# Internship report
# (short version)

Marc de Visme
École Normale Supérieure
France

Supervised by Glynn Winskel
University of Cambridge
United Kingdom

2015

**Abstract**

Event structures are a way to represent processes in which histories take the form of patterns of event occurrences. Often they can be seen as unfolded Petri Nets. [1] They are useful to model distributed games and strategies. One of the main restrictions of event structures is that the common way of representing disjunctive causes is not compatible with hiding, essential in the composition of strategies. [2] The main goal of my internship was to find a way to represent disjunctive causes while supporting both hiding and pull-backs, and to understand the relationship with the traditional approach. We express the relationship through an adjunction. [3] In fact the adjunction is one of a family of adjunctions. The new structures can be used to model strategies without the previous limitations.

Silvain Rideau and Glynn Winskel introduced in [RW11] a very general definition of games and strategies based on event structures, in which histories are partial orders of causal dependency between events. Their definition of strategy did not however accommodate disjunctive causes adequately. A way to overcome this limitation is described in this report. This involved the discovery of new structures as existing structures such as general event structures, while supporting disjunctive causes, failed to support an operation of hiding essential to the definition of composition of strategies.

*In this version of the report, almost no proof will be given, and a lot of properties found during the internship will not be presented. A complete version of the report (with more than fifty pages) can be found here : http ://www.cl.cam.ac.uk/ gw104/. The numeration of properties and definitions has been preserved between the two versions.*

The first section will define event structures, beginning with the simpler category : Prime Event Structures. Then, we will define General Event Structures with an equivalence relation. They give us to a global category which includes all the other categories presented in this report. After, we will talk about different useful subcategories, finishing with realisations. Realisations will be the tools for building the most important adjunction of this report (the adjunction between prime event structures with equivalence and general event

1. See [NPW81] for more informations at this subject.
2. In games semantics, pullbacks and hiding are used to define composition of strategies, and to give an abstract definition of strategies. Pullbacks correspond to a synchronisation of two objects (relatively to a third one).
3. This adjunction is a reflection, so no informations are lost from old representations to the new ones.

structure with equivalence). The second section will link all the categories we have introduced using adjunctions. The Figure 23 sum-up all the different adjunctions. The third section will describe some useful properties of our categories, such as the existence of pullbacks. The fourth section will present the basics of games semantics, and apply event structures to provide a general definition of strategy. In the complete version of the report, the last section is a compilation of examples and counterexamples discovered during this work and which have guided the choice of definitions which are needed to make all of this work.

# 1 Event Structures

## 1.1 Prime Event Structures

A prime event structure [4] is a way to represent a process, a game, a distributive algorithm, by a sets of events with causal dependencies and incompatibilities. Prime event structure are somewhat limited because they only support conjunctive dependencies, and non disjunctive ones. That is why we will introduce the notion of general event structure, and the notion of event structure with disjunctive causes.

**Definition 1.1** (Prime Event Structure)**.** *A PES $(E, \leq_E, Con_E)$ is a set of events $E$ with a partial order $\leq_E \subseteq E \times E$, and a consistency relation $Con_E \subseteq \mathcal{P}(E)$, such that :*

- (No inconsistent singleton) $\forall e \in E,\ \{e\} \in Con_E$
- (Independence) $\forall X \in Con_E,\ \forall Y \subseteq X,\ Y \in Con_E$
- (Continuous) $\forall X \subseteq E,\ X \in Con_E \iff \forall Y \subseteq_{finite} X,\ Y \in Con_E$
- (Down closed) $\forall X \in Con_E,\ \forall e \in X,\ \forall \tilde{e} \leq_E e,\ X \cup \{\tilde{e}\} \in Con_E$
- (Finite down-closure) $\forall e \in E,\ \{e' \mid e' \leq_E e\}$ *is finite.*

*For $e \in E$, we define*

- (Down-closure) $[e] = \{e' \mid e' \leq_E e\}$
- (Strict Down-closure) $[e) = \{e' \mid e' \leq_E e \ \&\ e' \neq e\}$

We represent event structures as oriented graphs (with extra information [5]). For example, the prime event structure $(\{A, B, C, D, E\}, \leq, Con)$ where $A, B \leq C \leq E$ and $Con = \{\ X \in \mathcal{P}(\{A, B, C, D, E\}) \mid D \in X \implies C \notin X \ \&\ E \notin X\ \}$ is represented as below :

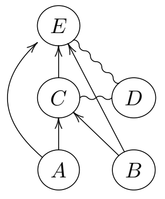

FIGURE 1 – Example of a Prime Event Structure.

In order to have shorter representations, some causal links and some inconsistency can be made implicit :

4. Generally named "event structure".
5. When the consistency exactly correspond to a binary inconsistency, we use a squiggly line to represent it.

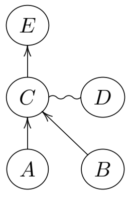

FIGURE 2 – Simplification of the Figure 1.

We now define configurations, they correspond to all possible states of the system.

**Definition 1.2** (Configurations)**.** *For a prime event structure* $(E, \leq_E, Con_E)$*, we define* $\mathcal{C}(E) \subseteq \mathcal{P}(E)$ *by* $X \in \mathcal{C}(E)$ *if :*
- (Consistent) $X \in Con_E$
- (Down closed) $\forall e \in X,\ [e] \subseteq X$

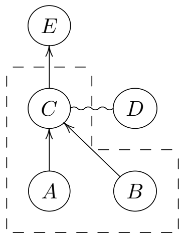

FIGURE 3 – Example of a configuration.

We have some immediate properties.

**Property 1.3.** *For a prime event structure* $(E, \leq_E, Con_E)$*, we have :*
- $\forall X \in Con_E,\ \exists Y \supseteq X,\ Y \in \mathcal{C}(E)$
- $\forall e \in E,\ [e] \in \mathcal{C}(E)\ \&\ [e) \in \mathcal{C}(E)$
- *A prime event structure is characterised by its configurations.*

We will now define maps of prime event structures, to have a category. The following definition say that it exists a total map from $E$ to $E'$ if you can go from $E$ to $E'$ by weakening causalities, strengthening consistency, merge inconsistent events, and introducing new events (such that previous events never depends of new events).

**Definition 1.4** (Map on prime event structures)**.** *A map between the prime event structure* $(E, \leq_E, Con_E)$ *and the prime event structure* $(E', \leq_{E'}, Con_{E'})$ *is a partial function* $f\ :\ \mathcal{D}(f) \subseteq E \to E'$ *such that :*
- (Locally Injective) $\forall X \in Con_E,\ \forall a, b \in X \cap \mathcal{D}(f),\ a \neq b \implies f(a) \neq f(b)$
- (Preserve Configurations) $\forall X \in \mathcal{C}(G),\ f(X) \in \mathcal{C}(G')$

*It is equivalent to :*
- (Locally Injective) $\forall X \in Con_E,\ \forall a, b \in X \cap \mathcal{D}(f),\ a \neq b \implies f(a) \neq f(b)$
- (Preserve Consistency) $\forall X \in Con_E,\ f(X) \in Con_{E'}$
- (Down closed Image) $\forall e' \in f(\mathcal{D}(f)),\ \forall \tilde{e}' \leq_{E'} e',\ \tilde{e'} \in f(\mathcal{D}(f))$
- (Reflects Order) $\forall e, \tilde{e} \in \mathcal{D}(f),\ f(\tilde{e}) \leq_{E'} f(e) \implies \tilde{e} \leq_E e$

In the Figure 4, the two events labelled $C$ are merged in one event (allowed because they are inconsistent), and the event $A$ is deleted, and a new event $F$ is created (allowed because the image is down-closed).

**Property 1.5** (Category of prime event structures)**.** *Prime event structures with their maps define a category.*

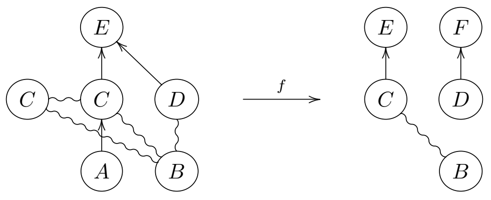

FIGURE 4 – Example of a map of $PES$.

A major restriction of prime event structures is that an event can only be enable in one way, but prime event structures have a lot of good properties, for example hiding of events does not lose informations, it mean :

**Definition 1.6** (Hiding of events). *Let $(E, \leq_E, Con_E)$ be a prime event structure, and $E' \subseteq E$. The restriction of $(E, \leq_E, Con_E)$ to $E'$ is $(E', \leq_{E'}, Con_{E'})$ with :*

- $\leq_{E'} = \leq_E$ *restricted to* $E'$
- $Con_{E'} = Con_E \cap \mathcal{P}(E')$

*It implies :*

$$\mathcal{C}(E') = \{X \cap E' \mid X \in \mathcal{C}(E)\}$$

*In other words, if we have a property on configurations of $E$ that use only events of $E'$ to be written (See Figure 5), this property is also true on configurations of $E'$.*

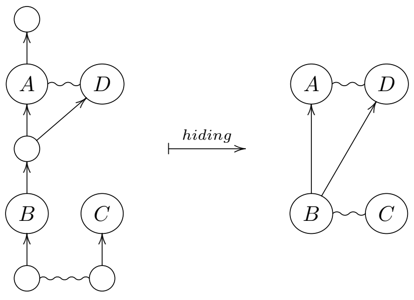

FIGURE 5 – The property "All configurations that contain the event $A$ contain also the event $B$" is preserved by hiding.

