge v < \alpha \blacklozenge v'$ .

1548 (5) Let  $v \# v'$ . If Clause (2a) of Definition 5.7 applies, then there are  $p :: \eta \in v$   
1549 and  $p :: \eta' \in v'$  such that  $\eta \# \eta'$ . From  $\alpha \blacklozenge (p :: \eta) = p :: (\alpha @ p) \cdot \eta$  and  $\alpha \blacklozenge (p :: \eta') =$   
1550  $p :: (\alpha @ p) \cdot \eta'$  we get  $(\alpha @ p) \cdot \eta \# (\alpha @ p) \cdot \eta'$ . If Clause (2b) of Definition 5.7 applies,  
1551 then there are  $p :: \eta \in v$  and  $q :: \eta' \in v'$  with  $p \neq q$  such that  $|\eta \upharpoonright q| = |\eta' \upharpoonright p|$   
1552 and  $\neg(\eta \upharpoonright q \bowtie \eta' \upharpoonright p)$ . Let  $\eta_0 = (\alpha @ p) \cdot \eta$  and  $\eta'_0 = (\alpha @ q) \cdot \eta'$ . If  $\text{part}(\alpha) \neq \{p, q\}$ ,  
1553 then  $(\alpha @ p) \upharpoonright q = \epsilon = (\alpha @ q) \upharpoonright p$  and thus  $\eta_0 \upharpoonright q = \eta \upharpoonright q$  and  $\eta'_0 \upharpoonright p = \eta' \upharpoonright p$ . If  
1554  $\text{part}(\alpha) = \{p, q\}$ , say  $\alpha = pq\lambda$ , then  $\eta_0 = q!\lambda \cdot \eta$  and  $\eta'_0 = p?\lambda \cdot \eta'$ , which implies  
1555  $|\eta_0 \upharpoonright q| = |\eta \upharpoonright q| + 1 = |\eta' \upharpoonright p| + 1 = |\eta'_0 \upharpoonright p|$  and  $\neg(\eta_0 \upharpoonright q \bowtie \eta'_0 \upharpoonright p)$ . In both cases we  
1556 conclude that  $\alpha \blacklozenge v \# \alpha \blacklozenge v'$ .

1557 (6) The proof is similar to that of Point (5), considering that  $\alpha \blacklozenge v$  and  $\alpha \blacklozenge v'$  are  
1558 defined.

1559 (7) Let  $\alpha \blacklozenge v \# \alpha \blacklozenge v'$ . If Clause (2a) of Definition 5.7 applies, then there are  $p :: \eta \in v$   
1560 and  $p :: \eta' \in v'$  such that  $(\alpha @ p) \cdot \eta \# (\alpha @ p) \cdot \eta'$ . Therefore  $\eta \# \eta'$  and thus  $v \# v'$ . If  
1561 Clause (2b) of Definition 5.7 applies, then there are  $p :: \eta_0 = \alpha \blacklozenge (p :: \eta) \in \alpha \blacklozenge v$   
1562 and  $q :: \eta'_0 = \alpha \blacklozenge (q :: \eta') \in \alpha \blacklozenge v'$  with  $p \neq q$  such that  $|\eta_0 \upharpoonright q| = |\eta'_0 \upharpoonright p|$  and  
1563  $\neg(\eta_0 \upharpoonright q \bowtie \eta'_0 \upharpoonright p)$ . It follows that  $\eta_0 = (\alpha @ p) \cdot \eta$  and  $\eta'_0 = (\alpha @ q) \cdot \eta'$  and  $p :: \eta \in v$

1564 and  $q :: \eta' \in v'$ . If  $\text{part}(\alpha) \neq \{p, q\}$ , then  $(\alpha @ p) \upharpoonright q = \epsilon = (\alpha @ q) \upharpoonright p$  and thus  
 1565  $\eta \upharpoonright q = \eta_0 \upharpoonright q$  and  $\eta' \upharpoonright p = \eta'_0 \upharpoonright p$ . If  $\text{part}(\alpha) = \{p, q\}$ , say  $\alpha = pq\lambda$ , then  $\eta_0 = q!\lambda \cdot \eta$   
 1566 and  $\eta'_0 = p?\lambda \cdot \eta'$ , and thus  $|\eta \upharpoonright q| = |\eta_0 \upharpoonright q| - 1 = |\eta'_0 \upharpoonright p| - 1 = |\eta' \upharpoonright p|$  and  
 1567  $\neg(\eta \upharpoonright q \bowtie \eta' \upharpoonright p)$ . In both cases we conclude that  $v \# v'$ .

1568 **Lemma 8.4 (Properties of  $\text{nec}(\cdot)$ )**

- 1569 1. Let  $\text{nec}(\sigma) = v_1; \dots; v_n$ . Then
- 1570 (a)  $\text{cm}(v_i) = \sigma[i]$  for all  $i, 1 \leq i \leq n$ ;
- 1571 (b) If  $1 \leq h, k \leq n$ , then  $\neg(v_h \# v_k)$ .
- 1572 2.  $\neg(\text{nec}(\alpha) \# \alpha \diamond v)$  for all  $v$ .
- 1573 3. Let  $\sigma = \alpha \cdot \sigma'$  and  $\sigma' \neq \epsilon$ . If  $\text{nec}(\sigma) = v_1; \dots; v_n$  and  $\text{nec}(\sigma') = v'_2; \dots; v'_n$ , then  
 1574  $\alpha \diamond v'_i = v_i$  and  $\alpha \blacklozenge v_i = v'_i$  for all  $i, 2 \leq i \leq n$ .

1575 **Proof** (1a) Immediate from Definition 8.3, since  $\text{cm}(\sigma \diamond v) = \text{cm}(v)$  for any event  $v$ .

1576 (1b) We show that neither Clause (2a) nor Clause (2b) of Definition 5.7 can be  
 1577 used to derive  $v_h \# v_k$ . Notice that  $v_i = \{p_i :: \sigma[1 \dots i] @ p_i, q_i :: \sigma[1 \dots i] @ q_i\}$ . So if  
 1578  $p :: \eta \in v_h$  and  $p :: \eta' \in v_k$  with  $h < k$ , then either  $\eta < \eta'$  or  $\eta = \eta'$ . Therefore Clause  
 1579 (2a) does not apply. If  $p :: \eta \in v_h$  and  $q :: \eta' \in v_k$  and  $p \neq q$  and  $|\eta \upharpoonright q| = |\eta' \upharpoonright p|$ , then  
 1580 it must be  $\eta \upharpoonright q = (\sigma[1 \dots h] @ p) \upharpoonright q \bowtie (\sigma[1 \dots k] @ q) \upharpoonright p = \eta' \upharpoonright p$ . Therefore Clause  
 1581 (2b) cannot be used.

