Trellis nets

A compact representation for runs of concurrent systems

Eric Fabre

DistribCom group

Irisa/Inria - Rennes - France

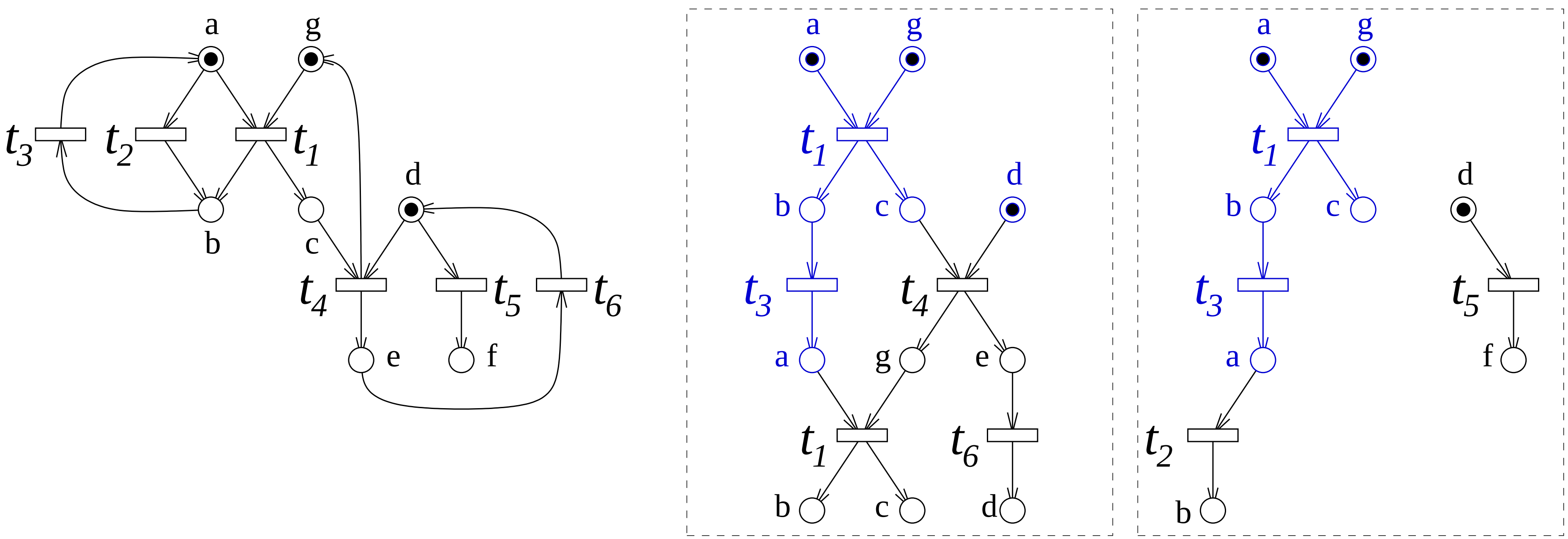
Outline

1. Recall : factorization of unfoldings
2. How to derive this result ?
3. Its use for distributed monitoring... and its limitations
4. Trellis nets, and their properties
5. Relations between nets, unfoldings, trellis nets

1 - Recall : factorization of unfoldings

Safe nets - Configurations :

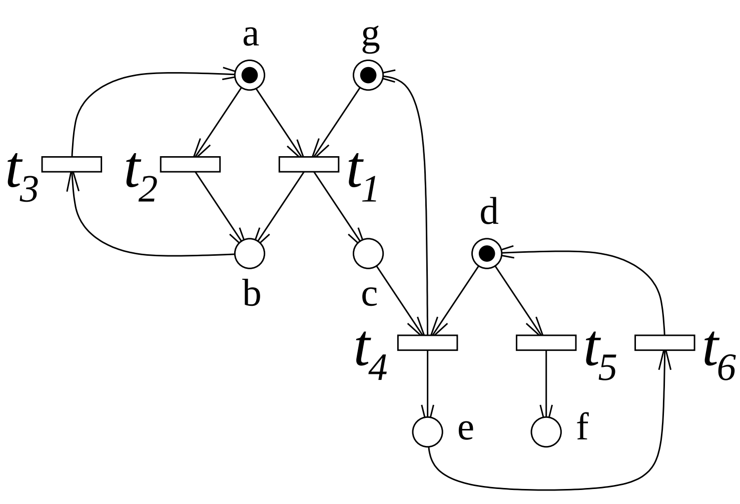
- Net : $\mathcal{N} = (P, T, \rightarrow, P^0, \lambda, \Lambda)$,
- Configuration κ : run of a safe net.

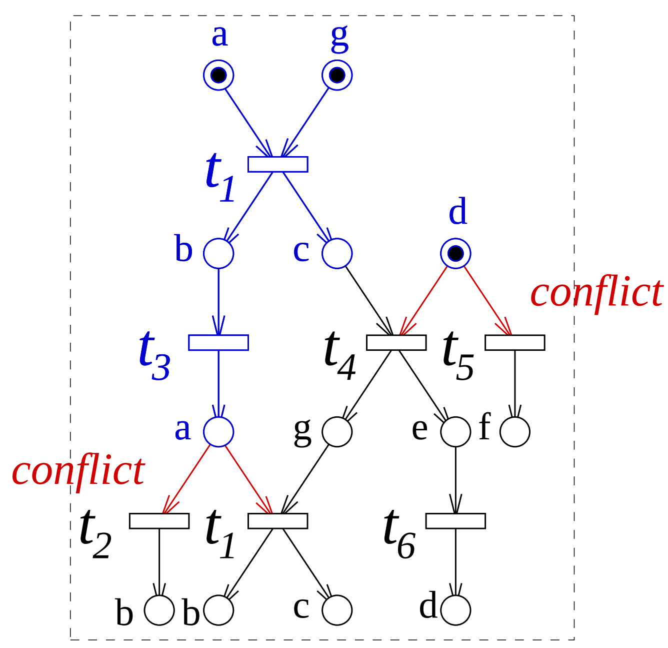


1 - Recall : factorization of unfoldings

Safe nets - Configurations :

- Net : $\mathcal{N} = (P, T, \rightarrow, P^0, \lambda, \Lambda)$,
- Occurrence net : a set of configurations.





Occurrence net : $\mathcal{O} = (C, E, \rightarrow, C^0, \lambda, \Lambda)$ is an ON iff

1. $\rightarrow^*$ is a well founded partial order,
2. C^0 = minimal nodes of $\rightarrow^*$,
3. forall $c \in C, \ \ |^\bullet c| \leq 1$: a single cause to every condition,
4. no node is in self-conflict.

Branching process : $\mathcal{O}$ is a BP of $\mathcal{N}$ iff

1. there exists a morphism (folding) $f : \mathcal{O} \rightarrow \mathcal{N}$, preserving labels,
2. parsimony : $\forall e, e' \in E, \ \ [^\bullet e = {}^\bullet e', \ f(e) = f(e')] \ \ \Rightarrow \ \ e = e'$

Unfolding of $\mathcal{N}$: $\mathcal{U}_\mathcal{N}$ is the maximal branching process of $\mathcal{N}$.

Product of nets : $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2$ is based on labels

1. places remain private : $P = P_1 \uplus \mathcal{P}_2$, disjoint union,
2. transitions labeled by $\Lambda_1 \cap \Lambda_2$ must synchronize

$$T_s \quad = \quad \{(t_1, t_2) : \lambda_1(t_1) = \lambda_2(t_2) \in \Lambda_1 \cap \Lambda_2\}$$

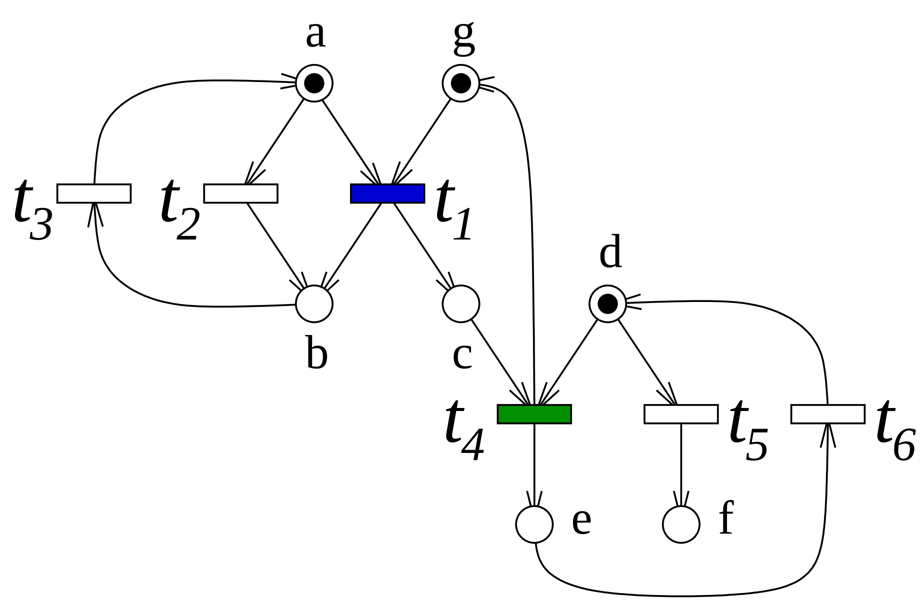
3. the other transitions remain private

$$\begin{aligned} T_p \quad = \quad & \{(t_1, *) : \lambda_1(t_1) \in \Lambda_1 \setminus \Lambda_2\} \\ & \cup \ \{(*, t_2) : \lambda_2(t_2) \in \Lambda_2 \setminus \Lambda_1\} \end{aligned}$$

4. $T = T_p \cup T_s$, and the flow follows naturally.

