



Modélisation de systèmes dynamiques

Réseaux de Pétri

Gauvain Bourgne (gauvain.bourgne@lip6.fr)

Représentations du temps et changement



RETOUR SUR LES LOGIQUES TEMPORELLES RÉIFIÉES

Preuve : déroulé

Au tableau

- a $OCCUR(Coupure, t_c)$
- b $\exists T_M (OCCUR(Meurtre, T_M) \wedge T_M\{d\}t_C)$
- c $\forall P \forall T ((Assassin(P) \wedge OCCUR(Meurtre, T)) \rightarrow HOLD(at(P, ch), T))$
- d $\exists T_S (T_S\{m\}t_C \wedge HOLD(at(s, sal), T_S))$
- e $\forall P \forall T \forall T' ((HOLD(at(P, sal), T) \wedge HOLD(at(P, ch), T') \wedge T\{<\}T') \rightarrow \exists T'' (OCCUR(Escalier(P), T'') \wedge T\{<\}T'' \wedge T''\{<\}T'))$
- f $\forall T_E, \exists T_B (ECAUSE(Escalier, T_E, Bruit, T_B) \wedge T_E\{o, d^t, e^t\}T_B)$
- g $\forall T_B (OCCUR(Bruit, T_B) \rightarrow T_B\{<, m, m^t, >\}t_C)$

O4 $\forall E \forall E' \forall T \forall T'$
 $(OCCUR(E, T) \wedge ECAUSE(E, T, E', T')) \rightarrow OCCUR(E', T')$



Autres logiques d'actions

- Calcul des évènements (discret)
 - Ensemble de fluents et d'événements (instantanés)
 - Temps linéaire (discret)
 - Prédicats
 - $\text{Holds}(f,t)$: le fluent f tient dans la situation s
 - $\text{Occurs}(e,t)$: l'evt e survient au temps t
 - Exemple :

$\text{Holds}(\text{ouvert},t+1) \Leftrightarrow \text{Occurs}(\text{ouvrir},t) \vee (\text{Holds}(\text{ouvert},t) \wedge \neg \text{Occurs}(\text{fermer},t))$

Autres logiques d'actions

- Calcul des situations
 - Ensemble de fluents et d'actions
 - Ensemble de situations s:
 - Situation initiale : s0
 - Situation $s' = do(a, s)$: situation obtenue en faisant l'action a dans la situation s
 - Ordre : $s < s'$ si $s' = do(a1, do(a2, ..., do(an, s) ...))$
 - Structure arborescente
 - Prédicats
 - Holds(f,s) : le fluent f tient dans la situation s
 - Exemple :
$$\text{Holds}(\text{ouvert}, \text{do}(a, s)) \Leftrightarrow a = \text{ouvrir} \vee (\text{Holds}(\text{ouvert}, s) \wedge a \neq \text{fermer})$$



Autres logiques d'actions

- Pour plus de détails sur modélisation de l'action et du changement (et planification symbolique) :

UE IAMSI au S2

Modéliser la dynamique



PARENTHÈSE CONTEXTUELLE : POINT DE VUE ‘CONCEPTION’

Spécification d'un système dynamique

Espace de spécification
-structuré autour de 3 axes :

- Axe Fonctionnel (F)
- Axe Structurel (S)
- Axe Temporel (T)

Axe Temporel ou Dynamique

(F, S, T)

Axe Structurel

Axe Fonctionnel



Aspects fonctionnels et structurels

- Aspect fonctionnel : « à quoi ça sert ? »
 - Exemple : le conducteur qui utilise une voiture la voit comme un objet permettant de se déplacer
- Aspect structurel : « quelle structure ? »
 - Exemple : le garagiste voit la voiture comme un ensemble de pièces qui interagissent et produisent un mouvement.



Vues complémentaires du système Fonctionnelle, Structurelle, Temporelle

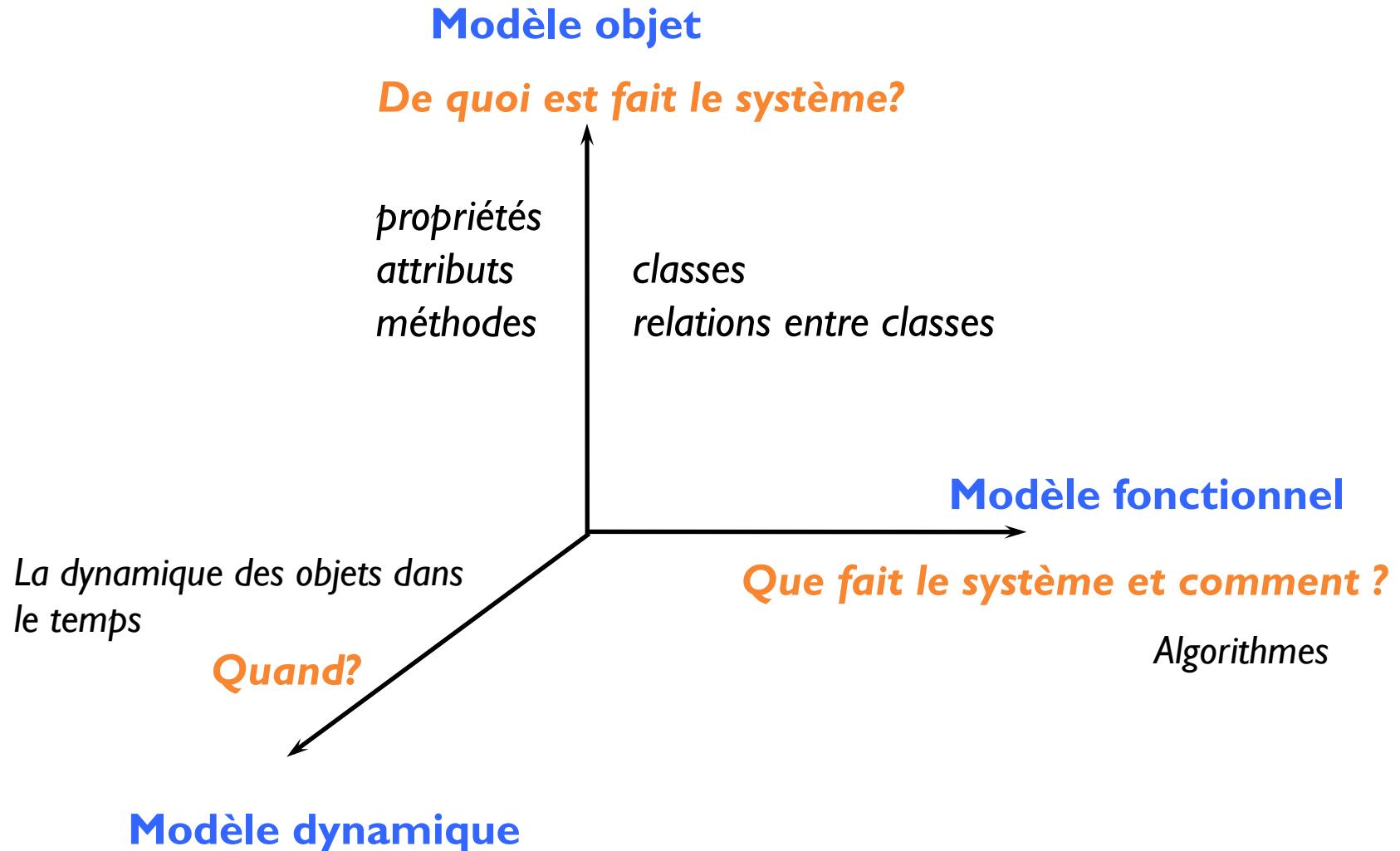
La complexité des systèmes oblige à étudier chacune de ces vues indépendamment des 2 autres.

Exemples de méthodes sur chacun des axes :

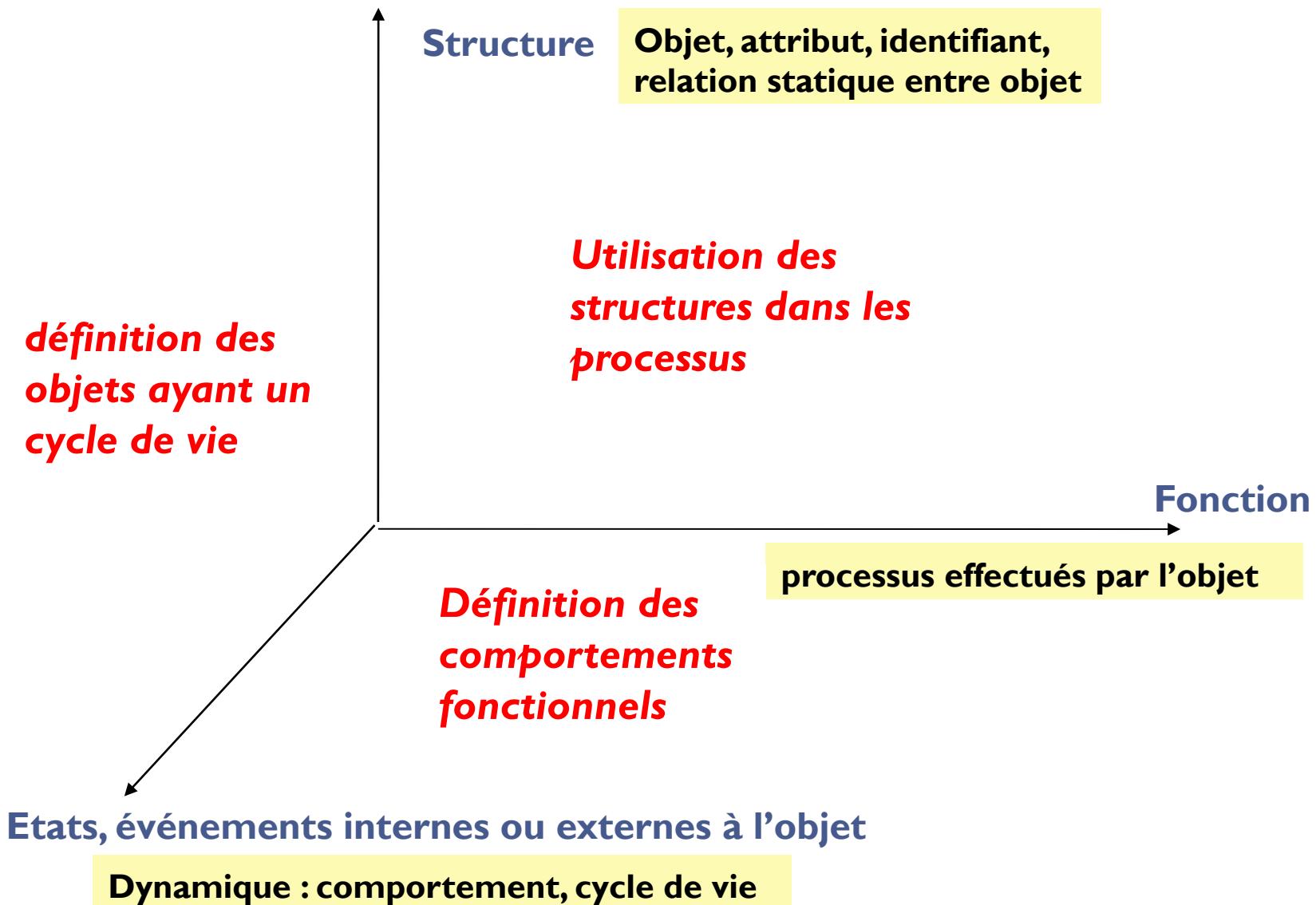
- **axe Fonctionnel** : SA (Structured Analysis)
illustre les fonctions du système
- **axe Structurel** : E/R (Entités/Relations)
illustre les données du système
- **axe Temporel** : Réseaux de Petri, Diagrammes Etats/transitions, etc.
illustre l'évolution du système

Modèles pour spécifier/analyser la dynamique du système

Exemple: axes de modélisation objet



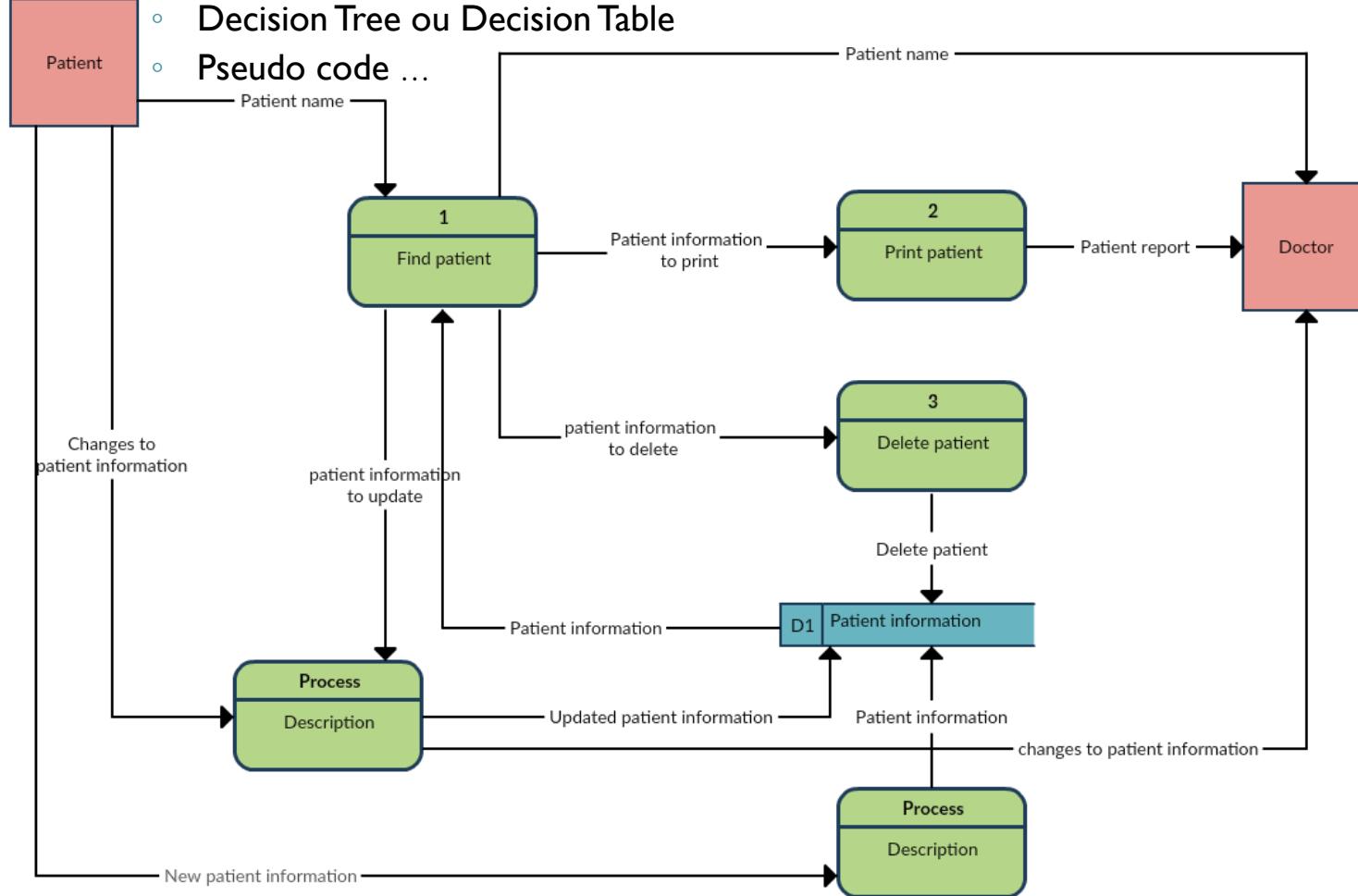
Exemple: axes de modélisation objet



Fonctionnel : Structured Analysis

- Différents outils

- Data Flow Diagram
- Decision Tree ou Decision Table
- Pseudo code ...

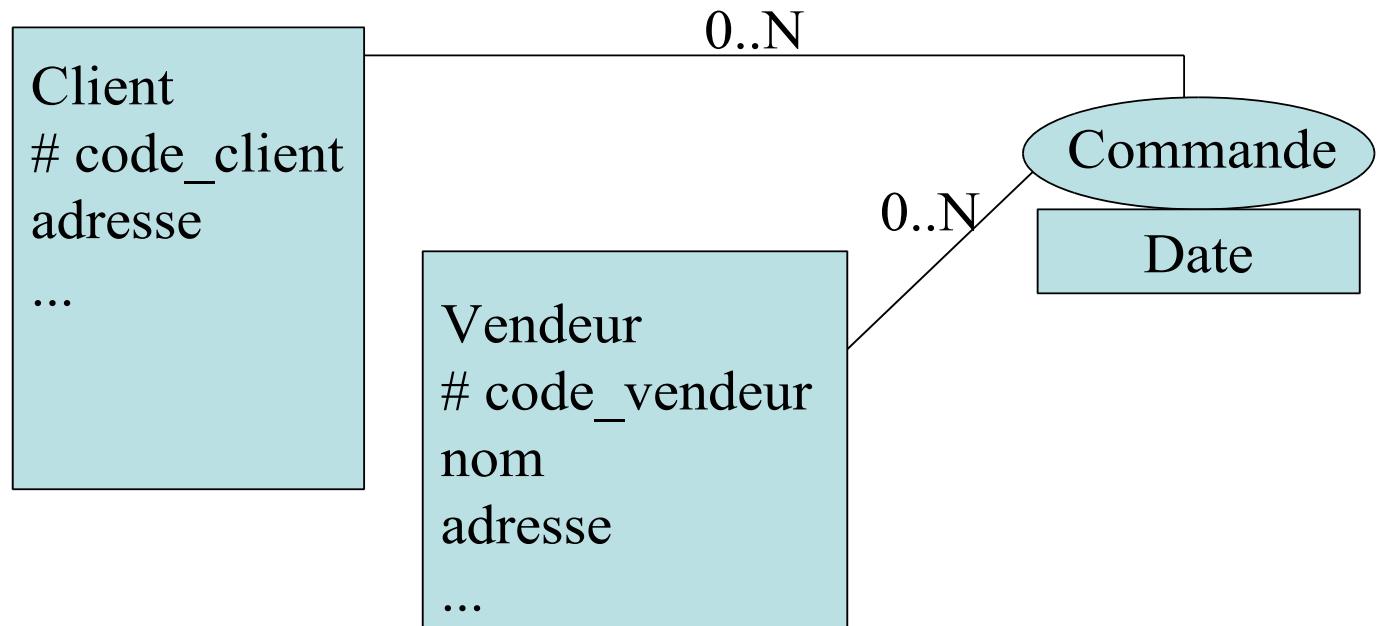


Structurel : modèle Entité/Relation

- **Entité** : Représente un ensemble, une collection, des objets du monde réel qui joue un rôle dans le système à étudier.

Une entité peut être décrit par plusieurs attributs

- **Relation** : représente un ensemble de liens ou associations entre les entités



Problème de cohérence...



Comment arriver à des spécifications cohérentes entre elles sur les 3 axes ?



Besoin de méthodes

- *Une méthode est un processus opératoire formel*
 - *Elle définit un **ordre logique** dans lequel les tâches doivent être réalisées pour atteindre un **but défini***
 - *Elle spécifie l'inventaire, la nature et le contenu des tâches ainsi que l'ordre d'exécution et les résultats escomptés de la tâche.*



Méthodes semi-formelles

- Une méthode semi-formelle
 - Un langage (parfois graphique) + syntaxe précise + sémantique non précise (ou inexistante)
 - Divers outils d'analyse
 - Exemple : SADT, SA-RT, Merise, OMT, **UML**
- Elles sont utiles pour défraîchir le problème
 - En particulier en phase d'analyse
- Impossibilité de raisonner formellement sur le système à concevoir
 - Ne permettent pas la preuve.



Méthodes formelles

- Une méthode formelle
 - Un langage formel :
syntaxe précise + sémantique précise (spécifications formelles)
 - Système de preuve ou de raisonnement formel
- Il existe différents types de méthodes formelles
 - La classification repose sur les spécifications formelles sous-jacentes



Classification des méthodes formelles

- Spécifications orientées *propriétés*
Description des *données* dans un langage permettant d'énoncer les propriétés attendues du système :
 - Logique classique
 - Logiques temporelles
 - Types abstraits algébriques (types de données + opérations, ex. listes, files, piles, etc.)

Classification des méthodes formelles

- Spécifications orientées *modèles*
Construction d'un système à partir d'*objets* fondamentaux pré-établis (les processus, les ambients, ...)
 - Réseaux de Petri (modélisent des systèmes concurrents, communicants et distribués)
 - Algèbre de processus : CCS, CSP, Pi-calcul, calcul des « ambients », etc.

Intérêt : développement formel

Principe :

- Transformation **systématique** des spécifications en programme
- Ces transformations utilisent des lois prédéfinies
- **Vérification** : s'assurer que le système est correct par rapport à des **propriétés**
- **Validation** : s'assurer que le système est correct par rapport aux **spécifications**
- **Raisonnement formel** : appliquer un **système formel** à une spécification



Raisonnement formel

- Plusieurs raisonnements formels :
 - Raffinement de spécification
 - Vérification des propriétés d'un système
 - Validation par vérification
 - Preuve de théorèmes
 - Analyse d'un système (représenté par une machine à états) par rapport à des propriétés (preuve de modèle ou Model checking)



Spécifier des propriétés Logiques modales temporelles

- Approche modale (LTL et CTL)
 - Succession de mondes avec relation d'accessibilité (temps + ou – implicite)
 - Temps discret (étapes)
 - Sérialité : temps infini vers futur
 - Temps linéaire pour LTL, arborescent pour CTL



Spécifier des propriétés Logiques modales temporelles

- Les formules spécifient des comportements/propriété

Ex (LTL): $G \text{ commande}(o) \rightarrow F \text{ livraison}(o)$)

Admettent typiquement plusieurs modèles

- Pour décrire/représenter un système, il faut construire un modèle de Kripke (énumérer toutes les évolutions possibles...)



Modélisation de la dynamique

- LTL / CTL
 - Permettent de spécifier/vérifier des comportement attendus
 - Pas idéal pour décrire/simuler un système dynamique
- Modèles dynamique (comme Réseau de Pétri)
 - Représenter les évolutions possibles d'un système
 - Permet simulation



Modélisation de la dynamique

- Réseau de Petri
 - Transitions discrètes et instantanées
 - Plusieurs évolutions possibles, potentiellement infinies, à partir d'un état initial.
 - Pas de temps explicite (mais des successions d'état / étapes)
⇒ Très compatible avec LTL/CTL
- Typiquement
 - Modélise/décrit le système avec Réseau de Pétri
 - Vérification formelle de spécifications écrites en LTL/CTL
 - On aura donc l'occasion de revenir sur ces logiques dans ce contexte



Modélisation de systèmes dynamiques

Réseaux de Pétri

Gauvain Bourgne (gauvain.bourgne@lip6.fr)

Amal El Fallah-Seghrouchni

Plan

- Introduction
- Syntaxe
- Modélisation
- Analyse (accessibilité / vivacité)
- Extensions
- Vérification en LTL

Plan

- **Introduction**
- **Syntaxe**
- **Modélisation**
- Analyse (accessibilité / vivacité)
- Extensions
- Vérification en LTL

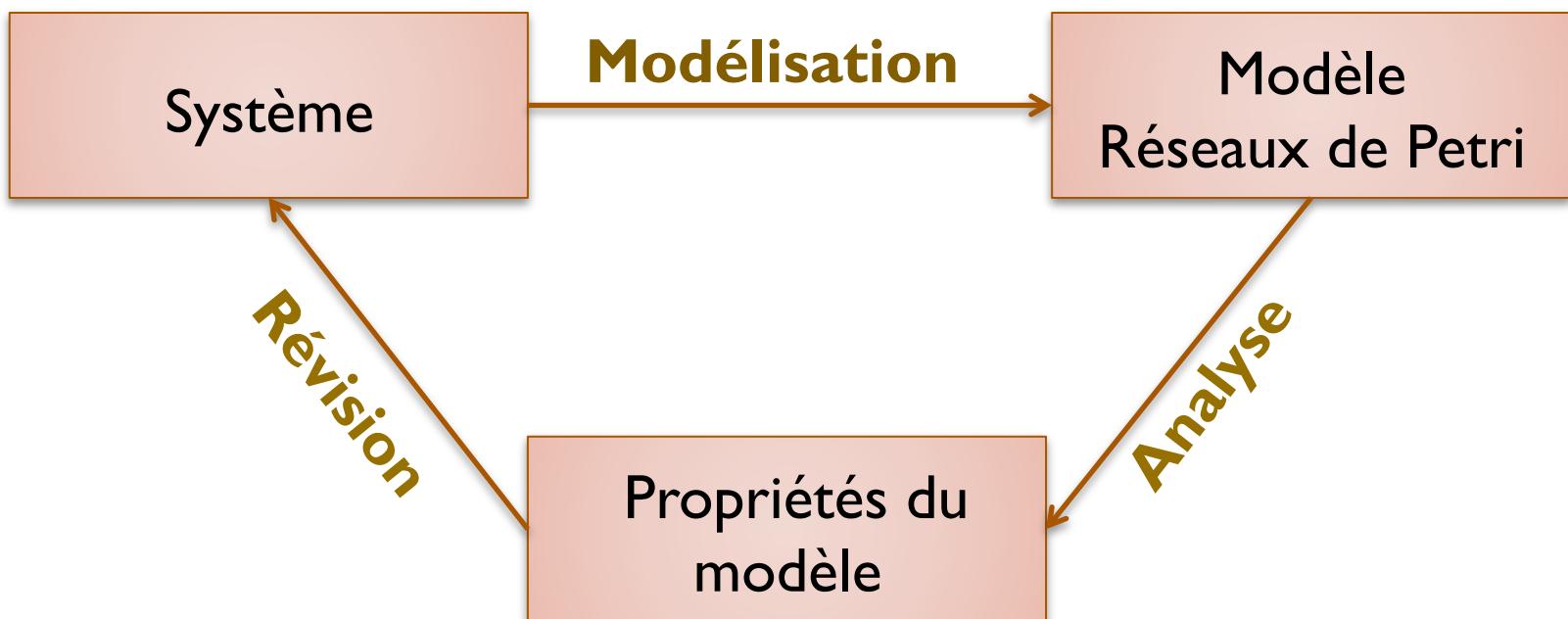


RdP : Spécifications formelles orientées modèle

- **Principe :**
 - Données -> états
 - Construction d'un modèle du système
 - Définition des propriétés du système
 - Raisonnement sur le fonctionnement du système
 - Outils utilisés : mathématiques, logiques
- **Intérêt :**
 - Modélisation
 - Etude des propriétés d'un modèle RdP
 - Vérification des propriétés

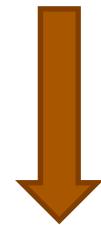
Modélisation par RdP

Quels systèmes ?
Discrets, Dynamiques et concurrents



Exemple

Boucle : le robot R ne cesse de prendre et de déposer des cubes



Définir les événements et les conditions du système.