1582 (2) We show that neither Clause (2a) nor Clause (2b) of Definition 5.7 can be used  
 1583 to derive  $\text{nec}(\alpha) \# \alpha \diamond v$ . Let  $\text{part}(\alpha) = \{p, q\}$ . Then  $\text{nec}(\alpha) = \{p :: \alpha @ p, q :: \alpha @ q\}$ .  
 1584 Note that  $p :: \eta \in \alpha \diamond v$  iff  $\eta = (\alpha @ p) \cdot \eta'$  and  $p :: \eta' \in v$ . Since  $\alpha @ p < (\alpha @ p) \cdot \eta'$ ,  
 1585 Clause (2a) of Definition 5.7 cannot be used. Now suppose  $r :: \eta \in \alpha \diamond v$  for some  
 1586  $r \notin \{p, q\}$ . In this case  $(\alpha @ p) \upharpoonright r = (\alpha @ q) \upharpoonright r = \epsilon$ . Therefore, since  $\epsilon \bowtie \epsilon$ , Clause (2b)  
 1587 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

$$\begin{aligned} v_i &= \sigma[1 \dots i - 1] \diamond \text{nec}(\sigma[i]) = \alpha \diamond (\sigma[2 \dots i - 1] \diamond \text{nec}(\sigma[i])) = \\ &\alpha \diamond (\sigma'[1 \dots i - 2] \diamond \text{nec}(\sigma'[i - 1])) = \alpha \diamond v'_i \end{aligned}$$

1588 for all  $i, 2 \leq i \leq n$ .

1589 By Lemma 8.2(2)  $\alpha \diamond v'_i = v_i$  implies  $\alpha \blacklozenge v_i = v'_i$  for all  $i, 2 \leq i \leq n$ .

1590 **Lemma 8.5** If  $N \xrightarrow{\alpha} N'$  and  $v \in \mathcal{NE}(N)$ , then  $v = \text{nec}(\alpha)$  or  $v \# \text{nec}(\alpha)$  or  $\alpha \blacklozenge v$  is defined.

1591 **Proof** Let  $\text{nec}(\alpha) = \{p :: \alpha @ p, q :: \alpha @ q\}$  and  $v = \{r :: \eta, s :: \eta'\}$ . By Definition 8.1(3)  
 1592  $\alpha \blacklozenge v$  is defined iff  $\eta = (\alpha @ r) \cdot \eta_0$  and  $\eta' = (\alpha @ s) \cdot \eta'_0$  for some  $\eta_0, \eta'_0$ .

1593 There are 2 possibilities:

- 1594 •  $\{r, s\} \cap \{p, q\} = \emptyset$ . Then  $\alpha @ r = \alpha @ s = \epsilon$  and  $\alpha \blacklozenge v = v$ ;
- 1595 •  $\{r, s\} \cap \{p, q\} \neq \emptyset$ . Suppose  $r = p$ . There are three possible subcases:
- 1596 1.  $\eta = \pi \cdot \zeta$  with  $\pi \neq \alpha @ p$ . Then  $r :: \eta \# p :: \alpha @ p$  and thus  $v \# \text{nec}(\alpha)$ ;

- 1597 2.  $\eta = \alpha @ p$ . Then either  $\eta' = \alpha @ q$  and  $v = \text{nec}(\alpha)$ , or  $\eta' \neq \alpha @ q$  and  
 1598  $v \# \text{nec}(\alpha)$  by Lemma 5.21;
- 1599 3.  $\eta = (\alpha @ p) \cdot \eta_0$ . Then  $\alpha \blacklozenge p :: \eta = p :: \eta_0$ . Now, if  $s \neq q$  we have  $\alpha \blacklozenge s :: \eta' =$   
 1600  $s :: \eta'$ , and thus  $\alpha \blacklozenge v = \{p :: \eta_0, s :: \eta'\}$ . Otherwise,  $v = \{p :: (\alpha @ p) \cdot \eta_0, q ::$   
 1601  $\eta'\}$ . By Definition 5.5  $p :: (\alpha @ p) \cdot \eta_0 \bowtie q :: \eta'$ , which implies  $\eta' =$   
 1602  $(\alpha @ q) \cdot \eta'_0$  for some  $\eta'_0$ .

1603 **Lemma 8.6** *Let  $N \xrightarrow{\alpha} N'$ . Then*

- 1604 1.  $\{\text{nec}(\alpha)\} \cup \{\alpha \blacklozenge v \mid v \in \mathcal{NE}(N')\} \subseteq \mathcal{NE}(N)$ ;
- 1605 2.  $\{\alpha \blacklozenge v \mid v \in \mathcal{NE}(N) \text{ and } \alpha \blacklozenge v \text{ defined}\} \subseteq \mathcal{NE}(N')$ .

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

$$N = p \llbracket \bigoplus_{i \in I} q! \lambda_i; P \rrbracket \parallel q \llbracket \sum_{j \in J} p? \lambda_j; Q_j \rrbracket \parallel N_0$$

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

$$N' = p \llbracket P_k \rrbracket \parallel q \llbracket Q_k \rrbracket \parallel N_0$$

1606 (1) Let  $\mathcal{RT} = \{\text{nec}(\alpha)\} \cup \{\alpha \blacklozenge v \mid v \in \mathcal{NE}(N')\}$ . We first show that  $\mathcal{RT} \subseteq \mathcal{CE}(N)$ .  
 1607 By Definition 5.13(1)  $\text{nec}(\alpha) \in \mathcal{CE}(N)$ . Let  $v = \{r :: \eta, s :: \eta'\} \in \mathcal{NE}(N')$ . We want to  
 1608 prove that  $\alpha \blacklozenge v \in \mathcal{CE}(N)$ . By Definition 5.13(1) there are  $R, S$  such that  $r \llbracket R \rrbracket \in N'$   
 1609 and  $s \llbracket S \rrbracket \in N'$  and  $\eta \in \mathcal{PE}(R)$  and  $\eta' \in \mathcal{PE}(S)$ . There are two possible cases:

- 1610 •  $\{r, s\} \cap \{p, q\} = \emptyset$ . Then  $r \llbracket R \rrbracket \in N$  and  $s \llbracket S \rrbracket \in N$  and thus  $\alpha \blacklozenge v = v \in \mathcal{CE}(N)$ ;
- 1611 •  $\{r, s\} \cap \{p, q\} \neq \emptyset$ . Suppose  $r = p$ . Then  $\eta \in \mathcal{PE}(P_k)$  and  $p :: q! \lambda_k \cdot \eta \in \alpha \blacklozenge v$  and  
 1612  $q! \lambda_k \cdot \eta \in \mathcal{PE}(\bigoplus_{i \in I} q! \lambda_i; P_i)$ . There are two subcases:
- 1613 –  $s = q$ . Then  $\eta' \in \mathcal{PE}(Q_k)$  and  $q :: p? \lambda_k \cdot \eta' \in \alpha \blacklozenge v$  and  $q! \lambda_k \cdot \eta' \in$   
 1614  $\mathcal{PE}(\sum_{j \in J} p? \lambda_j; Q_j)$ . We have  $\alpha \blacklozenge v = \{p :: q! \lambda_k \cdot \eta, q :: p? \lambda_k \cdot \eta'\} \in \mathcal{CE}(N)$ ;
- 1615 –  $s \neq q$ . Then  $\alpha \blacklozenge s :: \eta' = s :: \eta'$ , and thus  $\alpha \blacklozenge v = \{p :: q! \lambda_k \cdot \eta, s :: \eta'\} \in$   
 1616  $\mathcal{CE}(N)$ .

1617 Therefore in all cases  $\mathcal{RT} \subseteq \mathcal{CE}(N)$ . We want now to show that  $\mathcal{RT} \subseteq \mathcal{NE}(N)$ .

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

$$f_{\mathcal{CE}(N)}(X) = \{v_0 \in \mathcal{CE}(N) \mid \exists E_0 \subseteq X. E_0 \text{ is a causal set of } v_0 \text{ in } X\}$$

1618 Then  $\mathcal{NE}(N)$  is also the greatest post-fixed point of  $f_{\mathcal{CE}(N)}(X)$ , namely the greatest  $X$   
 1619 such that  $X \subseteq f_{\mathcal{CE}(N)}(X)$ . Therefore, to show that  $\mathcal{RT} \subseteq \mathcal{NE}(N)$ , it is enough to show  
 1620 that  $\mathcal{RT}$  is also a post-fixed point of  $f_{\mathcal{CE}(N)}(X)$ , namely that  $\mathcal{RT} \subseteq f_{\mathcal{CE}(N)}(\mathcal{RT})$ .

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

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

1621 where  $\Xi = \{\alpha \diamond v' \mid v' \in E \text{ and } E \text{ is a causal set of } v \text{ in } \mathcal{NE}(N')\}$ .

1622 We show that  $\text{pre}(\alpha, E, v)$  is a causal set of  $\alpha \diamond v$  in  $\mathcal{RT}$ , namely that it is a minimal  
1623 subset of  $\mathcal{RT}$  satisfying Conditions (1) and (2) of Definition 5.9.

1624 *Condition (1)* If  $\text{nec}(\alpha) \in \text{pre}(\alpha, E, v)$ , then  $\{r, s\} \cap \{p, q\} \neq \emptyset$ . A conflict between  $\text{nec}(\alpha)$   
1625 and any other event of  $\text{pre}(\alpha, E, v) \cup \{\alpha \diamond v\}$  can only be derived by Clause (2a) of  
1626 Definition 5.7, since  $\text{nec}(\alpha) = \{p :: q! \lambda, q :: p? \lambda\}$  and  $(\alpha @ p) \upharpoonright t = (\alpha @ q) \upharpoonright t = \epsilon$  for  
1627 all  $t \notin \{p, q\}$ . Suppose  $r = p$ . Then  $p :: q! \lambda \cdot \eta \in \alpha \diamond v$ . Since  $q! \lambda < q! \lambda \cdot \eta$ , Clause (2a)  
1628 cannot be used to derive a conflict  $\text{nec}(\alpha) \# \alpha \diamond v$ . Similarly, if  $\alpha \diamond v_1 \in \text{pre}(\alpha, E, v)$   
1629 and  $p :: \eta_1 \in v_1$ , then  $p :: q! \lambda \cdot \eta_1 \in \alpha \diamond v_1$ . Then  $q! \lambda < q! \lambda \cdot \eta_1$ , hence Clause (2a)  
1630 cannot be used to derive  $\text{nec}(\alpha) \# \alpha \diamond v_1$ .

1631 Suppose now  $\alpha \diamond v_1 \in \text{pre}(\alpha, E, v)$  and  $\alpha \diamond v_2 \in \text{pre}(\alpha, E, v)$ . Since  $E$  is a causal set, we  
1632 have  $\neg(v_1 \# v_2)$ . Thus  $\neg(\alpha \diamond v_1 \# \alpha \diamond v_2)$  by Lemma 8.2(7).

1633 *Condition (2)* Let  $v = \{r :: \eta, s :: \eta'\}$ , we have  $\alpha \diamond v = \{r :: (\alpha @ r) \cdot \eta, s :: (\alpha @ s) \cdot \eta'\}$ .  
1634 We show that if  $\eta_0 < (\alpha @ r) \cdot \eta$ , then  $r :: \eta_0 \in v_0$  for some  $v_0 \in \text{pre}(\alpha, E, v)$ . From  
1635  $\eta_0 < (\alpha @ r) \cdot \eta$  we derive  $\eta_0 = (\alpha @ r) \cdot \zeta$  for some  $\zeta$  such that  $\zeta < \eta$ . If  $\zeta \neq \epsilon$ ,  
1636 then  $\zeta = \eta'_0 < \eta$ . Since  $E$  is a causal set,  $\eta'_0 < \eta_0$  implies  $r :: \eta'_0 \in E$ . Hence  
1637  $r :: \eta_0 \in \text{pre}(\alpha, E, v)$ . If instead  $\zeta = \epsilon$ , then it must be  $\eta_0 = \alpha @ r \neq \epsilon$  and thus  $r \in \{p, q\}$ .  
1638 In this case  $\{\text{nec}(\alpha)\} \in \text{pre}(\alpha, E, v)$  and thus  $r :: \eta_0 \in \text{pre}(\alpha, E, v)$ .

