Internship report
(complete 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.

A shorter version of this report, with almost no proof and less properties, 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 families 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

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.

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. 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 :

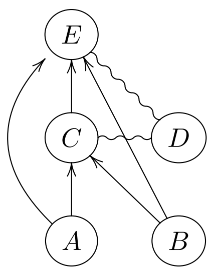


Figure 1 – Example of a Prime Event Structure.

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.

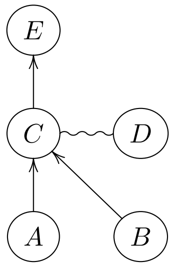


FIGURE 2 – Simplification of the Figure 1.

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) $\forall e \in X,\ [e] \subseteq X$

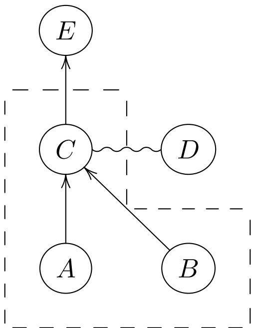


FIGURE 3 – Example of a configuration.

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) $\forall e, \tilde{e} \in \mathcal{D}(f),\ f(\tilde{e}) \leq_{E'} f(e) \implies \tilde{e} \leq_E e$

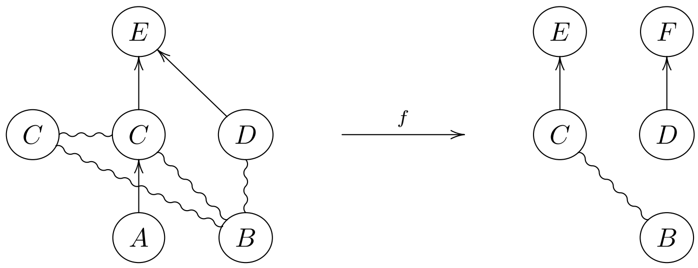


FIGURE 4 – Example of a map of PES.

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.*

Proof. All properties are trivially preserved by composition, and the identity is well define. □

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'.

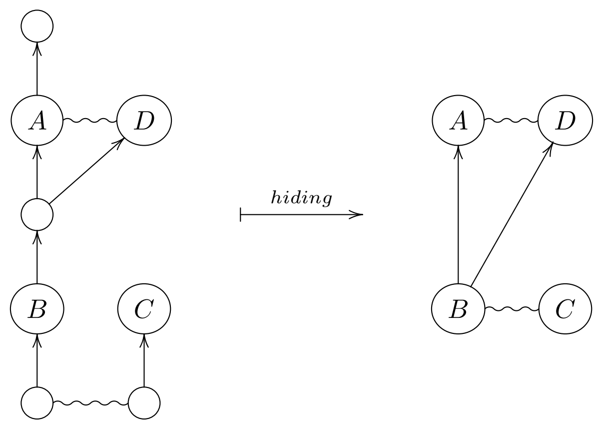


FIGURE 5 – The property "All configurations that contain the event A contain also the event B" is preserved by hiding.

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.*

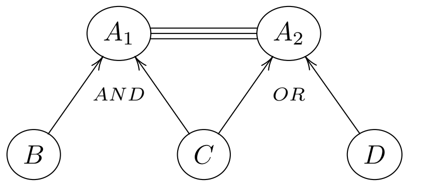


FIGURE 6 – Example of a simple $GES_{\equiv}$.

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

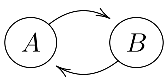


FIGURE 7 – Example of a non replete $GES_{\equiv}$.

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 :*

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

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 $GES_{\equiv}$.

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*

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)
  $\forall e \in X,\ \exists n \in \mathbb{N},\ \exists \{e_i\}_{1 \leq i \leq n},\ e_n = e\ \&\ \ \forall 1 \leq i \leq n,\ \{e_1, e_1, ...e_{i-1}\} \vdash_G e_i$

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 :

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

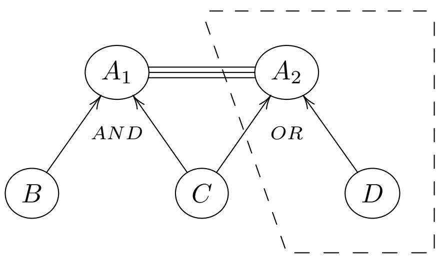


Figure 8 – Example of a configuration $GES_\equiv$.

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).

Property 1.12 ($GES_\equiv$ category)**.** *$GES_\equiv$ with maps of $GES_\equiv$ define a category. The identity map is the total function $a \mapsto a$, and the composition $g \circ f$ is the composition of functions.*

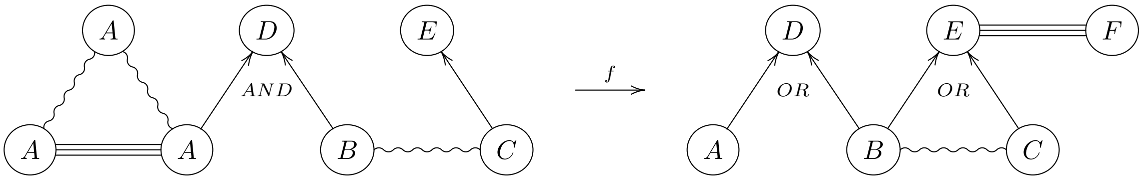


FIGURE 9 – Example of a map of $GES_{\equiv}$.

Proof. Let $f : F \to \mathcal{P}(G)$ and $g : G \to \mathcal{P}(H)$ be two maps of $GES_{\equiv}$. The property [Preserve Equivalence] and [Preserve Configurations] are trivially preserved by composition.

We take $a, b \in F$ with $a \equiv_F b$.
If $a \notin \mathcal{D}(f)$ or $b \notin \mathcal{D}(f)$, the [All or Nothing] property of f say that $f(a) = f(b) = undefined$, so $g \circ f(a) = g \circ f(b) = undefined$.
If $a \in \mathcal{D}(f)$ or $b \in \mathcal{D}(f)$, the [All or Nothing] property of f say that $a, b \in \mathcal{D}(f)$, and the [Preserve Equivalence] property of g say that $f(a) \equiv_G f(b)$. By the [All or Nothing] property of g, either $g \circ f(a) = g \circ f(b) = undefined$, either $g \circ f(a) \neq undefined \neq g \circ f(b)$.
So in all cases we have the [All or Nothing] property for $g \circ f$.

We take $a, b \in \mathcal{D}(g \circ f)$. We suppose that $g(f(a)) \equiv_H g(f(b))$.
If $\{a, b\} \in Con_F$, then, by [Preserve Consistency] of f, we have $\{f(a), f(b)\} = f(\{a, b\}) \in Con_G$. But $g(f(a)) \equiv_H g(f(b))$ so, by [Locally $\equiv$-Injective] of g, we have $f(a) \equiv_G f(b)$, so by [Locally $\equiv$-Injective] of f, we have $a \equiv_F b$.
That mean that $a \not\equiv_F b \implies \{a, b\} \notin Con_F$. So we have the [Locally $\equiv$-Injective] property for $g \circ f$.

So $g \circ f$ is a map of $GES_{\equiv}$. $\square$

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 :
- $\mathcal{D}(f) = \mathcal{D}(g)$
- $\forall a \in \mathcal{D}(f),\ f(a) \equiv_H g(a)$

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".

Property 1.14 ($GES_{\equiv}$ enriched category)**.** *$GES_{\equiv}$ with maps of $GES_{\equiv}$ is an 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.*

Proof. We take $f, \tilde{f} : F \to \mathcal{P}(G)$ and $g, \tilde{g} : G \to \mathcal{P}(H)$ four maps of $GES_{\equiv}$, with $f \equiv \tilde{f}$, and $g \equiv \tilde{g}$. We want to show $g \circ f \equiv \tilde{g} \circ f$ and $g \circ f \equiv g \circ \tilde{f}$.

We take $a \in \mathcal{D}(g \circ f)$, $a' = f(a)$ and $a'' = g \circ f(a)$.
We have $g \equiv \tilde{g}$ so $a' \in \mathcal{D}(\tilde{g})$, and $\tilde{g}(a') \equiv_H a''$. By symmetry, we have the first equivalence.

We take $a \in \mathcal{D}(g \circ f)$, $a' = f(a)$ and $a'' = g \circ f(a)$.
We have $f \equiv \tilde{f}$ so $a \in \mathcal{D}(\tilde{f})$ and $\tilde{f}(a) \equiv_G a'$. We have $a' \in \mathcal{D}(g)$, and g respects the [All or Nothing] property, so $\tilde{f}(a) \in \mathcal{D}(\tilde{g})$, and g preserves equivalence, so $g(\tilde{f}(a)) \equiv_H a''$. By symmetry, we have the second equivalence. $\square$

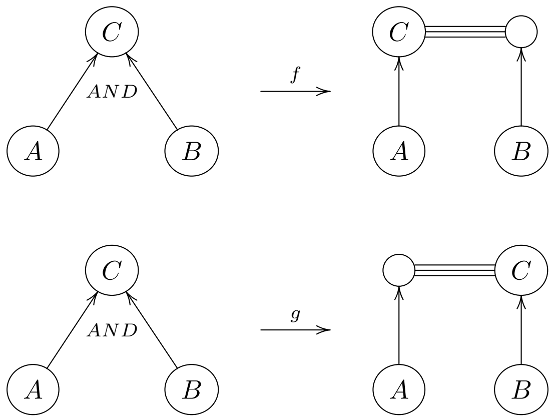


FIGURE 10 – Two equivalent maps.

1.3 Families with an equivalence relation

Definition 1.15 (Family with an equivalence relation)**.** *A $\mathcal{F}am_{\equiv}$ $(\mathcal{F}, \equiv_{\mathcal{F}})$ on a set of events E is a set of configurations $\mathcal{F} \subseteq \mathcal{P}(E)$ with an equivalence relation $\equiv_{\mathcal{F}} \subseteq E \times E$ on events (transitive, reflexive and symmetric) such that :*

- (Stable by finitely compatible union)
  $\forall \{X_i\}_{i\in I} \in \mathcal{F}^I$*, if* $\forall J \subseteq_{finite} I,\ \exists X_J \in \mathcal{F},\ \forall j \in J,\ X_j \subseteq X_J$ *then we have* $\bigcup_{i\in I} X_i \in \mathcal{F}$.
- (Secure chain)
  $\forall X \in \mathcal{F},\ \forall e \in X\ \exists n \in \mathbb{N},\ \exists \{e_i\}_{1\leq i\leq n},\ e_n = e\ \&\ \ \forall 0 \leq i \leq n,\ \{e_1, e_1, ...e_i\} \in \mathcal{F}$

Proposition 1.16 (Adjunction between $GES_{\equiv}$ and $\mathcal{F}am_{\equiv}$)**.**
For all $GES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_G)$, $(\mathcal{C}(G), \equiv_G)$ is a $\mathcal{F}am_{\equiv}$.
For all $\mathcal{F}am_{\equiv}$ $(\mathcal{F}, \equiv_{\mathcal{F}})$ on a set G, exist a unique replete *$GES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_{\mathcal{F}})$ such that $C(G) = \mathcal{F}$.*
We can deduce a category $\mathcal{F}am_{\equiv}$ such that $\mathcal{C}(-)$ is a right adjoint[7] between $GES_{\equiv}$ and $\mathcal{F}am$.

Proof. The proof is essentially the same as the proof of the Proposition 2.2. The fact that the adjunction is enriched is immediate. □

1.4 General Event Structures, and Families

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 in a same way families, and we have the same adjunction between GES and $\mathcal{F}am$

Definition 1.18 (Family)**.** *A $\mathcal{F}am$ $\mathcal{F}$ is a $\mathcal{F}am_{\equiv}$ $(\mathcal{F}, \equiv_{\mathcal{F}})$ with $\equiv_{\mathcal{F}}$ being the equality. It implies that the equivalence between maps is also the equality.*

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

7. See Definition 2.1 for more details. Here we have an enriched adjunction.

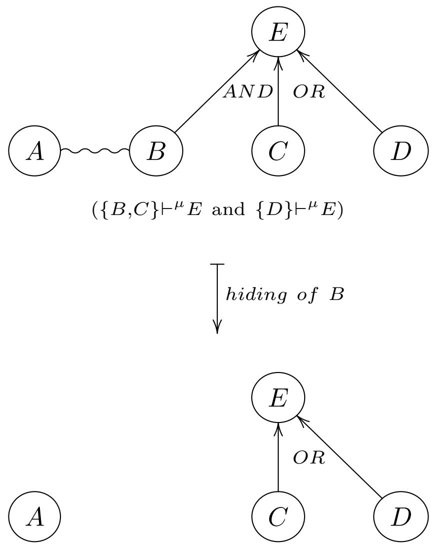


FIGURE 11 – The property "All configurations that contain A and E contain D" is lost under hiding.

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 replete $GES_\equiv$ $(P, \vdash_P, Con_P, \equiv_P)$ where :
- (Partial order) $\leq_P \subseteq P \times P$ *is a partial order*[8]
- (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.[9] 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[10] *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

8. So transitive, reflexive, and anti-symmetric. We recall that $[e] = \{e' \mid e' \leq_P e\}$ and $[e) = [e] \backslash \{e\}$.
9. But, because we have pull-back on $\mathcal{F}am_\equiv$, the $\equiv$-adjunction given by the Theorem 2.13 say that we have by-pull-back (so pull-back up to equivalence).
10. We will not talk a lot about them.

adjunction.[11] 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*[12]

- (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[13] *and the equivalence relation, and which respect the name function.*[14]

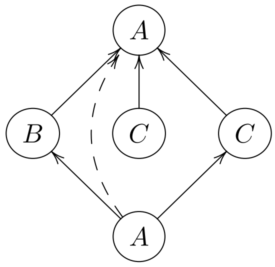


FIGURE 12 – Example of a POM on $\{A, B, C\}$ (The dash arrow is an implicit arrow).

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*

A partial order is characterised by the down closure of all its elements, so we have the following property.

Property 1.23 (Characterisation by the down-closure)**.**
For a set R, and $\{X_r\}_{r \in R} \subseteq \mathcal{P}(R)^R$ such that :

- (Reflexive) $\forall r \in R,\ r \in X_r$
- (Transitive) $\forall r \in R,\ \forall p \in X_r,\ X_p \subseteq X_r$
- (Antisymmetric) $\forall r \in Y,\ \forall p \in X_r,\ r \in X_p \implies r = p$

Exist a unique partial order $\leq_R$ such that $\forall r \in R,\ [r] = X_r$.
It implies that for a POM $(R, \leq_R, n_R)$, for $e \in R$, and $X \subseteq [e]$, there exists a unique POM $(R, \leq'_R, n_R)$ such that :

- $\forall p \neq e,\ [p]' = [p]$
- $[e]' = \{e\} \cup \bigcup_{x \in X \ \{e\}} [x]$

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.

11. In fact, we will not be able to build an adjunction, it will only be an $\equiv$-adjunction.
12. An important point is that realisation does *not* necessarily have the EDC property.
13. We will frequently say "order" instead of "partial order".
14. It mean that the image of an element by the bijection has the same name as the element

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[15] *on POM. We remark that two isomorphic POM are on the same cycle*[16] *for $\preceq_{fun}$.*

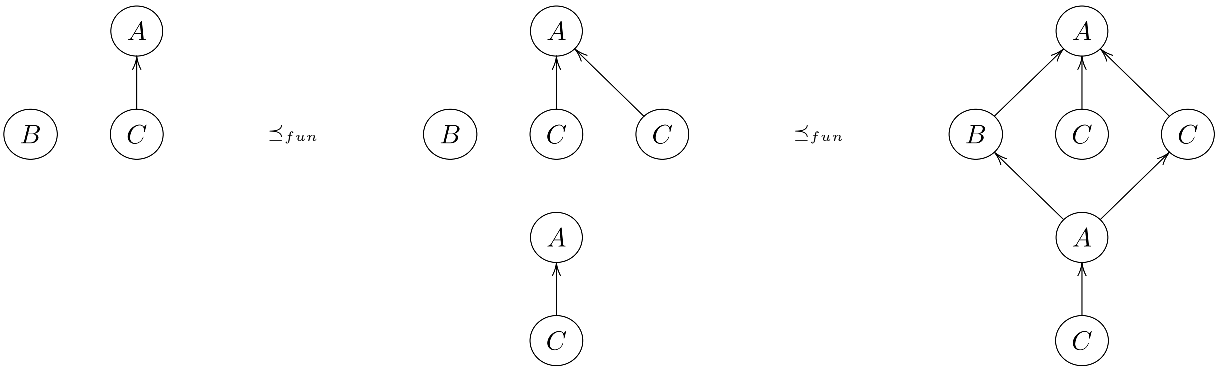


FIGURE 13 – Example of a sequel of $\preceq_{fun}$

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[17] *$m : \mathcal{D}(m) \subseteq R' \to R$ 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.
$\preceq_{sub}$ define a partial order up isomorphism.[18]

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

Property 1.26 (Top sub-structure)**.** *For $(R, \leq_R, n_R)$ a POM, for all $p \in R$, exist a unique (up to isomorphism) sub-structure of $(R, \leq_R, n_R)$ which has for top $m(p)$, where m is the mono-morphism of the sub-structure.*

---

15. We recall that a pre-order is a transitive and reflexive binary relation.
16. Cycles of a pre-order are usually called "equivalence classes", but we will not use this term to avoid confusion with equivalence classes of $GES_{\equiv}$.
17. 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.
18. 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.