Concepts de base : événement

- **Événements**
 - Actions se déroulant dans le système
 - Déclenchement d'un événement dépend de l'état du système
 - Un état du système peut être décrit comme un ensemble de conditions
- **Exemples**
 - un cube arrive (e1)
 - le robot saisit le cube (e2)
 - le robot dépose le cube (e3)



Concepts de base : conditions

- **Conditions ou états du système**
 - Une condition est un **prédicat** ou une **description d'un état** du système
 - Une condition est vraie ou fausse
- **Exemples**
 - Le robot est au repos (c1)
 - Un cube est en attente (c2)
 - Un cube est en cours de déplacement (c3)
 - Le cube a été déposé (c4)



Concepts de base : déclenchement

- Déclenchement
 - Les conditions nécessaires au déclenchement d'un événement sont les pré-conditions de l'événement
 - Le déclenchement d'un événement valide les post-conditions et peut invalider les pré-conditions de l'événement

Exemple

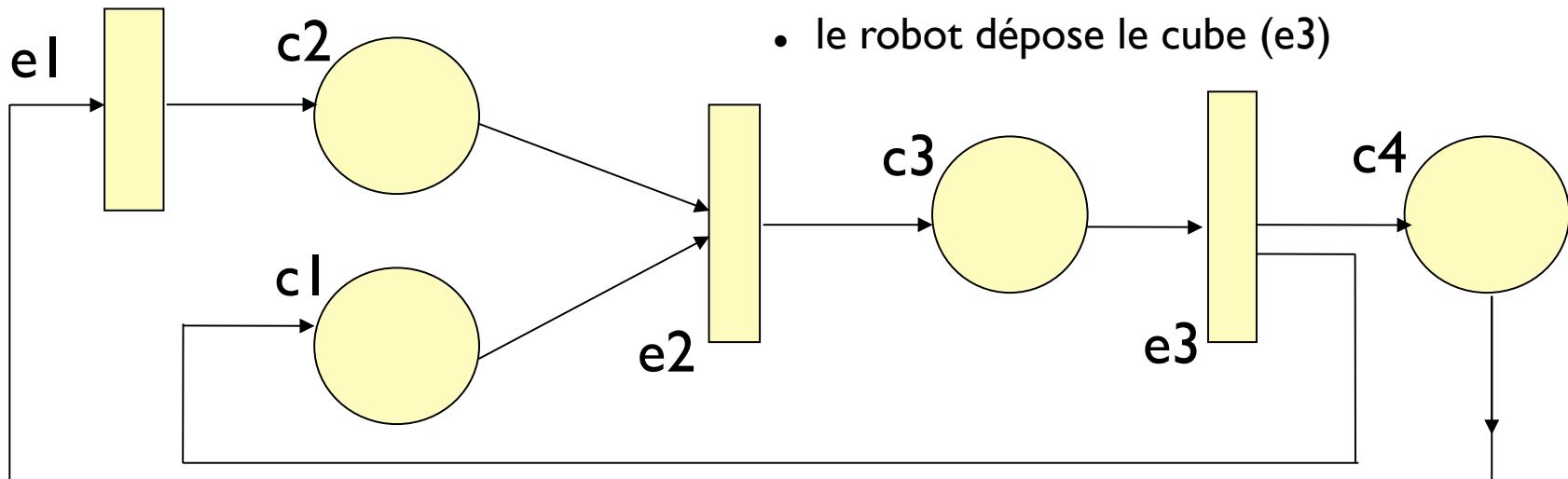
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

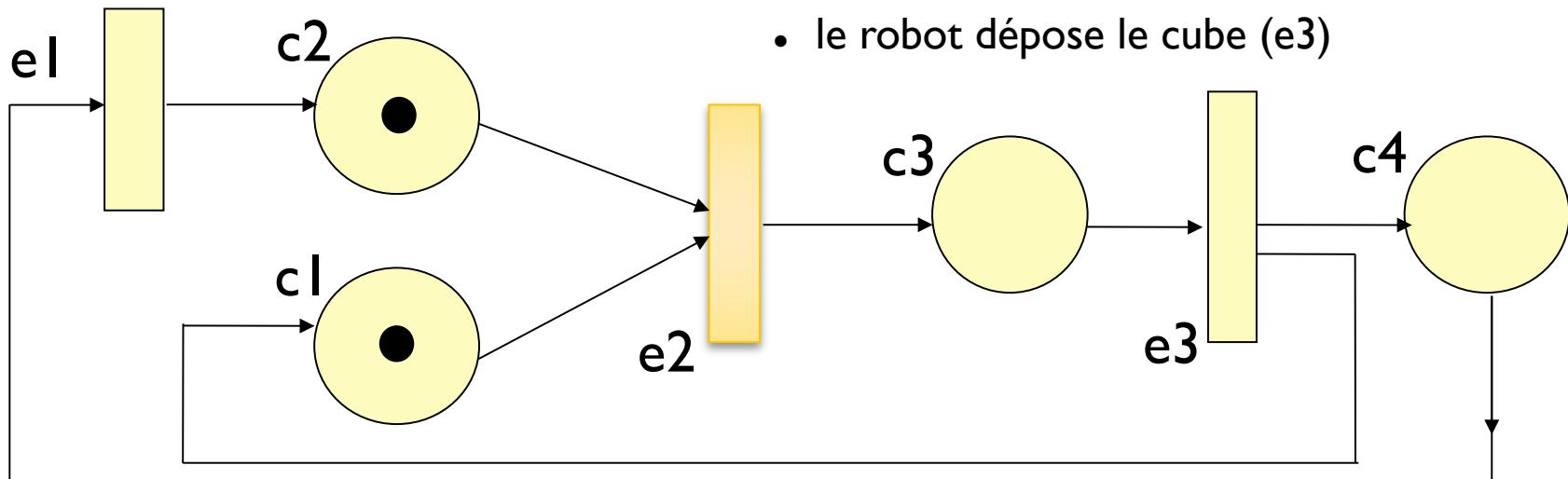
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

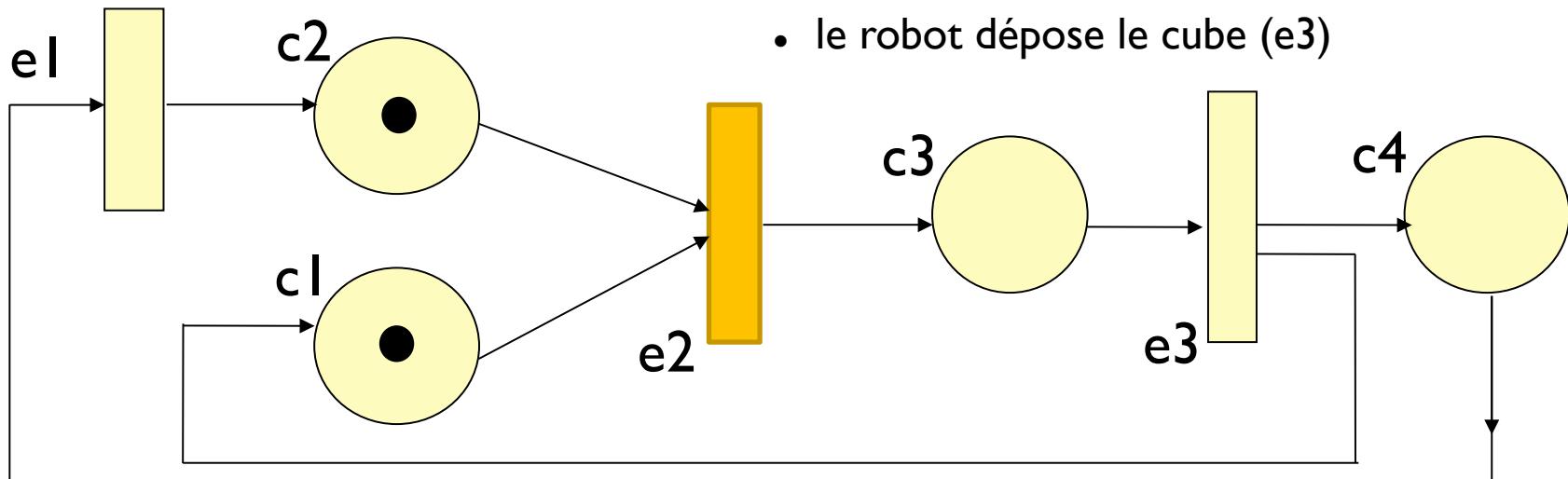
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

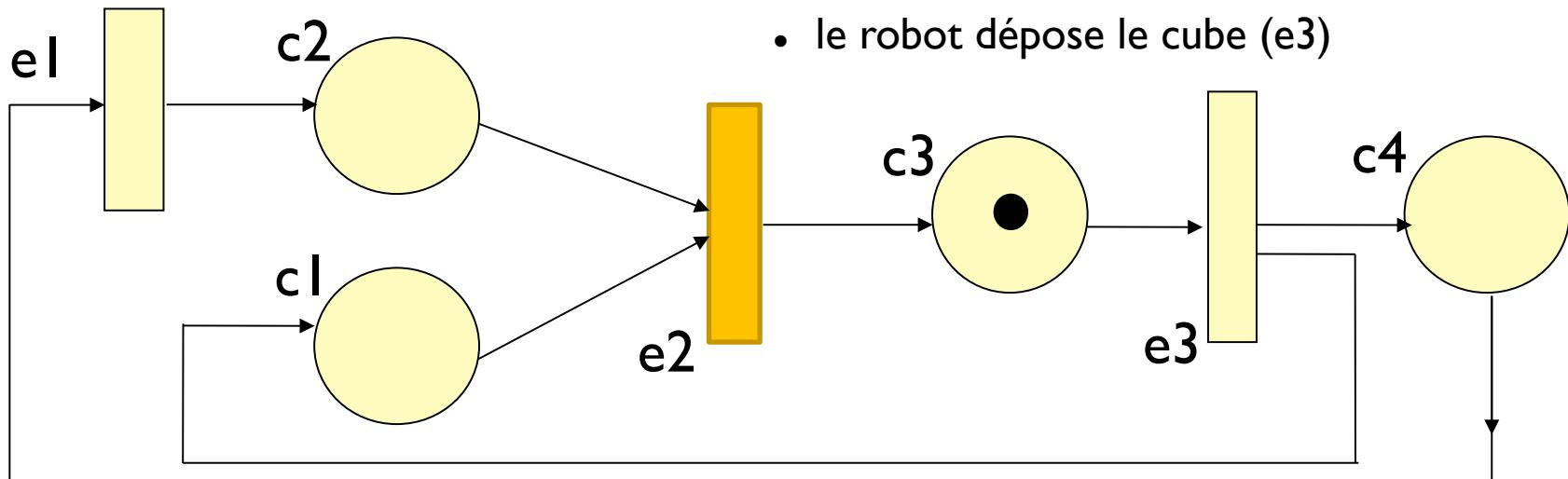
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

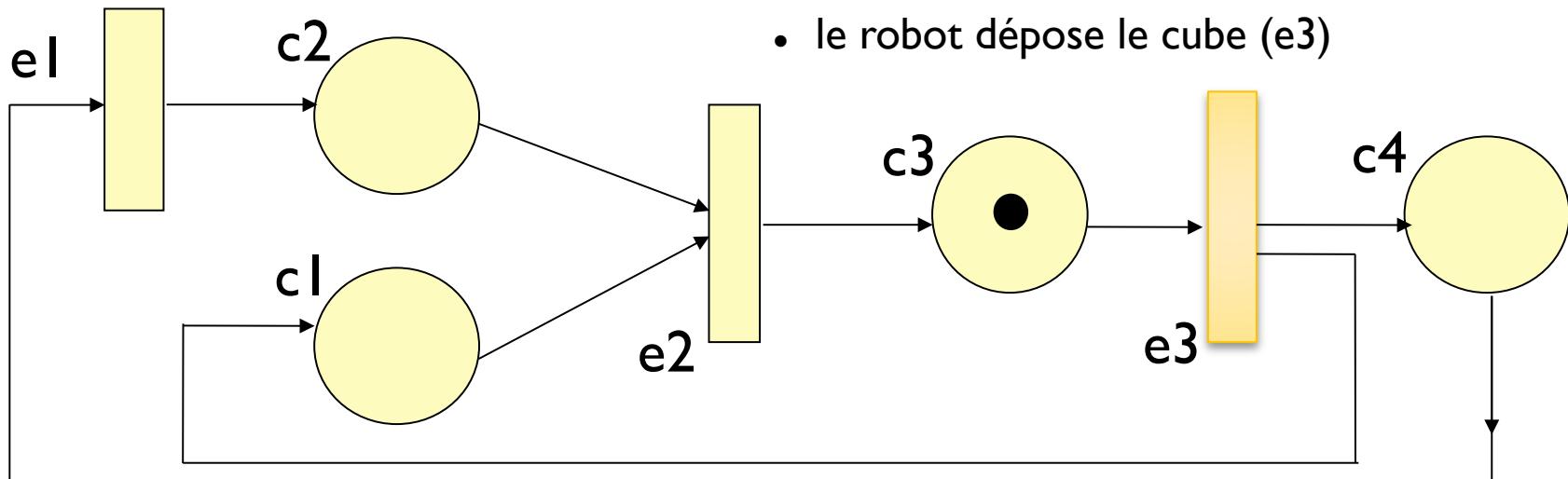
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

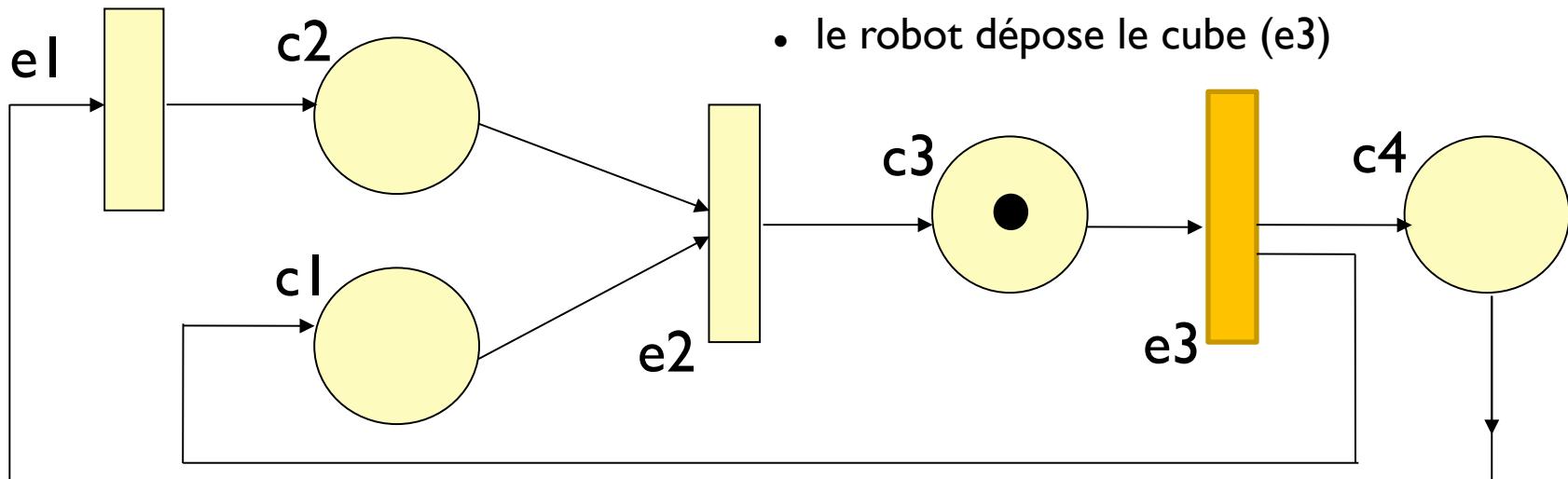
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

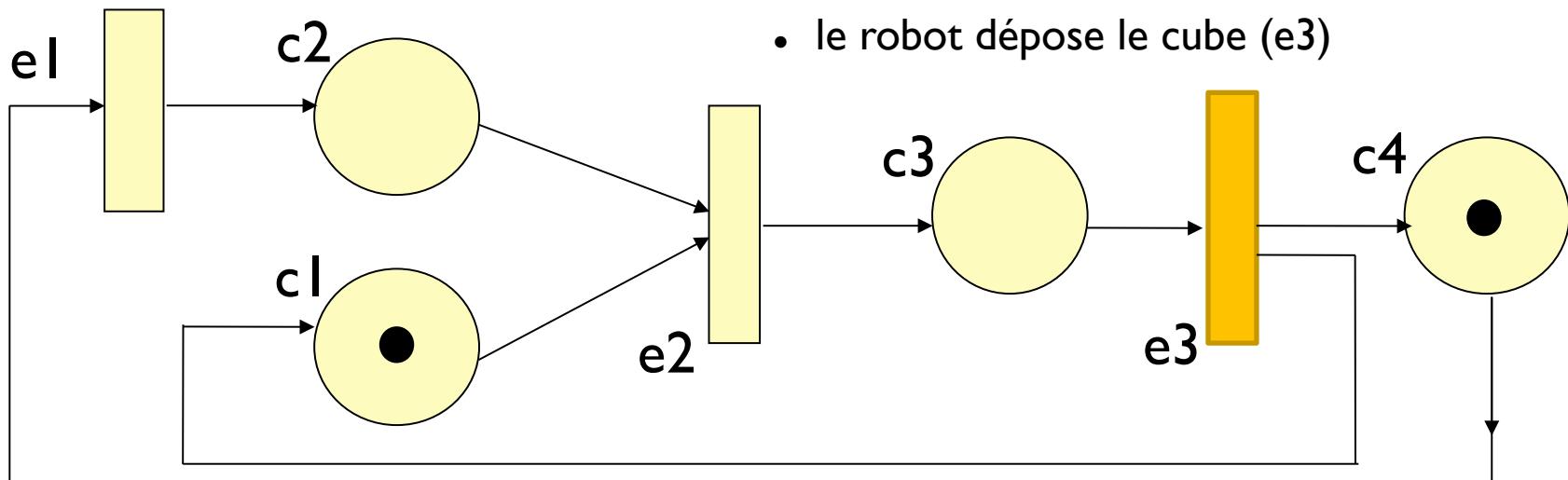
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

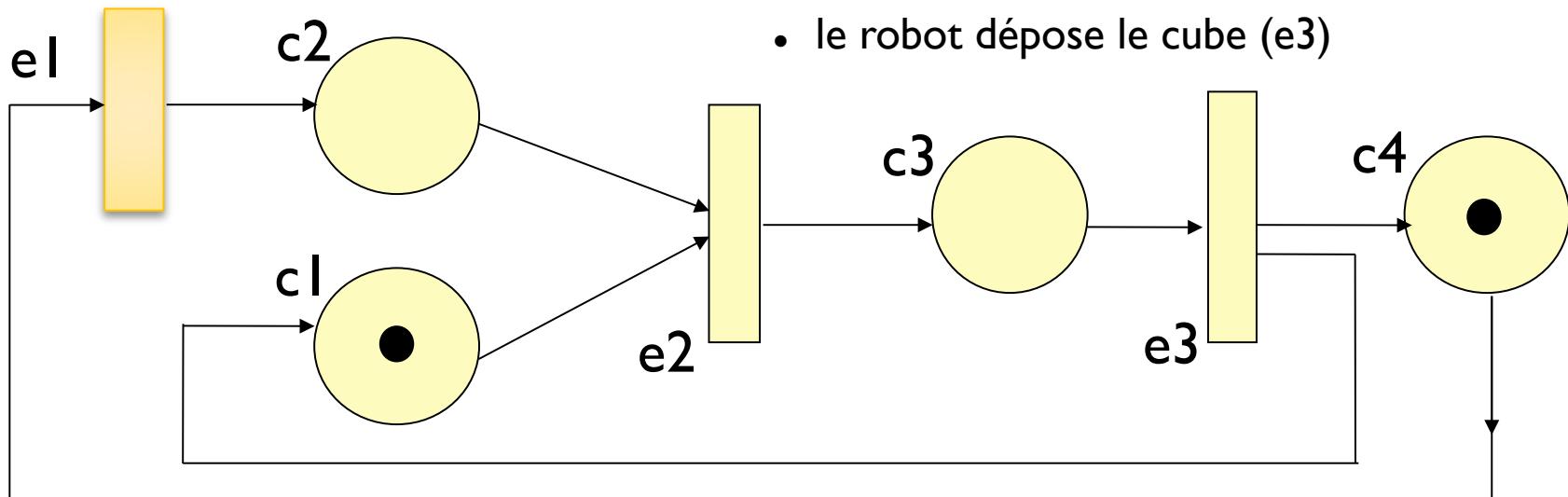
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

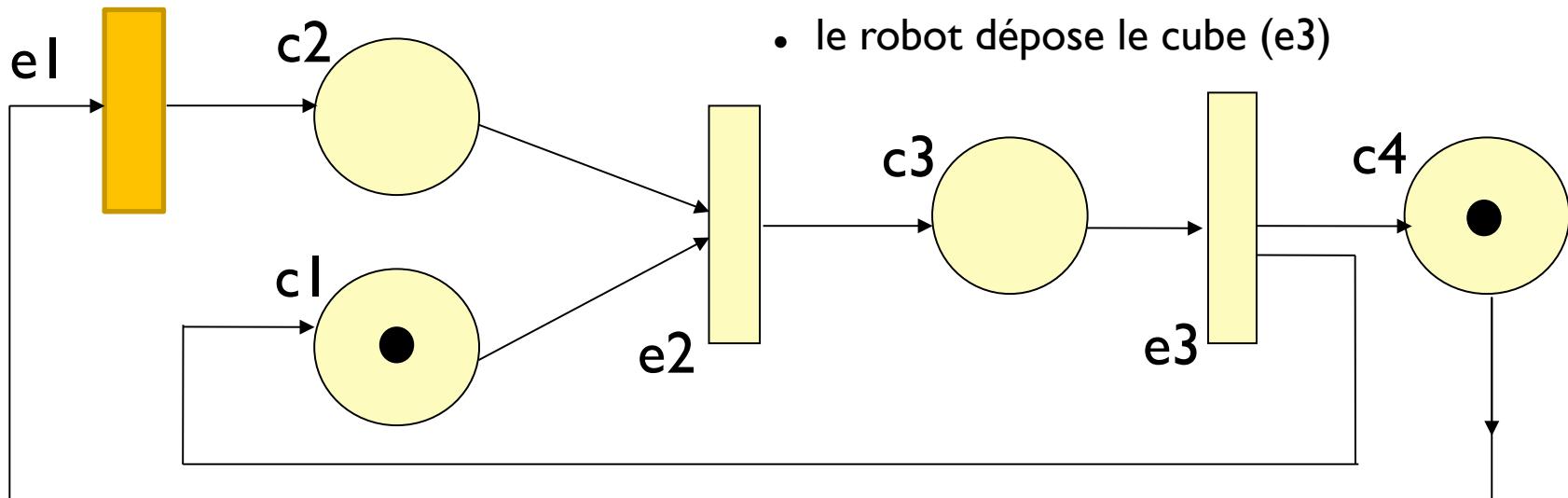
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

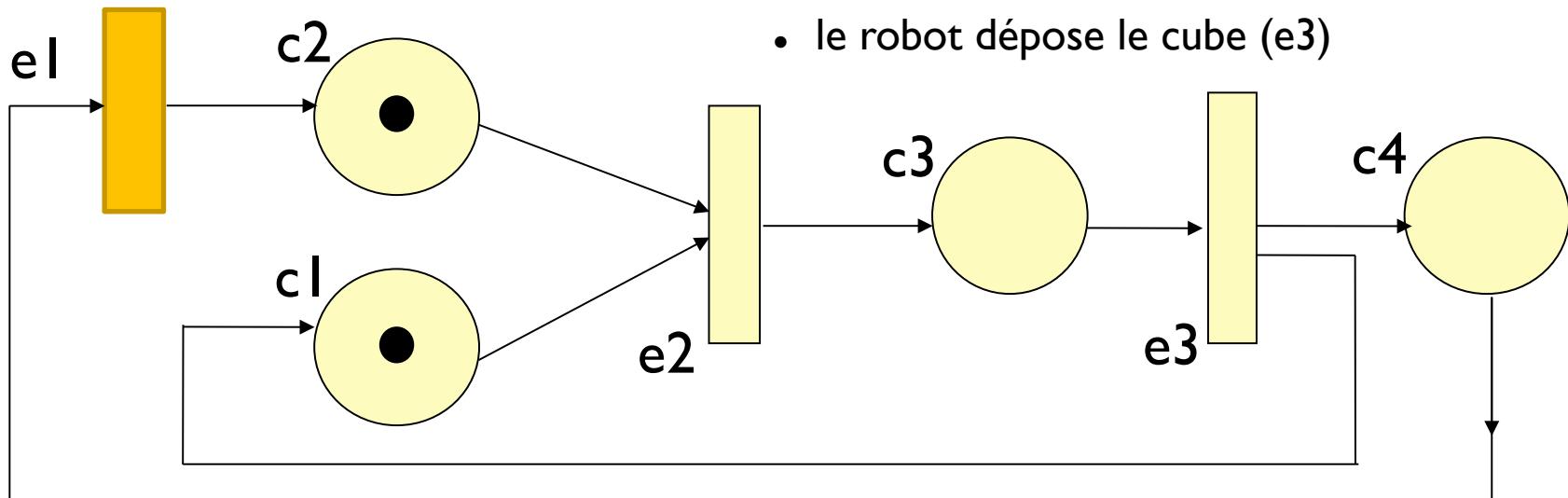
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Exemple

Modèle simulable

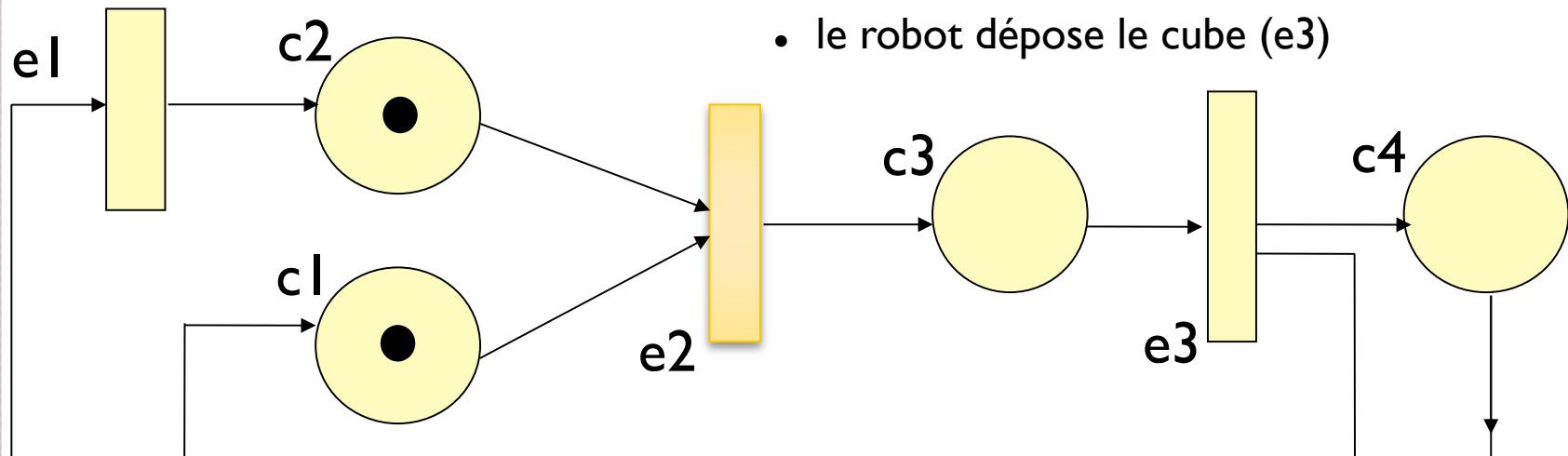
- Condition : place

Jeton =
condition vraie



- robot est au repos (c1)
- cube est en attente (c2)
- cube est en cours de déplacement (c3)
- cube a été déposé (c4)

- Evénement : transition



Boucle : le robot ne cesse de prendre et de déposer le cube

Plan

- Introduction
- **Syntaxe**
- Modélisation
- Analyse (accessibilité / vivacité)
- Extensions
- Vérification en LTL

Formalisation

Définition :

Un réseau de Petri est un quadruplet

$$R = (P, T, \text{Pré}, \text{Post})$$

où :

- P: ensemble des places
- T: ensemble des transitions
- Pré : $P \times T \rightarrow \mathbb{N}$ (Incidence avant)
- Post : $P \times T \rightarrow \mathbb{N}$ (Incidence arrière)

Formalisation

Définition :

Un réseau de Petri marqué est un quintuplet

$$R = (P, T, \text{Pré}, \text{Post}, M_0)$$

où :

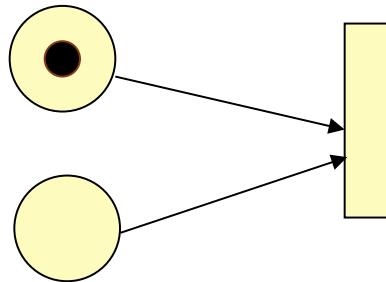
- P: ensemble des places
- T: ensemble des transitions
- Pré : $P \times T \rightarrow \mathbb{N}$ (Incidence avant)
- Post : $P \times T \rightarrow \mathbb{N}$ (Incidence arrière)
- $M_0 : P \rightarrow \mathbb{N}$ (Marquage initial)

Franchissement d'une transition

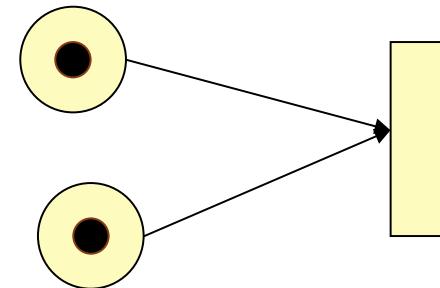
- Condition

Il faut assez de jetons dans toutes les places entrantes

La transition est dite **franchissable**



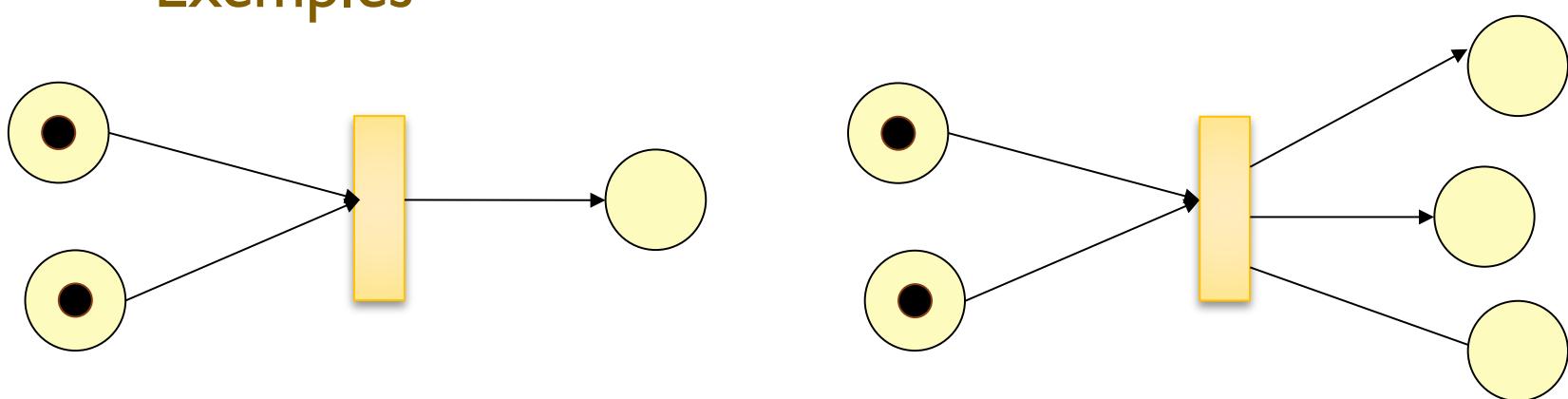
Non franchissable



Franchissable

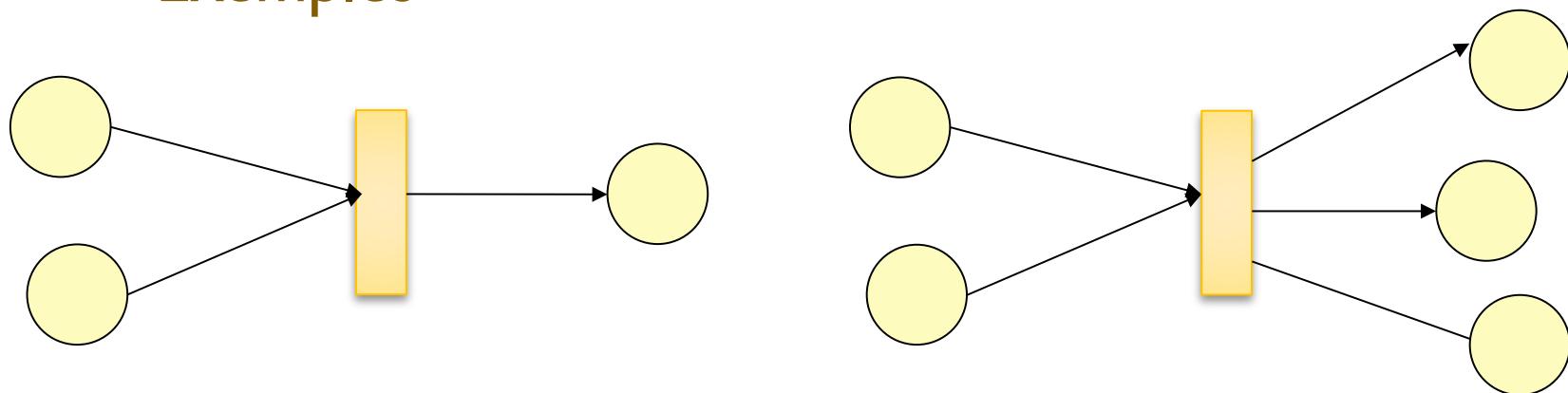
Franchissement d'une transition

- Déroulé d'un **franchissement** (aussi appelé **tir** ou **déclenchement**)
 1. Les jetons entrants sont consommés au début du franchissement (1 par arc ou plus si valuation)
 2. l'événement correspondant à la transition a lieu
 3. Les jetons sortant sont produits à la fin du franchissement (1 par arc ou plus si valuation)
- Exemples



