Modeling and Verifying Distributed Systems with Petri Nets :
Coloured Petri Nets

IWAISE'12

<u>Souheib Baarir,</u> Fabrice Kordon

First.last@lip6.fr

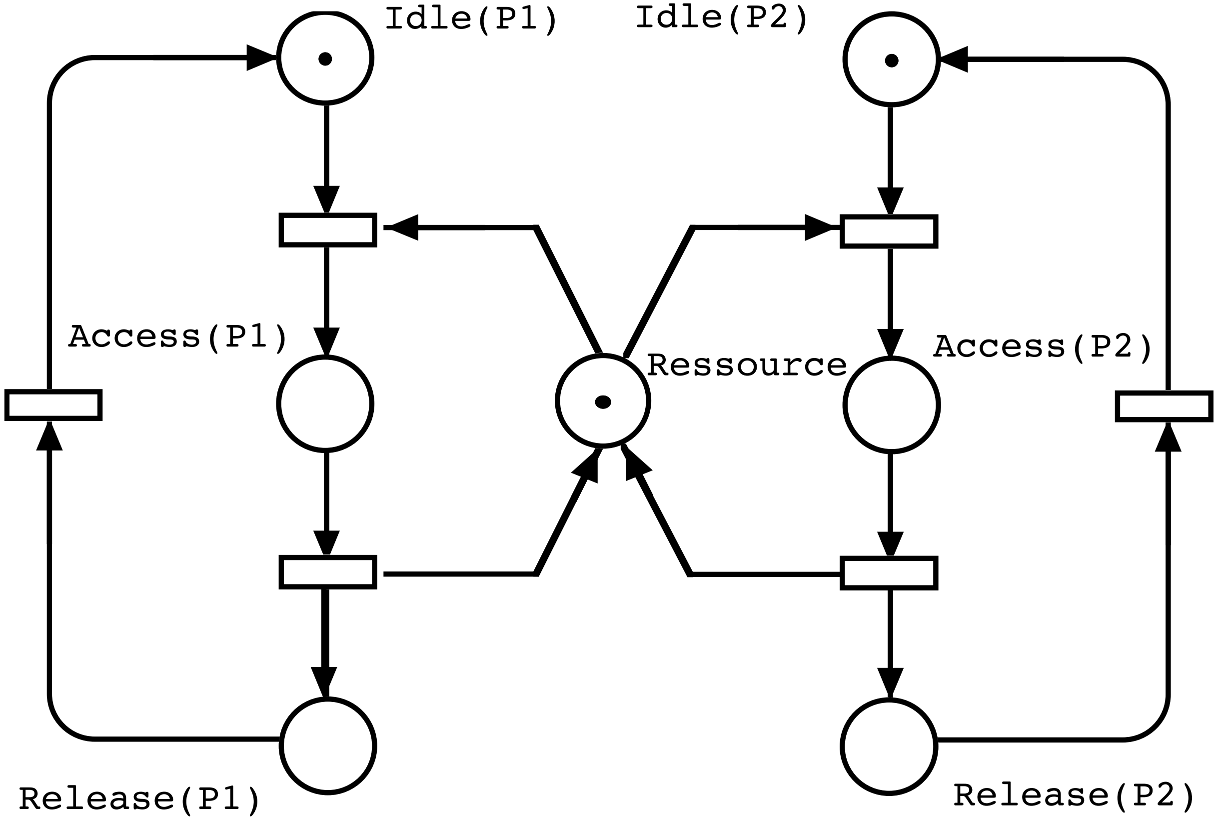
Labrotaoire d'Informatique de Paris 6

Why High level Petri Nets ? (1/3)

- Problem :

  n process in mutual exlusion on one ressource

n = 2

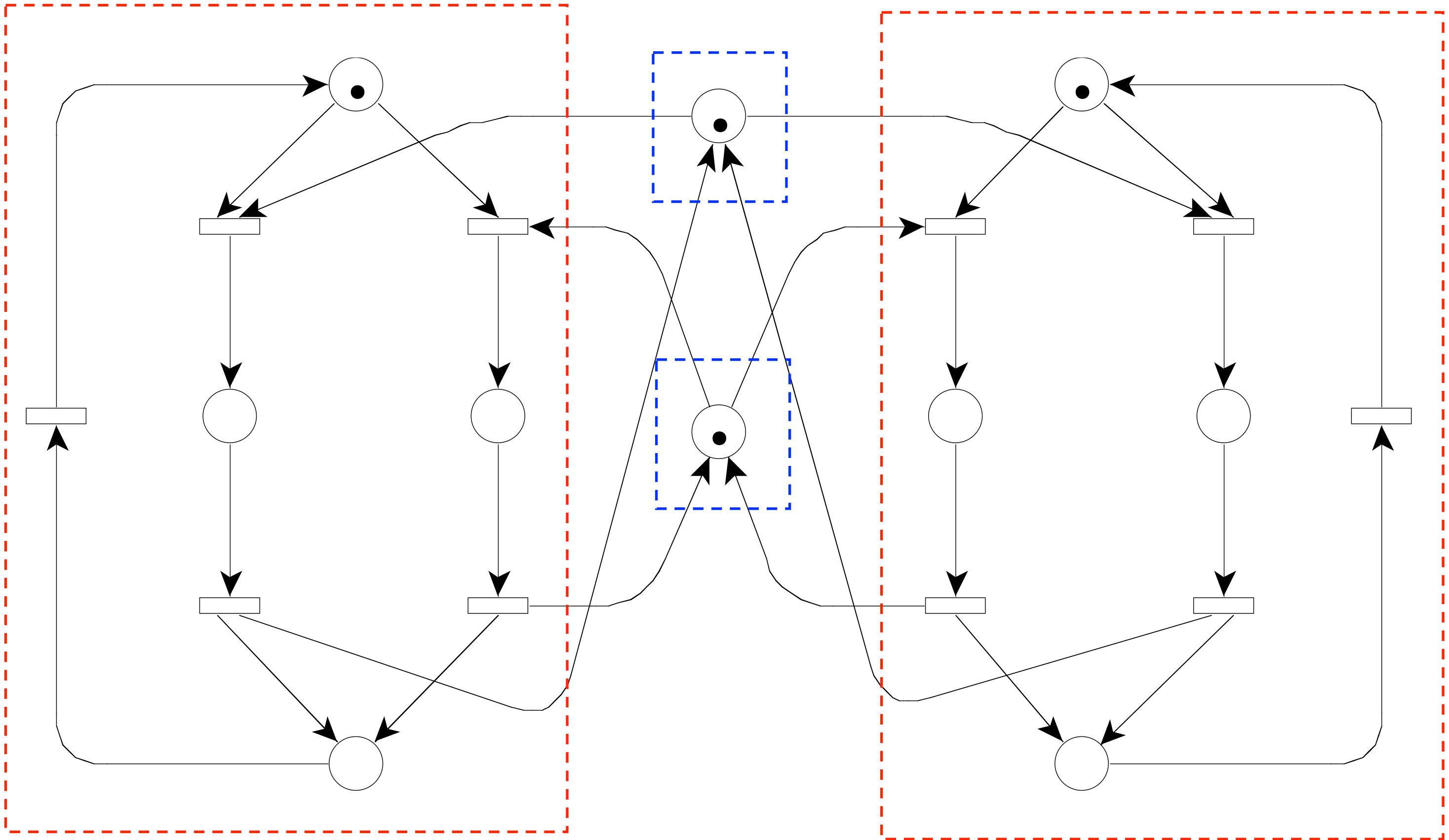


Why High level Petri Nets ? (2/3)

- Problem : n_1 process in mutuel exclusion on n_2 ressources

$n_1 = 2$

$n_2 = 2$



Why High level Petri Nets ? (3/3)

- Ordinary Petri (P/T) Nets:
  - do not capture symmetries of problems,
  - do not associate information to tokens,
  - do not allow parameterisation of solutions to problems

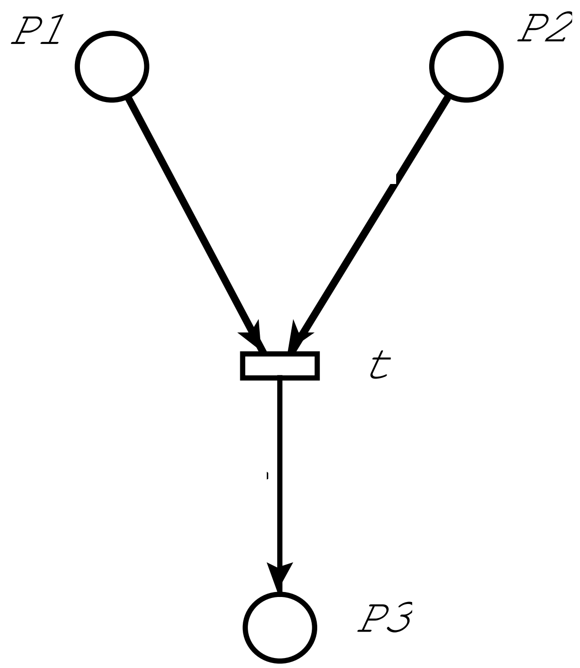
➥ Use of a concise and parameterized notation of Petri Nets:

High level Petri Nets

Coloured Petri Nets

Informal definition

- Each place p is characterized by a colour domain C(p).
- A token of p is an element of C(p).
- Each transition t is characterized by a colour domain C(t).
- The colour domain of a transition characterizes the signature of the transition.
- The colour functions on arcs determine the instances of token that are consumed and produced during the firing of a transition.