1639 As for *minimality*, we first show that  $v' < \alpha \diamond v$  for all  $v' \in \text{pre}(\alpha, E, v)$ . If  $\text{nec}(\alpha) \in$   
1640  $\text{pre}(\alpha, E, v)$ , then  $\{r, s\} \cap \{p, q\} \neq \emptyset$ . Then  $\text{nec}(\alpha) < \alpha \diamond v$ . If  $v_1 \in \text{pre}(\alpha, E, v)$  and  
1641  $v_1 \neq \text{nec}(\alpha)$ , then there exists  $v'_1 \in E$  such that  $v_1 = \alpha \diamond v'_1$ . Since  $E$  is a causal set for  
1642  $v$ , we have  $v'_1 < v$ . Therefore  $v_1 = \alpha \diamond v'_1 < \alpha \diamond v$  by Lemma 8.2(3). Assume now that  
1643  $\text{pre}(\alpha, E, v)$  is not minimal. Then there is  $E' \subset \text{pre}(\alpha, E, v)$  that verifies Condition (2)  
1644 of Definition 5.9 for  $\alpha \diamond v$ . Let  $v' \in \text{pre}(\alpha, E, v) \setminus E'$ . Then  $v' < \alpha \diamond v = \{r :: \eta_r, s :: \eta_s\}$ .  
1645 Assume that  $r :: \eta'_r \in v'$  with  $\eta'_r < \eta_r$  (the proof is similar for  $s$ ). By Condition (2),  
1646 there is  $v'' \in E'$  such that  $r :: \eta'_r \in v''$ . But then  $v' \# v''$  by Lemma 5.21, contradicting  
1647 the fact that  $\text{pre}(\alpha, E, v)$  verifies Condition (1). Therefore  $\text{pre}(\alpha, E, v)$  is minimal.

1648 (2) Let  $\mathcal{RS} = \{\alpha \diamond v \mid v \in \mathcal{NE}(N) \text{ and } \alpha \diamond v \text{ defined}\}$ . We first show that  $\mathcal{RS} \subseteq$   
1649  $\mathcal{CE}(N')$ . Let  $v = \{r :: \eta, s :: \eta'\} \in \mathcal{NE}(N)$  be such that  $\alpha \diamond v$  is defined. We want to  
1650 prove that  $\alpha \diamond v \in \mathcal{CE}(N')$ . By Definition 5.13(1) there are  $R, S$  such that  $r \ll R \ll N$   
1651 and  $s \ll S \ll N$  and  $\eta \in \mathcal{PE}(R)$  and  $\eta' \in \mathcal{PE}(S)$ . There are two possible cases:

- 1652 •  $\{r, s\} \cap \{p, q\} = \emptyset$ . Then  $r \ll R \ll N'$  and  $s \ll S \ll N'$  and thus  $\alpha \diamond v = v \in \mathcal{CE}(N')$ ;
- 1653 •  $\{r, s\} \cap \{p, q\} \neq \emptyset$ . Suppose  $r = p$ . Then  $\eta \in \mathcal{PE}(\bigoplus_{i \in I} q! \lambda_i; P_i)$  and since  $\alpha \diamond v$  is  
1654 defined we have that  $\eta = q! \lambda_k \cdot \eta_k$  where  $\eta_k \in \mathcal{PE}(P_k)$ . There are two subcases:
  - 1655 –  $s = q$ . Then  $\eta' \in \mathcal{PE}(\sum_{j \in J} p? \lambda_j; Q_j)$  and since  $\alpha \diamond v$  is defined  $\eta' = p? \lambda_k \cdot \eta'_k$   
1656 where  $\eta'_k \in \mathcal{PE}(Q_k)$ . In this case we have  $\alpha \diamond v = \{p :: \eta_k, q :: \eta'_k\} \in \mathcal{CE}(N')$ ;
  - 1657 –  $s \neq q$ . Then  $\alpha \diamond s :: \eta' = s :: \eta'$ , and thus  $\alpha \diamond v = \{p :: \eta_k, s :: \eta'\} \in \mathcal{CE}(N')$ .

1658 Therefore in all cases  $\mathcal{RS} \subseteq \mathcal{CE}(N')$ . We want now to show that  $\mathcal{RS} \subseteq \mathcal{NE}(N')$ .

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

$$f_{\mathcal{CE}(N')}(X) = \{v_0 \in \mathcal{CE}(N') \mid \exists E_0 \subseteq X. E_0 \text{ is a causal set of } v_0 \text{ in } X\}$$

1659 Then, in order to obtain  $\mathcal{RS} \subseteq \mathcal{NE}(N')$  it is enough to show that  $\mathcal{RS}$  is a post-fixed  
1660 point of  $f_{\mathcal{CE}(N')}(X)$ , namely that  $\mathcal{RS} \subseteq f_{\mathcal{CE}(N')}(X)$ .

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

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

1661 We show that  $\text{post}(\alpha, E, v)$  is a causal set of  $\alpha \blacklozenge v$  in  $\mathcal{RS}$ , namely that it is a minimal  
1662 subset of  $\mathcal{RS}$  satisfying Conditions (1) and (2) of Definition 5.9.

1663 *Condition (1)* Suppose  $\alpha \blacklozenge v_1 \in \text{post}(\alpha, E, v)$  and  $\alpha \blacklozenge v_2 \in \text{post}(\alpha, E, v)$ . Since  $E$  is a  
1664 causal set and  $v_1, v_2 \in E$ , we have  $\neg(v_1 \# v_2)$ . Thus  $\neg(\alpha \blacklozenge v_1 \# \alpha \blacklozenge v_2)$  by Lemma 8.2(5)  
1665 and (1).

1666 *Condition (2)* Since  $v = \{r :: \eta, s :: \eta'\}$  and  $\alpha \blacklozenge v$  is defined, we have  $\eta = (\alpha @ r) \cdot \eta_r$  and  
1667  $\eta' = (\alpha @ s) \cdot \eta_s$  and  $\alpha \blacklozenge v = \{r :: \eta_r, s :: \eta_s\}$ . Let  $\eta_0 < \eta_r$ . Then  $(\alpha @ r) \cdot \eta_0 < (\alpha @ r) \cdot \eta_r =$   
1668  $\eta$ . Since  $E$  is a causal set for  $v$  in  $\mathcal{NE}(\mathcal{N})$ , this implies  $r :: (\alpha @ r) \cdot \eta_0 \in E$ . Hence  
1669  $r :: \eta_0 \in \text{post}(\alpha, E, v)$ .

