Event Structure Semantics for Multiparty Sessions

Ilaria Castellani*

joint work with Mariangiola Dezani† and Paola Giannini‡

* INRIA Sophia Antipolis, France
† Dipartimento di Informatica, Università di Torino
‡ DiSIT, Università del Piemonte Orientale

OPCT'23, Bertinoro June 26-30, 2023

Overview (1/4)

Investigate connection between:

Multiparty Session Types

[HYC08,HYC16]

Event Structures

[Win80,NPW81]

Overview (2/4)

ES semantics for a core MPST calculus

Overview (2/4)

ES semantics for a core MPST calculus where sessions are:

- described as networks of sequential processes

Overview (2/4)

ES semantics for a core MPST calculus where sessions are:

- described as networks of sequential processes
- specified by global types

Overview (2/4)

ES semantics for a core MPST calculus where sessions are:

- described as networks of sequential processes
- specified by global types

ES interpretation for both sessions and types:

$$\begin{aligned} \text{session} &\longrightarrow \text{Flow ES} \\ \text{global type} &\longrightarrow \text{Prime ES} \end{aligned}$$

Main result: isomorphism of configuration domains.

Overview (2/4)

ES semantics for a core MPST calculus where sessions are:

- described as networks of sequential processes
- specified by global types

ES interpretation for both sessions and types:

$$\begin{aligned} \text{session} &\longrightarrow \text{Flow ES} \\ \text{global type} &\longrightarrow \text{Prime ES} \end{aligned}$$

Main result: isomorphism of configuration domains.

Holds for both synchronous and asynchronous communication.

Overview (3/4)

ES semantics for a core synchronous MPST calculus where sessions are:

- described as networks of sequential processes
- specified by global types

ES interpretation for both sessions and types:

$$\text{session} \longrightarrow \text{Flow ES}$$
$$\text{global type} \longrightarrow \text{Prime ES}$$

Main result: isomorphism of configuration domains.

I. Castellani, M. Dezani-Ciancaglini and P. Giannini. Event Structure Semantics for Multiparty Sessions, JLAMP 131, February 2023.

https://doi.org/10.1016/j.jlamp.2022.100844

Overview (4/4)

ES semantics for a core asynchronous MPST calculus where sessions are:

- described as networks of sequential processes with queues
- specified by asynchronous types

ES interpretation for both sessions and types:

$$\text{asynchronous session} \longrightarrow \text{Flow ES}$$
$$\text{asynchronous type} \longrightarrow \text{Prime ES}$$

Main result: isomorphism of configuration domains.

I. Castellani, M. Dezani-Ciancaglini and P. Giannini. Global Types and Event Structure Semantics for Asynchronous Multiparty Sessions.

`https://arxiv.org/abs/2102.00865`

A quick tour on Event Structures

Event Structures



- [NPW79,NPW81] M. Nielsen, G. Plotkin and G. Winskel. Petri Nets, Event Structures and Domains. LNCS 70, 1979, and TCS 13(1), 1981.
- [Win80] G. Winskel. Events in Computation. PhD Thesis, University of Edinburgh, 1980.

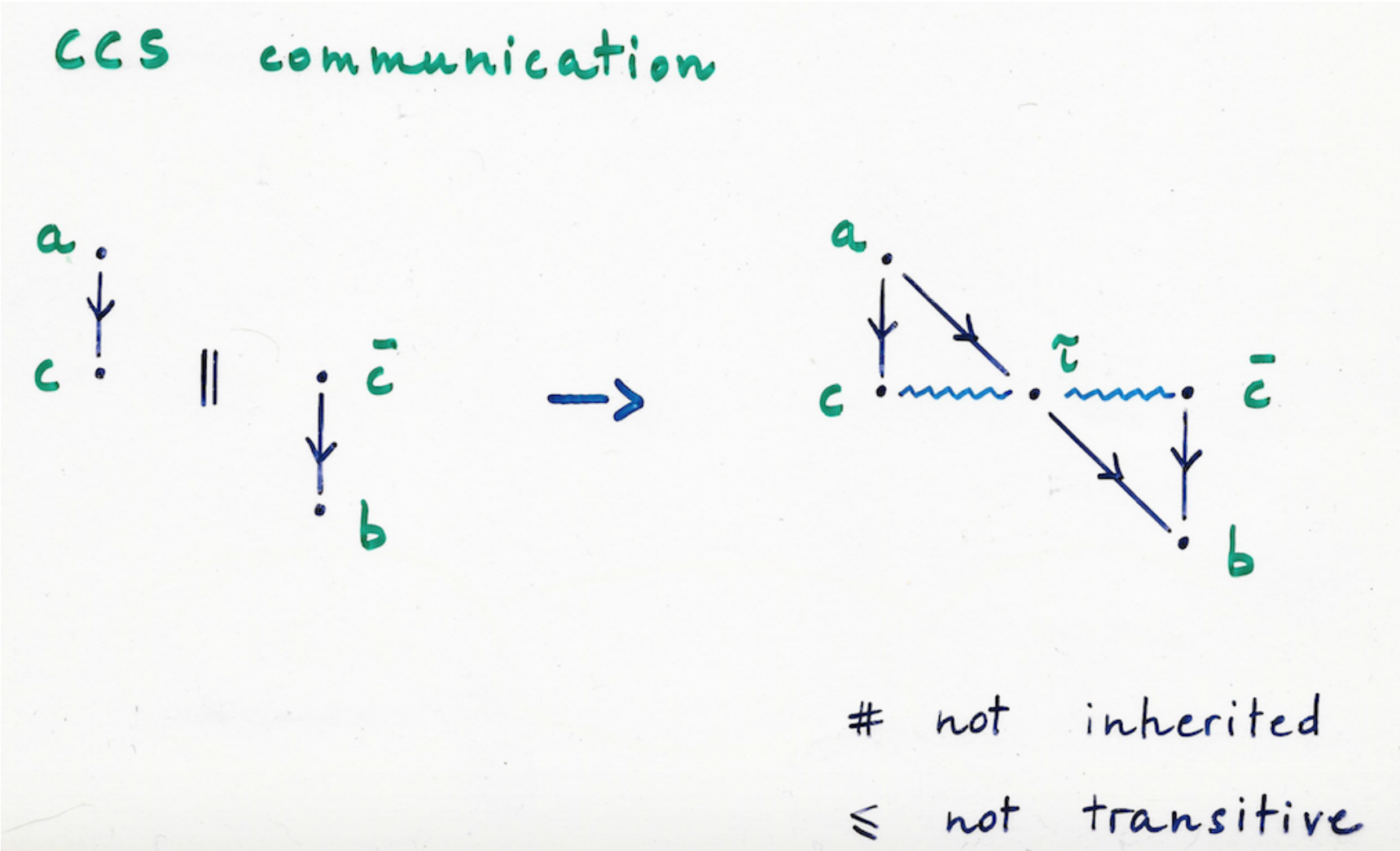
A canonical model for (true) concurrency, strongly connected with particular classes of Petri nets and domains.

Prime event structures

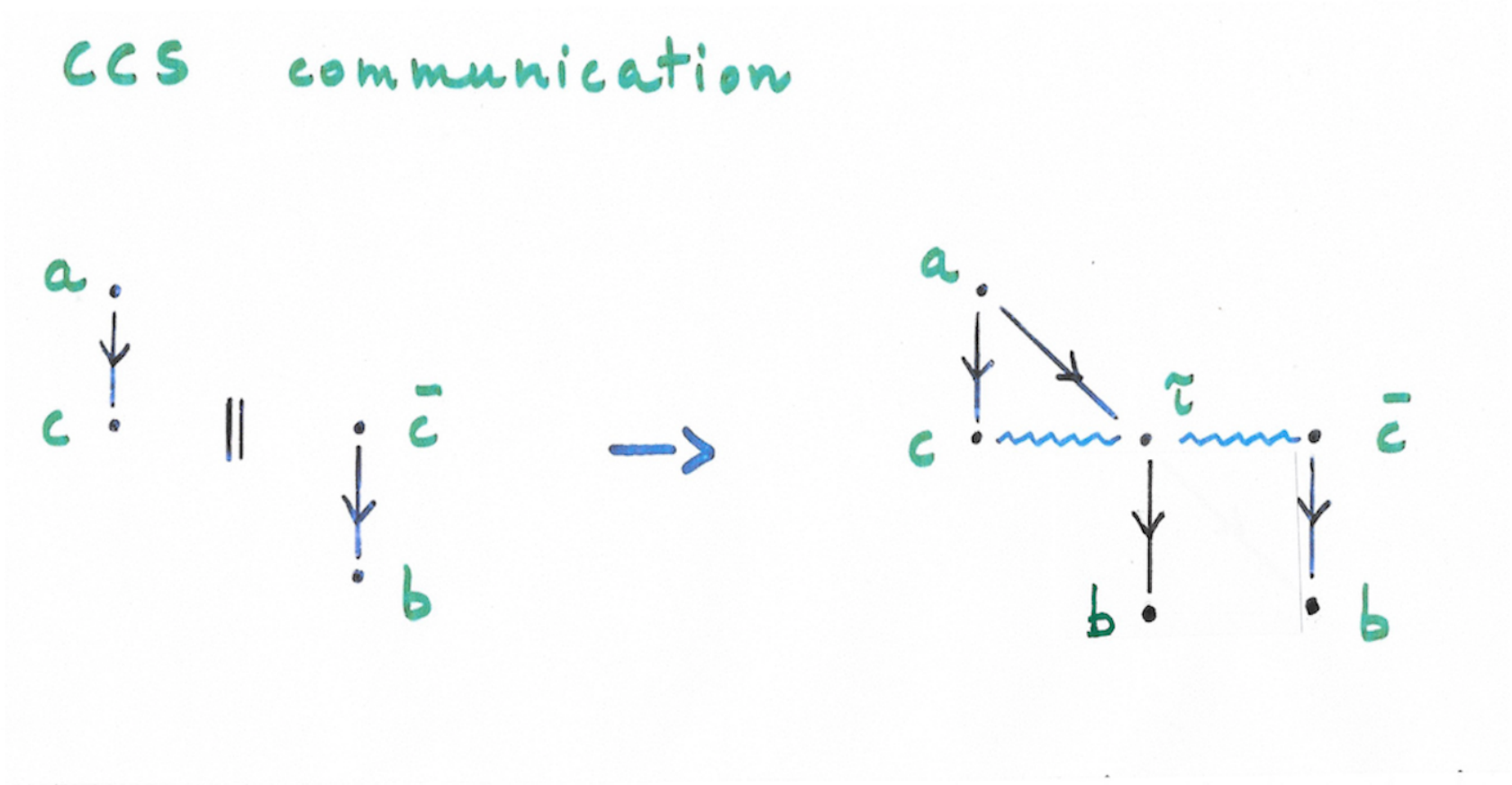
A prime event structure [NPW79] is a tuple $S = (E, \leq, \#)$ where:

1. E is a denumerable set of events;
2. the causality relation $\leq \subseteq (E \times E)$ is a partial order;
3. the conflict relation $\# \subseteq (E \times E)$ is irreflexive, symmetric and satisfies conflict hereditariness: $e \# e' \leq e'' \Rightarrow e \# e''$

Prime ES semantics for process calculi?



Prime ES semantics for process calculi



Drawback: duplication of events after communication.

Stable ES semantics for process calculi

Stable ESs: more general ESs allowing for conflicting causes, while ensuring causal non-ambiguity within configurations.
Used by Winskel to define the first ES semantics for CCS:

- [Win82] Glynn Winskel. Event structure semantics for CCS and related languages, ICALP 1982.

Flow ESs: subclass of Stable ESs, graphical representation.
Used by Boudol and C. to give semantics to CCS and SCCS:

- [BC88] G. Boudol and I. Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS, REX School/Workshop, 1988.
- [BC94] G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS, Inf. and Comp., 1994.