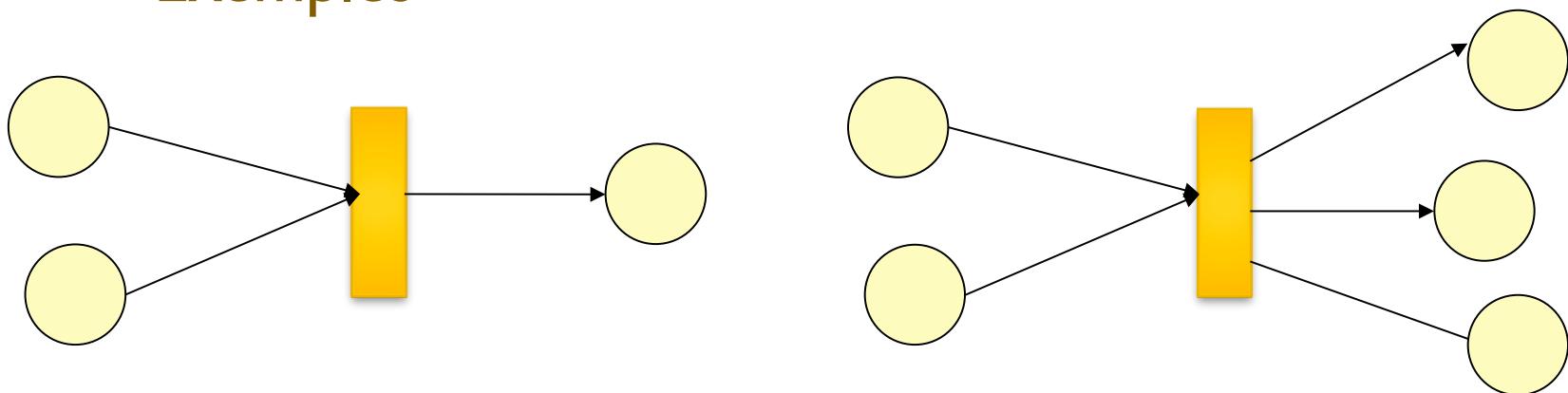
Franchissement d'une transition

- Déroulé d'un **franchissement** (aussi appelé **tir** ou **déclenchement**)
 1. Les jetons entrants sont consommés au début du franchissement (1 par arc ou plus si valuation)
 2. l'événement correspondant à la transition a lieu
 3. Les jetons sortant sont produits à la fin du franchissement (1 par arc ou plus si valuation)
- Exemples



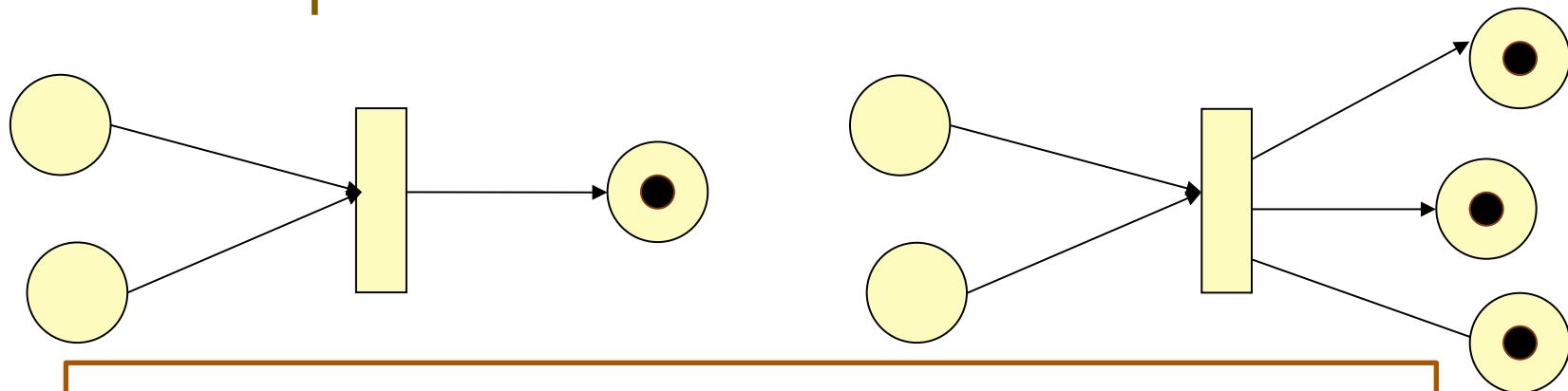
Franchissement d'une transition

- Déroulé d'un **franchissement** (aussi appelé **tir** ou **déclenchement**)
 1. Les jetons entrants sont consommés au début du franchissement (1 par arc ou plus si valuation)
 2. **l'événement correspondant à la transition a lieu**
 3. Les jetons sortant sont produits à la fin du franchissement (1 par arc ou plus si valuation)
- Exemples



Franchissement d'une transition

- Déroulé d'un **franchissement** (aussi appelé **tir** ou **déclenchement**)
 1. Les jetons entrants sont consommés au début du franchissement (1 par arc ou plus si valuation)
 2. l'événement correspondant à la transition a lieu
 3. **Les jetons sortant sont produits à la fin du franchissement** (1 par arc ou plus si valuation)
- Exemples



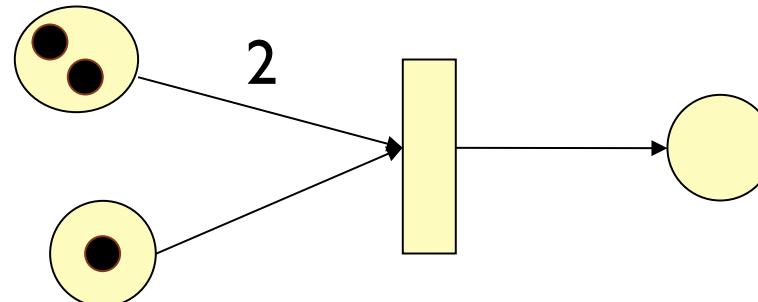
Le nombre de jetons produits et celui des jetons consommés sont indépendants !!

Valuation des arcs

- Signification

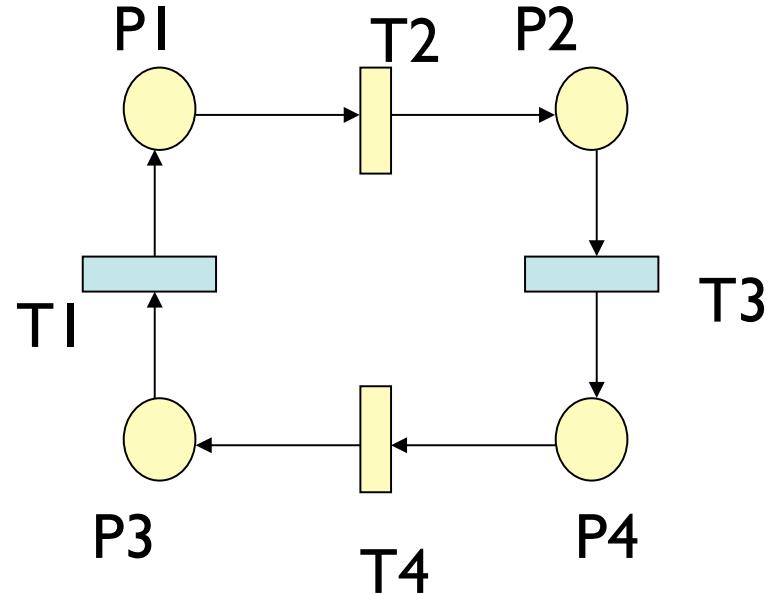
- Pré : nombre de jetons consommé
- Post : nombre de jetons produits

Par défaut, la valuation est de 1



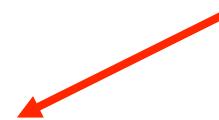
Exemple : 2 cubes et un robot qui va déposer l'un sur l'autre

Matrice d'incidence avant (Matrice d'entrée)



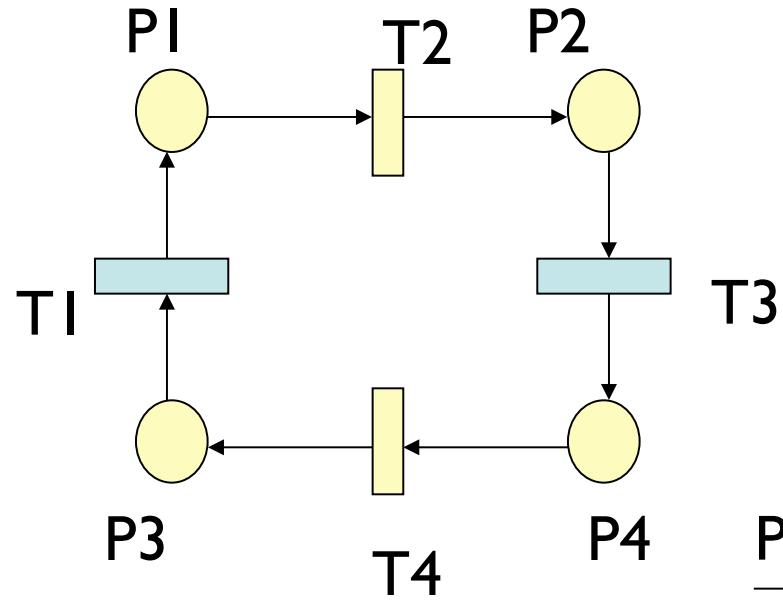
Pré : $P \times T \rightarrow IN$

Valuations des arcs entrants



Pré	T1	T2	T3	T4
P1		1		
P2			1	
P3	1			
P4				1

Matrice d'incidence arrière (Matrice de sortie)



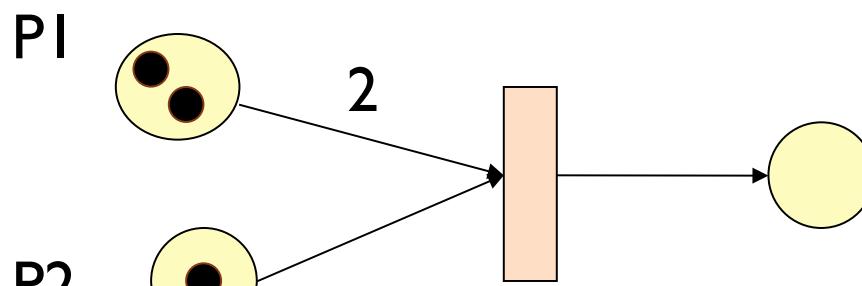
Post : $P \times T \rightarrow \mathbb{IN}$

Valuations des arcs sortants

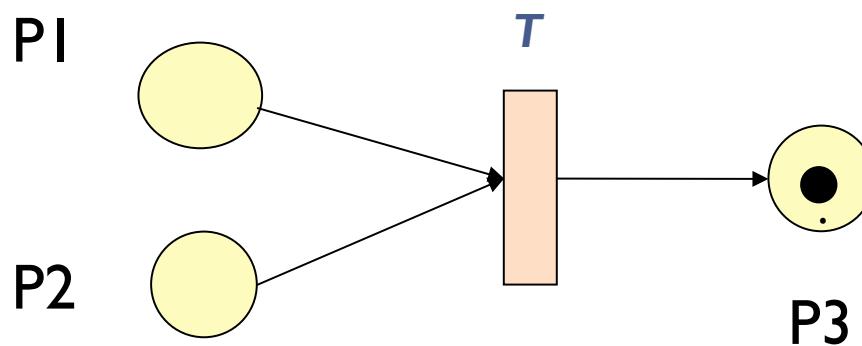
Post	T1	T2	T3	T4
P1	1			
P2			1	
P3				1
P4				1

Marquage du réseau

- Le marquage d'un RdP est son état $M : P \rightarrow \mathbb{N}$
- Il donne pour chaque place son marquage, i.e. le nombre de jetons qu'elle contient.
- Le marquage initial d'un réseau est M_0

 M_0 $P_1 \\ P_2 \\ P_3$

2
1
0



Après le franchissement de T

 M_1 $P_1 \\ P_2 \\ P_3$

0
0
1



Sémantique

- Une transition t est **franchissable** ssi :

$$\forall p \in P, M(p) \geq Pre(p, t)$$

- Le franchissement de t à partir du marquage M produit le marquage M' tel que :

$$\forall p \in P, M'(p) = M(p) - Pre(p, t) + Post(p, t)$$

Sémantique (version matricielle)

- T : vecteur représentant la transition t
e.g. $T_2 = [0, 1, 0, 0]$ $\bar{T}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$
- Une transition t est **franchissable** ssi :

$$M \geq \text{Pré.} \bar{T}$$

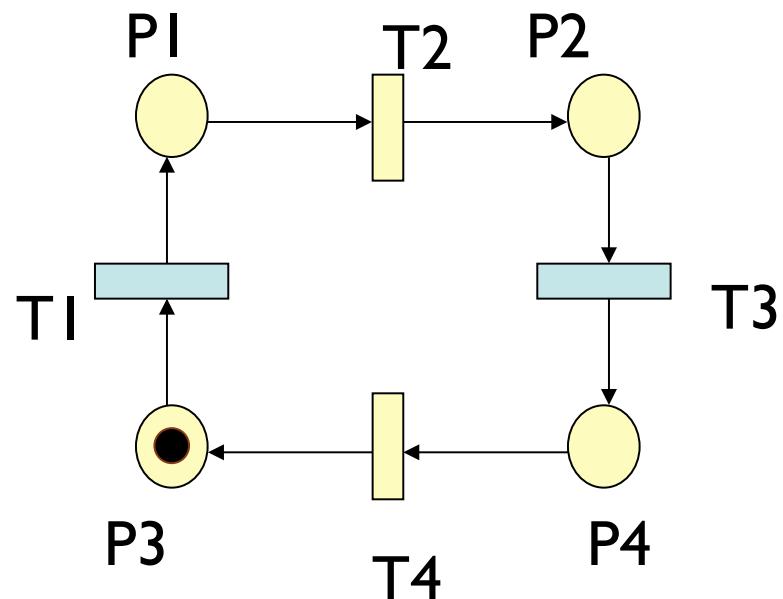
- Le franchissement de t à partir du marquage M produit le marquage M' tel que :

$$M' = M + (-\text{Pré} + \text{Post}).\bar{T}$$

Avec **Matrice d'incidence** $C = \text{Post} - \text{Pré}$:

$$M' = M + C.\bar{T}$$

Exemple



Pré =

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Post =

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

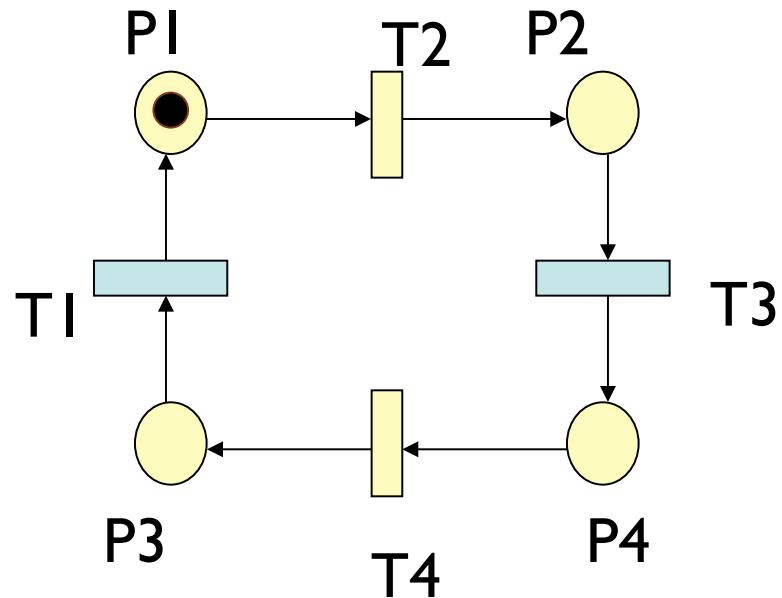
$C = \text{Post} - \text{Pré} =$

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Franchissement de $t_1 - T_1 = [1 \ 0 \ 0 \ 0]$

$$M_1 = M_0 + C \cdot \bar{T}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Exemple



Pré =

$$\begin{bmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

Post =

$$\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & I & 0 \end{bmatrix}$$

$C = \text{Post} - \text{Pré} =$

$$\begin{bmatrix} I & -I & 0 & 0 \\ 0 & I & -I & 0 \\ -I & 0 & 0 & I \\ 0 & 0 & I & -I \end{bmatrix}$$

Franchissement de $t_1 - T_1 = [I \ 0 \ 0 \ 0]$

$$M_1 = M_0 + C \cdot \bar{T}_1 = \begin{bmatrix} 0 \\ 0 \\ -I \\ 0 \end{bmatrix} + \begin{bmatrix} I \\ 0 \\ -I \\ 0 \end{bmatrix} = \begin{bmatrix} I \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Sémantique

- Marquage accessible

Le marquage M' est **accessible** à partir de M ssi il existe une séquence $s=(t_1, \dots, t_n)$ de transitions menant de M à M' (ie *pour chaque i entre 1 et n, t_i est franchissable dans M_{i-1} et mène à M_i avec $M_0=M$ et $M_n=M'$*)

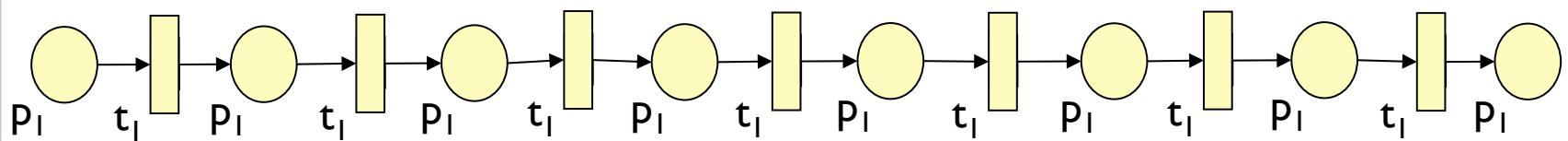
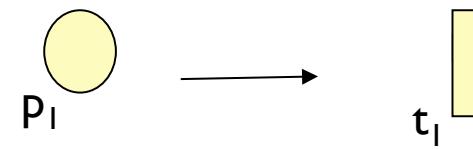
$$M=M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} \dots M_{n-1} \xrightarrow{t_n} M_n=M'$$

soit

$$M \xrightarrow{s} M'$$

Plan

- Introduction
- Syntaxe
- Modélisation
- Analyse (accessibilité / vivacité)
- Extensions
- Vérification en LTL

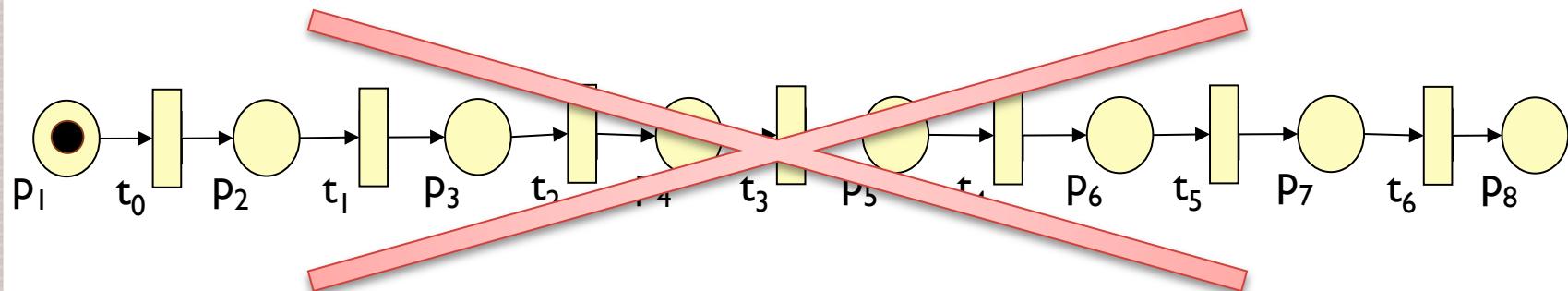


Exemple : Recette

- Mousse au chocolat

- [0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé
1. Séparer les blancs et les jaunes des œufs.
 2. Faire ramollir le chocolat au bain-marie.
 3. Hors du feu, incorporer les jaunes et le sucre.
 4. Battre les blancs en neige ferme.
 5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
 6. Verser dans des verrines et mettre au frais pendant 2h.

Lecture séquentielle :



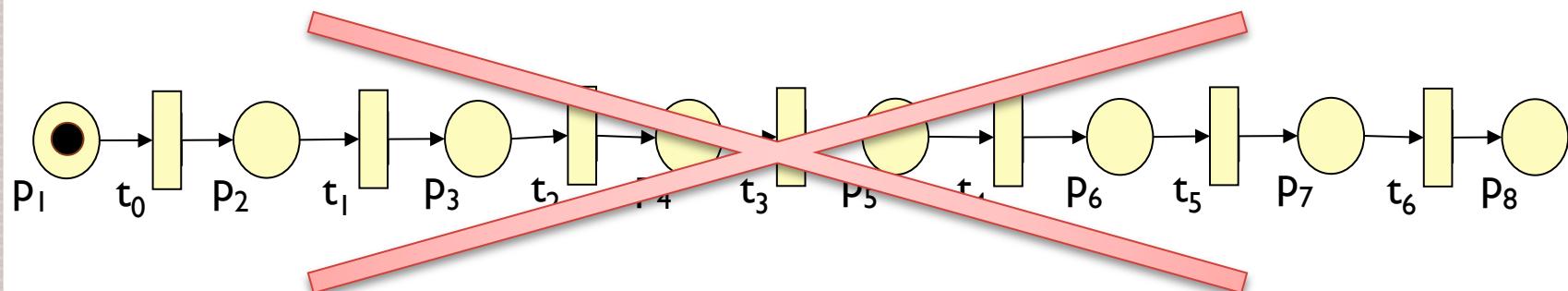
Exemple : Recette

- Mousse au chocolat

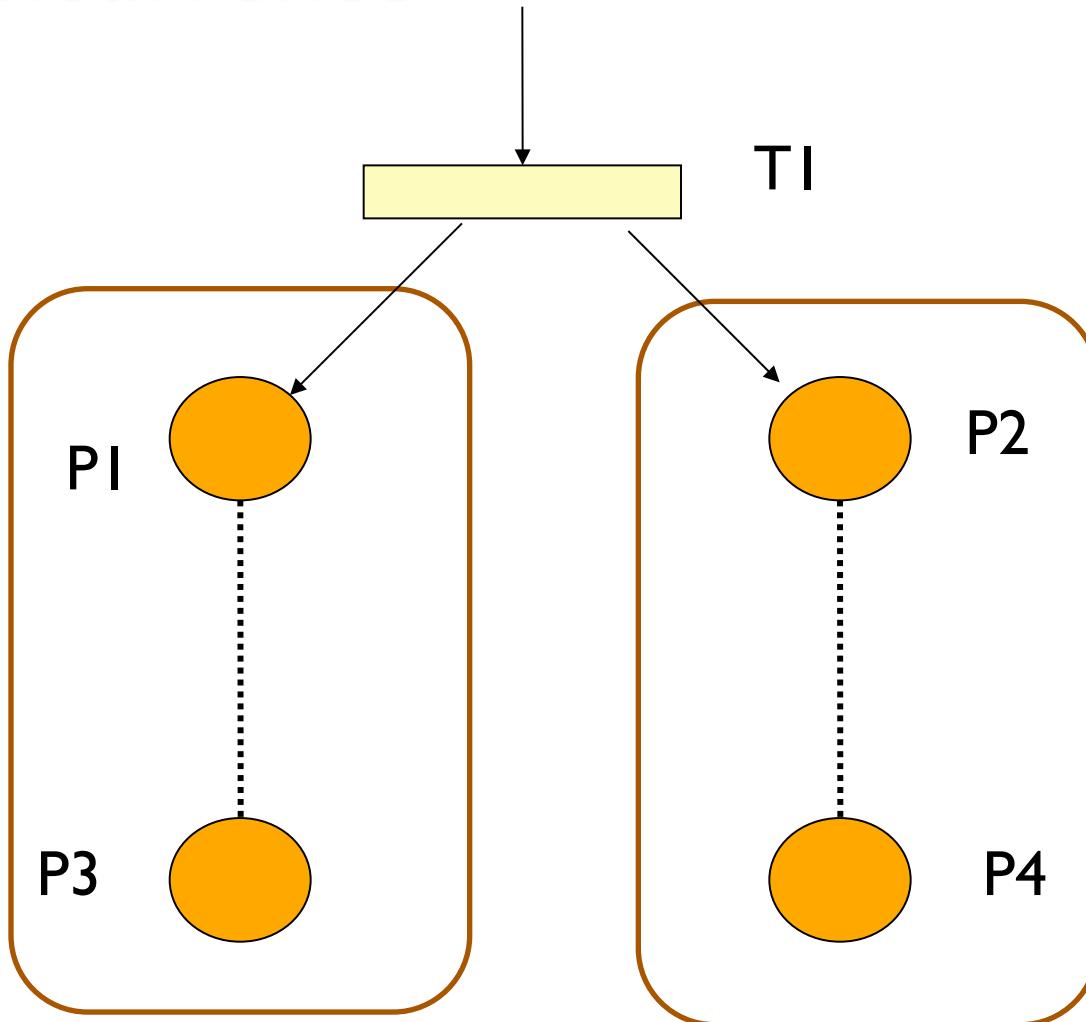
- [0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé
1. Séparer les blancs et les jaunes des œufs.
 2. Faire ramollir le chocolat au bain-marie.
 3. Hors du feu, incorporer les jaunes et le sucre.
 4. Battre les blancs en neige ferme.
 5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
 6. Verser dans des verrines et mettre au frais pendant 2h.

Après 0, on peut **paralléliser** la confection des blancs en neige (1,4) et la préparation du mélange avec le chocolat (2,3).

Lecture séquentielle :



Structure : Parallélisme ou concurrence



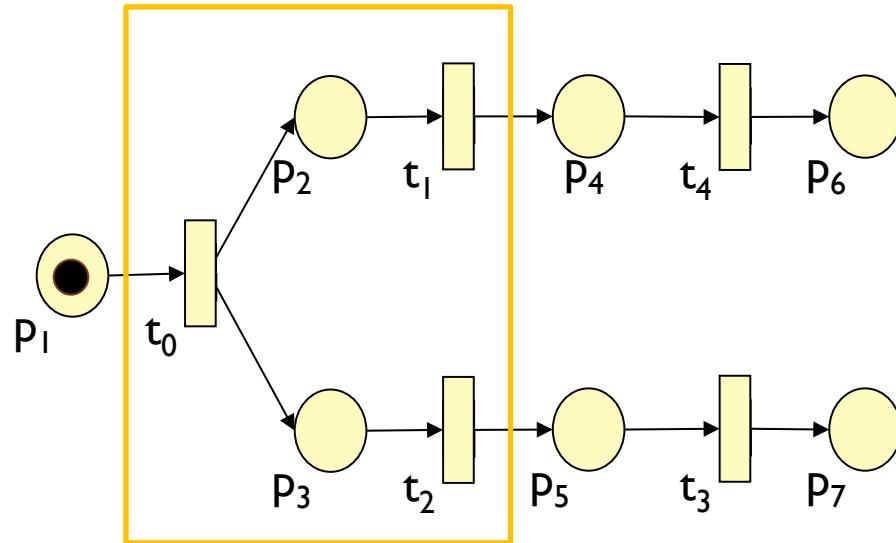
Exemple : Recette

- Mousse au chocolat

[0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé

1. Séparer les blancs et les jaunes des œufs.
2. Faire ramollir le chocolat au bain-marie.
3. Hors du feu, incorporer les jaunes et le sucre.
4. Battre les blancs en neige ferme.
5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
6. Verser dans des verrines et mettre au frais pendant 2h.