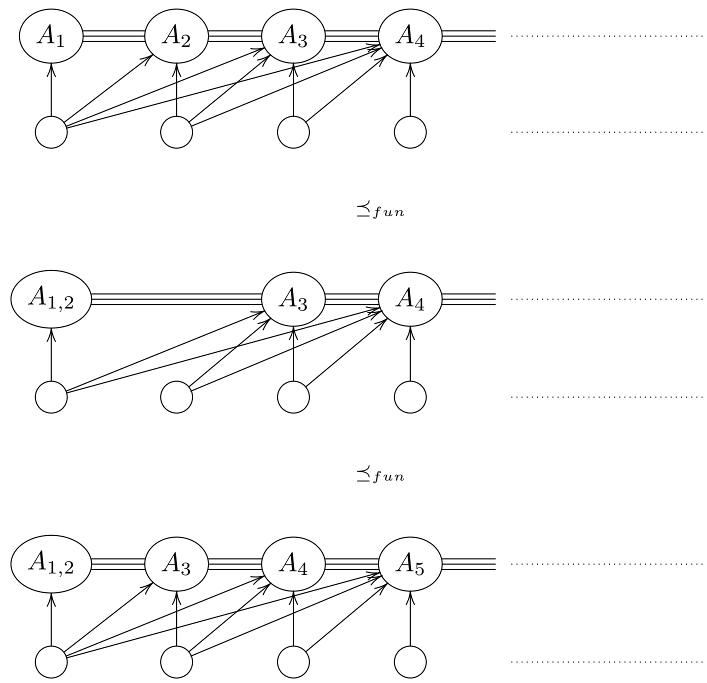


Figure 14 – Example of a case where $\preceq_{fun}$ is not an order : we have a loop of $\preceq_{fun}$ (On the figure, $p \equiv q \iff n(p) = n(q)$)

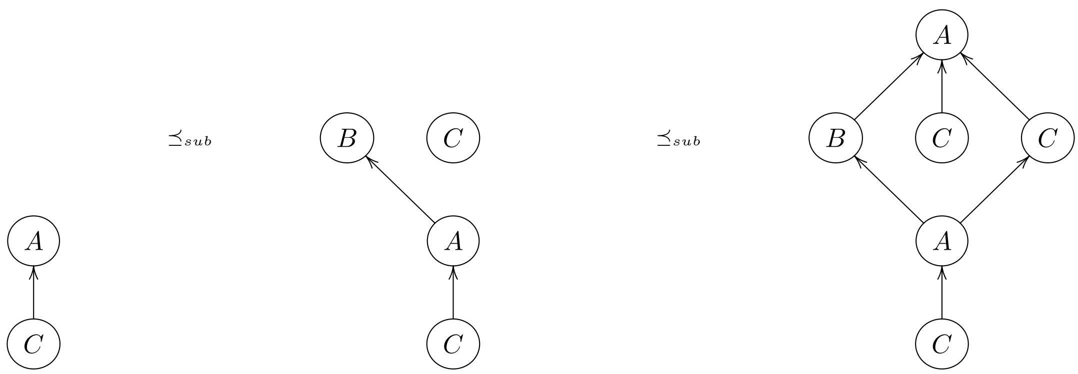


Figure 15 – Example of a sequel of $\preceq_{sub}$

Proof. We define $(R', \leq_{R'}, n_{R'})$ as below :

- $R' \subseteq R$
- $R' = [p]$
- $\leq_{R'} = \leq_R$ restricted to R'
- $n_{R'} = n_R$ restricted to R'

The mono-morphism $m : r \in R' \subseteq R \mapsto r \in R'$ shows that $(R', \leq_{R'}, n_{R'})$ is a sub-structure of $(R, \leq_R, n_R)$, and $p = m(p)$ is the top of $(R', \leq_{R'}, n_{R'})$.

We take $(R'', \leq_{R''}, n_{R''})$ a sub-structure, by the mono-morphism $\tilde{m}$, of $(R, \leq_R, n_R)$, with a top $\tilde{m}(p)$. We recall that the definition domain of $\tilde{m}$ has to be down-closed for $\leq$ ($\tilde{m}$ has a down-closed partiality), and has to have a top, so it is equal to $[p]$.

We have[19] that $\tilde{m} \circ m^{-1} : R' \to R''$ is a mono-morphism.

m^{-1} is a total mono-morphism, $m^{-1}(R') = [p]$, and $\tilde{m}$ is a surjective mono -morphism defined on $[p]$, so $\tilde{m} \circ m^{-1}$ is a total and surjective mono-morphism, so it is a bijection which preserves and reflects both order and equivalence. So $(R'', \leq_{R''}, n_{R''})$ is isomorphic to $(R', \leq_{R'}, n_{R'})$. □

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 $\preceq$)**.**

$$(R, \leq_R, n_R) \preceq (T, \leq_T, n_T)$$

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

So it give the diagram :

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

Proof. We will first prove the first equivalence. Because $\preceq$ is the transitive closure of $\preceq_{fun}$ and $\preceq_{sub}$, we only need to show that :

$$(R, \leq_R, n_R) \preceq_{fun} (S, \leq_S, n_S) \preceq_{sub} (T, \leq_T, n_T)$$

$$\implies \exists (S', \leq_{S'}, n_{S'}),\ (R, \leq_R, n_R) \preceq_{sub} (S', \leq_{S'}, n_{S'}) \preceq_{fun} (T, \leq_T, n_T)$$

Then, by induction, we can move all the $\preceq_{sub}$ to the beginning.

Let f be the function from S to R (defined by $\preceq_{fun}$) and m be the mono-morphism from T to S (defined by $\preceq_{sub}$). Up to isomorphism, m define a kind of inclusion. We rename[20] the element of S in a such way that m define an inclusion, it mean :

- $S \subseteq T$
- $\leq_S = \leq_T$ restricted to S
- $n_S = n_T$ restricted to S
- S is down-closed for $\leq_T$

We define $(S', \leq_{S'}, n_{S'})$ as below :

- $S' = R \uplus (T \backslash S)$
- $\forall a, b \in S',\ a \leq_{S'} b \iff \begin{cases} a \leq_R b & \&\ a \in R\ \&\ b \in R \\ a \leq_T b & \&\ a \in (T \backslash S)\ \&\ b \in (T \backslash S) \\ f(a) \leq_T b & \&\ a \in R\ \&\ b \in (T \backslash S) \end{cases}$
- $\forall a \in S', n_{S'}(a) = \begin{cases} n_R(a) & \text{if } a \in R \\ n_T(a) & \text{if } a \in (T \backslash S) \end{cases}$

19. Proof similar to the proof of EES be a category.
20. This renaming is not needed, but makes the proof easier to read.

Because S is down-closed for $\leq_T$, there is no $a \leq_{S'} b$ when $a \in (T \backslash S)$ and $b \in R$.
Defined in that way, it is obvious that $(R, \leq_R, n_R) \preceq_{sub} (S', \leq_{S'}, n_{S'})$.
We also have $(S', \leq_{S'}, n_{S'}) \preceq_{fun} (T, \leq_T, n_T)$ by taking the function f on R and the identity function on $(T \backslash S)$. It define a total and surjective map of $PES_{\equiv}$.

Now, we will prove the second equivalence, we only need to prove :

$$(R, \leq_R, n_R) \preceq_{sub} (S, \leq_S, n_S) \preceq_{fun} (T, \leq_T, n_T)$$

$$\implies \exists(\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}}),\ (R, \leq_R, n_R) \preceq_{fun} (\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}}) \preceq_{sub} (T, \leq_T, n_T)$$

Let f be the function from T to S (defined by $\preceq_{fun}$) and m be the mono-morphism from S to R (defined by $\preceq_{sub}$).
We define $(\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}})$ as below :
- $\tilde{S} = \{t \in T \mid m(f(t)) \text{ is defined}\}$
- $\leq_{\tilde{S}} = \leq_T$ restricted to $\tilde{S}$
- $n_{\tilde{S}} = n_T$ restricted to $\tilde{S}$

Clearly, we have $(\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}}) \preceq_{sub} (T, \leq_T, n_T)$. We now define the function $\tilde{f} = m \circ f$ from $\tilde{S} \subseteq T$ to R. We have immediately all the property of a total and surjective map of $PES_{\equiv}$, except the total property. Because $\tilde{S} = \{t \in T \mid m(f(t)) \text{ is defined}\}$, and f respect total property, then $tildef$ is total. So we have $(R, \leq_R, n_R) \preceq_{fun} (\tilde{S}, \leq_{\tilde{S}}, n_{\tilde{S}})$. ☐

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.*

Proof. We will prove that for all $k \in \mathbb{N}$, exists a finite number of perfect POM with k equivalence classes (up to isomorphism, on a fixed set E of cardinality k), by induction on k.
Let u_k be the maximal cardinal of perfect POM with k equivalence classes. $u_0 = 0$, which is finite.
Let $(R, \leq_R, n_R)$ be a perfect POM and $k + 1$ equivalence classes. And let $\mathcal{E}$ be one of the equivalence classes. Let $(R', \leq_{R'}, n_{R'})$ be the restriction of $(R, \leq_R, n_R)$ to $R' = \{r \in R \mid \mathcal{E} \cap [r) = \varnothing\}$. By the no-need-itself property, and because $\leq_R$ is antisymmetric :

$$\forall e \in \mathcal{E},\ \forall p \leq e \text{ with } p \neq e,\ [p) \cap \mathcal{E} = \varnothing$$

That mean that $\forall e \in \mathcal{E},\ [e) \subseteq R'$. By the [No Redundancy] property, we deduce that $card(\mathcal{E}) \leq 2^{card(R')} \leq 2^{u_k}$. That mean that $u_{k+1} \leq (k+1) \times 2^{u_k}$. That mean that $\forall k \in \mathbb{N},\ u_k$ is finite. With a finite number element, we can only build a finite number of partial order, so exits a finite number of perfect POM with k equivalence classes, up to isomorphism. ☐

1.6.2 Realisation

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

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*[21] *that :*

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

n_R is the name function *of the realisation $(R, \leq_R, n_R)$.*
We define in a similar way[22] *realisations of a $\mathcal{F}am_{\equiv}$.*

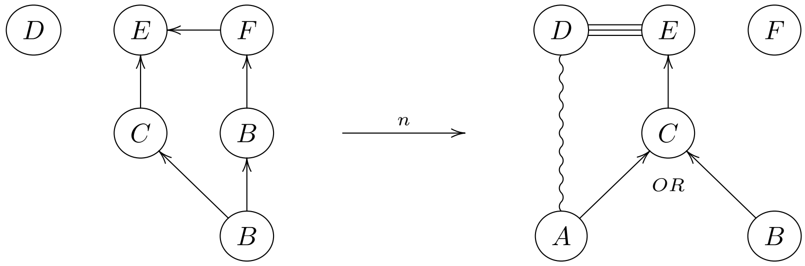


Figure 16 – Example of a realisation

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 $\preceq$.

Property 1.32 (Realisation up to $\succeq_{fun}$)**.**
If $(R', \leq_{R'}, n_{R'}) \succeq_{fun} (R, \leq_R, n_R)$ and $(R, \leq_R, n_R)$ is a realisation of $(G, \vdash_G, Con_G, \equiv_G)$, then $(R', \leq_{R'}, n_{R'})$ is a realisation of $(G, \vdash_G, Con_G, \equiv_G)$. If we call f the functional map defined by $\preceq_{fun}$, the following diagram commute :

$$\begin{tikzcd} R' \arrow[d, "f\ (fun)"'] \arrow[dr, "n_{R'}"] & \\ R \arrow[r, "n_R"'] & G \end{tikzcd}$$

Proof. $\succeq_{fun}$ implies a map f from R' to R such that $n_{R'} = n_R \circ f$, and $GES_{\equiv}$ is a category so the composition of two maps is a map, so $(R', \leq_{R'}, n_{R'})$ is a realisation of $(R, \vdash_R, Con_R, \equiv_R)$. □

Property 1.33 (Realisation up to sub-structure)**.** *If $(R', \leq_{R'}, n_{R'}) \preceq_{sub} (R, \leq_R, n_R)$ and $(R, \leq_R, n_R)$ is a realisation of $(G, \vdash_G, Con_G, \equiv_G)$, then $(R', \leq_{R'}, n_{R'})$ is a realisation of $(G, \vdash_G, Con_G, \equiv_G)$. If we call m the mono-morphism defined $\preceq_{sub}$, the following diagram commute*[23] *:*

$$\begin{tikzcd} R \arrow[d, "m\ (sub)"'] \arrow[dr, "n_R"] & \\ R' \arrow[r, "n_{R'}"'] \arrow[ur, phantom, "\subseteq"] & G \end{tikzcd}$$

Proof. $\succeq_{sub}$ implies a partial mono-morphism m from $\mathcal{D}(m) \subseteq R$ to R' such that $n_R = n_{R'} \circ m$ (when m is defined). m^{-1} define a map, and $n_{R'} = n_R \circ m^{-1}$, and $GES_{\equiv}$ is a

21. An important point is that we forget the equivalence relation $\equiv_G$.
22. By saying that all down closure go to elements of the family.
23. It is not a real commutation. The $\subseteq$ mean that the composition has a lesser definition domain than it should have to make the diagram commute.

category so the composition of two maps is a map, so $(R', \leq_{R'}, n_{R'})$ is a realisation of $(R, \vdash_R, Con_R, \equiv_R)$. $\square$

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. $\square$

Proposition 1.35 (Sub-structure of an image)**.**
Let $f : (G, \vdash_G, Con_G, \equiv_G) \to (H, \vdash_H, Con_H, \equiv_H)$ *be a map of* $GES_{\equiv}$*, and* $(R, \leq_R, n_R)$ *be a realisation of* $(G, \vdash_G, Con_G, \equiv_G)$*, and* $(S, \leq_S, n_S)$ *be a realisation of* $(H, \vdash_H, Con_H, \equiv_H)$*.*
If $(S, \leq_S, n_S) \preceq_{sub} f@(R, \leq_R, n_R)$ *then there exists* $(T, \leq_T, n_T) \preceq_{sub} (R, \leq_R, n_R))$ *such that* $(S, \leq_S, n_S) = f@(T, \leq_T, n_T)$*.*

Proof. We just have to take :
- $T = S$
- $\leq_T = \leq_R$ restricted to T
- $n_T = n_R$ restricted to T

$\square$

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

$\tilde{p}$ 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. $\square$

Now, we will define the notion of extremal [24] 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.

24. 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.

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.

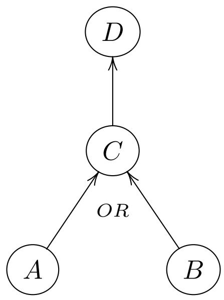


FIGURE 17 – A $GES_\equiv$.

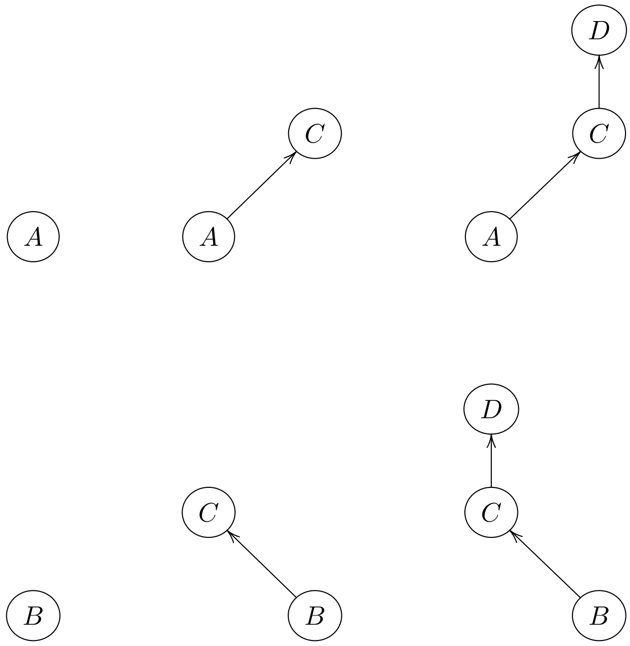


FIGURE 18 – All top extremal realisations of the $GES_\equiv$ of the Figure 17

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.