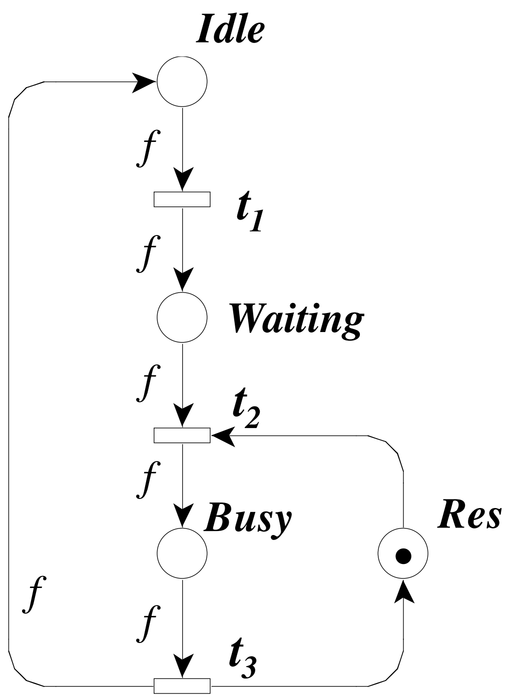


An example

- n processes of class $C = \{p_1, \ldots, p_n\}$, in mutual exclusion on a untyped resource.

- A process is either in an Idle state, or in a Waiting state, or in a Busy state.

- To move from the Waiting state to the Busy state, a process needs the resource.



$C(Idle) = C(Waiting) = C(Busy) = C$

$C(Res) = \{\varepsilon\}$

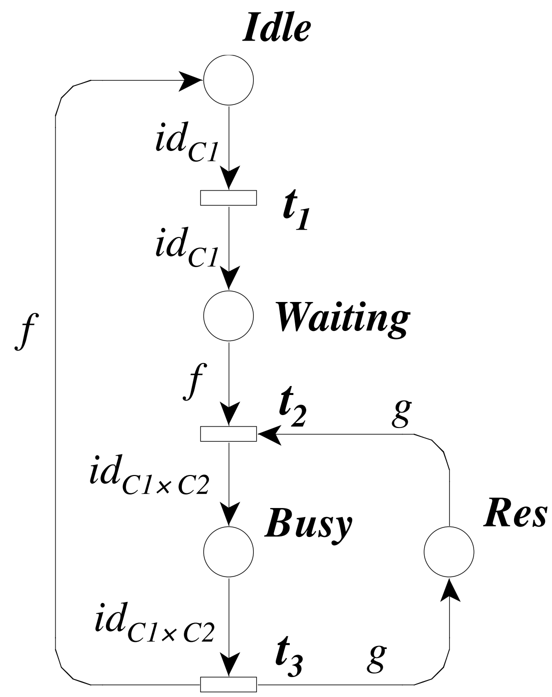
$C(t_1) = C(t_2) = C(t_3) = C$

$f: C \rightarrow C$
$\quad x \rightarrow x$

$M_0(Idle) = C.All$

An other example

- n_1 processes of class $C = \{p_1, \ldots, p_{n1}\}$, in mutual exclusion on n_2 ressources of class $C_2 = \{r_1, \ldots, r_{n2}\}$
- To move from Waiting to Busy, a process p_i needs a resource r_j.



$C(Idle) = C(Waiting) = C_1$

$C(Res) = C_2$

$C(Busy) = C_1 \times C_2$

$C(t_1) = C_1$

$C(t_2) = C(t_3) = C_1 \times C_2$

$f : C_1 \times C_2 \rightarrow C_1$
$(x_1, x_2) \rightarrow x_1$

$g : C_1 \times C_2 \rightarrow C_2$
$(x_1, x_2) \rightarrow x_2$

$M_0(Idle) = C_1 . All$

$M_0(Res) = C_2 . All$

Recall : multisets (bags)

- Let A be a non empty finite set.

- A bag a on A is a function:

  $$\mathbf{a} : A \rightarrow IN$$
  $$x \rightarrow a(x)$$

  a(x) denotes the number of occurrences of x in a.

- We note:

  $$a = \sum_{x \in A} a(x).x$$

- Bag(A) denotes the set of multisets of A.

Recall : functions on multisets

$f: Bag(C_1) \rightarrow Bag(C_2)$

$g: Bag(C'_1) \rightarrow Bag(C'_2)$

$h: Bag(C) \rightarrow Bag(C_1)$

$<f, g> : Bag(C_1) \times Bag(C'_1) \rightarrow Bag(C_2) \times Bag(C'_2)$

$(x, y) \rightarrow <f(x), g(y)>$

$f \circ h : Bag(C) \rightarrow Bag(C_2)$

$x \rightarrow f(h(x))$

Formal definition: the structure (1/2)

- A Coloured Petri Net (CPN) is a tuple : $<P, T, C, W^-, W^+, M_0>$

- P is the set of places, T is the set transitions ($P \cap T = \emptyset$, $P \cup T \neq \emptyset$).

- C defines for each place and transition a colour domain.

- W^- (= Pre) (resp. W^+ = Post), indexed on P x T, is backward (resp. forward) incidence matrix of the net.

- $W^-(p, t)$ and $W^+(p, t)$ are linear colour functions defined from **Bag(C(t)) to Bag(C(p))**

Formal definition: the structure (2/2)

- M_0 is the initial marking of the net:

  $$M_0(p) \in Bag(C(p))$$

- Transitions may be guarded by functions:

  $$Bag(C(t)) \rightarrow \{0, 1\}$$

- Colour domains are generally Cartesian products.

Formal definition: the dynamic (1/2)

Let CN = $<P, T, C, W^-, W^+, M_0>$ be a CPN.

- A marking M of CN is a vector: $M(p) \in Bag(C(p))$
- A transition t is enabled for an instance $c_t \in C(t)$ and a marking M *iff*:
  - Either t is not guarded, or the guard evaluates to true (for c_t), and
  - $\forall\ p \in P, M(p) \geq W^-(p, t)(c_t)$

Formal definition: the dynamic (2/2)

- M', the reached marking after the firing of t for an instance c_t, from the marking M is defined by:

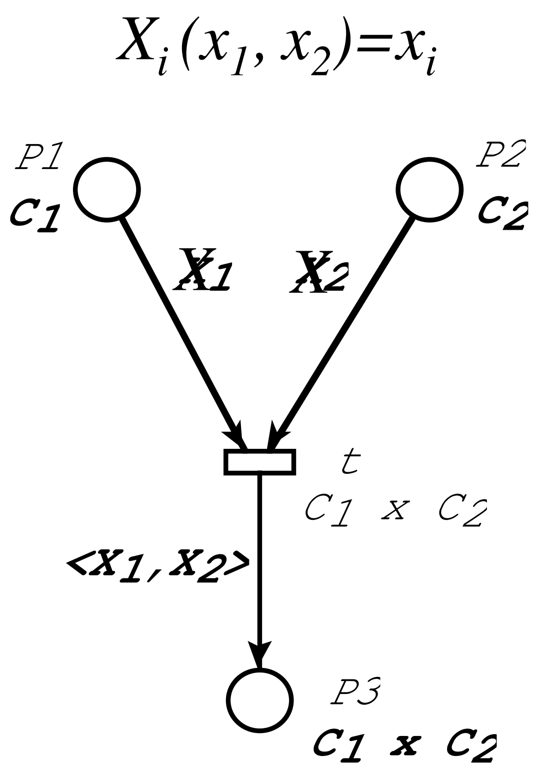
$$\forall\, p \in P,\ M'(p) = M(p) - W^-(p, t)(c_t) + W^+(p, t)(c_t)$$