## 1.2 General Event Structures with an equivalence relation

In a prime event structure, only conjunctive enabling are allowed. We would want to have event structures where an event can be enable in different ways (General Event Structures, see Definition 1.17), or event structures where different events correspond to the same thing (Prime Event Structure with an equivalence relation, see Definition 1.19). These two methods allow us to have disjunctive enabling.
We will now consider a category which includes all the event structures that we will need, so allow both having equivalent events, and having multiple way of enabling an event. General Event Structures with an equivalence relation ($GES_{\equiv}$) are quite complicated, but having a global category into which we can embed all our models will be useful.

**Definition 1.7** (General Event Structure with an equivalence relation).
*A $GES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_G)$ is a set of events $G$, with a relation $\vdash_G \subseteq \mathcal{P}(G) \times G$, an*

*equivalence relation*[6] *(transitive, reflexive and symmetric)* $\equiv_G$, *and a consistency relation* $Con_G \subseteq \mathcal{P}(G)$ *such that :*

- (No inconsistent singleton) $\forall e \in G,\ \{e\} \in Con_G$
- (Independence of consistency) $\forall X \in Con_G,\ \forall Y \subseteq X,\ Y \in Con_G$
- (Continuous consistency) $\forall X \subseteq G,\ X \in Con_G \iff \forall Y \subseteq_{finite} X,\ Y \in Con_G$
- (Down closed consistency) $\forall X \in Con_G,\ \forall e \in X,\ \exists Y \vdash_G e,\ X \cup Y \in Con_G$
- (Generalisation of enabling) $\forall e \in G,\ \forall X \vdash_G e,\ \forall Y \supseteq X,\ Y \vdash_G e$
- (Finite enabling) $\forall e \in G,\ \forall X \vdash e,\ \exists Y \subseteq X,\ Y \vdash e\ \&\ Y$ *is finite.*

*We allow to have strange ways to enable events such as loops or not transitive (i.e down closed) enabling.*

- *For* $e \in G$, *we define* $\{e\}_{\equiv_G} = \{e' \mid e' \equiv_G e\}$
- *For* $X \subseteq G$, *we define* $X_{\equiv_G} = \{\{e\}_{\equiv_G} \mid e \in X\}$

*We say that two* $GES_\equiv$ *are isomorph if there exists a bijection between the two which preserves and reflects the enabling, the equivalence relation and the consistency.*

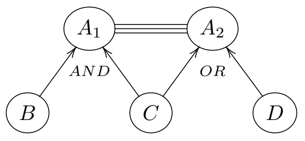

FIGURE 6 – Example of a simple $GES_\equiv$.

$GES_\equiv$ are a little too general, because we would prefer not having strange enabling. That why we will define replete $GES_\equiv$.

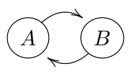

FIGURE 7 – Example of a non replete $GES_\equiv$.

**Definition 1.8** (Minimal enabling). *Let* $E$ *be a set of events, and* $\vdash_E \subseteq \mathcal{P}(E) \times E$. *We define* $\vdash^\mu_E \subseteq \mathcal{P}(E) \times E$ *as below :*

$$X \vdash^\mu_E e \iff \begin{cases} X \vdash_E e \\ \forall X \subseteq Y,\ Y \vdash_E \implies Y = X \end{cases}$$

**Definition 1.9** (Replete $GES_\equiv$). *We say that a* $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$ *is* replete *if :*

- (Minimal enabling without loops) $\forall e \in G,\ \forall X \vdash^\mu_G e,\ e \notin X$
- (Transitive Minimal Enabling) $\forall e \in G,\ \forall X \vdash_\mu e,\ \forall x \in X,\ X \vdash x$
- (Consistent minimal enabling) $\forall e \in G,\ \forall X \vdash^\mu_G e,\ X \in Con_G$

*The definition of a* $GES_\equiv$ *implies the property :*

- (Finite minimal enabling) $\forall e \in G,\ \forall X \vdash^\mu_G e,\ X$ *is finite*

*The notion of replete* $GES_\equiv$ *correspond to the notion of Families with an equivalence relation in the complete report.*

**Definition 1.10** (Configurations).
*For an* $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$, *we define* $\mathcal{C}(G) \subseteq \mathcal{P}(G)$ *by* $X \in \mathcal{C}(G)$ *if :*

- (Consistent) $X \in Con_G$
- (Secure chain)
  $\forall e \in X,\ \exists n \in \mathbb{N},\ \exists \{e_i\}_{1 \le i \le n},\ e_n = e\ \&\ \ \forall 1 \le i \le n,\ \{e_1, e_1, ...e_{i-1}\} \vdash_G e_i$

6. The $\equiv$ symbol of the notation $GES_\equiv$ correspond to the equivalence relation between maps of $GES_\equiv$, and not between events of an object of $GES_\equiv$.

*If $(G, \vdash_G, Con_G, \equiv_G)$ is a replete $GES_\equiv$, it is equivalent to :*
- (Consistent) $X \in Con_G$
- (Down closed) $\forall e \in X,\ X \vdash_G e$

*If $(G, \vdash_G, Con_G, \equiv_G)$ is a replete $GES_\equiv$, we have the immediate property :*

$$\forall X \in Con_G,\ \exists Y \supseteq X,\ Y \in \mathcal{C}(G)$$

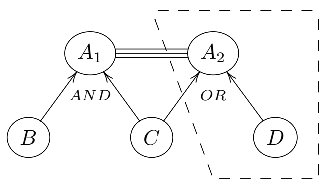

FIGURE 8 – Example of a configuration $GES_\equiv$.

To define a category, we also need maps.

**Definition 1.11** (Map on $GES_\equiv$)**.** *A map between the $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$ and the $GES_\equiv$ $(G', \vdash_{G'}, Con_{G'}, \equiv_{G'})$ is a partial function $f\ :\ \mathcal{D}(f) \subseteq G \to G'$ such that :*
- (All or Nothing) $\forall a \equiv_G b \in G,\ [\ a \in \mathcal{D}(f) \iff b \in \mathcal{D}(f)\ ]$
- (Preserve Equivalence) $\forall a \equiv_G b \in \mathcal{D}(f),\ f(a) \equiv_{G'} f(b)$
- (Locally *equiv*-Injective) $\forall X \in Con_G,\ \forall a, b \in X \cap \mathcal{D}(f),\ a \not\equiv_G b \implies f(a) \not\equiv_{G'} f(b)$
- (Preserve Configurations) $\forall X \in \mathcal{C}(G),\ f(X) \in \mathcal{C}(G')$

*Between replete $GES_\equiv$, the last property is equivalent to :*
- (Preserve Consistency) $\forall X \in Con_G,\ f(X) \in Con_{G'}$
- (Preserve Enabling) $\forall a \in \mathcal{D}(f),\ \forall X \vdash_G a,\ f(X) \vdash_{G'} f(a)$

*In all cases, [Preserve Consistency] and [Preserve Enabling] implies [Preserve Configurations], but they are not required for being a map.*
*We say that the function $f$ is a* quasi-map *of $GES_\equiv$ if it respect all property of a map, except the [All or Nothing] property.*

As on prime event structures, this definition means that a total map of $GES_\equiv$ allows to weaken causality, strengthen consistency, merge inconsistent equivalence classes, and introducing new events (such that previous events never depends of new events). Moreover, partial maps have to respect the equivalence relation ([All or Nothing] property).

**Definition 1.13** (Equivalence on map)**.**
*We will say that two maps of $GES_\equiv$ $f$ and $g$ are equivalent if they do the same thing up to equivalence. That means, if $f, g : (G, \vdash_G, Con_G, \equiv_G) \to (H, \vdash_H, Con_H, \equiv_H)$, then $f \equiv g$ if and only if :*
- $\mathcal{D}(f) = \mathcal{D}(g)$
- $\forall a \in \mathcal{D}(f),\ f(a) \equiv_H g(a)$

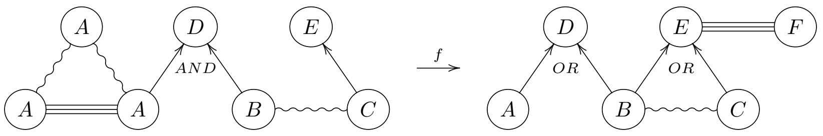

FIGURE 9 – Example of a map of $GES_\equiv$.