Après 0, on peut paralléliser la confection des blancs en neige (1,4) et la préparation du mélange avec le chocolat (2,3).

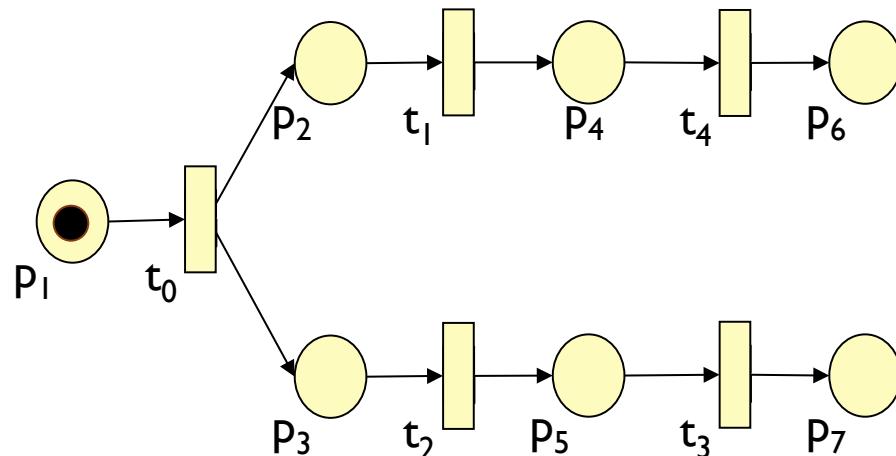


Exemple : Recette

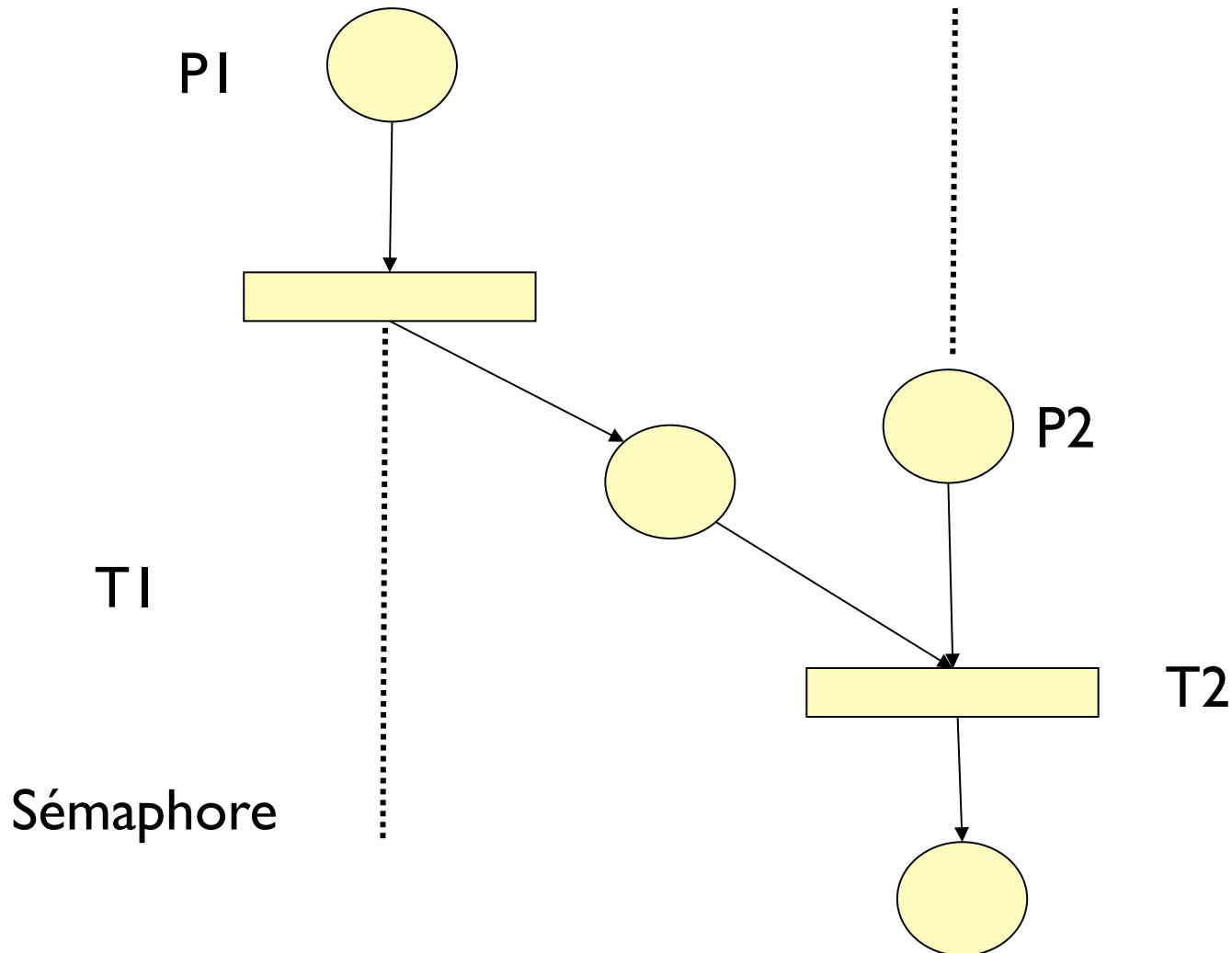
- Mousse au chocolat

- [0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé
1. Séparer les blancs et les jaunes des œufs.
 2. Faire ramollir le chocolat au bain-marie.
 3. Hors du feu, incorporer les jaunes et le sucre.
 4. Battre les blancs en neige ferme.
 5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
 6. Verser dans des verrines et mettre au frais pendant 2h.

Mais, pb de **synchronisation** : on ne peut incorporer les jaunes (3) tant que l'on ne les a pas séparés des blancs (1).



Struture : Synchronisation par séaphore

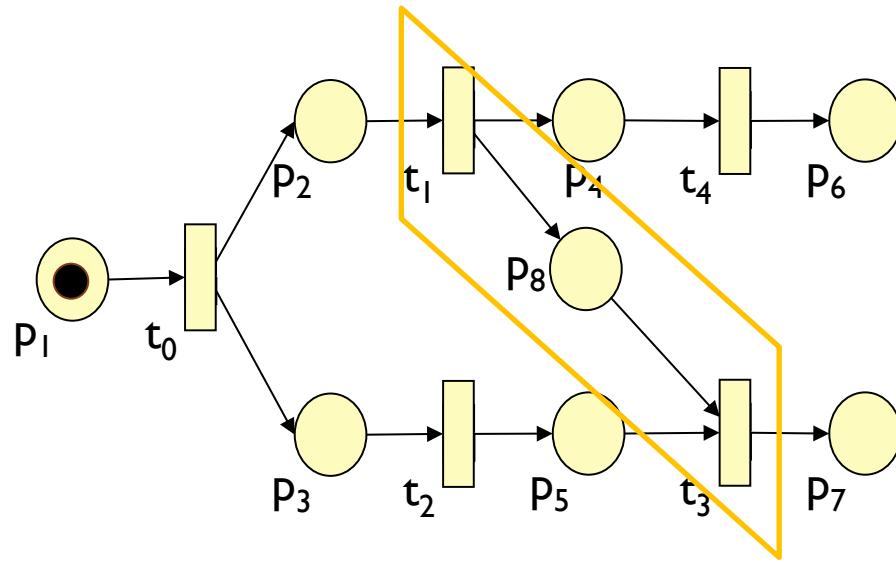


Exemple : Recette

- Mousse au chocolat

- [0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé
1. Séparer les blancs et les jaunes des œufs.
 2. Faire ramollir le chocolat au bain-marie.
 3. Hors du feu, incorporer les jaunes et le sucre.
 4. Battre les blancs en neige ferme.
 5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
 6. Verser dans des verrines et mettre au frais pendant 2h.

Mais, pb de **synchronisation** : on ne peut incorporer les jaunes (3) tant que l'on ne les a pas séparés des blancs (1).

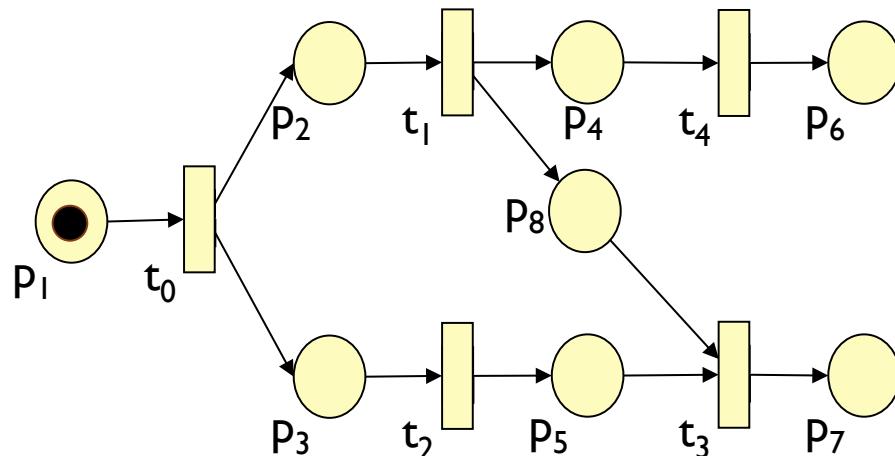


Exemple : Recette

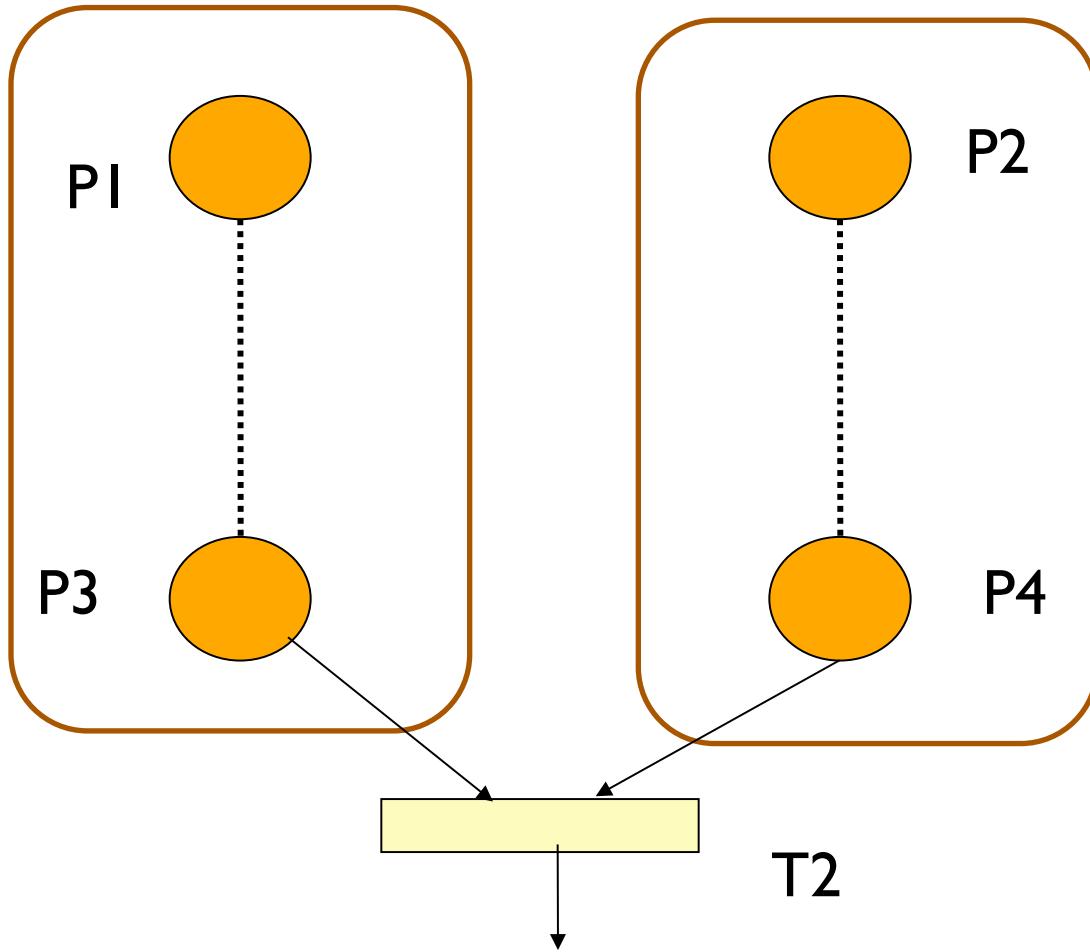
- Mousse au chocolat

- [0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé
1. Séparer les blancs et les jaunes des œufs.
 2. Faire ramollir le chocolat au bain-marie.
 3. Hors du feu, incorporer les jaunes et le sucre.
 4. Battre les blancs en neige ferme.
 5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
 6. Verser dans des verrines et mettre au frais pendant 2h.

Autre pb de **synchronisation** : il faut avoir fini les deux processus (2,3) et (1,4) pour pouvoir ajouter les blancs au mélange (5).



Structure : synchronisation par fusion de processus parallèles



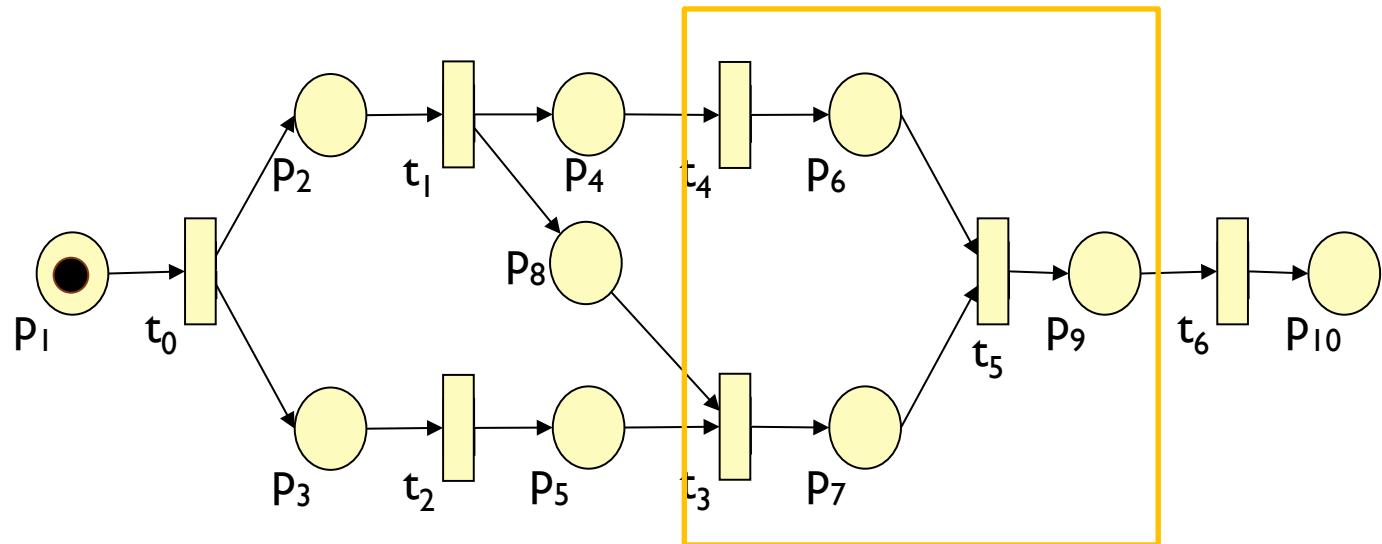
Exemple : Recette

- Mousse au chocolat

[0]. Ingrédients : 3 œufs, 100 g de chocolat au lait, 1 sachet de sucre vanillé

1. Séparer les blancs et les jaunes des œufs.
2. Faire ramollir le chocolat au bain-marie.
3. Hors du feu, incorporer les jaunes et le sucre.
4. Battre les blancs en neige ferme.
5. Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
6. Verser dans des verrines et mettre au frais pendant 2h.

Autre pb de **synchronisation** : il faut avoir fini les deux processus (2,3) et (1,4) pour pouvoir ajouter les blancs au mélange (5).



Exemple : Recette

- **Transitions**

t_0 : Sortir les ingrédients

t_1 : Séparer les blancs et les jaunes des œufs.

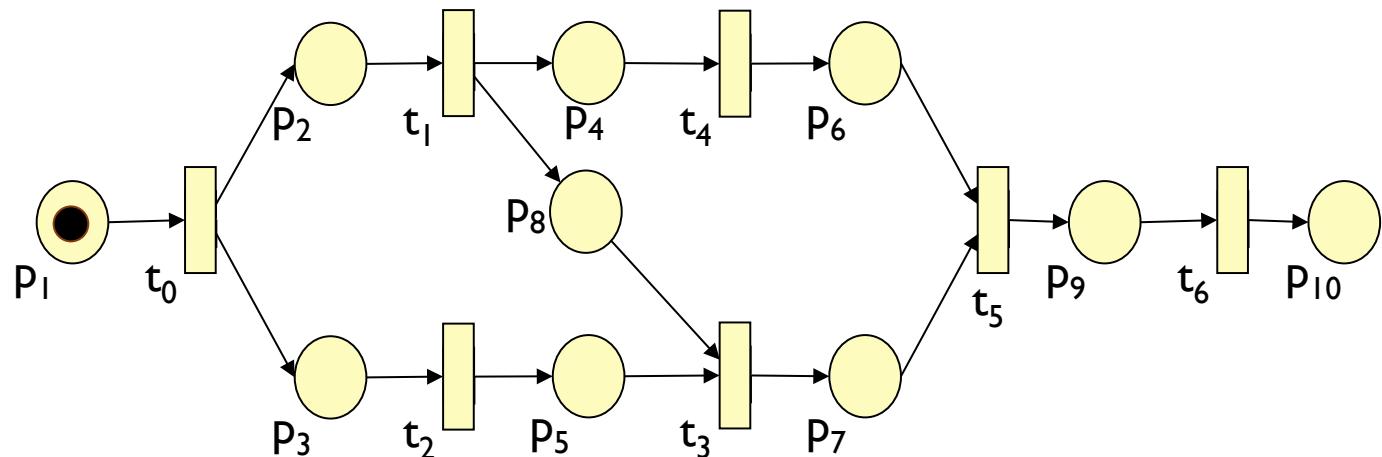
t_2 : Faire ramollir le chocolat au bain-marie.

t_3 : Incorporer les jaunes et le sucre au chocolat

t_4 : Battre les blancs en neige ferme.

t_5 : Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.

t_6 : Verser dans des verrines et mettre au frais pendant 2h.

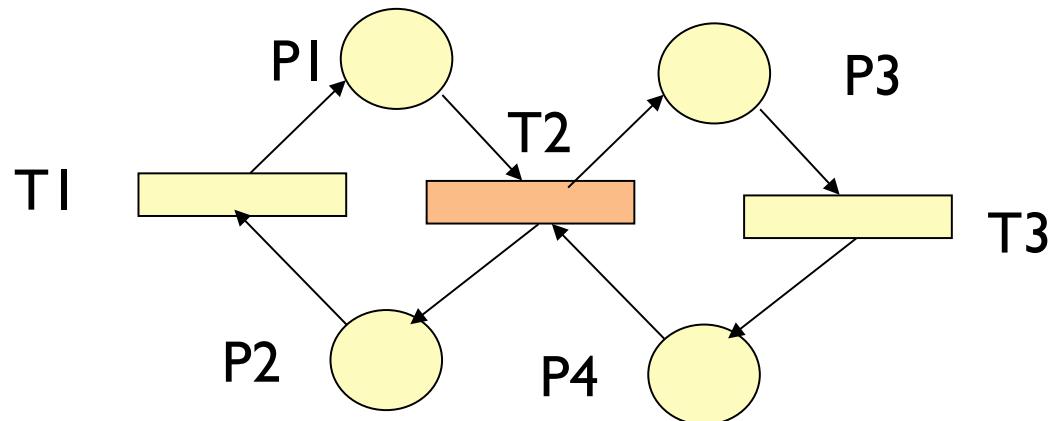
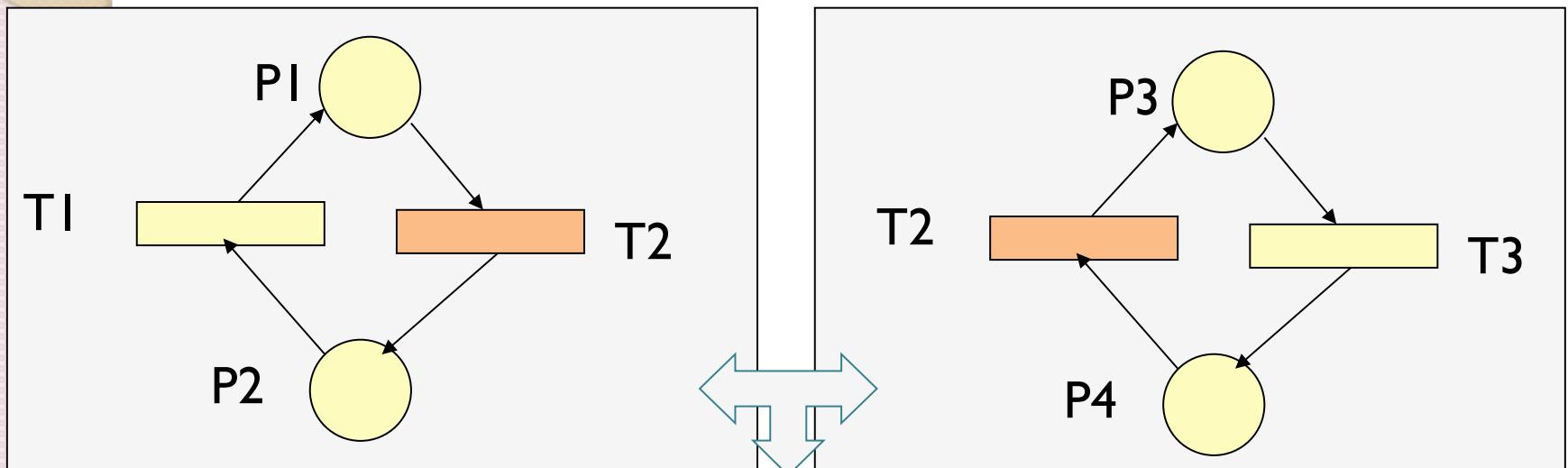


Représentation : transitions

- Une **transition** correspond à un **événement**.
 - Action interne du système (décision du système qui affecte le système)
Ex: les actions de la recette
 - Evénement externe (déclenché par extérieur, affecte le système)
Ex: entrée d'un client, arrivée d'une requête, timer...
 - Action externe (décision du système, affecte l'extérieur)
Ex: Envoi d'un message ou d'une requête, ...

Composition

Modèle offrant la composition (ex. transition commune)



Exemple : Recette

- Places

P_1 :

P_2 :

P_3 :

P_4 :

P_5 :

P_6 :

P_7 :

P_8 :

P_9 :

P_{10} :

- Transitions

t_0 : Sortir les ingrédients

t_1 : Séparer les blancs et les jaunes des œufs.

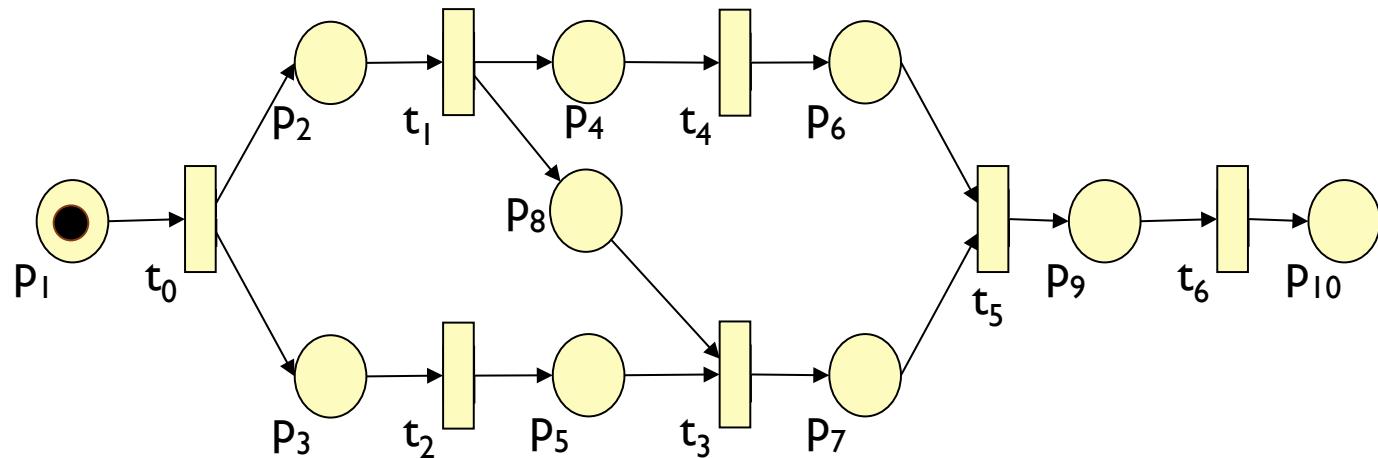
t_2 : Faire ramollir le chocolat au bain-marie.

t_3 : Incorporer les jaunes et le sucre au chocolat

t_4 : Battre les blancs en neige ferme.

t_5 : Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.

t_6 : Verser dans des verrines et mettre au frais pendant 2h.

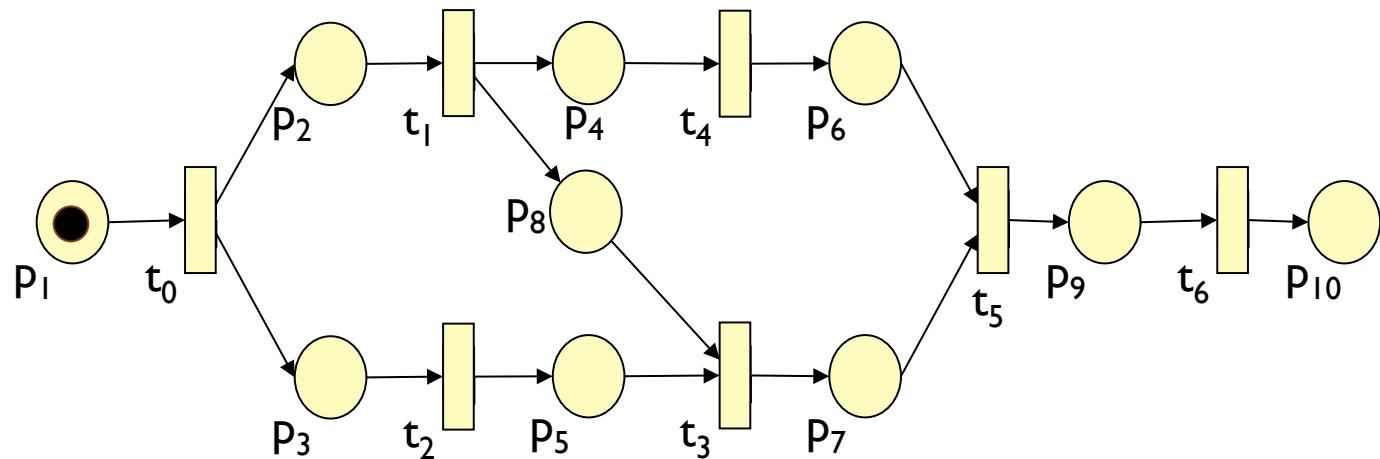


Représentation : places et jetons

- Une **place** correspond à une **condition**
 - Étape d'un processus (tâche faite ou à faire)
Ex: *mousse à faire, œufs à séparer, mousse faite*
 - État d'un élément du système
Ex: *libre/occupé, ouvert/fermé, feu rouge/orange/vert*
 - Ressources (dans un état donné)
Ex: *œufs entiers, blancs d'œuf, blancs en neige*
- Les **jetons** correspondent à des **compteurs**
Les jetons sont indiscernables
Leur sens dépend de la place dans laquelle ils sont.

Exemple : Recette

- Places (vision processus)
 - P_1 : mousse à faire (prêt à commencer)
 - P_2 : œufs à séparer (prêt à t_1)
 - P_3 : chocolat à ramollir (prêt à t_2)
 - P_4 : blancs à battre en neige (prêt à t_4)
 - P_5 : chocolat ramolli (prêt à t_3)
 - P_6 : blancs battus en neige (t_4 fini)
 - P_7 : mélange fait (t_3 fini)
 - P_8 : jaunes séparés des blancs (t_1 fini)
 - P_9 : mousse à mettre au frais (prêt à t_6)
 - P_{10} : mousse terminée (processus fini)
- Transitions
 - t_0 : Sortir les ingrédients
 - t_1 : Séparer les blancs et les jaunes des œufs.
 - t_2 : Faire ramollir le chocolat au bain-marie.
 - t_3 : Incorporer les jaunes et le sucre au chocolat
 - t_4 : Battre les blancs en neige ferme.
 - t_5 : Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
 - t_6 : Verser dans des verrines et mettre au frais pendant 2h.