Relations among some classes of ESs

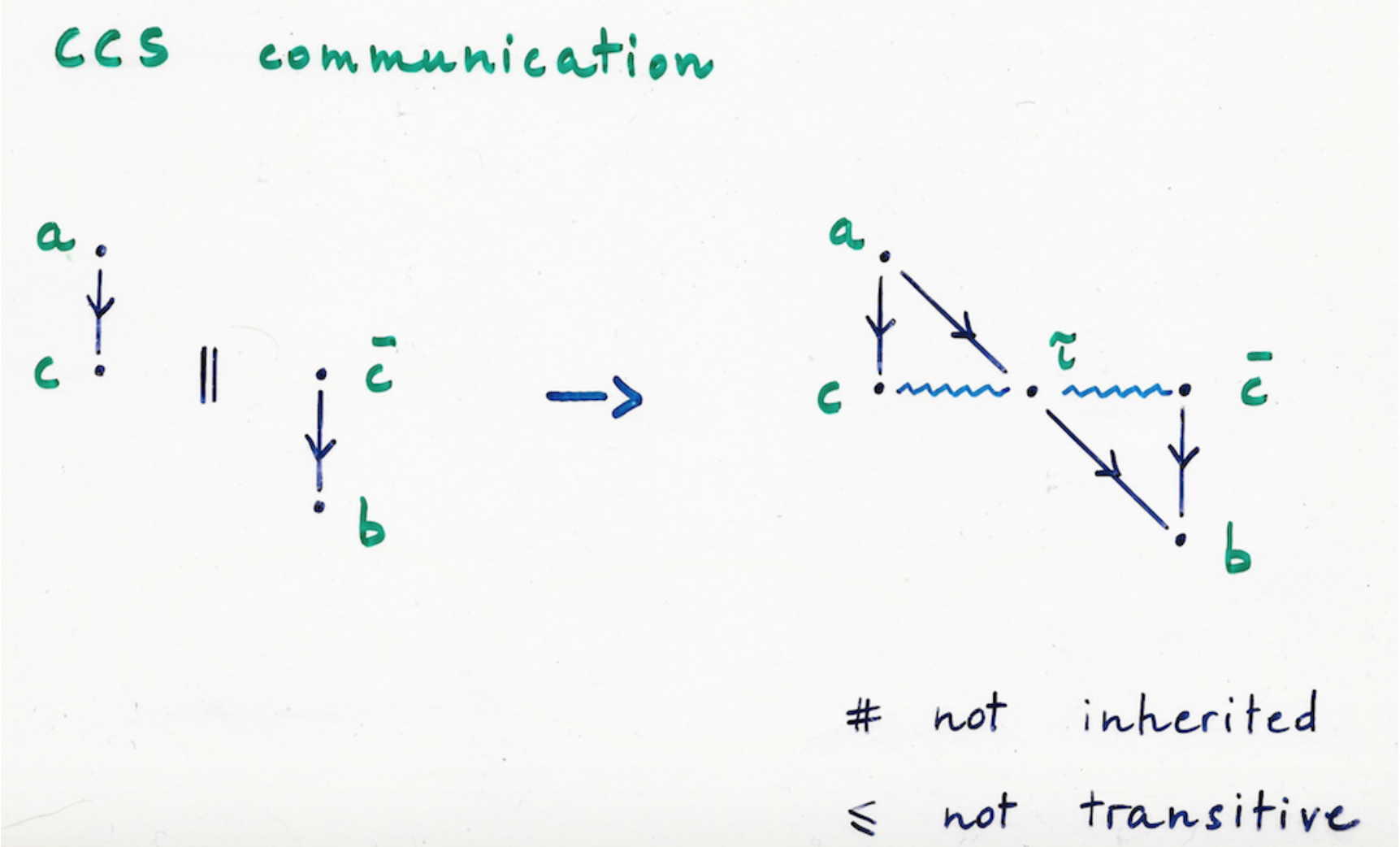
Bundle ESs: subclass of Stable ESs, using bundles (sets of pairwise conflicting causes of an event) to express causality. Used by Langerak and Katoen to give semantics to LOTOS:

- [Lan93] R. Langerak. Bundle Event Structures: a Non-Interleaving Semantics for LOTOS. Formal Description Techniques for Distributed Systems and Communication Protocols, North-Holland, 1993.
- [Kat96] J-P. Katoen. Quantitative and qualitative extensions of event structures. PhD Thesis, University of Twente, 1996.

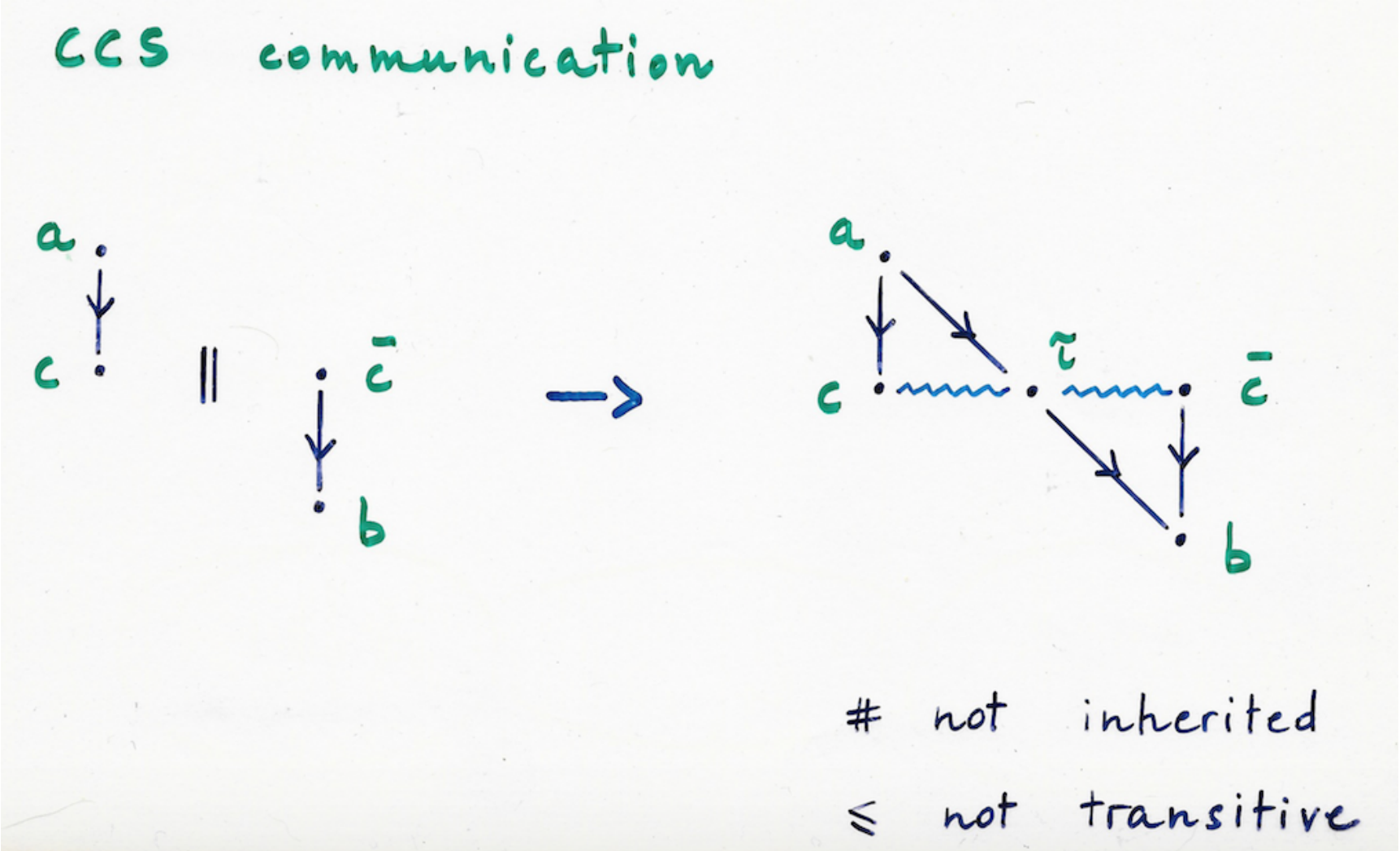
$$\textit{Prime ESs} \subset \textit{Bundle ESs} \subset \textit{Flow ESs} \subset \textit{Stable ESs} \subset \textit{General ESs}$$

- [BC91] Flow models of distributed computations: event structures and nets, G. Boudol and I. Castellani. INRIA Research Report n. 1482 (1991).

Flow ES semantics for process calculi



Flow ES semantics for process calculi



Idea: relax $\#$ and replace $\leq$ by $\prec$, immediate causality.

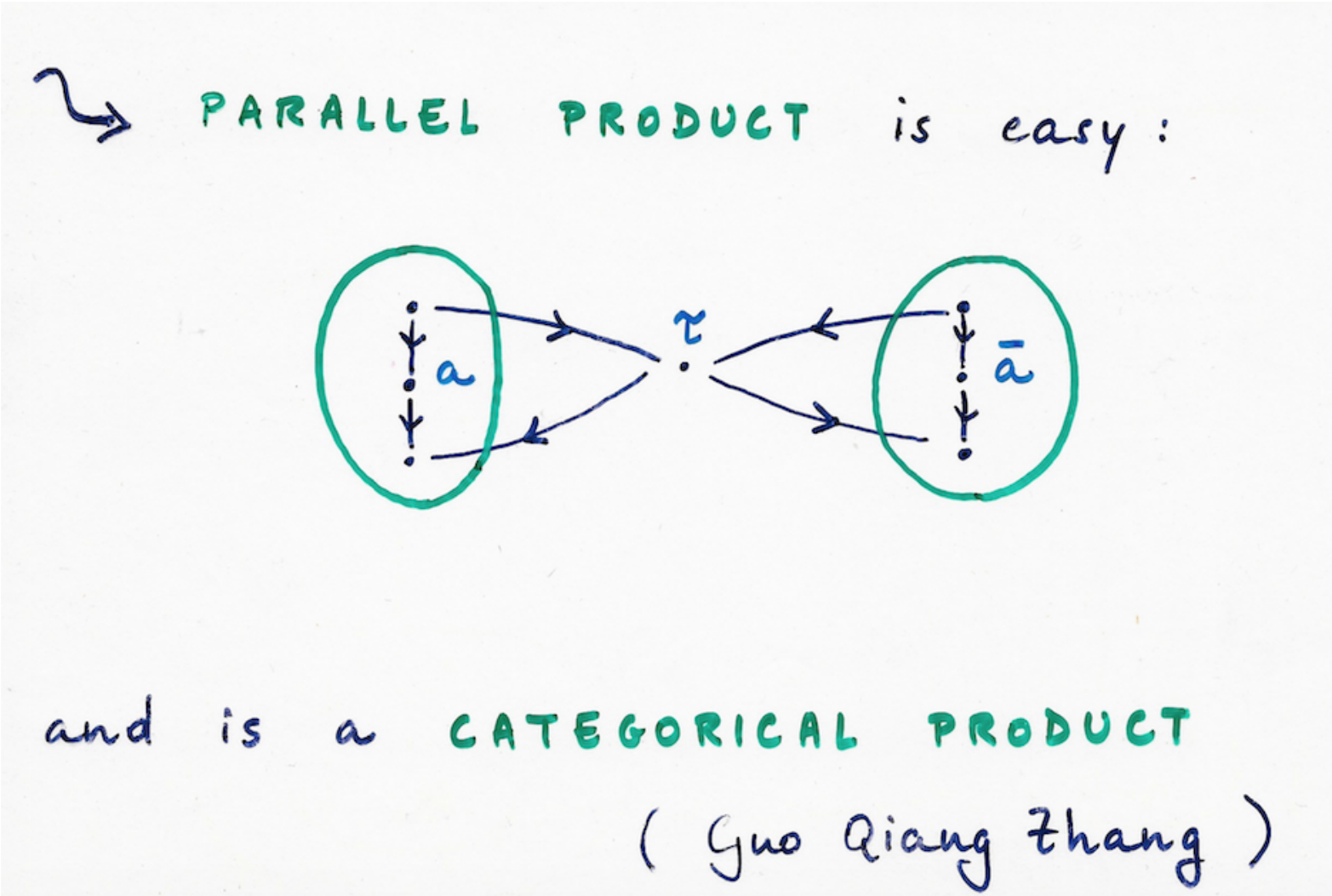
Flow event structures

A flow event structure [BC88] is a tuple $S = (E, \prec, \#)$ where:

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

New feature: self-conflicts (to model CCS restriction).

Flow ES product



– [CZ97] I. Castellani and G. Q. Zhang. Parallel product of event structures. TCS 179, 1997.

Prime ES Configurations

Configuration: set of events having occurred at some stage of execution.

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

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

Flow ES Configurations

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

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

Domain of configurations

Let S be a PES or a FES with set of configurations $\mathcal{C}(S)$.