This equivalence relation says that only equivalence classes of events are really important, and that events are just different parts of the same "disjunctive event".

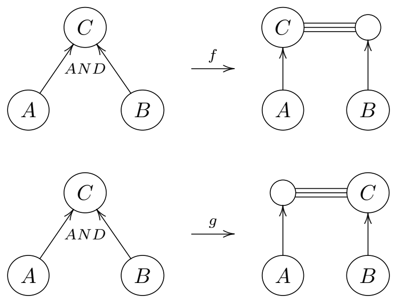

FIGURE 10 – Two equivalent maps.

**Property 1.14** ($GES_{\equiv}$ enriched category)**.** *$GES_{\equiv}$ with maps of $GES_{\equiv}$ is a category, and enriched category for the equivalence between maps. That means that the composition respect the equivalence relation.*
*In other words, $GES_{\equiv}$ with for maps* equivalence classes of maps of *$GES_{\equiv}$ is a category. We wrote $rGES_{\equiv}$ the enriched category corresponding to replete $GES_{\equiv}$.*

## 1.4 General Event Structures

As said before, the main restriction of prime event structures is that we cannot enable an event by different ways. A general event structure simply allow multiple enable.

**Definition 1.17** (General Event Structure)**.**
*A GES $(E, \vdash_E, Con_E)$ is a $GES_{\equiv}$ $(E, \vdash_E, Con_E, \equiv_E)$ with $\equiv_E$ being the equality. It implies that the equivalence between maps is also the equality. We define rGES as replete GES.*

A major problem of *GES* is that it does not work well with hiding of events. Some properties are lost under hiding :

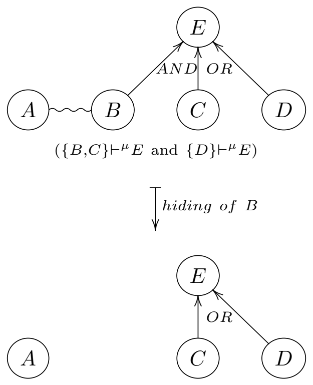

FIGURE 11 – The property "All configurations that contain $A$ and $E$ contain $D$" is lost under hiding.

## 1.5 Prime Event Structures with an equivalence relation, and Event structures with Disjunctive Causes

An other way of allowing having multiple way of enabling an event is by allowing us to duplicate an event into many equivalent events. Each of them corresponding to a way of enabling the initial event.

**Definition 1.19** (Prime Event Structure with an equivalence relation).
*A $PES_{\equiv}$ $(P, \leq_P, Con_P, \equiv_P)$ is a $rGES_{\equiv}$ $(P, \vdash_P, Con_P, \equiv_P)$ where :*
- (Partial order) $\leq_P \subseteq P \times P$ *is a partial order*[7]
- (Unique minimal enabling) $X \vdash_P e \iff [e) \subseteq X$

*If $\equiv_P$ is the equality, this definition exactly correspond to $PES$ (for objects and maps).*

As for prime event structures, $PES_{\equiv}$ work correctly with hiding. But some categorical constructions, such as pull-back, are not defined.[8] That is why we will add a property.

**Definition 1.20** (Event structure with Disjunctive Causes).
*An $EDC$ $(P, \leq_P, Con_P, \equiv_P)$ is a $PES_{\equiv}$ such that :*
- ($EDC$ property) $\forall p, p', q \in P, p \equiv p'$ & $p \leq_P q$ & $p' \leq_P q \implies p = p'$

*$EDC$ support hiding, and the Proposition 3.2 shows that it support pull-back, so it will be the category used for games and strategies (see Definition 4.3).*
*We can define some variants*[9] *of $EDC$ :*
- ($EDC^{weak}$) *$PES_{\equiv}$ with the property $\forall p, p' \in P, p \equiv p'$ & $p \leq_P p' \implies p = p'$*
- ($EDC^{not}$) *$PES_{\equiv}$ with the property $\forall p, p' \in P, p \equiv p'$ & $\{p, p'\} \in Con_P \implies p = p'$*

## 1.6 Extremal realisation

$GES$ and $EDC$ are two different way of representing events that can be enable in different ways, we would want to pass from one way to the other. That is why we will build an adjunction.[10] To do this, we need to define what a realisation of a $GES_{\equiv}$ is.

### 1.6.1 Partially Ordered Multisets

First, we need to talk about partially ordered multisets. Realisation will be partially ordered multisets linked in a good way to a $GES_{\equiv}$.

**Definition 1.21** (Partially Ordered Multisets). *A $POM$ $(R, \leq_R, n_R)$ on a set G, is a $PES_{\equiv}$ $(R, \leq_R, Con_R)$ where*[11]
- (Trivial Consistency) $Con_R = \mathcal{P}(R)$
- (Name function) *The* name function *$n_R : R \to G$ is a total function*
- (Same-name equivalence) $\forall a, b \in R, a \equiv_R b \iff n_R(a) = n_R(b)$

*We say that two $POM$ are isomorph if there exists a bijection between the two which preserves and reflects both the order*[12] *and the equivalence relation, and which respect the name function.*[13]

**Definition 1.22.** *For $(R, \leq_R, n_R)$ a $POM$, we define :*
- (Down-closure) $[p] = \{q \mid q \leq_R p\}$
- (Strict Down-closure) $[p) = \{q \mid q \leq_R p$ & $q \neq p\}$
- (Top) *$Top(Y) = p$ such that $[p] = Y$ (not always defined)*
- (Top $POM$) *When $Top(R)$ is defined, we say that $(R, \leq_R, n_R)$ is a top $POM$*

To be able to talk later about extremal realisations, we need to put an order between $POM$. In fact, we will have a pre-order and a partial order, the first affect mainly the internal partial order, and the second preserves the internal partial order but change the elements.

7. So transitive, reflexive, and anti-symmetric. We recall that $[e] = \{e' \mid e' \leq_P e\}$ and $[e) = [e] \backslash \{e\}$.
8. But, because we have pull-back on $GES_{\equiv}$, the $\equiv$-adjunction given by the Theorem 2.13 say that we have by-pull-back (so pull-back up to equivalence).
9. We will not talk a lot about them.
10. In fact, we will not be able to build an adjunction, it will only be an $\equiv$-adjunction.
11. An important point is that realisation does *not* necessarily have the $EDC$ property.
12. We will frequently say "order" instead of "partial order".
13. It mean that the image of an element by the bijection has the same name as the element.

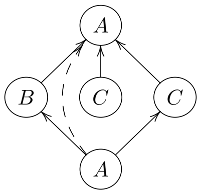

FIGURE 12 – Example of a $POM$ on $\{A,B,C\}$ (The dash arrow is an implicit arrow).

**Definition 1.24** (A pre-order on $POM$)**.** *For two POM* $(R, \leq_R, n_R)$ *and* $(R', \leq_{R'}, n_{R'})$ *on the same set G, we write* $(R, \leq_R, n_R) \preceq_{fun} (R', \leq_{R'}, n_{R'})$*, if there exists a total, surjective map of* $PES_{\equiv}$ *between the* $PES_{\equiv}$ *associated to* $(R, \leq_R, n_R)$ *and the* $PES_{\equiv}$ *associated to* $(R', \leq_{R'}, n_{R'})$ *which respect the name function. More precisely, it mean there exists* $f : R' \to R$ *such that :*

- (Total) $\forall p' \in R',\ f(p')$ *is defined*
- (Surjective) $\forall p \in R,\ \exists p' \in R',\ p = f(p')$
- (Respect the same function) $\forall p' \in R', n_{R'}(p') = n_R(f(p'))$
- (Reflects order) $\forall q' \in R,\ \forall p' \leq_{R'} q',\ f(p') \leq_R f(q')$

*It define a pre-order*[14] *on* $POM$*. We remark that two isomorphic* $POM$ *are on the same cycle*[15] *for* $\preceq_{fun}$*.*

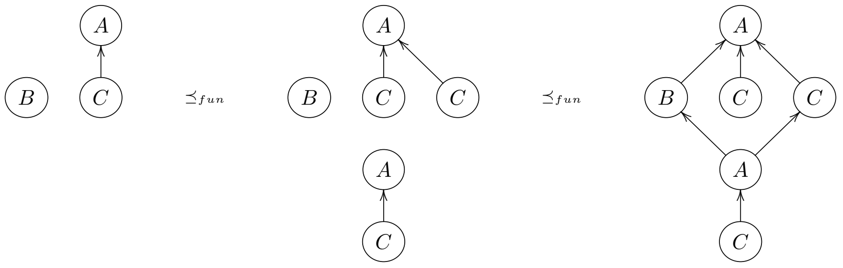

FIGURE 13 – Example of a sequel of $\preceq_{fun}$

**Definition 1.25** (Sub-structure)**.**
*For two POM* $(R, \leq_R, n_R)$ *and* $(R', \leq_{R'}, n_{R'})$ *on the same set G, we say that* $(R, \leq_R, n_R)$ *is a sub-structure of* $(R', \leq_{R'}, n_{R'})$*, and we write* $(R, \leq_R, n_R) \preceq_{sub} (R', \leq_{R'}, n_{R'})$*, if there exists a partial and surjective function*[16] $m : \mathcal{D}(m) \subseteq R' \to R$ *which is a mono-morphism, which mean that :*