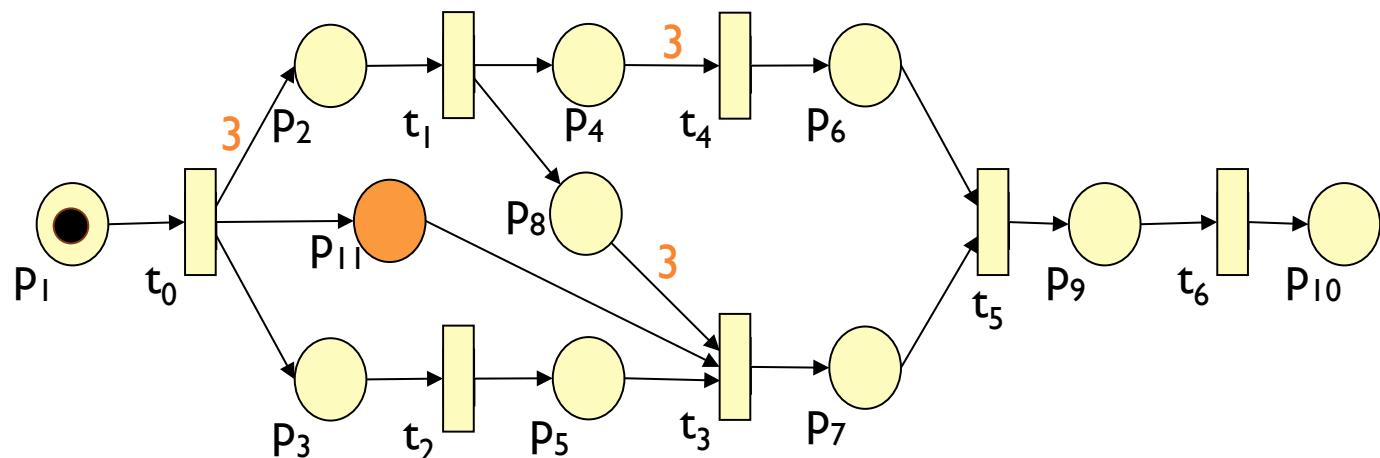
Exemple : Recette

- Places (vision **ressources**)

- P₁ : demande de mousse
- P₂ : œuf entier
- P₃ : tablette de 100g de chocolat
- P₄ : blanc d'œuf
- P₅ : chocolat ramolli
- P₆ : blancs en neige (de 3 œufs)
- P₇ : mélange chocolat, sucre et jaunes
- P₈ : jaune d'œuf
- P₉ : mélange complet
- P₁₀ : mousse terminée
- P₁₁ : **sachet de sucre vanillé**

- Transitions

- t₀ : Sortir les ingrédients
- t₁ : **Séparer le blanc et le jaune d'un œuf.**
- t₂ : Faire ramollir le chocolat au bain-marie.
- t₃ : Incorporer les jaunes et le sucre au chocolat
- t₄ : Battre les blancs en neige ferme.
- t₅ : Ajouter délicatement les blancs en neige au mélange à l'aide d'une spatule.
- t₆ : Verser dans des verrines et mettre au frais pendant 2h.

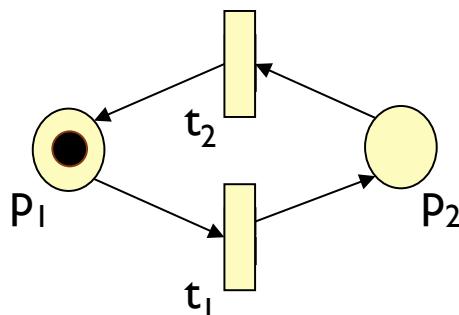


Différence : la séparation des œufs est interruptible

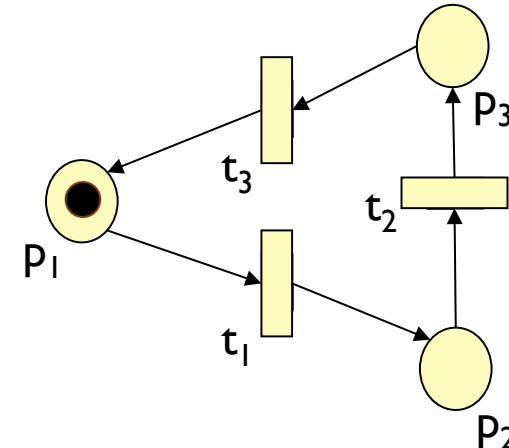
Exemple 2 : four à thermostat

- Four à thermostat
 - Four éteint ou allumé
 - Thermostat sur 170° ou 210°
 - Modéliser l'évolution de la température du four

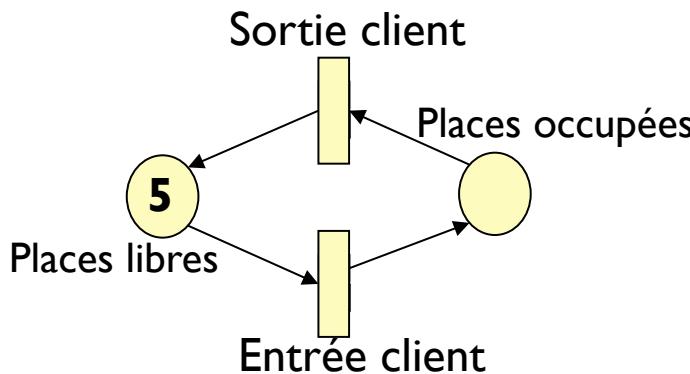
Structure : changement d'état



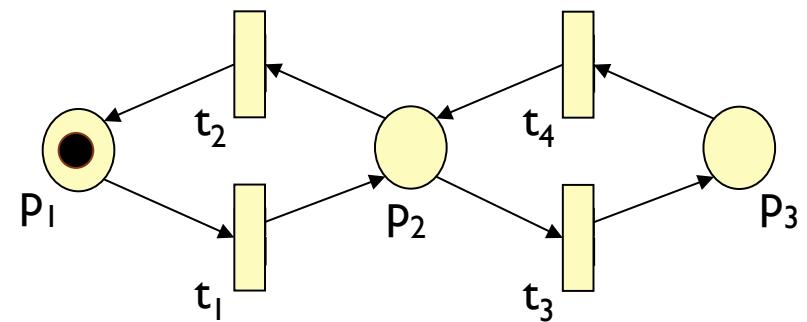
Bascule (e.g. ouvert/fermé)



Cycle (e.g. feu tricolore)



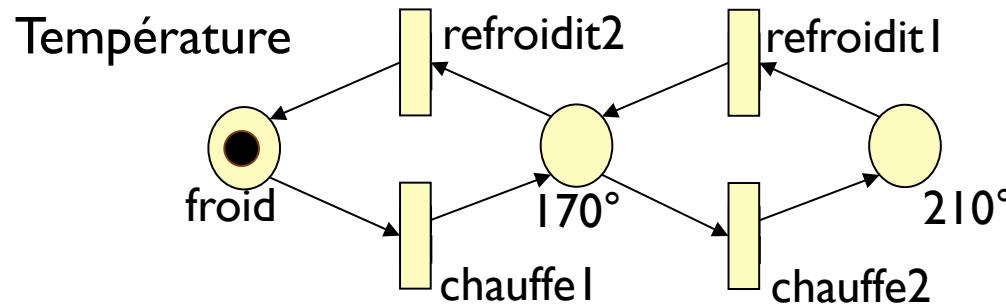
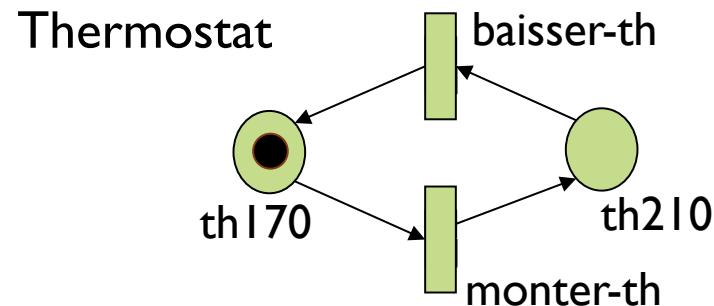
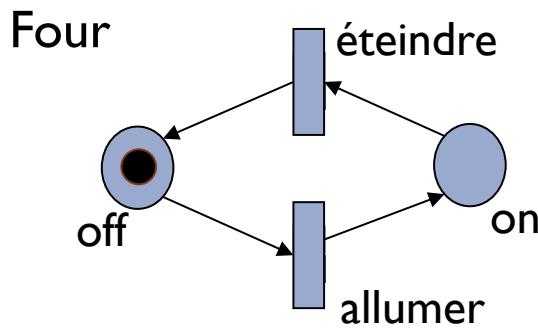
Exemple avec plusieurs jetons
(file d'attente)



Progression (e.g. molette)

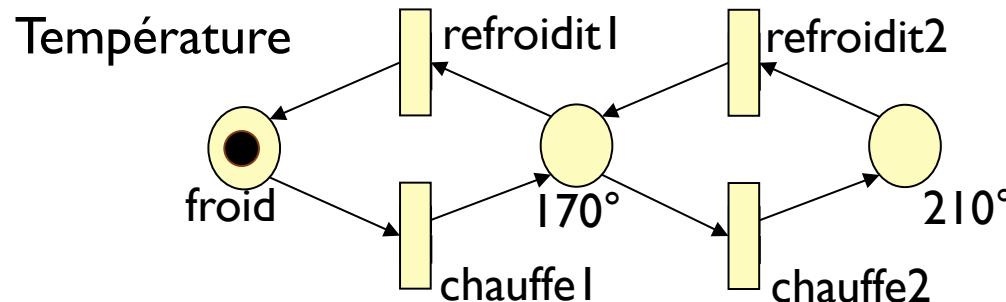
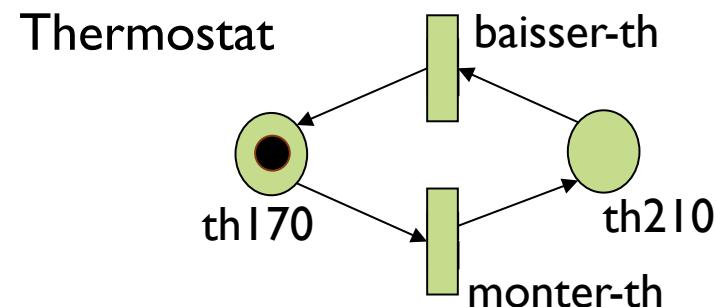
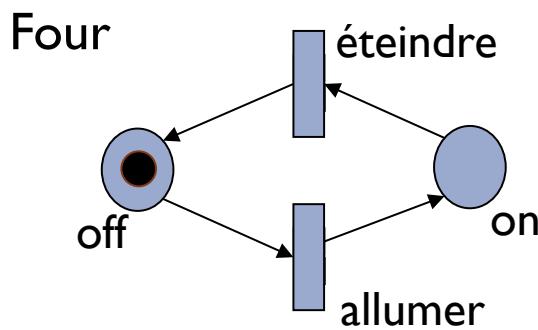
Exemple 2 : four à thermostat

- Four à thermostat
 - Four éteint ou allumé
 - Thermostat sur 170° ou 210°
 - Modéliser l'évolution de la température du four

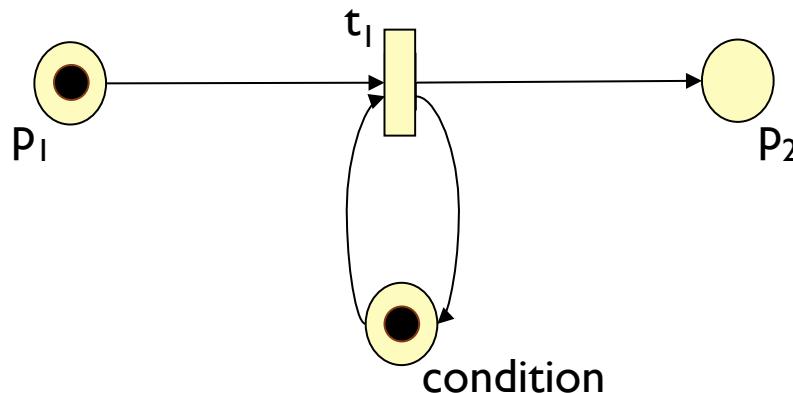


Exemple 2 : four à thermostat

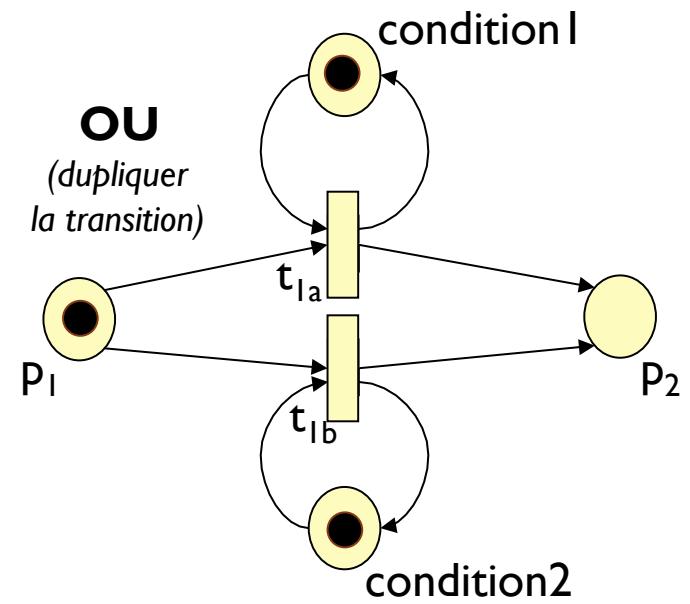
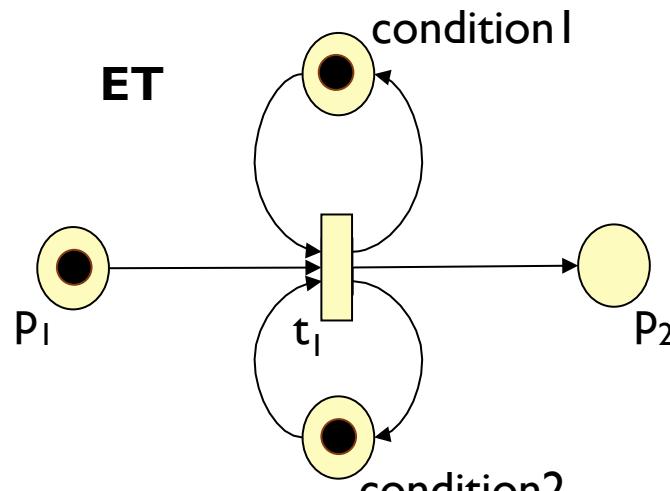
- Analyse
 - chauffe1 : si four allumé (et th=170 ou 210 : check)
 - chauffe2 : si four allumé **ET** th=210
 - refroidit2 : si four éteint **OU** si th=170
 - refroidit1 : si four éteint



Structure : conditionnement d'une transition



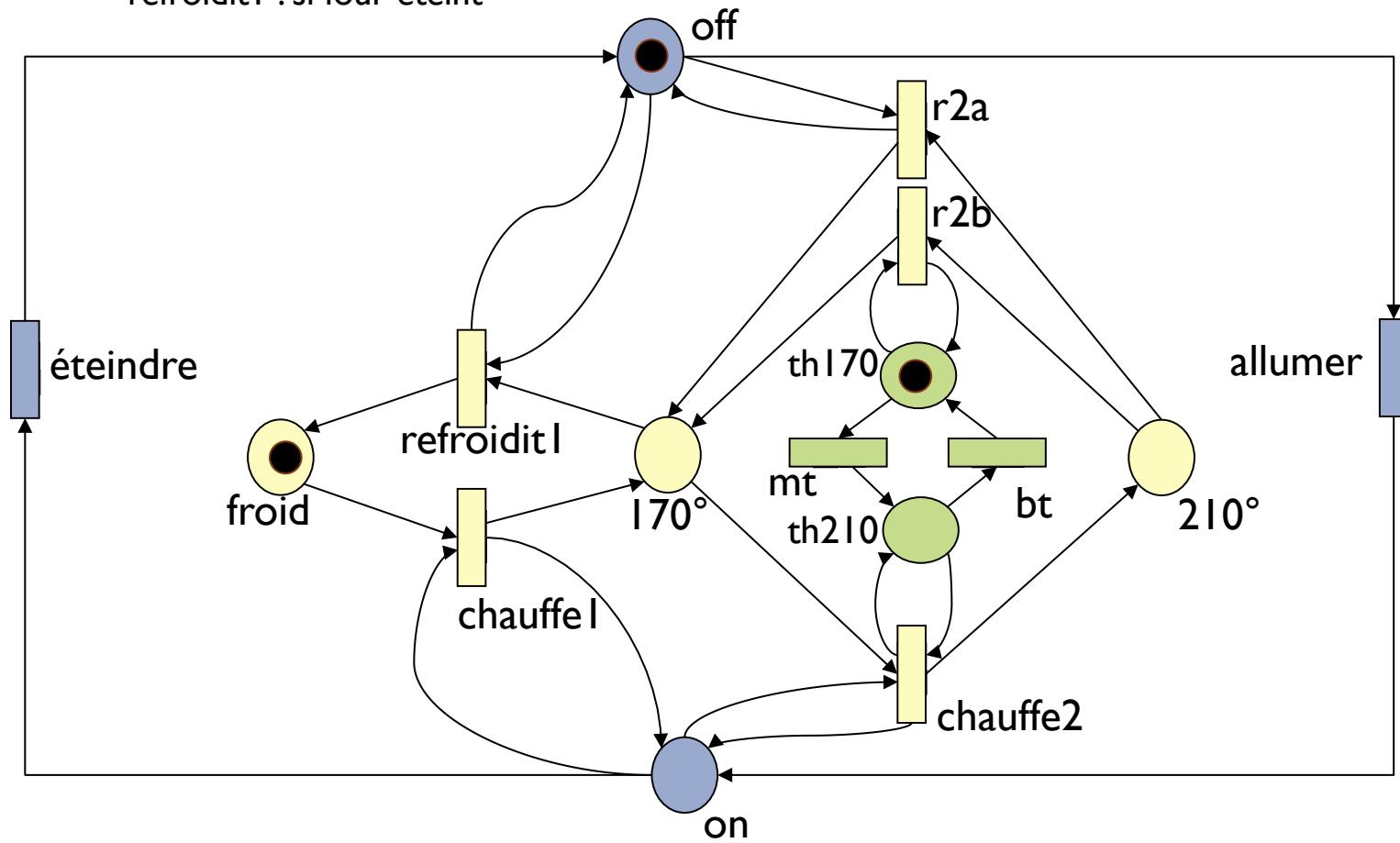
Si plusieurs conditions :



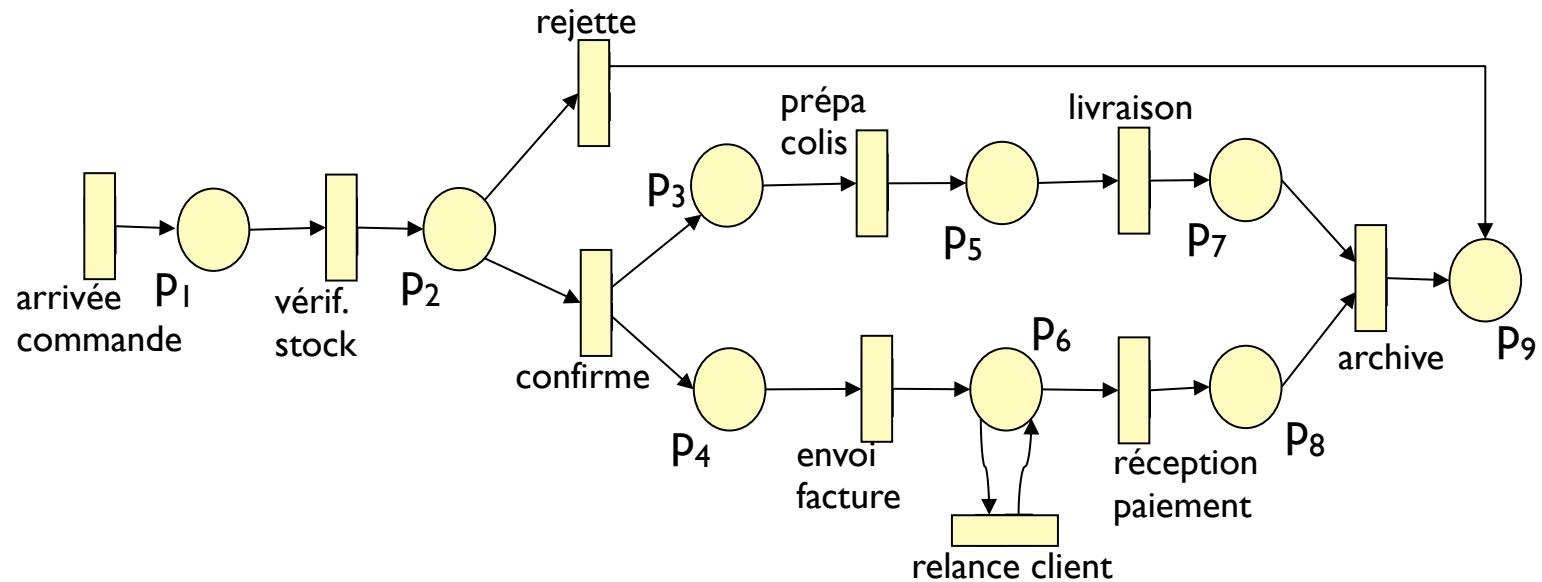
Exemple 2 : four à thermostat

- Analyse

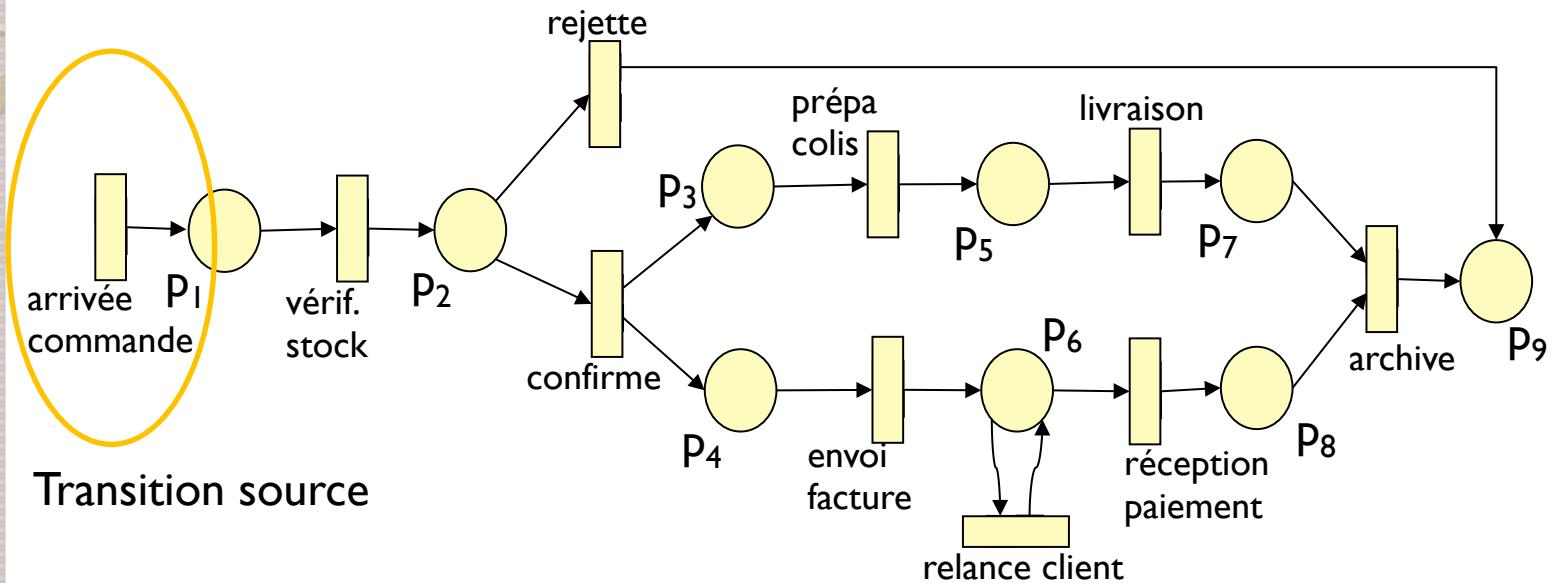
- chauffel : si four allumé (et $th=170$ ou 210 : check)
- chauffe2 : si four allumé **ET** $th=210$
- refroidit2 : si four éteint ($r2a$) **OU** si $th=170$ ($r2b$)
- refroidit1 : si four éteint