The domain of configurations of S is the partially ordered set

$$S =_{\text{def}} (\mathcal{C}(S), \subseteq)$$

Prime algebraic coherent domains, whose prime elements are in 1-1 correspondence with the events of S (if S is a PES).

Cf picture from [NPW79] on next slide

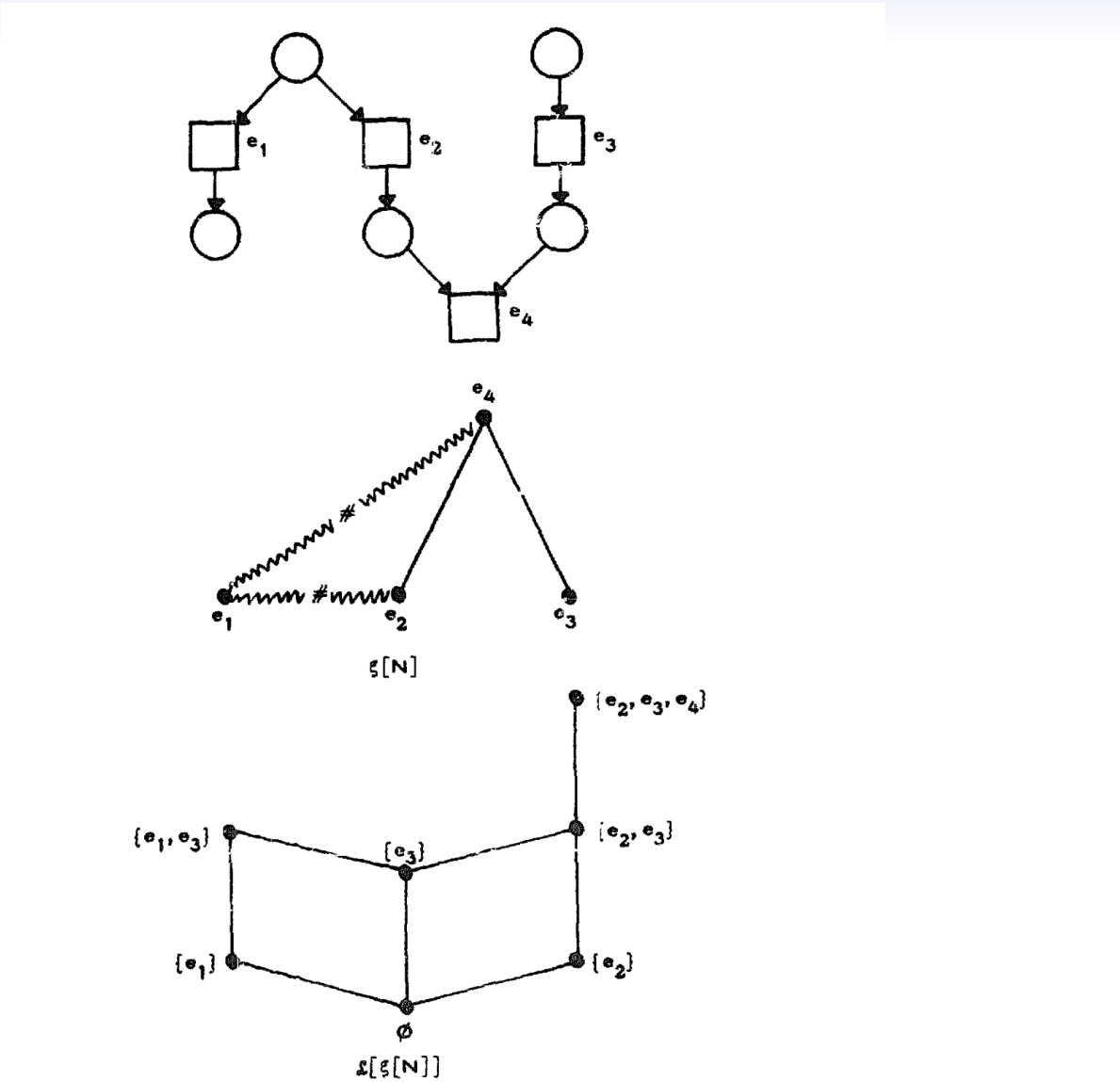


Fig. 7.

Session Types in one slide

- Types for interaction: STs specify process interaction within units called sessions, regulated by a protocol.
- First introduced for binary sessions, then extended to multiparty sessions.

Session Types in one slide

- Types for interaction: STs specify process interaction within units called sessions, regulated by a protocol.
- First introduced for binary sessions, then extended to multiparty sessions.
- K. Honda, V. Vasconcelos and M. Kubo. Language Primitives and Type Disciplines for Structured Communication-based Programming. ESOP 1998.
- K. Honda, N. Yoshida and M. Carbone. Multiparty asynchronous session types. POPL 2008. Most Influential POPL Paper Award 2018.




Menu

Menu

- Multiparty sessions

Menu

- Multiparty sessions

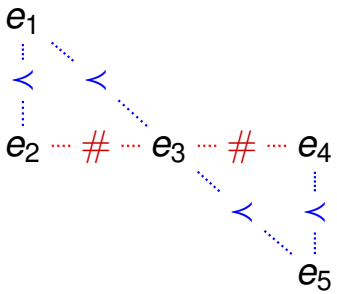
- Global types

Menu

- Multiparty sessions

- Global types

- Event Structure semantics

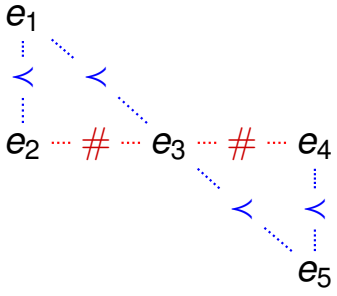


Menu

- Multiparty sessions

- Global types

- Event Structure semantics



- Isomorphism result

A core synchronous MPST calculus

Processes

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

Processes

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

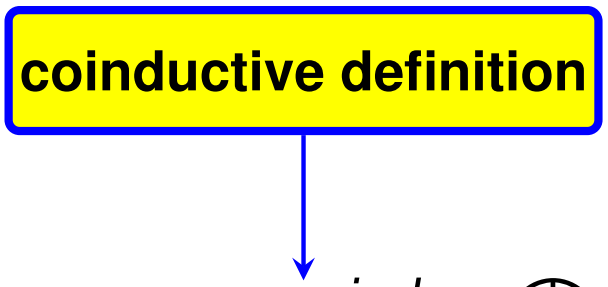
$$\lambda_i \neq \lambda_j \text{ for any } i \neq j$$

Processes

coinductive definition

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

Processes



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

We focus on regular processes.

Trees with leaves decorated by **0**, internal nodes by p! or p?, and edges by labels λ.

Multiparty Sessions

$$\mathsf{p}_1[\![\, P_1 \,]\!] \,||\, \cdots \,||\, \mathsf{p}_n[\![\, P_n \,]\!]$$

$$\mathsf{p}_i \neq \mathsf{p}_j \;\text{ for any }\; i \neq j$$

LTS for Multiparty Sessions

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

Transitions are labelled by communications.

$\mathsf{pq}\lambda$: communication of label λ from p to q.

LTS for sessions: classical deadlock

Concurrent mutual sends:

$$\mathsf{p}[\![\, \mathsf{q}!\lambda; \mathsf{q}?\lambda' \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}!\lambda'; \mathsf{p}?\lambda \,]\!]$$

Deadlock due to mismatching expectations.

LTS for sessions: 3-philosopher deadlock

Deadlock due to circular dependencies

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

NB Here participants have matching expectations.

LTS for sessions: starvation

Starvation due to a denying choice

$$\mathsf{N} = \mathsf{p}[\![P]\!] \,\|\, \mathsf{q}[\![Q]\!] \,\|\, \mathsf{r}[\![R]\!]$$

$$\begin{aligned} P &= \mathsf{q}!\lambda_1 && \oplus && \mathsf{q}!\lambda_2 ; P \\ Q &= \mathsf{p}?\lambda_1 ; \mathsf{r}!\lambda && + && \mathsf{p}?\lambda_2 ; Q \\ R &= \mathsf{q}?\lambda \end{aligned}$$

Eluding path $\mathsf{N} \xrightarrow{\mathsf{pq}\lambda_2} \cdot \xrightarrow{\mathsf{pq}\lambda_2} \cdots$

Global Types: syntax

Global Types: syntax

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

$$\lambda_i \neq \lambda_j \ \text{ for any } \ i \neq j$$

Again, we focus on regular terms.

Trees with internal nodes decorated by channels pq, leaves by End, and edges by labels λ.

LTS for Global Types

LTS for Global Types

$$\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i ; \mathsf{G}_i \xrightarrow{\mathsf{pq}\lambda_k} \mathsf{G}_k \qquad \text{where} \quad k \in I$$

LTS for Global Types

$$\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i ; \mathsf{G}_i \xrightarrow{\mathsf{pq}\lambda_k} \mathsf{G}_k \qquad \text{where} \quad k \in I$$

Also internal communications are allowed:

LTS for Global Types

$$\mathsf{p} \to \mathsf{q} : \boxplus_{i \in I} \lambda_i ; \mathsf{G}_i \xrightarrow{\mathsf{pq}\lambda_k} \mathsf{G}_k \qquad \text{where} \quad k \in I$$

Also internal communications are allowed:

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

Projection

Projection

Global types are projected on participants, yielding processes.

Projection

Global types are projected on participants, yielding processes.

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

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

Boundedness

Boundedness

G is bounded if for each subtree G′ of G the depth of the 1st occurrence of each participant in the paths of G' is bounded.

Boundedness

G is bounded if for each subtree G′ of G the depth of the 1st occurrence of each participant in the paths of G' is bounded.

$$\mathsf{G} = \mathsf{p} \to \mathsf{q} : (\lambda_1; \mathsf{q} \xrightarrow{\lambda} \mathsf{r} \boxplus \lambda_2; \mathsf{G})$$

The type G is projectable but not bounded.

Boundedness ensures that each participant can move.

Type system

Type system

Well formedness: boundedness + projectability

Type system

Well formedness: boundedness + projectability

$$0 \leq 0$$

Type system

Well formedness: boundedness + projectability

$$0 \leq 0 \qquad \frac{P_i \leq Q_i \quad \text{for all } i \in I}{\bigoplus_{i \in I} \mathsf{p}!\lambda_i; P_i \leq \bigoplus_{i \in I} \mathsf{p}!\lambda_i; Q_i}$$

Type system

Well formedness: boundedness + projectability

$$0 \leq 0 \qquad \frac{P_i \leq Q_i \quad \text{for all } i \in I}{\bigoplus_{i \in I} \mathsf{p}!\lambda_i; P_i \leq \bigoplus_{i \in I} \mathsf{p}!\lambda_i; Q_i}$$

$$\frac{P_i \leq Q_i \quad \text{for all } i \in I}{\sum_{i \in I \cup J} \mathsf{p}?\lambda_i; P_i \leq \sum_{i \in I} \mathsf{p}?\lambda_i; Q_i}$$

Type system

Well formedness: boundedness + projectability

$$0 \leq 0 \qquad \frac{P_i \leq Q_i \quad \text{for all } i \in I}{\bigoplus_{i \in I} \mathtt{p}!\lambda_i; P_i \leq \bigoplus_{i \in I} \mathtt{p}!\lambda_i; Q_i}$$

$$\frac{P_i \leq Q_i \quad \text{for all } i \in I}{\sum_{i \in I \cup J} \mathtt{p}?\lambda_i; P_i \leq \sum_{i \in I} \mathtt{p}?\lambda_i; Q_i}$$

$$\frac{P_i \leq \mathtt{G} \upharpoonright \mathtt{p}_i \quad \text{for all } i \in I \qquad \mathrm{part}(\mathtt{G}) \subseteq \{\mathtt{p}_i \mid i \in I\}}{\vdash \prod_{i \in I} \mathtt{p}_i [\![P_i]\!] : \mathtt{G}}$$

Type system

Well formedness: boundedness + projectability

$$0 \leq 0 \qquad \frac{P_i \leq Q_i \quad \text{for all } i \in I}{\bigoplus_{i \in I} \mathtt{p}!\lambda_i; P_i \leq \bigoplus_{i \in I} \mathtt{p}!\lambda_i; Q_i}$$

$$\frac{P_i \leq Q_i \quad \text{for all } i \in I}{\sum_{i \in I \cup J} \mathtt{p}?\lambda_i; P_i \leq \sum_{i \in I} \mathtt{p}?\lambda_i; Q_i}$$

$$\frac{P_i \leq \mathtt{G} \upharpoonright \mathtt{p}_i \quad \text{for all } i \in I \qquad \mathrm{part}(\mathtt{G}) \subseteq \{\mathtt{p}_i \mid i \in I\}}{\vdash \prod_{i \in I} \mathtt{p}_i [\![P_i]\!] : \mathtt{G}}$$

Type system

Well formedness: boundedness + projectability

$$0 \leq 0 \qquad \frac{P_i \leq Q_i \quad \text{for all } i \in I}{\bigoplus_{i \in I} \mathtt{p}!\lambda_i; P_i \leq \bigoplus_{i \in I} \mathtt{p}!\lambda_i; Q_i}$$

$$\frac{P_i \leq Q_i \quad \text{for all } i \in I}{\sum_{i \in I \cup J} \mathtt{p}?\lambda_i; P_i \leq \sum_{i \in I} \mathtt{p}?\lambda_i; Q_i}$$

$$\frac{P_i \leq \mathtt{G} \upharpoonright \mathtt{p}_i \quad \text{for all } i \in I \qquad \mathtt{part}(\mathtt{G}) \subseteq \{\mathtt{p}_i \mid i \in I\}}{\vdash \prod_{i \in I} \mathtt{p}_i [\![P_i]\!] : \mathtt{G}}$$

Properties

Properties

Theorem (Subject Reduction)

If $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$, *then* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

Properties

Theorem (Subject Reduction)

If $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$, *then* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

Theorem (Session Fidelity)

If $\vdash \mathsf{N} : \mathsf{G}$ *and* $\mathsf{G} \xrightarrow{\alpha} \mathsf{G}'$, *then* $\mathsf{N} \xrightarrow{\alpha} \mathsf{N}'$ *and* $\vdash \mathsf{N}' : \mathsf{G}'$.

