Sorbonne Université - Master Informatique
parcours STL : Paradigmes de Programmation Concurrente MU5IN553

Cours 1 - Introduction à la concurrence
Modélisation avec des résaux de Petri

Carlos Agon (agonc@ircam.fr) - Romain Demangeon

18 Septembre 2024

Le "ou" concurrent, G. Berry

- or (e_1 , e_2) Evalue les deux
- orL (e_1 , e_2) Evalue les e_1 puis e_2
- orD (e_1 , e_2) Evalue les e_2 puis e_1

Peut on faire ouP (e_1 , e_2) Vrai si l'un des deux est vrai ?

Le "ou" concurrent, G. Berry

- or (e_1 , e_2) Evalue les deux
- orL (e_1 , e_2) Evalue les e_1 puis e_2
- orD (e_1 , e_2) Evalue les e_2 puis e_1

Peut on faire ouP (e_1 , e_2) Vrai si l'un des deux est vrai ?

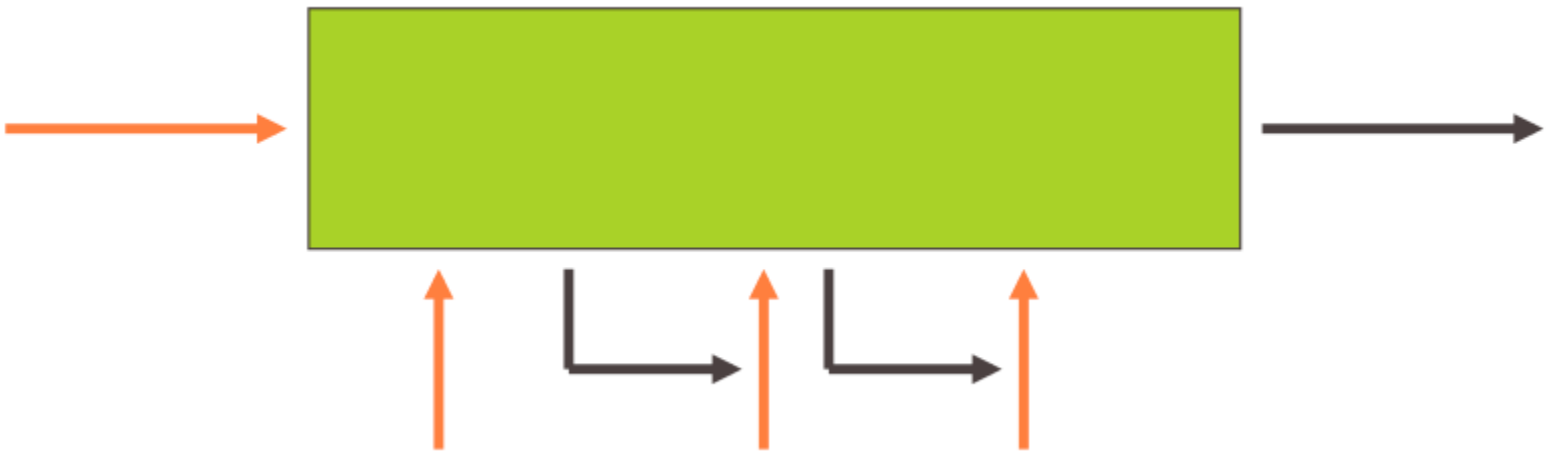
Pas en mode séquentiel

Interaction

Un système interactif est une application informatique qui prend en compte, au cours de son exécution, d'informations communiquées par le ou les utilisateurs du système, et qui produit, au cours de son exécution, une représentation perceptible de son état interne.

Interaction

Un système interactif est une application informatique qui prend en compte, au cours de son exécution, d'informations communiquées par le ou les utilisateurs du système, et qui produit, au cours de son exécution, une représentation perceptible de son état interne.

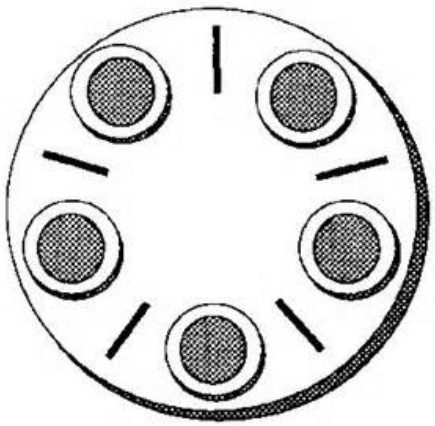


Historique

E. W. Dijkstra. Solution of a problem in concurrent programming control.

Communications of the ACM, 8(9) :569, September 1965.