- (Injective) $m$ *is injective.*
- (Respect the same function) $\forall p' \in \mathcal{D}(m), n_{R'}(p') = n_R(m(p'))$
- (Down-closed partiality) $\forall q' \in D(m),\ \forall p' \leq_{R'} q',\ p' \in D(m)$
- (Preserve and reflect order)
  $\forall p', q' \in \mathcal{D}(m),\ p' \leq_{R'} q' \iff p' \in \mathcal{D}(m)\ \&\ m(p') \leq_R m(q')$

*This definition means that, up to isomorphism, R is include in* $R'$*, is down-closed by* $\leq_{R'}$*, and has the same pre-order and the same equivalence relation.*

14. We recall that a pre-order is a transitive and reflexive binary relation.
15. Cycles of a pre-order are usually called "equivalence classes", but we will not use this term to avoid confusion with equivalence classes of $GES_{\equiv}$.
16. $m$ does not define a map of $PES_{\equiv}$ because it has not the [All or Nothing] property. However, $m^{-1}$ is a map of $PES_{\equiv}$, and a total mono-morphism.

*$\preceq_{sub}$ define a partial order up isomorphism.* [17]

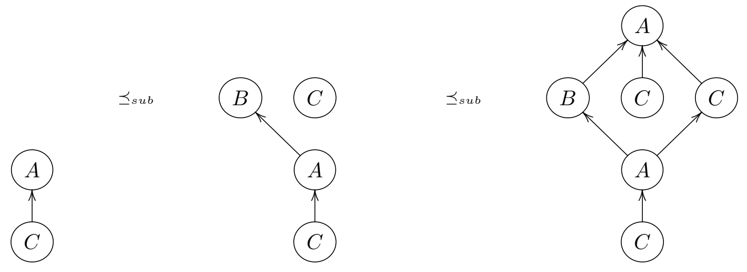

FIGURE 15 – Example of a sequel of $\preceq_{sub}$

The order $\preceq_{sub}$ allow to define restrict a $POM$ to one element and its down closure.

We will now merge these two pre-orders.

**Definition 1.27** (The pre-order $\preceq$)**.** *We define $\preceq$ as the transitive (and reflexive) closure of the union of $\preceq_{fun}$ and $\preceq_{sub}$.*

**Proposition 1.28** (Decomposition of $\preceq$)**.**

$$\begin{aligned} &(R, \leq_R, n_R) \preceq (T, \leq_T, n_T) \\ \iff & \exists (S, \leq_S, n_S),\ (R, \leq_R, n_R) \preceq_{sub} (S, \leq_S, n_S) \preceq_{fun} (T, \leq_T, n_T) \\ \iff & \exists (\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}}),\ (R, \leq_R, n_R) \preceq_{fun} (\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}}) \preceq_{sub} (T, \leq_T, n_T) \end{aligned}$$

*So it give the diagram :*

$$\begin{array}{ccc} R & \preceq_{sub} & S \\ \preceq_{fun} & & \preceq_{fun} \\ \tilde{S} & \preceq_{sub} & T \end{array}$$

We define also perfect $POM$. They are just $POM$ with no useless duplication of elements.

**Definition 1.29** (Perfect $POM$)**.** *We say that a $POM$ $(R, \leq_R, n_R)$ is perfect if :*

- (No redundancy) $\forall p, q \in R,\ \left[ \begin{cases} n_R(p) = n_R(q) \\ [p) \subseteq [q) \end{cases} \implies p = q \right]$

*We can deduce from the [No redundancy] property :*

- (No need itself) $\forall p, q \in R,\ \left[ \begin{cases} n_R(q) = n_R(p) \\ p \leq q \end{cases} \implies p = q \right]$

The main good property of prefect $POM$ is that there is only a finite number of prefect $POM$ (on a finite set)

**Property 1.30** (Bounded number of perfect $POM$)**.** *For all $E$ a finite set, exits a finite number of perfect $POM$ on $E$, up to isomorphism.*

### 1.6.2 Realisation

Now that we have define $POM$, we can link them to a $GES_{\equiv}$.

17. A order up to isomorphism is an order between the isomorphism classes. Equivalently, it is a pre-order $\leq$ which is antisymmetric up to isomorphism : $x \leq y$ & $y \leq x \implies x$ is isomorphic to $y$.

**Definition 1.31** (Realisation)**.** *Let $(G, \vdash_G, Con_G, \equiv_G)$ be a $GES_\equiv$. We say that a $POM$ $(R, \leq_R, n_R)$ on $G$ is a realisation of $(G, \vdash_G, Con_G, \equiv_G)$ if the name function $n_R$ define a total map of $GES_\equiv$ between the $GES_\equiv$ associated to $(R, \leq_R, n_R)$ and $(G, \vdash_G, Con_G, =)$ , which mean*[18] *that :*

- (Realisation) $\forall p \in R,\ n([p]) \in \mathcal{C}(G)$

*$n_R$ is the* name function *of the realisation $(R, \leq_R, n_R)$.*

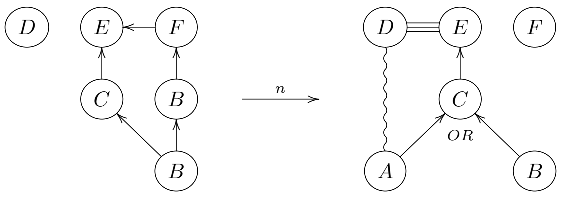

Figure 16 – Example of a realisation

This definition works correctly with the pre-order and the partial order defined on $POM$, but not in the same direction. Realisations are preserved by increasing along $\preceq_{fun}$ and decreasing along $\preceq_{sub}$, so we have no properties for $\preceq$.

Because $GES_\equiv$ is a category, a natural things to do is defining the image of a realisation by a map of $GES_\equiv$. This image has good properties for the sub-structure order.

**Definition 1.34** (Image of realisations)**.**
*For $f : (G, \vdash_G, Con_G, \equiv_G) \to (H, \vdash_H, Con_H, \equiv_H)$ a map of $GES_\equiv$, and for $(R, \leq_R, n_R)$ a realisation of $(G, \vdash_G, Con_G, \equiv_G)$, we define $f@(R, \leq_R, n_R) = (S, \leq_S, n_S)$ a realisation of $(H, \vdash_H, Con_H, \equiv_H)$ as :*

- $S = R$
- $\forall r, r' \in S,\ [\ r \leq_S r' \iff r \leq_R r'\ ]$
- $\forall r \in S, n_S(r) = f(n_R(r))$

*So only the name function change.*

*Proof.* $(S, \leq_S, n_S)$ is a realisation because $f$ preserves configurations. □

The notion of perfect $POM$ correspond to a notion of $POM$ which have no strange things. We would want to only manipulate only perfect $POM$, that why we would want to always be able to extract a perfect realisation from a realisation.

**Proposition 1.36** (Perfect Realisations)**.** *If $(R, \leq_R, n_R)$ is a realisation of the $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$, then there exists $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (R, \leq_R, n_R)$ which is a perfect realisation of $(G, \vdash_G, Con_G, \equiv_G)$.*

*Proof.* We will first create $\leq''$ such that $(R, \leq'', n_R)$ respects the [Weak No Redundancy] property :

$$\forall p, q \in R,\ \left[ \begin{cases} n_R(p) = n_R(q) \\ [p) \subseteq [q) \end{cases} \implies [p) = [q) \right]$$

Then, we will merge elements with the same down-closure to have $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (R, \leq'', n_R)$ a perfect realisation.
We can simply define $\leq''$ :

18. An important point is that we forget the equivalence relation $\equiv_G$.

- $\forall p \in R,\ \exists \tilde{p} \in R,\ \begin{cases} n_R(\tilde{p}) = n_R(p) \\ [\tilde{p}) \subseteq [p) \\ \begin{cases} n_R(q) = n_R(\tilde{p}) \\ [q) \subseteq [\tilde{p}) \end{cases} \implies [q) = [\tilde{p}) \end{cases}$
- $e \leq'' p \iff e \leq_R \tilde{p}$

$\tilde{p}$ is well defined because $R$ has finite down-closure (so $[p]$ is finite). We trivially preserves the realisation property. The merge cause no problems too. □

Now, we will define the notion of extremal [19] realisation. They are minimal realisations for the order $\preceq_{fun}$. The minimum for the partial order $\preceq_{sub}$ and for the pre-order $\preceq$ are the void realisation, so it is not interesting.