Properties

Progress: every participant whose process is not terminated will eventually move.

$\mathsf{p}[\![P]\!] \in \mathsf{N}$: means $P \neq \mathbf{0}$ and $\mathsf{N} \equiv \mathsf{p}[\![P]\!] \parallel \mathsf{N}'$ for some N'.

Under some fairness assumption:

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

Event Structure Semantics

Event Structure Semantics

Both networks and global types are modelled as ESs:

$$\text{network} \longrightarrow \text{Flow ES}$$

$$\text{type} \longrightarrow \text{Prime ES}$$

Event Structure Semantics

Both networks and global types are modelled as ESs:

$$\text{network} \longrightarrow \text{Flow ES}$$
$$\text{type} \longrightarrow \text{Prime ES}$$

First step to model networks:

$$\text{process} \longrightarrow \text{Prime ES}$$

Prime ES Semantics of Processes

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda;(\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad \ldots$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda; (\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$ $\quad$ $\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$ $\quad \ldots$

$\eta \leq \eta'$ if η is a prefix of η'

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda;(\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$$\mathsf{p}!\lambda \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$$
$$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2 \qquad \ldots$$

$\eta \leq \eta'$ if η is a prefix of η'

$$\mathsf{p}!\lambda \;\leq\; \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$$

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda;(\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2 \qquad \ldots$

$\eta \leq \eta'$ if η is a prefix of η'

$$\mathsf{p}!\lambda \ \leq \ \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$$

$\eta \# \eta'$ if η and η' fork at some point

Prime ES Semantics of Processes

Process event (p-event): path in the syntactic tree of a process:

$\eta = \zeta \cdot \pi$ represents a specific occurrence of the action π preceded by its local history ζ.

$$P = \mathsf{p}!\lambda;(\mathsf{q}?\lambda_1; P + \mathsf{q}?\lambda_2)$$

$\mathsf{p}!\lambda \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda$

$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \qquad \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2 \qquad \ldots$

$\eta \leq \eta'$ if η is a prefix of η'

$$\mathsf{p}!\lambda \;\leq\; \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1$$

$\eta \# \eta'$ if η and η' fork at some point

$$\mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_1 \cdot \mathsf{p}!\lambda \;\#\; \mathsf{p}!\lambda \cdot \mathsf{q}?\lambda_2$$

Flow ES Semantics of Sessions: events

Located event p :: η: p-event η pertaining to a participant p.

$\eta \twoheadrightarrow$ p : projection of p-event η on participant p.

Duality of located events:

$$\mathsf{p} :: \eta \ \bowtie \ \mathsf{q} :: \eta' \text{ if}$$
$$\eta \twoheadrightarrow \mathsf{q} \ \bowtie \ \eta' \twoheadrightarrow \mathsf{p} \text{ and } \mathrm{part}(\mathrm{act}(\eta)) = \mathsf{q} \text{ and } \mathrm{part}(\mathrm{act}(\eta')) = \mathsf{p}$$

Network event (n-event): unordered pair of dual located events:

$$\{\mathsf{p} :: \zeta \cdot \pi, \mathsf{q} :: \zeta' \cdot \pi'\}$$

with matching actions π and π' and matching histories ζ and ζ'.

Flow ES Semantics of Sessions: relations

Network event (n-event): $\nu = \{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}$

Flow relation $\nu \prec \nu'$ if $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta < \eta'$

Flow ES Semantics of Sessions: relations

Network event (n-event): $\nu = \{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}$

Flow relation $\nu \prec \nu'$ if $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta < \eta'$

Conflict relation $\nu \# \nu'$ if 1) $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta \# \eta'$

Flow ES Semantics of Sessions: relations

Network event (n-event): $\nu = \{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}$

Flow relation $\nu \prec \nu'$ if $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta < \eta'$

Conflict relation $\nu \# \nu'$ if 1) $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta \# \eta'$

or 2) ($\mathsf{p} :: \eta \in \nu$ & $\mathsf{q} :: \eta' \in \nu'$ & $\mathsf{p} \neq \mathsf{q}$ &
$|\eta \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}|$ & $\neg(\eta \upharpoonright \mathsf{q} \bowtie \eta' \upharpoonright \mathsf{p})$)

Flow ES Semantics of Sessions: relations

Network event (n-event): $\nu = \{\mathsf{p} :: \eta, \mathsf{q} :: \eta'\}$

Flow relation $\nu \prec \nu'$ if $\mathsf{p} :: \eta \in \nu \;\&\; \mathsf{p} :: \eta' \in \nu' \;\&\; \eta < \eta'$

Conflict relation $\nu \# \nu'$ if 1) $\mathsf{p} :: \eta \in \nu \;\&\; \mathsf{p} :: \eta' \in \nu' \;\&\; \eta \# \eta'$

or 2) $(\mathsf{p} :: \eta \in \nu \;\&\; \mathsf{q} :: \eta' \in \nu' \;\&\; \mathsf{p} \neq \mathsf{q} \;\&\; |\eta \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}| \;\&\; \neg(\eta \upharpoonright \mathsf{q} \bowtie \eta' \upharpoonright \mathsf{p}))$

Conflict may hold between n-events with disjoint participants.

$\nu = \{\mathsf{p} :: \mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\} \quad \# \quad \nu' = \{\mathsf{q} :: \mathsf{p}?\lambda_2 \cdot \mathsf{s}!\lambda, \mathsf{s} :: \mathsf{q}?\lambda\}$

Flow ES Semantics of Sessions: relations

Network event (n-event): $\nu = \{\mathsf{p} :: \zeta \cdot \pi, \mathsf{q} :: \zeta' \cdot \pi'\}$

Flow relation $\nu \prec \nu'$ if $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta < \eta'$

Conflict relation $\nu \# \nu'$ if 1) $\mathsf{p} :: \eta \in \nu$ & $\mathsf{p} :: \eta' \in \nu'$ & $\eta \# \eta'$

or 2) ($\mathsf{p} :: \eta \in \nu$ & $\mathsf{q} :: \eta' \in \nu'$ & $\mathsf{p} \neq \mathsf{q}$ &
$|\eta \upharpoonright \mathsf{q}| = |\eta' \upharpoonright \mathsf{p}|$ & $\neg(\eta \upharpoonright \mathsf{q} \bowtie \eta' \upharpoonright \mathsf{p})$)

Conflict may hold between n-events with disjoint participants.

$\nu = \{\mathsf{p} :: \mathsf{q}!\lambda_1 \cdot \mathsf{r}!\lambda, \mathsf{r} :: \mathsf{p}?\lambda\} \quad \# \quad \nu' = \{\mathsf{q} :: \mathsf{p}?\lambda_2 \cdot \mathsf{s}!\lambda, \mathsf{s} :: \mathsf{q}?\lambda\}$

Pruning non causally-founded n-events

Some n-events $\nu = \{\mathsf{p} :: \zeta \cdot \pi, \mathsf{q} :: \zeta' \cdot \pi'\}$ are not admissible.

The network

$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}?\lambda \cdot \mathsf{r}!\lambda' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\lambda' \,]\!]$$

has the unique n-event $\nu = \{\mathsf{p} :: \mathsf{q}?\lambda \cdot \mathsf{r}!\lambda', \mathsf{r} :: \mathsf{p}?\lambda'\}$.

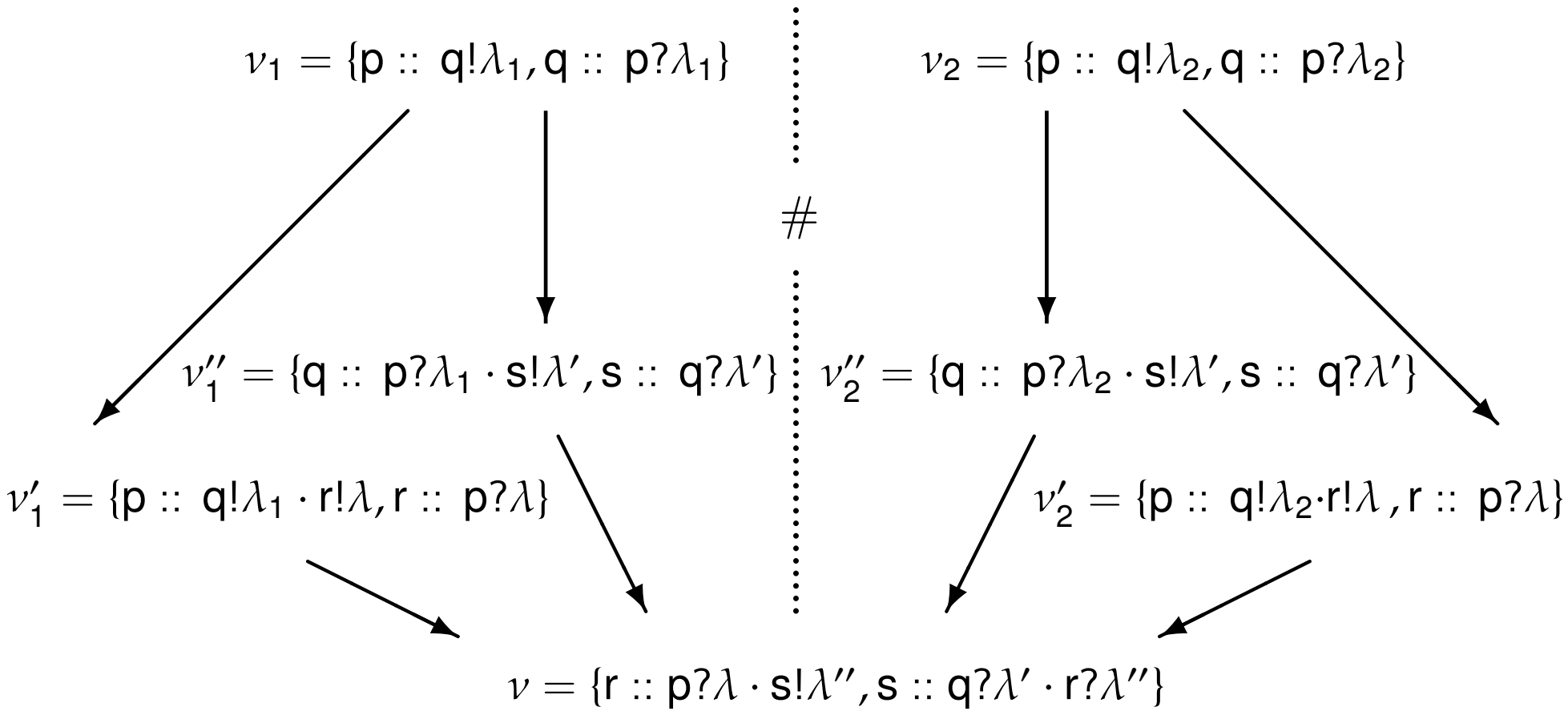
Since ν is minimal, then $\{\nu\}$ would be a configuration of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$.