Product of nets : $\times_{nets}$

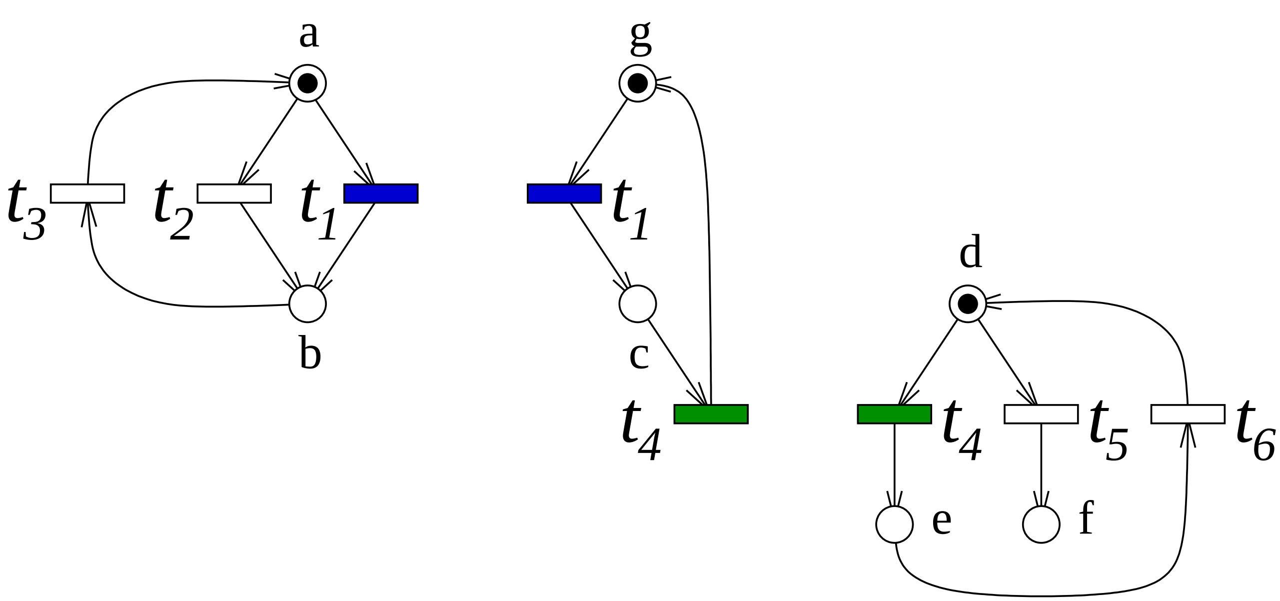
- Example :



- In general, the number of transitions is *larger* in the product.

Product of nets : $\times_{nets}$

- Example :



- In general, the number of transitions is *larger* in the product.

Product of (labeled) occurrence nets : $\mathcal{O} = \mathcal{O}_1 \times_{occ} \mathcal{O}_2$

- There exists such a product (we don't detail the definition).
- Again, an ON resulting of a product is generally more complex than its factors (the number of events and conditions increases).

Theorem : If $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$, then

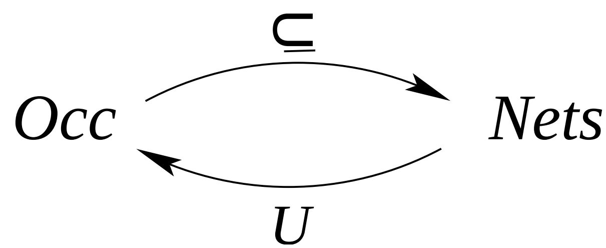
$$\mathcal{U}_{\mathcal{N}} = \mathcal{U}_{\mathcal{N}_1} \times_{occ} \mathcal{U}_{\mathcal{N}_2} \times_{occ} \ldots \times_{occ} \mathcal{U}_{\mathcal{N}_n}$$

- Interest : provides a more *compact* description for runs of a compound system.

2 - How to derive this result ?

[Winskel, 1984]

Adjunction :



- between categories Occ and $Nets$
- two functors, working in opposite directions

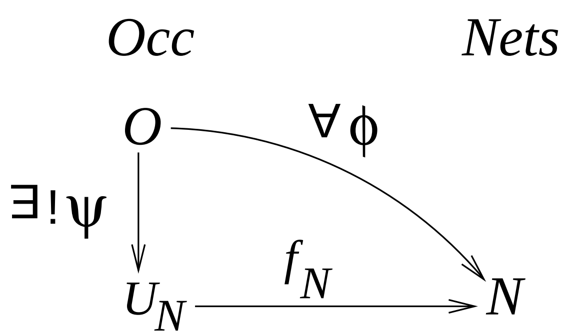
$$\subseteq : \ Occ \longrightarrow Nets$$

$$\mathcal{U} \ : \ Nets \rightarrow Occ$$

Construction : based on three ingredients

- (H1) The unfolding operation is a functor $\mathcal{U} : Nets \rightarrow Occ$
- (H2) Universal property of unfoldings :

$$\forall \phi : \mathcal{O} \rightarrow \mathcal{N}, \;\; \exists ! \psi : \mathcal{O} \rightarrow \mathcal{U}(\mathcal{N}), \;\; \phi = f_{\mathcal{N}} \circ \psi$$



- (H3) $\mathcal{U}$ is invariant on occurrence nets : $\forall \mathcal{O} \in Occ, \;\; \mathcal{U}(\mathcal{O}) \cong \mathcal{O}$

then...

- Right adjoints preserve limits, in particular products :

$$\mathcal{U}(\mathcal{N}_1 \times_{nets} \mathcal{N}_2) \quad = \quad \mathcal{U}(\mathcal{N}_1) \times_{occ} \mathcal{U}(\mathcal{N}_2)$$

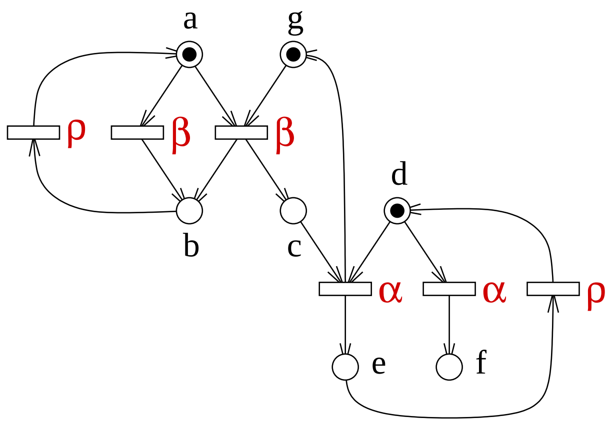
- By (H3), the product $\times_{occ}$ can be defined by

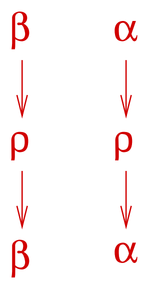
$$\mathcal{O}_1 \times_{occ} \mathcal{O}_2 \quad \cong \quad \mathcal{U}(\mathcal{O}_1) \times_{occ} \mathcal{U}(\mathcal{O}_2) \; = \; \mathcal{U}(\mathcal{O}_1 \times_{nets} \mathcal{O}_2)$$

3 - Unfolding factorization and distributed diagnosis

Centralized diagnosis :

- Single system ; a partial order of labels produced by the system
- Recover runs explaining observations

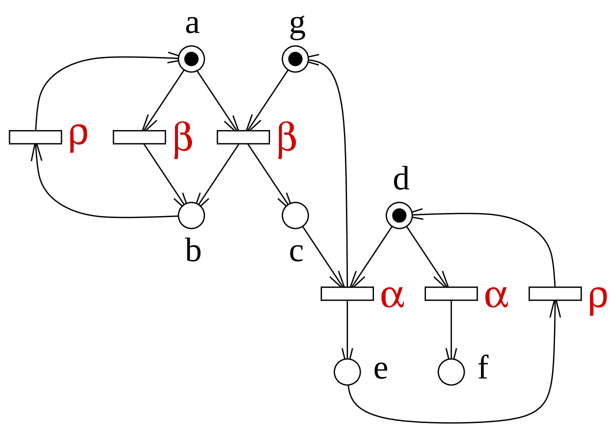


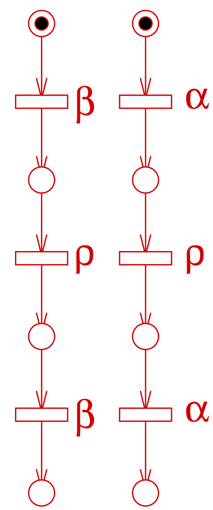


3 - Unfolding factorization and distributed diagnosis

Centralized diagnosis :

- Single system ; a partial order of labels produced by the system
- Recover runs explaining observations

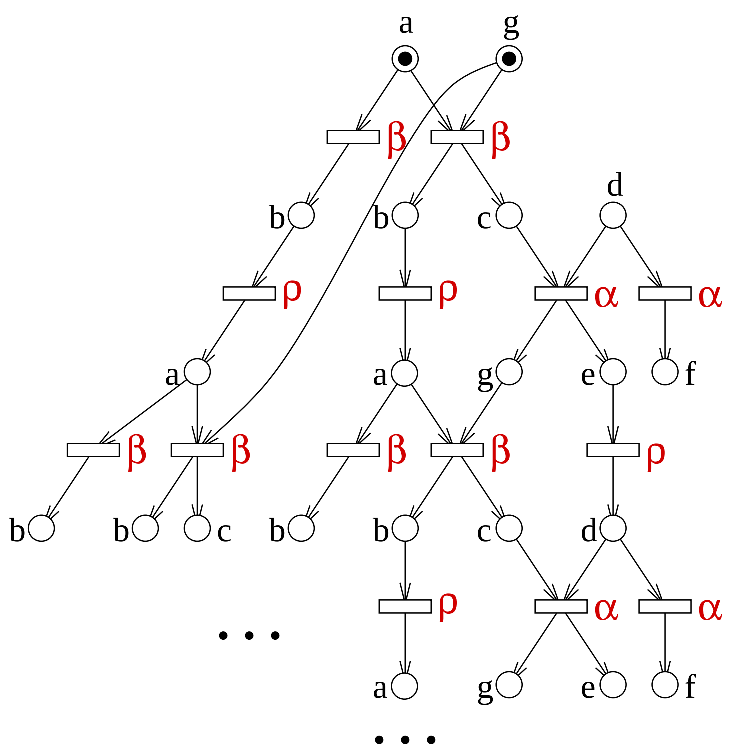


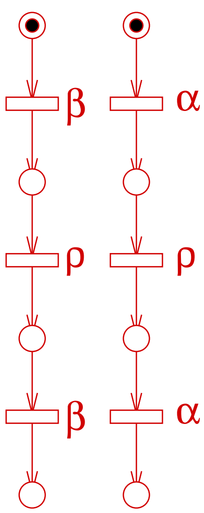


3 - Unfolding factorization and distributed diagnosis

Centralized diagnosis :

- Single system ; a partial order of labels produced by the system
- Recover runs explaining observations