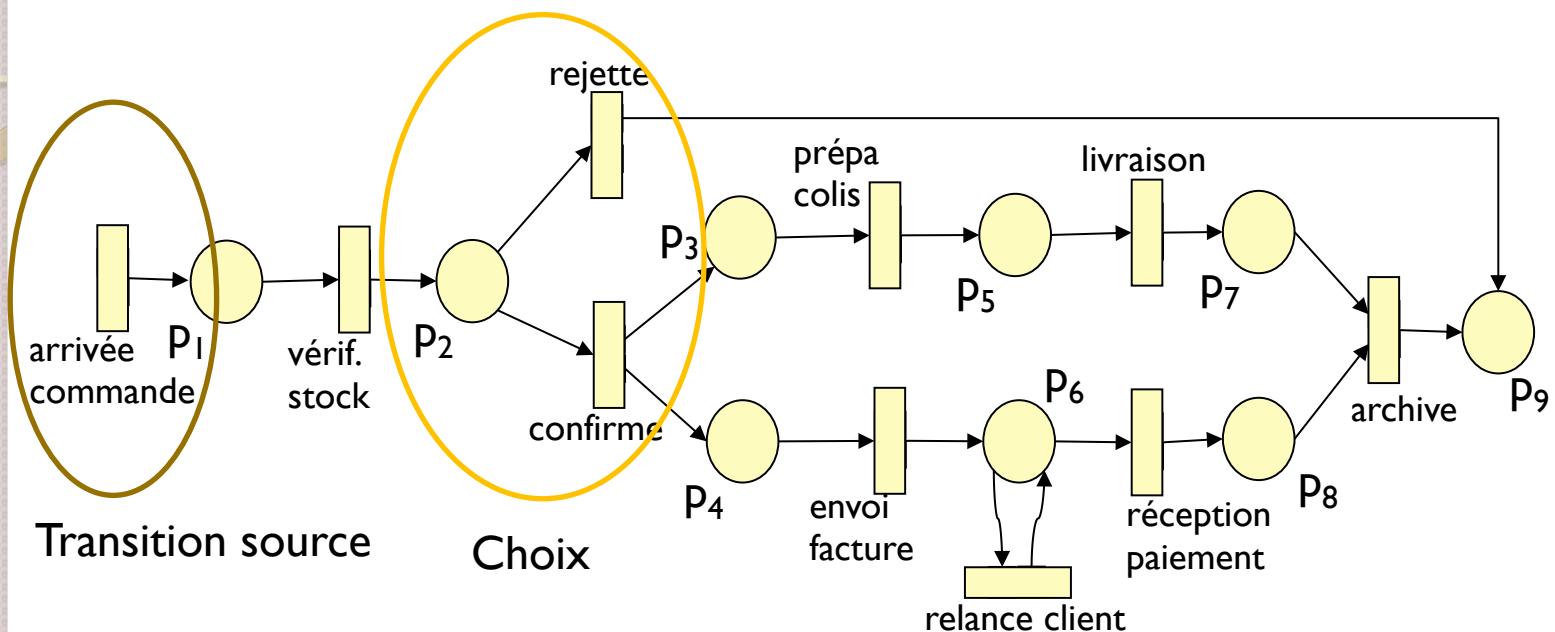
Exemple 3 : commande



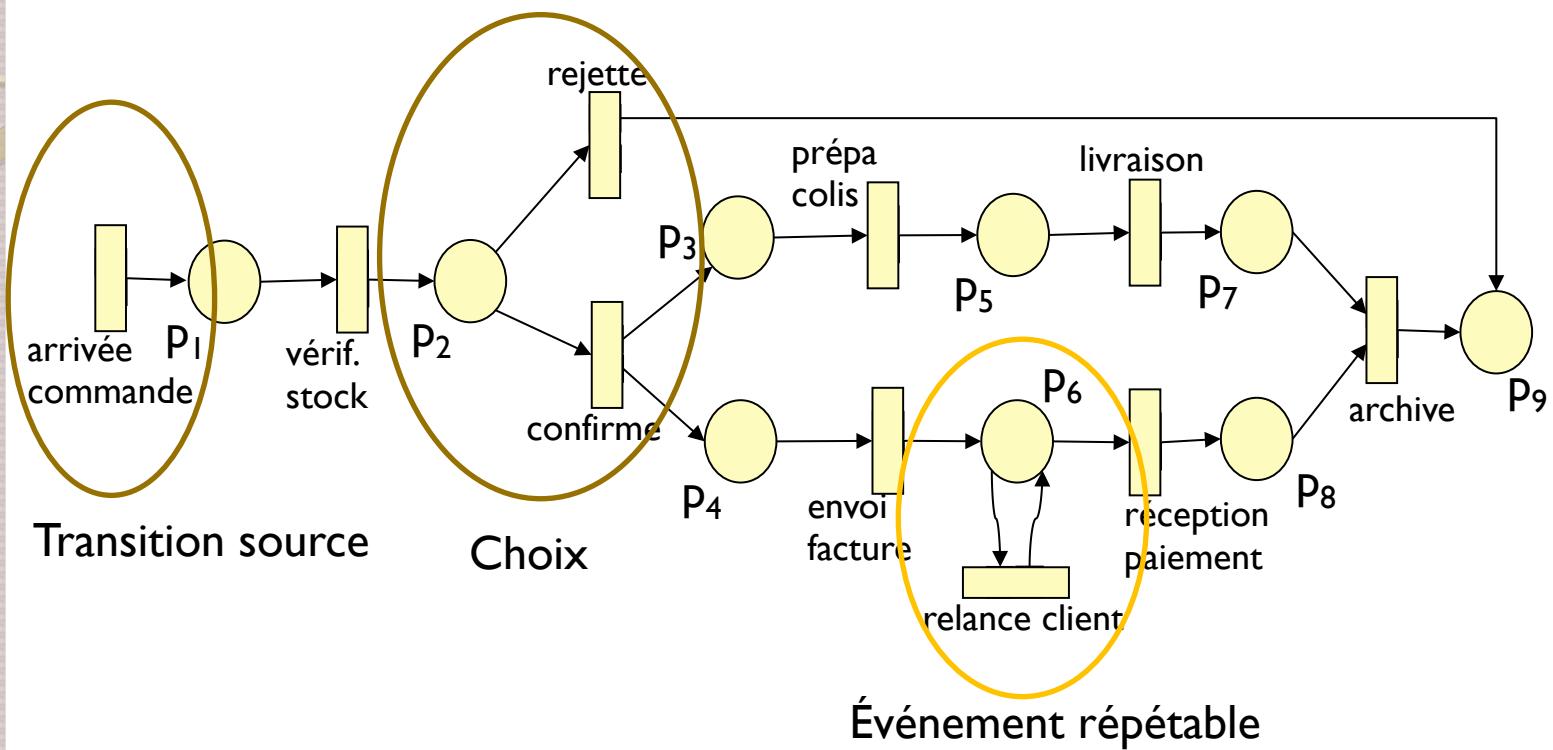
Exemple 3 : commande



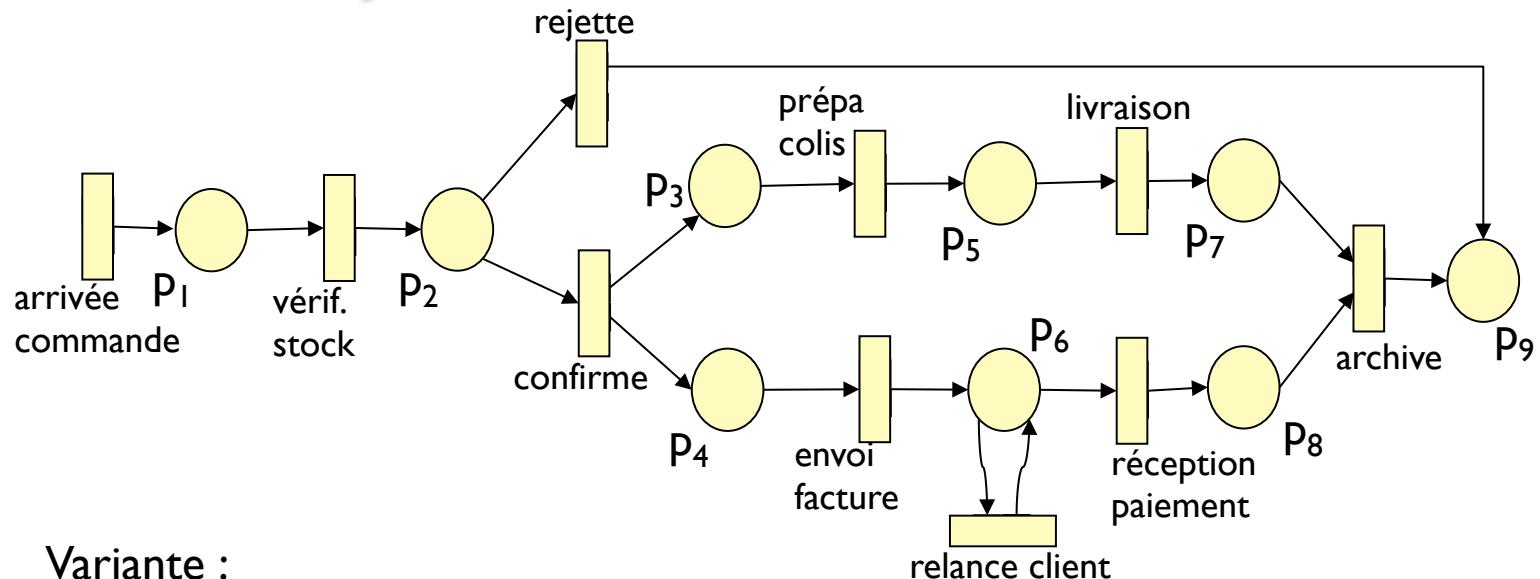
Exemple 3 : commande



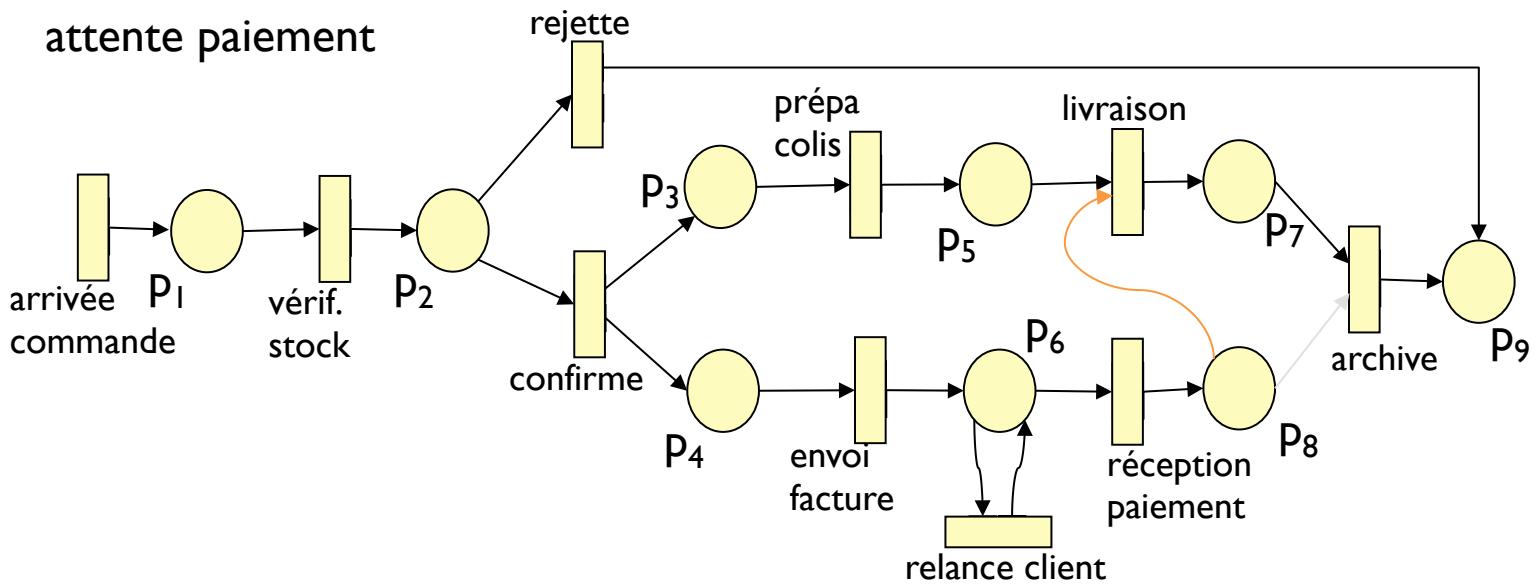
Exemple 3 : commande



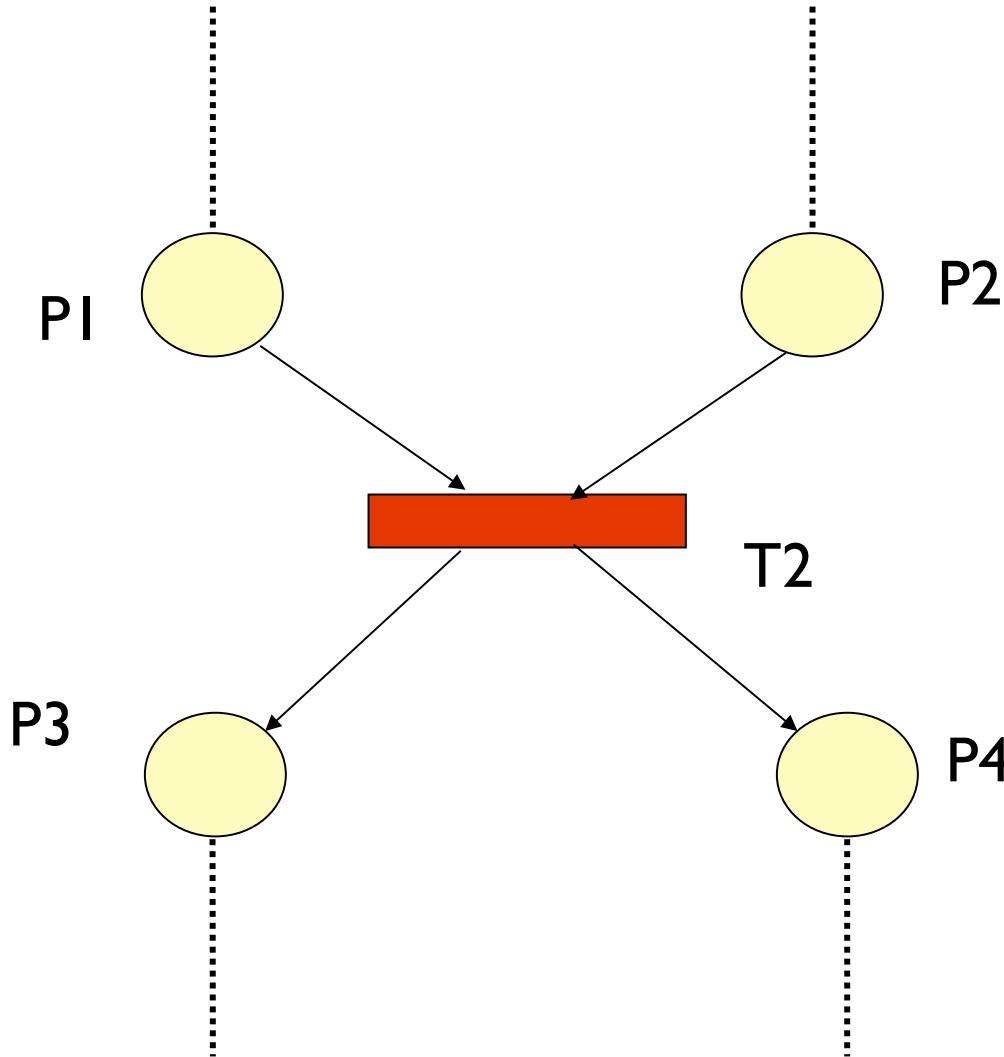
Exemple 3 : commande



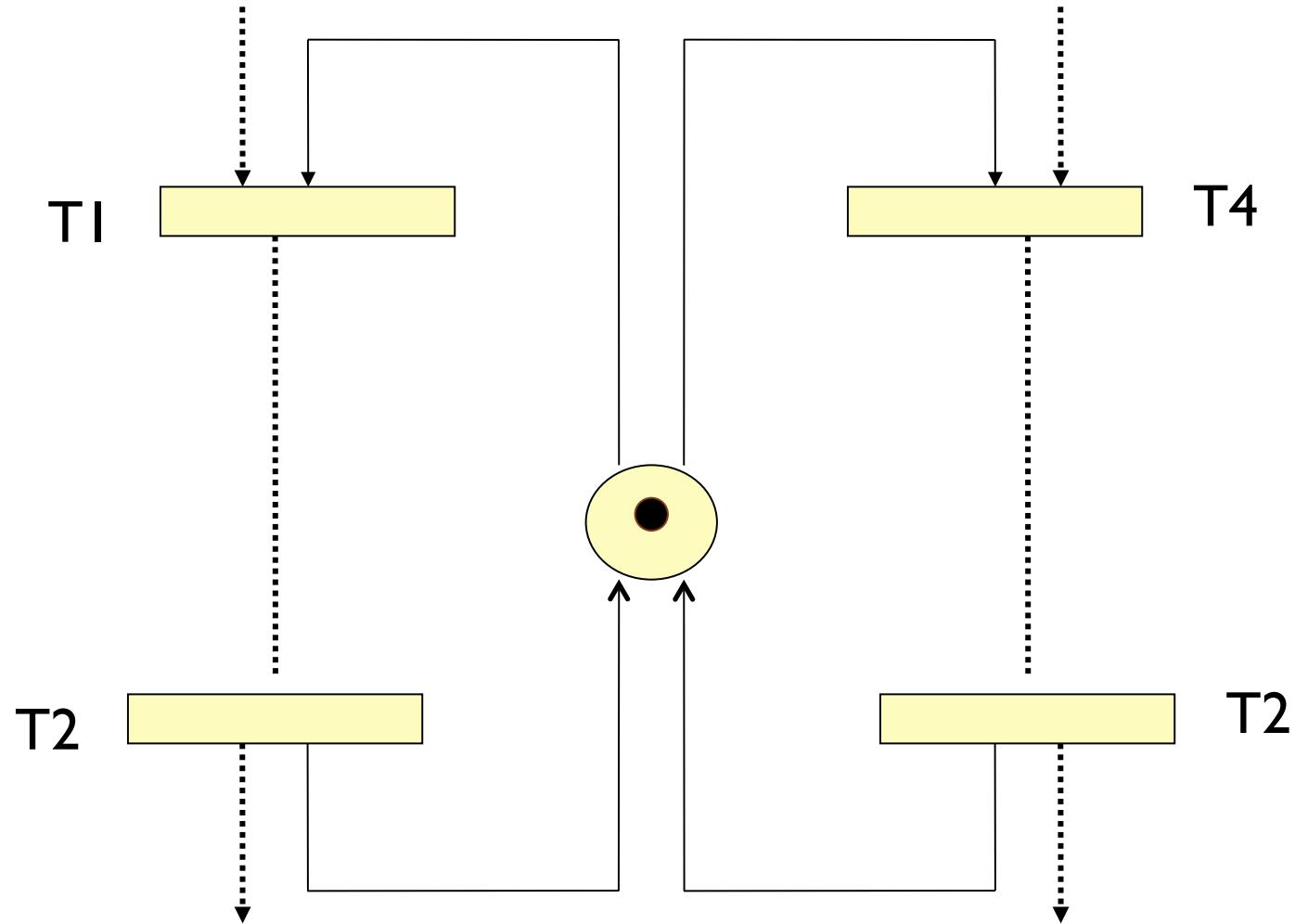
Variante :
attente paiement



Structure: synchronisation par rendez-vous



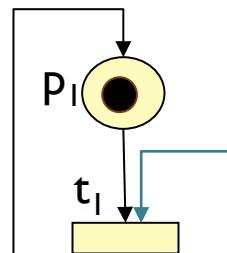
Structure : partage de ressources



Exclusion

Interblocage

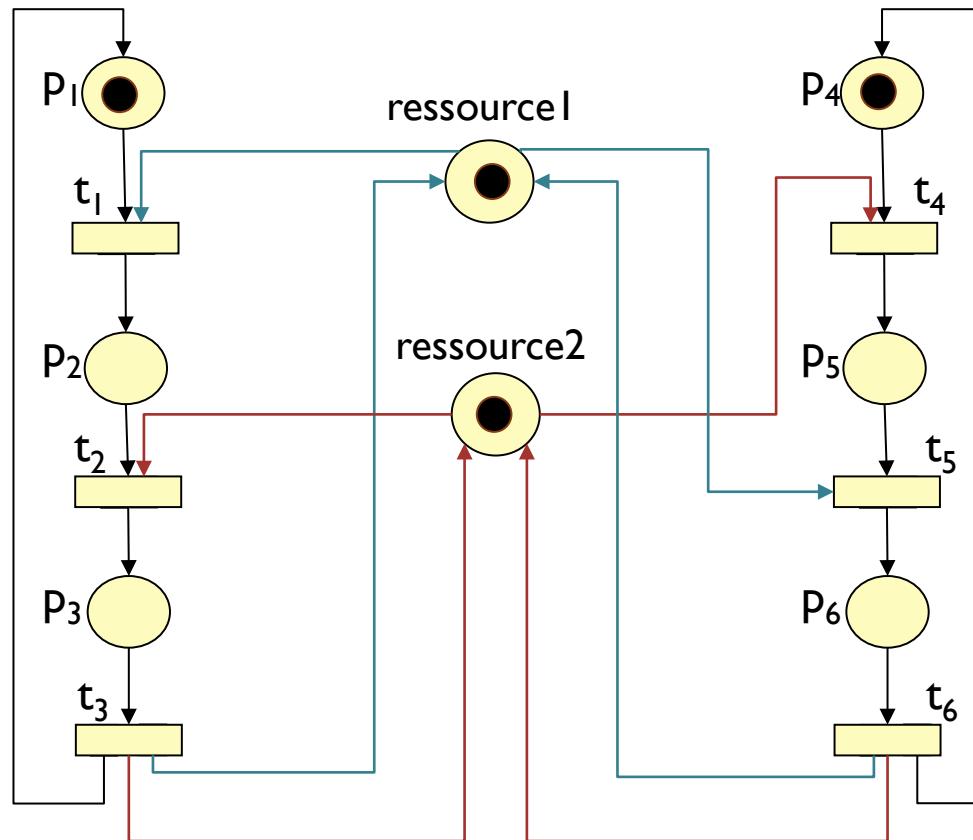
Processus 1



ressource1

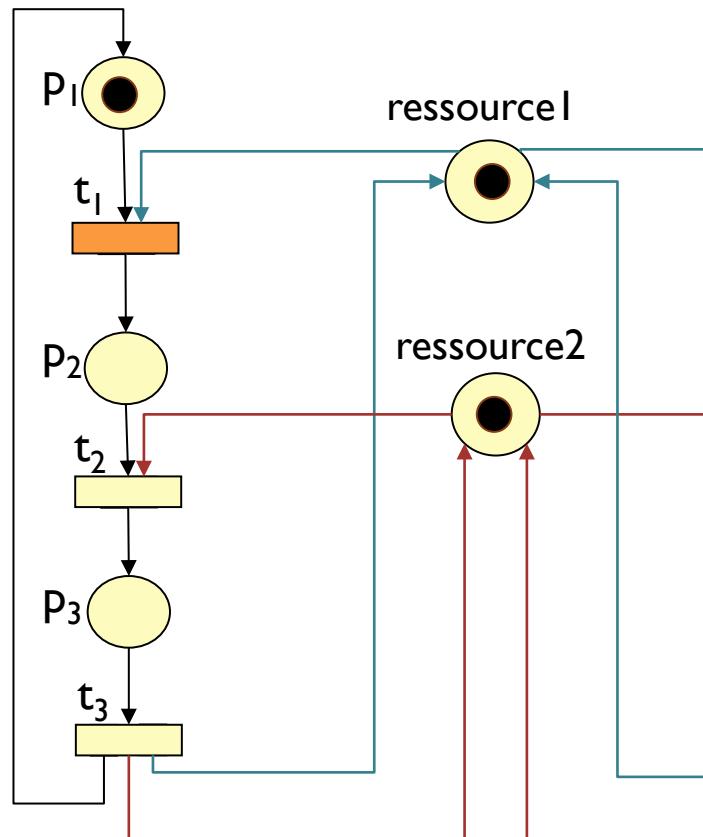
ressource2

Processus 2



Interblocage

Processus 1

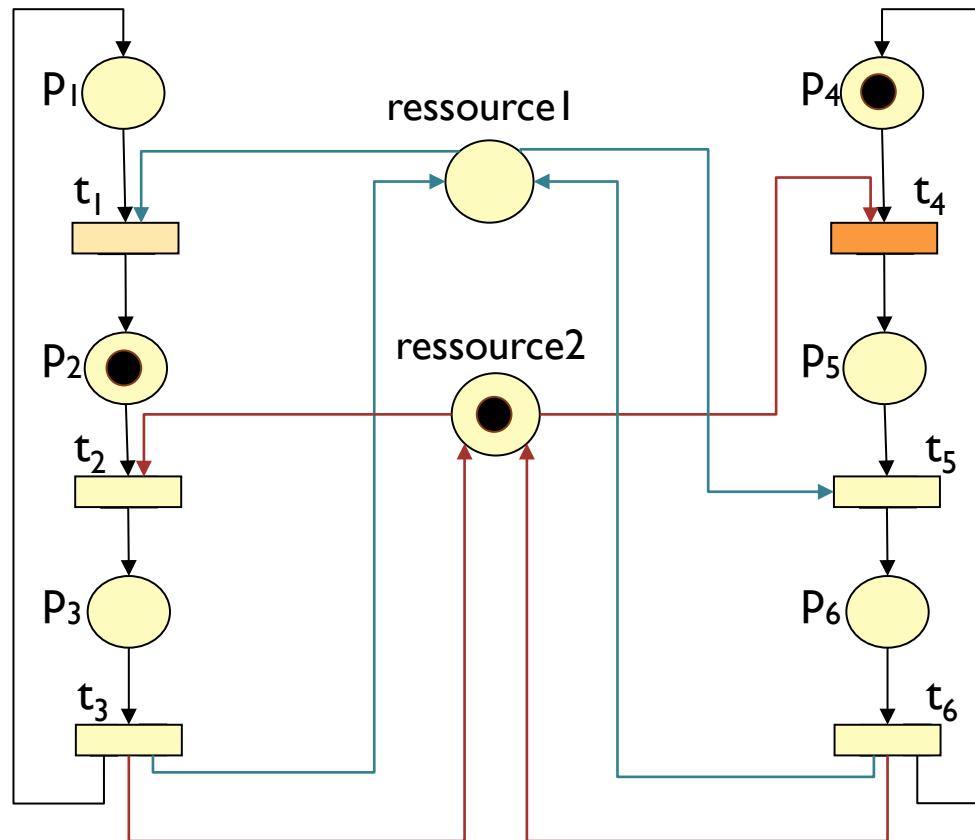


Processus 2

Tir de t_1

Interblocage

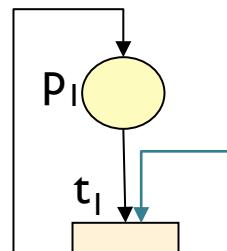
Processus 1



Tir de t_2

Interblocage

Processus 1



t_1

t_2

t_3

ressource1

ressource2

Processus 2



t_4

t_5

t_6

Blocage !

Détection des possibilités de blocage : analyse des propriétés du RdP marqué

Plan

- Introduction
- Syntaxe
- Modélisation
- Analyse (accessibilité / vivacité)
- Extensions
- Vérification en LTL

Rappel sémantique

- Marquage accessible

Le marquage M' est **accessible** à partir de M ssi il existe une séquence $s=(t_1, \dots, t_n)$ de transitions menant de M à M' (ie *pour chaque i entre 1 et n, t_i est franchissable dans M_{i-1} et mène à M_i avec $M_0=M$ et $M_n=M'$*)

$$M=M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} \dots M_{n-1} \xrightarrow{t_n} M_n=M'$$

soit

$$M \xrightarrow{s} M'$$

Propriétés d'un RdP marqué

- Propriétés relatives à l'état du système

Le nombre de jetons est-il borné ?

Si oui, **nombre fini de marquages accessibles**

- Un **RdP marqué** est **(k-)borné** ssi toutes ses places sont **(k-)bornées**
 - Une **place p** est **k-bornée** ssi pour tout marquage **M** accessibles depuis **M_0** , $M(p) \leq k$
 - Une **place p** est **bornée** ssi il existe un **k** tel que **p** est **k-bornée**

Propriétés d'un RdP marqué

- Propriétés relatives à l'activité

Est-ce que le réseau, ou une partie du réseau, peut toujours évoluer ?

- Un RdP marqué est **quasi-vivant** ssi toutes ses transitions sont quasi-vivantes

- Une **transition** t est **quasi-vivante** ssi elle peut être franchie au moins une fois depuis le marquage initial (il existe au moins un marquage accessible depuis M_0 où t est franchissable)

Toute partie du RdP peut servir au moins une fois

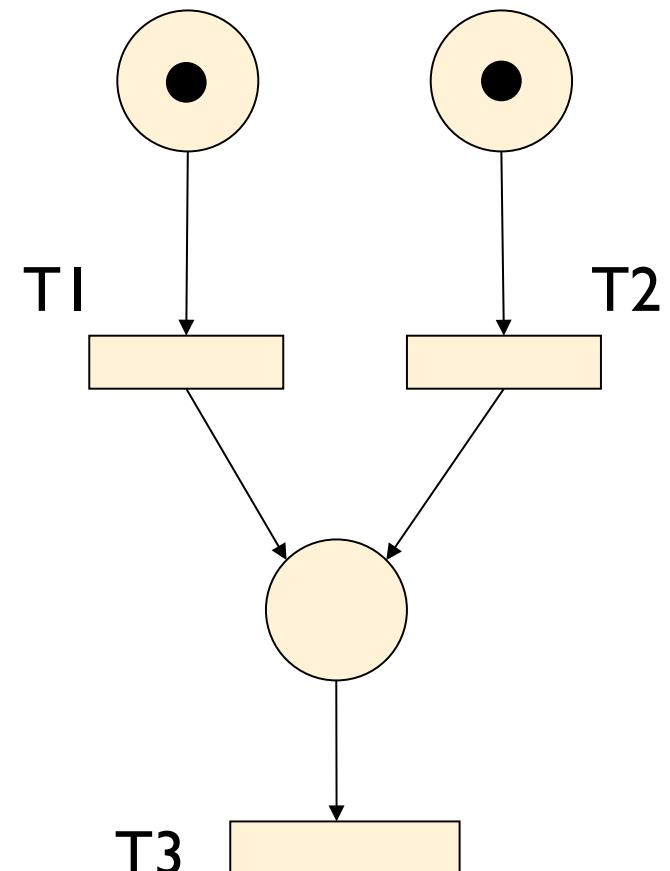
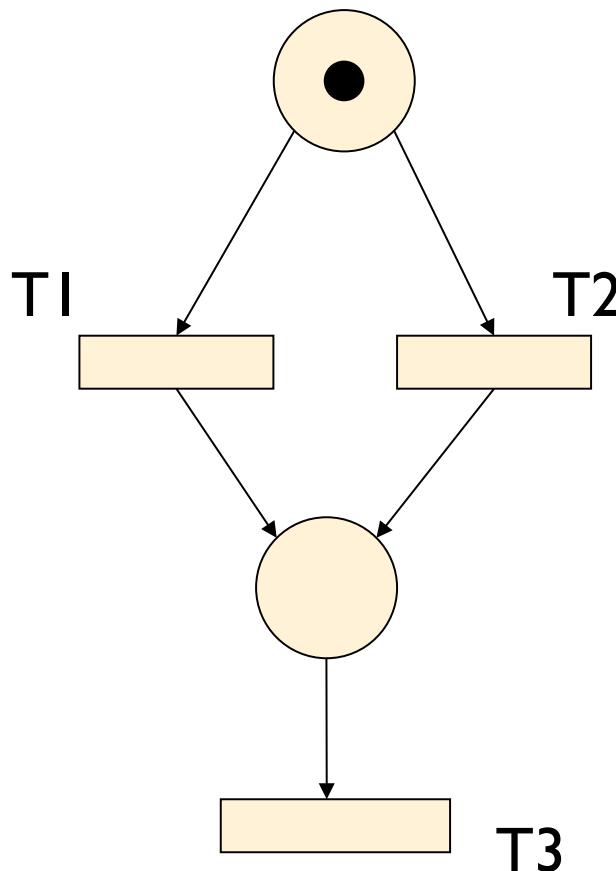
- Un RdP marqué est **vivant** ssi toutes ses transitions sont vivantes

- Une **transition** est **vivante** ssi quelque soit l'évolution il reste toujours possible de la franchir (i.e. depuis tout marquage M accessible depuis M_0 , il existe un marquage M' accessible depuis M tel que t est franchissable)

