Event structures for mixed choice
Marc de Visme

To cite this version:
Marc de Visme. Event structures for mixed choice. The 30th International Conference on Concurrency Theory, CONCUR 2019, Aug 2019, Amsterdam, Netherlands. hal-02167420

HAL Id: hal-02167420
https://hal.archives-ouvertes.fr/hal-02167420
Submitted on 27 Jun 2019

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.
Event structures for mixed choice

Marc de Visme
Univ Lyon, EnsL, UCBL, CNRS, LIP, F-69342, LYON Cedex 07, France

Abstract
In the context of models with mixed nondeterministic and probabilistic choice, we present a concurrent model based on partial orders, more precisely Winskel’s event structures. We study its relationship with the interleaving-based model of Segala’s probabilistic automata. Lastly, we use this model to give a truly concurrent semantics to an extension of CCS with probabilistic choice, and relate this concurrent semantics to the usual interleaving semantics, thus generalising existing results on CCS, event structures and labelled transition systems.

2012 ACM Subject Classification Theory of computation → Parallel computing models

Keywords and phrases probability, nondeterminism, concurrency, event structures, CCS

Related Version Conference version published at CONCUR 2019

Funding Marc de Visme: Supported by Labex MiLyon (ANR-10-LABX-0070) of Université de Lyon, within the program “Investissements d’Avenir” (ANR-11-IDEX-0007), operated by the French National Research Agency (ANR).

1 Introduction
We study models of mixed choice [7], i.e. models representing both probabilistic choice and nondeterministic choice. The need for such models arises with systems where unconstrained nondeterministic behaviours coexist with quantified and controlled nondeterministic behaviours; for example, parallel threads using random number generators (hence probabilistic choices) while operating on a shared memory (hence nondeterministic races).

While many different models have been developed through the years, Segala’s probabilistic automata [13, 14] are a widely used model, both general and practical. They are automata where transitions are from states to probability distributions on states, hence modelling an alternation between a nondeterministic choice (the choice of the transition) and a probabilistic choice (the probability distribution). It is an interleaving model, as it represents “A and B occur in parallel” by “A then B or B then A”. In this paper, we are interested in models that are not interleaving, and represent “A and B occur in parallel” by “A and B are causally unrelated” instead. Those models are called truly concurrent models, and are particularly useful to study races, concurrency, and causality. We specifically focus on truly concurrent models based on partial orders, more precisely Winskel’s event structures [9, 11, 17]. We present here mixed probabilistic event structures, which are event structures enriched to model mixed nondeterministic and probabilistic choice. This work is in continuation of works on event structure models for languages with effects: parallelism [17], probabilities [5, 16], quantum effects [6], shared weak memory [4], …

In order to ensure that mixed probabilistic event structures are an adequate model for mixed choice, we show how to relate them to the existing model of Segala automata. Indeed, a mixed probabilistic event structure can be unfolded to a (tree-like) Segala automaton through a sequentialisation procedure, similar to the unfolding of a partial order into a tree. This sequentialisation procedure is well-behaved; rather than listing all the ad-hoc properties it satisfies, we express mixed probabilistic event structures as a category, such that sequentialisation forms an adjunction, from which those properties can be deduced. Sections 3 and 4 are dedicated to the development of this new model for mixed choice.
We apply this model to give a semantics to a language featuring parallelism, nondeterminism and probabilistic choice. Namely, we choose an extension of CCS [10] with probabilistic choice [2]. Extensions of our work to more complex languages, such as the probabilistic \( \pi \)-calculus, should be possible using methods similar to [16]. In that paper, Varacca and Yoshida give a concurrent semantics using event structures to a “deterministic” probabilistic \( \pi \)-calculus, i.e. they forbid processes with a nondeterministic behaviour, like \((a|a|\overline{a})\) where two inputs race to interact with an output. We do not have those restrictions, and fully support both nondeterministic and probabilistic behaviours.

Using partial-order based models to represent parallel processes is not a new idea. Indeed, the relationship between CCS and event structures is well-studied [17], and recalled in detail in Section 2. The core result of this relationship is the factorisation theorem (Theorem 9), which states that if one considers the event structure representing the concurrent semantics of a process, and unfolds it into a labelled tree, then the result would match the interleaving semantics of the process. It follows that the concurrent semantics is sound w.r.t. the interleaving semantics. In Section 4, we lift the known relations between CCS, event structures, and labelled trees, to relations between Probabilistic CCS, mixed probabilistic event structures, and (tree-like) Segala automata; concluding with a factorisation theorem (Theorem 26).

2 Preliminaries

The process algebra CCS [10] is used to represent concurrent processes communicating with each other through channels. A process \( P \) can evolve into a process \( Q \) by performing an action. The graph having processes as nodes and transitions labelled by actions as edges is known as the Labelled Transition System (LTS) of the process language. From this graph and any process \( P \), one can work with the (possibly infinite) labelled tree obtained by unfolding the graph starting by the node \( P \). We may express the semantics of a process \( P \) in terms of that unfolded tree, the approach we take in this paper.

The LTS of processes yield an interleaving semantics, since they do not distinguish the process \((a|b)\) that perform \( a \) and \( b \) in parallel from the process \((a.b \otimes b.a)\) – where \( \otimes \) represents nondeterministic choice – which can perform \( a \) and \( b \) in any sequential order. Semantics that do distinguish between the two actions in parallel and the two actions in any sequential order are called truly concurrent semantics. In this paper, we will consider the usual truly concurrent semantics of CCS: event structures [9, 11, 17], which are partial orders with a notion of conflict (Definition 1).

The semantics of CCS in labelled trees and in event structures are both quite straightforward, except for the parallel composition \( (P|Q) \), which is syntactically complex to compute. It is often practical to use a more abstract approach than syntactically computing \( (P|Q) \), and remark that the parallel composition of CCS arises in both semantics [17] as a categorical product \( \times \), (for a suitable notion of maps) followed by a restriction.

As stated before, we can recover the interleaving semantics from the truly concurrent semantics. This means that we can unfold event structures as labelled trees, while preserving the interpretation of CCS processes. This unfolding consists in finding the best labelled tree to approximate the behaviour of a given event structure.

In the literature of models of concurrency, following Winskel [12, 17], such unfoldings are often expressed through a categorical coreflection – a special case of adjunction. The unfolding of an event structure into a labelled tree, named sequentialisation, is the right adjoint of this adjunction, ensuring that the sequentialisation \( \text{Seq}(E) \) of an event structure \( E \)
(A prime) event structure [18] is a representation of computational events of a concurrent system. Its events are related by a partial order representing causal dependency: if \( a \leq b \) then \( b \) can only occur if \( a \) has occurred beforehand; and a conflict relation representing incompatibility: if \( a \# b \) then each of \( a \) and \( b \) can only occur if the other does not.

**Definition 1.** An event structure \( E \) is a triple \( E = (|E|, \leq_E, \#_E) \) where:

- \(|E|\) is a set whose elements are called events.
- \( \leq_E \) is a (partial) order, called causality, such that \( \{ b \ | \ b \leq_E a \} \) is finite for all \( a \in |E| \).
- \( \#_E \) is a symmetric irreflexive binary relation, called conflict.
- \( \forall a,b,c \in |E|, \) if \( a \#_E b \leq_E c \) then \( a \#_E c \).

In such a structure, we want to describe the set of states in which the system under study can exist. We define the set of (finite) configurations of \( E \), written \( C(E) \), as the set of reachable finite sets of events, in other words:

\[
x \in C(E) \iff x \in T_{\text{fin}}(|E|) \text{ and } \forall a \in x, \forall b \in |E|, \begin{cases} b \#_E a \implies b \notin x \text{ and } \\ b \leq_E a \implies b \in x \end{cases}
\]

We say that a configuration \( x \) enables an event \( a \notin x \), and we write \( x - a \subseteq \), if \( x \cup \{a\} \) is a configuration. There are two canonical configurations associated to an event \( a \in |E| \): the minimal configuration containing it \( [a] := \{ b \ | \ b \leq_E a \} \), and the minimal configuration enabling it \( [a] := [a] \setminus \{a\} \).

The causality \( \leq_E \) and the conflict \( \#_E \) contain a lot of redundant information: if you know that \( a \#_E b \) and \( b \leq_E c \), then the definition of event structures enforces \( a \#_E c \). When representing event structures, we want a concise representation, so instead of \( \leq_E \) and \( \#_E \), we use immediate causality \( \rightarrow_E \) (represented by arrows) and minimal conflict \( \sim_E \) (represented by wiggly lines), defined as follows:

\[
a \rightarrow_E b \iff \begin{cases} a \leq_E b \\ \forall a \leq_E c \leq_E b, c \in \{a,b\} \end{cases}
\]

\[
a \sim_E b \iff \begin{cases} a \#_E b \\ \forall a' <_E a, \neg(a' \#_E b) \\ \forall b' <_E b, \neg(a \#_E b') \end{cases}
\]

In Figure 1, we deduce \( c \#_E e \) and \( a \leq_E e \) from the minimal conflict and immediate causality.