We note :

$$M\ [(t, c_t)\rangle\ M'$$

$$M \xrightarrow{(t,\, c_t)} M'$$

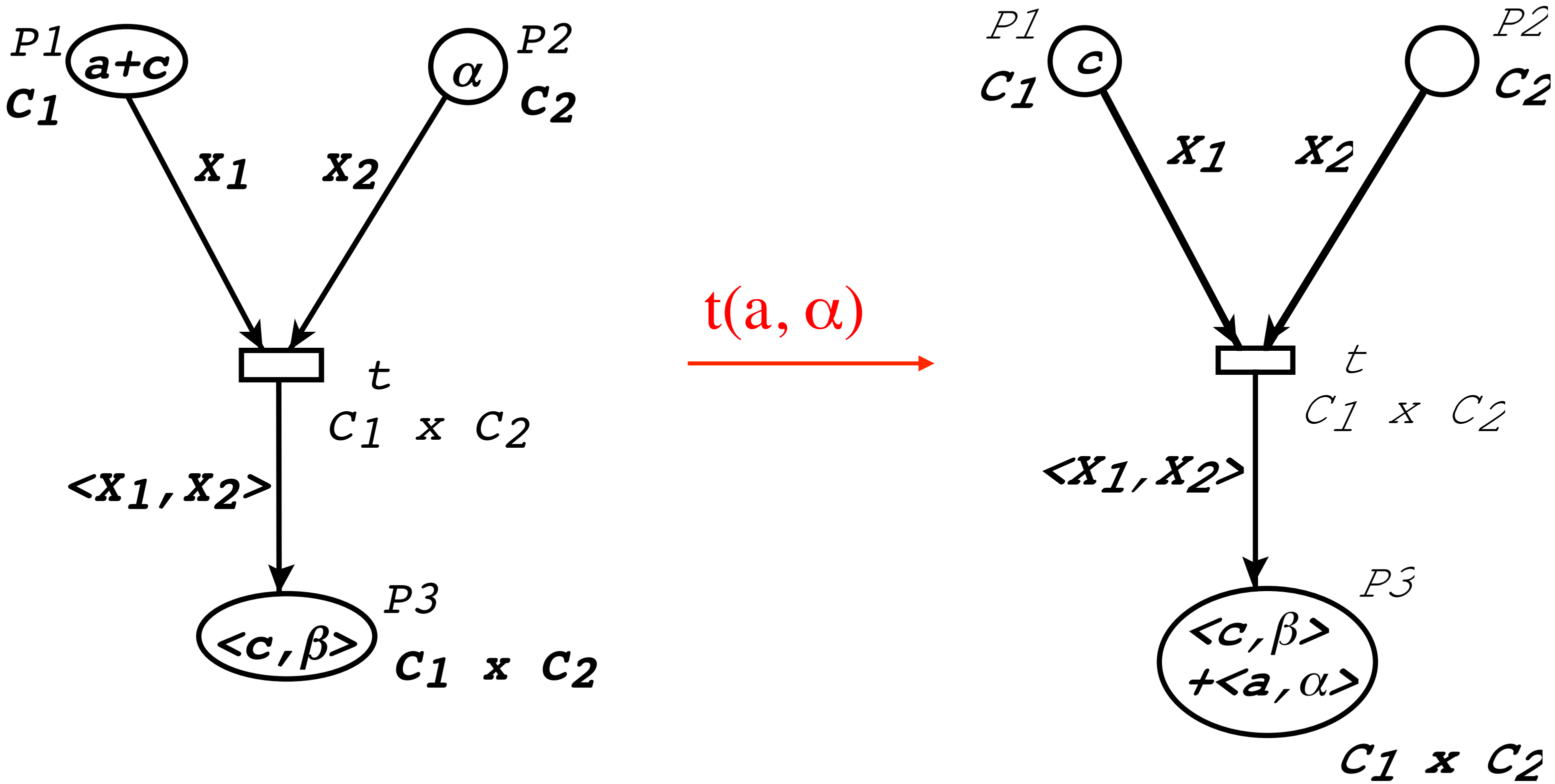
Example of firing (1/2)



- Let $x_1 \in C_1$, $x_2 \in C_2$
- t is enabled for (x_1, x_2) *iff*:
  1) P1 is marked by a token of colour x_1
  2) P2 is marked by a token of colour x_2
- If t is fired for (x_1, x_2) then:
  1) A token of colour x_1 is removed from P1
  2) A token of colour x_2 is removed from P2
  3) A token of colour $<x_1, x_2>$ is produced in P3 : $<X_1, X_2> (x_1, x_2) = <x_1, x_2>$

Example of firing (2/2)

$C_1 = \{a, b, c\}$ $C_2 = \{\alpha, \beta\}$



Basic colour functions (1/2)

$$C = \prod_{i=1}^{n} \prod_{j=1}^{e_i} C_i$$

A colour domain constructed on top of a Cartesian product of colour classes, in which **C_i** appears **e_i** times.

$$c = <c_1^1, c_1^2, \ldots, c_1^{e1}, \ldots, c_n^1, c_n^2, \ldots, c_n^{en}> \in C$$

- **Identity/Projection :**
  - Noted by a variable: X, Y, or X_1, or X_1^1, or p, q, …

$$X_i^j(c) = c_i^j \qquad Y(<x, y>) = y$$

$$q(<p, q, r>) = q$$

Basic colour functions (2/2)

- **Successor (on a circularly ordered C_i)**

  Noted X_i++ or ($X_i \oplus 1$) or X_i!

  $$X_i^j{+}{+}(c) = successor(c_i^j)$$

  The successor relation is defined par the enumeration order of elements in class C_i

- **Diffusion / Synchronization (on C_i)**

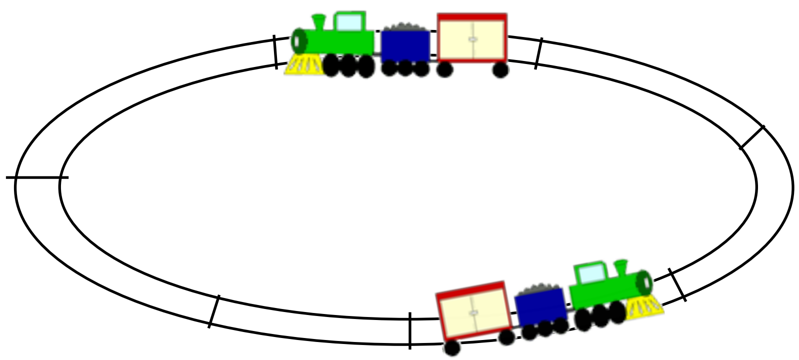
  Noted C_i.All or S_{Ci}

  $$C_i.All(c) = \sum_{x \in C_i} x$$

The Trains Problem (TP)

Problem :

- n_1 trains distributed on a circularly way, decomposed into n_2 sections.
- For security reasons, a train can enter a section only if this section and the next one are free.