Les dîner des philosofes



Interblocage ou Famine

Quelque problèmes classiques

- **The dining savages problem** : *The Little Book of Semaphores, Allen B. Downey*
  A tribe of savages eats communal dinners from a large pot that can hold *m* servings of stewed missionary. When a savage wants to eat, he helps himself from the pot, unless it is empty. If the pot is empty, the savage wakes up the cook and then waits until the cook has refilled the pot.
- **The barbershop problem** : *Operating Systems Concepts, Silberschatz and Galvin*
  A barbershop consists of a waiting room with *n* chairs, and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

Quelque problèmes classiques

- **The Santa Claus problem** : *Operating Systems, William Stallings*
  Santa Claus sleeps in his shop at the North Pole and can only be awakened by either (1) all nine reindeer being back from their vacation in the South Pacific, or (2) some of the elves having difficulty making toys ; to allow Santa to get some sleep, the elves can only wake him when three of them have problems. When three elves are having their problems solved, any other elves wishing to visit Santa must wait for those elves to return. If Santa wakes up to find three elves waiting at his shop's door, along with the last reindeer having come back from the tropics, Santa has decided that the elves can wait until after Christmas, because it is more important to get his sleigh ready. (It is assumed that the reindeer do not want to leave the tropics, and therefore they stay there until the last possible moment.) The last reindeer to arrive must get Santa while the others wait in a warming hut before being harnessed to the sleigh.

Comment décrire le comportement d'un système ? avec quel langage

"The only mathematics we need to describe computations are sets, functions, and simple predicate logic." (L. Lamport Teaching Concurrency 2009)

"It is more useful to think about states than sequences of steps because what a computing device does next depends on its current state, not on what steps it took in the past. Computer engineers should learn to think about a computation in terms of states rather than verbs."

C'est quoi la programmation concurrente ?

- Modeliser ou Calculer ?
- Resultat ou invariance ?
- Specifier et verifier
- Determinisme vs non determinisme
- Enrichir le séquentiel pour faire du parallèle ou partir de primitives parallèles ?

Questions de base

- C'est quoi un processus ?
- Comment communiquent les processus ?
  - *Shared variables*
  - *Message passing*
  - *Handshaking*
- Comment se synchronisent les processus ?
  - Synchrone
  - Asynchrone
  - GALS

Déroulement du cours

1. Introduction (Cours 1)
   - Reseaux de Petri
2. Algebre de processus (Cours 2-7)
   - CCS
   - Bisimulation
   - μ-calcul
   - TCCS
   - π-calcul
3. Programmation asynchrone (Cours 8-11)
   - AKKA le model actors
4. Programmation synchrone (Cours 12-14)
   - Lustre, SCADE

Evaluation

1. Evaluation Partielle 40%
2. Projet AKKA | Synchrone 20%
3. Evaluation final 40%

Outils de simulation et vérifaction

Besoin d'outils pour suivre le fonctionnement du système et vérifier certaines propriétés :

- L'activité : si le fonctionnement d'une partie ou de tout le système évolue.
- La répétitivité : s'il y a des séquences qui se répètent.
- La bornitude : quand une partie ou tout le système n'évolue plus, une fois qu'un état spécifique est atteint.
- La vivacité : Vérifie qu'un état du système puisse être atteignable, quel que soit l'état dans lequel il se trouve.

L'un de modèles le plus connu sont les réseaux de Petri.