There is a notion of (partial or total) maps of event structures, altogether forming a category. Formally, a (partial or total) map \( f \) from \( E \) to \( E' \) is a (partial or total) function \( f : |E| \rightarrow |E'| \) such that:

---

We choose to use binary conflicts for pedagogical reasons. Our work can be extended to non-binary conflicts as in [19].
Event structures for mixed choice

\begin{itemize}
  \item \textbf{Figure 1} An event structure \( E \)
  \[ a \xleftarrow{b} \sim \xrightarrow{c} \]
  \[ \downarrow \]
  \[ \downarrow \]
  \[ d \]
  \[ e \]
  \[ C(E) = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{a,b,d\}, \{a,b,d,e\} \} \]
  \end{itemize}

\begin{itemize}
  \item \textbf{Figure 2} A total map of event structures
  \[ a_1 \sim a_2 \]
  \[ b_1 \sim b_2 \]
  \[ c_1 \xrightarrow{b_1,b_2 \sim b} \]
  \[ c_2 \]
  \[ \downarrow \]
  \[ \downarrow \]
  \[ a_1 \quad a_2 \]
  \[ b \quad c_1 \]
  \[ \downarrow \]
  \[ \downarrow \]
  \[ b_1 \quad a_2 \]
  \[ c_2 \]
  \[ d \]
  \[ configuration preserving \text{ for all } x \in C(E) \text{ we have } f(x) \in C(E') \].

local injectivity \text{ for all } a, b \text{ distinct in } x \in C(E), \text{ if } f \text{ is defined on both then } f(a) \neq f(b).

Note that total maps can be interpreted as a form of simulation: if \( x \in C(E) \) enables an event \( e \notin x \), then \( f(x) \in C(E') \) enables \( f(e) \notin f(x) \). In the example of Figure 2, the map is total. It merges \( b_1 \) and \( b_2 \) into a single event \( b \), which is allowed since they are in conflict, and it does not contain the event \( d \) in its image, which is allowed because \( d \) is not causally required by any event in the image of the map.

With this notion of maps, we recall the induced notion of isomorphism: \( E \cong E' \) if and only if there is a total bijective map \( f : E \rightarrow E' \) with \( f^{-1} \) also a map of event structures. In other words, two event structures are isomorphic if and only if they are the same up to renaming of the events.

\begin{itemize}
  \item \textbf{Proposition 2.} Event structures and (partial) maps of event structures form a category, written \( \mathbf{ES} \). We write \( \emptyset \) for the empty event structure, which is a terminal object\(^2\). \( \mathbf{ES} \) has a subcategory \( \mathbf{ES}_t \) of event structures with total maps of event structures.
\end{itemize}

Event structures are used to represent causality and independence, but they can also be used to represent interleavings. However, concurrent systems and sequential systems simulating concurrency through interleaving will be represented in drastically different ways in event structures. Figure 3 shows a concurrent system able to perform two events \( a \) and \( b \) in parallel, and its interleaving counterpart where \( a \) and \( b \) can both happen in any order but not simultaneously. We can characterise event structures that are fully interleaved, in other words sequential event structures, as follows:

\begin{itemize}
  \item \textbf{Definition 3.} An event structure \( E \) is sequential if it satisfies one of the following equivalent properties:
    \begin{itemize}
        \item \( E \) is forest-shaped, with branches being in conflict with each other.
        \item For every \( a, b \in x \in C(E) \), either \( a \leq_E b \) or \( b \leq_E a \).
        \item There exists a (necessarily unique) total map from \( E \) to \( N' \), where \( N = (\mathbb{N}, \leq, \emptyset) \) is the event structure with an infinite succession of events.
        \item \( E \) is isomorphic to \( E \times N' \), where \( \times \) is the synchronous product defined in Section 2.2.
    \end{itemize}
\end{itemize}

\(^2\) i.e. for every object \( A \), there exists a unique map from \( A \) to \( \emptyset \).

\begin{itemize}
  \item \textbf{Figure 3} Concurrent system, and its interleaving counterpart.
    \[ a \quad b \]
    \[ a_1 \sim a_2 \]
    \[ b_1 \quad a_2 \]
  \end{itemize}
The first characterisation means that trees are in a one-to-one correspondence with sequential event structures. Through this correspondence, transitions correspond to events. For the remainder of this paper, we use sequential event structures to represent trees.

Proposition 4. Sequential event structures and (partial) maps of event structures form a subcategory of $\text{ES}_*$, written $\text{Seq-ES}_*$. It has a subcategory $\text{Seq-ES}$ of sequential event structures with total maps of event structures, with $\mathcal{N}$ for terminal object.

2.2 The Categorical Product

In this paper, we exclusively rely on the universal property of categorical products, without giving any concrete definition for such products. We write $\times_*$ for the product in $\text{ES}_*$ [17], called asynchronous product, and $\times$ for the product in $\text{ES}$ [17], called synchronous product.

Informally speaking, both $A \times_* B$ and $A \times B$ explore all the ways to pair up the events of $A$ with the events of $B$ while respecting causal dependencies and conflicts. The synchronous product expects all the events to be paired, while the asynchronous product allows events to remain unpaired, see Figure 4. Note that events of the products are not uniquely determined by their projections: in Figure 4 both $(\star, b_2)$ and $(\star, b_2)'$ have the same projections.

Proposition 5. $\text{ES}_*$ is a cartesian category, with $\times_*$ as a product and $\emptyset$ as a unit. $\text{ES}$ has a product $\times$, and its subcategory $\text{Seq-ES}$ is a cartesian category with $\mathcal{N}$ as unit.

Using the synchronous product $\times$, we can simply define the sequentialisation as $\text{Seq}(E) := E \times \mathcal{N}$. It is always a sequential event structure. The asynchronous product $\times_*$ will later be used to define the semantics of the parallel composition of CCS.

Theorem 6. $\text{Seq}$ is a functor from $\text{ES}_*$ (and ES) to $\text{Seq-ES}_*$ (and Seq-ES), which is right adjoint to the inclusion functor, forming coreflections [17].

Since right adjoints preserve categorical limits, we have that $\text{Seq}(A \times_* B) \cong \text{Seq}(A) \times_* \text{Seq}(B)$. Since the restriction (Definition 8) is also preserved by $\text{Seq}$, it means that when interpreting the parallel composition of CCS using the asynchronous product in $\text{ES}_*$, then sequentialising, we get the same result as when interpreting the parallel composition directly as the asynchronous product in $\text{Seq-ES}_*$ (i.e. trees).

2.3 Semantics of CCS

We consider the process algebra CCS [1, 3, 10] over a set of channels $\text{Chan}$. A channel $a \in \text{Chan}$ can be used by processes as an input $a$ or as an output $\overline{a}$. For convenience, we
Event structures for mixed choice

assume that $\bar{a} = a$. On top of these external actions, processes can also perform internal actions, written $\tau$, and we write $\mathbb{A} = \text{Chan} \cup \{\bar{a} \mid a \in \text{Chan}\} \cup \{\tau\}$ for the set of all actions.

Definition 7. The set $\mathbb{P}$ of processes is defined by the following syntax:

$$P ::= 0 \mid (P_1 \downarrow P_2) \mid (P_1 \odot P_2) \mid X \mid \mu X.P \mid a.P \mid \bar{a}.P \mid \tau.P \mid va.P \ (\text{for} \ a \in \text{Chan})$$

where the constructs are empty process, parallel composition, nondeterministic choice, process variable, process recursion, input prefix, output prefix and $\tau$ prefix, and restriction, respectively. We only consider closed processes, in which every variable is bound by a recursion $\mu X$.

Figure 5a describes the interleaving semantics of the language. The states of the LTS reachable from a process $P$ can be unfolded into a (potentially infinite) labelled tree represented using a (potentially infinite) sequential event structure, written $[P]_{\text{Is}}$, with labels in $\mathbb{A}$.

Definition 8. An event structure with labels in $\mathbb{L}$ is an event structure $E$ together with a total function $\ell : |E| \to \mathbb{L}$. For $L \subseteq \mathbb{L}$, we define $E \setminus L$ as the restriction of $E$ to the set of events $\{e \mid \forall e' \in E, \ell(e') \notin L\}$. In other words, we remove every event with a label in $L$, and every event causally dependent on it. Maps of labelled event structures are required to preserve labels.