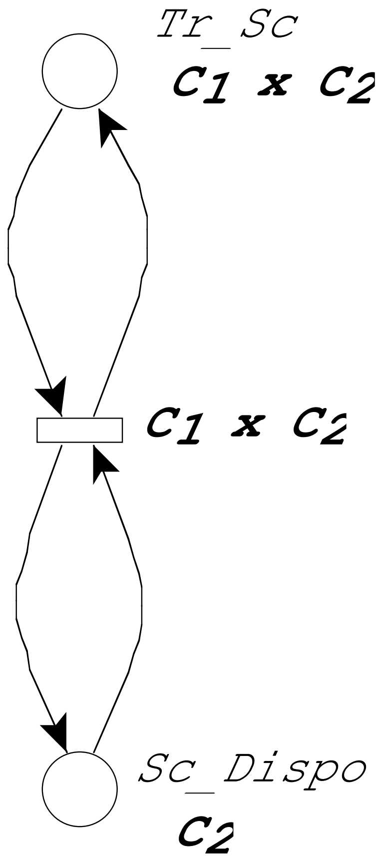


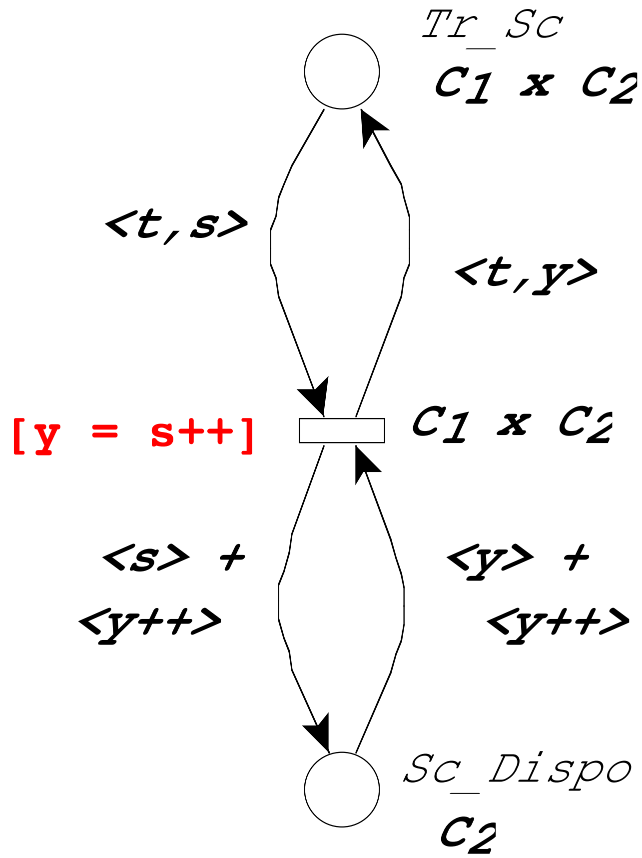
TP: models ?

- Colour Domains:
  - $C_1 = \{tr\_1, ..., tr\_n_1\}$
  - $C_2 = \{sc\_1, ..., sc\_n_2\}$

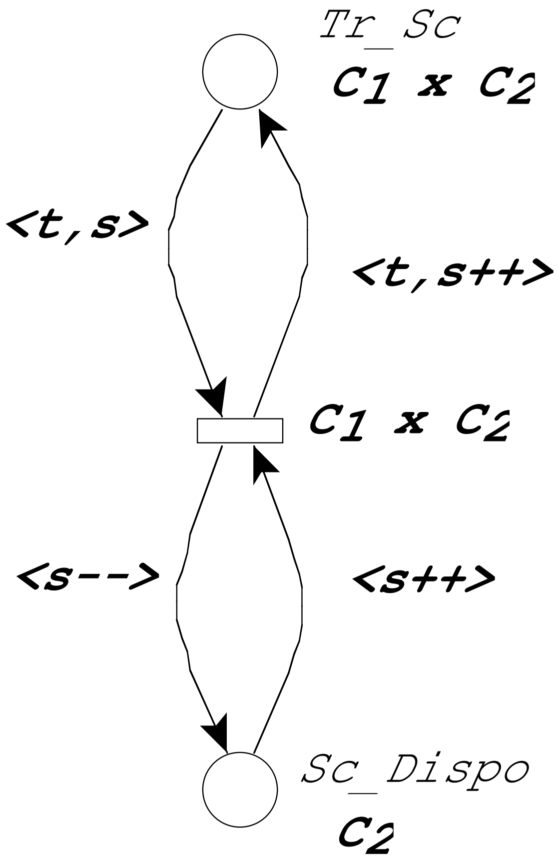
- The dynamic :
  - The system state is given by a set of associations < train n°, section n°>
    → **place Tr_Sc**
  - A free section is a resource that allows the move of a train
    → **place Sc_Dispo**
  - A transition representing the progression of a train.

TP: first models





TP: an other model



- Now, s represents the requested section

- Take care of the initial marking !

Unfolding a Coloured Net (1/2)

- To obtain an ordinary P/T Net, having the same behaviour than the CPN:
  - for each place or transition, we create a number of instances equals to the number of elements in the colour domain.
  - The connexions are obtained by « unfolding » the colour functions.
- Some times, it is the only way to get results on the model.
  - However, we do not know how to express theses results on the original model.
- Easy to automatize.

Unfolding a Coloured Net (2/3)

Let CN = $<P, T, C, W^-, W^+, M_0>$ be a CPN. The underling P-T Net is defined by $CN_d = <P_d, T_d, C_d, W^-_d, W^+_d, M_{0d}>$, where,

- $P_d = \bigcup_{p \in P, c_p \in C(p)} (p, c_p)$ is the set of places.
- $T_d = \bigcup_{t \in T, c_t \in C(t)} (t, c_t)$ is the set of transitions.
- $M_{0d}(p, c_p) = M_0(p)(c_p)$ is the initial marking.

Unfolding a Coloured Net (3/3)

- $W^{-}_{d}(p, c_p)(t, c_t) = W^{-}(p, t)(c_t)(c_p)$ is the backward incidence matrix

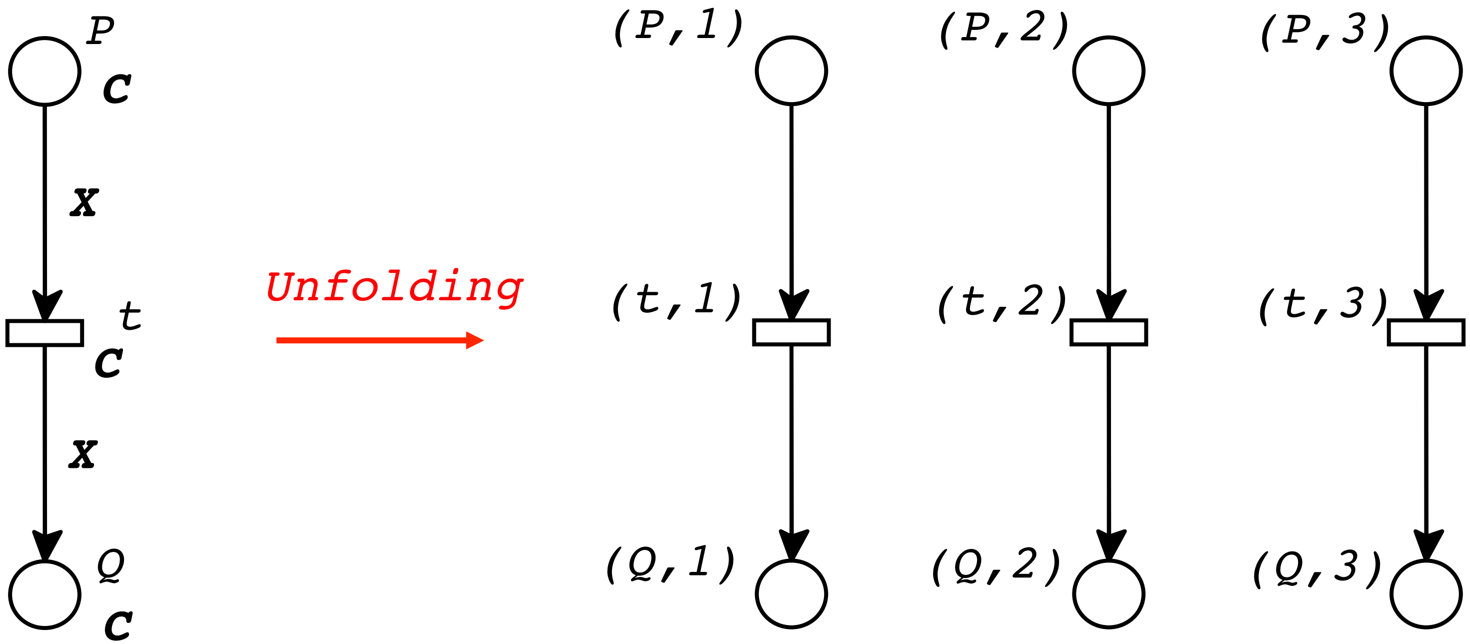
- $W^{+}_{d}(p, c_p)(t, c_t) = W^{+}(p, t)(c_t)(c_p)$ is the forward incidence matrix