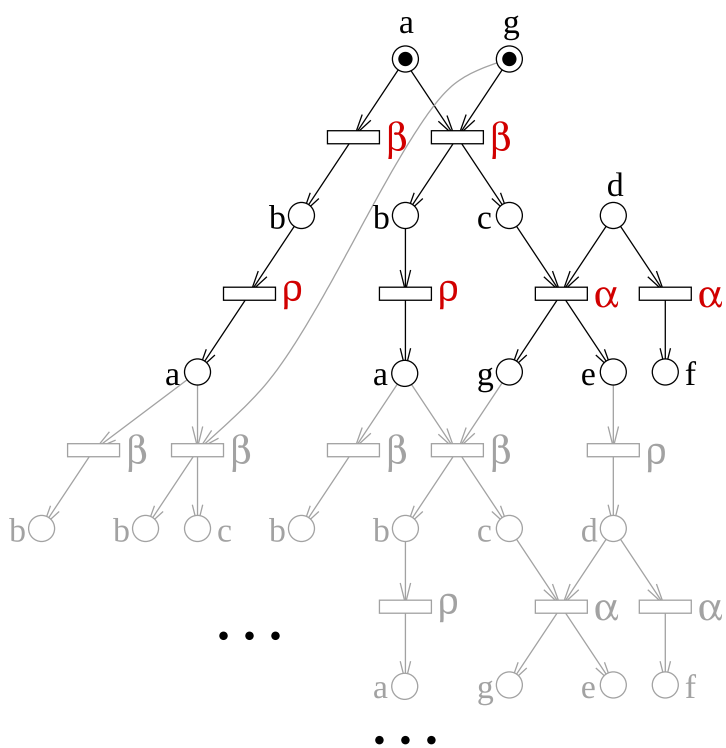


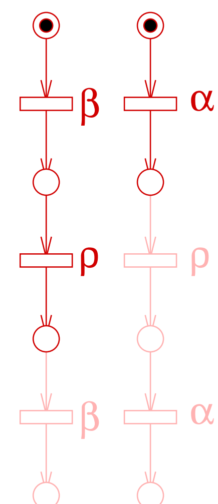


3 - Unfolding factorization and distributed diagnosis

Centralized diagnosis :

- Single system ; a partial order of labels produced by the system
- Recover runs explaining observations

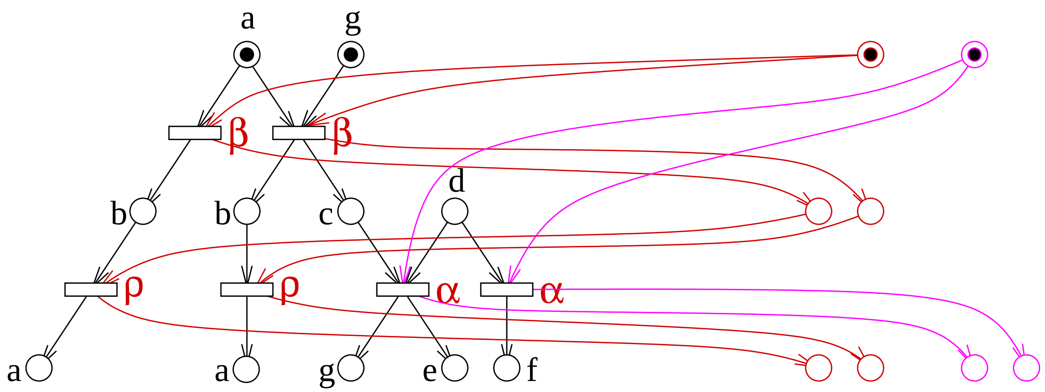




3 - Unfolding factorization and distributed diagnosis

Centralized diagnosis :

- Single system ; a partial order of labels produced by the system
- Recover runs explaining observations



Distributed diagnosis :

- Distributed system : $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$
- Distributed observations : $\mathcal{A} = \mathcal{A}_1 \times_{occ} \mathcal{A}_2 \times_{occ} \ldots \times_{occ} \mathcal{N}_n$

$$U_N$$

$$\times_{occ}$$

$$A$$

Distributed diagnosis :

- Distributed system : $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$
- Distributed observations : $\mathcal{A} = \mathcal{A}_1 \times_{occ} \mathcal{A}_2 \times_{occ} \ldots \times_{occ} \mathcal{N}_n$

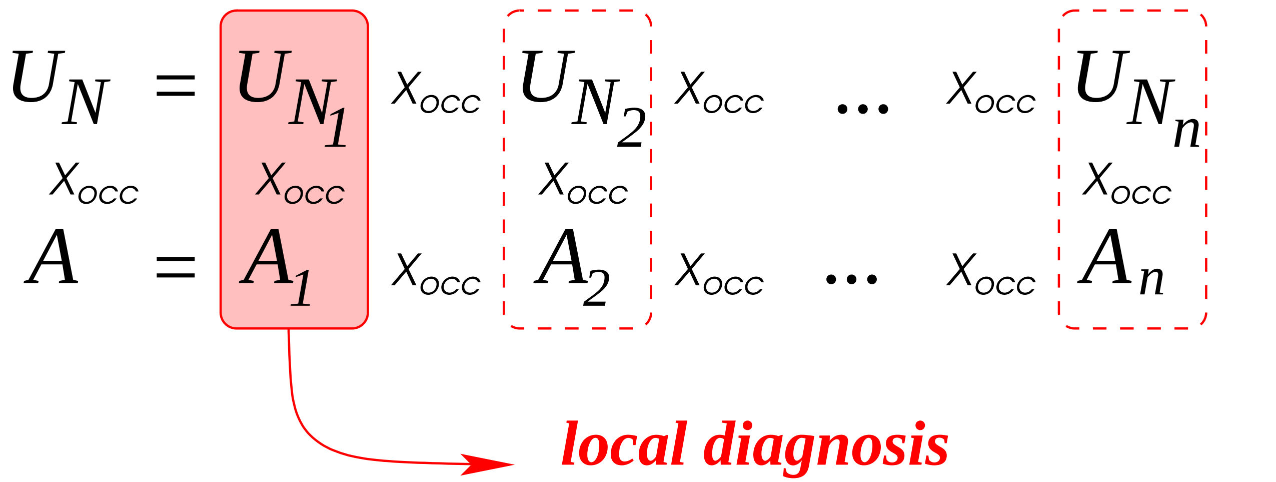
$$U_N = U_{N_1} \times_{occ} U_{N_2} \times_{occ} \ldots \times_{occ} U_{N_n}$$

$$\times_{occ}$$

$$A = A_1 \times_{occ} A_2 \times_{occ} \ldots \times_{occ} A_n$$

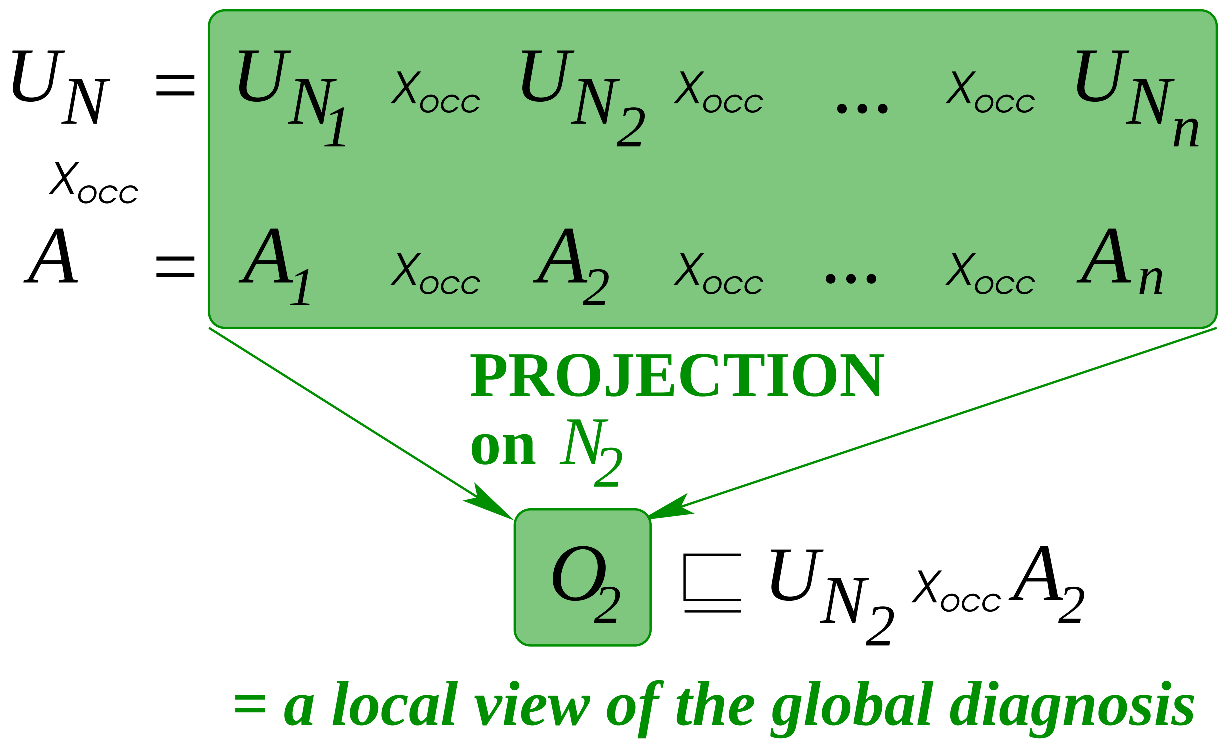
Distributed diagnosis :

- Distributed system : $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$
- Distributed observations : $\mathcal{A} = \mathcal{A}_1 \times_{occ} \mathcal{A}_2 \times_{occ} \ldots \times_{occ} \mathcal{N}_n$



Distributed diagnosis :

- Distributed system : $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$
- Distributed observations : $\mathcal{A} = \mathcal{A}_1 \times_{occ} \mathcal{A}_2 \times_{occ} \ldots \times_{occ} \mathcal{N}_n$



Distributed diagnosis :