We define the labels on $\text{Seq}(E)$ from the labels on $E$ as $\ell_{\text{Seq}}(E)(e) := \ell_E(\pi_1(e))$. We define the labels of $A \times B$ using the following synchronisation operation $\star : \mathbb{A} \cup \{\star\} \times \mathbb{A} \cup \{\star\} \to \mathbb{A} \cup \{0\}$, where $\star$ stands for “undefined”: $\ell_{A \times B}(e) = \ell_A(\pi_1(e)) \star \ell_B(\pi_2(e))$

$$a \star \bar{a} = \tau \quad a \star a = \tau \quad \bar{a} \star a = \tau \quad a \star \alpha = \alpha \quad \bar{a} \star \alpha = \alpha \quad \alpha \star \alpha = \alpha \quad \alpha \star 0 = 0 \quad \alpha \star \beta = 0 \ \text{otherwise}$$

Figure 5b describes the concurrent semantics of the language. For any process $P$, we write $[P]_{\text{cs}}$ for the (potentially non-sequential, potentially infinite) event structure with labels in $\mathbb{A}$ representing $P$. See Figure 6 for an example. On top of the previous operations, we need some additional operations on labelled event structures:

- To represent nondeterministic choice, we need to put two event structures in parallel, while allowing only one of them to be used. For $A$ and $B$ two event structures, we define $A \# B$ as the event structure with $|A| \uplus |B|$ as events, with the same order, conflict and labels as in $A$ and in $B$, but with every event of $A$ being in conflict with every event of $B$.

- To represent prefixes, we need to create a new event. We write $\downarrow e, a$ for the event structure with events $|E| \uplus \{e\}$, with $e$ labelled by $a$ being the minimal event for $\leq_{\downarrow e, a}$, everything else being the same as in $E$.

- To represent process recursion, we use a least fixpoint. The order used for that least fixpoint is given by “substructure maps”, i.e. total maps that are an inclusion on events, and preserve and reflect order and conflicts.

As claimed before, we can recover the interleaving semantics from the truly concurrent one. The proof of this theorem relies on the fact that $\text{Seq}$ forms a coreflection.

Theorem 9 (Factorisation [17]). For any process $P$ of CCS, we have $[P]_{\text{Is}} \cong \text{Seq}(\lfloor P \rfloor_{\text{cs}})$.

2.4 Probabilistic CCS

The goal of this paper is to extend existing results on CCS to Probabilistic CCS, which is CCS enriched with a probabilistic choice $\sum_{i \in I} p_i \cdot a_i.P_i$, with $I$ a possibly infinite set.

---

Some papers [2] use a less general sum where the $a_i$ are assumed to be equal.
We now develop the main contribution of this paper: an event structure model able to represent mixed internal and external choices, that we will use in the last section to give a concurrent semantics.

### Figure 5
Semantics of CCS, with \( a \in A \) and \( c \in Chan \)

(a) Interleaving semantics, using LTS

(b) Concurrent semantics, using event structures

\[
\begin{align*}
\overrightarrow{a.P} & \rightarrow \overrightarrow{P} \\
\text{if } a \notin \{c, \bar{c}\} & \text{ then } \overrightarrow{P} \rightarrow \overrightarrow{P} \\
\nu c & \overrightarrow{P} \rightarrow \nu c \overrightarrow{P} \\
\overrightarrow{P} \rightarrow \overrightarrow{P'} & \text{ if } P \rightarrow \overrightarrow{P'} \\
\nu c \overrightarrow{P} \overrightarrow{Q} & \rightarrow \nu c \overrightarrow{P} \overrightarrow{Q} \\
\overrightarrow{(P \parallel Q)} & \rightarrow \overrightarrow{P'} \\
\overrightarrow{P} \rightarrow \overrightarrow{P'} & \text{ if } P \rightarrow \overrightarrow{P'} \\
\overrightarrow{(Q \parallel P)} & \rightarrow \overrightarrow{P'} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{P'} \\
\overrightarrow{\mu X.P} & \rightarrow \overrightarrow{Q} \\
\nu c \overrightarrow{P \parallel Q} & \rightarrow \nu c \overrightarrow{P \parallel Q} \\
\nu c \overrightarrow{P} \overrightarrow{Q} & \rightarrow \nu c \overrightarrow{P} \overrightarrow{Q} \\
\overrightarrow{(P \parallel Q) \parallel (P' \parallel Q')} & \rightarrow \overrightarrow{(P \parallel Q) \parallel (P' \parallel Q')} \\
\nu c \overrightarrow{P \parallel Q} \overrightarrow{(\mu X.P \parallel Q)} & \rightarrow \nu c \overrightarrow{P \parallel Q} \overrightarrow{(\mu X.P \parallel Q)} \\
\overrightarrow{\mu X.P} & \rightarrow \overrightarrow{Q} \\
\nu c \overrightarrow{\mu X.P} & \rightarrow \nu c \overrightarrow{(\mu X.P \parallel Q)}
\end{align*}
\]

### Figure 6

(a) A CCS process

\[
u c. (a. (c.a \oplus c.b) | b. \bar{c})
\]

(b) Its interleaving semantics: a labelled tree

(c) Its concurrent semantics: a labelled event structure

\[\begin{align*}
\forall i \in I, 0 < p_i \leq 1, a_i \in A, \text{ and } \sum_{i \in I} p_i & \leq 1. \text{ We also assume the pairs } (a_i, P_i) \text{ to be pairwise distinct. Notice that we allow sub-probabilistic sums like } \frac{1}{2} \cdot a.P, \text{ but we forbid unguarded sums like } \frac{1}{2} \cdot (P_i \parallel P_j) + \frac{1}{2} \cdot Q. \text{ This guard restriction is similar to that found in the probabilistic } \pi \text{-calculus \cite{8, 16}. The interleaving semantics we present here uses Segala automata \cite{13, 14}, and relies on this absence of unguarded sums to be well-defined. In this semantics, reductions are interpreted by an alternation of probabilistic choices and nondeterministic choices. Mathematically, the reduction } \rightarrow \text{ is a subset of } \mathbb{P} \times D(A \times \mathbb{P}), \text{ where } D(U) \text{ is the set of discrete sub-probability distributions over } U. \text{ So for every process } P, \text{ there is first a nondeterministic choice of } P \rightarrow S, \text{ and then a probabilistic choice inside } S = \{(p_i, a_i, P_i) \mid i \in I\} \text{ of the action } a_i \text{ and the reduced process } P_i. \text{ Figure 7 describes this interleaving semantics.}
\]

### 3 Mixed Event Structures

We now develop the main contribution of this paper: an event structure model able to represent mixed internal and external choices, that we will use in the last section to give a concurrent semantics to the mixed probabilistic and nondeterministic choices of PCCS.

### Figure 7
Interleaving semantics of PCCS, using Segala automata

\[
\begin{align*}
\overrightarrow{a.P} & \rightarrow \overrightarrow{(\langle 1, a.P \rangle)} \\
\overrightarrow{\mu X.P} \rightarrow \overrightarrow{S} \\
\overrightarrow{P} & \rightarrow \overrightarrow{(\langle p_i, a_i, P_i \rangle) | i \in I} \\
\overrightarrow{P} \rightarrow \overrightarrow{S} & \rightarrow \overrightarrow{Q} \rightarrow \overrightarrow{S} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle p_i, a_i, P_i \rangle | i \in I \mid a_i \notin \{c, \bar{c}\})} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle q_j, b_j, Q_j \rangle | j \in J)} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle p_i, q_j, P_i \parallel Q_j \rangle | i \in I, j \in J, a_i = b_j)} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle p_i, a_i, P_i \rangle | i \in I \mid a_i \notin \{c, \bar{c}\})} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle q_j, b_j, Q_j \rangle | j \in J)} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle p_i, q_j, P_i \parallel Q_j \rangle | i \in I, j \in J, a_i = b_j)} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle p_i, a_i, P_i \rangle | i \in I)} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle q_j, b_j, Q_j \rangle | j \in J)} \\
\overrightarrow{P \parallel Q} & \rightarrow \overrightarrow{(\langle p_i, q_j, P_i \parallel Q_j \rangle | i \in I, j \in J, a_i = b_j)}
\end{align*}
\]
3.1 Definition

When trying to represent processes of PCCS in event structures, one could think of simply using the existing probabilistic event structures [20], which are event structures \( E \) together with a valuation \( v_E : C(E) \to (0, 1) \) satisfying some well-chosen properties (Definition 21). However, looking at the two processes \((\frac{1}{2}a + \frac{1}{2}b)\) and \((\frac{1}{2}a) \oplus (\frac{1}{2}b)\), we remark that:

- They have only two different computational events: “receiving on \( a \)” and “receiving on \( b \)”; which means that natural representations using event structures will only have two events.
- Those two computational events cannot occur together, so natural representations using event structures have those two events in conflict.
- Those two computational events are associated with probability half, which means that the valuation is necessarily \( v(\{a\}) = \frac{1}{2} = v(\{b\}) \)

In other words, if we try to represent them using probabilistic event structures, both of them will be represented by the same structure. But they have very different interleaving semantics in Segala automata, so it would not be a sound representation. A similar example using full probabilistic distributions is \((\frac{1}{2}a + \frac{1}{2}b) \oplus (\frac{1}{2}c + \frac{1}{2}d)\) and \((\frac{1}{2}a + \frac{1}{2}c) \oplus (\frac{1}{2}b + \frac{1}{2}d)\).