Tout le RdP reste actif quelque soit l'évolution

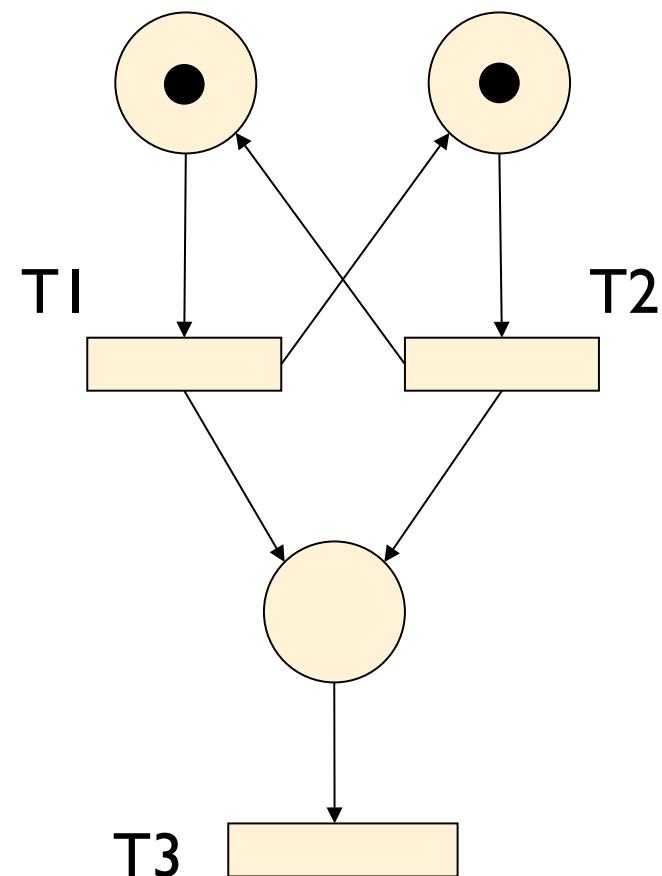
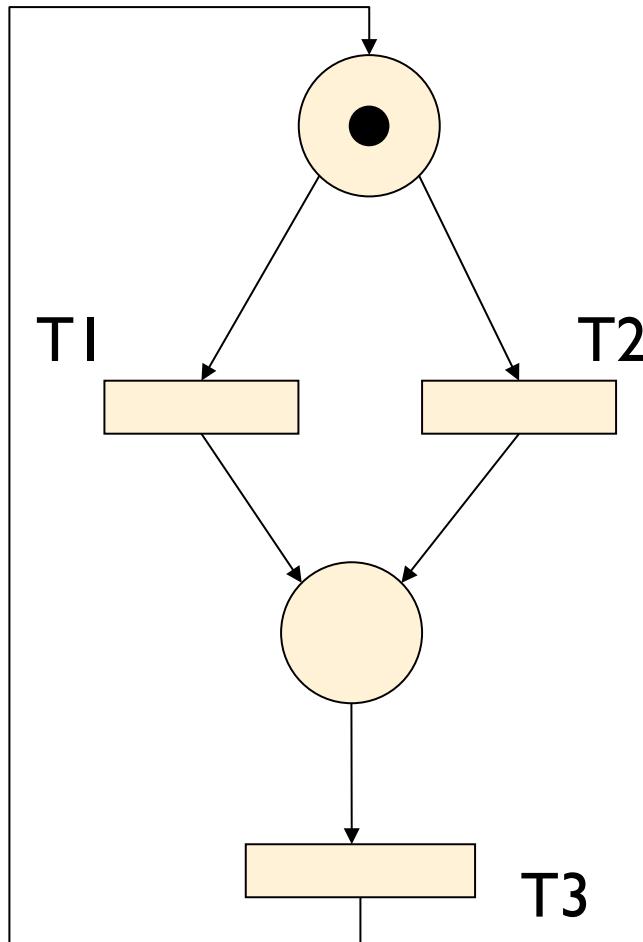
Exemple

Quasi vivant



Exemple

Vivant



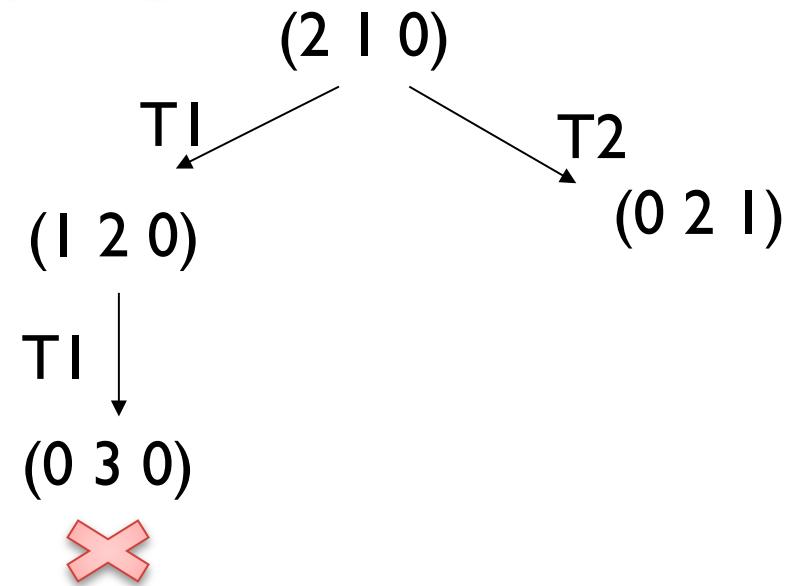
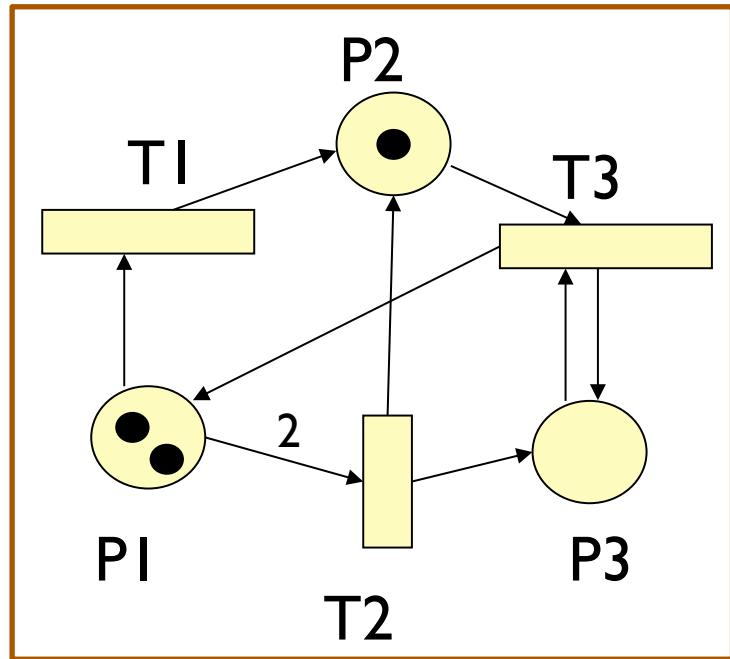


Propriétés d'un RdP marqué

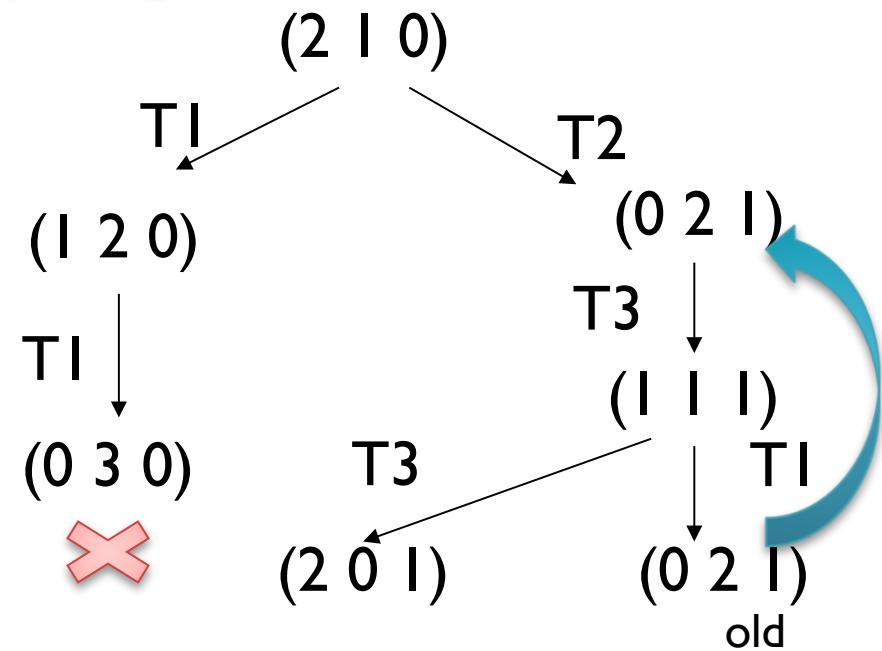
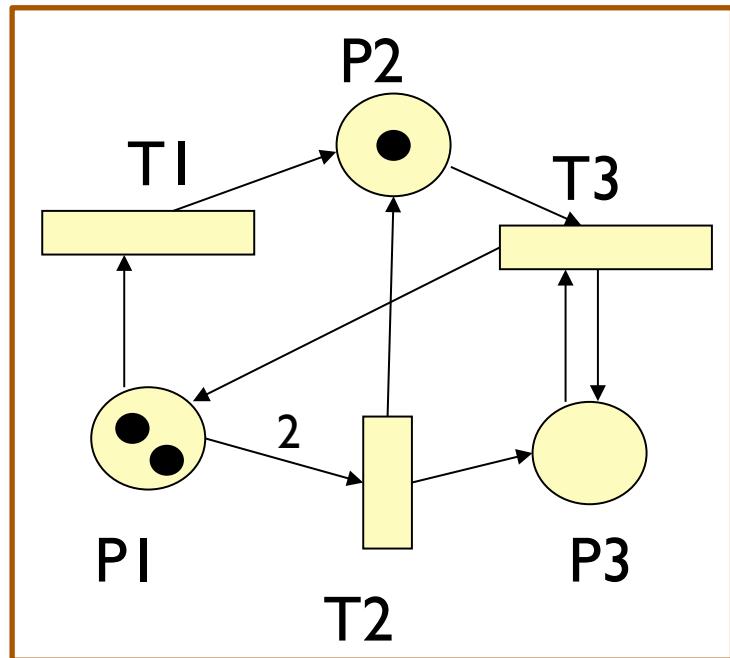
- Propriétés relatives à l'état du système
 - Borné
 - k-borné
- Propriétés relatives à l'activité
 - Quasi-vivant
 - Vivant
 - Sans blocage
 - Réversible

Toutes ses propriétés sont liées aux marquages accessibles.

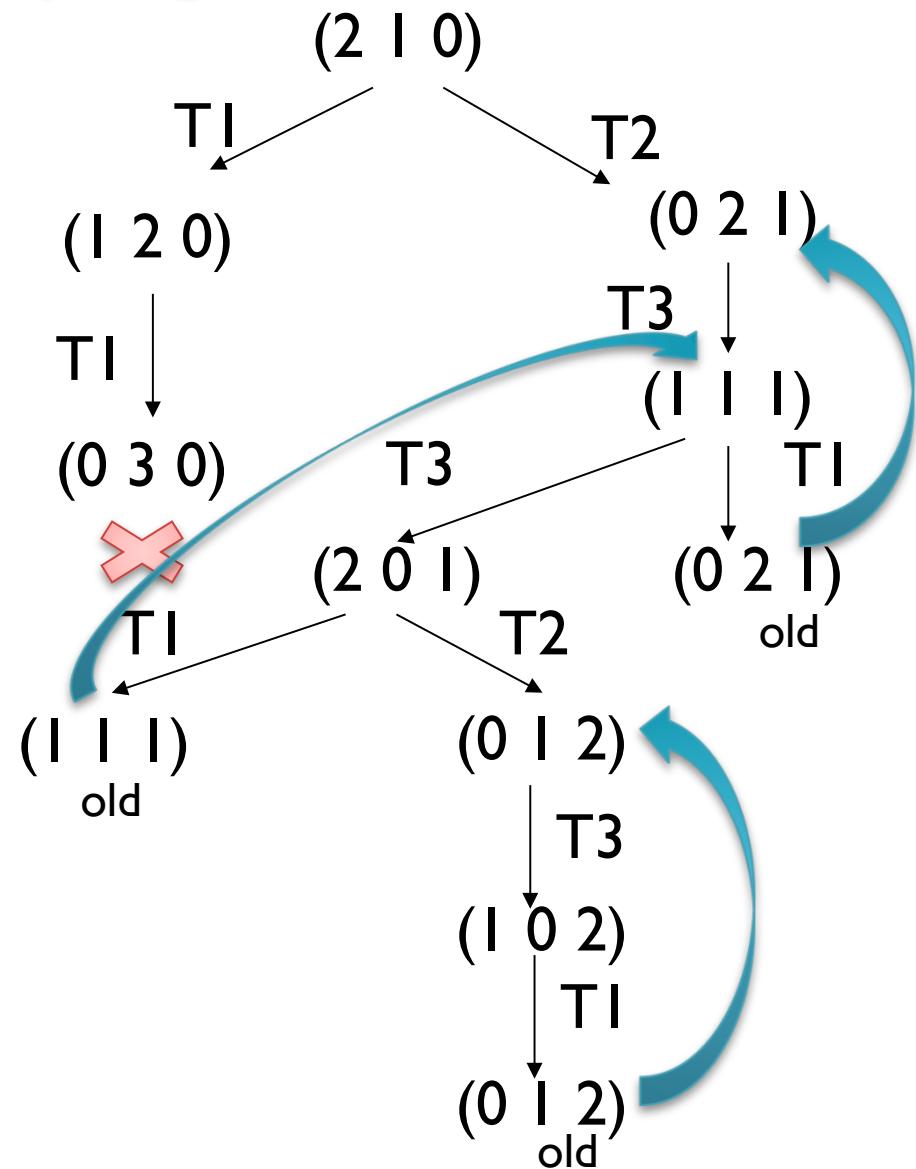
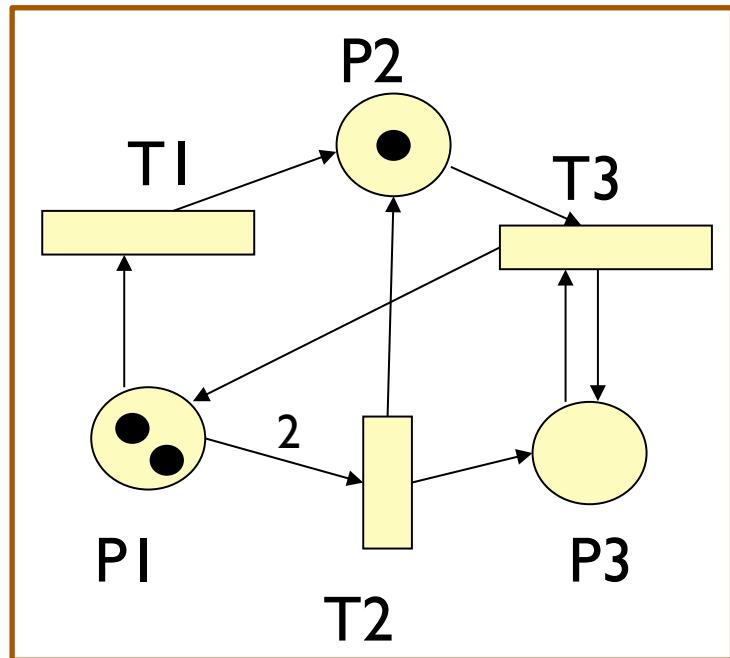
Arbre des marquages accessibles



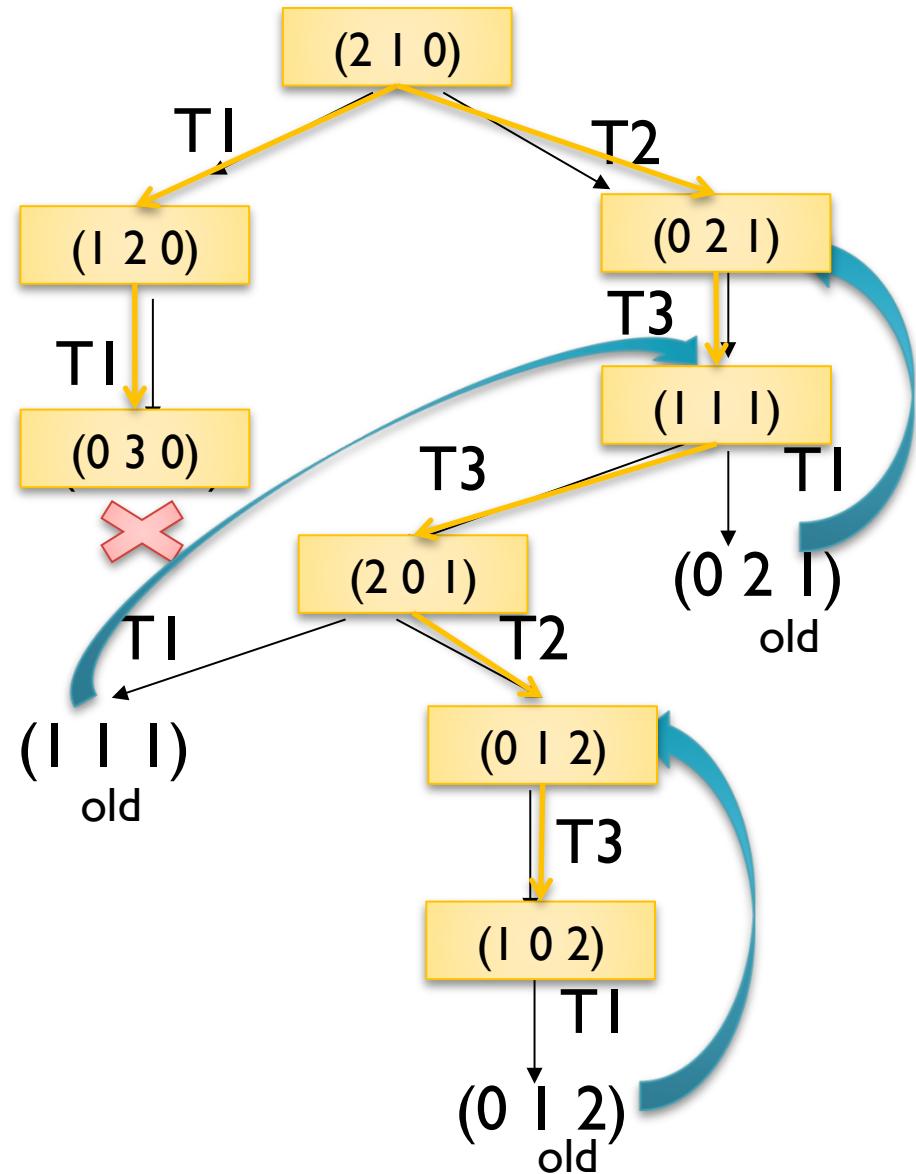
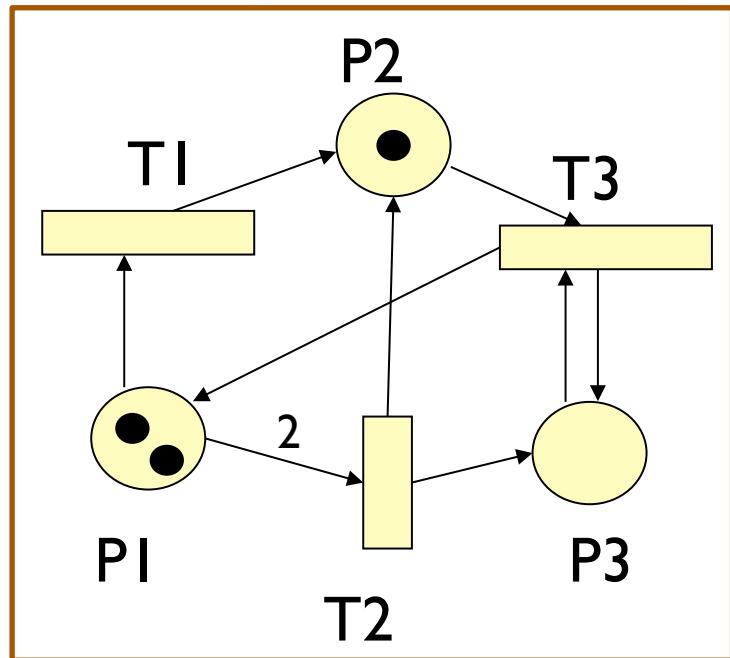
Arbre des marquages accessibles



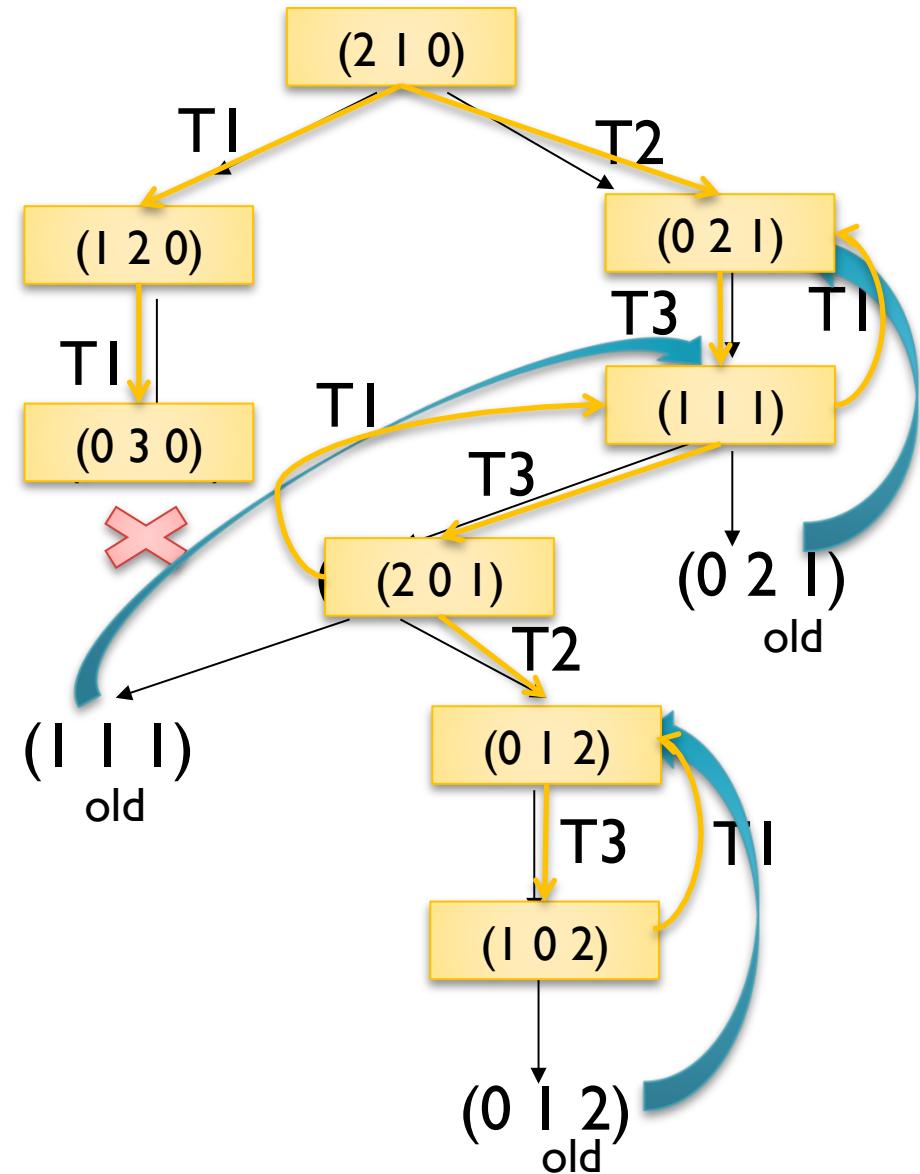
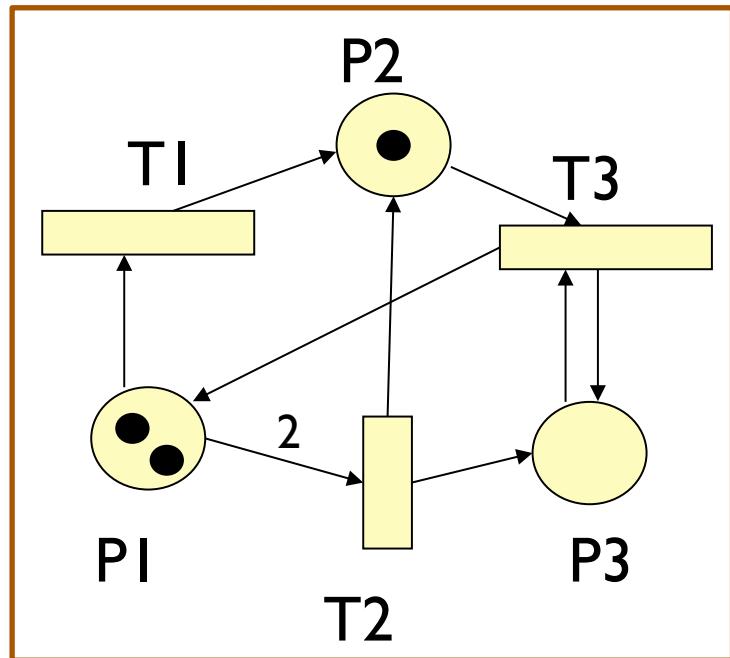
Arbre des marquages accessibles



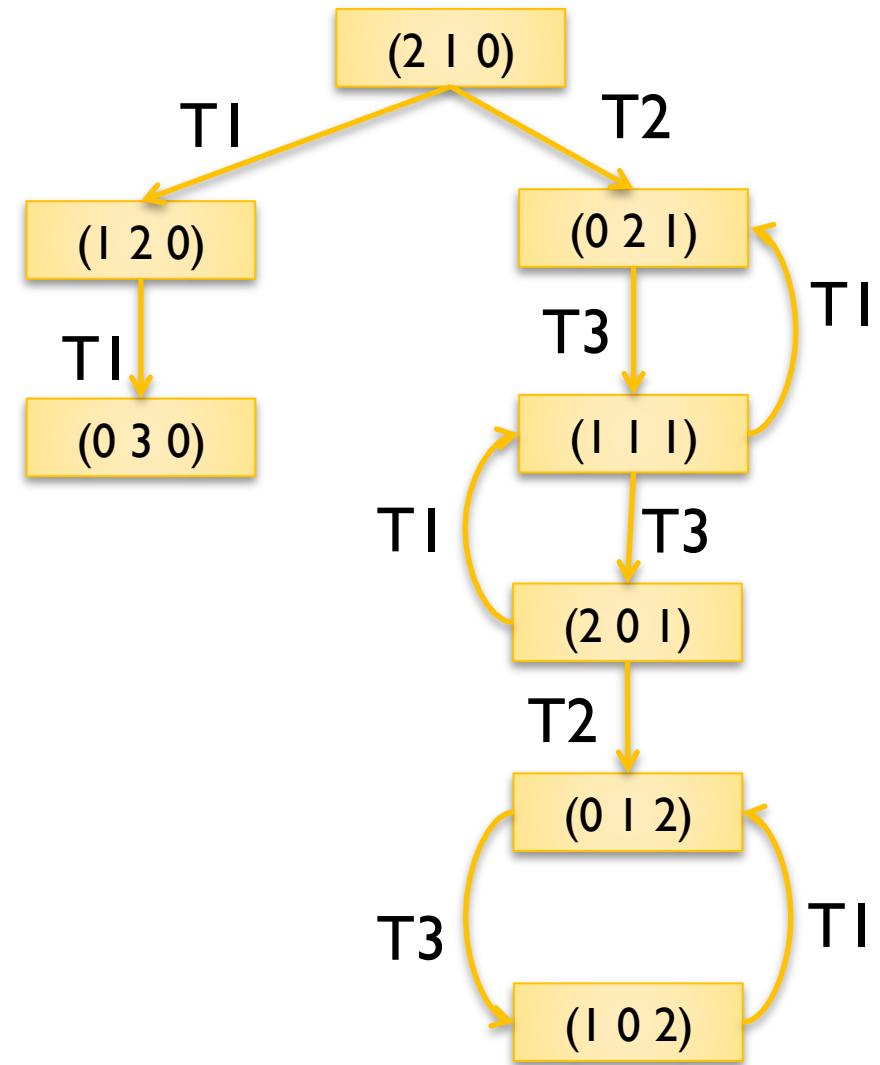
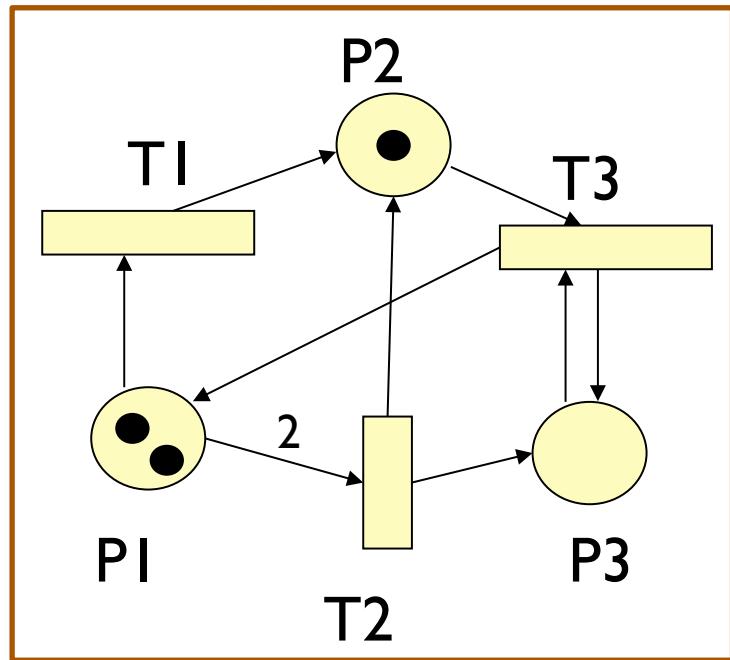
Graphe des marquages accessibles