- Distributed system : $\mathcal{N} = \mathcal{N}_1 \times_{nets} \mathcal{N}_2 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$
- Distributed observations : $\mathcal{A} = \mathcal{A}_1 \times_{occ} \mathcal{A}_2 \times_{occ} \ldots \times_{occ} \mathcal{N}_n$

$$U_N = U_{N_1} \times_{occ} U_{N_2} \times_{occ} \ldots \times_{occ} U_{N_n}$$

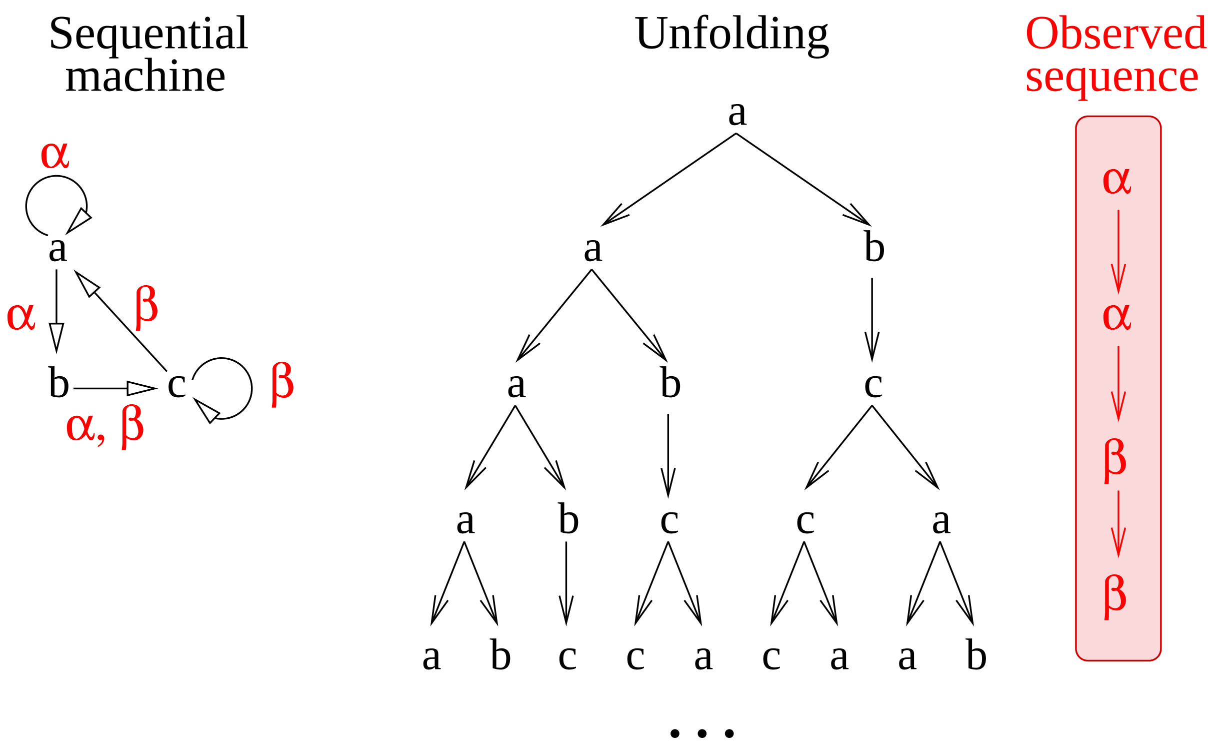
$$\times_{occ}$$

$$A = A_1 \times_{occ} A_2 \times_{occ} \ldots \times_{occ} A_n$$

$$U_N \times_{occ} A = O_1 \times_{occ} O_2 \times_{occ} \ldots \times_{occ} O_n$$

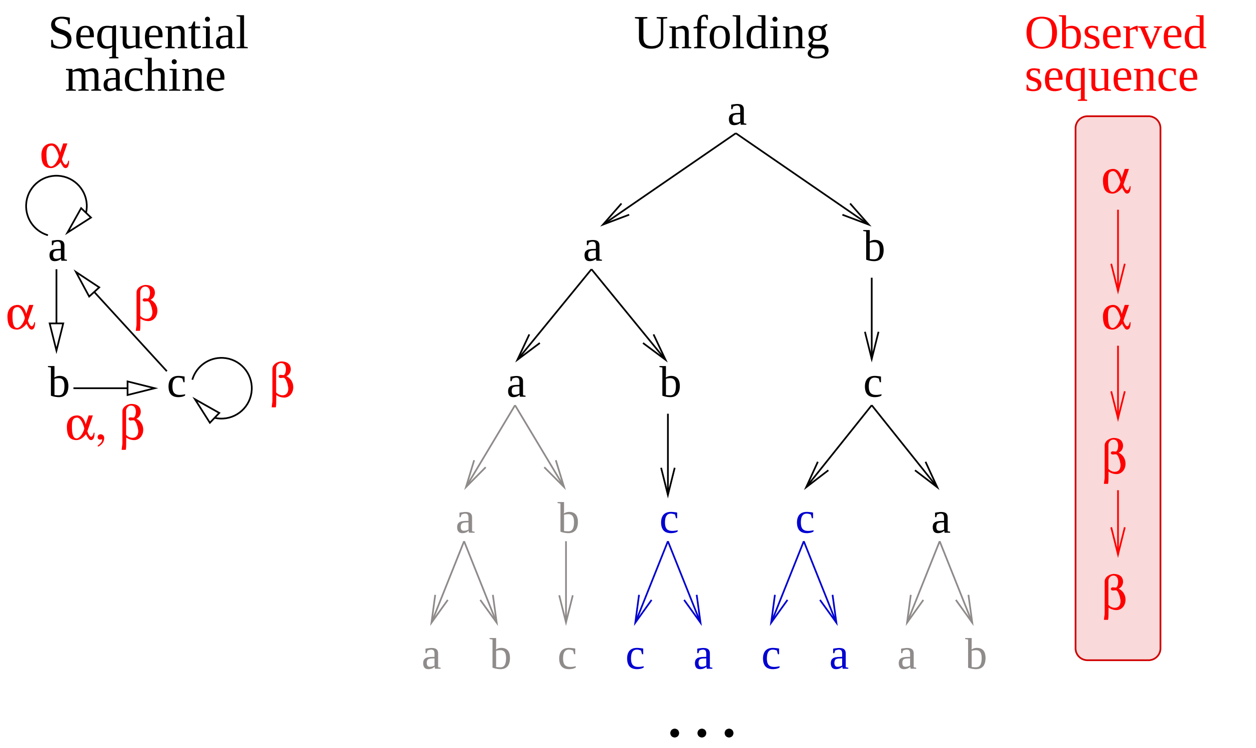
Limitation of unfoldings/branching processes :

- In the simple case of a sequential machine : the size of the unfolding explodes with the length of trajectories.
- (Max likelihood) diagnosis algorithms rather use a trellis, for example dynamic programming.



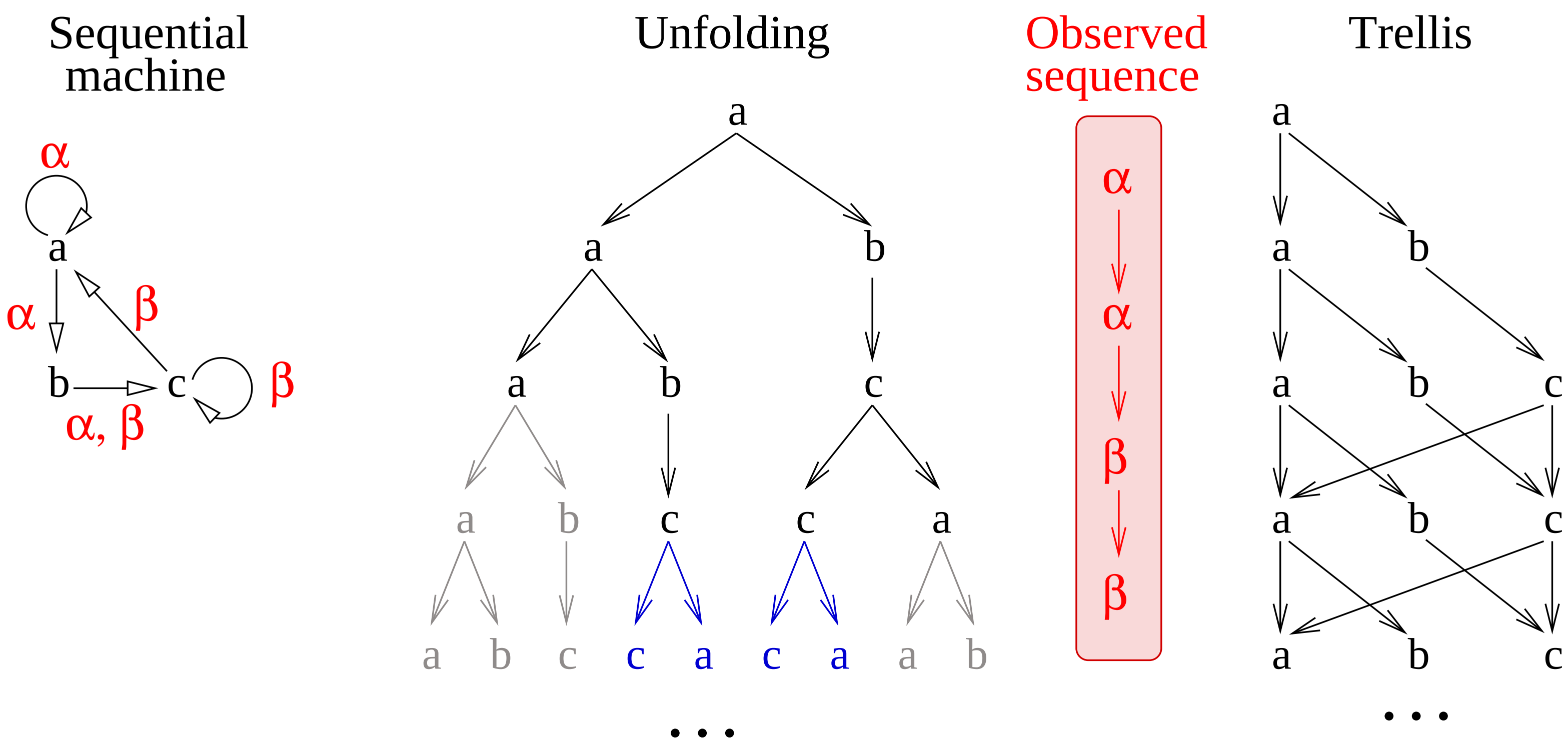
Limitation of unfoldings/branching processes :

- In the simple case of a sequential machine : the size of the unfolding explodes with the length of trajectories.
- (Max likelihood) diagnosis algorithms rather use a trellis, for example dynamic programming.