WRONG! $\Longrightarrow$ need for causal well-foundedness

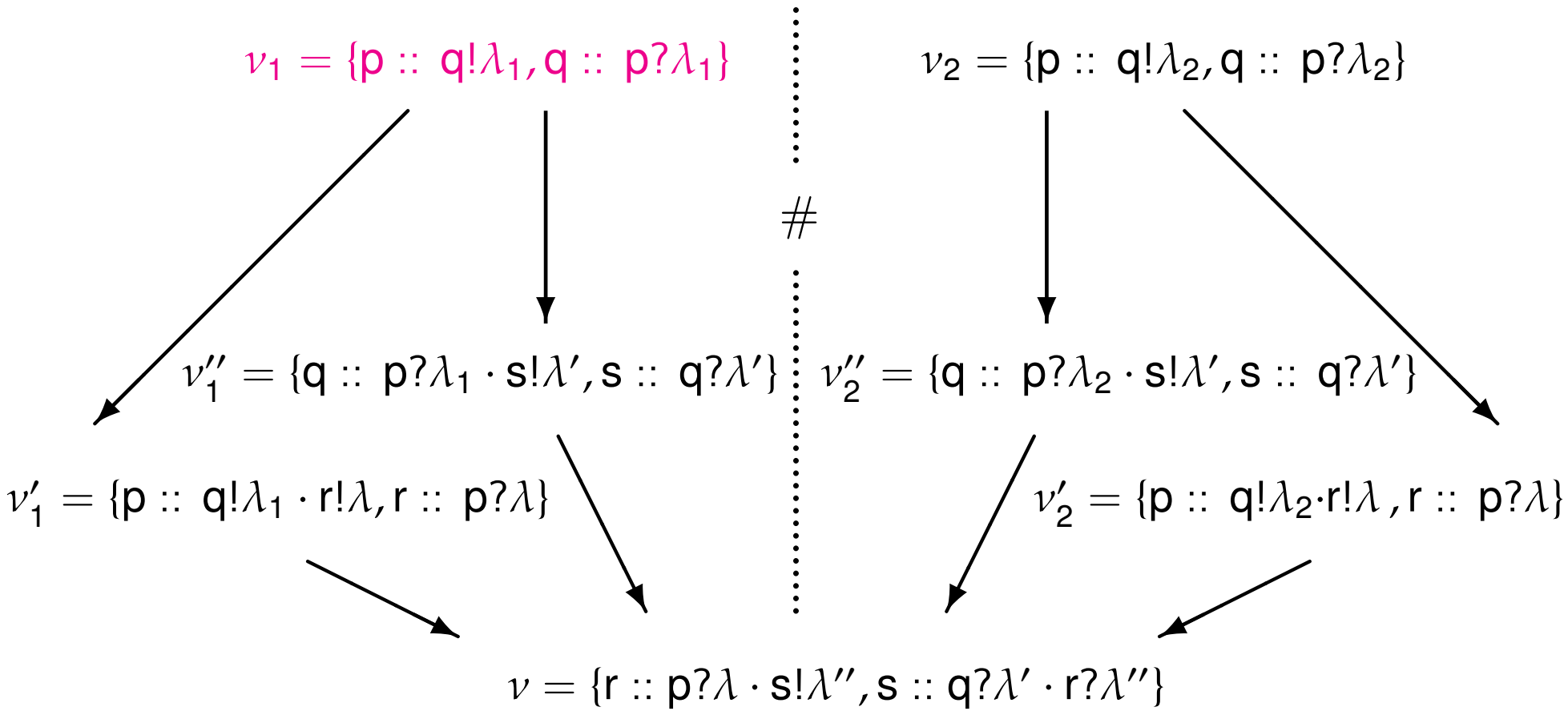
$\Rightarrow$ Pruning operation ("narrowing") on the set of events:

removes non causally well-founded n-events.

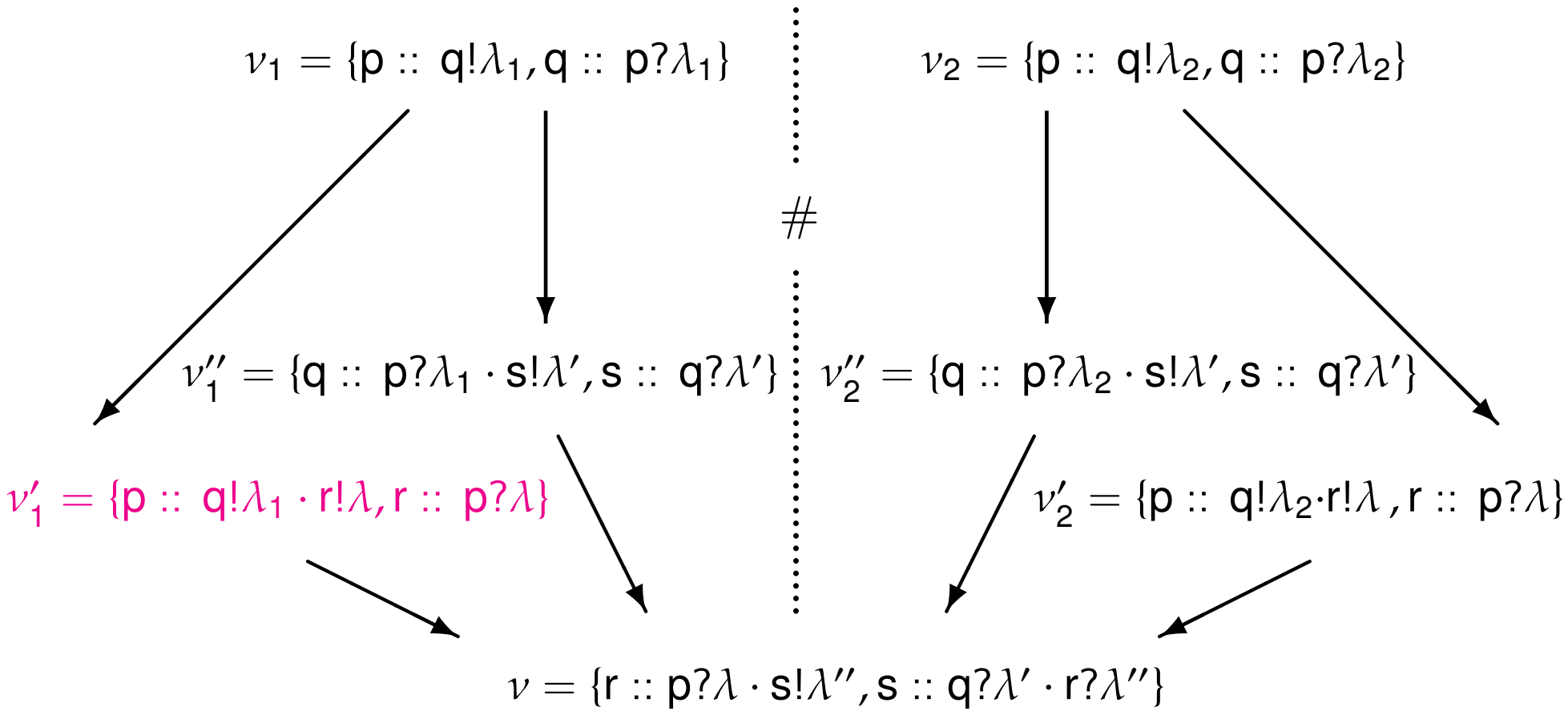
$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}!\lambda_1; \mathsf{r}!\lambda \oplus \mathsf{q}!\lambda_2; \mathsf{r}!\lambda \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda' + \mathsf{p}?\lambda_2; \mathsf{s}!\lambda' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda'' \,]\!] \parallel \mathsf{s}[\![\, \mathsf{q}?\lambda'; \mathsf{r}?\lambda'' \,]\!]$$



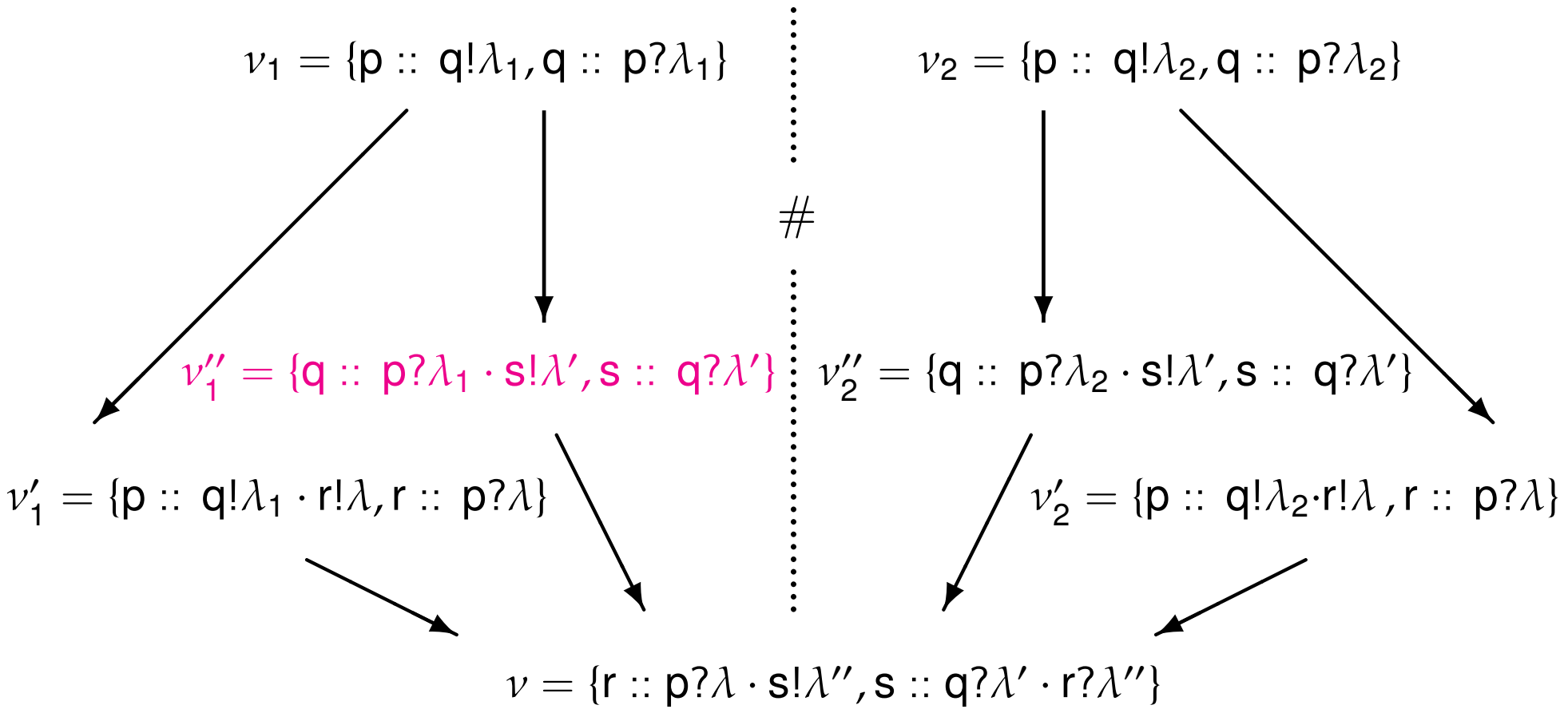
$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}!\lambda_1; \mathsf{r}!\lambda \oplus \mathsf{q}!\lambda_2; \mathsf{r}!\lambda \,]\!] \;\|\; \mathsf{q}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda' + \mathsf{p}?\lambda_2; \mathsf{s}!\lambda' \,]\!] \;\|\; \mathsf{r}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda'' \,]\!] \;\|\; \mathsf{s}[\![\, \mathsf{q}?\lambda'; \mathsf{r}?\lambda'' \,]\!]$$