1670 As for *minimality*, we first show that  $v' < \alpha \blacklozenge v$  for all  $v' \in \text{post}(\alpha, E, v)$ . If  $v_1 \in$   
1671  $\text{post}(\alpha, E, v)$ , then there exists  $v'_1 \in E$  such that  $v_1 = \alpha \blacklozenge v'_1$ . Since  $E$  is a causal set for  
1672  $v$ , we have  $v'_1 < v$ . Therefore  $v_1 = \alpha \blacklozenge v'_1 < \alpha \blacklozenge v$  by Lemma 8.2(3). Assume now that  
1673  $\text{post}(\alpha, E, v)$  is not minimal. Then there is  $E' \subset \text{post}(\alpha, E, v)$  that verifies Condition (2)  
1674 of Definition 5.9 for  $\alpha \blacklozenge v$ . Let  $v' \in \text{post}(\alpha, E, v) \setminus E'$ . Then  $v' < \alpha \blacklozenge v = \{r :: \eta_r, s :: \eta_s\}$ .  
1675 Assume that  $r :: \eta'_r \in v'$  with  $\eta'_r < \eta_r$  (the proof is similar for  $s$ ). By Condition (2),  
1676 there is  $v'' \in E'$  such that  $r :: \eta'_r \in v''$ . But then  $v' \# v''$  by Lemma 5.21, contradicting  
1677 the fact that  $\text{post}(\alpha, E, v)$  verifies Condition (1). Therefore  $\text{post}(\alpha, E, v)$  is minimal.

## 1678 E. Proofs of Subsection 8.2

1679 This section contains the proofs of Lemmas 8.10, 8.11 and 8.12.

### 1680 Lemma 8.10 (Properties of retrieval and residual for g-events).

- 1681 1. If  $\alpha \bullet \gamma$  is defined, then  $\alpha \circ (\alpha \bullet \gamma) = \gamma$ ;
- 1682 2.  $\alpha \bullet (\alpha \circ \gamma) = \gamma$ ;
- 1683 3. If  $\gamma_1 < \gamma_2$ , then  $\alpha \circ \gamma_1 < \alpha \circ \gamma_2$ ;
- 1684 4. If  $\gamma_1 < \gamma_2$  and both  $\alpha \bullet \gamma_1$  and  $\alpha \bullet \gamma_2$  are defined, then  $\alpha \bullet \gamma_1 < \alpha \bullet \gamma_2$ ;
- 1685 5. If  $\gamma_1 \# \gamma_2$ , then  $\alpha \circ \gamma_1 \# \alpha \circ \gamma_2$ ;
- 1686 6. If  $\gamma < \alpha \circ \gamma'$ , then either  $\gamma = [\alpha]_{\sim}$  or  $\alpha \bullet \gamma < \gamma'$ ;
- 1687 7. If  $\text{part}(\alpha_1) \cap \text{part}(\alpha_2) = \emptyset$ , then  $\alpha_1 \circ (\alpha_2 \circ \gamma) = \alpha_2 \circ (\alpha_1 \circ \gamma)$ ;
- 1688 8. If  $\text{part}(\alpha_1) \cap \text{part}(\alpha_2) = \emptyset$  and both  $\alpha_2 \bullet (\alpha_1 \circ \gamma)$ ,  $\alpha_2 \bullet \gamma$  are defined, then  $\alpha_1 \circ (\alpha_2 \bullet \gamma) =$   
1689  $\alpha_2 \bullet (\alpha_1 \circ \gamma)$ .

1690 **Proof** (1) If  $\alpha \bullet [\sigma]_{\sim}$  is defined, then in case  $\text{part}(\alpha) \cap \text{part}(\sigma) = \emptyset$  we get  $\alpha \bullet [\sigma]_{\sim} = [\sigma]_{\sim}$   
1691 and also  $\alpha \circ [\sigma]_{\sim} = [\sigma]_{\sim}$ , so  $\alpha \circ (\alpha \bullet [\sigma]_{\sim}) = [\sigma]_{\sim}$ . Instead if  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$ ,  
1692 then  $\alpha \bullet [\sigma]_{\sim} = [\sigma']_{\sim}$  where  $\sigma \sim \alpha \cdot \sigma'$  and  $\sigma' \neq \epsilon$ . From  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$  we get  
1693  $\alpha \circ [\sigma']_{\sim} = [\alpha \cdot \sigma']_{\sim}$  by Definition 7.6. This implies  $\alpha \circ (\alpha \bullet [\sigma]_{\sim}) = [\sigma]_{\sim}$ .

1694 (2) By Definition 7.6 either  $\alpha \circ [\sigma]_{\sim} = [\alpha \cdot \sigma]_{\sim}$  if  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$ , or  $\alpha \circ \sigma = [\sigma]_{\sim}$ .  
 1695 In the first case  $\alpha \bullet [\alpha \cdot \sigma]_{\sim} = [\sigma]_{\sim}$  and in the second  $\alpha \bullet [\sigma]_{\sim} = [\sigma]_{\sim}$ , which proves  
 1696 the result.

1697 (3) Let  $\gamma_1 = [\sigma]_{\sim}$  and  $\gamma_2 = [\sigma \cdot \sigma']_{\sim}$ . If  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$ , then  $\text{part}(\alpha) \cap$   
 1698  $\text{part}(\sigma \cdot \sigma') \neq \emptyset$ , and we have  $\alpha \circ \gamma_1 = [\alpha \cdot \sigma]_{\sim}$  and  $\alpha \circ \gamma_2 = [\alpha \cdot \sigma \cdot \sigma']_{\sim}$ . Whence  
 1699  $\alpha \circ \gamma_1 \leq \alpha \circ \gamma_2$ . Suppose now  $\text{part}(\alpha) \cap \text{part}(\sigma) = \emptyset$ . Then  $\alpha \circ \gamma_1 = [\sigma]_{\sim} = \gamma_1$ . If  
 1700 also  $\text{part}(\alpha) \cap \text{part}(\sigma') = \emptyset$ , then  $\alpha \circ \gamma_2 = [\sigma \cdot \sigma']_{\sim} = \gamma_2$  and we are done. If instead  
 1701  $\text{part}(\alpha) \cap \text{part}(\sigma') \neq \emptyset$ , then  $\alpha \circ \gamma_2 = [\alpha \cdot \sigma \cdot \sigma']_{\sim} = [\sigma \cdot \alpha \cdot \sigma']_{\sim}$ , whence  $\gamma_1 \leq \alpha \circ \gamma_2$ .