In order to distinguish these two processes, the solution we propose is to have two kinds of conflicts: an external conflict, used for the nondeterministic choice \( \oplus \), and an internal conflict, used for the probabilistic choice \(+\). In this section, we explore in more detail event structures with two kinds of conflicts, named mixed event structures.

We will keep using the notation \( # \) for the union of both conflicts, and we will use the notation \( \Box \) for internal conflicts and \( \hat{\sqcup} \) for external conflicts. Since we have two kinds of conflicts, we will have two kinds of “configurations”: the usual set of configurations, computed from \( # \), inside which no conflicts are tolerated, and the set of worlds inside which internal conflicts \( \Box \) are accepted. These considerations give rise to the following definition.

► Definition 10. A mixed event structure (mes) \( E \) is a quadruple \((|E|, \leq E, \#E, \hat{\sqcup} E)\) where

- \( \mathcal{U}(E) = (|E|, \leq E, \#E, \hat{\sqcup} E) \) is an event structure. We write \( C(E) = C(\mathcal{U}(E)) \) for the set of configurations. We define \( \rightsquigarrow E \to \mathcal{E}(E), [e], [c] \) as previously.
- \( (|E|, \leq E, \hat{\sqcup} E) \) is an event structure. We write \( \mathcal{W}(E) = C(|E|, \leq E, \hat{\sqcup} E) \) for the set of worlds.
- \( \hat{\sqcup} E \subseteq \#E \), or equivalently \( C(E) \subseteq \mathcal{W}(E) \).

We define the internal conflict \( \Box E \) as \( (\#E \setminus \hat{\sqcup} E) \).

The internal conflict \( \Box E \) is a symmetric irreflexive relation, but \((|E|, \leq E, \Box E)\) may not be an event structure. In fact, we will later (Definition 16) consider the special case \( \Box E \subseteq \rightsquigrightarrow E \). The operation \( \mathcal{U} \) turns out to be the right adjoint in a coreflection (Theorem 13) between mes and event structures. Graphically, we use dashes to represent minimal internal conflict (i.e. \( \rightsquigrightarrow E \cap \Box E \)), wiggly lines to represent the minimal conflict of \((|E|, \leq E, \hat{\sqcup} E)\), and arrows to represent \( \rightarrow E \). Those are enough to characterise the conflicts. For example, in Figure 8, we can deduce \( c \hat{\sqcup} c \) in both \( E_1 \) and \( E_2 \), in \( E_1 \) we also have \( d \hat{\sqcup} c \) while in \( E_2 \) we have \( d \Box c \).

Our goal is to extend all the operations previously defined on event structures. So we want mes to have (a)synchronous products, and a sequentialisation \( \text{Seq} \) functor forming a coreflection. In particular, it means sequentialising a mes which is already sequential (Definition 16) should be an isomorphism, so in Figure 9, \( \text{Seq}(A) \cong A \). Moreover, we want this extension to be conservative: a mes with \( \Box = \emptyset \) (i.e. \( \mathcal{W} = \mathcal{C} \)) should behave as an event structure. In particular, the sequentialisation of an event structure should be preserved by
Firstly, since we want $\mathcal{U}$ to be a functor, then for every map of mes $f$ from $A$ to $B$, there corresponds a map of event structures $\mathcal{U}(f)$ from $\mathcal{U}(A)$ to $\mathcal{U}(B)$.

Secondly, since we want $\mathcal{U}$ to form a coreflection, for every mes $A$, we need a map from $\mathcal{U}(A)$ to $A$ which is the identity function on events. This implies that we have an identity-on-events map from the mes with two events $a \mathcal{I} b$ to the mes with two events $a \mathcal{I} b$. Since we do not want these two mes to be isomorphic, we do not want any identity-on-events map from $a \sqsubseteq b$ to $a \mathcal{I} b$. This leads to the second condition on maps of mes: maps of mes preserve worlds.

Thirdly, considering Figure 9 again, any total map $g$ from $\text{Seq}(A)$ to $\text{Seq}(B)$ has to send the world $\{a_1, a'_1\}$ to a world, hence necessarily $g(a_1) = g(a'_1)$. It follows that for Seq to be a functor, any map $f$ from $A$ to $B$ should respect $f(a) = f(a')$. So we need to forbid $f(a) = b$ and $f(a') = b'$ from forming a map. Which leads to the following restriction: if $a \mathcal{I} a'$, $f$ defined on $a$ and $a'$, and $f(a) \neq f(a')$, then $f(a) \not\mathcal{I} f(a')$.

Lastly, due to some more subtle problems in the case of partial maps (see Appendix A.1), we need the domain of the map to be closed under $\mathcal{I}$.

Definition 11. A map $f$ of mes from $E$ to $F$ is a (possibly partial) function from $|E|$ to $|F|$ satisfying:

map of event structures

- for every $w \in \mathcal{W}(E)$, $f(w) \in \mathcal{W}(F)$;
- $\Box$-preserving for every $a \sqsubseteq b$ where $f$ is defined and $f(a) \neq f(b)$, then $f(a) \Box f(b)$; and
- $\Box$-equidefinability for every $a \sqsubseteq b$, $f$ is defined on both or none.

Proposition 12. Mes and partial maps of mes form a category, written $\text{MES}_p$, with the empty event structure as a terminal object. Mes and total maps of mes form a subcategory, written $\text{MES}$.

Theorem 13. The functor $\mathcal{U}$ is a left adjoint to the inclusion, forming a coreflection.
3.2 Products and Sequentiality

From the previous coreflection, we know that mes coming from event structures have synchronous and asynchronous products. These constructions extend to all mes. The proof is technical, and can be found in Appendix B.1.

**Proposition 14.** MES is a cartesian category, with \( \times_* \) as a product and \( \emptyset \) as a unit. MES has a product \( \times \).

As a remark, since we have \( U(A \times_\ast B) = U(A) \times U(B) \), we know that the events, order, and conflict of the mes \( A \times_\ast B \) are the same as those of the event structure \( U(A) \times U(B) \). And we have a similar property for \( A \times B \). One can then characterise the internal conflict:

**Proposition 15.** For \( E_1 \) and \( E_2 \) two mes, we have:

\[
a \sqcap_{E_1 \times_* E_2} b \iff [a] \cup [b], [a] \cup [b] \in W(E_1 \times_* E_2) \text{ and } \forall i \in \{1, 2\}, \quad \begin{cases} \pi_i^* \text{ unde\textcircled{ }} \text{ on } \{a, b\}, \text{ or} \\ \pi_i^*(a) = \pi_i^*(b), \text{ or} \\ \pi_i^*(a) \sqcap_{E_i} \pi_i^*(b) \end{cases}
\]

\[
a \sqcap_{E_1 \times E_2} b \iff [a] \cup [b], [a] \cup [b] \in W(E_1 \times E_2) \text{ and } \forall i \in \{1, 2\}, \quad \begin{cases} \pi_i(a) = \pi_i(b), \text{ or} \\ \pi_i(a) \sqcap_{E_i} \pi_i(b) \end{cases}
\]

This characterisation is recursive, since it refers to worlds and worlds are defined by the internal conflict. However, this is a well-founded recursion\(^4\), so this proposition could be used as a definition of the (a)synchronous product. Figure 10 is an example of the synchronous and asynchronous product of two mes.

We now aim to extend the definition of sequentiality to mes. However, a problem arises: the usual definition of sequentiality “\( \forall a, b \in c \in C(E) \implies a \leq b \) or \( b \leq a \)” gives only a many-to-one correspondence with trees, and is no longer equivalent to the categorical definition “there exists a total map from \( E \) to \( \mathcal{N} \).” Indeed, since maps are expected to be \( \sqcap \)-preserving, the mes \( E_2 \) from Figure 8 satisfies the first but not the second. So we strengthen the first definition so that it is equivalent to the second: we require internal conflicts to be minimal, i.e. \( \sqcap \subseteq \sim \). This allow to recover a one-to-one correspondence with alternating trees (Proposition 20).

**Definition 16.** A mes \( E \) is sequential if it satisfies one of the following equivalent properties:

\(^4\) The well-founded relation is \( (a', b') \prec (a, b) \iff a' \leq_{E_1 \times_\ast E_2} a \) and \( b' \leq_{E_1 \times_\ast E_2} b \) and \( (a', b') \neq (a, b) \).
Figure 11 Alternating trees and mes

(a) A alternating tree and its sequential mes

E is forest-shaped, with branches being in conflict with each other, and \( \Box_E \subseteq \sim_E \).