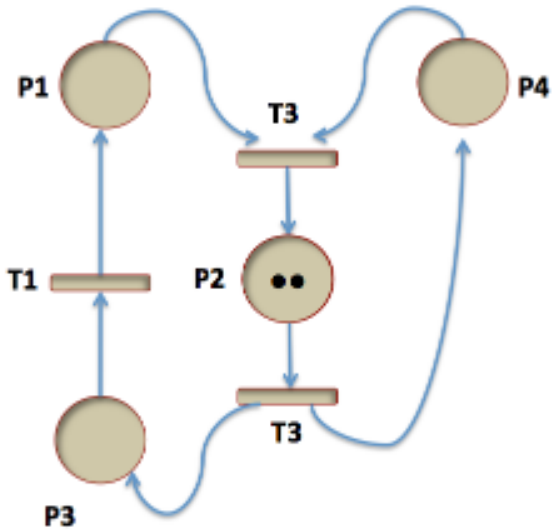
Réseaux de Petri

Proposés par Carl Adam Petri (1962)
Utilisés pour la modélisation de systèmes interactifs.

Un réseau de Petri (rdP) est un graphe biparti orienté :

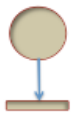
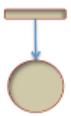
- **Places** (variable d'état du système)
- **Transitions** (événement ou action)
- **Jetons** (valeur de la variable)
- **Arcs**
  - la valeur **p** influence l'occurrence de l'action associée à **t**
  - l'occurrence de l'action associée à **t** influence la valeur de **p**

Réseaux de Petri

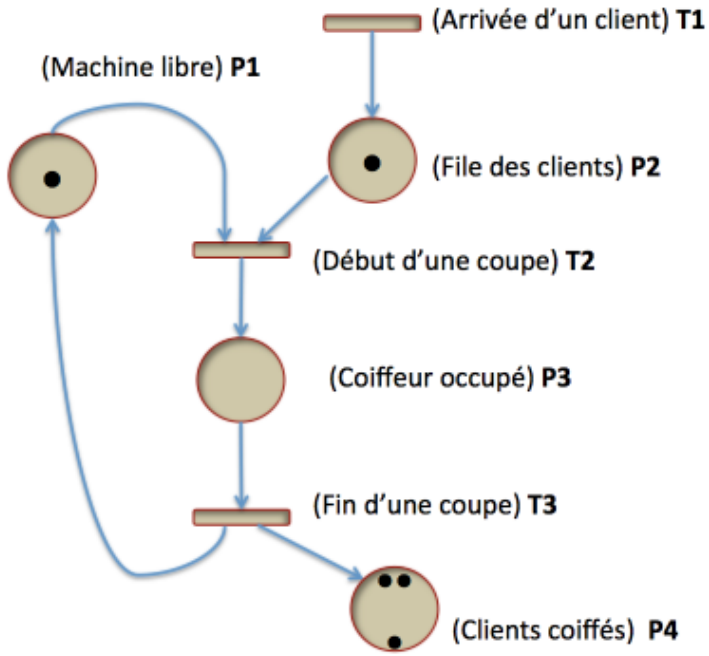


Un arc relie une place à une transition ou une transition à une place, mais jamais une place à une place ou une transition à une transition

Cas particuliers

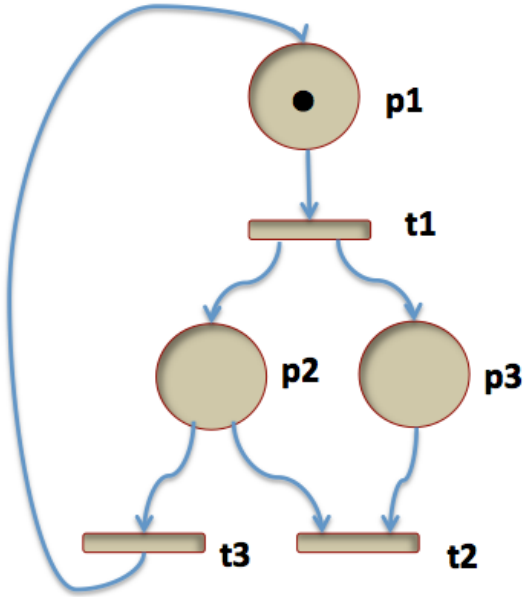
-  **transition puits** (pas de **p** en sortie de **t**)
-  **transition source** (pas de **p** en entrée de **t**)