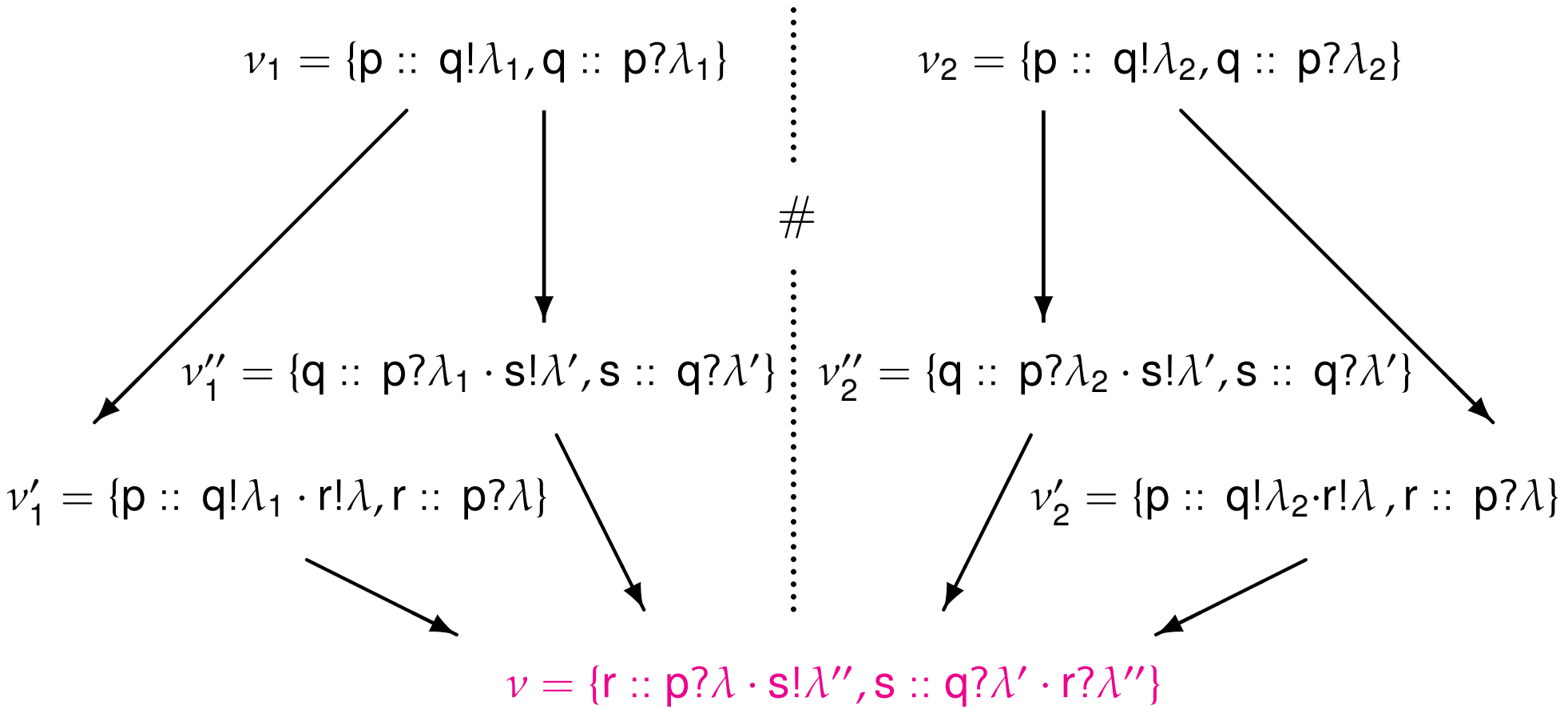
$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}!\lambda_1; \mathsf{r}!\lambda \oplus \mathsf{q}!\lambda_2; \mathsf{r}!\lambda \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda' + \mathsf{p}?\lambda_2; \mathsf{s}!\lambda' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda'' \,]\!] \parallel \mathsf{s}[\![\, \mathsf{q}?\lambda'; \mathsf{r}?\lambda'' \,]\!]$$



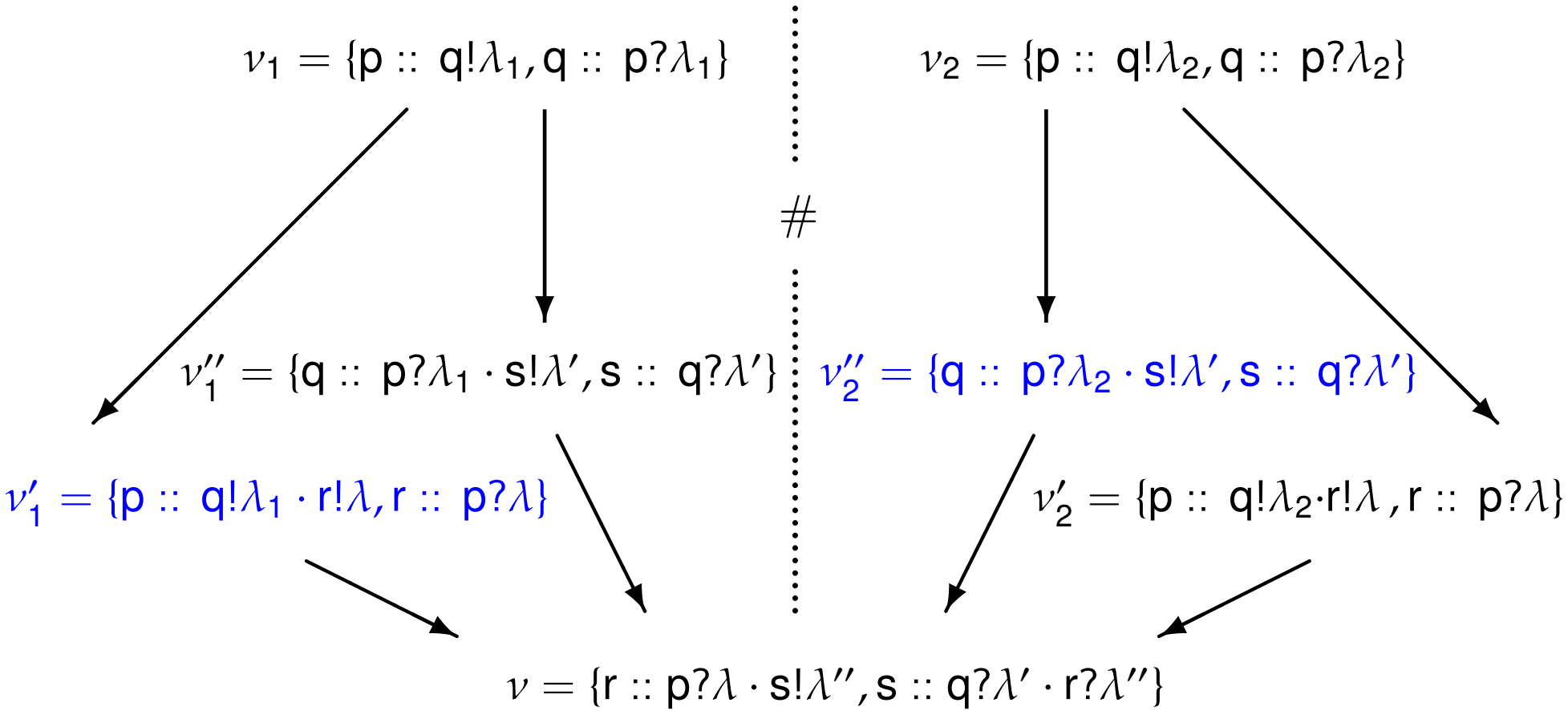
$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}!\lambda_1; \mathsf{r}!\lambda \oplus \mathsf{q}!\lambda_2; \mathsf{r}!\lambda \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda' + \mathsf{p}?\lambda_2; \mathsf{s}!\lambda' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda'' \,]\!] \parallel \mathsf{s}[\![\, \mathsf{q}?\lambda'; \mathsf{r}?\lambda'' \,]\!]$$



$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}!\lambda_1; \mathsf{r}!\lambda \oplus \mathsf{q}!\lambda_2; \mathsf{r}!\lambda \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda' + \mathsf{p}?\lambda_2; \mathsf{s}!\lambda' \,]\!] \parallel \mathsf{r}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda'' \,]\!] \parallel \mathsf{s}[\![\, \mathsf{q}?\lambda'; \mathsf{r}?\lambda'' \,]\!]$$



$$\mathsf{N} = \mathsf{p}[\![\, \mathsf{q}!\lambda_1; \mathsf{r}!\lambda \oplus \mathsf{q}!\lambda_2; \mathsf{r}!\lambda \,]\!] \;\|\; \mathsf{q}[\![\, \mathsf{p}?\lambda_1; \mathsf{s}!\lambda' + \mathsf{p}?\lambda_2; \mathsf{s}!\lambda' \,]\!] \;\|\; \mathsf{r}[\![\, \mathsf{p}?\lambda; \mathsf{s}!\lambda'' \,]\!] \;\|\; \mathsf{s}[\![\, \mathsf{q}?\lambda'; \mathsf{r}?\lambda'' \,]\!]$$



Conflict between n-events with disjoint participants.

Prime ES Semantics of Global Types: events

Trace σ: finite sequence of communications.

Permutation equivalence $\sim$ on traces:

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

Prime ES Semantics of Global Types: events

Trace σ: finite sequence of communications.

Permutation equivalence $\sim$ on traces:

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

A trace $\sigma = \alpha_1 \cdots \alpha_n$ is pointed if

$$\text{for all } i,\ 1 \leq i < n,\ \mathsf{part}(\alpha_i) \cap \mathsf{part}(\{\alpha_{i+1}, \ldots, \alpha_n\}) \neq \emptyset$$

Prime ES Semantics of Global Types: events

Trace σ: finite sequence of communications.

Permutation equivalence $\sim$ on traces:

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

A trace $\sigma = \alpha_1 \cdots \alpha_n$ is pointed if

$$\text{for all } i,\ 1 \leq i < n,\ \mathsf{part}(\alpha_i) \cap \mathsf{part}(\{\alpha_{i+1}, \ldots, \alpha_n\}) \neq \emptyset$$

Global type event (g-event): $\gamma = [\sigma]_\sim$ where σ is pointed.

Prime ES Semantics of Global Types: events

Trace σ: finite sequence of communications.

Permutation equivalence $\sim$ on traces:

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

A trace $\sigma = \alpha_1 \cdots \alpha_n$ is pointed if

$$\text{for all } i,\ 1 \leq i < n,\ \mathsf{part}(\alpha_i) \cap \mathsf{part}(\{\alpha_{i+1}, \ldots, \alpha_n\}) \neq \emptyset$$

Global type event (g-event): $\gamma = [\sigma]_\sim$ where σ is pointed.

$\gamma = [\sigma \cdot \alpha]_\sim$ represents an occurrence of communication α preceded by its global history σ (considered modulo $\sim$).

Prime ES Semantics of Global Types: relations

Global type event (g-event):

$$\gamma = [\sigma]_\sim \text{ where } \sigma \text{ is pointed.}$$

Causality relation

$\gamma \leq \gamma'$ if $\gamma = [\sigma]_\sim$ & $\gamma' = [\sigma \cdot \sigma']_\sim$ for some σ, σ'