For every \( a, b \in x \in C(E) \), either \( a \leq_E b \) or \( b \leq_E a \). Moreover, \( \Box_E \subseteq \sim_E \).

There exists a (necessarily unique) total map from \( E \) to \( N \).

\( E \) is isomorphic to \( E \times N \).

Analogously to the correspondence between trees and event structures, alternating trees (Definition 18) will correspond to sequential mes.

Proposition 17. We write \( \text{Seq-MES} \) (resp. \( \text{Seq-MES}_\star \)) for the subcategory of \( \text{MES} \) (resp. \( \text{MES}_\star \)) with only sequential mes. It is a cartesian category, with \( \times \) (resp. \( \times_\star \)) as a product and \( N \) (resp. \( \varnothing \)) as a terminal object.

Now, we define the sequentialisation as previously: for a mes \( E \), we define the sequential mes \( \text{Seq}(E) \) as \( E \times N \). The coreflection (Theorem 6) still holds in the case of mes. A proof can be found in Appendix B.4.

3.3 Labelled Mixed Event Structures

As in the case of event structures, one can add labels to mes. A mes labelled in \( \mathcal{L} \) is a mes \( E \) together with a labelling function \( \ell_E : |E| \rightarrow \mathcal{L} \). Maps are required to preserve labels, \( \text{Seq} \) preserves labels, and we compute the labels of asynchronous product using \( \bullet \).

3.4 Alternating Trees

Just as we can choose a root node of an LTS and unfold it into a tree, so can we choose a root node in a Segala automaton and unfold it into a Segala tree. We define here labelled alternating trees which can be understood as “Segala trees, but without probabilities”; the complete definition of a Segala tree will come later (Definition 24), and we will omit the definition of Segala automaton [13, 14].

Definition 18. A labelled alternating tree is a tree such that:

- Nodes of even depth (including the root) are called states. All leaves must be states.
- Nodes of odd depth are called intermediary positions.
- Transitions from intermediary positions to states are labelled. They are called internal transitions and written \( \ell \rightarrow \rightarrow \ell \) if labelled by \( \ell \).
- Transitions from states to intermediary positions are unlabelled. They are called external transitions and written \( \rightarrow \rightarrow \).

Labelled alternating trees can be represented directly by labelled sequential mes. Figure 11a shows an alternating tree and its representation by a labelled sequential mes. Every intermediary position of the labelled alternating tree corresponds to a cell (set of events that are pairwise related by \( \Box \)), and each labelled transition from this intermediary position corresponds to a labelled event of this cell. However, this is not a one-to-one correspondence, since labelled alternating trees will only generate mes where \( \Box \) is a transitive relation. In
particular, the labelled sequential mes in Figure 11b cannot be built using labelled alternating trees.

\textbf{Definition 19.} A mes $E$ is said to be locally transitive if for every $x \overset{a}{\rightarrow} y$, $x \overset{b}{\rightarrow} z$, and $x \overset{c}{\rightarrow} w$ with $x \in C(E)$ and $a, b, c$ pairwise distinct, we have $a \sqsubseteq_E b \sqsubseteq_E c \Rightarrow a \sqsubseteq_E c$.

We use this more restricted notion of transitivity because “$\sqsubseteq$ is transitive” is not a property stable under (a)synchronous products, while local transitivity is: the (a)synchronous product of two locally transitive mes is locally transitive. In fact every definition and result previously stated still applies if we restrict ourselves to locally transitive mes.

\textbf{Proposition 20.} There is a one-to-one correspondence between labelled alternating trees and labelled locally transitive sequential mes. Moreover, for a mes $E$, $\text{Seq}(E)$ corresponds to an alternating tree if and only if $E$ is locally transitive.

4 Concurrent Semantics of Probabilistic CCS

The goal of this section is to give a concurrent semantics to Probabilistic CCS, with a factorisation property similar to the one of CCS. We first add a probabilistic valuation to mes, in order to relate them to Segala trees, used for the interleaving semantics of PCCS. Then, we describe how to extend the concurrent semantics for CCS into one for PCCS.

4.1 Mixed Probabilistic Event Structures

We recall some notions of probabilistic event structures \cite{15,20}. They are event structures together with a probability valuation $v : C(E) \rightarrow [0,1]$, interpreted as the probability of reaching at least this configuration, such that $v(\emptyset) = 1$ and a condition of monotonicity (Definition 21). Simple consequences of the condition of monotonicity are:

- The valuation $v$ is decreasing: $x \subseteq y \Rightarrow v(x) \geq v(y)$
- Events in conflict have conditional probability whose sum is less than or equal to one:
  \[ \forall i \leq j \leq n, x \overset{a_i}{\rightarrow} y \text{ and } \forall i < j \leq n, a_i \neq a_j \Rightarrow \sum_{i=1}^{n} v(A_{\langle i \rangle}) \leq 1 \]
- Events not in conflict respect a variant of the inclusion-exclusion principle:
  \[ x, y, x \cap y, x \cup y \in C(E) \Rightarrow v(x \cup y) - v(x) - v(y) + v(x \cap y) \geq 0 \]

We want to add similar conditions to mes. We consider a mes with two events $a$ and $b$. We aim to use the internal conflict $\square$ of mes for probabilistic choices $+$ of PCCS, meaning that if $a \sqsubseteq b$, we can expect $v(\{a\}) + v(\{b\}) \leq 1$, so they respect the monotone condition. However, since we aim to use the external conflict $\uplus$ for the nondeterministic choice $\ominus$ of PCCS, if $a \uplus b$, we can have $v(\{a\}) = 1 = v(\{b\})$, which does not respect the monotone condition. This guides us to the following condition: within worlds, conflicts are necessarily $\square$, so the monotone condition has to be respected, however no condition is expected to hold across different worlds.

\textbf{Definition 21.} A mixed probabilistic event structure (mpes) is a mes $E$ together with a probabilistic valuation $v_E : C(E) \rightarrow [0,1]$ such that:

- **Initialised** $v_E(\emptyset) = 1$
- **Monotone** For $x, y_1, \ldots, y_n \in C(E)$, with $\forall i \leq j \leq n, x \subseteq y_i$, we write:
  - For $\emptyset \neq I \subseteq \{1, \ldots, n\}$, $y_I := \bigcup_{i \in I} y_i$
  - For $X \notin C(E)$, $v_E(X) := 0$
  - $d(x; y_1, \ldots, y_n) := v_E(x) - \sum_{Z \neq I \subseteq \{1, \ldots, n\}} (-1)^{|I|+1} v_E(y_I)$

We then demand that $y_{\{1, \ldots, n\}} \in W(E) \Rightarrow d(x; y_1, \ldots, y_n) \geq 0$
It is labelled if the underlying mes is labelled. It is sequential if the underlying mes is sequential. A map of mes is a map of mes \( f : A \rightarrow B \) such that \( \forall x \in C(A), v_A(x) \leq v_B(f(x)) \).

Unlike [20], we use a valuation that is strictly positive on configurations instead of positive or null. This choice come from the absence of 0-probability branches in Segala trees.

We write \( \text{MPES}_* \), \( \text{MPES} \), \( \text{Seq-MPES}_* \), and \( \text{Seq-MPES} \) for the categories of mes with maps restricted to total ones and/or with objects restricted to sequential ones. The empty mes \( \emptyset \) is a terminal object for \( \text{MPES}_* \) and \( \text{Seq-MPES}_* \), while \( \mathcal{N} = (\mathbb{N}, \leq, \emptyset, \emptyset, v_{\mathcal{N}} : x \mapsto 1) \) is a terminal object for \( \text{Seq-MPES} \).

We now want to extend the (a)synchronous product to mes, but it is unfortunately not always possible to preserve the universal property of (a)synchronous products.

\textbf{Proposition 22.} \( \text{MPES} \) and \( \text{MPES}_* \) do not have all products.

A counterexample can be found in Appendix B.2. While no categorical product can be defined, both \( \text{MPES} \) and \( \text{MPES}_* \) have symmetric monoidal products \( \otimes \) and \( \otimes_* \) which generalise \( \times \) and \( \times_* \) of mes. The underlying problem of the absence of products is a problem of probabilistic correlation. We fail to define \( A \times B \) because we are missing the information on the correlations between the events of \( A \) and the events of \( B \). So when defining \( A \otimes B \), we make the most canonical choice and consider that \( A \) and \( B \) are probabilistically independent. This choice is also consistent with the processes of PCCS: when considering \( (P|Q) \), the probabilistic choices inside \( P \) are assumed to be independent of the probabilistic choices inside \( Q \).