Exemple informel

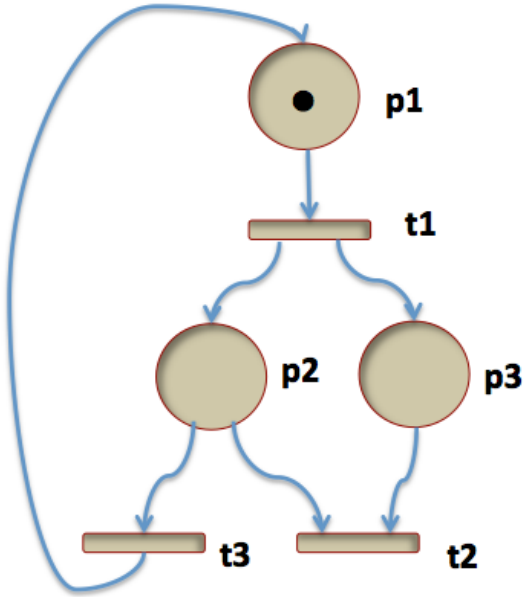


- Les places sont les variables d'état du système
- Les transitions sont des actions qui font évoluer l'état du système
- Les jetons sont indiscernables et interprétables par rapport à la place

Franchissement

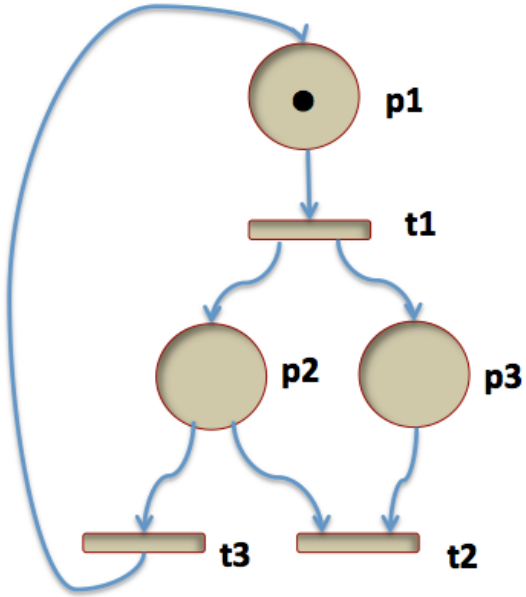


Franchissement



- $t_1 \rightarrow t_2$

Franchissement

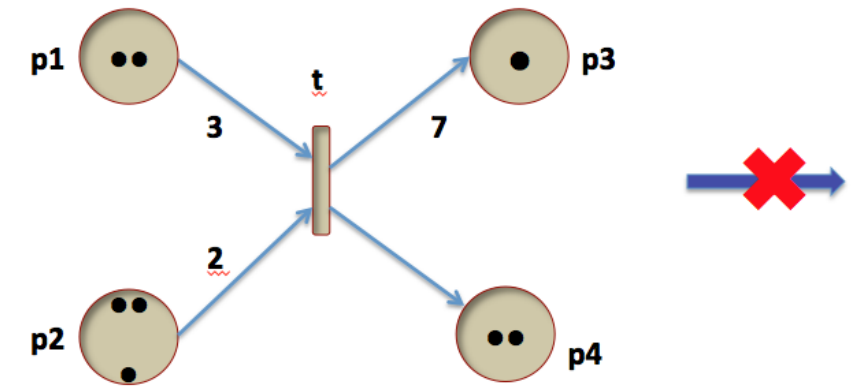


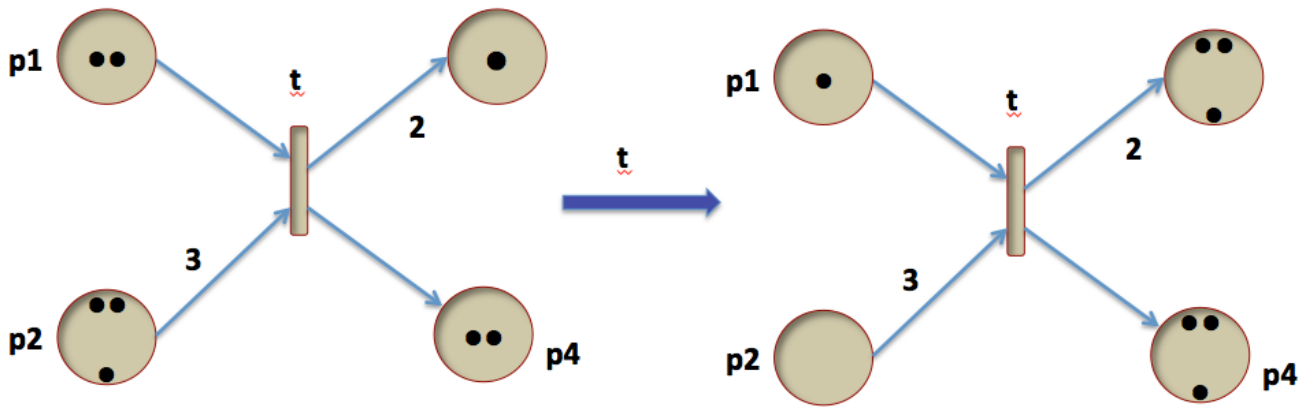
- $t_1 \rightarrow t_2$

ou

- $t_1 \rightarrow t_3 \rightarrow t_1 \rightarrow t_2$

Exemple





Définition Réseau de Petri généralisé

définition

Un rdP est un graphe biparti défini par (P,T, Pre, Post), avec :

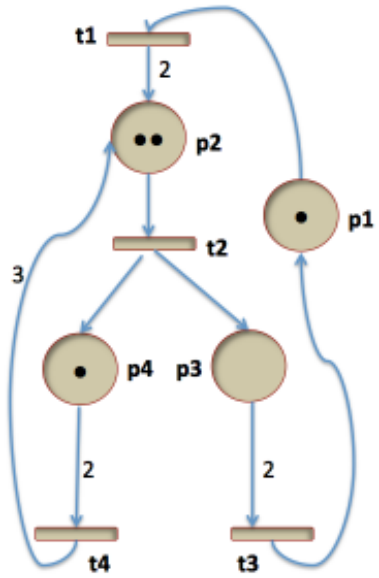
- $P = \{p_1, p_2, ..., p_n\}$
- $T = \{t_1, t_2, ..., t_m\}$
- Pre : $P \times T \rightarrow \mathbb{N}$ (de P vers T)
- Post : $P \times T \rightarrow \mathbb{N}$ (de T vers P)

Un marquage est un vecteur avec le nombre de jetons pour chaque place à un instant donné.
M_0 dénote le marquage initial

définition

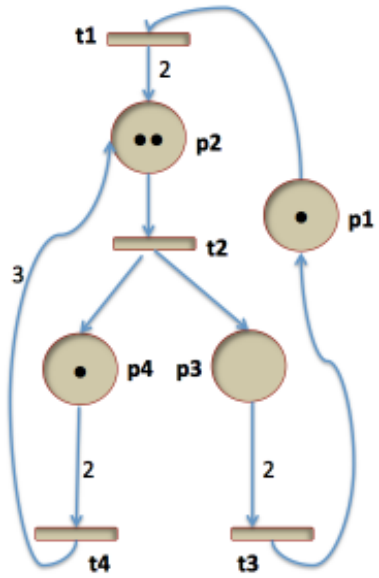