Proposition :

$$M [(t, c_t) >_{CN} M' \quad \Leftrightarrow \quad M_d [(t, c_t) >_{CNd} M'_d$$

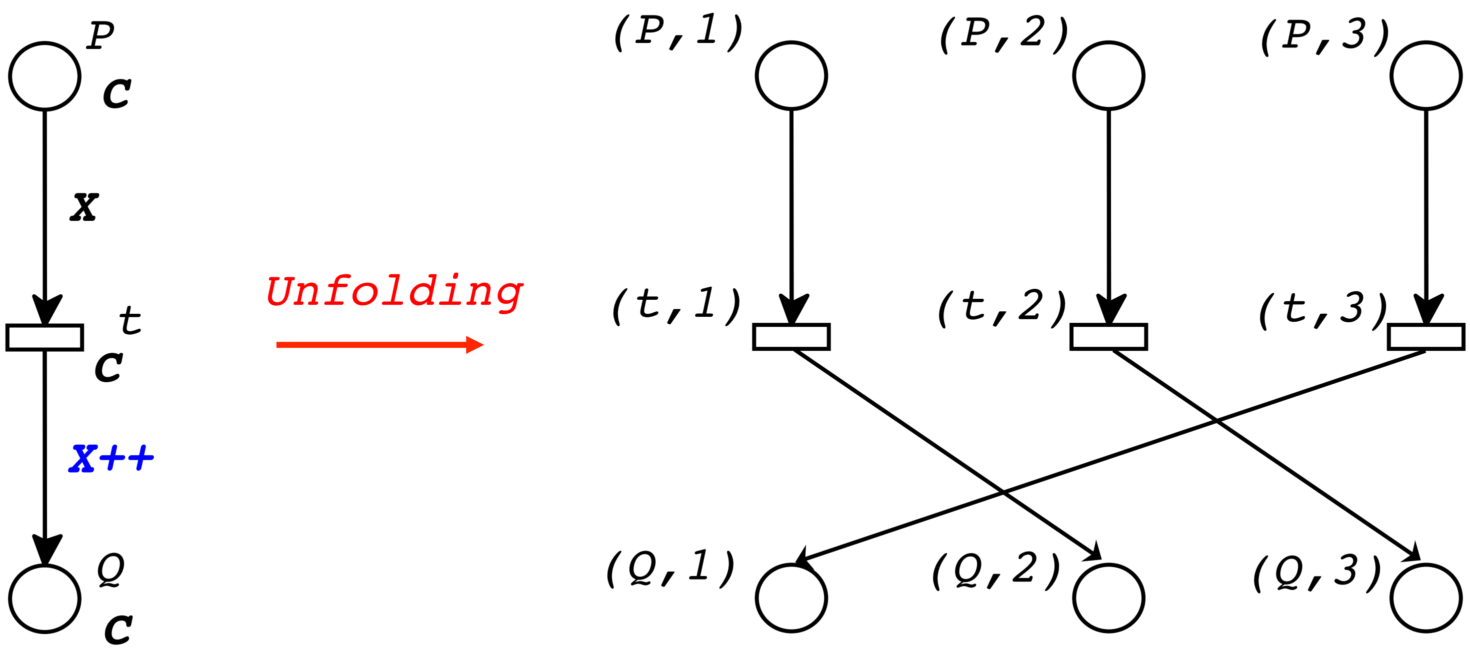
$$\text{where, } M_d(p, c) = M(p)(c)$$

Unfolding example (1/4)



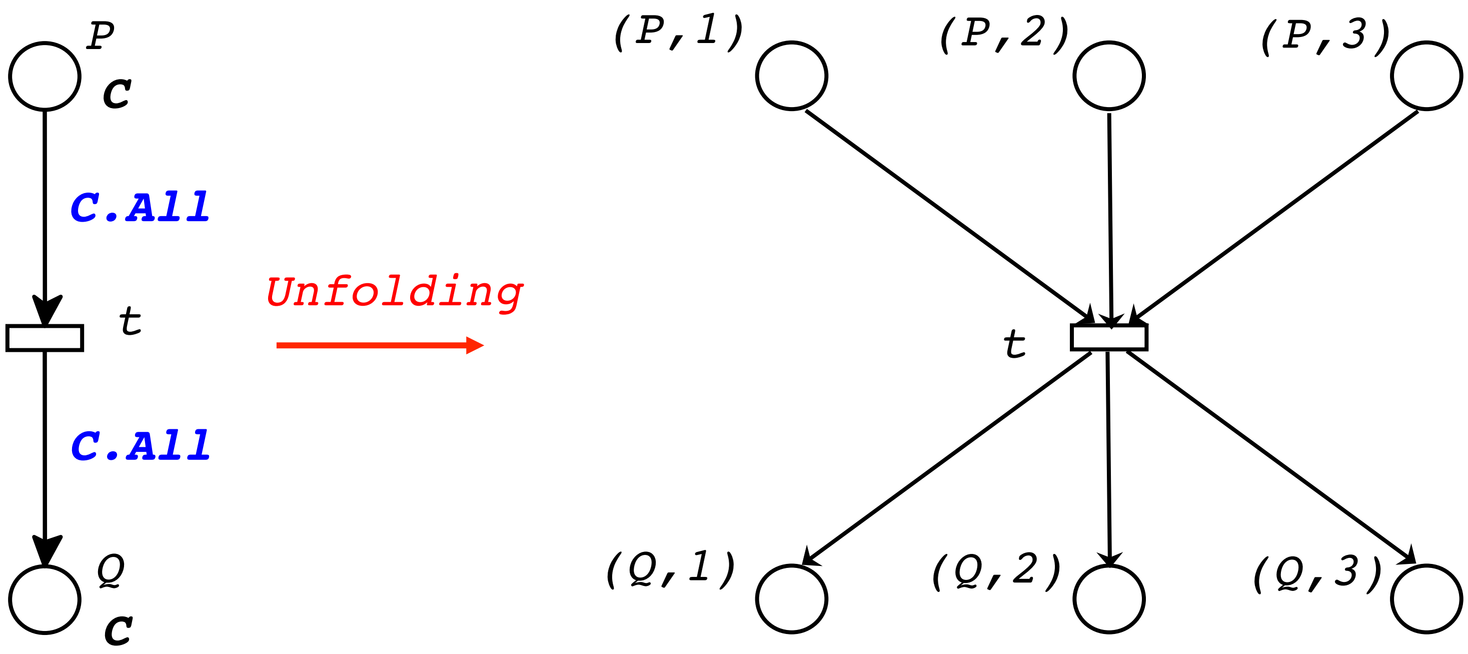
C = {1, 2, 3}

Unfolding example (2/4)