Proof. We take $(R, \leq_R, n_R)$ an extremal realisation.
By the Proposition 1.36, we take $(P, \leq_P, n_P) \preceq_{fun} (R, \leq_R, n_R)$ a *perfect* realisation of $(G, \vdash_G, Con_G, \equiv_G)$, then $(R, \leq_R, n_R) \preceq_{fun} (P, \leq_P, n_P)$ and $(P, \leq_P, n_P)$ necessarily extremal.
Let $f : R \to P$ and $g : P \to R$ the two function induced by $\preceq_{fun}$.
We define $h = f \circ g : P \to P$.

We know that $(P, \leq_P, n_P)$ has finite down-closures, so we can show by induction that $\forall p \in P, h(p) = p$.
We take $p \in P$ such that $\forall q \leq_P p$ with $q \neq p$, he have $h(q) = q$. We know that h reflects order, so $[p) = h(\ [p)\ ) \supseteq [\ h(p)\ )$. Moreover, h respects the name function, so $n_P(p) = n_P(h(p))$. Because $(P, \leq_P, n_P)$ is perfect, the [No redundancy] property says that we have $h(p) = p$.
So $h = id_P$, that means that both $f = g^{-1}$ so f preserves and reflects order, respects the name function, and is total and bijective. So $(R, \leq_R, n_R)$ is isomorphic to $(P, \leq_P, n_P)$.

So if we take $(S, \leq_S, n_S) \preceq_{fun} (P, \leq_P, n_P)$, we have $(S, \leq_S, n_S)$ extremal and $(P, \leq_P, n_P) \preceq_{fun} (S, \leq_S, n_S)$, so $(S, \leq_S, n_S)$ is isomorphic to $(P, \leq_P, n_P)$, so to $(R, \leq_R, n_R)$ too. □

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

$\Longrightarrow$ 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 $p \in n(P)$.

$\Longrightarrow$ By the Proposition 1.36, we can take $(P, \leq_P, n_P)$ perfect.
$\Longrightarrow$ 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 $n_T(Top(T)) = e$. □

Extremal realisation have other good properties :

Property 1.40 (Sub-realisation of an extremal realisation)**.** *Let $(R, \leq_R, n_R) \xrightarrow{n} (G, \vdash_G, Con_G, \equiv_G)$ be an extremal realisation. Let $(R', \leq_{R'}, n_{R'}) \preceq_{sub} (R, \leq_R, n_R)$ be a realisation. Then $(R', \leq_{R'}, n_{R'})$ is extremal.*

Proof. We take $(S', \leq_{S'}, n_{S'}) \preceq_{fun} (R', \leq_{R'}, n_{R'}) \preceq_{sub} (R, \leq_R, n_R)$. By the Proposition 1.28, exist $(S, \leq_S, n_S)$ such that $(S', \leq_{S'}, n_{S'}) \preceq_{sub} (S, \leq_S, n_S) \preceq_{fun} (R, \leq_R, n_R)$. So $(S, \leq_S, n_S)$ is isomorphic to $(R, \leq_R, n_R)$.
If we look at the proof of the Proposition 1.28, we see that the function defined by $(S', \leq_{S'}$

$, n_{S'}) \preceq_{fun} (R', \leq_{R'}, n_{R'})$ is the function defined by $(S, \leq_S, n_S) \preceq_{fun} (R, \leq_R, n_R)$ restricted to R'. That mean that $(S', \leq_{S'}, n_{S'})$ is isomorphic to $(R', \leq_{R'}, n_{R'})$, so $(R', \leq_{R'}, n_{R'})$ is extremal. ☐

Property 1.41 ($\preceq$ on extremal realisation is $\preceq_{sub}$)**.** *Let $(R, \leq_R, n_R)$ extremal realisation of the $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$. Let $(R', \leq_{R'}, n_{R'}) \preceq (R, \leq_R, n_R)$ be a realisation.*
Then $(R', \leq_{R'}, n_{R'})$ is an extremal realisation and $(R', \leq_{R'}, n_{R'}) \preceq_{sub} (R, \leq_R, n_R)$.
Moreover, if $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (R, \leq_R, n_R)$, then they are isomorph.

Proof. By the Property 1.40, we know that $(R', \leq_{R'}, n_{R'})$ is an extremal realisation.

By the Proposition 1.28, we know that there exists $(T, \leq_T, n_T)$ such that $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (T, \leq_T, n_T) \preceq_{sub} (R', \leq_{R'}, n_{R'})$. The Property 1.33 say that $(T, \leq_T, n_T)$ is a realisation. It is an extremal realisation. So $(R', \leq_{R'}, n_{R'})$ is isomorphic to $(T, \leq_T, n_T)$, and $(R', \leq_{R'}, n_{R'}) \preceq_{sub} (R, \leq_R, n_R)$.
The fact that $(R', \leq_{R'}, n_{R'})$ is isomorphic to $(T, \leq_T, n_T)$ prove that if $(R', \leq_{R'}, n_{R'}) \preceq_{fun} (R, \leq_R, n_R)$, then they are isomorph. ☐

Proposition 1.42 (Elements of an extremal realisation correspond to top extremal realisations)**.** *Let $(R, \leq_R, n_R)$ be an extremal realisation of $(G, \vdash_G, Con_G, \equiv_G)$. Then $(X, \leq_X, n_X)$ is isomorphic to $(R, \leq_R, n_R)$, where :*

- *$X = \{(S, \leq_S, n_S)$ top extremal realisation $\mid (S, \leq_S, n_S) \preceq (R, \leq_R, n_R)\}$ quotiented by the being-isomorphic equivalence relation.*
- *$\leq_X = \preceq$ restricted to X.*
- *$n_X(A) = n_A(Top(A))$*

It implies :

$$\{Top(S) \mid (S, \leq_S, n_S) \preceq (R, \leq_R, n_R) \ \& \ (S, \leq_S, n_S) \ \textit{top extremal realisation}\ \} = R$$

Proof. The fact that $(X, \leq_X, n_X)$ is isomorphic to $(R, \leq_R, n_R)$ comes from our working with extremals, so $\preceq = \preceq_{sub}$ (by the Property 1.41), and from the Property 1.26 which says that there exists a top extremal realisation, for each top. ☐

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

Proof. We suppose that $(R, \leq_R, n_R)$ is a extremal realisation.

By definition, a top extremal realisation has finite down-closures.

We take $\forall r \in R$ and $X \subseteq [r)$ such that X down-closed and $n_R(X) \vdash_G n_R(r)$. We can define $\leq'_R$ as below :

$$a \leq'_R b \iff \begin{cases} a \leq_R b & \& \ b \neq r \\ a \in X & \& \ b = r \end{cases}$$

$(R, \leq'_R, n_R)$ is a realisation of $(G, \vdash_G, Con_G, \equiv_G)$ and $(R, \leq'_R, n_R) \preceq_{fun} (R, \leq_R, n_R)$ (by the identity function). But $(R, \leq_R, n_R)$ is extremal, so $(R, \leq'_R, n_R)$ is isomorphic to $(R, \leq_R, n_R)$. That mean $X = [p)$. So we have the [Minimal] property.

We know that extremal realisation are perfect (see Property 1.36), so we have the [No multiplicity] property.

Now we suppose that $(R, \leq_R, n_R)$ respect all the different properties, and we will show that is an extremal realisation.

We have immediately that $(R, \leq_R, n_R)$ is a realisation.

We take a realisation $(S, \leq_S, n_S) \preceq_{fun} (R, \leq_R, n_R)$. Let f be the function from R to S defined by $\preceq_{fun}$. We take $r \in R$.
Because f is total, we can define without problems $X = f^{-1}(\ [\ f(r)\ )\ )$. Because f reflects order, we have $X \subseteq [r)$ and X down-closed.
Because $(S, \leq_S, n_S)$ is a realisation, we have $n_S([f(r))) \vdash_G n_S(f(r))$.
Because f respect the name function, we have $n_R([r)) = n_S([f(r)))$ and $n_R(r) = n_S(f(r))$.
So $n_R(X) \vdash_G n_R(r)$ and $X \subseteq [r)$ with X down-closed. So, by the [Minimal] property, $X = [r)$. So the image of the down-closure is equal to the down-closure of the image (that mean that f preserves order).

We will now prove by induction on the structure of $(R, \leq_R, n_R)$ that f is injective. We take $p, q \in R$ such that $f(p) = f(q)$ and $\forall r, r' \in [p) \cup [q),\ f(r) = f(r') \iff r = r'$.
Because f respect the name function, we have $n_R(p) = n_R(q)$.
Because the image of the down closure is the down-closure of the image, and because f is injective on $[p) \cup [q)$, we have $[p) = [q)$.
So, by the [No multiplicity] property, we have $p = q$. By structural induction on $\leq_R$ (which is well-founded because has finite bases), f is injective.

f is by definition surjective, so f is a bijection which preserves and reflects order, and respect the name function. So $(S, \leq_S, n_S)$ is isomorphic to $(R, \leq_R, n_R)$. So $(R, \leq_R, n_R)$ is extremal. $\square$

The next proposition is not used for the proof of the different adjunctions of the next section, but shows that the partial order $\preceq_{sub}$ on extremal realisation is an order regarded as category. [25]

Proposition 1.44 (Uniqueness of function between extremal realisations)**.** *Let $(R, \leq_R, n_R)$ and $(S, \leq_S, n_S)$ be two extremal realisations of the $GES_\equiv$ $(G, \vdash_G, Con_G, \equiv_G)$ such that $(R, \leq_R, n_R) \preceq (S, \leq_S, n_S)$.*
We remark that composition of total and surjective maps of $GES_\equiv$ with surjective mono-morphisms give a surjective quasi-map [26] of $GES_\equiv$.
There exists a unique surjective quasi-map f of $GES_\equiv$ from [27] $(S, \leq_S, n_S)$ to $(R, \leq_R, n_R)$ which respect the name function. [28] Moreover, f is a mono-morphism.

Proof. Because of the Property 1.41, $\preceq$ is $\preceq_{sub}$, and we have a surjective mono-morphism m (which is also a surjective quasi-map of $GES_\equiv$) from S to R which respect the name-function (see Property 1.33). That prove the existence.
We take surjective quasi-map f of $GES_\equiv$ between $(R, \leq_R, n_R)$ and $(S, \leq_S, n_S)$.
We remark that m^{-1} is a total quasi-map, so [29] $m^{-1} \circ f$ is a quasi-map from S to S.
We suppose that $m^{-1} \circ f \neq id_S$. Because $(S, \leq_S, n_S)$ has finite bases, we can find $s \in S$ such that $\forall p \leq_S s,\ m^{-1}(f(p)) = p$ and $m^{-1}(f(s)) \neq s$. That mean that $[\ m^{-1}(f(s))\ ) \supseteq [s)$. S is extremal so necessarily $[\ m^{-1}(f(s))\ ) = [s)$. Because f and m respect the name-functions, we have $n_S(m^{-1}(f(s))) = n_S(s)$. By the [No Redundancy] condition, we have a contradiction. So $m^{-1} \circ f = id_S$. We have m^{-1} total so $f = m$. $\square$

25. In the category theory, orders are categories such that for all objects A and B, there is at most one morphism from A to B or from B to A (and never both in the same time if $A \neq B$).
26. It mean that it has not necessarily the [All or Nothing] property.
27. More precisely, the map is between the two $GES_\equiv$ associated.
28. It mean that $\forall s \in S,\ n_S(s) = n_R(f(s))$.
29. The proof is the same that the proof of $GES_\equiv$ being an enriched category (see Property 1.14).

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 $\mathcal{F}am$

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 $\mathcal{F}am$)**.** *The functor $\mathcal{C} : GES \to \mathcal{F}am : (E, \vdash_E, Con_E) \mapsto \mathcal{C}(E)$ define a right adjoint between GES and $\mathcal{F}am$.*

Proof. The left adjoint is :

$$\mathcal{F}am \to GES : \mathcal{F} \mapsto (E, \vdash_E, Con_E)$$

Where E is the minimal set such that $\mathcal{F}$ is a family on E, $X \vdash_E e \iff \exists Y \subseteq X \cup \{e\},\ Y \in \mathcal{F}$, and $X \in Con_E \iff \exists Y \supseteq X,\ Y \in \mathcal{F}$.
The image by $\mathcal{C}$ of a map $f : E \to E'$ is $f : E \to E'$, same thing with the rigth adjoint, so we have the one-to-one correspondence between maps. $\square$

2.2 The enriched adjunction between $\mathcal{F}am$ and $\mathcal{F}am_{\equiv}$

We recall that $\mathcal{F}am$ correspond to replete GES, and $\mathcal{F}am_{\equiv}$ correspond to replete $GES_{\equiv}$. 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 Definition 5.2 for more details.

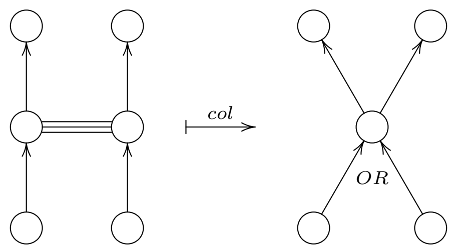


Figure 19 – Simple example of the effect of the *col* functor on a replete $GES_{\equiv}$

Definition 2.3 (The *col* functor)**.** *The functor $col : \mathcal{F}am_{\equiv} \to \mathcal{F}am$ is defined as below :*

$$(\mathcal{F}, \equiv_{\mathcal{F}}) \xmapsto{col} \mathcal{G}'$$

where $\mathcal{F} \subseteq \mathcal{P}(E)$, $\mathcal{G} \subseteq \mathcal{P}(E_{\equiv_\mathcal{F}})$ *and :*

$$\mathcal{G} = \left\{ Y \mid \begin{cases} \forall y \in Y,\ \exists X \in \mathcal{F},\ y \in X_{\equiv_\mathcal{F}} \subseteq Y \\ \forall Z \subseteq_{finite} Y,\ \exists X \in \mathcal{F},\ Z \subseteq X_{\equiv_\mathcal{F}} \end{cases} \right\}$$

and :

$$(f : E \to E') \xmapsto{col} (g : E_{\equiv_\mathcal{F}} \to E'_{\equiv_\mathcal{F}})$$

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).

Proof. The image of an object is well defined (i.e respect the property [Secured chain] and [Stable by finitely compatible union]).

We first need to proof that it is well defined, which means than (1) and (2) are compatibles, and defined a map of $\mathcal{F}am$.

- (1) and (2) are compatibles because f preserves equivalence and respect the [All or Nothing] property.
- g is locally *equiv*-injective because f is locally *equiv*-injective.
- g preserves configurations because f preserves configurations.

The fact that *col* of the identity is the identity, and the fact that *col* of a composition of two maps is the composition of the *col* of the maps, are immediate.

Because of the way that it is defined, naturality conditions are obvious.

Lemma 2.4 (Equivalence of maps and *col*)**.**
If $f, g : (G, \vdash_G, Con_G, \equiv_G) \to (H, \vdash_H, Con_H, \equiv_H)$ *are two maps of GEDC, then :*

$$f \equiv g \iff col(f) = col(g)$$

Proof. Saying that $col(f) = col(g)$ implies that they have the same definition domain $\mathcal{D}$. But $col(f) = col(g)$ also implies that the image of an element by f and by g are equivalent, so $col(f) = col(g) \implies f \equiv g$.

If $f \equiv g$, f and g do the same thing on equivalence classes, so $col(f) = col(g)$. ☐

It proves the fact that *col* is an enriched functor.

Definition 2.5 (Functor from replete $GES_\equiv$ to replete GES)**.** *We can also see col as a functor from* $GES_\equiv$ *to* GES*. It give :*

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

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* $\{p\}_{\equiv_G} = e,\ \{f(p)\}_{\equiv_G} = e'$

Proof. It works exactly for the same reasons. ☐

Definition 2.6 (Inclusion [30] functor of $\mathcal{F}am$ in $\mathcal{F}am_{\equiv}$)**.**

$$\mathcal{I} : \mathcal{F}am \to \mathcal{F}am_{\equiv}$$
$$\mathcal{F} \mapsto (\mathcal{F}, =)$$
$$(f : E \to E') \mapsto (f : E \to E')$$

$\mathcal{I}$ is a functor.