\textbf{Definition 23.} For \( A \) and \( B \) two mes, we define \( A \otimes_* B \) as follows:

\begin{itemize}
  \item The underlying mes is \( A \times_* B \), where \( A \) and \( B \) are seen as mes.
  \item \( \forall x \in C(A \otimes_* B), v_{A \otimes_* B}(x) = v_A(\pi_1^*(x)) \cdot v_B(\pi_2^*(x)) \).
\end{itemize}

This is a mes (proof in Appendix B.3). We define \( A \otimes B \) similarly.

A remarkable property is that when \( A \) (or \( B \)) respects \( \forall x \in C(A), v_A(x) = 1 \), then \( A \otimes_* B \) is a product in \( \text{MPES}_* \) and \( A \otimes B \) too in \( \text{MPES} \). The sequentialisation \( \text{Seq}(E) := E \otimes \mathcal{N} \) still induces a coreflection.

We stated earlier that alternating trees were “Segala trees without probabilities”, and we will now define Segala trees, and explain their correspondence with locally transitive sequential mes.

\textbf{Definition 24.} A Segala tree labelled by \( \mathcal{L} \) is an alternating tree labelled by \( (0,1] \times \mathcal{L} \), such that if we have \( j \rightarrow^\cdot_{(p_k,a_k)} s_k \) for \( 1 \leq k \leq n \) (with \( s_k \) being distinct), then \( \sum_{k=1}^n p_k \leq 1 \).

\textbf{Theorem 25.} There is a one-to-one correspondence between Segala trees and labelled locally transitive sequential mes.

This correspondence is given by the correspondence between labelled alternating trees and labelled locally transitive sequential mes, with the probability of a transition \( j \rightarrow^\cdot_{(p,a)} \) corresponding to an event \( e \) given by \( p = \frac{v_{\mathcal{L}}([e])}{\sum_{e \in \mathcal{L}} v_{\mathcal{L}}([e])} \), and reciprocally the valuation of a configuration \( x \) being the product of all the probabilities on transitions of the Segala tree along the branch corresponding to \( x \).

\subsection{4.2 Concurrent Semantics of PCCS}

In order to define the concurrent semantics of PCCS, we first extend the constructors on event structures used for CCS to mes.
The concurrent semantics of PCCS is given by Figure 5b together with this additional rule: $[\sum_{i \in I} p_i \cdot a_i p_i]_{cs} := \sum_{i \in I} p_i \cdot \llbracket a_i \rrbracket_{cs}$. See Figure 12 for an example.

We note that for every process $P$ of PCCS, $[P]_{cs}$ is locally transitive, meaning that its sequentialisation will correspond to a Segala tree. If we write $[P]_{is}$ for the semantics of the process $P$ with Segala trees, seen as labelled, locally transitive, sequential mpes, then we have the following factorisation theorem.

\textbf{Theorem 26 (Factorisation).} For any process $P$ of PCCS, we have $[P]_{is} \cong \text{Seq}([P]_{cs})$.

The proof of this theorem relies on the sequentialisation coreflection, which ensures that the interpretation of the parallel composition is preserved.

This concludes the main contribution of this paper: we have built a concurrent model able to represent both probabilistic and nondeterministic choices, and used it to give to PCCS a concurrent semantics compatible with its usual interleaving semantics.

As future work, we wish to investigate PCCS without the guard restriction, hence allowing \textit{unguarded} probabilistic choice $\sum_{i \in I} p_i \cdot P_i$. We are able to give to Unguarded PCCS a concurrent semantics using potentially non locally transitive mpes, however the relationship with existing interleaving semantics remains unclear.
References


We argue in this subsection the need for the □-equideﬁnability condition on maps of mes. We consider the example in Figure 13. The mes $P$ is equal to the event structure $U(A) \times_s U(B)$ seen as a mes, so for Theorem 13 to be true, we would want $P$ to be the asynchronous product of $A$ and $B$. However, we have two functions from $Q$ to $A$ $((a,\star),(a,b)\mapsto a$, undefined on $(\star,b))$ and $Q$ to $B$ $((\star,b),(a,b)\mapsto b$, undefined on $(a,\star))$ that are maps of event structures, preserve worlds, and are □-preserving, but any function from $Q$ to $P$ that commute with them is not world preserving. So without □-equideﬁnability, $P$ would not be an asynchronous product of $A$ and $B$.

It could appear that not taking □-equideﬁnability and accepting $Q$ as the asynchronous product of $A$ and $B$ instead of $P$ would work. This is however only a temporary solution. If we consider the mes $A'$ which is $A$ with the valuation $v_A : x \mapsto 1$, and the mes $B'$ which is $B$ with valuation $v_B : x \mapsto 1$, then we would want to deﬁne $A' \otimes_s B'$ to be $Q$ with a valuation $v_Q$. However, the monotone condition on the valuation $v_Q$ would forbid to have $v_Q : x \mapsto 1$, and we would be forced to introduce arbitrary probabilities. This would eventually break the factorisation theorem (Theorem 26).

A.2 Coreflection of Event Structures into Mes

In this subsection, we aim to prove Theorem 13. We will strengthen it by showing that on top of having a coreflection between $ES_s$ and $MES_s$, we also have a reﬂection. Indeed, the inclusion functor from $ES_s$ is both the left adjoin of the forgetful functor $U$, and the right adjoin of the merging functor $Mer$. We start by deﬁning all those functors.

**Definition 27.** For $E$ an event structure and $f$ a map of event structures, we deﬁne the inclusion functor $\hookrightarrow$ of $ES_s$ into $MES_s$ as $\hookrightarrow (E) := (\|E\|, \leq_E, \#_E, \#_E)$ and $\hookrightarrow (f) := f$.

**Definition 28.** For $E$ a mes and $f$ a map of mes, we deﬁne the forgetful functor $U$ from $MES_s$ to $ES_s$ as $U(E) := (\|E\|, \leq_E, \#_E)$ and $U(f) := f$.

The merging functor $Mer$ is an operation that merges every events related by □ together. It will however only merges event that are in conﬂict, and will delete the events that cannot be merged. For example, the mes with three events $a \Box b \Box c$ and $a \neq c$ will be merged into the event structure with only one event, while the event structure with three events $a \Box b \Box c$ with $\{a, c\}$ a conﬁguration will be merged into the empty event structure. For a set $S$ and a binary relation $R$ on $S$, we write $S/R$ for the quotient of $S$ by the reﬂexive transitive closure of $R$.

**Definition 29.** For $E$ a mes and $f$ a map of mes, we deﬁne the merging functor $Mer$ from $MES_s$ to $ES_s$ as:

- $|Mer(E)| = \{X \in |E|/\Box_E | \forall a \neq b \in X, a \#_E b\}$
- $X \#_s Mer(E) Y \iff \forall a \in X, \forall b \in Y, a \#_E b$
- $X \leq_s Mer(E) Y \iff \forall b \in Y, \exists a \in X, a \leq_E b$
- $dom(Mer(f)) = \{X | \forall a \in X, a \in dom(f)\} = \{X | \exists a \in X, a \in dom(f)\}$ (well-deﬁned since $f$ is □-equideﬁned)
Mer(f)(X) = Y such that ∀a ∈ X, f(a) ∈ Y
(well-defined since f is □-preserving)

Note that showing the antisymmetry of \( \leq_{\text{Mer}(E)} \) is non-trivial: if we assume \( X \leq_{\text{Mer}(E)} Y \leq_{\text{Mer}(E)} X \), it means that ∀a ∈ X, b ∈ Y, ∃e ∈ X, c ∈ E, a ≤E b ≤E a, but a, c ∈ X so either a = c or a ≠E c. Since a ≤ c, we necessarily have a = c, hence a = b = c. Since events of Mer(E) are disjoints set of events of E, it means that \( X = Y \).

Theorem 30. The functor \( U \) is the right adjoint to the inclusion, for partial and total maps, forming a coreflection. The functor Mer is the left adjoint to the inclusion, for partial maps, forming a reflection.

\[
\begin{array}{ccc}
\text{ES} & \xrightarrow{i} & \text{MES} \\
\downarrow{U} & & \downarrow{\text{Mer}} \\
\text{ES} & \xrightarrow{\iota} & \text{MES}_\perp
\end{array}
\]

\( U \circ \iota \leftrightarrow \) is an isomorphism and \( \text{Mer} \circ \iota \leftrightarrow \) is an isomorphism

Additionally, while \( \iota \leftrightarrow \) is not a right adjoint for total maps, it still preserves categorical limits.

Proof. Firstly, we prove the coreflections. We prove that a map of event structure from \( E \) to \( U(F) \) can be seen as a map of mes from \( \iota(E) \) to \( F \): worlds of \( \iota(E) \) are configurations, so are preserved by the map, and \( \square_{\iota(E)} = \emptyset \) so the map is \( \square \)-equidefined and \( \square \)-preserving. Reciprocally, a map from \( \iota(E) \) to \( F \) is by definition also a map from \( U(\iota(E)) \) to \( U(F) \). And also by definition, \( U(\iota(E)) = E \).