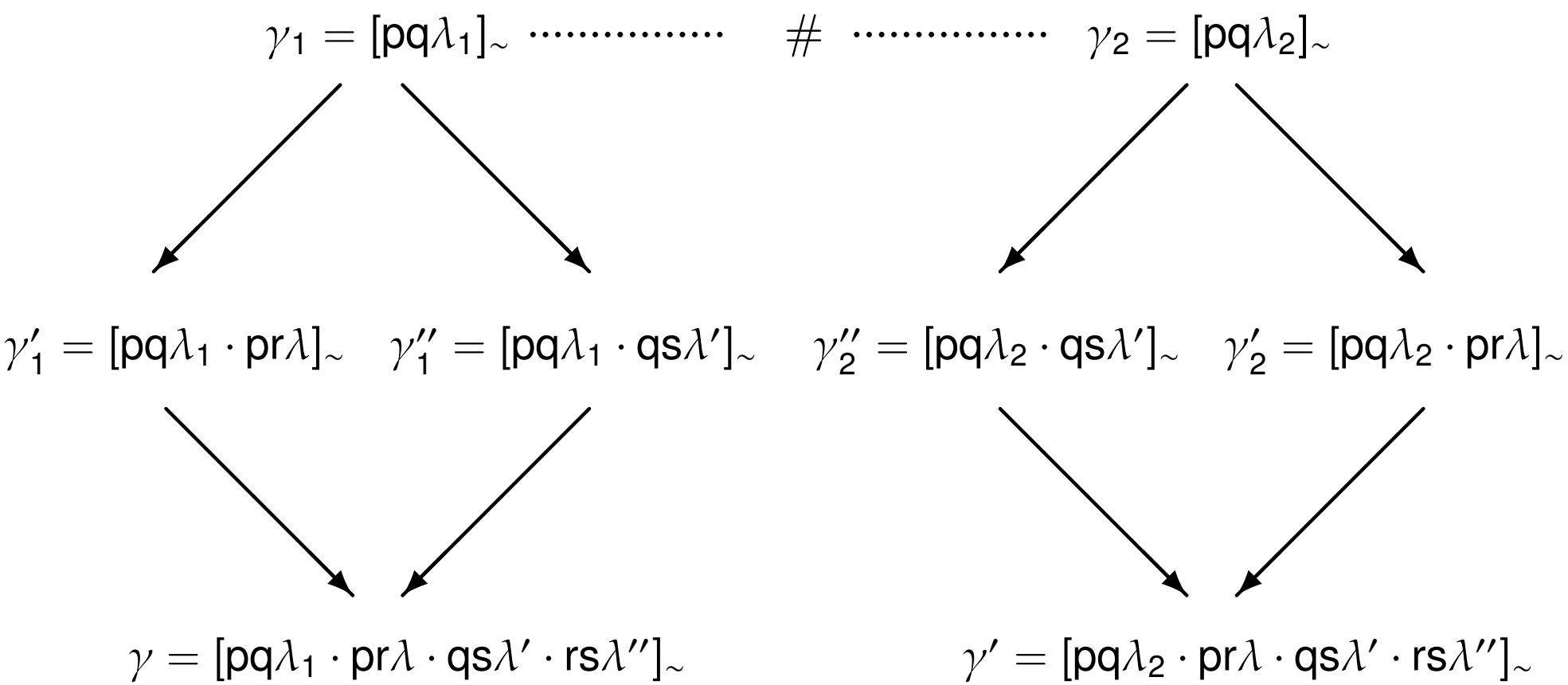
Conflict relation

$[\sigma]_\sim \# [\sigma']_\sim$ if $\sigma@p \ \# \ \sigma'@p$ for some p

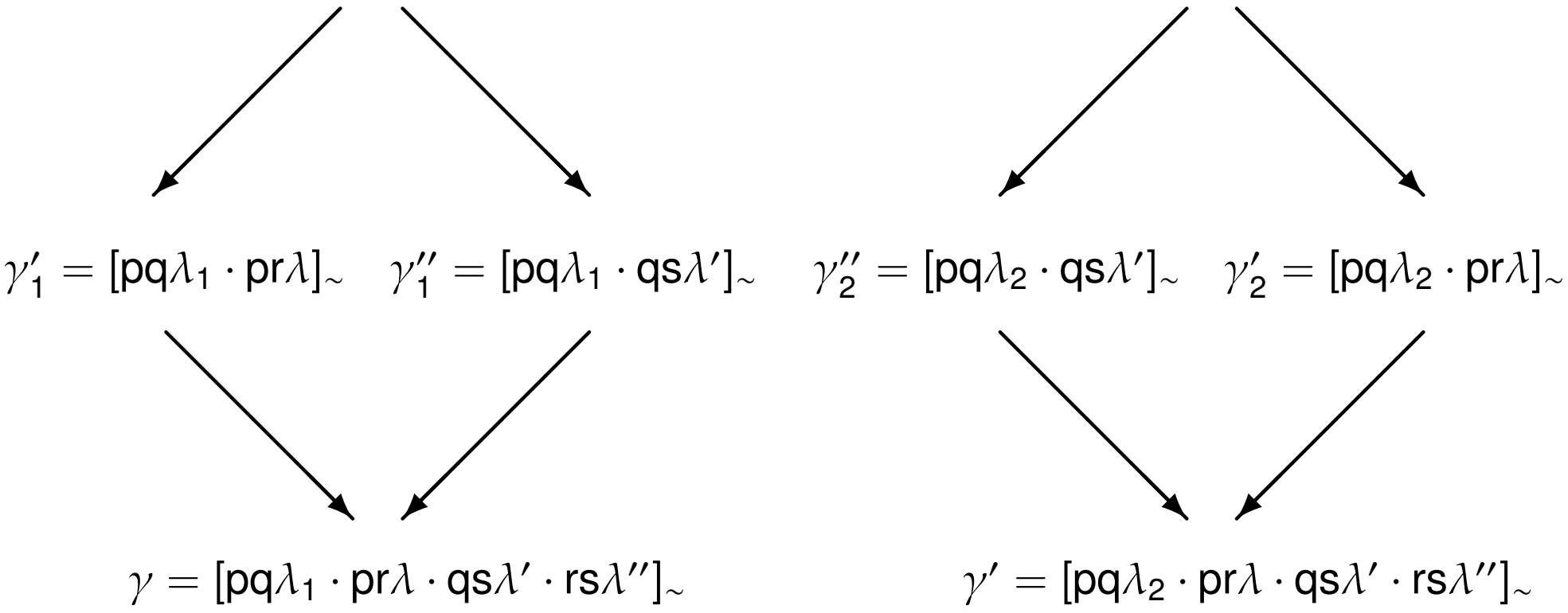
$$[pq\lambda \cdot rp\lambda_1 \cdot qp\lambda']_\sim \ \# \ [pq\lambda \cdot rp\lambda_2]_\sim \quad \text{because}$$

$$(pq\lambda \cdot rp\lambda_1 \cdot qp\lambda')@p = q!\lambda \cdot r?\lambda_1 \cdot q?\lambda' \ \# \ q!\lambda \cdot r?\lambda_2 = (pq\lambda \cdot rp\lambda_2)@p$$

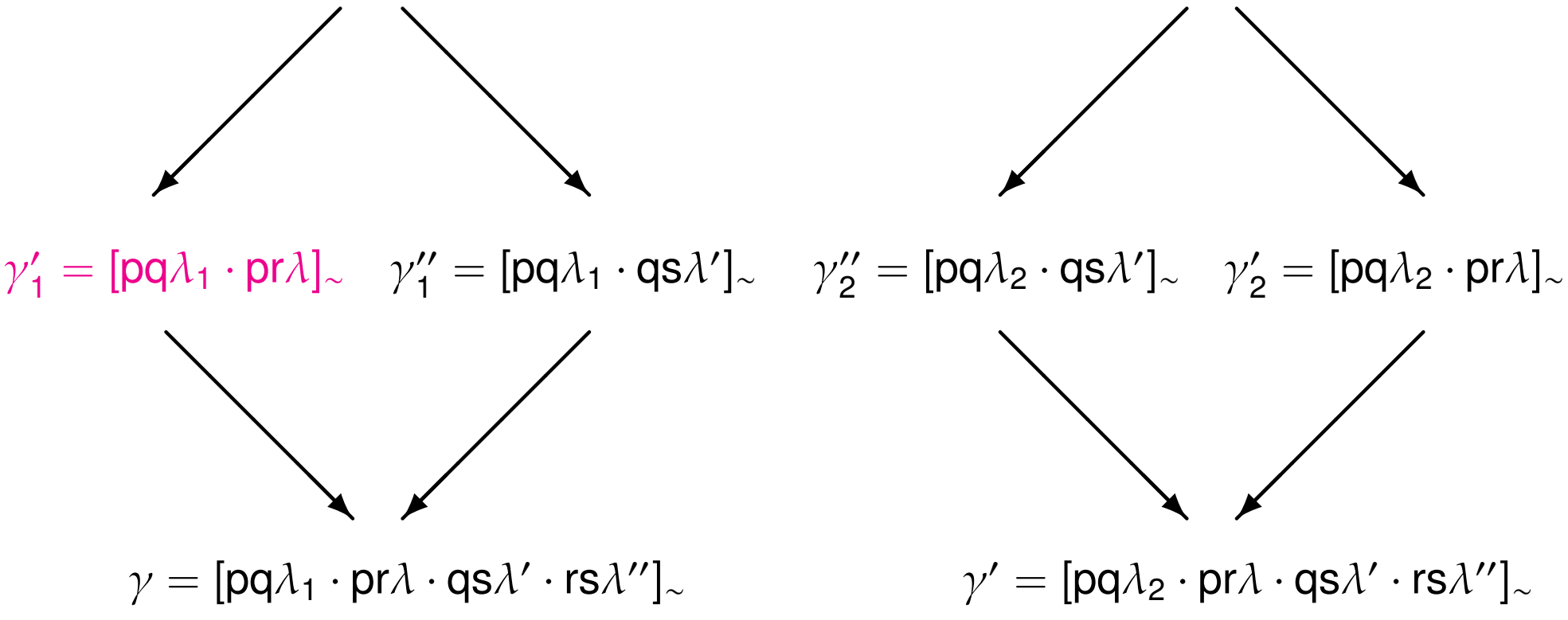
$$\mathsf{G} = \mathsf{p} \to \mathsf{q} : (\lambda_1; \mathsf{p} \xrightarrow{\lambda} \mathsf{r}; \mathsf{q} \xrightarrow{\lambda'} \mathsf{s}; \mathsf{r} \xrightarrow{\lambda''} \mathsf{s} \boxplus \lambda_2; \mathsf{p} \xrightarrow{\lambda} \mathsf{r}; \mathsf{q} \xrightarrow{\lambda'} \mathsf{s}; \mathsf{r} \xrightarrow{\lambda''} \mathsf{s})$$



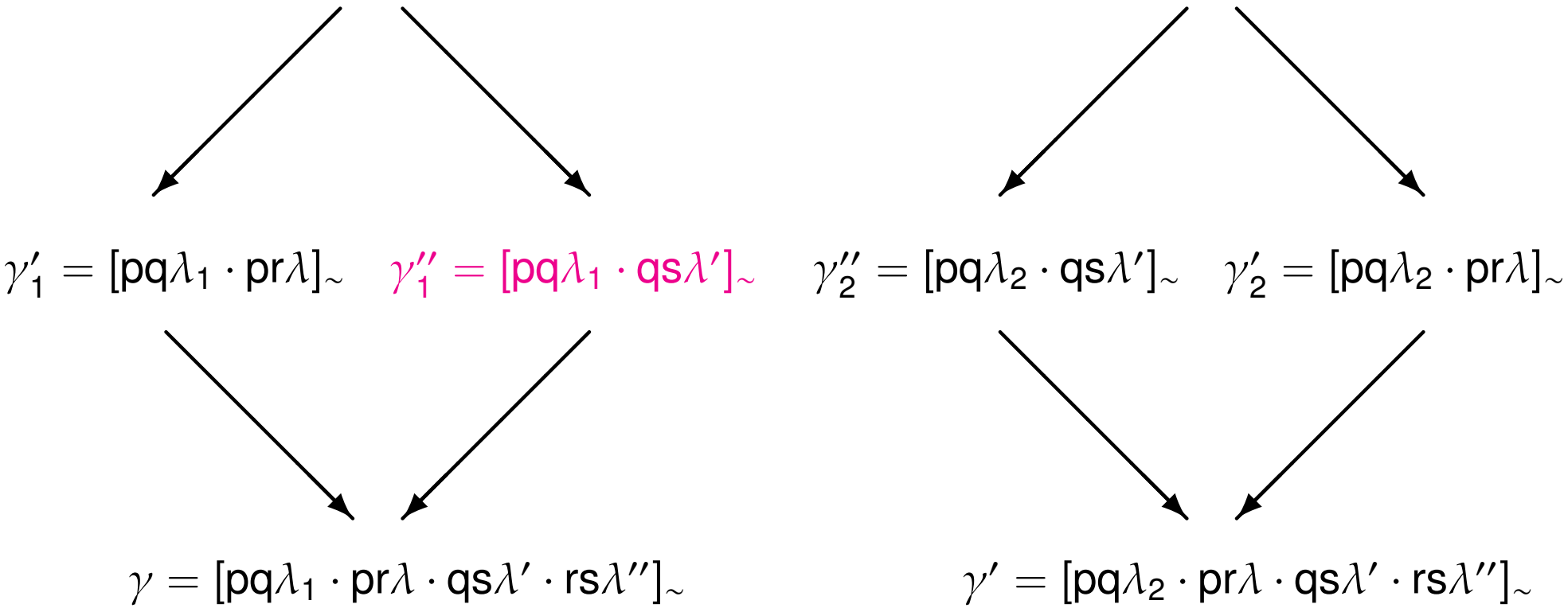
$$G = p \to q : (\lambda_1; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s \boxplus \lambda_2; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s)$$