**Definition 1.37** (Extremal Realisations). *We sat that $(R, \leq_R, n_R)$ is an* extremal *realisation of the $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$ if for all other realisations $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (R, \leq_R, n_R)$, we have $(R', \leq_{R'}, n_{R'}) \succeq_{fun} (R, \leq_R, n_R)$.*
*We sat that $(R, \leq_R, n_R)$ is an* unambiguous extremal *realisation of the $GES \equiv (G, \vdash_G, Con_G, \equiv_G)$ if for all other realisation $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (R, \leq_R, n_R)$, we have $(R', \leq_{R'}, n_{R'})$ is isomorphic to $(R, \leq_R, n_R)$.*
*Equivalently, unambiguous extremal realisation are extremal realisation such that all realisation of its cycle (for $\preceq_{fun}$) are isomorph.*
*Extremal realisation can be infinite.*
*The Proposition 1.38 shows that all extremal realisations are unambiguous.*
*The Proposition 1.36 shows that extremal realisations are perfects.*

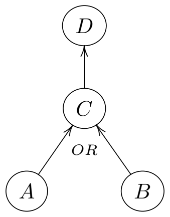

FIGURE 17 – A $GES_\equiv$.

**Proposition 1.38** (Extremal realisations are unambiguous).
*An extremal realisation $(R, \leq_R, n_R)$ of a $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$ is an unambiguous extremal pre-realisations.*

We want to be able to extract top extremal realisations from any realisations, the following property says that it is always possible.

**Proposition 1.39** (Existence of top extremal realisation). *For all realisations $(R, \leq_R, n_R)$, for all event $e \in n_R(R)$ of this realisation, there exists an extremal realisation $(T, \leq_T, n_T) \preceq (R, \leq_R, n_R)$ which has a top $t$ with $n_T(t) = e$.*

*Proof.* We first deduce a top realisation with top $p \in n_R^{-1}(e)$ by taking the top sub-structure $(\tilde{R}, \leq_{\tilde{R}}, n_{\tilde{R}})$ defined by :
- $\tilde{R} = [p]$
- $\leq_{\tilde{R}} = \leq_R$ restricted to $\tilde{R}$
- $n_{\tilde{R}} = n_R$ restricted to $\tilde{R}$

Then, by the Proposition 1.36, we can take $(S, \leq_S, n_S) \preceq_{fun} (\tilde{R}, \leq_{\tilde{R}}, n_{\tilde{R}})$ a perfect realisation.
We are now in finite cases (see below), so we can do the following algorithm :

19. We use the term *extremal* and no *minimal* because the pre-order $\preceq_{fun}$ correspond to the existence of a function from the greater element to the lesser, which is the contrary of what is usually done.

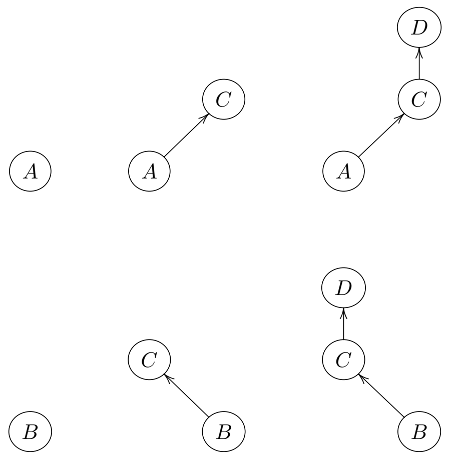

FIGURE 18 – All top extremal realisations of the $GES_{\equiv}$ of the Figure 17

- Either $(S, \leq_S, n_S)$ is extremal.

$\implies$ End of the algorithm.

- Either exists a realisation $(P, \leq_P, n_P) \preceq_{fun} (S, \leq_S, n_S)$ which is not in the same cycle (of the pre-order $\preceq_{fun}$), and with $p \in n(P)$.

$\implies$ By the Proposition 1.36, we can take $(P, \leq_P, n_P)$ perfect.
$\implies$ Go to the beginning with $(P, \leq_P, n_P)$ instead of $(S, \leq_S, n_S)$.

We know that there is a finite number of perfect realisation on $n_R(R)$ (Property 1.30), so the algorithm will end. So we produce a top extremal realisation $(T, \leq_T, n_T) \preceq (R, \leq_R, n_R)$ with $n_T(Top(T)) = e$. $\square$

**Proposition 1.43** (Characterisation of extremal realisations). *A POM $(R, \leq_R, n_R)$ on $G$ is an extremal realisation of the $GES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_G)$ if and only if :*

- (Realisation) $\forall r \in R,\ n_R([r)) \vdash_G n_R(r)$
- (Minimal) $\forall r \in R,\ \forall X \subseteq [r), \begin{cases} X \textit{ down closed for } \leq_R \\ n_R(X) \vdash_G n_R(r) \end{cases} \implies X = [r)$
- (No multiplicity) $\forall p, q \in R, \begin{cases} n_R(p) = n_R(q) \\ [p) = [q) \end{cases} \implies p = q$

# 2 The $\equiv$-adjunction between *GES* and *EDC*

*GES* correspond to the common way of adding disjunctive enabling to event structures, it support pull-backs, but does not support hiding, whereas *EDC* support both pull-backs and hiding. That is why we would want a way to pass from one to the other.
In this section, we will build an $\equiv$-adjunction (see Definition 2.9) between these two categories by composing multiple little $\equiv$-adjunctions.

## 2.1 The adjunction between *GES* and *rGES*

**Definition 2.1** (Adjunction)**.** *For $\mathcal{A}$ and $\mathcal{B}$ two categories, $L : \mathcal{A} \to \mathcal{B}$ and $R : \mathcal{B} \to \mathcal{A}$ two functors, we said that L and R define an adjunction between $\mathcal{A}$ and $\mathcal{B}$, and we wrote $L \dashv R$, if for all $A \in \mathcal{A}$, and for all $B \in \mathcal{B}$, there is a one-to-one correspondence between maps $L(A) \to B$ and maps $A \to R(B)$.*
*If $\mathcal{A}$ and $\mathcal{B}$ are two categories enriched by an equivalence relation, then we say that there is an enriched adjunction if L and R preserve the equivalence relation.*

An enriched adjunction correspond to an adjunction which is also a $\equiv$-adjunction (see Definition 2.9). An adjunction is always a enriched adjunction with the equality as the equivalence relation.

**Proposition 2.2** (Adjunction between *GES* and *rGES*)**.** *There is an adjunction between GES and rGES. The right adjoint is the inclusion functor, and the left adjoint correspond to "completing by transitivity the enabling relation and deleting meaningless way of enabling".*

## 2.2 The enriched adjunction between *rGES* and $rGES_\equiv$

What we want is a functor which collapse equivalence classes by adding disjunctive enabling. We will here describe the abstract way of defining *col*, but there is an inductive way of defining it, see the complete report for more details.

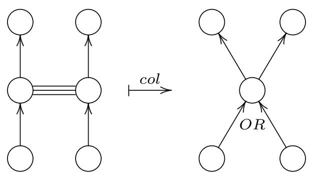

FIGURE 19 – Simple example of the effect of the *col* functor on a replete $GES_\equiv$

**Definition 2.5** (The *col* functor)**.**

$$(G, \vdash_G, Con_G, \equiv_G) \xmapsto{col} (E, \vdash_E, Con_E)$$

*where*
- $E = G_{\equiv_G}$
- $\forall x \in E,\ \forall X \subseteq E,\ [\ X \vdash_E x \iff \exists y \in G,\ \exists Y \subseteq G,\ Y \vdash_G y\ \&\ \{y\}_{\equiv_G} = x\ \&\ Y_{\equiv_G} = X\ ]$
- $X \in Con_E \iff \exists Y \in Con_G,\ Y_{\equiv_G} = X$

*and*

$$(f : G \to G') \xmapsto{col} (g : E \to E')$$

*where*
- (1) $g(e) = e' \iff \exists p \in G$ *such that* $\{p\}_{\equiv_G} = e,\ \{f(p)\}_{\equiv_G} = e'$
- (2) $g(e) = e' \iff \forall p \in G$ *such that* $\{p\}_{\equiv_G} = e,\ \{f(p)\}_{\equiv_G} = e'$

*This functor respect naturality conditions, and is an enriched functor (for the equivalence relation).*

**Theorem 2.8** (The adjunction between $rGES$ and $rGES_\equiv$)**.** *The inclusion functor* $\mathcal{I}$[20] *and col define an enriched adjunction between rGES and* $rGES_\equiv$*, more precisely col* $\dashv \mathcal{I}$*. It mean that for* $(E, \vdash_E, Con_E)$ *a rGES, and* $(G, \vdash_G, Con_G, \equiv_G)$ *a* $rGES_\equiv$*, we have*[21] *:*

$$\forall f : col(G) \to E,\ \exists! h : G \to \mathcal{I}(E),\ f = col(h)$$

*(More rigorously* $f = r_E \circ col(h)$*, where* $r_E : col \circ \mathcal{I}(E) \to E$ *is the isomorphism)*

## 2.3 The $\equiv$-adjunction between $rGES_\equiv$ and $PES_\equiv$

What we want is a functor which replace events that can be enable in different way by equivalent events that can be enable in a unique way. We will here describe the abstract way of defining *ter*, but there is (under some restrictions) an inductive way of defining it, see the complete report for more details.