Secondly, we prove the reflection in the case of partial maps. Let \( m \) be the map of mes from \( E \) to \( \iota(Mer(E)) \) defined as \( m(a) = X \) if \( a \in X \in |Mer(E)| \) and undefined otherwise. By unfolding the definition, we check without difficulties that this map reflect order, reflect conflict, is \( \square \)-equidefined, is \( \square \)-preserving, and hence preserved worlds. The map \( m \) define a natural transformation, and together with the identity transformation of \( \iota \), they form a unit-counit adjunction. We remark that since \( m \) may be a partial map, this does not form an adjunction for total maps.

Thirdly, we prove that \( \iota \leftrightarrow \) still preserves categorical limits in the case of total maps. We remark that \( ES \) does not have a terminal object, so categorical limits of arity zero are preserved. Then, we use an intermediary lemma.

Lemma 31. We write \( \text{MES}^{wt} \) the restriction of \( \text{MES} \) to the mes \( E \) such that there exists an event structure \( F \) and a total map from \( E \) to \( \iota(F) \). A mes is in \( \text{MES}^{wt} \) if and only if it is weakly transitive, i.e. \( \forall a, b, c \text{ pairwise distinct}, a \square b \square c \implies a \neq c \). With those definitions we have:

\[
\begin{array}{ccc}
\text{ES} & \xrightarrow{i} & \text{MES}^{wt} \\
\downarrow{U} & & \downarrow{\text{Mer}} \\
\text{ES} & \xrightarrow{\iota} & \text{MES}_\perp
\end{array}
\]

\( U \circ \iota \leftrightarrow \) is an isomorphism and \( \text{Mer} \circ \iota \leftrightarrow \) is an isomorphism

Proof. \( m \) is always a total map in \( \text{MES}^{wt} \). Both the equivalence between the two definitions and the reflection follows from this observation.

Using this lemma, we obtain that \( \iota \leftrightarrow \) send categorical limits of \( ES \) into categorical limits of \( \text{MES}^{wt} \). But if we consider \( \iota \leftrightarrow \) from \( ES \) to \( \text{MES} \), except for arity zero categorical limits, the universal property of categorical limits ensure that there is always a map from \( E \) to \( \iota(F) \) for some \( F \), hence \( \iota \leftrightarrow \) does preserve categorical limits from \( ES \) to \( \text{MES} \).
Appendices on Products

B.1 Existence of Products of Mes

In this subsection, we prove that we can deduce the existence of (a)synchronous products of mes from the existence of (a)synchronous product of event structures \cite{17}.

\begin{theorem}
\text{MES} is a cartesian category, and so is its subcategory of locally transitive mes.
\end{theorem}

\textbf{Proof.} The terminal object in the empty mes $\emptyset$.

For $A$ and $B$ two mes, we define $P$ as follows: $\mathcal{U}(P) = \mathcal{U}(A) \times \mathcal{U}(B)$ and

\[ a \triangleleft P b \iff [a] \cup [b], [a] \cup [b] \in \mathcal{W}(P) \text{ and } \forall i \in \{1, 2\}, \begin{cases} \pi_i^1 \text{ undefined on } a \text{ and } b, \text{ or} \\ \pi_i^1(a) = \pi_i^1(b), \text{ or} \\ \pi_i^2(a) \sqcup_E \pi_i^2(b) \end{cases} \]

This is a mes, and the projections to $A$ and $B$ are maps of mes.

We will prove that $P$ is a product. We take $(E, f, g)$ with $f : E \to A$ and $g : E \to B$. We have $\mathcal{U}(f) : \mathcal{U}(E) \to \mathcal{U}(A)$ and $\mathcal{U}(g) : \mathcal{U}(A) \to \mathcal{U}(B)$ so there exists $h : \mathcal{U}(E) \to \mathcal{U}(P)$ which commute with the projections. We need to prove that $h$ can be seen as a map from $E$ to $P$.

\textbf{□-equidefinability} If $a \sqsubseteq_E b$, then both $f$ and $g$ are equidefined on $a$ and $b$. Since the domain of $h$ is the union of the domain of $f$ and $g$, it means that $h$ is equidefined on $a$ and $b$.

\textbf{□-preservation} If $a \sqsubseteq_E b$ with $h$ defined and injective on $\{a, b\}$, then we have $[a] \cup [b], [a] \cup [b] \in \mathcal{W}(E)$. We recursively assume that $h$ preserves those two worlds, which implies $[h(a)] \cup [h(b)], [h(a)] \cup [h(b)] \in \mathcal{W}(P)$. By commutation with the projections, we know that the projections are equidefined on $h(a)$ and $h(b)$. And since $f$ and $g$ preserve $\square$, we know that $\pi_1^1$ and $\pi_2^1$ preserves $\square$.

\textbf{world preservation} We take a world $w \in \mathcal{W}(E)$. We knows that $h$ preserves every configuration included in $w$, and we can recursively assume that it preserves every $\square$ included in it. It follows that $h(w) \in \mathcal{W}(P)$.

We now assume that $A$ and $B$ are locally transitive. We take $x \xrightarrow{a_1} x \xrightarrow{a_2} x \xrightarrow{a_3} x$ with $x \in \mathcal{C}(P)$ and $a_1 \sqsubseteq_P a_2 \sqsubseteq_P a_3$. We know that $\pi_1^1$ and $\pi_2^1$ are equidefined on $\{a_1, a_2, a_3\}$. If defined, we have $\pi_i^1(x) \xrightarrow{\pi_i^1(a_k)}$ for $i \in \{1, 2\}$ and $k \in \{1, 2, 3\}$. Using local transitivity on $A$ and $B$, we knows that if defined, $\pi_i^1(a_1) \sqcup \pi_i^2(a_3)$ or $\pi_i^1(a_1) = \pi_i^3(a_3)$. From the definition of $\square_p$, it follows that $a_1 \sqsubseteq_P a_3$.

Similarly, \text{MES} has products, and so does its subcategory of locally transitive mes.

B.2 Counterexample to the Existence of Product of Mpes

\begin{proposition}
\text{MPES} and \text{MPES} do not have all products.
\end{proposition}

\textbf{Proof.} We consider the mpes $E$ with two events $a \sqsubseteq b$ and with $v(\{a\}) = \frac{1}{2} = v(\{b\})$. If $E \times E$ was a well defined product in \text{MPES}, the universal property would implies it has at least four events $(a, a)$, $(a, b)$, $(b, a)$ and $(b, b)$ all related by $\square$, and with projections induced from their name. Using the universal property for $(E, \text{id}_E, \text{id}_E)$, we obtain a map $f : E \to E \times E$ with $f(a) = (a, a)$ and $f(b) = (b, b)$. Since $f$ is a map, it means that $v((\{a, a\})) \geq \frac{1}{2}$ and $v((\{b, b\})) \geq \frac{1}{2}$. Using the monotone condition on the mpes $E \times E$, we obtain that $v((\{a, b\})) = 0 = v((\{b, a\}))$. However, we also have a map $g : E \to E$
with \( g(a) = b \) and \( g(b) = a \), so using the universal property for \((E, \text{id}_E, g)\), we obtain \( h : E \to E \times E \) with \( h(a) = (a, b) \) and \( h(b) = (b, a) \), meaning that \( v(\{(a, b)\}) \geq \frac{1}{2} \) and \( v(\{(b, a)\}) \geq \frac{1}{2} \). Contradiction. (A similar counterexample applies for MPES\(_s\)).

### B.3 Existence of Symmetric Monoidal Product of MPES

In this subsection, we show that the definition of \( \otimes \) and \( \otimes_s \), given previously actually work.

> **Proposition 34.** For \( A \) and \( B \) two mpes, we define \( A \otimes_s B \) as follows:

1. The underlying mes is \( A \times_s B \), where \( A \) and \( B \) are seen as mes.
2. \( \forall x \in \mathcal{C}(A \otimes_s B), v_{A \otimes_s B}(x) = v_A(\pi_1^*(x)) \cdot v_B(\pi_2^*(x)) \).

The operation \( \otimes_s \) form a symmetric monoidal product in MPES\(_s\). The operation \( \otimes \) defined similarly form a symmetric monoidal product in MPES.

**Proof.** We recover most of the properties from the underlying product of mes. The only non-trivial property to prove is that the valuations \( v_{A \otimes B} \) and \( v_{A \otimes B} \) are monotone.