Proof. $\mathcal{I}$ correspond to an inclusion. All the property of a functor are trivially respected. □

Property 2.7 ($col \circ \mathcal{I}(E) = E$)**.** *$col \circ \mathcal{I}(E) = E$ is isomorphic to E*

Proof. The equivalence in $\mathcal{I}(E)$ is the identity. So the *col* merge nothing.
Because of [Stable by finitely compatible union], *col* does nothing. □

Theorem 2.8 (The adjunction between $\mathcal{F}am$ and $\mathcal{F}am_{\equiv}$)**.** *$\mathcal{I}$ and col define an enriched adjunction between $\mathcal{F}am$ and $\mathcal{F}am_{\equiv}$, more precisely $col \dashv \mathcal{I}$.*
It mean that for $\mathcal{F}$ a $\mathcal{F}am$ on F, and $(\mathcal{G}, \equiv_{\mathcal{G}})$ a $\mathcal{F}am_{\equiv}$ on G, we have [31] :

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

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

Proof. We will first prove the existence of h. We define $h : \mathcal{G} \to \mathcal{I}(\mathcal{F})$ by : for $a \in G$, $\{a\}_{\equiv_{\mathcal{G}}} \in G_{\equiv}$, $f(\{a\}_{\equiv_G}) \in F$ and because $\mathcal{I}$ is an inclusion functor, we can take $h(a) = f(\{a\}_{\equiv_G}) \in \mathcal{I}(F)$. We have to check all maps property.

- (All or nothing) f respect the [All or nothing] property, so g too.
- (Preserve equivalence) Equivalent element are mapped to the same event, so g preserves equivalence.
- (Preserves Configurations) $col(\mathcal{F})$ is $\mathcal{F}$ where equivalent events are merged, and completed in a way such that we have a $\mathcal{F}am$, and $col(\mathcal{G})$ is $\mathcal{G}$ merge and then completed in a same way than $col(\mathcal{F})$. The completion is done in the same way, and f preserves configurations, so g preserves configurations.
- (Locally *equiv*-Injective) f is locally *equiv*-injective, and the equivalence of $\mathcal{I}(\mathcal{F})$ is the equality, g preserves configurations, so g is locally *equiv*-injective.

With this definition, we have immediately $f = col(h)$.

The lemma 2.4 prove the uniqueness up to equivalence. But here, we use map from a from a $GEDC$ to a $GEDC$ which come from a GES, so equivalence classes have cardinality one, so we have the uniqueness. □

2.3 The $\equiv$-adjunction between $\mathcal{F}am_{\equiv}$ and $PES_{\equiv}$

We recall that $\mathcal{F}am_{\equiv}$ correspond to replete $GES_{\equiv}$. 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 Definition 5.1 for more details.

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 $B/_{\equiv}$.*

30. 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.
31. We recall that the equivalence on maps of $\mathcal{F}am$ is the equality. Moreover, equivalence classes on maps from a $\mathcal{F}am_{\equiv}$ to a $\mathcal{F}am_{\equiv}$ which come from a $\mathcal{F}am$, have cardinality one. So this property is also true up to equivalence.

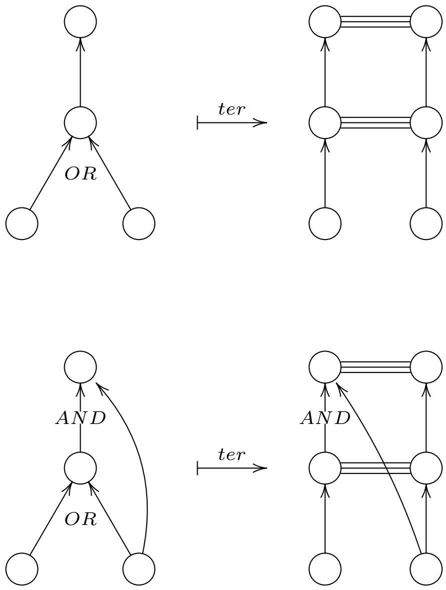


FIGURE 20 – Simple examples of the effect of the ter functor on a replete $GES_{\equiv}$

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 [32] pseudo-functor)**.** *The pseudo-functor* $ter : \mathcal{F}am_{\equiv} \to PES_{\equiv}$ *is defined as below :*

$$(\mathcal{F}, \equiv_{\mathcal{F}}) \stackrel{ter}{\longmapsto} (P, \leq_P, Con_P, \equiv_P)$$

where $(\mathcal{F}, \equiv_{\mathcal{F}})$ *is a* $\mathcal{F}am_{\equiv}$ *on* E *and :*

- $P = \{(R, \leq_R, n_R)$ *top extremal realisation of* $\mathcal{F}$ $\}$ [33]
- $(R, \leq_R, n_R) \leq_P (S, \leq_S, n_S) \iff (R, \leq_R, n_R) \preceq (S, \leq_S, n_S)$
- $(R, \leq_R, n_R) \equiv_P (S, \leq_S, n_S) \iff n_R(Top(R)) = n_S(Top(S))$
- $X \in Con_P \iff \exists Y \in \mathcal{F}, \bigcup_{(R, \leq_R, n_R) \in X} n_R(R) \subseteq Y$

and

$$\{f : \mathcal{D}(f) \subseteq E \to E'\}_{\equiv} \stackrel{ter}{\longmapsto} \{g : \mathcal{D}(g) \subseteq P \to P'\}_{\equiv}$$

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.

Proof. We will first prove that for any map f of $\mathcal{F}am_{\equiv}$, we have a map g of $PES_{\equiv}$ which is an image by ter. Then we will prove that all possible g are equivalent, and after we will prove that if we take $\tilde{f} \equiv f$, we obtain $\tilde{g} \equiv g$.

We build g an image of f by induction on $\leq_P$:
For $(R, \leq_R, n_R) \in P$, we suppose that g is already defined on $[\ (R, \leq_R, n_R)\ )$. We know that

32. ter means "top extremal realisations".
33. Quotiented by the being-isomorphic equivalence relation.

$f@(R, \leq_R, n_R)$ is a realisation of $\mathcal{F}'$.
Moreover [34], $\forall (S, \leq_S, n_S) <_P (R, \leq_R, n_R)$, $g((S, \leq_S, n_S)) \preceq f@(S, \leq_S, n_S)$ and $f@(S, \leq_S, n_S) \preceq_{sub} f@(R, \leq_R, n_R)$, so $g((S, \leq_S, n_S)) \preceq f@(R, \leq_R, n_R)$.
Using the Proposition 1.28, we wrote h_S the map and m_S the monomorphism given by $\preceq$ such that $m_S \circ h_S : f@(R, \leq_R, n_R) \to g((S, \leq_S, n_S))$.
We define $(T, \leq_T, n_T)$ a realisation of $(\mathcal{F}', \equiv_{\mathcal{F}'})$ as below :

- $T = R$
- $n_T = f \circ n_R$
- $\forall t \in T,\ t \leq_T Top(R)$
- $\forall (S, \leq_S, n_S) <_P (R, \leq_R, n_R)$, $\forall p, q \in \mathcal{D}(m_S \circ h_S)\ p \leq_T q \iff m_S \circ h_S(p) \leq_S m_S \circ h_S(q)$

This definition is coherent because g has a down-closed image (on $[\ (R, \leq_R, n_R)\ ))$, and reflects order (on $[\ (R, \leq_R, n_R)\ ))$, so different $(S, \leq_S, n_S)$ will give the same order on their intersection.
We have immediately $(T, \leq_T, n_T) \preceq_{fun} f@(R, \leq_R, n_R)$, and $(T, \leq_T, n_T)$ is a finite realisation, so we can use the Proposition 1.39, and we have $(P, \leq_P, n_P) \preceq (T, \leq_T, n_T)$ which is a top extremal realisation of $(\mathcal{F}', \equiv_{\mathcal{F}'})$, with $n_P(Top(P)) = n_T(Top(T))$.
We complete g with $g((R, \leq_R, n_R)) = (P, \leq_P, n_P)$. In that way, all properties are preserved :

- g respects [Same partiality], [Image] and [Respects the name function] by construction.
- g is locally *equiv*-injective because f is locally *equiv*-injective.
- g preserves equivalence because f preserves equivalence and because $n_R(Top(R')) = f(n_R(Top(R))) = n_{R'}(f(Top(R))$.
- g respect the [All or Nothing] property because it has the same partiality as f.
- g has a down-closed image and reflects order by construction.
- g preserves consistency, see below for the proof.

We take $X \in Con_P$, that mean there exists $Y \in \mathcal{C}(P)$ such that $X \subseteq Y$.
Y is down closed, so $Y' = g(Y)$ is down closed too. By definition, we have :

$$Y' \in Con_{P'} \iff \exists F' \in \mathcal{F}, \bigcup_{(R', \leq_{R'}, n_{R'}) \in Y'} n_{R'}(R') \subseteq F$$

Because Y' is down closed, and by the Proposition 1.42, we have :

$$\bigcup_{(R', \leq_{R'}, n_{R'}) \in Y'} n_{R'}(R') = \{n_{R'}(Top(R')) \mid (R', \leq_{R'}, n_{R'}) \in Y'\}$$

Because $n_R(Top(R')) = f(n_R(Top(R))) = n_{R'}(f(Top(R))$, we have :

$$\{n_{R'}(Top(R')) \mid (R', \leq_{R'}, n_{R'}) \in Y'\} = \{n_R(Top(R)) \mid (R, \leq_R, n_R) \in Y\}$$

Because Y is down closed, and by the Proposition 1.42, we have :

$$\{n_R(Top(R)) \mid (R, \leq_R, n_R) \in Y\} = \bigcup_{(R, \leq_R, n_R) \in Y} n_R(R)$$

We know that $Y \in Con_P$, so $\exists F \in \mathcal{F}, \bigcup_{(R, \leq_R, n_R) \in Y} n_R(R) \subseteq F$, so we have $Y' \in Con_{P'}$, which mean $g(X) \in Con_{P'}$.

So g preserves consistency. ***So the image of f by ter exists***

The property [Same Partiality] and [Respect the name function] implies that all possible g are equivalents.

34. We define $<_P$ as $\leq_P \ \cap \ \neq$.

If we take $\tilde{f} \equiv f$, it will have the same partiality as f, so $\tilde{g}$ and g have the same partiality. Moreover, $\tilde{f}$ and f will do the same things up to equivalence, so, by [Respects the name function], $\tilde{g}$ and g too.

So the image of f by ter is well defined

$id@(R, \leq_R, n_R)$ is isomorphic to $(R, \leq_R, n_R)$ so the image by ter of the identity map is the identity map.

We now need to proof that, ter preserves composition up to equivalence. That mean that, if we take tree $\mathcal{F}am_{\equiv}$ $(\mathcal{F}_1, \equiv_1)$, $(\mathcal{F}_2, \equiv_2)$ and $(\mathcal{F}_3, \equiv_3)$, and two maps of $\mathcal{F}am_{\equiv}$ $f : E_1 \to E_2$ and $g : E_2 \to E_3$, then $ter(g \circ f) \equiv ter(g) \circ ter(f)$.
The [Same partiality] property say that we will have no problems with the definition domains.

We take $p \in ter(\mathcal{F}_1)$, $p' = ter(g)(p)$ and $p'' = ter(g) \circ ter(f)(p)$. We wrote $a = Top(p) \in E_1$, $a' = Top(p') \in E_2$ and $a'' = Top(p'') \in E_3$. By definition of ter (or more precisely, by the definition of an image of a realisation), we have $a' = f(a)$ and $a'' = g(a')$. So $a'' = g \circ f(a)$, so by [Respect the name function], if we wrote, $q = ter(g \circ f)(p)$ then $Top(q) = a''$, and $q \equiv_{ter(H)} p''$.

So $ter(g \circ f) \equiv ter(g) \circ ter(f)$.

So ter is a pseudo-functor.

Because of the way that it is defined, naturality conditions are obvious. ☐

Definition 2.11 (Inclusion functor of $PES_{\equiv}$ in $\mathcal{F}am_{\equiv}$)**.**

$$\mathcal{I} : PES_{\equiv} \to \mathcal{F}am_{\equiv}$$

$$(P, \leq_P, Con_P, \equiv_P) \mapsto (\mathcal{C}(P), \equiv_P)$$
$$(f : P \to Q) \mapsto (f : P \to Q)$$

$\mathcal{I}$ is a functor, and a pseudo-functor.

Proof. $\mathcal{I}$ correspond to an inclusion of $PES_{\equiv}$ in $\mathcal{F}am_{\equiv}$. All the property of a functor are trivially respected. ☐

Property 2.12 ($ter \circ \mathcal{I}(P) = P$)**.** *$ter \circ \mathcal{I}(P)$ is isomorphic to P*

Proof. All events of $\mathcal{I}(P)$ have a unique minimal enabling. So from a pre-realisation, we can extract a unique extremal realisation by $\preceq_{fun}$, and so each event correspond to a unique top extremal realisation.
The order, the equivalence relation, and the consistency are necessarily the same. ☐

Theorem 2.13 (The $\equiv$-adjunction between $\mathcal{F}am_{\equiv}$ and $PES_{\equiv}$)**.** *$\mathcal{I}$ and ter define an $\equiv$-adjunction between $\mathcal{F}am_{\equiv}$ 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 $(\mathcal{F}, \equiv_{\mathcal{F}})$ a $\mathcal{F}am_{\equiv}$, and up to equivalence, we have :

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

(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 $ter(\{h\}_{\equiv})$)

Proof. We define $\vdash_P$ the enabling associated to $\leq_P$ and $\vdash_{ter(\mathcal{F})}$ the enabling associated to $\leq_{ter(\mathcal{F})}$.

For $p \in P$, we write $f(p) = (R_p, \leq_{R_p}, n_{R_p})$.

We define $h(p) = n_{R_p}(Top(R_p))$ and we check that it define a map of $GES_\equiv$:

- (All or Nothing) $p \in \mathcal{D}(p) \iff p \in \mathcal{D}(f)$ so OK.
- (Preserve Equivalence) f preserves equivalence so h too, so OK.
- (Locally *equiv*-Injective) f is locally *equiv*-injective, so h too, so OK.
- (Preserve Consistency) the definition of ter does that if $X \subseteq ter(\mathcal{F})$ is consistent, then $\exists F \in \mathcal{F},\ \{n_R(Top(R)) \mid (R, \leq_R, n_R) \in X\} \subseteq F$, so OK.
- (Preserve Enabling) see below

We take $p \in P$ and $X \vdash_P p$
By definition of h, $n_{R_p}(Top(R_p)) = h(p)$.
Because f preserves enabling, $f(X) \vdash_{ter(\mathcal{F})} (R_p, \leq_{R_p}, n_{R_p})$
Because $ter(\mathcal{F})$ is an $PES_\equiv$, it have a unique minimal enabling, so we have $\{(R', \leq_{R'}, n_{R'}) \in ter(\mathcal{F}) \mid (R', \leq_{R'}, n_{R'}) \preceq (R_p, \leq_{R_p}, n_{R_p})\} \subseteq f(X)$
Because of the Proposition 1.42 R_p is isomorphic to $\{(R', \leq_{R'}, n_{R'}) \in ter(\mathcal{F}) \mid (R', \leq_{R'}, n_{R'}) \preceq (R_p, \leq_{R_p}, n_{R_p})\}$, so by taking the name, $n_{R_p}(R_p) \subseteq h(X)$.
Because $(R_p, \leq_{R_p}, n_{R_p})$ is a realisation, then $n_{R_p}(R_p) \vdash_{ter(\mathcal{F})} n_{R_p}(Top(R_p))$, and so $h(X) \vdash_{ter(\mathcal{F})} h(p)$

So h is a map.

We also need to proof that $f \equiv ter(h)$

For $p \in P$, we write $f(p) = (R_p, \leq_{R_p}, n_{R_p})$ and $t = n_{R_p}(Top(R_p))$.
We also write $ter(h)(p) = (R'_p, \leq_{R'_p}, n_{R'_p})$ and $t' = n_{R'_p}(Top(R'_p))$.
By definition of h, we have $t \equiv_\mathcal{F} t'$.