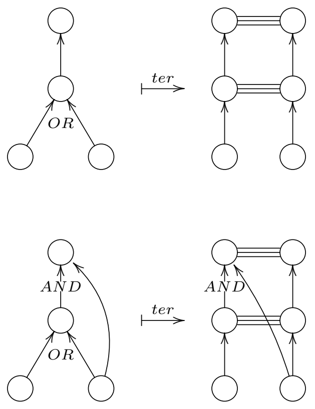

FIGURE 20 – Simple examples of the effect of the *ter* functor on a replete $GES_\equiv$

**Definition 2.9** (Pseudo-functor and $\equiv$-adjunction)**.** *We take A and B two categories enriched by an equivalence relation on maps. We define* $A/_\equiv$ *the category which have the objects of A, and for maps the equivalence classes of maps of A. We define* $B/_\equiv$ *in a same way.*
*A* pseudo-functor $f : A \to B$ *is a functor from* $A/_\equiv$ *to* $B/_\equiv$*.*
*An* $\equiv$-adjunction *is an adjunction between* $A/_\equiv$ *and* $B/_\equiv$*.*

We will frequently assimilate a function and its equivalence classes, or implicitly take an arbitrary element of an equivalence classes of functions.

**Definition 2.10** (The *ter*[22] pseudo-functor)**.** *The pseudo-functor ter :* $rGES_\equiv \to PES_\equiv$ *is defined as below :*

$$(G, \vdash_G, Con_G, \equiv_G) \xmapsto{ter} (P, \leq_P, Con_P, \equiv_P)$$

20. We use the same name $\mathcal{I}$ for all inclusions functor. Because they have no effects on objects or maps, it is not a problem.

21. We recall that the equivalence on maps of *rGES* is the equality. Moreover, equivalence classes on maps from a $rGES_\equiv$ to a $rGES_\equiv$ which come from a *rGES*, have cardinality one. So this property is also true up to equivalence.

22. *ter* means "top extremal realisations".

- $P = \{(R, \leq_R, n_R)$ *top extremal realisation of* $G$ $\}$[23]
- $(R, \leq_R, n_R) \leq_P (S, \leq_S, n_S) \iff (R, \leq_R, n_R) \preceq (S, \leq_S, n_S)$
- $(R, \leq_R, n_R) \equiv_P (S, \leq_S, n_S) \iff n_R(Top(R)) = n_S(Top(S))$
- $X \in Con_P \iff \exists Y \in \mathcal{C}(G), \bigcup_{(R,\leq_R,n_R)\in X} n_R(R) \subseteq Y$

*and*

$$\{f : \mathcal{D}(f) \subseteq G \to G'\}_{\equiv} \xmapsto{ter} \{g : \mathcal{D}(g) \subseteq P \to P'\}_{\equiv}$$

*where*
- (Same partiality) $(R, \leq_R, n_R) \in \mathcal{D}(g) \iff n_R(Top(R)) \in \mathcal{D}(f)$
- (Image) $g((R, \leq_R, n_R)) = (R', \leq_{R'}, n_{R'}) \preceq f@(R, \leq_R, n_R)$
- (Respect the name function) $n_{R'}(Top(R')) = f(n_R(Top(R)))$
- (Coherent choices) $g$ *is a map of* $PES_{\equiv}$

*This pseudo-functor respect naturality conditions.*

**Theorem 2.13** (The $\equiv$-adjunction between $rGES_{\equiv}$ and $PES_{\equiv}$)**.** *$\mathcal{I}$ and ter define an $\equiv$-adjunction between $rGES_{\equiv}$ and $PES_{\equiv}$, more precisely $\mathcal{I} \dashv ter$ up to equivalence.*
*It mean that for $(P, \leq_P, Con_P, \equiv_P)$ a $PES_{\equiv}$, and $(G, \vdash_G, Con_G, \equiv_G)$ a $rGES_{\equiv}$, and up to equivalence, we have :*

$$\forall f : P \to ter(G),\ \exists! h : \mathcal{I}(P) \to G,\ f \equiv ter(h)$$

*(More rigorously $f \equiv \tilde{h} \circ r_P$, where $r_P : P \to ter \circ \mathcal{I}(P)$ is an isomorphism, and $\tilde{h}$ and element of $ter(\{h\}_{\equiv})$)*

**Proposition 2.15** (Extremal Realisations are Configurations)**.** *Extremal realisations of a $rGES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_G)$, exactly correspond to configurations of $ter(G)$.*
*That implies that if we define : $Top_G : ter(G) \to G : (R, \leq_R, n_R) \mapsto n_R(Top(R))$ , then :*

$$X \in \mathcal{F} \iff \exists Y \in \mathcal{C}(ter(G)),\ Top_G(Y) = X$$

## 2.4 The enriched adjunction between $PES_{\equiv}$ and $EDC$

**Proposition 2.17** (The enriched adjunction between $PES_{\equiv}$ and $EDC$)**.** *The inclusion functor $\mathcal{I} : EDC \to PES_{\equiv}$ and the restriction functor $restr : PES_{\equiv} \to EDC$ define an enriched adjunction between $PES_{\equiv}$ and $EDC$, more precisely $\mathcal{I} \dashv restr$.*

$$restr : (P, \leq_P, Con_P, \equiv_P) \mapsto (P', \leq_{P'}, Con_{P'}, \equiv_{P'})$$

*Where $P' = \{p \in P \mid \forall q \leq_P p,\ \forall q' \leq_P p,\ q \equiv_P q' \iff q = q'\}$, and $\leq_{P'}$, $Con_{P'}$, and $\equiv_{P'}$ are the restriction of $\leq_P$, $Con_P$ and $\equiv_P$ to $P'$.*

$$restr : (f : P \to Q) \mapsto (g : P' \to Q')$$

*Where g is the restriction of $f$ to $P'$*

In a similar way, there is a sequence of enriched adjunction between $PES_{\equiv}$, $EDC^{weak}$, $EDC$, $EDC^{not}$, and an adjunction[24] (not enriched) between $EDC^{not}$ and $PES$.

## 2.5 The composite adjunction

The Figure 23[25] sum-up all the precedent adjunctions. Some of them still work if we take relations[26] instead of functions.
Because the adjunction between $EDC$ (or $EDC^{not}$) and $PES$ is not enriched, and the

23. Quotiented by the being-isomorphic equivalence relation.
24. The right adjoint is "forgetting the equivalence relation", and the left adjoint is the inclusion functor.
25. This figure use colors in order to be more readable.
26. See the complete report for more details.

≡-adjunction between $rGES_{\equiv}$ and $PES_{\equiv}$ is not an adjunction, we cannot deduce an ≡-adjunction (nor an adjunction) between $PES$ and $GES$.

Most of the immediate adjunctions (or ≡-adjunction) are trivially reflection[27] or co-reflection[28], but the fact that the adjunction between $rGES$ and $EDC$ (or $PES_{\equiv}$) is a reflection is more complicated.

**Proposition 2.18** (The ≡-adjunction between $rGES$ and $EDC$ is a reflection)**.** *The co-unit of the ≡-adjunction between rGES and EDC is an isomorphism :* $\epsilon_E : col \circ \iota \circ \iota \circ ter \circ \mathcal{I}(E) \to E$ *(where* $\iota$ *correspond to the composition of different inclusion functor and restriction functors)*
*It mean that if we take a replete* $GES$*, then we take the* $EDC$ *corresponding to all top extremal realisations that respect the* $EDC$ *property, and then collapse equivalent events, we obtain a* $GES$ *isomorphic to the initial* $GES$*.*
*Similarly, the the co-unit of the ≡-adjunction between rGES and* $PES_{\equiv}$ *is also an isomorphism.*

The precedent property say, approximatively, that anything that can be expressed with a replete $GES$ can be expressed with a $PES_{\equiv}$ (and also with an $EDC$). We can find a characterisation of $PES_{\equiv}$ coming from $rGES$. See the complete report for the characterisation.

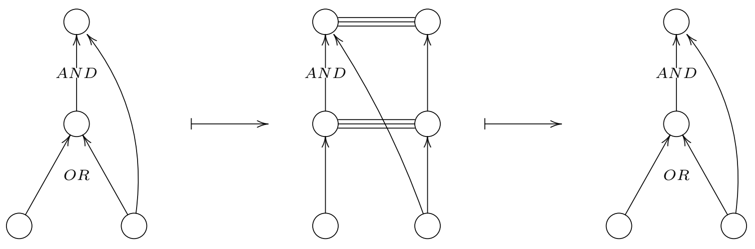

FIGURE 21 – Example of the co-unit being an isomorphism

27. If you take an object, apply the right adjoint, then apply the left adjoint, and obtain something isomorphic to the initial object, then the adjunction is a reflection.
28. If you take an object, apply the left adjoint, then apply the right adjoint, and obtain something isomorphic to the initial object, then the adjunction is a co-reflection.