Un rdP marqué = (P,T,Pre, Post, M_0)

Exemple



lines = etats et cols = transitions

Exemple



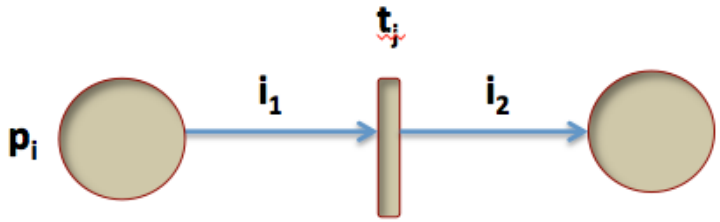
lines = etats et cols = transitions

$$Pre = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \quad Post = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad M_0 = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

Evolution du marquage

- L'évolution d'un rdP correspond à l'évolution du M_0
- Les jetons peuvent passer d'une place à l'autre par franchissement d'une transition
- Une transition est franchissable (valide), si les places d'entrée possèdent un nombre de jetons $\geqslant$ au poids des arcs entre chaque place et la transition
- Franchir une transition consiste à enlever à chaque place d'entrée le nombre de jetons égal au poids de l'arc d'entrée et à déposer dans la place de sortie le nombre de jetons de son arc d'entrée.
- Lorsqu'une transition est franchissable, cela n'implique pas qu'elle sera franchie tout de suite. Le réseau évolue en franchissant une seule transition à la fois parmi l'ensemble de transitions valides.