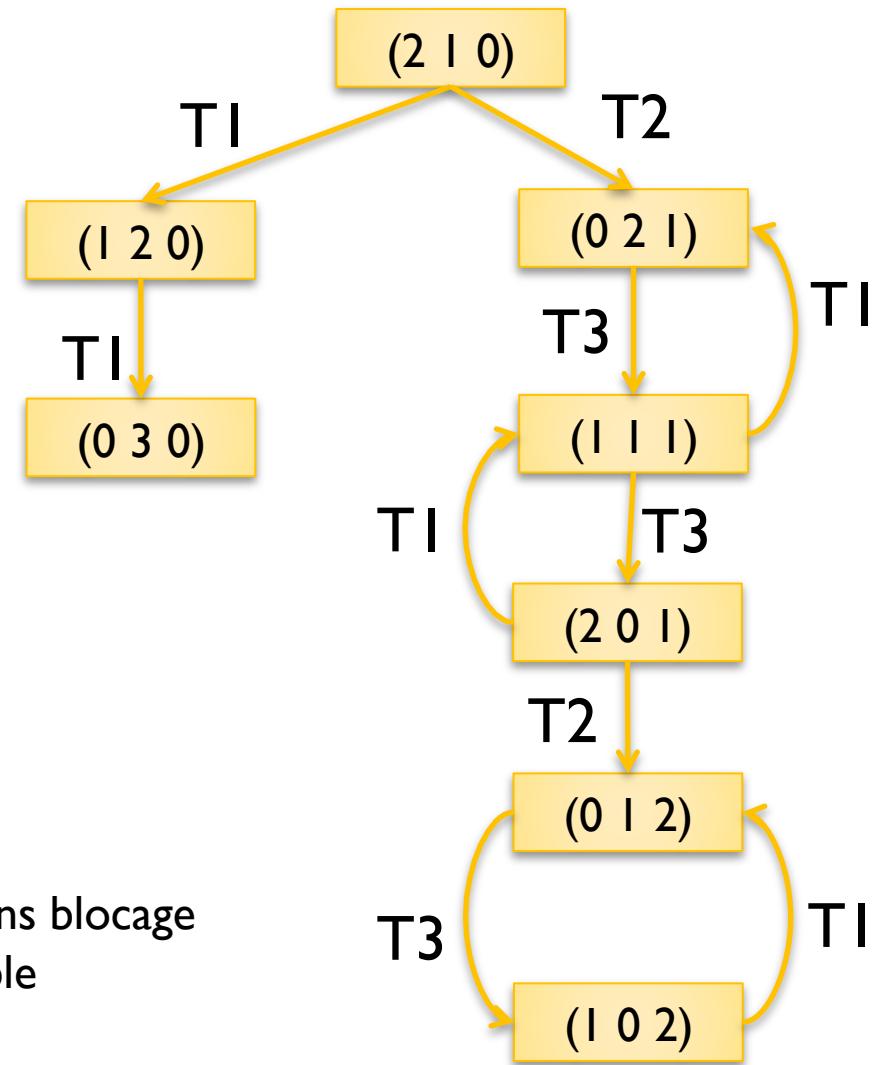
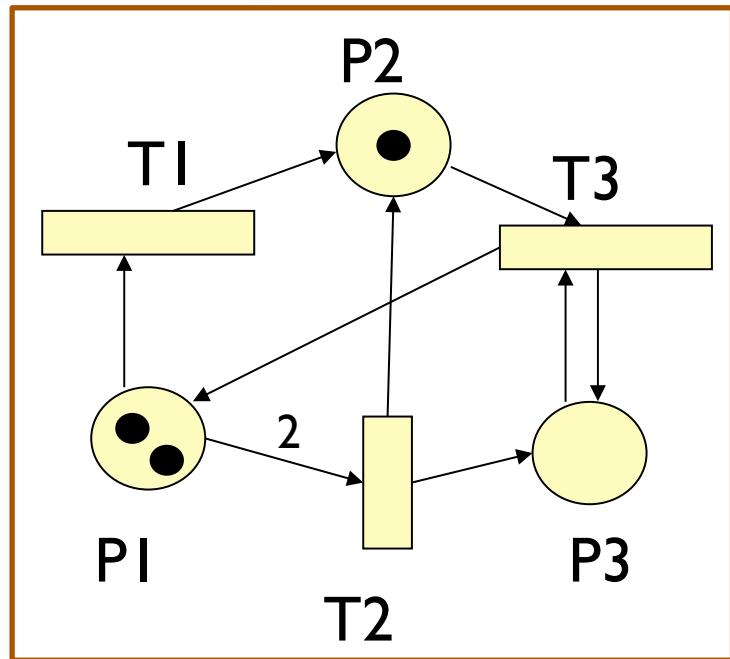
Graphe des marquages accessibles



Graphe des marquages accessibles



Graphe des marquages accessibles



Analyse :

3-borné

1 marquage puit (**0 3 0**) : pas sans blocage

A fortiori pas vivant, ni réversible

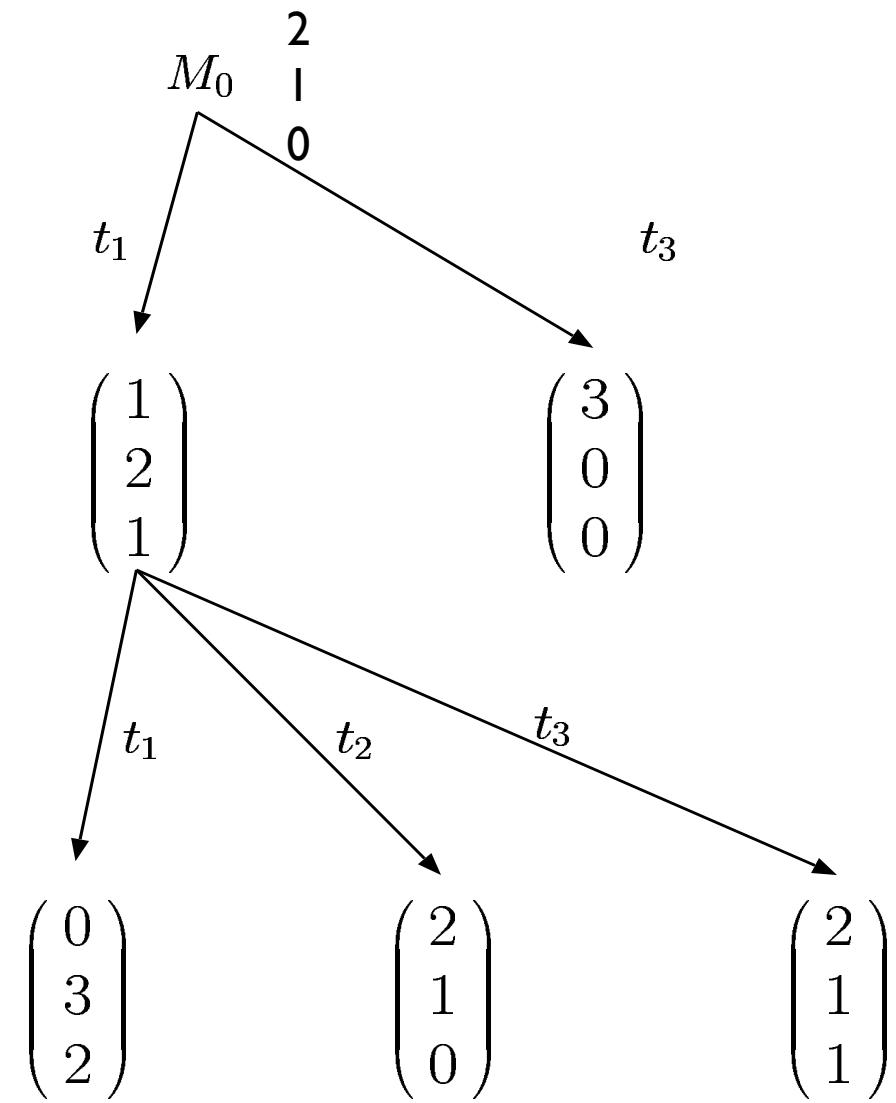
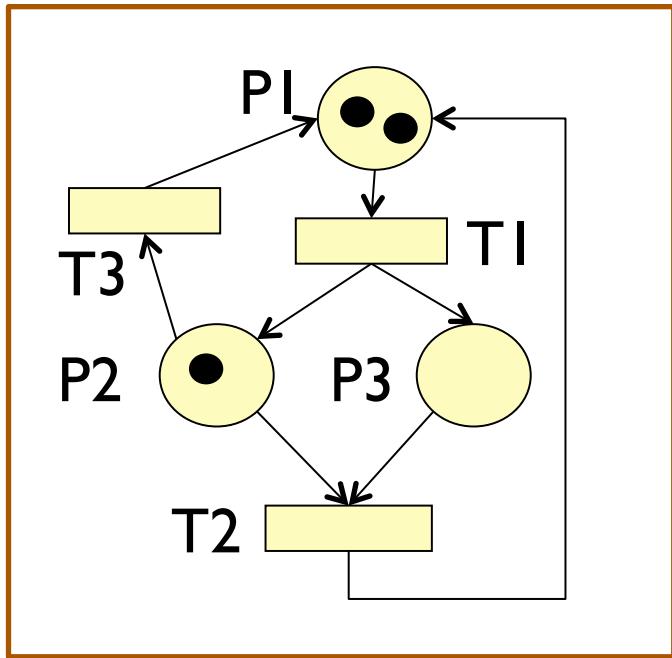
Quasi-vivant



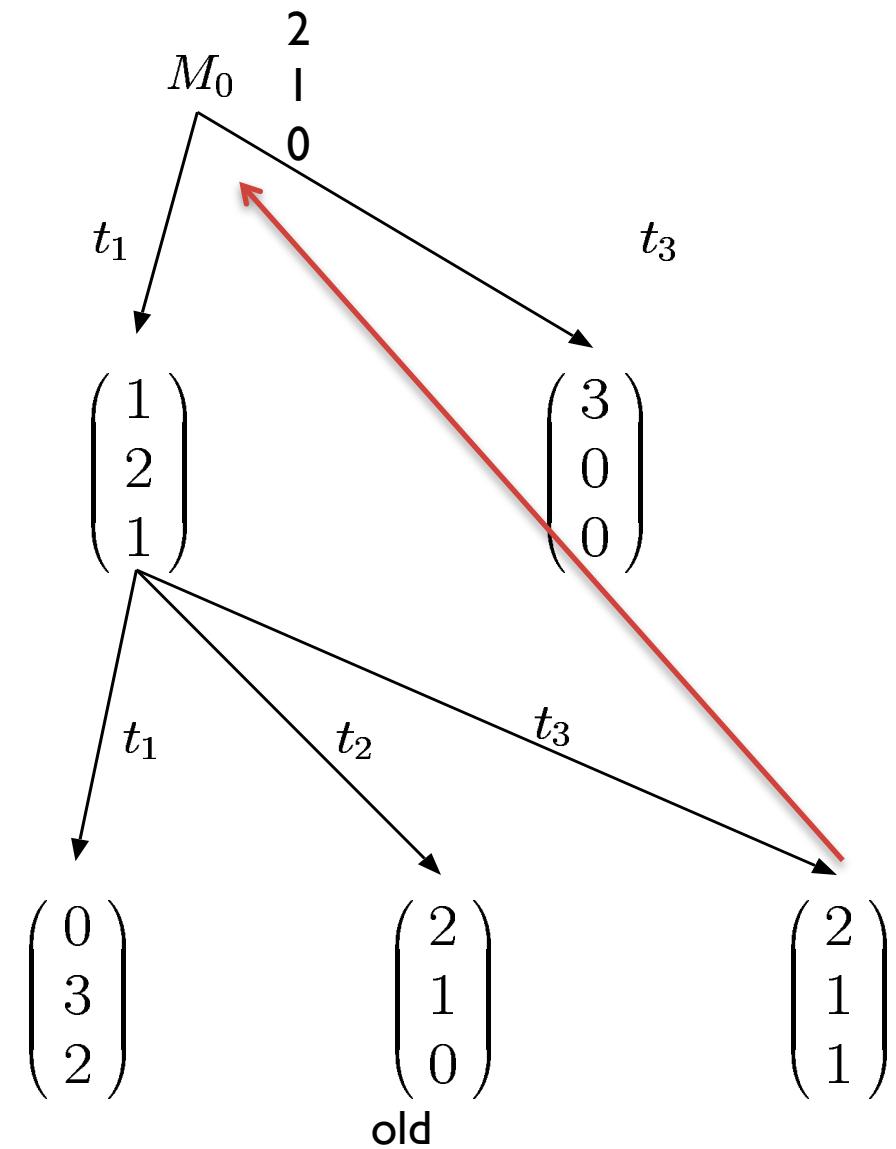
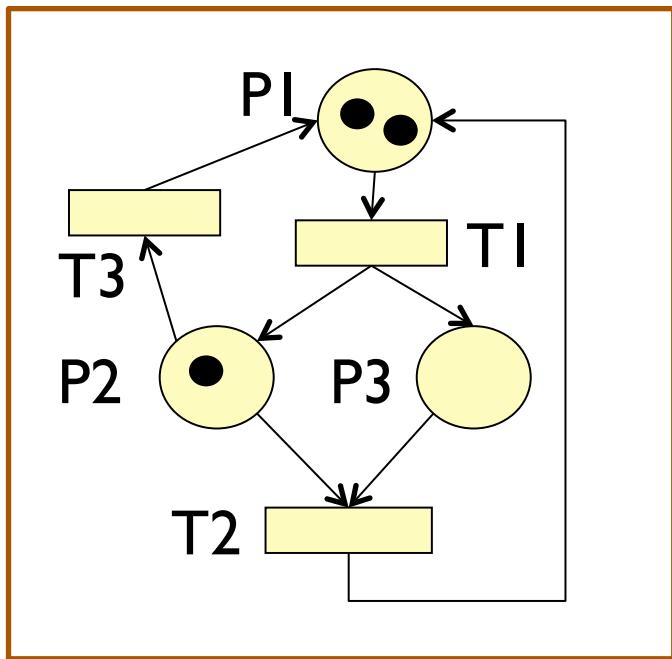
Graphe des marquages

- Graphe fini (RdP borné): cas idéal car toutes les propriétés peuvent être déduites simplement
- Graphe infini : les propriétés ne peuvent être déduites. On construit l'arbre de couverture.

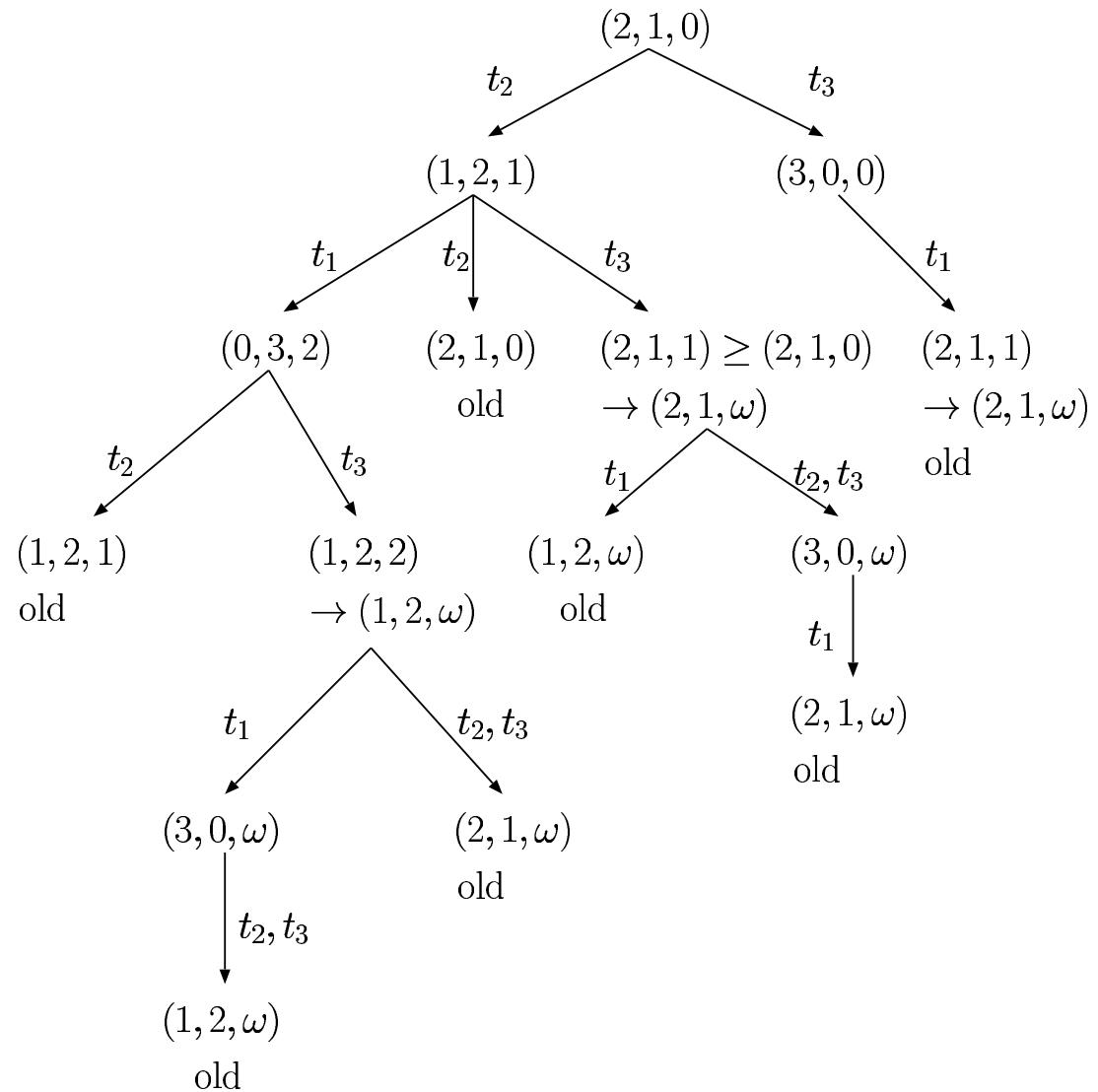
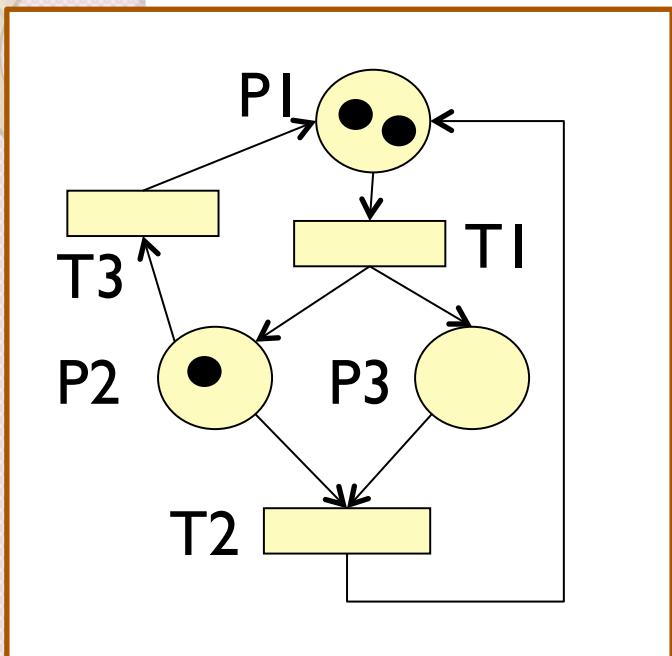
Arbre de couverture



Arbre de couverture



Arbre de couverture



Plan

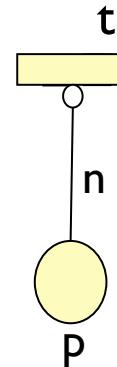
- Introduction
- Syntaxe
- Modélisation
- Analyse (accessibilité / vivacité)
- **Extensions**
- Vérification en LTL

Principales extensions

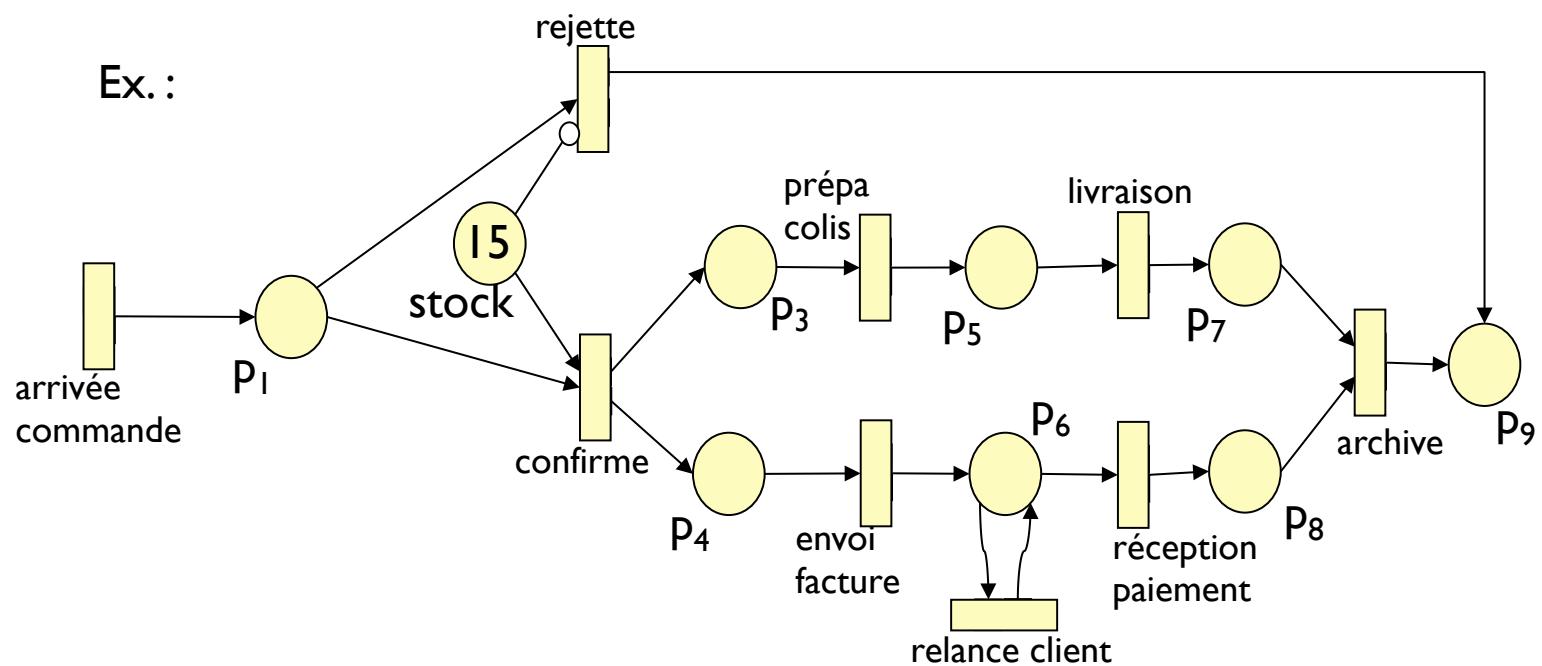
- Arcs inhibiteurs
- RdP colorés
- RdP temporisés
- RdP récursifs

Arcs inhibiteurs

- Un arc inhibiteur de valuation n est validé si la place de départ de l'arc n'a pas **n jetons**.
 - Souvent, arc de valuation 1 : la transition n'est franchissable que si la place d'entrée est vide de tout jeton

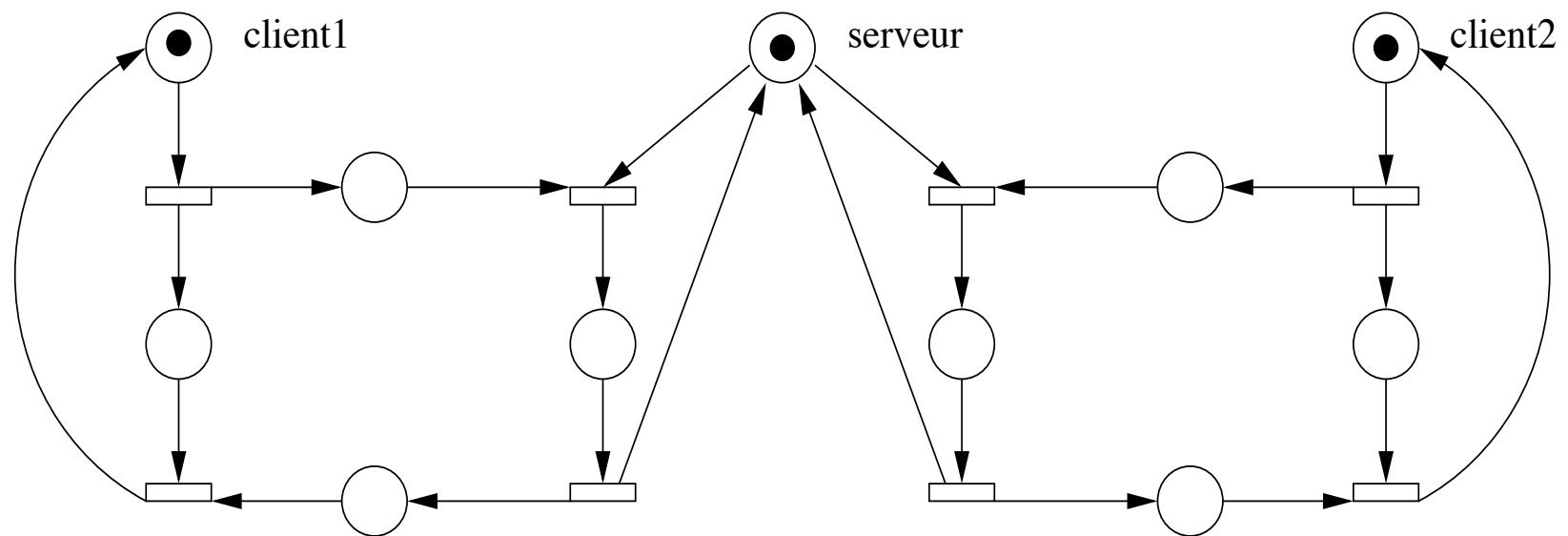


Ex. :



RdP colorés

- Exemple : 2 clients et 1 serveur





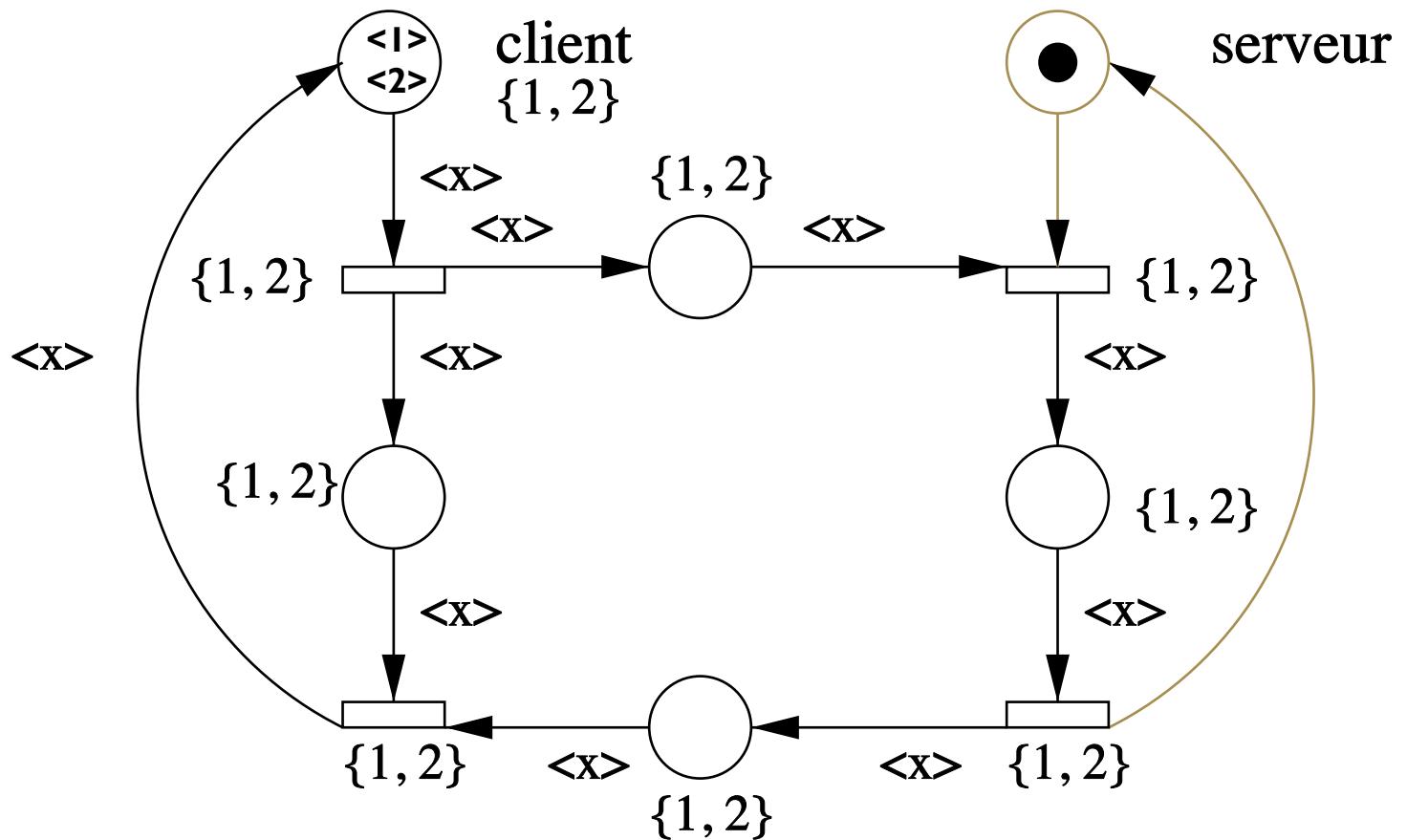
RdP colorés

Pour gérer de façon différenciées les 2 clients et le serveur

- Les jetons peuvent être différenciés
 - <1>, <2> pour les clients
 - « classique » pour le serveur

RdP colorés

- Modèle client-serveur avec RdP coloré
 - Arcs, places et transitions peuvent être associés à type de jetons





RdP Temporisés

- Transitions franchissable dans un certain intervalle de temps $[i,j]$
 - Les jetons sont consommés
 - Puis le temps passe (entre i et j sec)
 - Puis les jetons sont produits
- Horloge associée au simulation

RdP Récursifs

- Les transitions sont typées et la sémantique de leur franchissement dépend du type de transition
 - Transition élémentaire : franchissement ordinaire
 - Transition de fin : franchissement ferme le réseau courant
 - Transition abstraite : raffinement par un sous-réseau marqué