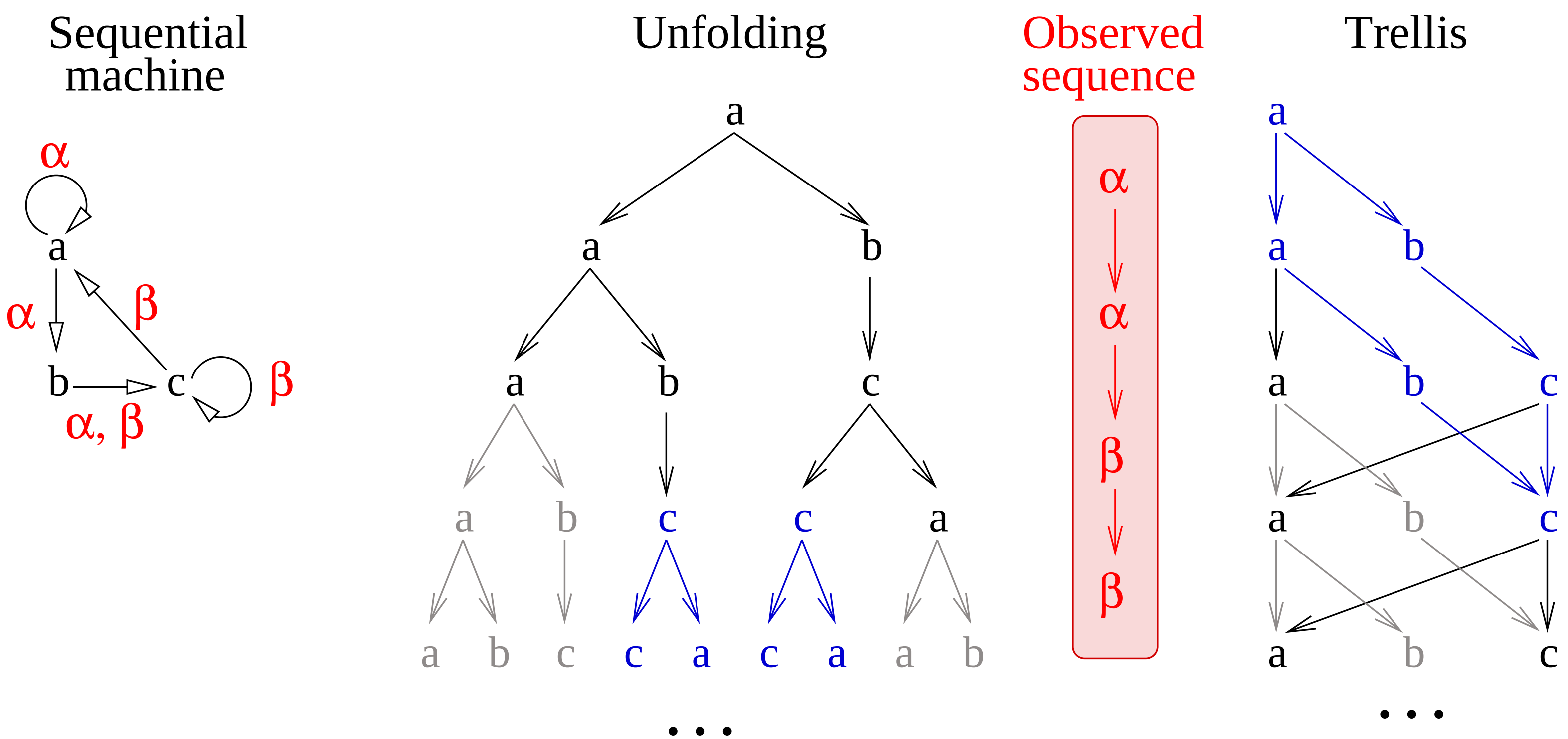
Limitation of unfoldings/branching processes :

- In the simple case of a sequential machine : the size of the unfolding explodes with the length of trajectories.
- (Max likelihood) diagnosis algorithms rather use a trellis, for example dynamic programming.



Limitation of unfoldings/branching processes :

- In the simple case of a sequential machine : the size of the unfolding explodes with the length of trajectories.
- (Max likelihood) diagnosis algorithms rather use a trellis, for example dynamic programming.



Questions :

- Can we adapt the notion of trellis to concurrent systems ?
- Is there a factorization property ?

  (this is necessary for distributed monitoring algorithms)
- What are the relations between nets, trellisses, unfoldings ?

Surprisingly, there exist simple answers to these questions !

4 - Trellis nets, and their properties

Occurrence net : $\mathcal{O} = (C, E, \rightarrow, C^0, \lambda, \Lambda)$

1. $\rightarrow^*$ is a well founded partial order,
2. C^0 = minimal nodes of $\rightarrow^*$,
3. forall $c \in C, \;\; |{}^\bullet c| \leq 1$: a single cause to every condition,
4. no node is in self-conflict.

4 - Trellis nets, and their properties

Pre-trellis net : $\mathcal{T} = (C, E, \rightarrow, C^0, \lambda, \Lambda)$

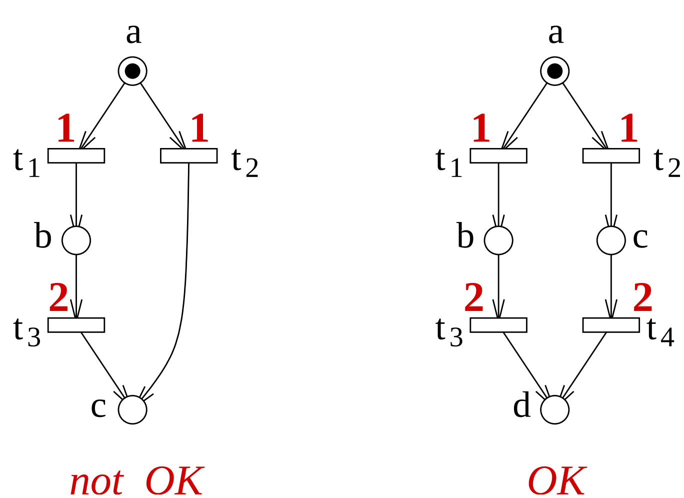
1. $\rightarrow^*$ is a well founded partial order,
2. C^0 = minimal nodes of $\rightarrow^*$,

4 - Trellis nets, and their properties

Pre-trellis net : $\mathcal{T} = (C, E, \to, C^0, \lambda, \Lambda)$

1. $\to^*$ is a well founded partial order,
2. C^0 = minimal nodes of $\to^*$,
3. $\forall c \in C, \;\; \forall e, e' \in {}^\bullet c, \;\; H(e) = H(e')$ where H is a *height function*, for example :

$$H(e) = \max\{N \;: \exists e_1, e_2, \ldots, e_N \in E, \; e_1 \to^* e_2 \to^* \ldots \to^* e_N = e\}$$

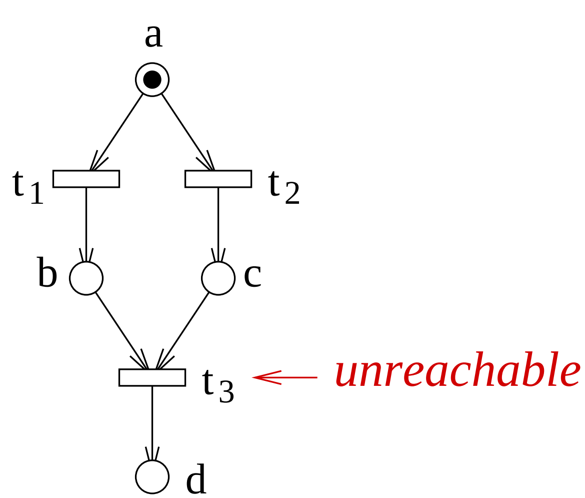


4 - Trellis nets, and their properties

Pre-trellis net : $\mathcal{T} = (C, E, \rightarrow, C^0, \lambda, \Lambda)$

1. $\rightarrow^*$ is a well founded partial order,
2. C^0 = minimal nodes of $\rightarrow^*$,
3. $\forall c \in C, \;\; \forall e, e' \in {}^\bullet c, \;\; H(e) = H(e')$ where H is a *height function*, for example :

$$H(e) = \max\{N \;:\exists e_1, e_2, \ldots, e_N \in E,\; e_1 \rightarrow^* e_2 \rightarrow^* \ldots \rightarrow^* e_N = e\}$$



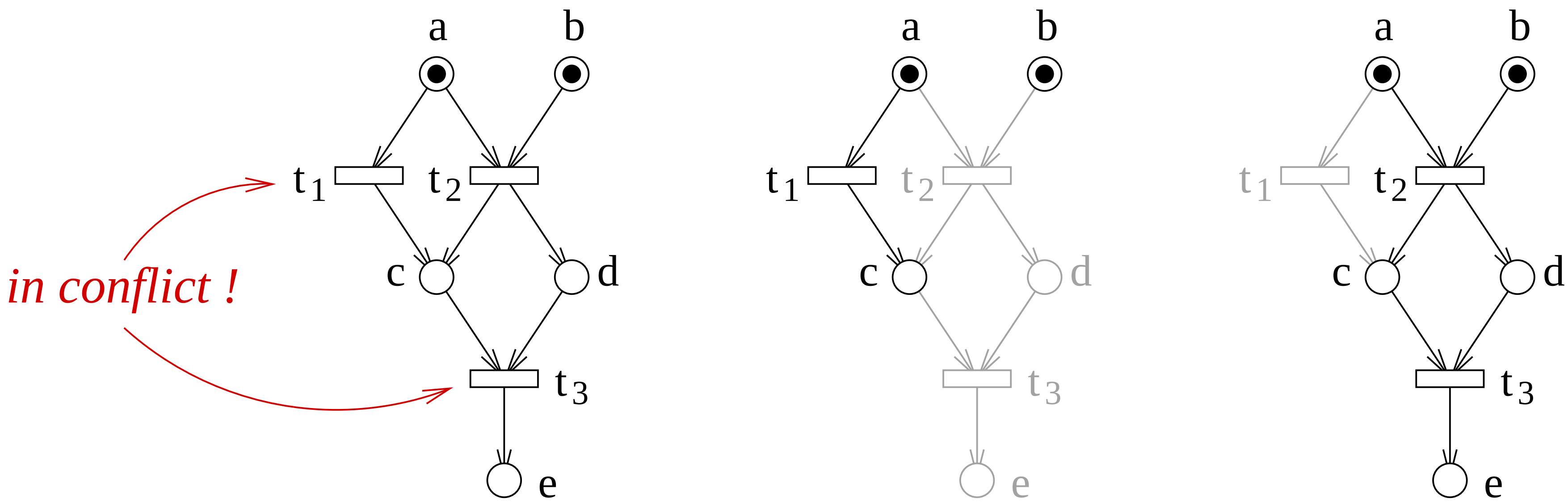
Configuration : it's a sub-net κ of $\mathcal{T} = (C, E, \rightarrow, C^0, \lambda, \Lambda)$ satisfying

1. $C^0 \subseteq \kappa$: it contains all initial conditions of $\mathcal{T}$,
2. $\forall e \in E \cap \kappa, \ \ {}^\bullet e \subseteq \kappa$ and $e^\bullet \subseteq \kappa$: each event comes with all its causes and consequences,
3. $\forall c \in C \cap \kappa, |{}^\bullet c|_\kappa = 1$ or $c \in C^0$: each condition is either minimal or has one of its possible causes,
4. $\forall c \in C \cap \kappa, |c^\bullet|_\kappa \leq 1$: each condition triggers at most one event.

- To read out configurations, one must solve conflicts in both directions of time.