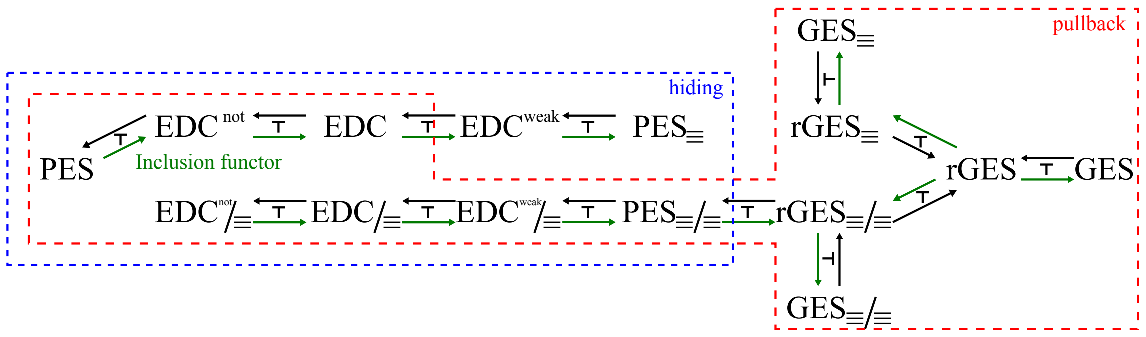

FIGURE 23 – The composite adjunction.

# 3 More properties on Event Structures

## 3.1 Pull-back

A pull-back is a categorical construction used to synchronise two objects (relatively to a third one). In the case of event structures, a pull-back of two event structures [29] correspond to a superposition of the constrains (causal dependencies, inconsistency, ...).
We will see in the Definition 4.1 how to use event structures in order to represent games and strategies. Pull-backs and hiding are needed in order to define the notion of composition of strategies.

**Definition 3.1** (Pull-back). *Let $A$, $B$, $C$, and $P$ be four objects of a same category $\mathcal{C}$. Let $f_A : A \to C$, $f_B : B \to C$, $g_A : P \to A$ and $g_B : P \to B$ be four morphisms.* [30] *We say that $(P, g_A, g_B)$ is the pull-back of $(A, f_A)$ and $(B, f_B)$ relatively to $C$ if :*

- $f_A \circ g_A = f_b \circ g_B$
- $\forall (P', g'_A, g'_B)$ *such that* $f_A \circ g'_A = f_b \circ g'_B$, $\exists ! \varphi : P' \to P$, $\begin{cases} g_A \circ \varphi = g'_A \\ g_B \circ \varphi = g'_B \end{cases}$

$$\begin{tikzcd}
& \forall P' \arrow[d, dashed, "\exists ! \varphi"'] \arrow[ddl, bend right, "g'_B"'] \arrow[ddr, bend left, "g'_A"] & \\
& P \arrow[dl, "g_B"] \arrow[dr, "g_A"'] & \\
A \arrow[dr, "f_A"'] & & B \arrow[dl, "f_B"] \\
& C &
\end{tikzcd}$$

*When the pull-back $(P, g_A, g_B)$ exists, it is unique (up to isomorphism).*
*If $\mathcal{C}$ is an enriched category for $\equiv$, we define bi-pull-packs as pull-backs in $\mathcal{C}/_{\equiv}$. Bi-pull-backs are unique up to equivalence* [31]

**Property 3.2** (Existence of pull-backs). *We consider a category $\mathcal{C} \in \{GES$, $rGES$, $GES_{\equiv}$, $rGES_{\equiv}$, $PES_{\equiv}$, $EDC^{weak}$, $EDC$, $EDC^{not}$, $PES\}$.*
*$\mathcal{C}$ has bi-pull-backs.*
*If $\mathcal{C} \neq PES_{\equiv}$ and $\mathcal{C} \neq EDC^{weak}$, then $\mathcal{C}$ has pull-backs.*
*We recall that pull-backs (and bi-pull-backs) are preserved by right adjoint.*

## 3.2 Relations instead of functions

A lot of things also work with relations instead of functions. See the complete report for more informations.

29. The third event structure correspond to the part that is common between the two others.
30. When the category has a notion of total morphism and partial morphism, only total morphisms are taken.
31. An equivalence relation between morphisms induce an equivalence relation between objects : $A \equiv B \iff \exists f : A \to B,\ \exists g : B \to A,\ g \circ f \equiv id_A$.

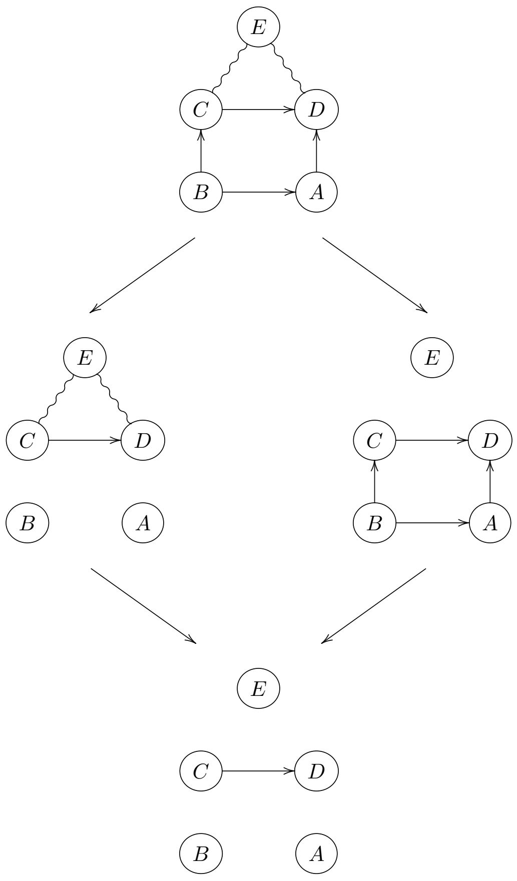

FIGURE 24 – Example of a pull-back on prime event structures.

# 4 Games and Strategies

We will use $PES$ in order to models games, and $EDC$ to models strategies [32]. It is an extension of [RW11], which was using only $PES$. We will define strategies as pre-strategies which are stable by composition by the copy-cat strategy, and deduce from this abstract definition all intuitive properties of a strategy. [33]

## 4.1 Games and pre-strategies

We use $PES$ to models games. Events correspond to player (polarity $\oplus$) or opponent (polarity $\ominus$) moves. The partial order and the consistency correspond to rules of the game.

**Definition 4.1** (Game)**.**
*A game is $PES$ $(A, \leq_A, Con_A)$ with a polarity function $p : A \rightarrow \{\oplus, \ominus\}$.*

**Definition 4.2.**
*For a game $A$, the dual game $A^\perp$ corresponds to the game with reversed polarities.*
*For two games $A$ and $B$, the parallel game $A \mid\mid B$ corresponds to the disjoint union of the two games.*

A strategy corresponds to a set of restriction that the player put on his own moves. He can, for example, choose to never do a particular move. Intuitively, we know that the player is not allowed to restrict opponent moves [34], but determining what is exactly allowed is not simple. Pre-strategy allow any kind of restrictions, and we will define strategies as pre-strategies that have good properties.

**Definition 4.3** (Pre-strategy)**.**
*We say that $(S, \sigma)$ is a pre-strategy on the game $A$ if $S$ is an $EDC$ and $\sigma : S \rightarrow A$ is a map [35] of $EDC$ which respect polarities.*

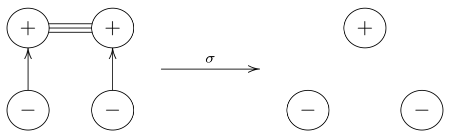

FIGURE 28 – Example of a pre-strategy $\sigma : S \rightarrow A$

## 4.2 Composition of pre-strategies

We will define the notion of "pre-strategy from one game to another". They can be seen as compiler which translate any pre-strategy of the first game in a pre-strategy of the second game.

**Definition 4.4** (Pre-strategy from one game to another)**.** *We say that $(S, \sigma)$ is a pre-strategy from the game $A$ to the game $B$ if $(S, \sigma)$ is a pre-strategy on the game $A^\perp \mid\mid B$.*

In order to use those pre-strategies as compiler, we need to be able to "apply" them to pre-strategy of the first game. More generally, we will define composition of pre-strategies. [36] We will do the composition by synchronising (with a pullback and with hiding) the two pre-strategies.