So $f \equiv ter(h)$

Now we have to proof the uniqueness up to equivalence

Let $g : I(P) \to \mathcal{F}$ such that $ter(g) \equiv f$
For $p \in P$, we write $f(p) = (R_p, \leq_{R_p}, n_{R_p})$ and $t = n_{R_p}(Top(R_p))$.
We also write $ter(g)(p) = (R'_p, \leq_{R'_p}, n_{R'_p})$ and $t' = n_{R'_p}(Top(R'_p))$. By definition of ter, $g(p) = t'$
Because $ter(g) \equiv f$, we have $t \equiv_\mathcal{F} t'$ (and no problems with definition domains)

So we have $\forall p \in P,\ g(p) \equiv_\mathcal{F} h(p)$ so $g \equiv h$

So h is unique □

Corollary 2.14 (The $\equiv$-adjunction between $GES_\equiv$ and $PES_\equiv$)**.** *By composition with the enriched adjunction between $GES_\equiv$ and $\mathcal{F}am_\equiv$, we obtain an $\equiv$-adjunction between $GES_\equiv$ and $PES_\equiv$. The two functor of the $\equiv$-adjunction are essentially the same as before.*

Proposition 2.15 (Extremal Realisations are Configurations)**.** *Extremal realisations of a $\mathcal{F}am_\equiv$ $(\mathcal{F}, \equiv_\mathcal{F})$ on E, exactly correspond to configurations of $ter(\mathcal{F})$.*
That implies that if we define : $Top_\mathcal{F} : ter(\mathcal{F}) \to E : (R, \leq_R, n_R) \mapsto n_R(Top(R))$, then :

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

Proof. If we take an extremal realisation $(R, \leq_R, n_R)$, then , by the Proposition 1.42, we have that R correspond to a set of elements of E, which is down-closed (by isomorphism, and by definition of a realisation), and consistent (by definition of a realisation), do it correspond to a configuration.
Reciprocally, from a configuration $X \in \mathcal{C}(ter(\mathcal{F}))$, we can build a realisation $(X, \leq_X, n_X)$ with $\leq_X = \leq_{ter(\mathcal{F})}$ restricted to X and $n_X(p) = Top(p)$. We extract an extremal realisation with the same elements. Using what has been proved just before, this extremal realisation correspond to X. □

Theorem 2.16 (The *ter* pseudo-functor preserves and reflects properties (on equivalence classes of configurations))**.**
Let $(\mathcal{F}, \equiv_{\mathcal{F}})$ *be a* $\mathcal{F}am_{\equiv}$ *on* E*, and* $(P, \leq_P, Con_P, \equiv_P) = ter(\mathcal{F}, \equiv_{\mathcal{F}})$
Using the definition of the ter pseudo-functor, we can identify $\equiv_{\mathcal{F}}$ *and* $\equiv_P$*. In other words, we can define* $\equiv\, \subseteq\, (E \uplus P) \times (E \uplus P)$ *such that :*

$$a \equiv b \iff \begin{cases} a \equiv_{\mathcal{F}} b & \text{with } a, b \in E \\ a \equiv_P b & \text{with } a, b \in P \\ a = n_b(Top(b)) & \text{with } a \in E,\ b \in P \\ n_a(Top(a)) = b & \text{with } a \in P,\ b \in E \end{cases}$$

We define $\mathcal{C}_{\equiv}(P) = \{X_{\equiv} \mid X \in \mathcal{C}(G)\}$*, and* $\mathcal{C}_{\equiv}(\mathcal{F}) = \{X_{\equiv} \mid X \in \mathcal{F}\}$ *in a similar way. Then we have :*

$$\mathcal{C}_{\equiv}(\mathcal{F}) = \mathcal{C}_{\equiv}(P)$$

That implies that any property (such as a logical formulas [35]*) on equivalence classes of configurations is preserved and reflected by the ter functor.*

Proof. By definition of *ter*, and using the Proposition 2.15, $nt : P \to E : (R, \leq_R, n_R) \to n_R(Top(R))$ preserves and reflects equivalence, preserves configurations, and reach all configurations. [36]
Because nt preserves and reflects equivalence, we have no problem defining $\equiv$.
We have $\mathcal{C}_{\equiv}(\mathcal{F}) \subseteq \mathcal{P}(E_{\equiv}) = \mathcal{P}(P_{\equiv}) \supseteq \mathcal{C}_{\equiv}(ter(G))$.
Because nt preserves configurations, and reach all configurations, we can define $nt_{\equiv} : \mathcal{C}_{\equiv}(P) \to \mathcal{C}_{\equiv}(\mathcal{F})$ a surjective function.
We can decompose what does $nt_{\equiv}$ on $X \in \mathcal{C}_{\equiv}(P)$ in that way :

- Taking $Y \in \mathcal{C}(P)$ such that $Y_{\equiv} = X$.
- Applying nt to Y.
- Going to the equivalence classes.

By the Proposition 2.15, Y correspond to an extremal configuration $(R, \leq_R, n_R)$ of G, and $nt(Y)$ correspond to $n_R(R)$. ☐

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'

Proof. $g = restr(f)$ is well defined because is an event p is mapped to an event q, and g need two different equivalent events, then p need two different equivalent events ([Preserve Configurations] property).
The one-to-one correspondence of maps is immediate. ☐

In a similar way, there is a sequence of enriched adjunction between $PES_{\equiv}$, EDC^{weak}, EDC, EDC^{not}, and an adjunction [37] (not enriched) between EDC^{not} and PES.

35. For example, the property "for all configurations, if there is an event of the equivalence classes a, then there is an event of the equivalence classes b, and no events of the equivalence classes c" is preserved and reflected by ter. It also work for properties concerning only a subset of "all configurations".
36. It mean that any configuration of $\mathcal{F}$ correspond to at least one configuration of P.
37. The right adjoint is "forgetting the equivalence relation", and the left adjoint is the inclusion functor.

2.5 The composite adjunction

The Figure 23 [38] sum-up all the precedent adjunctions. Some of them still work if we take relations [39] instead of functions.
Because the adjunction between EDC (or EDC^{not}) and PES is not enriched, and the $\equiv$-adjunction between $\mathcal{F}am_{\equiv}$ and $PES_{\equiv}$ is not an adjunction, we cannot deduce an $\equiv$-adjunction (nor an adjunction) between PES and GES.

Most of the immediate adjunctions (or $\equiv$-adjunction) are trivially reflection [40] or co-reflection [41], but the fact that the adjunction between $\mathcal{F}am$ and EDC (or $PES_{\equiv}$) is a reflection is more complicated.

Proposition 2.18 (The $\equiv$-adjunction between $\mathcal{F}am$ and EDC is a reflection)**.** *The co-unit of the $\equiv$-adjunction between $\mathcal{F}am$ and EDC is an isomorphism : $\epsilon_{\mathcal{F}} : col \circ \iota \circ ter \circ \mathcal{I}(\mathcal{F}) \to \mathcal{F}$ (where ι 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 $\equiv$-adjunction between $\mathcal{F}am$ and $PES_{\equiv}$ is also an isomorphism.

Proof. Events of $\mathcal{F}$ correspond to events of $\mathcal{I}(\mathcal{F})$, which correspond to equivalence classes of $ter \circ \mathcal{I}(\mathcal{F})$.
We remark that if an event of $ter \circ \mathcal{I}(\mathcal{F})$ break the EDC property, then there exits [42] an equivalent event that does not break the EDC property.
That mean that equivalence classes of $ter \circ \mathcal{I}(\mathcal{F})$ correspond to equivalence classes of $\iota \circ ter \circ \mathcal{I}(\mathcal{F})$, which correspond to events of $col \circ \iota \circ ter \circ \mathcal{I}(\mathcal{F})$. That mean that events of $\mathcal{F}$ correspond to events of $\epsilon_{\mathcal{F}}(\mathcal{F})$.
We have to prove that they have the same configurations.

A configuration $X \in \mathcal{F}$ correspond to at least one extremal realisation $(X, \leq_X, id_X)$ of $\mathcal{F}$. By the Property 2.15, it correspond to a configuration Y of $ter \circ \mathcal{I}(\mathcal{F})$. Because the name function id_X is injective, Y is a configuration that respect the EDC property, so Y is also a configuration of $\iota \circ ter \circ \mathcal{I}(\mathcal{F})$. The functor col preserves configurations, so Y correspond to a configuration on $\epsilon_{\mathcal{F}}(\mathcal{F})$, so configurations of $\mathcal{F}$ are include in configurations of $\epsilon_{\mathcal{F}}(\mathcal{F})$.

We nee to show that any configuration $X' \in \epsilon_{\mathcal{F}}(\mathcal{F})$ correspond to a configuration of $\mathcal{F}$. Let $(E, \vdash_E, Con_E)$ be the replete GES corresponding to $\mathcal{F}$ and $(E', \vdash_{E'}, Con_{E'})$ the replete GES corresponding to $\epsilon_{\mathcal{F}}(\mathcal{F})$. We now need to show that any consistent set $X' \in Con_{G'}$ correspond to a consistent set $X \in Con_G$, and that any enabling $X' \vdash_{G'} e'$ correspond to an enabling $X \vdash_G e$.

If $X' \in Con_{G'}$, it exists $Y \in Con_{\iota \circ ter \circ \mathcal{I}(\mathcal{F})}$ such that $Y_{\equiv_{\iota \circ ter \circ \mathcal{I}(\mathcal{F})}} \supseteq X'$. Moreover, Y correspond to a consistent set in $ter \circ \mathcal{I}(\mathcal{F})$, so it correspond to a consistent set Z in $\mathcal{F}$. We

38. This figure use colors in order to be more readable.
39. See 3.3 for more details.
40. 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.
41. 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.
42. A way of building it is taking the top extremal realisation corresponding to the initial event. Then, when an event appear multiple times, replace the down closure of all of them by the down closure of one of them. It give a top realisation, and by the Property 1.39, we have a top extremal realisation which work, and this realisation correspond to an equivalent event that does not break the EDC property.

have X' which correspond to $X \subseteq Z \in Con_G$, so $X \in Con_G$.

Instead of showing that any enabling $X' \vdash_{G'} e'$ correspond to an enabling $X \vdash_G e$, we will show that any non-enabling $X \not\vdash_G e$ correspond to a non-enabling $X' \not\vdash_{G'} e'$. Saying $X \not\vdash_G e$ is saying that $\forall Z \vdash_G e,\ Z \backslash X \neq \varnothing$. That mean that for all $(R, \leq_R, n_R) \in ter \circ \mathcal{I}(\mathcal{F})$ such that $top(R) = e,\ R \backslash (X \cup \{e\}) \neq \varnothing$. So we have :

$$\forall (R, \leq_R, n_R) \in \iota \circ ter \circ \mathcal{I}(\mathcal{F}) \text{ such that } top(R) = e,\ R \backslash (X \cup \{e\}) \neq \varnothing$$

After applying the *col* functor, we have $X' \not\vdash_{G'} e'$. □

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 (replete) GES.

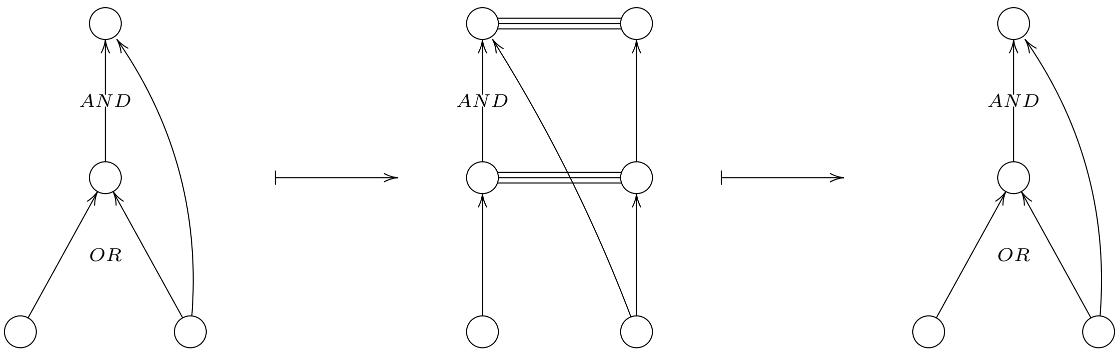


FIGURE 21 – Example of the co-unit being an isomorphism

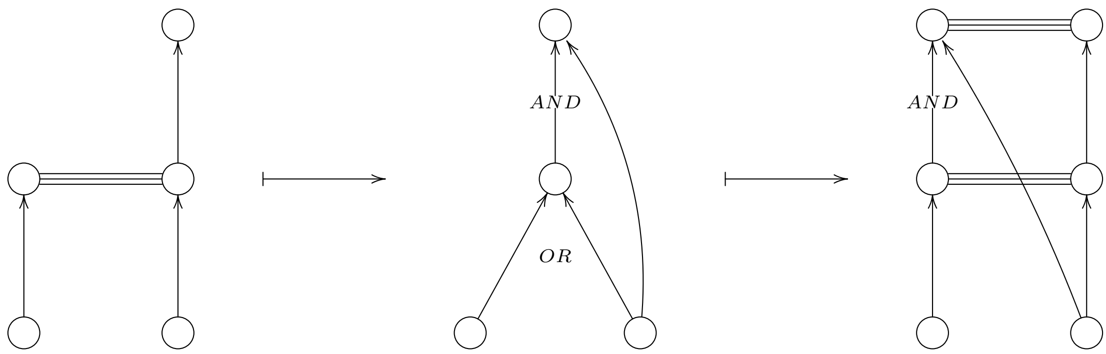


FIGURE 22 – Example of the unit not being an isomorphism (the missing property is [Shortcut])

Proposition 2.19 (Characterisation of $PES_\equiv$ that come from GES)**.**
A $PES_\equiv$ $(P, \leq_P, Con_P, \equiv_P)$ come from a GES if and only if :

- (Consistency up to equivalence)

$$\forall X, Y \in \mathcal{P}(P), \begin{cases} X \text{ down closed for } \leq_P \\ Y \in Con_P \\ X_{\equiv_P} = Y_{\equiv_P} \end{cases} \iff X \in \mathcal{C}(P)$$

- (No multiplicity) $\forall p,q \in P, \begin{cases} p \equiv_P q \\ [p) = [q) \end{cases} \implies p = q$
- (No inclusion [43]) $\forall p,q \in P,\ \forall X \in \mathcal{P}(P), \begin{cases} p \equiv_P q \\ X \ \textit{down closed for} \ \leq_P \\ X \subseteq [p) \\ [q)_{\equiv_P} \subseteq X_{\equiv_P} \end{cases} \implies X = [p)$
- (Shortcut [44])

$$\begin{cases} X, Y, X \cup Y \in \mathcal{C}(P) \\ \{e_i \mid i \in I\} = Y_{\equiv_P} \backslash X_{\equiv_P} \end{cases} \implies \exists \{t_i\}_{i \in I} \in P^I,\ X \cup \{t_i \mid i \in I\} \in \mathcal{C}(P),\ \forall i \in I,\ t_i \in e_i$$

Proof. We remark that [Shortcut] is equivalent to :

$$\begin{cases} X, T \in \mathcal{C}(P) \\ X \subseteq T \\ \{e_i \mid i \in I\} = T_{\equiv_P} \backslash X_{\equiv_P} \end{cases} \implies \exists \{t_i\}_{i \in I} \in P^I,\ X \cup \{t_i \mid i \in I\} \in \mathcal{C}(P),\ \forall i \in I,\ t_i \in e_i$$

If we suppose [Consistency up to equivalence], then, by induction, [Shortcut] is equivalent to :

$$\begin{cases} X, T \in \mathcal{C}(P) \\ X \subseteq T \\ \{e\} = T_{\equiv_P} \backslash X_{\equiv_P} \end{cases} \implies \exists t \in e,\ X \cup \{t\} \in \mathcal{C}(P)$$

We will first prove that those four properties are needed.
By the way the consistency of the image by ter is defined, we have immediately that a $PES_{\equiv}$ $(P, \leq_P, Con_P, \equiv_P)$ which come from a GES $(E, \vdash_E, Con_E)$ respect the [Consistency up to equivalence] property.
Using the Proposition 1.42, if $p \equiv_P q$ and $[p) = [q)$, then p and q correspond to the same top extremal realisation, so are equals, so we have [No multiplicity].
Using the Proposition 1.43, we have the if the [No inclusion] property is broken, then the corresponding realisation is not extremal. So we have [No inclusion].
We take $X, T \in \mathcal{C}(P)$, with $X \subseteq T$ and $T_{\equiv_P} \backslash X_{\equiv_P} = \{e\}$ with $e \in E$. By the Proposition 2.15, T and X correspond to two extremal realisations $(T, \leq_T, n_T)$ and $(X, \leq_X, n_X)$ of E, so $T_{\equiv} \in \mathcal{C}(E)$, and so $T_{\equiv} \backslash \{e\} \vdash_E e$. We can create $(X \cup \{e\}, \leq_{X \cup \{e\}}, n_{X \cup \{e\}})$ the realisation obtained from X by adding e as a top element. By the Proposition 1.39, we can extract a top extremal realisation t from $(X \cup \{e\}, \leq_{X \cup \{e\}}, n_{X \cup \{e\}})$, and by the Proposition 1.42, we have that $X \cup \{t\} \in \mathcal{C}(P)$. So we have [Shortcut].
No, we will prove that those four properties characterise $PES_{\equiv}$ which come from a GES. More precisely, we will prove that if we apply col and then ter, we obtain something isomorphic. We take $(P, \leq_P, Con_P, \equiv_P)$ a $PES_{\equiv}$ respecting the four properties, and we note $(E, \vdash_E, Con_E)$ the GES obtained after col. We will try to build an isomorphism between P and $ter(E)$, it is equivalent to build an isomorphism F between $\mathcal{C}(P)$ and $\{(R, \leq_R, n_R)$ extremal realisation of $E\}$.
We define the function $F : \mathcal{C}(P) \to POM(E)$ where $POM(E)$ corresponds to all the POM on E (up to isomorphism), by $F : X \mapsto (X, \leq_X, n_X)$ with $n_X : p \to \{p\}_{\equiv}$ and $\leq_X = \leq_P$ restricted to X. We have immediately that for all $X \in \mathcal{C}(P)$, $F(X)$ is a realisation of E.
Using the characterisation of extremal realisations (Proposition 1.43), and by [No multiplicity] and [No inclusion], we have that for all $X \in \mathcal{C}(P)$, $F(X)$ is an extremal realisation of E.
Using [No multiplicity], and doing a trivial induction on $\mathcal{C}(P)$, we have that F is injective.