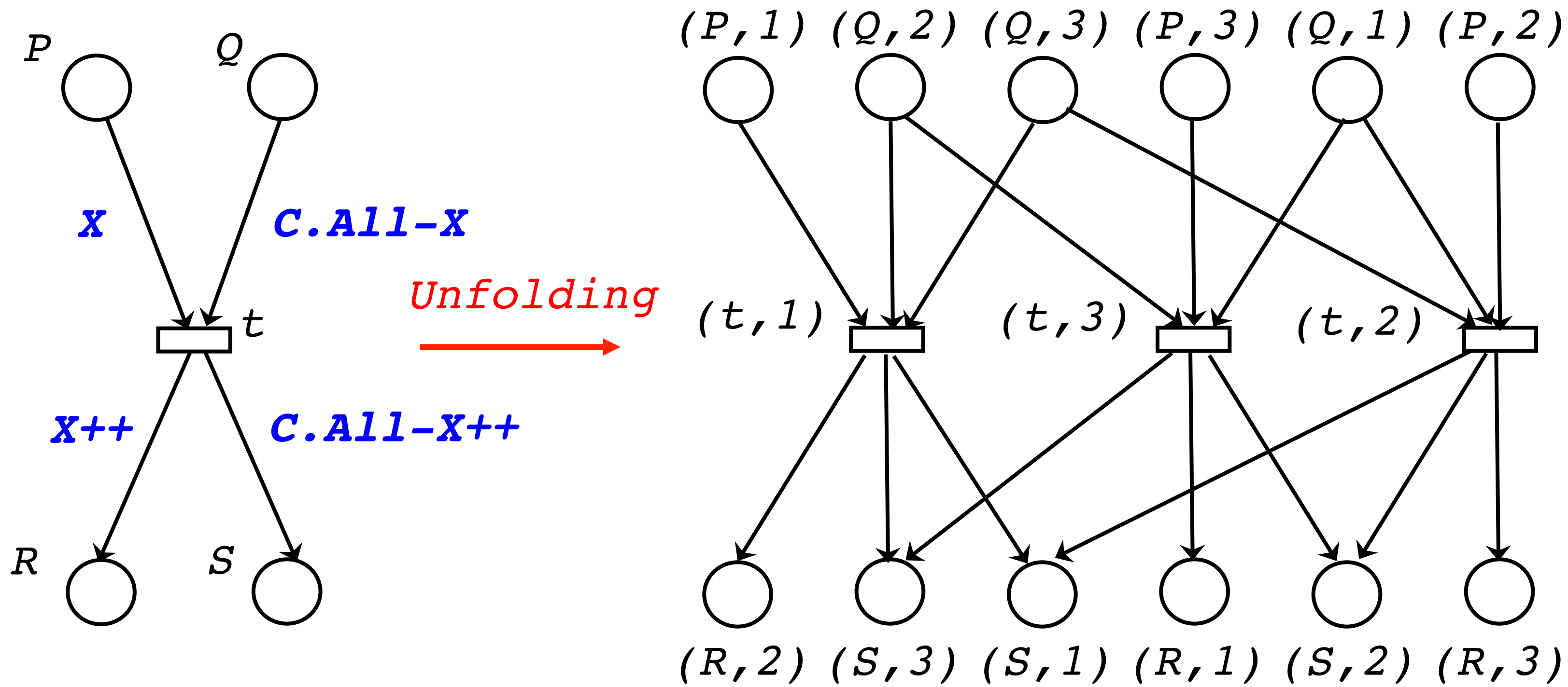
C = {1, 2, 3}

Unfolding example (3/4)



C = {1, 2, 3}

Unfolding example (4/4)

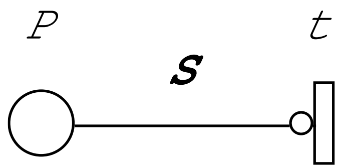


C = {1, 2, 3}

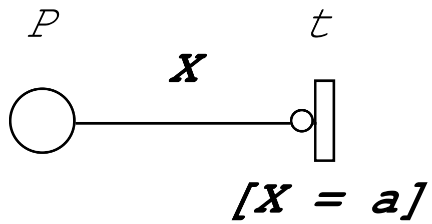
C(P)= C(Q)= C(R)= C(S)= C

Coloured inhibitor arcs

- To test the emptiness of a place:



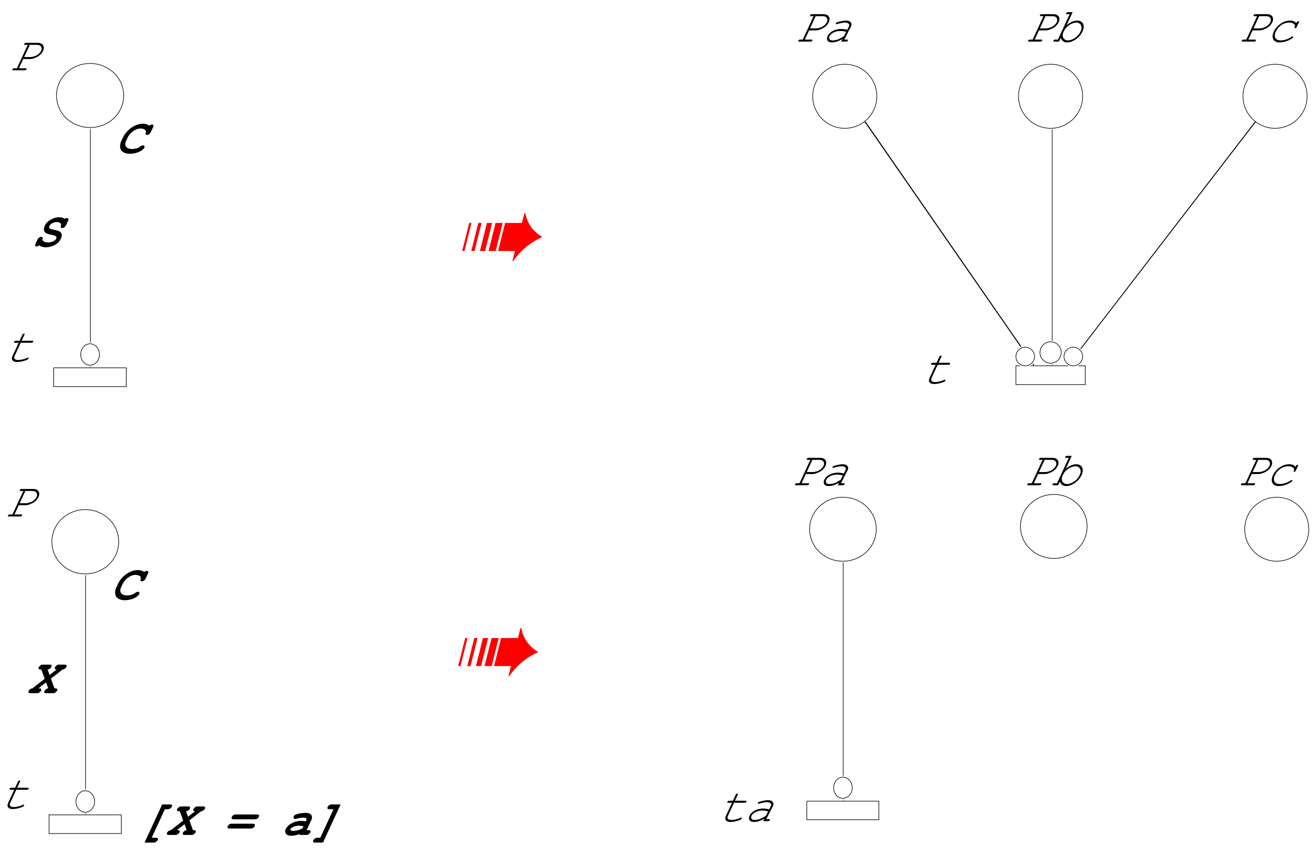
- To test that a place does note contain a colour a:



- The sets of colours must be finite!

Unfolding inhibitor arcs

$C = \{a, b, c\}$



Peterson's Algorithm

- Peterson algorithm : mutual exclusion of two processes.
  - The two processes are symmetrical.
  - A shared memory contains the variables: `turn`, $\texttt{dem}_p$ and $\texttt{dem}_q$.
- Code of process p:

```
A :     dem_p = true
B :     turn = q
C :     wait (turn == p || dem_q  == false)
D :     < Section critique >
E :     dem_p = false; goto A
```

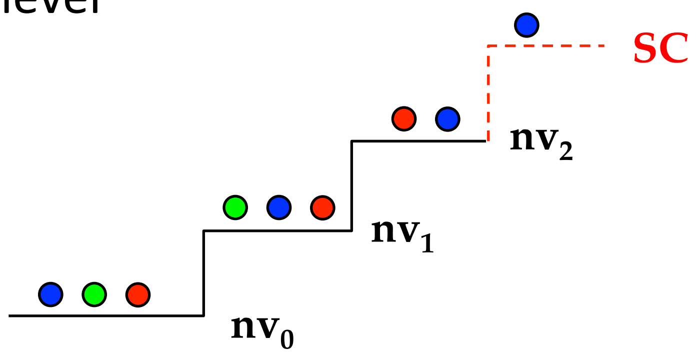
  - Initially :
    - $\texttt{dem}_p = \texttt{dem}_q = \texttt{false}$

Peterson's Algorithm: generalization to N processes (1/2)

- Principal :
  - Stairs of (N-1) levels
  - A process can move from level j to j+1 if:
    - It is not the last to get to the level j
    - It is the only process in the level j and all higher levels are free.

➥ Only one process can get beyond the level N-1

➤ Critical section



N = 3

Peterson's Algorithm: generalization to N processes (2/2)

Process x (x == 1 . . N-1)

```
Flag[x] = 0;
While (1) {
   For (j=1; j<N; j++) {
    Flag[x] = j;
    Turn[j] = x;
    wait until
        ((∀ y ≠ x, (Flag[y] < j)) || (Turn[j] ≠ x))
   }
   <Section critique>
   Flag[x] = 0;
}
```

Analyse of a CPN

- The precedent models are they conform to the specification ?