1702 (4) Let  $\gamma_1 = [\sigma]_{\sim}$  and  $\gamma_2 = [\sigma \cdot \sigma']_{\sim}$ . If  $\text{part}(\alpha) \cap \text{part}(\sigma) = \text{part}(\alpha) \cap \text{part}(\sigma \cdot \sigma') = \emptyset$ ,  
 1703 then  $\alpha \bullet \gamma_1 = \gamma_1$  and  $\alpha \bullet \gamma_2 = \gamma_2$ . If  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$ , then  $\sigma \sim \alpha \cdot \sigma_0$ , which implies  
 1704  $\alpha \bullet \gamma_1 = [\sigma_0]_{\sim}$  and  $\alpha \bullet \gamma_2 = [\sigma_0 \cdot \sigma']_{\sim}$ . If  $\text{part}(\alpha) \cap \text{part}(\sigma) = \emptyset$  and  $\text{part}(\alpha) \cap \text{part}(\sigma \cdot \sigma') \neq \emptyset$ ,  
 1705 then  $\alpha \bullet \gamma_1 = [\sigma]_{\sim}$  and  $\sigma' \sim \alpha \cdot \sigma_0$ , which implies  $\alpha \bullet \gamma_2 = [\sigma \cdot \sigma_0]_{\sim}$ .

1706 (5) Let  $\gamma_1 = [\sigma]_{\sim}$  and  $\gamma_2 = [\sigma']_{\sim}$  and  $\sigma @ p \# \sigma' @ p$  for some  $p$ . The only interesting  
 1707 case is  $\text{part}(\alpha) \cap \text{part}(\sigma) = \emptyset$  and  $\text{part}(\alpha) \cap \text{part}(\sigma') \neq \emptyset$ . This implies  $\alpha \circ \gamma_1 = [\sigma]_{\sim}$   
 1708 and  $\alpha \circ \gamma_2 = [\alpha \cdot \sigma']_{\sim}$ . We get  $(\alpha \cdot \sigma') @ p = \sigma' @ p$  since  $\text{part}(\alpha) \cap \text{part}(\sigma) = \emptyset$  implies  
 1709  $p \notin \text{part}(\alpha)$ . We conclude  $\alpha \circ \gamma_1 \# \alpha \circ \gamma_2$ .

1710 (6) The case  $\gamma = [\alpha]_{\sim}$  is immediate. If  $\alpha \bullet \gamma$  is defined, we get  $\alpha \bullet \gamma < \alpha \bullet (\alpha \circ \gamma')$   
 1711 by Point 4 and  $\alpha \bullet (\alpha \circ \gamma') = \gamma'$  by Point 2. Otherwise let  $\gamma = [\sigma]_{\sim}$  and  $\alpha \circ \gamma' =$   
 1712  $[\sigma \cdot \sigma']_{\sim}$ . From  $\alpha \bullet \gamma$  undefined we get  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$  and  $\sigma \not\sim \alpha \cdot \sigma_0$ . Since  
 1713  $\text{part}(\alpha) \cap \text{part}(\sigma) \neq \emptyset$  implies  $\text{part}(\alpha) \cap \text{part}(\sigma \cdot \sigma') \neq \emptyset$  we get  $\sigma \cdot \sigma' \sim \alpha \cdot \sigma_1$  for some  
 1714  $\sigma_1$  by Definition 7.6(1). Then this case is impossible.

1715 (7) Let  $\gamma = [\sigma]_{\sim}$ . By Definition 7.6(1) we have four cases:

1716 (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  $\text{part}(\alpha_1) \cap \text{part}(\sigma) \neq$   
 1717  $\emptyset$  and  $\text{part}(\alpha_2) \cap \text{part}(\sigma) \neq \emptyset$ , since  $\text{part}(\alpha_1) \cap \text{part}(\alpha_2) = \emptyset$ ;

1718 (b)  $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\alpha_1 \cdot \sigma]_{\sim} = \alpha_2 \circ (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) \neq \emptyset$  and  $\text{part}(\alpha_2) \cap$   
 1719  $\text{part}(\sigma) = \emptyset$ ;

1720 (c)  $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\alpha_2 \cdot \sigma]_{\sim} = \alpha_2 \circ (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) = \emptyset$  and  $\text{part}(\alpha_2) \cap$   
 1721  $\text{part}(\sigma) \neq \emptyset$ ;

1722 (d)  $\alpha_1 \circ (\alpha_2 \circ \sigma) = [\sigma]_{\sim} = \alpha_2 \circ (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) = \emptyset$  and  $\text{part}(\alpha_2) \cap \text{part}(\sigma) =$   
 1723  $\emptyset$ .

1724 (8) Let  $\gamma = [\sigma]_{\sim}$ . By Definitions 7.6(1) and 8.9(1) we have four cases:

1725 (a)  $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\alpha_1 \cdot \sigma']_{\sim} = \alpha_2 \bullet (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) \neq \emptyset$  and  $\sigma \sim \alpha_2 \cdot \sigma'$ ;

1726 (b)  $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\alpha_1 \cdot \sigma]_{\sim} = \alpha_2 \bullet (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) \neq \emptyset$  and  $\text{part}(\alpha_2) \cap$   
 1727  $\text{part}(\sigma) = \emptyset$ ;

1728 (c)  $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\sigma']_{\sim} = \alpha_2 \bullet (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) = \emptyset$  and  $\sigma \sim \alpha_2 \cdot \sigma'$ ;

1729 (d)  $\alpha_1 \circ (\alpha_2 \bullet \sigma) = [\sigma]_{\sim} = \alpha_2 \bullet (\alpha_1 \circ \sigma)$  if  $\text{part}(\alpha_1) \cap \text{part}(\sigma) = \emptyset$  and  $\text{part}(\alpha_2) \cap \text{part}(\sigma) =$   
 1730  $\emptyset$ .