43. We recall that $[p]_{\equiv_P} = \{\{\tilde{p}\}_{\equiv_P} \mid \tilde{p} \leq_P p\}$.
44. This property say in the idea that there always exists a path which use at most one element by equivalence classes.

In order to prove that F is the expected isomorphism, we need to prove that for all extremal realisation $(R, \leq_R, n_R)$ of E, there exists $X \in \mathcal{C}(P)$ such that $F(X) = (R, \leq_R, n_R)$. We will prove it by induction on the order between extremal realisations.
We take $(R, \leq_R, n_R)$ and extremal realisation such that every extremal realisation lesser (for $\preceq_{sub}$) is reached by F.
If $(R, \leq_R, n_R)$ is not a top extremal realisation, we can wrote it as an union of lesser top extremal realisations (by the Proposition 1.42). Each of those realisations have an antecedent by F. Using [Consistency up to equivalence], and the definition of *col*, the union of those antecedent is a configuration $X \in \mathcal{C}(P)$. By the way F is defined, we have $F(X) = (R, \leq_R, n_R)$.
If $(R, \leq_R, n_R)$ is a top extremal realisation, with $Top(R) = r$, we define $(R \backslash \{r\}, \leq_{R\backslash\{r\}}, n_{R\backslash\{r\}})$ the realisation corresponding to $(R, \leq_R, n_R)$ without its top, and we have $Y \in \mathcal{C}(P)$ such that $F(Y) = (R\backslash\{r\}, \leq_{R\backslash\{r\}}, n_{R\backslash\{r\}})$. Because $(R, \leq_R, n_R)$ is a realisation, we have $n_R(R)$ configurations of E. By the definition of *col*, we have $Z \in \mathcal{P}(P)$ down-closed such that $Z_{\equiv_P} = n_R(R)$, and $W \in Con_P$ such that $W_{\equiv_P} = n_R(R)$. By [Consistency up to equivalence], $Z \in \mathcal{C}(P)$.
We have $\{n_R(r)\} = Z_{\equiv_P} \backslash Y_{\equiv_P}$, so by [Shortcut] (with $T = Z \cup Y \in \mathcal{C}(P)$ by [Consistency up to equivalence]), we have $t \in n_R(r)$ such that $Y \cup \{t\} \in \mathcal{C}(P)$.
$F(Y \cup \{t\})$ coincide with $(R, \leq_R, n_R)$ except, possibly, for the top element of $(R, \leq_R, n_R)$ which may not be a top element for $F(Y \cup \{t\}$. But $(R, \leq_R, n_R)$ is extremal, and by the [Minimal] property of the characterisation of extremal realisations (Proposition 1.43), the top element cannot be enabled with less elements, so $F(Y \cup \{t\}) = (R, \leq_R, n_R)$.
So F is an isomorphism. ☐

Corollary 2.20 (Characterisation of $PES_{\equiv}$ that come from GES)**.**
An EDC $(P, \leq_P, Con_P, \equiv_P)$ come from a GES if and only if :

- (Consistency up to equivalence)

$$\forall X, Y \in \mathcal{P}(P), \begin{cases} X \text{ down closed for } \leq_P \\ Y \in Con_P \\ X_{\equiv_P} = Y_{\equiv_P} \end{cases} \iff X \in \mathcal{C}(P)$$

- (No multiplicity) $\forall p, q \in P, \begin{cases} p \equiv_P q \\ [p) = [q) \end{cases} \implies p = q$

- (No inclusion) $\forall p, q \in P,\ \forall X \in \mathcal{P}(P), \begin{cases} p \equiv_P q \\ X \text{ down closed for } \leq_P \\ X \subseteq [p) \\ [q)_{\equiv_P} \subseteq X_{\equiv_P} \end{cases} \implies X = [p)$

- (Weak Shortcut)

$$\begin{cases} X, Y, X \cup Y \in \mathcal{C}(P) \\ X, Y \text{ unambiguous} \\ \{e_i \mid i \in I\} = Y_{\equiv_P} \backslash X_{\equiv_P} \end{cases} \implies \exists \{t_i\}_{i \in I} \in P^I,\ X \cup \{t_i \mid i \in I\} \in \mathcal{C}(P),\ \forall i \in I,\ t_i \in e_i$$

Where unambiguous configurations are configurations X such that

$$p, q \in X \ \&\ p \equiv_P q \implies p = q$$

.

Proof. By remarking that if p break the [EDC property], and $q \geq_P p$, then q break the [EDC property], and using the Proposition 2.19, we have that those four properties are needed for begin an EDC which come from a GES.
We take an EDC $(P, \leq_P, Con_P, \equiv_P)$ which respect those properties, and we define $(E, \vdash_E, Con_E)$ the image by *col*.

As in the proof of the Proposition 2.19, we define the function $F : \mathcal{C}(P) \to POM(E)$, and we have in the same way that F is injective and for all $X \in \mathcal{C}(P)$, $F(X)$ is an extremal realisation.
We remark [45] that the proof of the surjectivity of F still work : by using the EDC property, we can take our configuration unambiguous. $\square$

45. The proof will be written properly in [WV15].

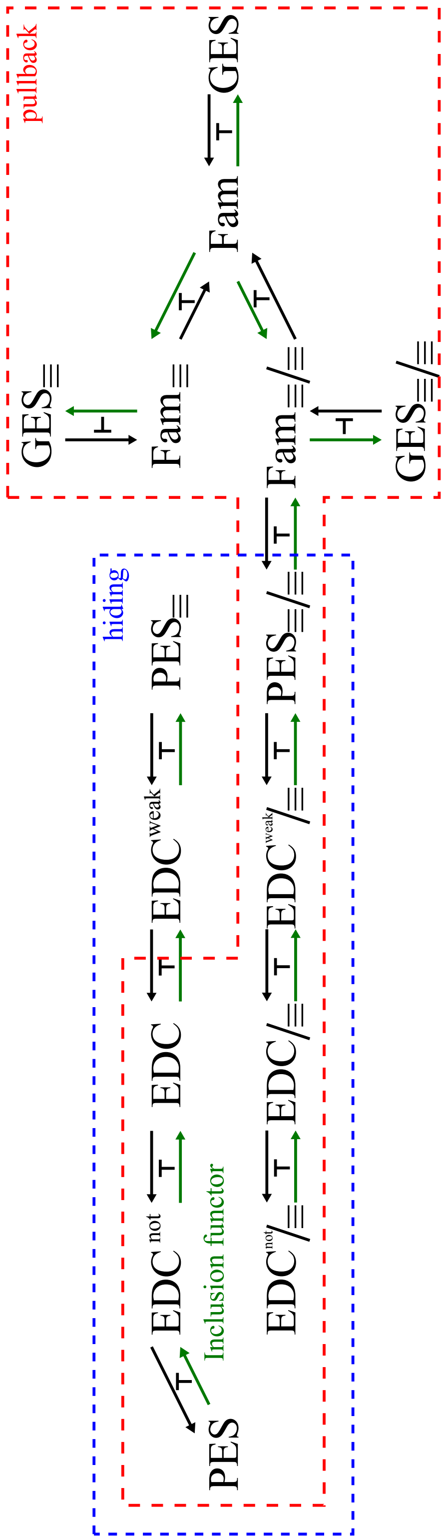


FIGURE 23 – The composite adjunction.

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 [46] 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.* [47] *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 [48]

Property 3.2 (Existence of pull-backs)**.** *We consider a category $\mathcal{C} \in \{GES, \mathcal{F}am, GES_{\equiv}, \mathcal{F}am_{\equiv}, PES_{\equiv}, EDC^{weak}, EDC, EDC^{not}, PES\}$.*
$\mathcal{C}$ 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.

Proof. We can define without problems the pull-back on $GES_{\equiv}$ and GES, and deduce a pull-back on $\mathcal{F}am$ and $\mathcal{F}am_{\equiv}$. Then, by the adjunction (and using the fact that the left adjoint is the inclusion) we have a by-pull-back on $PES_{\equiv}$, EDC^{weak}, and EDC. For EDC, we can proof that the by-pull-back corresponds to a pull-back, and deduce a pull-back on EDC^{not} and recover the well-known pull-back on PES.
See [WV15] for more precisions. (Paper in progress) □

EDC supports both hiding and pull-backs, but they have to be done carefully because these two operations do not commute with each other. The Figure 25 and the Figure 26 show that, on EDC, depending if we hide neutral events [49] before doing the pull-back, or after

46. The third event structure correspond to the part that is common between the two others.
47. When the category has a notion of total morphism and partial morphism, only total morphisms are taken.
48. An equivalence relation between morphisms induce an equivalence relation between objects : $A \equiv B \iff \exists f : A \to B,\ \exists g : B \to A,\ g \circ f \equiv id_A$.
49. The notion of polarities on events is useful for game and strategies, but have no consequences here. The example use event structure with polarities (and maps that respects all condition for being strategies) in order to show that adding polarities does not solve the problem.

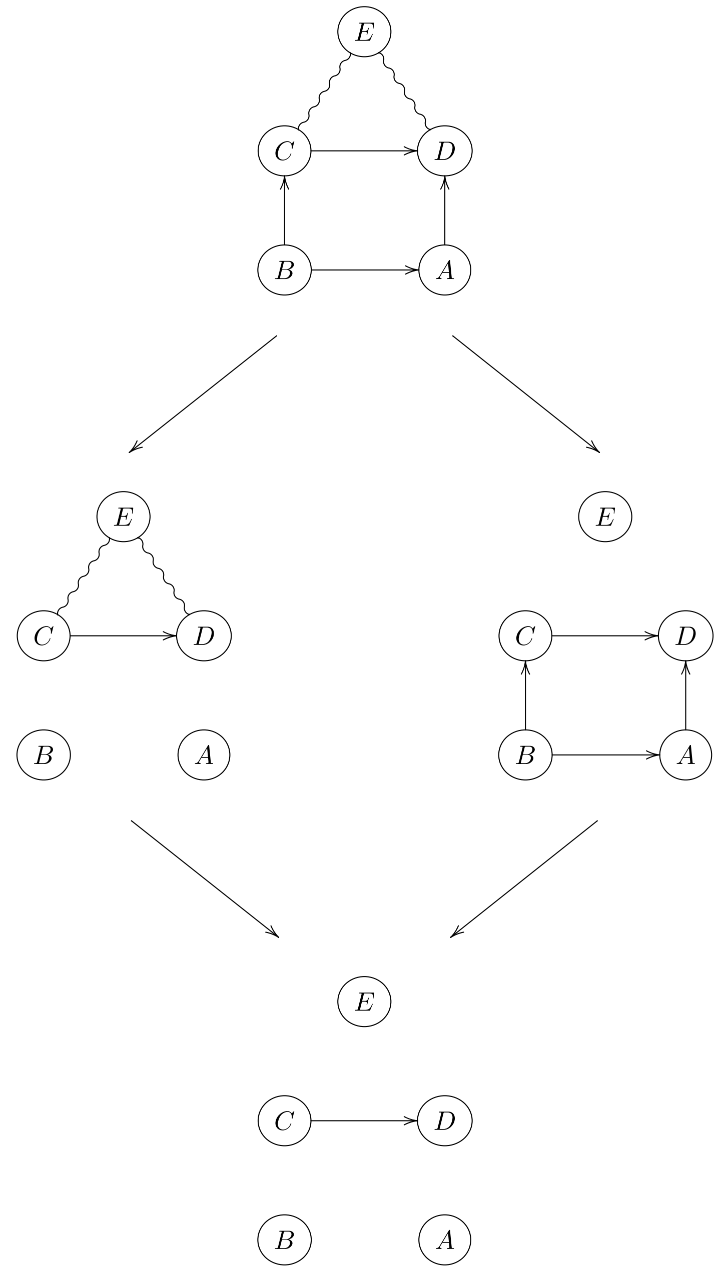


FIGURE 24 – Example of a pull-back on prime event structures.

doing it, we can have different results. We do not have this problem on EDC^{not} and on PES.

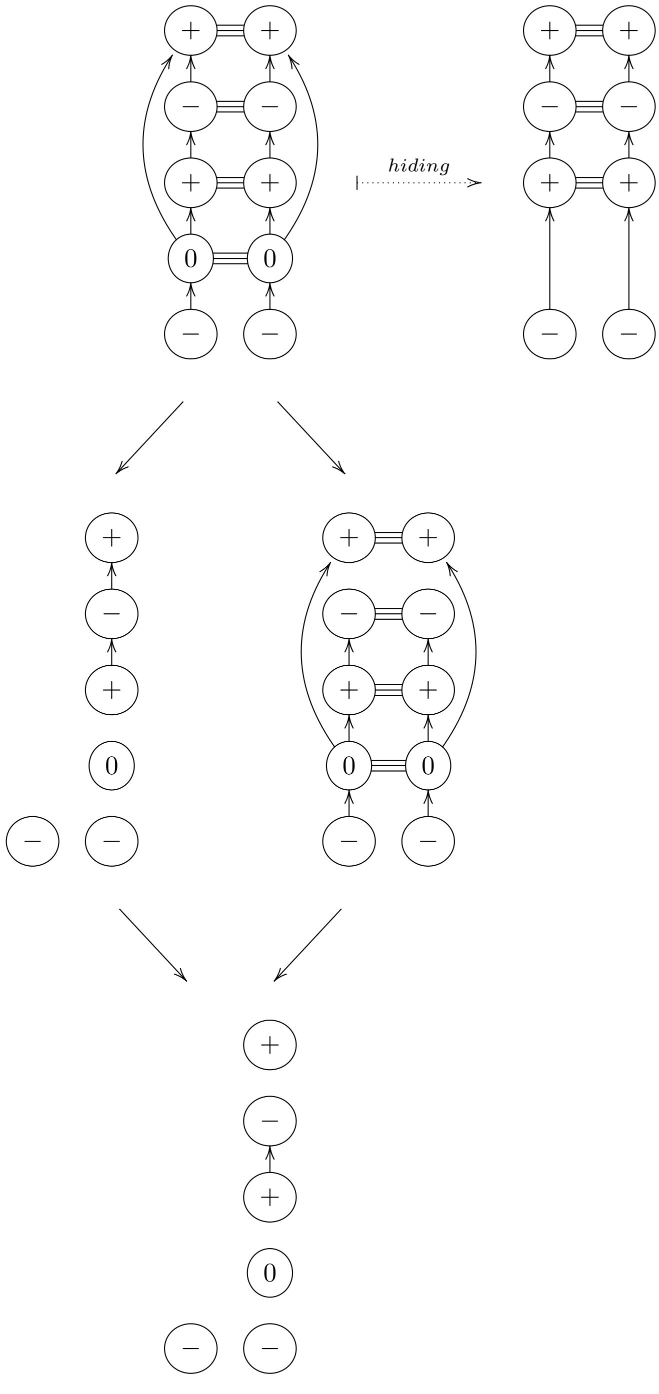


FIGURE 25 – If we apply the pull-back before the hiding.