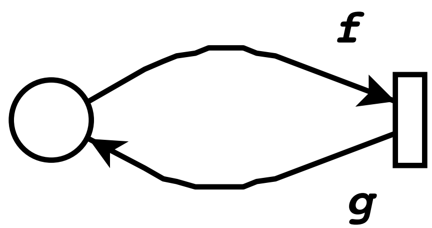
- Possibility of answers thanks to:
  – The construction of the reachability graph
  – Linear invariants
  – The reduction theory

- Try to take benefits from the structure of the model induced by the colour functions.

Why limit the colour function ?

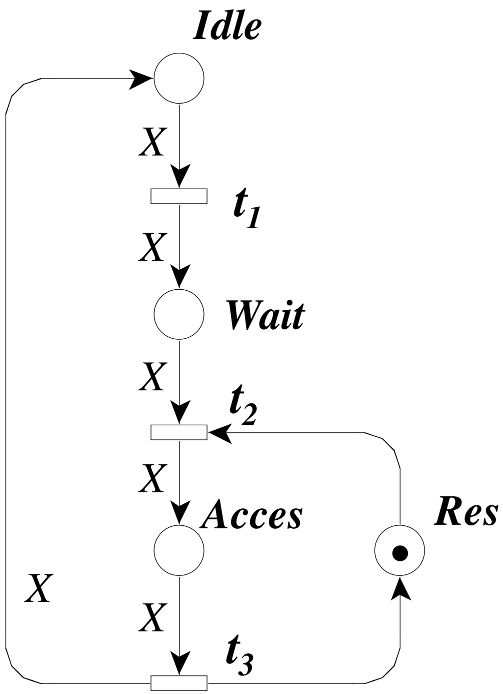
- To preserve the readability of the model
  - Each Petri Net can be represented…

Like that !



- Because the properties of the functions allow the automatic construction of graph of classes instead of an ordinary graph

Example of simple critical section



$C = \{c_1, c_2, c_3\}$

$M_0(Idle) = C.All$

$\boldsymbol{M_0}$

$Idle(c_1 + c_2 + c_3) + Res$

(t_1, c_1) → $idle(c_2 + c_3) + Res + Wait(c_1)$ $\boldsymbol{M_1}$

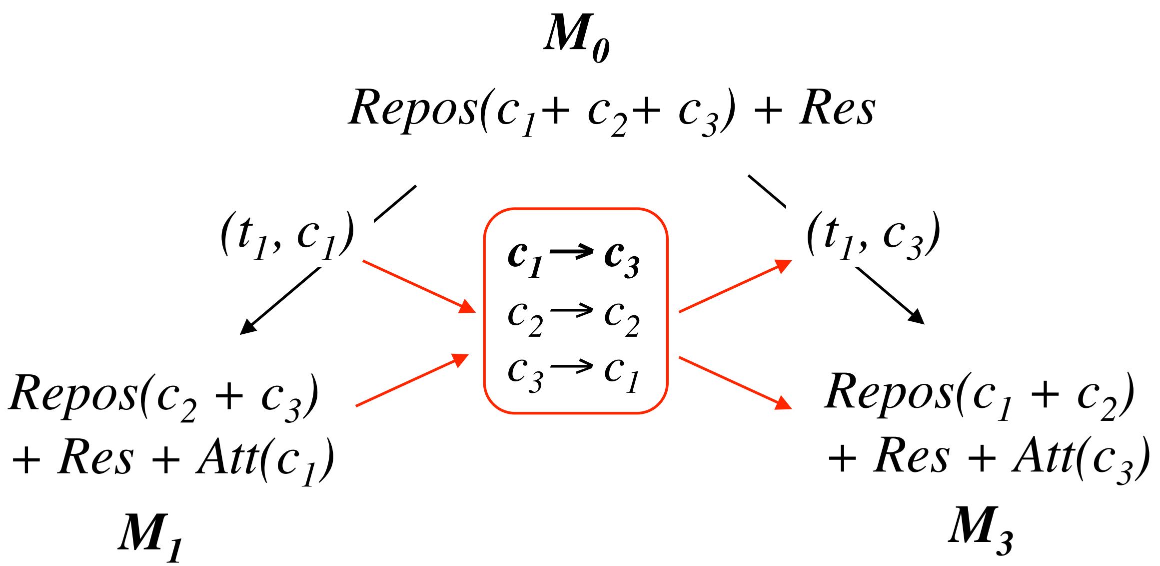
(t_1, c_2) → $Idle(c_1 + c_3) + Res + Wait(c_2)$ $\boldsymbol{M_2}$

(t_1, c_3) → $idle(c_1 + c_2) + Res + Wait(c_3)$ $\boldsymbol{M_3}$

Towards the use of symmetry (1/2)

- In the initial Marking, t_1 is enabled for each colour instance marking the place *Idle.*

- If we apply a permutation on the transition colour, the obtained marking are identical up to this permutation.



Towards the use of symmetry (2/2)

- We can represent this set of firings using variables :

$$idle(x+y+z) + Res \xrightarrow{(t_1, z)} Idle(x + y) + Wait(z) + Res$$

$$x, y, z \in C, \quad x \neq y \neq z$$

- Then, we obtain the actual firings by testing all possible instantiations for x, y et z.

- Is it general ?

Permutations on a Bag

- Let A be a set, *s* a permutation on A, and *a* a bag of A.

$$s.a = s.\left(\sum_{x \in A} a(x).x\right) = \sum_{x \in A} a(x).s(x)$$

- In particular : $s.a(s.x) = a(x)$ (notation : s.c = s(c))

- Example :

$$\mathbf{M(p) = c_1 + 2.c_2} \qquad s.c_1 = c_3 \qquad s.c_2 = c_1 \qquad s.c_3 = c_2$$

$$s.M(p)(s.c_1) = M(p)(c_1) = 1 \qquad s.M(p)(s.c_2) = M(p)(c_2) = 2$$

$$\mathbf{s.M(p) = c_3 + 2.c_1}$$

Enabling and firing equivalence

(t, c_t) is enabled in M ⇔ **$(t, s.c_t)$ is enabled from s.M**

$$M \xrightarrow{(t,\, c_t)} M' \quad \Leftrightarrow \quad \mathbf{s.M} \xrightarrow{(\mathbf{t},\, \mathbf{s.c_t})} \mathbf{s.M'}$$

Markings Equivalence

- Set of symmetries.
  - For each unordered class C_i, we associate a permutation group S_i
  - For each ordered class C_i, we associate a rotation group S_i
  - The symmetries of the Net are defined by the set S : $S = \{<s_1, \ldots, s_n> \mid s_i \in S_i\}$

- Markings equivalence ($\equiv$) :

$$M \equiv M' \Leftrightarrow \exists\, s \in S,\ M' = s.M$$

Classes of markings

- For each marking M, we define Cl(M):

  $$Cl(M) = \{ M' \mid \exists s \in S, M' = s.M \}$$

- Fundamental properties of Cl(M) :
  - $M \xrightarrow{(t,\, c)} M' \Rightarrow \forall s \in S,\ s.M \xrightarrow{(t,\, s.c)} s.M'$
  - If M_0 is symmetric ($\forall s \in S,\ s.M_0 = M_0$), and M is reachable, then

    $$\forall M' \in Cl(M),\ M' \text{ is reachable}$$

  - $\forall s \in S$ such that s.M = M,

    $$M \xrightarrow{(t,\, c)} M' \Rightarrow M \xrightarrow{(t,\, s.c)} s.M'$$

Thus, we can define classes of firings.

What else ?

- By defining an adequate representation for marking classes,
  - Dynamic Subclasses
  - Symbolic marking

- By defining a firing rule that applies directly on this representation,
  - Symbolic firing rule

✓ We can construct directly a ***quotient graph*** that represents the set of reachable markings.