Matrice d'incidence



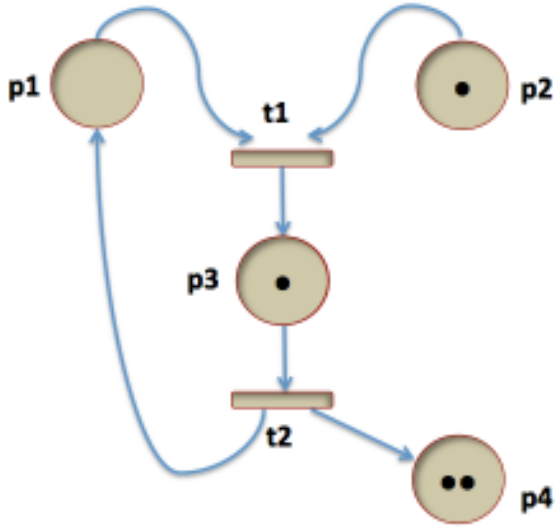
- Condition nécessaire : $M(p_i) \geqslant \text{Pre}(p_i, t_j)$ où M est le marquage courant
- Franchir t_j produit un nouveau marquage M' tq. $\forall p_i \in P, M'(p_i) = M(p_i) - \text{Pre}(p_i, t_j) + \text{Post}(p_i, t_j)$

Définition

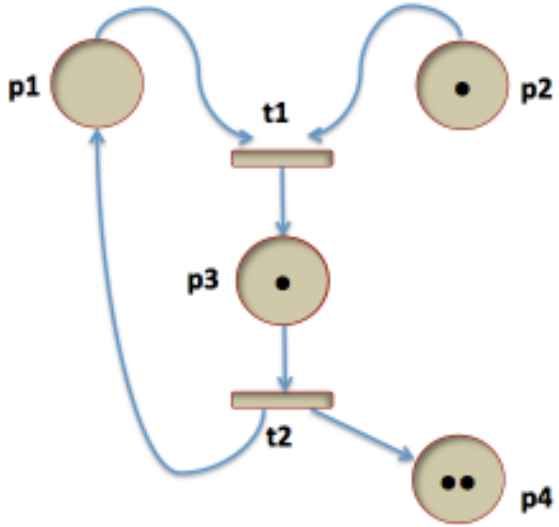
On défini la matrice d'incidence W par :
$W(p_i, t_j) = \text{Post}\ (p_i, t_j) - \text{Pre}(p_i, t_j) \qquad \forall p_i \in P \forall t_i \in T$

Un nouveau marquage M' à partir du Marquage M par franchissement de la transition t_i est obtenu par : $\text{M'} = \text{M} + \text{W} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow t_j$ (notation $\text{M}[t_j > \text{M'}$)

Algebre

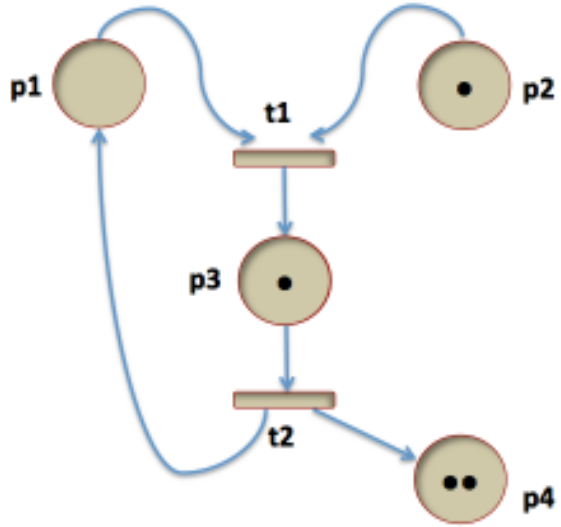


Algebre



$$M_0 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} \quad Pre = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad Post = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad W = Post - Pre = \begin{pmatrix} -1 & 1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}$$