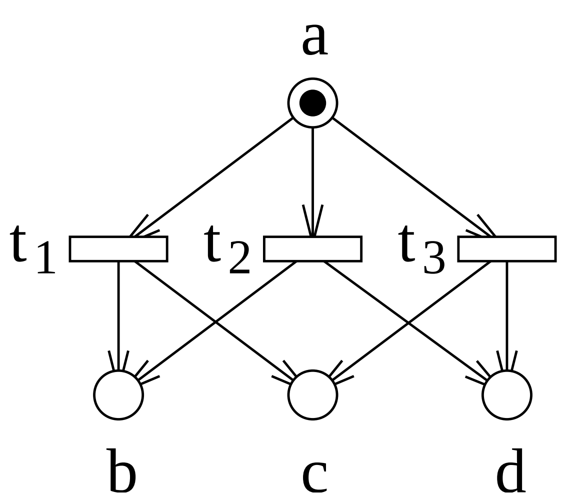
Trellis net : a pre-trellis net $\mathcal{T}$ is a trellis net iff each event is reachable, *i.e.* belongs at least to one configuration.

A simple example



Concurrency and conflict :

- Not easy to define graphically ! So we use indirect definitions.
- Two nodes are in *conflict* iff they never appear in the same configuration.
- Two nodes are *concurrent* iff there *exists* a configuration κ where they are concurrent.
- The conflict is not binary...



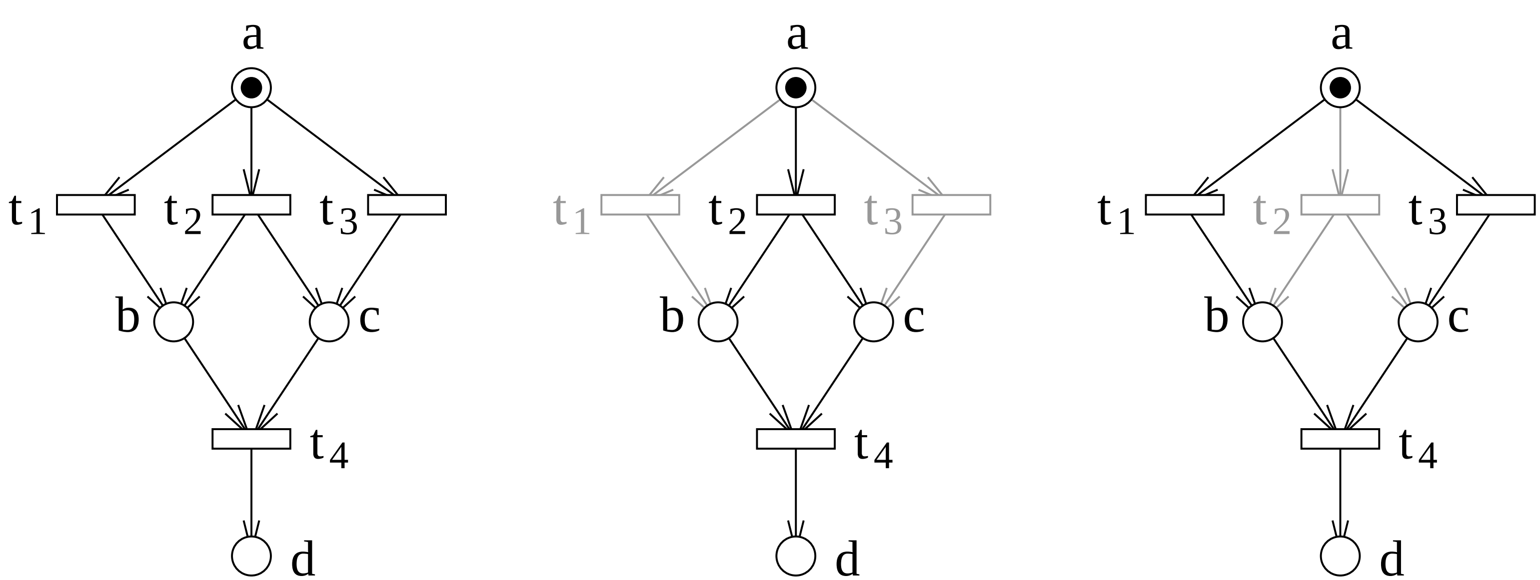
Prefix of a TN : $\mathcal{T}' \sqsubseteq \mathcal{T}$ iff

1. $\mathcal{T}'$ is a sub-net of $\mathcal{T}$,

2. $\min \mathcal{T}' = \min \mathcal{T}$: same initial conditions,

3. $\forall e \in E, \;\; e \in E' \;\; \Rightarrow \;\; [{}^\bullet e \subseteq \mathcal{T}'$ and $e^\bullet \subseteq \mathcal{T}']$: events come with all their neighbourhood,

4. $\mathcal{T}'$ is a trellis net.



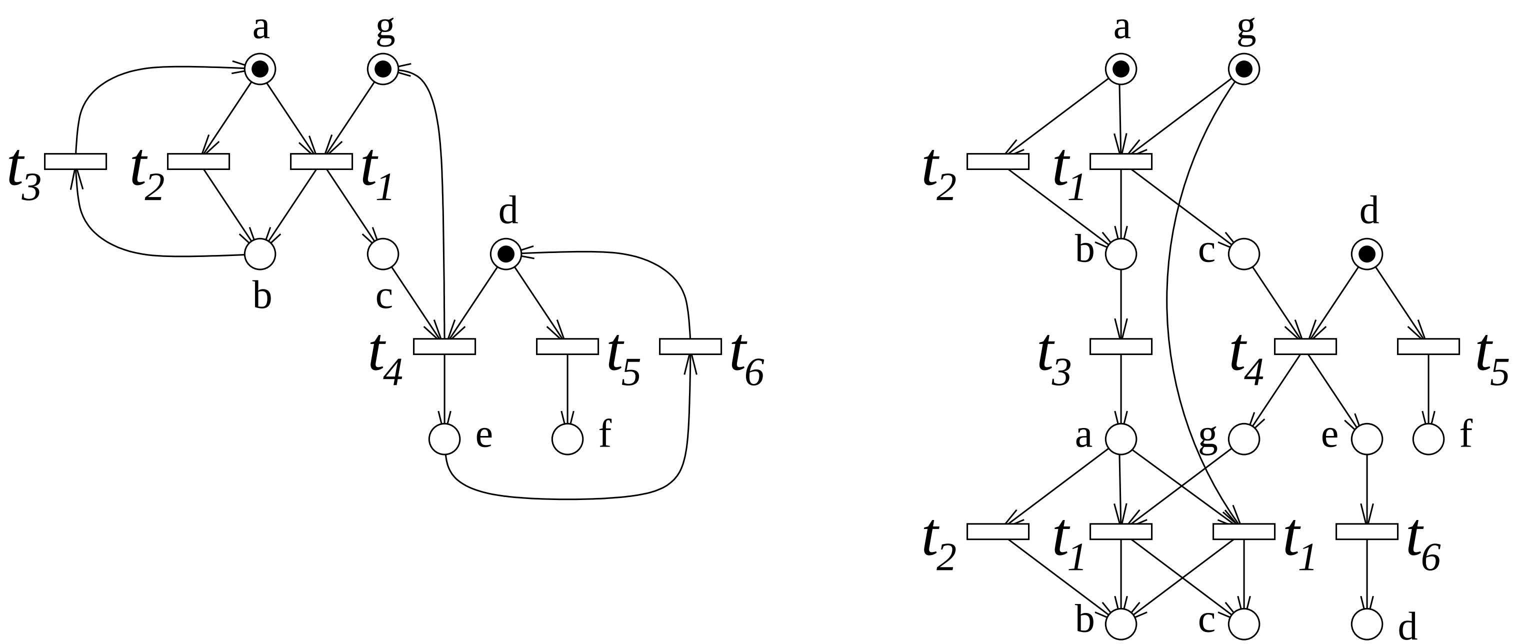
a prefix

not a prefix !

Trellis process : $\mathcal{T}$ is a TP of a net $\mathcal{N}$ iff

1. there exists a morphism (folding) $f^t : \mathcal{T} \to \mathcal{N}$, preserving labels,
2. parsimony 1 : $\forall e, e' \in E, \ \ [{}^\bullet e = {}^\bullet e', \ f(e) = f(e')] \ \Rightarrow \ e = e'$,
3. parsimony 2 : $\forall c, c' \in C, \ [H(c) = H(c'), \ f(c) = f(c')] \ \Rightarrow \ c = c'$

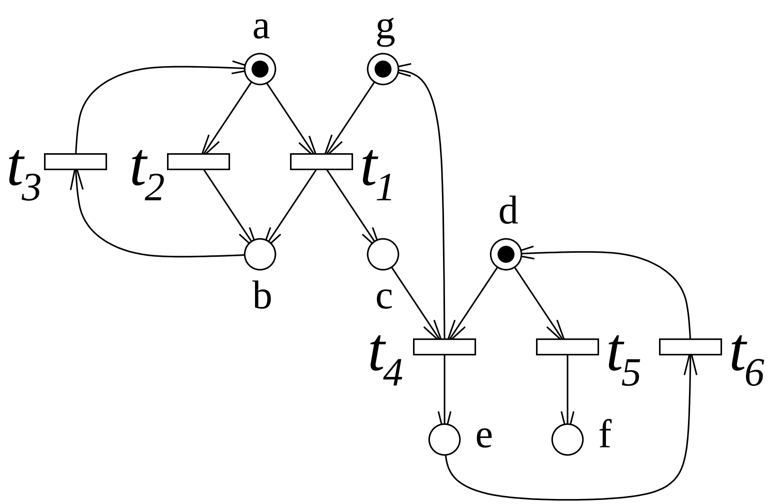
Time-unfolding (or trellis) of $\mathcal{N}$: $\mathcal{U}^t_{\mathcal{N}}$ is the maximal trellis process of $\mathcal{N}$.

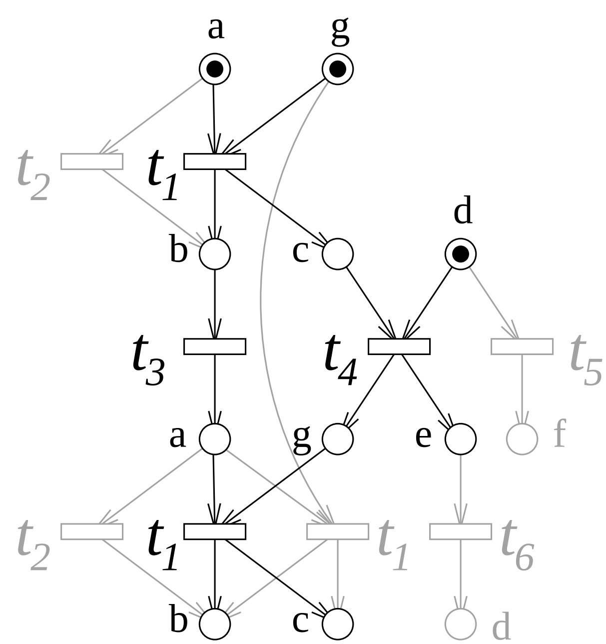


Trellis process : $\mathcal{T}$ is a TP of a net $\mathcal{N}$ iff

1. there exists a morphism (folding) $f^t : \mathcal{T} \to \mathcal{N}$, preserving labels,
2. parsimony 1 : $\forall e, e' \in E, \ \ [{}^\bullet e = {}^\bullet e', \ f(e) = f(e')] \ \Rightarrow \ e = e'$,
3. parsimony 2 : $\forall c, c' \in C, \ [H(c) = H(c'), \ f(c) = f(c')] \ \Rightarrow \ c = c'$

Time-unfolding (or trellis) of $\mathcal{N}$: $\mathcal{U}^t_{\mathcal{N}}$ is the maximal trellis process of $\mathcal{N}$.