32. We should be able to use $EDC$ for both games and strategies, but some details seems more complicated.
33. For example, the player cannot forbid opponent moves, but can choose to restrict his own moves.
34. If an opponent move is allowed by the rules, nothing can prevent the opponent to do it.
35. The $PES$ $A$ is view as an $EDC$ with the equality for $\equiv_A$.
36. An application of a function can be seen as a particular case of a composition.

**Definition 4.5** (Composition of pre-strategies)**.**
*We take $\sigma : S \to A^\perp \mid\mid B$ and $\tau : T \to B^\perp \mid\mid C$.*

- *We extend $\sigma$ with the identity to $\sigma' : S \mid\mid C \to A^\perp \mid\mid B \mid\mid C$.*
- *We extend $\tau$ with the identity to $\tau' : A^\perp \mid\mid T \to A^\perp \mid\mid B^\perp \mid\mid C$.*
- *If we forget polarities of the events of $B$, both $\sigma'$ and $\tau'$ are pre-strategies on $A^\perp \mid\mid B^0 \mid\mid C$.*
- *We define $(P, p_S, p_T)$ as the pullback of $(S \mid\mid C, \sigma')$ and $(A^\perp \mid\mid T, \tau')$ relative to $A^\perp \mid\mid B^0 \mid\mid C$.*
- *We define $\tau \star \sigma : P \to A^\perp \mid\mid B^0 \mid\mid C$ as $\tau \star \sigma = \sigma' \circ p_S = \tau' \circ p_T$.*
- *We define $(H, \tau \circ \sigma)$ as the pre-strategy on the game $A^\perp \mid\mid C$ corresponding to $(P, \tau \star \sigma)$ after hiding the events of $B$.*

$$\begin{tikzcd}
 & H \arrow[ddd, bend left=60, "\tau\circ\sigma"] & \\
 & P \arrow[u, mapsto, "hiding"] \arrow[dl] \arrow[dr] \arrow[dd, "\tau\star\sigma"] & \\
S \mid\mid C \arrow[dr, "\sigma'"] & & A^\perp \mid\mid T \arrow[dl, "\tau'"'] \\
 & A^\perp \mid\mid B^0 \mid\mid C &
\end{tikzcd}$$

**Proposition 4.6** (The composition is well defined)**.** *The composition of pre-strategies, defined as a pullback followed by a hiding, is always defined and is associative.*

*Proof.* See [WV15] for more precisions. (Paper in progress) □

## 4.3 Copy-cat and strategies

We defined composition of pre-strategies, but we did not talk about the existence of a pre-strategy corresponding to the identity. The nearest thing to the identity is the copy-cat pre-strategy. Copy-cat on $A$ is defined on $A^\perp \mid\mid A$ and corresponds to the idea "If my opponent do a move, then I do the symmetric move". Graphically, it correspond to adding arrow from $\ominus$ moves to the corresponding $\oplus$ moves.

**Definition 4.7** (Copy-cat)**.** *For a game $A$, the copy-cat pre-strategy $(CC_A, \gamma_A)$ from $A$ to $A$ is defined as below :*

- (Moves) $CC_A = A \times \{0, 1\}$
- (Polarities) $p_{CC_A} : \begin{cases} (e,1) \mapsto p_A(e) \\ (e,0) \mapsto -1 \times p_A(e) \end{cases}$
- (Partial order)

$$(e,s) \leq_{CC_A} (e',s') \iff \begin{cases} s = s' \;\&\; e \leq_A e' \\ OR \\ s \neq s' \;\&\; e = e' \;\&\; p_{CC_A}(e) = \ominus \\ OR \\ \exists a, b \in CC_A,\; (e,s) <_{CC_A} a \leq_{CC_A} b <_{CC_A} (e',s') \end{cases}$$

- (Consistency) $(X \times \{0\}) \cup (Y \times \{1\}) \in Con_{CC_A} \iff X \in Con_A \;\&\; Y \in Con_A$

In most cases where a pre-strategy corresponds to what we would want intuitively to be a strategy, copy behave as the identity on it.

**Definition 4.8** (Strategy)**.** *We say that $(S, \sigma)$ is a strategy from the game $A$ to the game $B$ if $(S, \sigma)$ is a pre-strategy from $A$ to $B$ and if $\sigma \circ \gamma_A = \sigma = \gamma_B \circ \sigma$ (up to isomorphism). We say that $(S, \sigma)$ is a strategy on $A$ if $(S, \sigma)$ is a strategy from $\varnothing$ to $A$.*

**Proposition 4.9** (Characterization of a strategy)**.** *A pre-strategy $(S, \sigma)$ on $A$ is a strategy if and only if :*

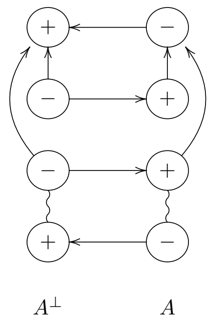

FIGURE 29 – Example of a copy-cat on a game $A$.

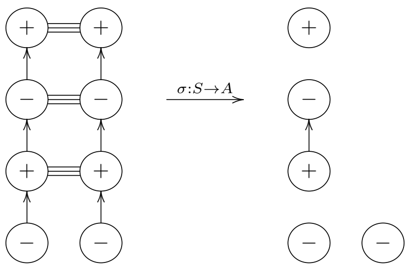

FIGURE 30 – Example of a strategy $\sigma : S \to A$

- ($\equiv$-injectivity [37]) $\sigma(s_1) = \sigma(s_2) \implies s_1 \equiv_S s_2$
- ($\ominus$-receptivity [38])
$$\begin{cases} X \in \mathcal{C}(S) \\ \sigma(X) \cup \{a\} \in \mathcal{C}(A) \\ p_A(a) = \ominus \end{cases} \implies \exists s \in S \;\; \equiv_S, \; \begin{cases} X \cup \{s\} \in \mathcal{C}(S) \\ \sigma(s) = a \end{cases}$$
- (No $\ominus$-redundancy)
$$\begin{cases} s_1 \equiv_S s_2 \\ p_S(s_1) = \ominus (= p_S(s_2)) \\ [s_1) \subseteq [s_2) \end{cases} \implies s_1 = s_2$$
- ($\oplus$-consistency)
$X \in Cons_S \iff [X^\oplus] = \{s \in S \mid \exists x \in X,\ s \leq_S x \;\&\; p_S(x) = \oplus\} \in Cons_S$
- (Innocence)
$$s_1 \leq_S s_2 \implies \exists t_1, t_2 \in S,\ s_1 \leq_S t_1 \leq_S t_2 \leq_S s_2,\ \begin{cases} \sigma(t_1) \leq_A \sigma(t_2) \\ OR \\ p_S(t_1) = \ominus \;\&\; p_S(t_2) = \oplus \end{cases}$$

*Proof.* See [WV15] for more precisions. (Paper in progress) □

This characterisation means that :

- ($\equiv$-injectivity) Even if maps of $EDC$ allow us to merge inconsistent (and non-equivalent) events, we cannot do it in a strategy.
- ($\ominus$-receptivity) We cannot restrict the set of possible moves for the opponents.

37. We recall that the equivalence relation on $A$ is the equality.
38. Using the $\equiv$-injectivity, we have the fact that all the possible $s$ are equivalents.

- (No $\ominus$-redundancy) We cannot duplicate opponent moves without any reason.
- ($\oplus$-consistency) Inconsistency comes from players moves, it means that we cannot put consistency restriction on opponent moves.
- (Innocence) We can only add dependencies from opponent moves to players moves, it means that we cannot put causal restrictions on opponents moves, nor between players moves.

Those conditions correspond to the intuitive definition of a strategy, except for the restriction "we cannon put causal restrictions between players moves".

This restriction comes from the fact that the copy-cat strategy does not exactly correspond to want we would want. The intuitive copy-cat is "if the opponent do a move, we do immediately after the symmetric move", whereas our copy-cat is "if the opponent do a move, we are allowed to do the symmetric move". There is two main difference : the symmetric move is allowed but not forced, and there is no "reaction window" for the player so the opponent can chain multiple moves without allowing the player to react [39].
If we extend event structures with the notion of "forced moves" and "immediate reaction", the extra restriction would probably disappear.

If we consider a team of player instead of a unique player, this impossibility of putting causal dependencies between players corresponds to a restriction of communications between players (which have a meaning in the case of distributives games).

## References

[NPW81] Mogens NIELSEN, Gordon PLOTKIN and Glynn WINSKEL. Petri net, event structures and domains, part 1. *Theorical Computer Science 13 (1981) 85-108, North-Holland Publishing Company.*

[RW11] Silvain RIDEAU and Glynn WINSKELL. Concurrent Strategies. *LICS 2011, ISBN : 978-0-7695-4412-0, pp : 409-418*

[W15] Glynn WINSKELL. On Probabilistic Distributed Strategies. *Invited paper. ICTAC 2015.*

[WV15] Glynn WINSKELL and Marc de Visme. Strategies with parallel causes. *Draft, 2015*

39. We can without problems imagine a opponent who plays one move and then plays another move which make impossible to play symmetrically.