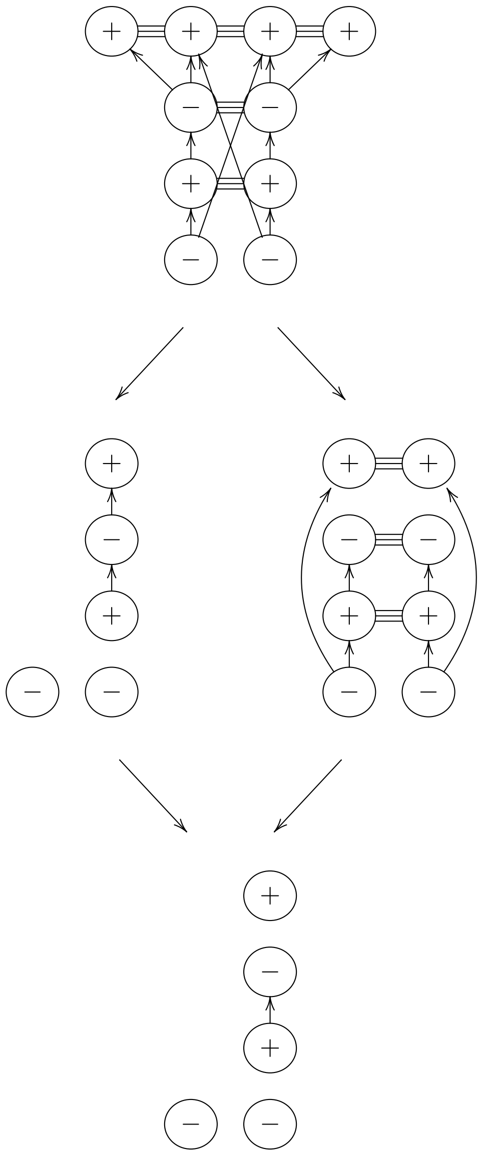


FIGURE 26 – If we apply the pull-back after the hiding.

3.2 Relations instead of functions

A lot of things also work with relations instead of functions.

Definition 3.3 (Relational maps on $GES_{\equiv}$)**.**
For a function $f : A \to \mathcal{P}(B)$*, and for* $X \subseteq A$*, we define* $f(X) = \bigcup_{x \in X} f(x) \in \mathcal{P}(B)$*.*
A relational 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 function*[50] $f : G \to \mathcal{P}(G')$ *such that :*

- (All or Nothing) $\forall a \equiv_G b \in G,\ [\ f(a) = \varnothing \iff f(b) = \varnothing\ ]$
- (Preserve Equivalence) $\forall a \equiv_G b \in G,\ \forall a' \in f(a),\ \forall b' \in f(B),\ a' \equiv_{G'} b'$
- (Locally *equiv*-Injective) $\forall X \in Con_G,\ \forall a \not\equiv_G b \in X,\ \forall a' \in f(a),\ \forall b' \in f(b),\ a' \not\equiv_{G'} b'$
- (Preserve Configurations) $\forall X \in \mathcal{C}(G),\ f(X) \in \mathcal{C}(G')$

The last properties imply :

- (Preserve Consistency) $\forall X \in Con_G,\ f(X) \in Con_{G'}$
- (Preserve Enabling)[51] $\forall a \in G,\ \forall a' \in f(a),\ \forall X \vdash_G a,\ f(X \cup \{a\}) \vdash_{G'} a'$

Definition 3.4 (Equivalence on relational maps)**.** *We will say that two relational 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 :*

- $\forall a \in G,\ \forall b \in f(a),\ \exists \tilde{b} \in g(a),\ b \equiv_H \tilde{b}$
- $\forall a \in G,\ \forall \tilde{b} \in g(a),\ \exists b \in f(a),\ \tilde{b} \equiv_H b$

Property 3.5.

- (1) *All the enriched categories are still enriched categories if we use relational maps instead of functional maps.*
- (2) *All the enriched adjunctions between* GES*,* $\mathcal{F}am$*,* $GES_{\equiv}$*, and* $\mathcal{F}am_{\equiv}$ *work in the same way.*
- (3) *All the enriched adjunction between* $PES_{\equiv}$*,* EDC^{weak}*,* EDC *and* EDC^{not} *completely disappear*[52]
- (4) *The* $\equiv$*-adjunction between* $\mathcal{F}am_{\equiv}$ *and* $PES_{\equiv}$ *remain.*

Proof. The proof of (1) and (2) is similar to the functional case.
For (4), we can easily create counter example of this shape :

50. We can also see f as a binary relation between G and G'.
51. The $\cup\{a\}$ is here to represent the fact that an events can need equivalent events to be enabled.
52. It mean that there is no adjunctions, and no $\equiv$-adjunctions based on the initial enriched adjunction.

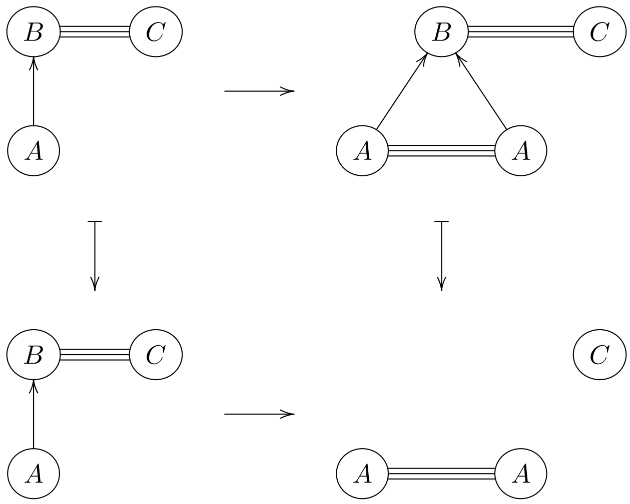


FIGURE 27 – Problem with relational maps between EDC^{weak} and EDC : we can lose the [All or Nothing] property.

The proof of (3) is quite complicated, because the notion of realisation too strong and have to be re-written. So we will only describe the different steps of the proof.
We call *pre-realisation* a weak version of a realisation where we allow to have cycles (so we have a pre-order) and infinite down-closures (in order to allow infinite cycles).
The image of a realisation by a relational map is a pre-realisation. Contrary to the functional case, we do more that just changing the name function : if an event e is mapped to N events, then we duplicate the elements that have the name e in N different copy (corresponding to the N images), and we put this copy in a same cycle (in infinite cases, it can create infinite cycles, but we always have a finite number of cycles).
Then we have to prove that there exists at least one realisation lesser than this pre-realisation, by $\preceq_{fun}$. We can prove it by a trans-finite induction (the fact that there is a finite number of cycle, and that replete $GES_{\equiv}$ have finite enabling, are the main argument).
Then, the properties corresponding to the functional cases can be extended in order to finish the proof. $\square$

4 Games and Strategies

We will use PES in order to models games, and EDC to models strategies [53]. 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. [54]

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 $p : A \to \{\oplus, \ominus\}$.

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 $A \parallel B$ 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 [55], 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, σ) is a pre-strategy on the game A if S is an EDC and $\sigma : S \to A$ is a map [56] of EDC which respect polarities.

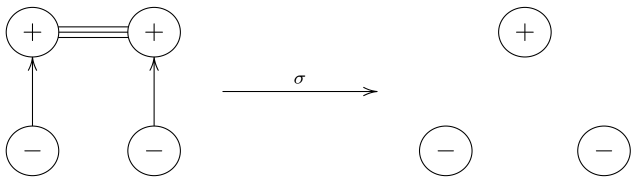


FIGURE 28 – Example of a pre-strategy $\sigma : S \to A$

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, σ) is a pre-strategy from the game A to the game B if (S, σ) is a pre-strategy on the game $A^\perp \parallel B$.*

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. [57] We will do the composition by synchronising (with a pullback and with hiding) the two pre-strategies.

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

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

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

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

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 $A^{\perp} \parallel A$ 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, γ_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, σ) is a strategy from the game A to the game B if (S, σ) 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, σ) is a strategy on A if (S, σ) is a strategy from $\varnothing$ to A.*

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

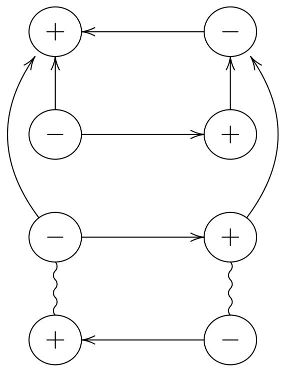


FIGURE 29 – Example of a copy-cat on a game A.

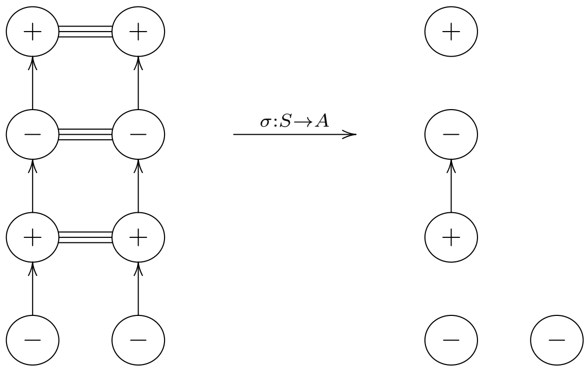


FIGURE 30 – Example of a strategy $\sigma : S \to A$

- ($\equiv$-injectivity [58]) $\sigma(s_1) = \sigma(s_2) \implies s_1 \equiv_S s_2$
- ($\ominus$-receptivity [59])
  $\begin{cases} X \in \mathcal{C}(S) \\ \sigma(X) \cup \{a\} \in \mathcal{C}(A) \\ p_A(a) = \ominus \end{cases} \implies \exists s \in S \ \equiv_S, \ \begin{cases} X \cup \{s\} \in \mathcal{C}(S) \\ \sigma(s) = a \end{cases}$
- (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)
  $X \in Con_S \iff [X^{\oplus}] = \{s \in S \mid \exists x \in X, \ s \leq_S x \ \& \ p_S(x) = \oplus\} \in Con_S$
- (Innocence)

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

$$s_1 \leq_S s_2 \implies \exists t_1, t_2 \in S,\ s_1 \leq_S t_1 \leq_S t_2 \leq_S s_2,\ \begin{cases} \sigma(t_1) \leq_A \sigma(t_2) \\ OR \\ p_S(t_1) = \ominus \ \&\ p_S(t_2) = \oplus \end{cases}$$

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.
- (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 [60].
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).

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

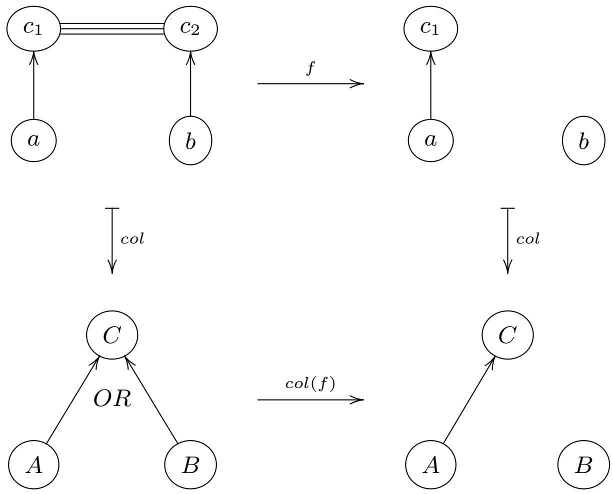
5 Examples and counter-examples

Following examples could help to understand why some properties are needed.

5.1 About the choice of the different categories

5.1.1 The All-or-Nothing property on maps

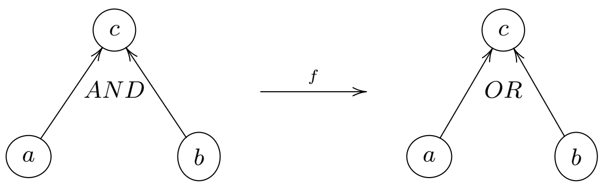
This following example shows that the image by *col* of a map which does not respect the [All or Nothing] property can give a map which does not preserves enabling.



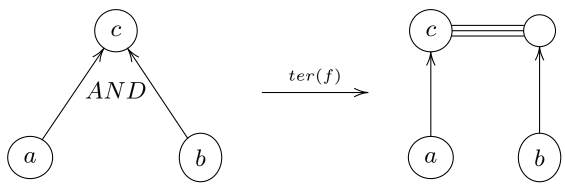
Here, $\{B\} \vdash A$ in the initial event structure, but $ter(f)(\{B\}) \not\vdash ter(f)(A)$.

5.1.2 The equivalence relation between maps

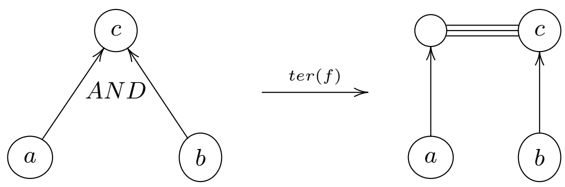
To have an adjunction, we need to have a unique map image by *ter* of the following map :



But we have two possible choices :



Or :



So we need to say that the two possibles maps are essentially the same, this means we need an equivalence relation between maps.

5.1.3 No consistency condition on the definition of equivalence between maps

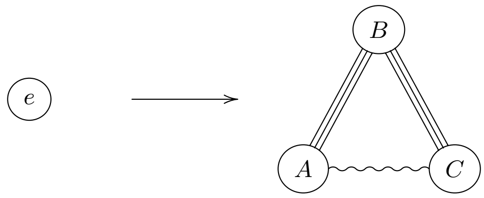
We define $f \equiv g$ if and only if :

- $\mathcal{D}(f) = \mathcal{D}(g)$
- $\forall a \in G,\ f(a) \equiv_H g(a)$

But we would want to add a consistency condition, which mean $f \equiv g$ if and only if :

- $\mathcal{D}(f) = \mathcal{D}(g)$
- $\forall a \in G,\ f(a) \equiv_H g(a)$
- $\{f(a), g(a)\} \in Con_H$

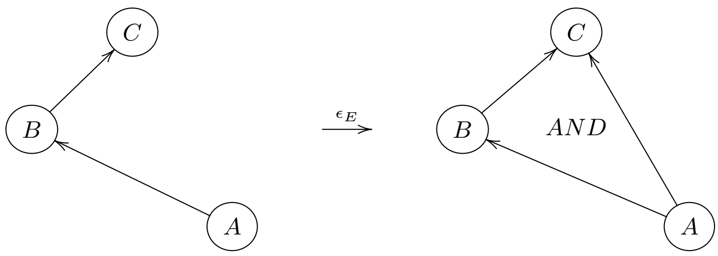
Unfortunately, this relation is not transitive :



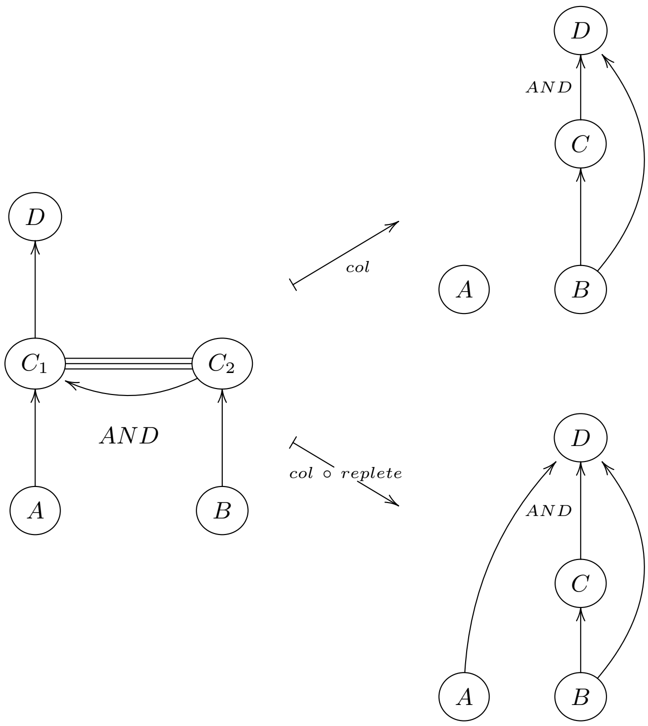
Here, the map $f : e \mapsto \{A\}$ is equivalent to the map $g : e \mapsto \{B\}$, which is equivalent to the map $h : e \mapsto \{C\}$. But with the definition with consistency, we have $f \not\equiv h$ because $\{A, C\} \notin Con_H$.

5.1.4 Using replete GES and $GES_{\equiv}$

We use $\mathcal{F}am$ and $\mathcal{F}am_{\equiv}$, which correspond to replete GES and replete $GES_{\equiv}$, because the fact that enabling are not necessarily down-closed can cause some problems.
If we do not take the replete condition on GES, then the co-unit ϵ_E (between GES and EDC) is not an isomorphism :



If we do not take the replete condition on $GES_{\equiv}$, the *col* functor does not work as we want :

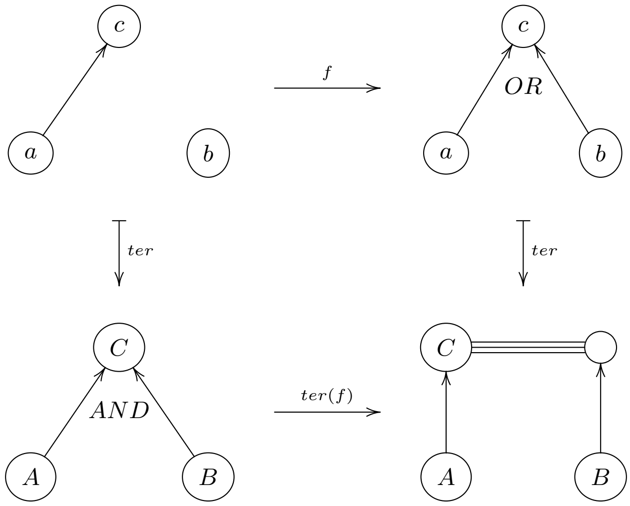


Here, we take a non-replete $GES_{\equiv}$, that mean that D does not need A to be enabled, but need C_1 which need A, so their is no configurations with D and without A. We lose this information by *col*. If we first complete the $GES_{\equiv}$ by transitivity (action which should change nothing to the result [61]), then we preserve this information by *col*.