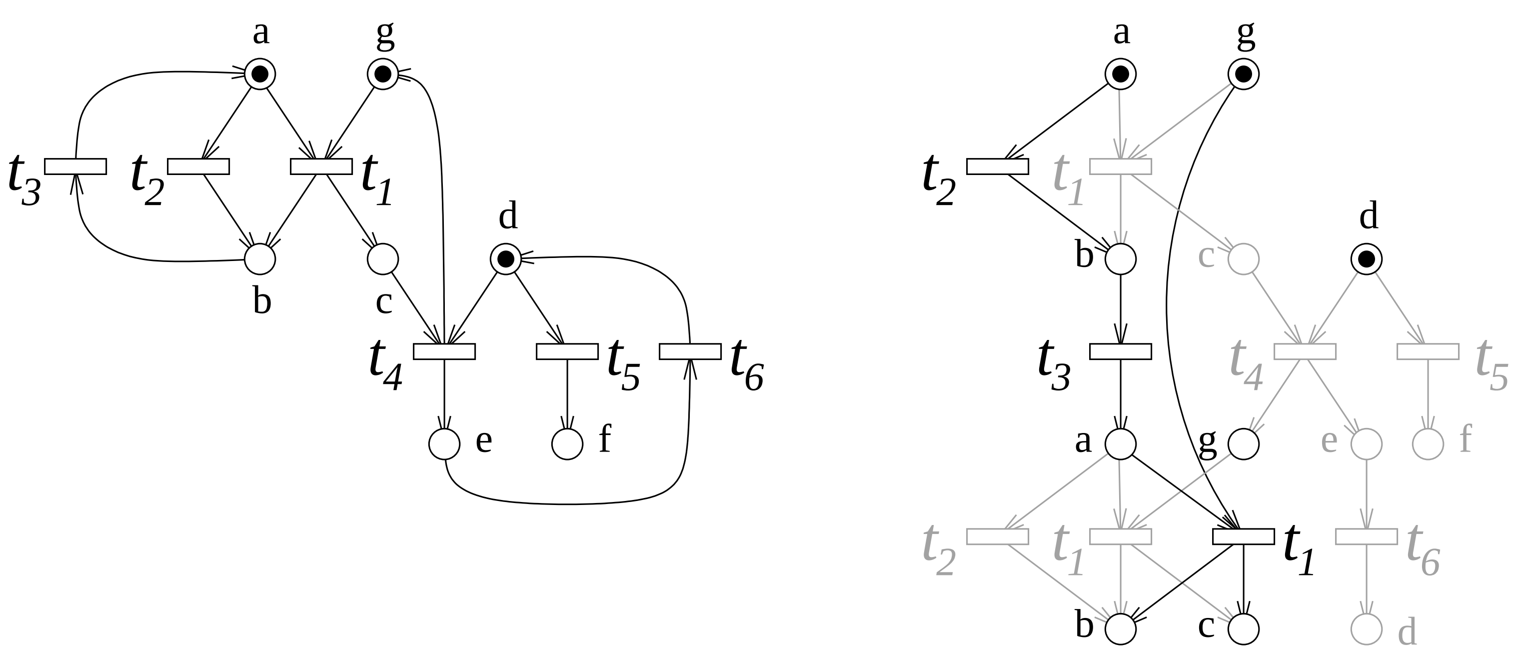




Trellis process : $\mathcal{T}$ is a TP of a net $\mathcal{N}$ iff

1. there exists a morphism (folding) $f^t : \mathcal{T} \rightarrow \mathcal{N}$, preserving labels,
2. parsimony 1 : $\forall e, e' \in E,\ \ [{}^\bullet e = {}^\bullet e',\ f(e) = f(e')] \ \Rightarrow\ e = e'$,
3. parsimony 2 : $\forall c, c' \in C,\ [H(c) = H(c'),\ f(c) = f(c')] \ \Rightarrow\ c = c'$

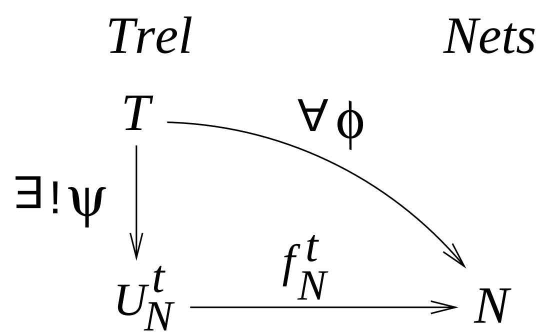
Time-unfolding (or trellis) of $\mathcal{N}$: $\mathcal{U}^t_{\mathcal{N}}$ is the maximal trellis process of $\mathcal{N}$.



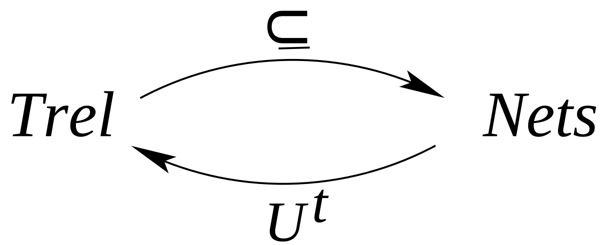
Adjunction : between $Trel$ and $Nets$

- (H1) The time-unfolding operation is a functor $\mathcal{U}^t : Nets \rightarrow Trel$
- (H2) Universal property of trellisses/time-unfoldings :

$$\forall \phi : \mathcal{T} \rightarrow \mathcal{N}, \;\; \exists ! \psi : \mathcal{T} \rightarrow \mathcal{U}^t(\mathcal{N}), \;\; \phi = f^t_{\mathcal{N}} \circ \psi$$



- (H3) $\mathcal{U}^t$ is invariant on trellis nets : $\forall \mathcal{T} \in Trel, \;\; \mathcal{U}^t(\mathcal{T}) \cong \mathcal{T}$



then...

- Right adjoints preserve limits, in particular products :

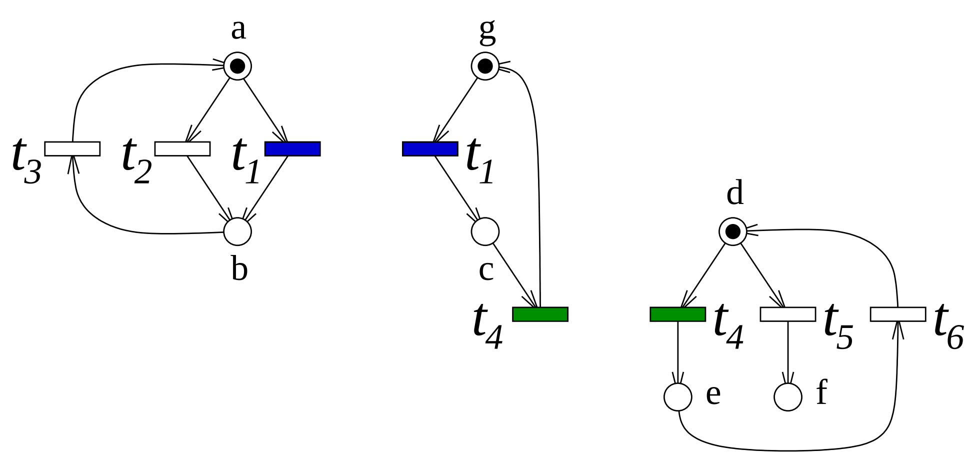
$$\mathcal{U}^t(\mathcal{N}_1 \times_{nets} \mathcal{N}_2) \;\;=\;\; \mathcal{U}^t(\mathcal{N}_1) \times_{trel} \mathcal{U}^t(\mathcal{N}_2)$$

- By (H3), the product $\times_{trel}$ can be defined by

$$\mathcal{T}_1 \times_{trel} \mathcal{T}_2 \;\;\cong\;\; \mathcal{U}^t(\mathcal{T}_1) \times_{trel} \mathcal{U}^t(\mathcal{T}_2) \;=\; \mathcal{U}^t(\mathcal{T}_1 \times_{nets} \mathcal{T}_2)$$

Remark :

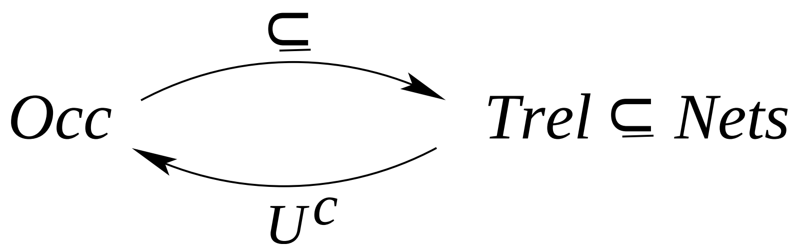
- A safe net $\mathcal{N}$ can be expressed as a product of sequential machines $\mathcal{N}_1 \times_{nets} \ldots \times_{nets} \mathcal{N}_n$



- For each component $\mathcal{N}_i$, the time-unfolding coincides with the usual notion of trellis for an automaton, thanks to the height constraint.