1731 **Lemma 8.11** *The following hold:*  
 1732



- 1733 1. If  $\gamma \in \mathcal{G}(\mathbf{G})$ , then  $\text{pq}\lambda \circ \gamma \in \mathcal{G}(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$ , where  $\lambda = \lambda_k$  and  $\mathbf{G} = \mathbf{G}_k$   
 1734 for some  $k \in I$ ;
- 1735 2. If  $\gamma \in \mathcal{G}(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$  and  $\text{pq}\lambda_k \bullet \gamma$  is defined, then  $\text{pq}\lambda_k \bullet \gamma \in \mathcal{G}(\mathbf{G}_k)$ ,  
 1736 where  $k \in I$ .

1737 **Proof** (1) By Definition 7.11(1)  $\gamma \in \mathcal{G}(\mathbf{G})$  implies  $\gamma = \text{ev}(\sigma)$  for some  $\sigma \in \text{Tr}^+(\mathbf{G})$ .  
 1738 Since  $\text{pq}\lambda \circ \gamma = \text{ev}(\text{pq}\lambda \cdot \sigma)$  by Definitions 7.6, 7.7 and  $\text{pq}\lambda \cdot \sigma \in \text{Tr}^+(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$   
 1739 we conclude  $\text{pq}\lambda \circ \gamma \in \mathcal{G}(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$  by Definition 7.11(1).

1740 (2) By Definition 7.11(1)  $\gamma \in \mathcal{G}(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$  implies  $\gamma = \text{ev}(\sigma)$  for some  
 1741  $\sigma \in \text{Tr}^+(\mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i)$ . We get  $\sigma = \text{pq}\lambda_h \cdot \sigma'$  with  $\sigma' \in \text{Tr}^+(\mathbf{G}_h)$  or  $\sigma' = \epsilon$  for  
 1742 some  $h \in I$ . The hypothesis  $\text{pq}\lambda_k \bullet \gamma$  defined implies either  $h = k$  and  $\sigma' \neq \epsilon$  or  
 1743  $\text{part}(\sigma') \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$  and  $\text{pq}\lambda_k \bullet \gamma = \text{ev}(\sigma')$  by Definition 8.9(1). In the first case  
 1744  $\sigma' \in \text{Tr}^+(\mathbf{G}_k)$ . In the second case  $\sigma'' \in \text{Tr}^+(\mathbf{G}_k)$  for some  $\sigma'' \sim \sigma'$  by definition of  
 1745 projection, which prescribes the same behaviours to all participants different from  
 1746  $\mathbf{p}, \mathbf{q}$ , see Figure 2. We conclude  $\text{pq}\lambda_k \bullet \gamma \in \mathcal{G}(\mathbf{G}_k)$  by Definition 7.11(1).

1747 **Lemma 8.12** Let  $\mathbf{G} \xrightarrow{\alpha} \mathbf{G}'$ .

- 1748 1. If  $\gamma \in \mathcal{G}(\mathbf{G}')$ , then  $\alpha \circ \gamma \in \mathcal{G}(\mathbf{G})$ ;
- 1749 2. If  $\gamma \in \mathcal{G}(\mathbf{G})$  and  $\alpha \bullet \gamma$  is defined, then  $\alpha \bullet \gamma \in \mathcal{G}(\mathbf{G}')$ .

1750 **Proof** Both proofs are by induction on the inference of the transition  $\mathbf{G} \xrightarrow{\alpha} \mathbf{G}'$ , see  
 1751 Figure 4.

1752 (1) For rule [Ecomm] we get  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\mathbf{G}' = \mathbf{G}_k$  and  $\alpha = \text{pq}\lambda_k$   
 1753 for some  $k \in I$ . We conclude  $\alpha \circ \gamma \in \mathcal{G}(\mathbf{G})$  by Lemma 8.11(1).  
 1754 For rule [Icomm] we get  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\mathbf{G}' = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}'_i$  and  
 1755  $\mathbf{G}_i \xrightarrow{\alpha} \mathbf{G}'_i$  for all  $i \in I$  and  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . By Definition 7.11(1)  $\gamma \in \mathcal{G}(\mathbf{G}')$  implies  
 1756  $\gamma = \text{ev}(\sigma)$  for some  $\sigma \in \text{Tr}^+(\mathbf{G}')$ . This implies  $\sigma = \text{pq}\lambda_k \cdot \sigma'$  and  $\gamma = [\sigma_0]_{\sim}$  with either  
 1757  $\sigma_0 \sim \text{pq}\lambda_k \cdot \sigma'_0$  for some  $k \in I$  or  $\text{part}(\sigma_0) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$  by Definition 7.6. Then  $\text{pq}\lambda_k \bullet \gamma$   
 1758 is defined unless  $\sigma_0 = \text{pq}\lambda_k$  by Definition 8.9(1). We consider two cases.  
 1759 If  $\sigma_0 = \text{pq}\lambda_k$ , then  $\alpha \circ \gamma = [\text{pq}\lambda_k]_{\sim}$  since  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . We conclude  $\alpha \circ \gamma \in \mathcal{G}(\mathbf{G})$   
 1760 by Definition 7.11(1). Otherwise let  $\gamma' = \text{pq}\lambda_k \bullet \gamma$ . By Lemma 8.11(2)  $\gamma' \in \mathcal{G}(\mathbf{G}'_k)$ .  
 1761 By induction  $\alpha \circ \gamma' \in \mathcal{G}(\mathbf{G}_k)$ . By Lemma 8.11(1)  $\text{pq}\lambda_k \circ (\alpha \circ \gamma') \in \mathcal{G}(\mathbf{G})$ . We now  
 1762 show that  $\text{pq}\lambda_k \circ (\alpha \circ \gamma') = \alpha \circ \gamma$ . By Lemma 8.10(7) and  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$  we  
 1763 get  $\text{pq}\lambda_k \circ (\alpha \circ \gamma') = \alpha \circ (\text{pq}\lambda_k \circ \gamma')$  and by Lemma 8.10(1) we have  $\text{pq}\lambda_k \circ \gamma' =$   
 1764  $\text{pq}\lambda_k \circ (\text{pq}\lambda_k \bullet \gamma) = \gamma$ . Therefore  $\text{pq}\lambda_k \circ (\alpha \circ \gamma') = \alpha \circ \gamma \in \mathcal{G}(\mathbf{G})$ .