61. Configurations correspond to all possible states of the system, and if two system have the same possible states (and the same relation between them), they should have the same behaviour. Completing by transitivity the enabling does not change configurations, so it should not change the behaviour.

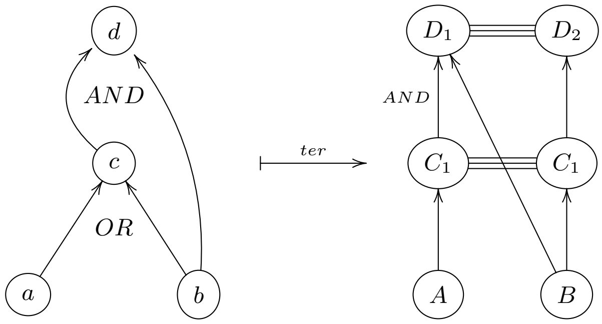
5.1.5 We cannot use only surjectives maps

The image by *ter* of a surjective map can be not surjective :



5.2 About the definition of *ter*

5.2.1 Irreducible configurations



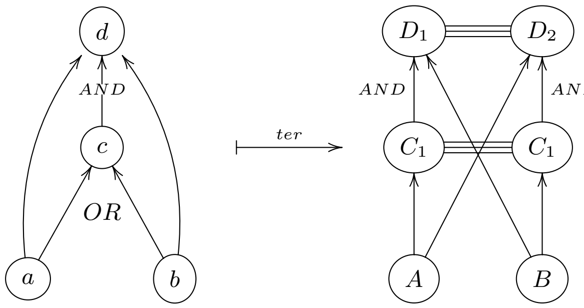
Here, the event D_1 represent d being enabled by a, b, and c, and the event D_2 represent d being enabled by b and c. So D_1 seems useless.

If you look at irreducible configurations [62], we build directly, in this case, $ter(G)$ without the "useless" event. But using irreducible configurations instead of top extremal realisation causes problems in general :

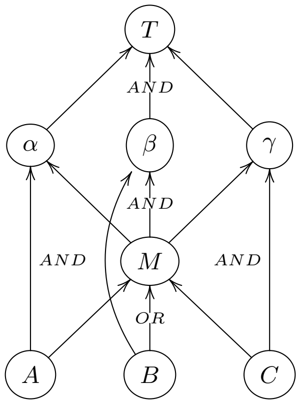
First, you sometimes need to create multiple copy of the same irreducible (disambiguation

62. A configuration is irreducible if it not the union of strictly smaller configurations.

step), for example in this case :

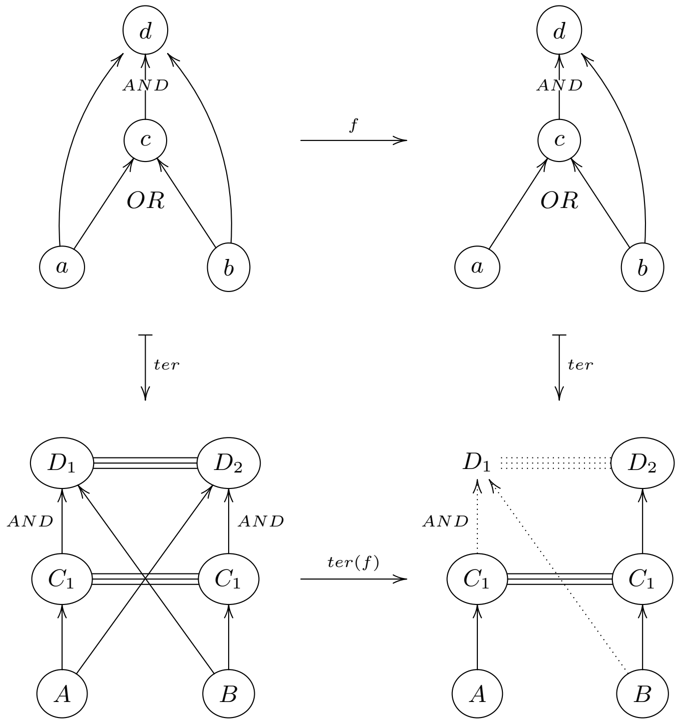


This disambiguation step can be very complicated (or can fail, depending the way you defined it) in some more complicated cases :



Second, the "useless" event is not useless, it is needed for *ter* being a pseudo-functor. In fact, without this event, we have image of some maps which does not respect the [All or

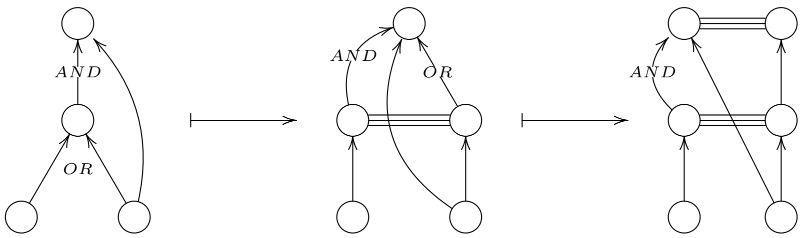
Nothing] property :



So using irreducible configurations does not work.

5.2.2 An inductive construction of *ter*

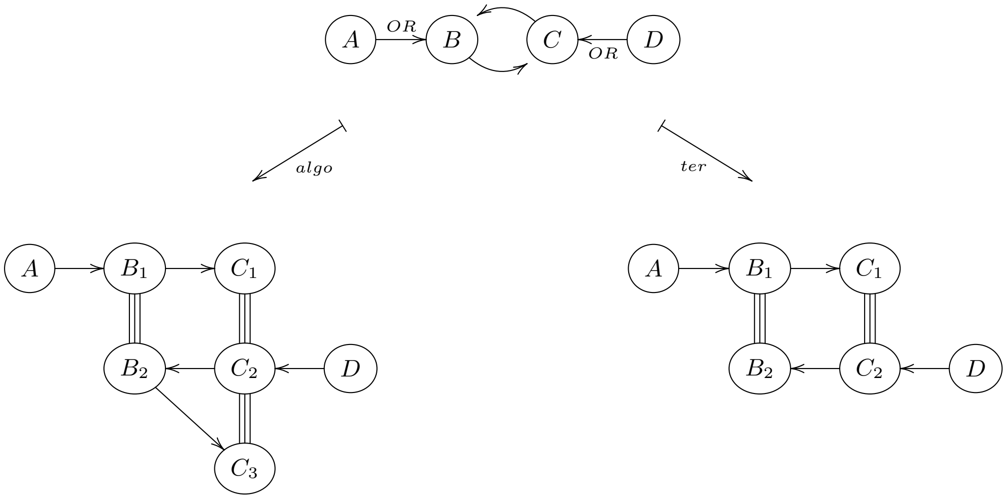
In the idea, the *ter* pseudo-functor duplicate events, making on copy for each way of enable it. We can do it inductively, duplicating events for each of its minimal enabling :



In fact, one can prove that, at each step of this algorithm, the realisations of the event structure are the same, except for the name function. You can also prove that extremal realisation of the initial event structure remain extremal at each step.

Unfortunately, in some cases, you can create new extremal realisations. [63] For example :

63. By changing the name function, some elements of a realisation which use to have the same name can now have different names. If there is no loops, and if we do the inductive operation from the root to the top, no supplementary extremal realisations shall be created.



In this example, the algorithm can build two different event structures, depending of the order of the iteration of the inductive step, an each of the event structure has one more event comparing to *ter*. This event correspond to a realisation which was not extremal, but became extremal.

An important point is that the new extremal realisations always break the *EDC* property [64], so we can make without problems an inductive definition of the functor from $\mathcal{F}am_{\equiv}$ to *EDC*.

Definition 5.1 (The *dup* operation)**.** *We take a replete* [65] $GES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_G)$*. We take* $e \in G$*.*

$$(G, \vdash_G, Con_G, \equiv_G) \xmapsto{dup_e} (G', \vdash_{G'}, Con_{G'}, \equiv_{G'})$$

Where :

- $\mathcal{X} = \{X \in \mathcal{P}(G) \mid X \vdash^{\mu}_{G} e\}$
- $G' = (G \backslash \{e\}) \uplus \mathcal{X}$
- $n : G' \to G : \begin{cases} a \in G \mapsto a \\ a \in \mathcal{X} \mapsto e \end{cases}$
- $a \equiv_{G'} b \iff n(a) \equiv_G n(b)$
- $X \in Con_{G'} \iff n(X) \in Con_G$
- $X \vdash_{G'} a \iff \begin{cases} a \in G \;\&\; n(X) \vdash_G a \\ OR \\ a \in \mathcal{X} \;\&\; n(X) \supseteq a \end{cases}$

So we duplicate the event e for each of his minimal enabling (and each copy have now a unique way of being enabled).

You can define a symmetric operation, which allow to compute the *col* functor in an inductive way. [66]

Definition 5.2 (The *merge* operation)**.** *We take a replete* [67] $GES_{\equiv}$ $(G, \vdash_G, Con_G, \equiv_G)$*. We take* $\mathcal{E} \in G_{\equiv_G}$*.*

$$(G, \vdash_G, Con_G, \equiv_G) \xmapsto{merge_{\mathcal{E}}} (G', \vdash_{G'}, Con_{G'}, \equiv_{G'})$$

64. $p, p' \leq q \;\&\; p \equiv p' \implies p = p'$.
65. The *dup* operation should be defined on $\mathcal{F}am_{\equiv}$, but it is easier to describe it on replete $GES_{\equiv}$.
66. No problems with this direction.
67. The *merge* operation should be defined on $\mathcal{F}am_{\equiv}$, but it is easier to describe it on replete $GES_{\equiv}$.

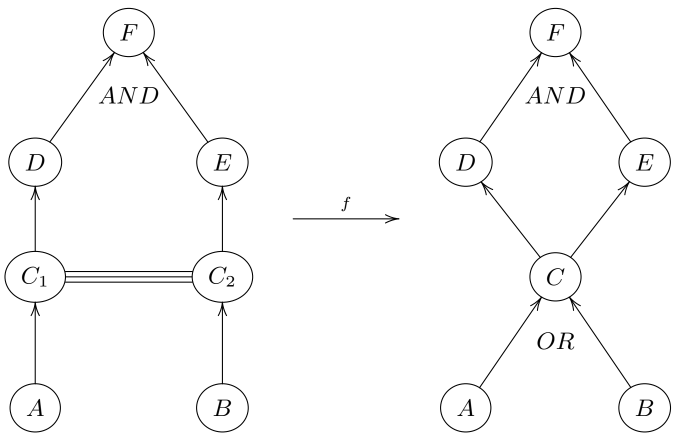
Where :

- $G' = (G \backslash \mathcal{E}) \uplus \{\mathcal{E}\}$
- $n : G \to G' : \begin{cases} a \notin \mathcal{E} \mapsto a \\ a \in \mathcal{E} \mapsto \mathcal{E} \end{cases}$
- $a' \equiv_{G'} b' \iff \exists a \in n^{-1}(a'),\ \exists b \in n^{-1}(b'),\ a \equiv_G b$
- $X' \in Con_{G'} \iff \exists X \ \textit{such that}\ n(X) = X',\ X \in Con_G$
- $X' \vdash_{G'} a' \iff \exists X \ \textit{such that}\ n(X) = X',\ \begin{cases} a' \in G \ \&\ X \vdash_G a' \\ OR \\ a' = \mathcal{E} \ \&\ \exists e \in \mathcal{E},\ X \vdash_G e \end{cases}$

So we merge all the events of $\mathcal{E}$*.*

5.2.3 Multi-sets for Realisations

The fact that we need to allow to have multiple time the same event in a realisation is not obvious. In fact, if you forget $\mathcal{F}am_\equiv$, and just take the $\equiv$-adjunction between GES and EDC, you never need multi-sets. But is you want *ter* to be well-defined on $\mathcal{F}am_\equiv$, you have to find a image of this function $f : P \to Q$:

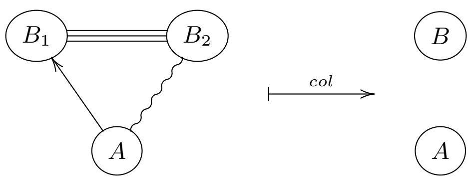


And more precisely, the top extremal realisation $(R, \leq_R, n_R)$ of P with $n_R(Top(R)) = F$ has to be mapped to a top extremal realisation of Q, and this realisation has to use multi-sets.

5.3 About the unit η between $PES_\equiv$ and GES

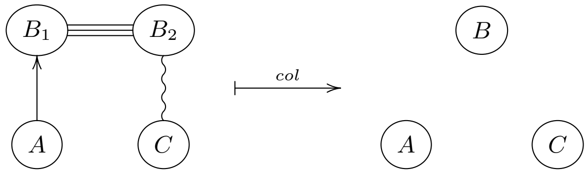
5.3.1 The *col* functor can loose information

The *col* functor merge events, and we can lose some informations about the initial event structure.



Here, we lose the order. More precisely, in the replete $GES_\equiv$, we can do A then B_1, or B_2 and then we cannot do A. In the replete GES, we can do A then B, but we also can do B

then A, which was not possible in the replete $GES_{\equiv}$.



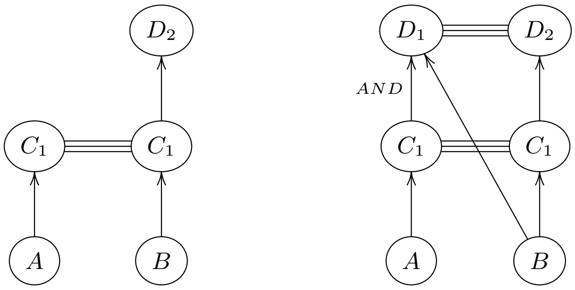
Here, the configuration $\{B, C\}$ in the replete GES correspond to no configurations in the replete $GES_{\equiv}$, so *col* has created configurations which does not correspond to any previous configuration.

5.3.2 About the [Shortcut] property

We recall the [Shortcut] property :

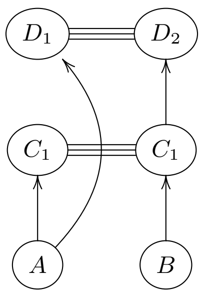
$$\begin{cases} X, Y, X \cup Y \in \mathcal{C}(P) \\ \{e_i \mid i \in I\} = Y_{\equiv_P} \backslash X_{\equiv_P} \end{cases} \implies \exists \{t_i\}_{i \in I} \in P^I,\ X \cup \{t_i \mid i \in I\} \in \mathcal{C}(P),\ \forall i \in I,\ t_i \in e_i$$

The following example shows an $PES_{\equiv}$ P with $ter \circ col(P) \neq P$ (up to isomorphism), because P does not respect the [Shortcut] property, and an $PES_{\equiv}$ Q which is a completion of P with the [Shortcut] property. We have $ter \circ col(Q) = Q$ (up to isomorphism).

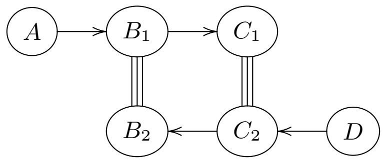


This example give the meaning of the [Shortcut] property, which look like a forward symmetry property :
If an event (D_2) need an other event (C_2), then there is a copy of the first event (D_1) for each event equivalent to the second event (C_1).

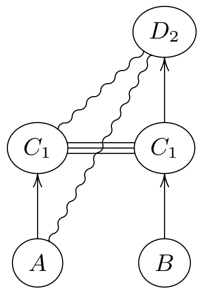
This is a simplification of the real [Shortcut] property. You can find multiple exceptions to this simple rule :



This event structure is stable by $ter \circ col$, but the event C_1 is never needed for enable an event equivalent to D_2. The reason is that something lesser than $[C_1]$, here $\{A\}$, can enable an event equivalent to D_2. We have another example of this case :



This event structure is stable by $ter \circ col$. Here, there is no event equivalent to C_1 which need B_2, because B_2 need an event equivalent to C_1 to be enabled.



This event structure is stable by $ter \circ col$, but the event C_1 is never needed for enable an event equivalent to D_2. The reason is that $\{C_1, D_2\}$ is inconsistent.

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*