5 - Relations between nets, unfoldings, trellis nets

- $(\subseteq, \mathcal{U})$ defines an adjunction between Occ and $Nets \supseteq Trel$. Its restriction to $Trel$ induces another adjunction



- On trellis nets, functor $\mathcal{U}$ unfolds only *conflicts* (time is already unfolded) : we denote it by $\mathcal{U}^c$.
- We have :

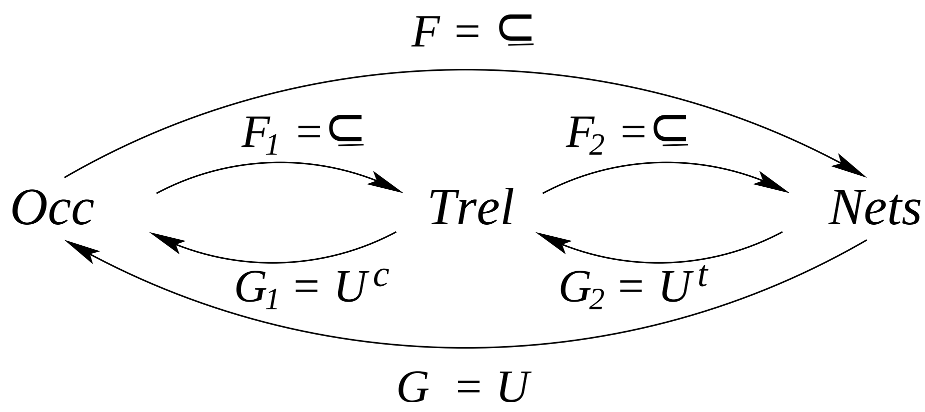
$$\mathcal{U}^c(\mathcal{T}_1 \times_{trel} \mathcal{T}_2) \;\; = \;\; \mathcal{U}^c(\mathcal{T}_1) \times_{occ} \mathcal{U}^c(\mathcal{T}_2)$$

- and we can redefine $\times_{occ}$ by

$$\mathcal{O}_1 \times_{occ} \mathcal{O}_2 \;\; \cong \;\; \mathcal{U}^c(\mathcal{O}_1) \times_{occ} \mathcal{U}^c(\mathcal{O}_2) \; = \; \mathcal{U}^c(\mathcal{O}_1 \times_{trel} \mathcal{O}_2)$$

Gathering results

- Three nested categories $Occ \subset Trel \subset Nets$, three adjunctions

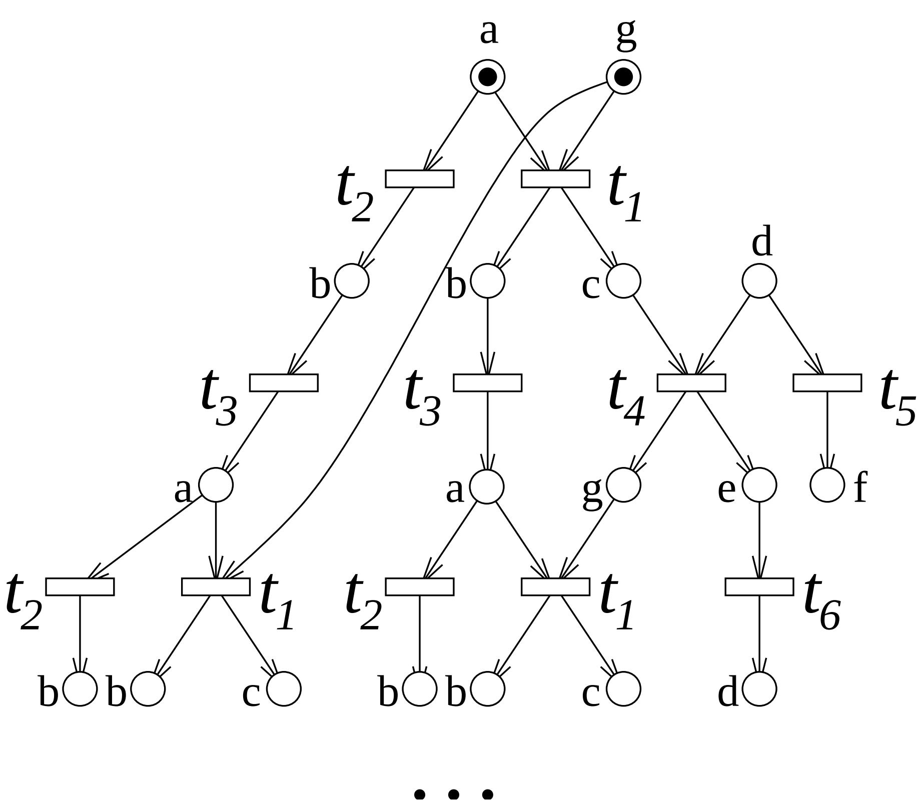


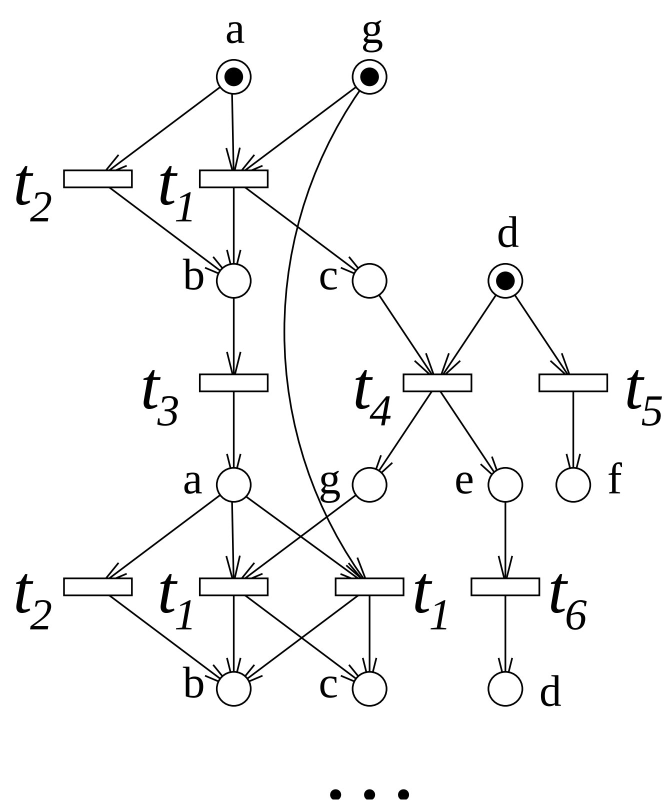
- Adjunctions can be composed : functors $\mathcal{U}$ and $\mathcal{U}^c \circ \mathcal{U}^t$ are naturally equivalent

$$\mathcal{U}(\mathcal{N}) \quad \cong \quad \mathcal{U}^c \circ \mathcal{U}^t(\mathcal{N})$$

- The trellis of $\mathcal{N}$ is obtained by a conflict-folding on $\mathcal{U}_{\mathcal{N}}$.
- The trellis and the unfolding of $\mathcal{N}$ describe the same sets of configurations.

Comparison :


a
g
t2
t1
b
b
c
d
t3
t3
t4
t5
a
a
g
e
f
t2
t1
t2
t1
t6
b
b
c
b
b
c
d
. . .


a
g
t2
t1
b
c
d
t3
t4
t5
a
g
e
f
t2
t1
t1
t6
b
c
d
. . .

Conclusion

The + and the - of trellis nets :

+ trellis processes remain (more) compact in time,
- configurations are less easy to read,
+ factorization property : small components may be tractable.

Future work :

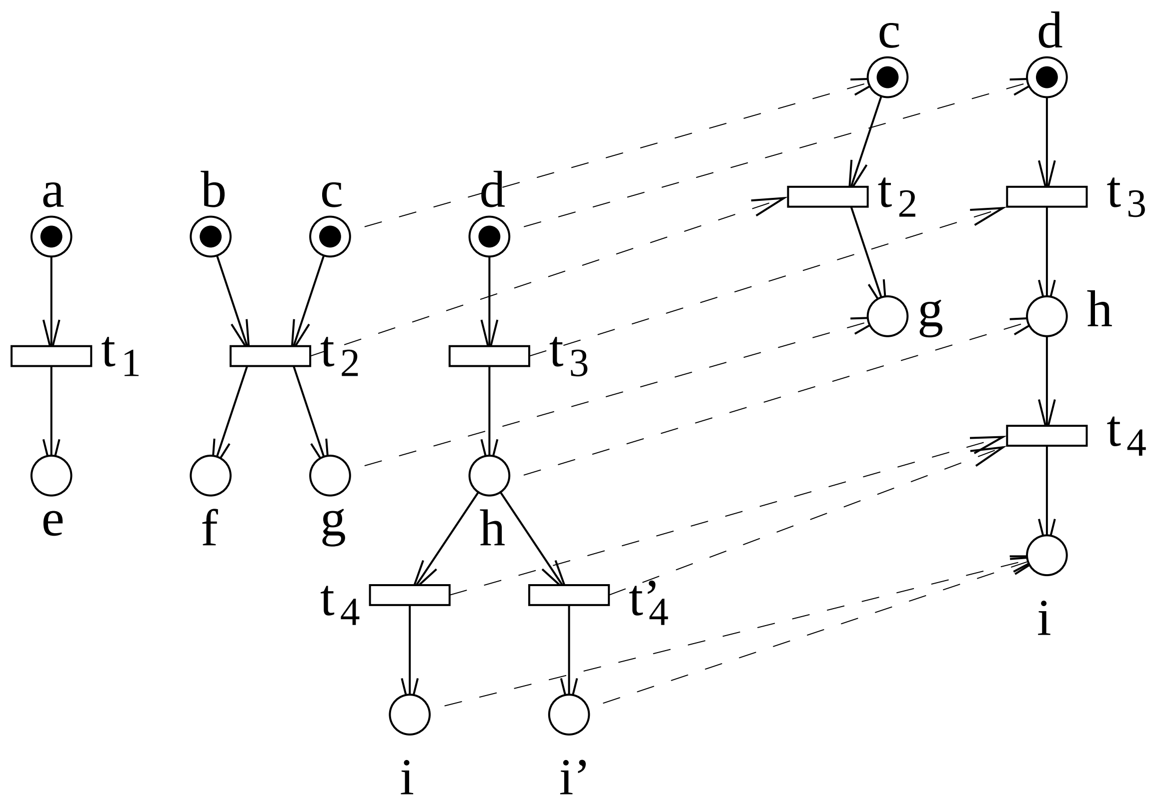
- for distributed processings we need a projection operator,
- notion of (factorized) finite complete prefix,
- can we imagine intermediate structures where conflicts are partially unfolded (to make configurations more easily readable) ?

Question for specialists : are trellis nets known ? Trivial ? Useful ?

Secret Slides

Morphism : $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$

1. ϕ = partial function on places and transitions,
2. ϕ preserves the flow relation (and labels),
3. the restriction $\phi : M_1^0 \rightarrow M_2^0$ is bijective (on its def. domain),
4. ϕ defined on $p_1 \quad \Rightarrow \quad \phi$ defined on ${}^\bullet p_1$ and $p_1^\bullet$,
5. ϕ defined on $t_1 \quad \Rightarrow \quad$ the restrictions $\phi : {}^\bullet t_1 \rightarrow {}^\bullet\phi(t_1)$ and $\phi : t_1^\bullet \rightarrow \phi(t_1)^\bullet$ are bijective (on their definition domain).



- we also add the possibility to *duplicate* places (otherwise the category of $Nets$ is uncomplete and doesn't have a product)