Dynamic subclasses for unordered class

- We group in a set (dynamic subclass) the objects of C_i that have the same marking.
- Example :

$$M = Idle(c_1+c_2) + Wait(c_3) + Res$$

$$\Rightarrow \quad Idle(x + y) + Wait(z) + Res$$

$$M(x) = M(y) \quad \Rightarrow \quad Z^1, \ |Z^1| = 2$$

$$M(z) \neq M(x) \text{ et } M(z) \neq M(y) \quad \Rightarrow \quad Z^2, \ |Z^2| = 1$$

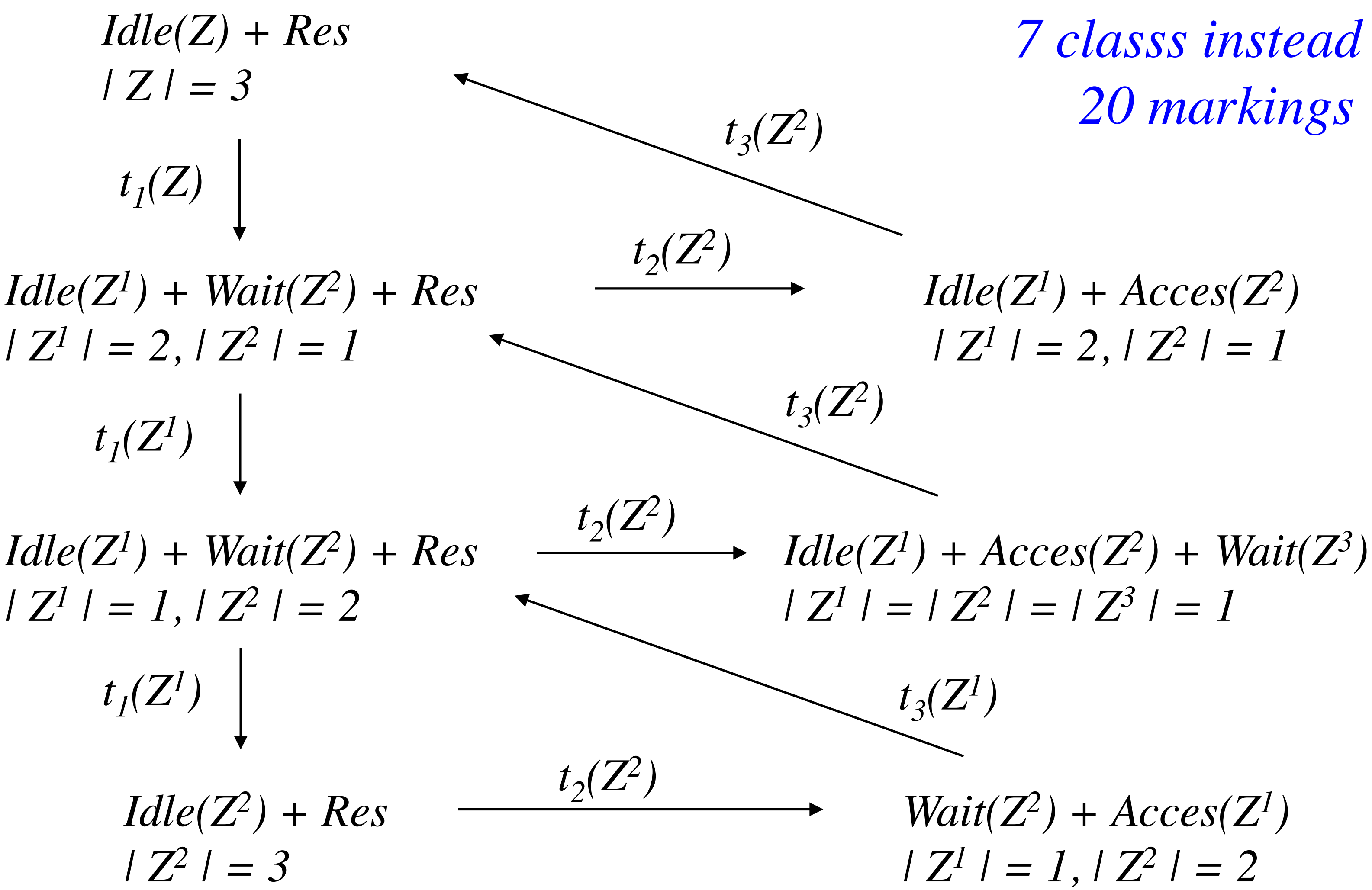
$$\Rightarrow \quad \boxed{Idle(Z^1) + Wait(Z^2) + Res, \ |Z^1| = 2, \ |Z^2| = 1}$$

$$Idle(c_2+c_3) + Wait(c_1) + Res$$

$$Idle(c_1+c_3) + Wait(c_2) + Res$$

$$Idle(c_1+c_2) + Wait(c_3) + Res$$

Informal example



7 classs instead of 20 markings

Firing rule

- Before firing, we decompose the dynamic sub-classes to isolate the objects that are used to instantiate the colour functions.

- <u>Example</u> :

$$\mathbf{Repos(Z) + Res} \quad \Rightarrow \quad \mathbf{Repos(Z^{1} + Z^{1,0}) + Res}$$

$$|Z| = 3 \qquad\qquad |Z^{1}| = 2, |Z^{1,0}| = 1$$

$Z^{1,0}$ contains the chosen object to instantiate X, Z^{1} those that are not participating to the firing.

- We then apply the classical firing rule.

- After the firing, we must group the resulting subclasses…

What does the Symbolic Reachability Graph preserve ?

- Each marking represented by a class (a symbolic marking) is reachable.

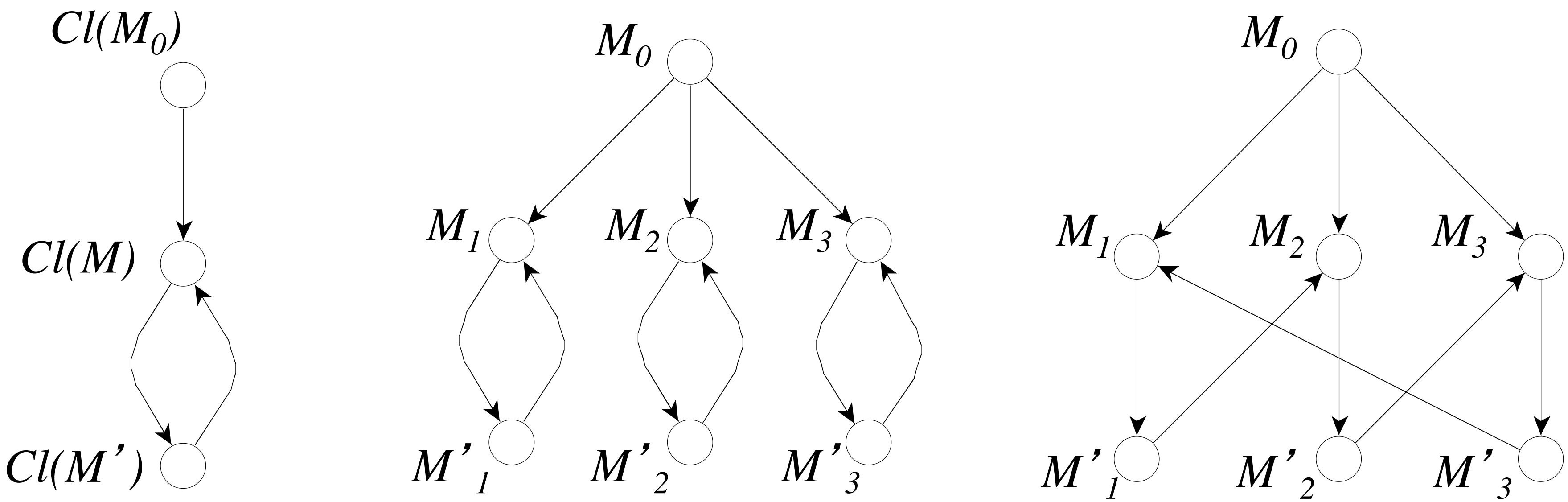
- Each reachable marking is represented by a class.

- Each firing sequence of the RG is represented in the SRG.

- To each sequence of the symbolic graph corresponds a sequence of the RG.

Then, what is missing ?

- We can not distinguish the following situations :



➠ **Miss of information on the home state.**

Can we represent any P-T Petri net ?

- Yes, but…
  - No interest if the representation is not reduced…

- The presented model imposes that all objects of the same class behave identically,
  - A class groups a set of objects that have the same nature

- We must be able to divide the class in subclasses of elements that can evolve differently: $C_i = D_i^1 \cup D_i^2$
  - Elements of D_i^1 evolve differently from those of D_i^2
  - The Diffusion functions are defined at the level of subclasses : D_i^1.All

Conclusion

- The construction of symbolic graphs applies on any coloured Petri net, but
  - Its efficiency depends directly from the symmetries…
  - The structuration of the model is necessary to have a direct and automatic construction.

- Almost all properties of interest can be checked on the symbolic graph.