We take \( x \subseteq y_i \) for \( 1 \leq i \leq n \), and we consider \( d_{A \otimes B}(x; y_1, \ldots, y_n) \). We use an intermediary lemma:

> **Definition 35.** A drop family over a set \( S \) is a family \((a_I)_{I \subseteq S}\) of non-negative real numbers such that \( \sum_{I \subseteq S} (-1)^{|I|} a_I \geq 0 \) and such that for all \( S = U \cup V \cup W \) with \(|V| < |S|\), \((a_{I \cup V})_{I \subseteq V}\) is a drop family over \( V \). In particular, we have \( I \subseteq I \cup \{e\} \subseteq S \implies a_I \geq a_{I \cup \{e\}} \).

> **Lemma 36.** Given two drop families \((a_I)_{I \subseteq S}\) and \((b_I)_{I \subseteq S}\), then \((a_I \cdot b_I)_{I \subseteq S}\) is a drop family.

**Proof.** We proceed by induction on the cardinal of \( S \). If \(|S| = 0\), then \( a_S \cdot b_S \geq 0 \), so it is a drop family.

If \( S \neq \emptyset \), then we decompose it into \( S = T \cup \{e\} \). We have:

\[
\sum_{I \subseteq S} (-1)^{|I|} a_I \geq 0 = \sum_{I \subseteq T} (-1)^{|I|} (a_I - a_{I \cup \{e\}}) \\
\sum_{I \subseteq S} (-1)^{|I|} b_I \geq 0 = \sum_{I \subseteq T} (-1)^{|I|} (b_I - b_{I \cup \{e\}}) \\
(a_I - a_{I \cup \{e\}})_{I \subseteq T} \quad \text{and} \quad (b_I - b_{I \cup \{e\}})_{I \subseteq T} \quad \text{are two drop families.}
\]

\[\text{and} \quad (a_{I \cup \{e\}})_{I \subseteq T} \quad \text{are two drop families.}\]

Using the induction hypothesis:

\[\text{and} \quad (a_{I \cup \{e\}} \cdot b_I)_{I \subseteq T} \quad \text{is a drop family.}\]

Then since we have:

\[
\sum_{I \subseteq S} (-1)^{|I|} a_I \cdot b_I = \sum_{I \subseteq T} (-1)^{|I|} (a_I - a_{I \cup \{e\}})(b_I - b_{I \cup \{e\}}) + \sum_{I \subseteq T} (-1)^{|I|} a_I \cdot b_{I \cup \{e\}} + \sum_{I \subseteq T} (-1)^{|I|} a_{I \cup \{e\}} \cdot b_I
\]

We can deduce that \( \sum_{I \subseteq S} (-1)^{|I|} a_I \cdot b_I \geq 0 \). The induction hypothesis ensure that for all \( S = U \cup V \cup W \) with \(|V| \leq |S|\), we have a drop family over \( V \), so \((a_I \cdot b_I)_{I \subseteq S}\) is a drop family.

Using this lemma, we just need to prove that \((v_A(\pi_1^*(y_i)))_{I \subseteq \{1, \ldots, n\}}\) and \((v_B(\pi_2^*(y_i)))_{I \subseteq \{1, \ldots, n\}}\) are two drop families. Since maps preserve worlds, this is obviously true. So \((v_{A \otimes B}(y_i))_{I \subseteq \{1, \ldots, n\}}\) is a drop family, hence the monotone condition is respected. We can do a similar reasoning for \( v_{A \otimes B} \).
Figure 14 Proof of the functoriality of Seq

(a) The diagram we want

(b) The diagram we have

(b.4 Sequentialisation Coreflection)

In this subsection, we prove the extension of Appendix B.4 to mes and mpes. We start with some preliminary properties. One can remark that the synchronous product is a substructure of the asynchronous product. The exact lemma we will need for our proofs is the following one:

Lemma 37. For \( A \) and \( B \) two mes, there exists a unique map \( \iota : A \times B \to A \times_* B \), necessarily total, such that \( \pi_1^* \circ \iota = \pi_1 \) and \( \pi_2^* \circ \iota = \pi_2 \). There exists also a unique partial map \( \rho : A \times_* B \to A \times B \) such that \( \pi_1 \circ \rho = \pi_1^* \) and \( \pi_2 \circ \rho = \pi_2^* \).

Proof. The existence of \( \iota \) is simply the universal property of the asynchronous product. The totality of \( \iota \) comes from the commutation \( \pi_1^* \circ \iota = \pi_1 \).

We define \( \iota(A \times B) \) as \( A \times_* B \) restricted to the image of \( \iota \). We remark that \( \pi_1^* \) and \( \pi_2^* \) restrict to total maps on \( \iota(A \times B) \), so using the universal property of the total product, we obtain that \( \iota(A \times B) \) is isomorphic to \( A \times B \).

We have a partial surjective identity-on-events map from \( A \times_* B \) to \( \iota(A \times B) \). Since \( \iota(A \times B) \) is isomorphic to \( A \times B \), we define \( \rho \) as the induced map (partial and bijective) from \( A \times_* B \) to \( A \times B \). The uniqueness of the map is ensured by the commutation with the projection and the universal property of the total product. We have by definition \( \rho \circ \iota = \text{id}_{A \times B} \).

We moreover have \( \rho \circ \iota = \text{id}_{A \times B} \).

We now prove that \( \text{Seq} \) is well-defined.

Proposition 38. \( \text{Seq} \) is a functor from \( \text{MES}_* \) (and \( \text{MES} \)) to \( \text{Seq-MES}_* \) (and \( \text{Seq-MES} \))

Proof. There is a total map from \( \text{Seq}(E) = E \times N \) to \( N \), so \( \text{Seq}(E) \) is sequential. For the functoriality, we take \( f : A \to B \) a partial map of mes, and we will show that there exist a unique partial map of mes \( \text{Seq}(f) : A \times N \to B \times N \) such that the diagram Figure 14a.

We now prove that \( \text{Seq} \) is well-defined.
Figure 15 Proof of the coreflection

(a) The diagram we want

\[ \begin{array}{c}
A \\
\downarrow \frown \downarrow \\
B \times N \\
\downarrow \frown \downarrow \\
B \\
\end{array} \]

(b) The diagram we have

\[ \begin{array}{c}
A \downarrow \downarrow \\
\downarrow \downarrow \\
B \times N \\
\downarrow \downarrow \\
B \\
\end{array} \]

commute. We now consider the diagram Figure 14b, where \( g \) is the unique partial map from the universal property of a partial product and \( \rho \) and \( \iota \) come from Lemma 37.

We can define \( \text{Seq}(f) \) as \( \rho \circ g \) and prove the unicity: we consider another map \( h : A \rightarrow B \times N \). By unicity of \( g \), we have \( \iota \circ h = g \), so \( \text{Seq}(f) = \rho \circ \iota \circ h = h \).

**Theorem 39.** \( \text{Seq} \) is a functor from \( \text{MES}_* \) (and \( \text{MES} \)) to \( \text{Seq-MES}_* \) (and \( \text{Seq-MES} \)), which is a right adjunct to the inclusion, forming a coreflection.

\[ \text{Seq-MES} \rightarrow \text{MES} \]
\[ \text{Seq-MES}_* \rightarrow \text{MES}_* \]

**Proof.** We take \( A \) a sequential mes and \( B \) any mes. We take \( f : A \rightarrow B \) a partial map of mes, and we will show that there exist a unique partial map of mes \( h : A \rightarrow B \times N \) such that the diagram Figure 15a commute. We now consider the diagram Figure 15b, where \( g \) is the unique partial map from the universal property of a partial product and \( \rho \) and \( \iota \) come from Lemma 37.

Using the same reasoning as in the previous proposition, we deduce that \( h = \rho \circ g \) is the unique adequate map. Which prove \( \text{MES}_*(A, B) \cong \text{MES}_*(A, B \times N) \). The adjunction follows. Since the sequentialisation of a sequential mes is isomorphic to the mes, it follows without problems that it is a coreflection. A similar proof works for \( \text{MES} \) and \( \text{Seq-MES} \).

This coreflection extends to mpes.

**Theorem 40.** \( \text{Seq} \) is a functor from \( \text{MPES}_* \) (and \( \text{MPES} \)) to \( \text{Seq-MPES}_* \) (and \( \text{Seq-MPES} \)), which is a right adjunct to the inclusion, forming a coreflection.

\[ \text{Seq-MPES} \rightarrow \text{MPES} \]
\[ \text{Seq-MPES}_* \rightarrow \text{MPES}_* \]

**Proof.** Since \( N \) has valuation equal to 1 everywhere, the map \( \pi_1 : \text{Seq}(E) \rightarrow E \) preserves the valuation: \( \forall x \in C(\text{Seq}(E)), v_{\text{Seq}(E)}(x) = v_E(x) \). It follows that we have \( \text{MPES}_*(A, B) \cong \text{MPES}_*(A, B \times N) \), which mean we have the coreflection.