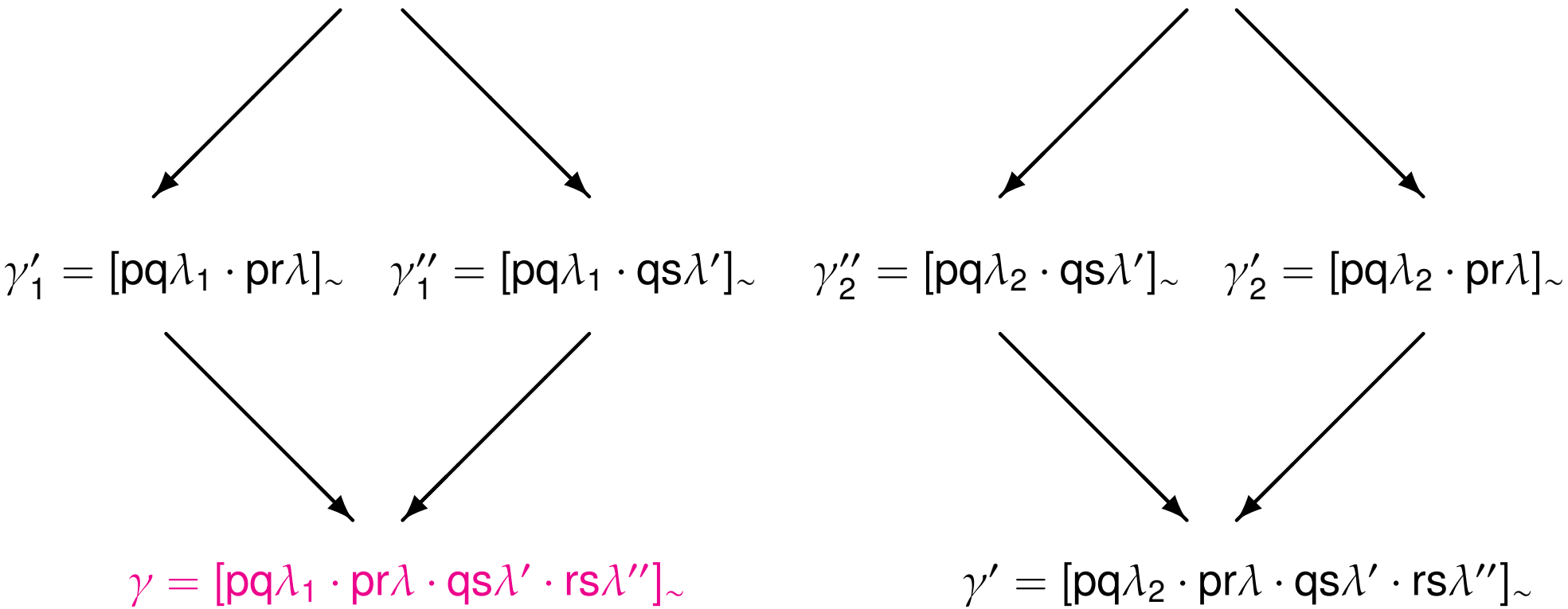
$$G = p \to q : (\lambda_1; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s \boxplus \lambda_2; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s)$$



$$G = p \to q : (\lambda_1; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s \boxplus \lambda_2; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s)$$



$$G = p \to q : (\lambda_1; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s \boxplus \lambda_2; p \xrightarrow{\lambda} r; q \xrightarrow{\lambda'} s; r \xrightarrow{\lambda''} s)$$



Relating a session FES with its type PES

Relating a session FES with its type PES

The FES of a session is not isomorphic to the PES of its type:

$$\nu = \{\mathsf{r} :: \mathsf{p}?\lambda \cdot \mathsf{s}!\lambda'', \mathsf{s} :: \mathsf{q}?\lambda' \cdot \mathsf{r}?\lambda''\}$$

The n-event ν has a single local history but two global histories.

Hence, there are two g-events corresponding to it:

$$\gamma = [\mathsf{pq}\lambda_1 \cdot \mathsf{pr}\lambda \cdot \mathsf{qs}\lambda' \cdot \mathsf{rs}\lambda'']_\sim \qquad \gamma' = [\mathsf{pq}\lambda_2 \cdot \mathsf{pr}\lambda \cdot \mathsf{qs}\lambda' \cdot \mathsf{rs}\lambda'']_\sim$$

However:

The FES of a session $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$ and the PES of its type $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$ yield isomorphic domains of configurations.

Configurations and proving sequences

Proving sequence of a FES $S = (E, \prec, \#)$: sequence $e_1; \cdots; e_n$ of distinct non-conflicting events such that

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

Proving sequence of a PES $S = (E, \leq, \#)$: sequence $e_1; \cdots; e_n$ of distinct non-conflicting events such that

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

Every configuration of a FES or a PES can be enumerated (possibly in several different ways) as a proving sequence.

Auxiliary functions

The function comm(·) extracts the communication from n-events and g-events:

$$\begin{aligned} \mathsf{comm}(\nu) &= \mathsf{pq}\lambda \text{ if } \nu = \{\mathsf{p} :: \zeta \cdot \mathsf{q}!\lambda, \mathsf{q} :: \zeta' \cdot \mathsf{p}?\lambda\} \\ \mathsf{comm}([\sigma]_\sim) &= \alpha \text{ if } \sigma = \sigma' \cdot \alpha \end{aligned}$$

The function nec(·) reconstructs a sequence of n-events from a sequence of communications

$$\mathsf{nec}(\alpha_1 \cdots \alpha_n) = \nu_1; \ldots; \nu_n \text{ where} \ldots$$

The function gec(·) reconstructs a sequence of g-events from a sequence of communications

$$\mathsf{gec}(\alpha_1 \cdots \alpha_n) = \gamma_1; \ldots; \gamma_n \text{ where} \ldots$$

Correspondence between network transition sequences and FES proving sequences

Transition sequences of N $\iff$ proving sequences of $\mathcal{S}^{\mathcal{N}}(\mathrm{N})$

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

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

Transition sequences of G $\iff$ proving sequences of $\mathcal{S}^{\mathcal{G}}(\mathrm{G})$

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

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

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash N : G$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$\vdash \mathsf{N} : \mathsf{G}$

$\nu_1 ; \ldots ; \nu_n$ proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$$\nu_1 ; \ldots ; \nu_n \qquad \text{proving sequence of } \mathcal{S}^{\mathcal{N}}(\mathsf{N})$$

$$\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n \qquad \text{where } \alpha_i = \mathsf{comm}(\nu_i)$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$$\nu_1 ; \ldots ; \nu_n \qquad \text{proving sequence of } \mathcal{S}^{\mathcal{N}}(\mathsf{N})$$

$$\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n \qquad \text{where } \alpha_i = \mathsf{comm}(\nu_i)$$

$$\Downarrow \; \mathit{SR}$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$\nu_1 ; \ldots ; \nu_n$ proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$

$\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n$ where $\alpha_i = \mathsf{comm}(\nu_i)$

$$\Downarrow \textit{ SR}$$

$$\mathsf{G} \xrightarrow{\alpha_1} \mathsf{G}_1 \cdots \xrightarrow{\alpha_n} \mathsf{G}_n$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$\nu_1 ; \ldots ; \nu_n$ proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$

$\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n$ where $\alpha_i = \mathsf{comm}(\nu_i)$

$$\Downarrow \textit{ SR}$$

$\mathsf{G} \xrightarrow{\alpha_1} \mathsf{G}_1 \cdots \xrightarrow{\alpha_n} \mathsf{G}_n$

$\mathsf{gec}(\alpha_1 \cdots \alpha_n) =$

$\gamma_1 ; \ldots ; \gamma_n$ proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash N : G$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$\vdash N : G$

$\gamma_1; \ldots; \gamma_n$ proving sequence of $\mathcal{S}^{\mathcal{G}}(G)$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$$\gamma_1 ; \ldots ; \gamma_n \qquad \text{proving sequence of } \mathcal{S}^{\mathcal{G}}(\mathsf{G})$$

$$\mathsf{G} \xrightarrow{\alpha_1} \mathsf{G}_1 \cdots \xrightarrow{\alpha_n} \mathsf{G}_n \qquad \text{where } \alpha_i = \mathsf{comm}(\gamma_i)$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$$\gamma_1 ; \ldots ; \gamma_n \qquad \text{proving sequence of } \mathcal{S}^{\mathcal{G}}(\mathsf{G})$$

$$\mathsf{G} \xrightarrow{\alpha_1} \mathsf{G}_1 \cdots \xrightarrow{\alpha_n} \mathsf{G}_n \qquad \text{where } \alpha_i = \mathsf{comm}(\gamma_i)$$

$$\Downarrow \; \textit{SF}$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$\gamma_1 ; \ldots ; \gamma_n$ proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$

$\mathsf{G} \xrightarrow{\alpha_1} \mathsf{G}_1 \cdots \xrightarrow{\alpha_n} \mathsf{G}_n$ where $\alpha_i = \mathsf{comm}(\gamma_i)$

$$\Downarrow \; \mathit{SF}$$

$$\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n$$

Isomorphism of configuration domains

Relies on Subject Reduction (SR) and Session Fidelity (SF).

$$\vdash \mathsf{N} : \mathsf{G}$$

$\gamma_1 ; \ldots ; \gamma_n$ proving sequence of $\mathcal{S}^{\mathcal{G}}(\mathsf{G})$

$\mathsf{G} \xrightarrow{\alpha_1} \mathsf{G}_1 \cdots \xrightarrow{\alpha_n} \mathsf{G}_n$ where $\alpha_i = \mathsf{comm}(\gamma_i)$

$\Downarrow$ *SF*

$\mathsf{N} \xrightarrow{\alpha_1} \mathsf{N}_1 \cdots \xrightarrow{\alpha_n} \mathsf{N}_n$

$\mathsf{nec}(\alpha_1 \cdots \alpha_n) \;=\; \nu_1 ; \ldots ; \nu_n$ proving sequence of $\mathcal{S}^{\mathcal{N}}(\mathsf{N})$

Conclusion and future/related work

Conclusion and future/related work

- ES semantics for synchronous sessions and types

Conclusion and future/related work

- ES semantics for synchronous sessions and types
- For typable sessions: equivalence of the two semantics (isomorphism of configuration domains).

Conclusion and future/related work

- ES semantics for synchronous sessions and types
- For typable sessions: equivalence of the two semantics (isomorphism of configuration domains).
- Future work: semantic counterpart for well-formedness conditions $\Rightarrow$ would enable synthesis of the FES of a network from the PES of its global type.
- Related work:
  - [TG18] E. Tuosto, R. Guanciale. Semantics of global view of choreographies. JLAMP 95, 2018.
  - [LMT22] U. de' Liguoro, Hernán C. Melgratti, E. Tuosto: Towards refinable choreographies. JLAMP. 127, 2022.

Conjectures / Open problems

Conjectures / Open problems

1. Do all well-formed global types for the same network yield the same PES? Conjecture: YES.

Conjectures / Open problems

1. Do all well-formed global types for the same network yield the same PES? Conjecture: YES.
2. Our core session calculus is a linear subcalculus of CCS. Does our FES semantics for networks coincide with that of [BC88] for CCS, which uses self-conflicts? Conjecture: YES.

Conjectures / Open problems

1. Do all well-formed global types for the same network yield the same PES? Conjecture: YES.
2. Our core session calculus is a linear subcalculus of CCS. Does our FES semantics for networks coincide with that of [BC88] for CCS, which uses self-conflicts? Conjecture: YES.
3. Do the results about prime decomposition obtained for CCS, ACP etc. by Faron Moller, Bas Luttik, Luca Aceto, Anna Ingolfsdottir and collaborators, apply also to our calculus? Are they simpler to prove on ESs ?

Conjectures / Open problems

1. Do all well-formed global types for the same network yield the same PES? Conjecture: YES.
2. Our core session calculus is a linear subcalculus of CCS. Does our FES semantics for networks coincide with that of [BC88] for CCS, which uses self-conflicts? Conjecture: YES.
3. Do the results about prime decomposition obtained for CCS, ACP etc. by Faron Moller, Bas Luttik, Luca Aceto, Anna Ingolfsdottir and collaborators, apply also to our calculus? Are they simpler to prove on ESs ?
4. Characterise the FESs/PESs obtained by interpreting sessions/global types (as done previously for fragments of CCS), or at least identify some of their structural properties.

Thank you