1765 (2) For rule [Ecomm] we get  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\mathbf{G}' = \mathbf{G}_k$  and  $\alpha = \text{pq}\lambda_k$   
 1766 for some  $k \in I$ . We conclude  $\alpha \bullet \gamma \in \mathcal{G}(\mathbf{G}')$  by Lemma 8.11(2).  
 1767 For rule [Icomm] we get  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}_i$  and  $\mathbf{G} = \mathbf{p} \rightarrow \mathbf{q} : \boxplus_{i \in I} \lambda_i; \mathbf{G}'_i$  and  
 1768  $\mathbf{G}_i \xrightarrow{\alpha} \mathbf{G}'_i$  for all  $i \in I$  and  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . By Definition 7.11(1)  $\gamma \in \mathcal{G}(\mathbf{G})$  implies  
 1769  $\gamma = \text{ev}(\sigma)$  for some  $\sigma \in \text{Tr}^+(\mathbf{G})$ . This implies  $\sigma = \text{pq}\lambda_k \cdot \sigma'$  and  $\gamma = [\sigma_0]_{\sim}$  with either  
 1770  $\sigma_0 \sim \text{pq}\lambda_k \cdot \sigma'_0$  for some  $k \in I$  or  $\text{part}(\sigma_0) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$  by Definition 7.6. Then  $\text{pq}\lambda_k \bullet \gamma$   
 1771 is defined unless  $\sigma_0 = \text{pq}\lambda_k$  by Definition 8.9(1). We consider two cases.  
 1772 If  $\sigma_0 = \text{pq}\lambda_k$ , then  $\alpha \bullet \gamma = [\text{pq}\lambda_k]_{\sim}$  since  $\text{part}(\alpha) \cap \{\mathbf{p}, \mathbf{q}\} = \emptyset$ . We conclude  $\alpha \bullet \gamma \in \mathcal{G}(\mathbf{G}')$   
 1773 by Definition 7.11(1). Otherwise let  $\gamma' = \text{pq}\lambda_k \bullet \gamma$ . By Lemma 8.11(2)  $\gamma' \in \mathcal{G}(\mathbf{G}_k)$ .

1774 We first show that  $\alpha \bullet \gamma'$  is defined. Since  $\alpha \bullet \gamma$  and  $\text{pq}\lambda_k \bullet \gamma$  are defined, by  
 1775 Definition 8.9(1) we have four cases:

- 1776 (a)  $\sigma_0 \sim \alpha \cdot \sigma_1$  for some  $\sigma_1$  and  $\sigma_0 \sim \text{pq}\lambda_k \cdot \sigma'_0$ ;
- 1777 (b)  $\sigma_0 \sim \alpha \cdot \sigma_1$  and  $\text{part}(\sigma_0) \cap \{p, q\} = \emptyset$ ;
- 1778 (c)  $\text{part}(\alpha) \cap \text{part}(\sigma_0) = \emptyset$  and  $\sigma_0 \sim \text{pq}\lambda_k \cdot \sigma'_0$ ;
- 1779 (d)  $\text{part}(\alpha) \cap \text{part}(\sigma_0) = \emptyset$  and  $\text{part}(\sigma_0) \cap \{p, q\} = \emptyset$ .

1780 In case (a)  $\sigma_0 \sim \alpha \cdot \text{pq}\lambda_k \cdot \sigma'_1 \sim \text{pq}\lambda_k \cdot \alpha \cdot \sigma'_1$  for some  $\sigma'_1$  since  $\text{part}(\alpha) \cap \{p, q\} = \emptyset$ . Notice  
 1781 that  $\sigma'_1 \neq \epsilon$  since  $\sigma_0$  is pointed and  $\text{part}(\alpha) \cap \{p, q\} = \emptyset$ . We get  $\gamma' = \text{pq}\lambda_k \bullet \gamma = [\alpha \cdot \sigma'_1]_{\sim}$   
 1782 and  $\alpha \bullet \gamma' = [\sigma'_1]_{\sim}$ .

1783 In case (b)  $\gamma' = \gamma$  and  $\alpha \bullet \gamma' = [\sigma_1]_{\sim}$ .

1784 In case (c)  $\gamma' = [\sigma'_0]_{\sim}$  and  $\alpha \bullet \gamma' = [\sigma'_0]_{\sim}$ , since  $\text{part}(\alpha) \cap \text{part}(\sigma_0) = \emptyset$  implies  
 1785  $\text{part}(\alpha) \cap \text{part}(\sigma'_0) = \emptyset$ .

1786 In case (d)  $\gamma' = \gamma$  and  $\alpha \bullet \gamma' = \gamma$ .

1787 By induction  $\alpha \bullet \gamma' \in \mathcal{GE}(G'_k)$ . By Lemma 8.11(1)  $\text{pq}\lambda_k \circ (\alpha \bullet \gamma') \in \mathcal{GE}(G')$ .

1788 We now show that  $\text{pq}\lambda_k \circ (\alpha \bullet \gamma') = \alpha \bullet \gamma$ . From  $\gamma' = \text{pq}\lambda_k \bullet \gamma$  and Lemma 8.10(1)  
 1789  $\text{pq}\lambda_k \circ \gamma' = \gamma$ . Therefore from  $\alpha \bullet \gamma$  defined we have  $\alpha \bullet (\text{pq}\lambda_k \circ \gamma')$  defined.  
 1790 Since  $\alpha \bullet \gamma'$  is also defined and  $\text{part}(\alpha) \cap \{p, q\} = \emptyset$ , by Lemma 8.10(8) we get  
 1791  $\text{pq}\lambda_k \circ (\alpha \bullet \gamma') = \alpha \bullet (\text{pq}\lambda_k \circ \gamma')$ . Therefore  $\text{pq}\lambda_k \circ (\alpha \bullet \gamma') = \alpha \bullet \gamma \in \mathcal{GE}(G')$ .

## 1792 F. Glossary of Symbols and Table of Notations

Symbol	Meaning
$\pi$	input/output action: $p!\lambda, p?\lambda$
$\alpha$	communication $\text{pq}\lambda$
$\sigma$	trace, finite sequence of communications
$S$	event structure
$\mathcal{X}$	configuration of an event structure
$\eta$	p-event, non-empty finite sequence of input/output actions
$\mathcal{PE}$	set of p-events
$\zeta$	(possibly empty) finite sequence of input/output actions
$\vartheta$	undirected action: $!\lambda, ?\lambda$
$\Theta$	(possibly empty) finite sequence of undirected actions
$\nu$	n-event, unordered pair of dual located p-events
$\mathcal{NE}$	set of n-events
$\gamma$	g-event, equivalence class $[\sigma]_{\sim}$ with $\sigma$ pointed
$\mathcal{GE}$	set of g-events

1794

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