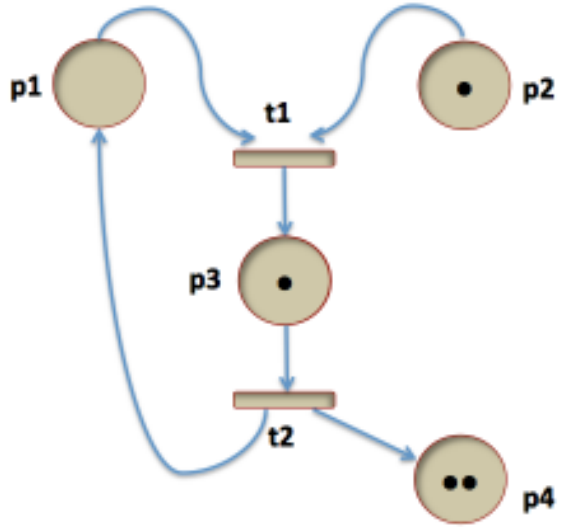
Algebre



$$M_0 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} \quad Pre = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad Post = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad W = Post - Pre = \begin{pmatrix} -1 & 1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}$$

$$M' = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \end{bmatrix}$$

Algebre



$$M_0 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} \quad Pre = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad Post = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad W = Post - Pre = \begin{pmatrix} -1 & 1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}$$

$$M' = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \end{bmatrix} \quad M'' = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

Sequence de franchissement

$M[t_1 > M_1[t_2 > M_2 \Rightarrow M[t_1 t_2 > M_2$
(notation $\sigma = t_1 t_2$ et $M[\sigma > M_2$)

Definition : vector caracteristique

Un vecteur avec le nombre d'occurrences de chaque transition dans une séquence σ.
Pour $T = \{t_1, t_2, t_3, t_4\}$

Si $\sigma = t_1 t_2 t_3$ alors $\vec{\sigma} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Sequence de franchissement

$M[t_1 > M_1[t_2 > M_2 \Rightarrow M[t_1 t_2 > M_2$
(notation $\sigma = t_1 t_2$ et $M[\sigma > M_2$)

Definition : vector caracteristique

Un vecteur avec le nombre d'occurrences de chaque transition dans une séquence σ.
Pour $T = \{t_1, t_2, t_3, t_4\}$

Si $\sigma = t_1 t_2 t_3$ alors $\vec{\sigma} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Si $\sigma = t_1 t_2 t_2 t_1 t_2$ alors $\vec{\sigma} = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 0 \end{bmatrix}$

Sequence de franchissement

$M[t_1 > M_1[t_2 > M_2 \Rightarrow M[t_1 t_2 > M_2$
(notation $\sigma = t_1 t_2$ et $M[\sigma > M_2$)

Definition : vector caracteristique

Un vecteur avec le nombre d'occurrences de chaque transition dans une séquence σ.
Pour T $= \{t_1, t_2, t_3, t_4\}$

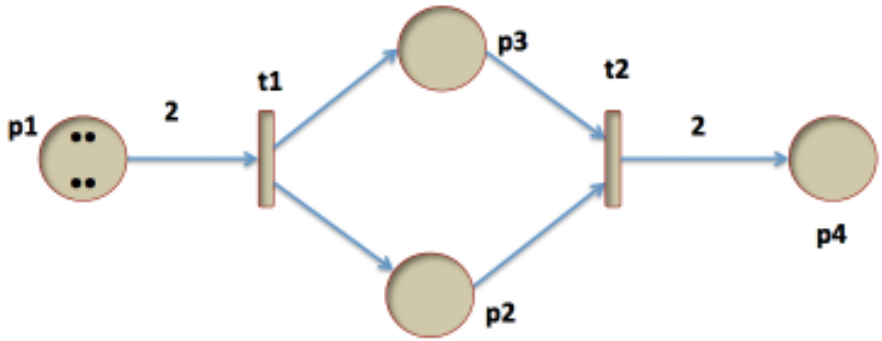
Si $\sigma = t_1 t_2 t_3$ alors $\vec{\sigma} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Si $\sigma = t_1 t_2 t_2 t_1 t_2$ alors $\vec{\sigma} = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 0 \end{bmatrix}$

Equation d'état

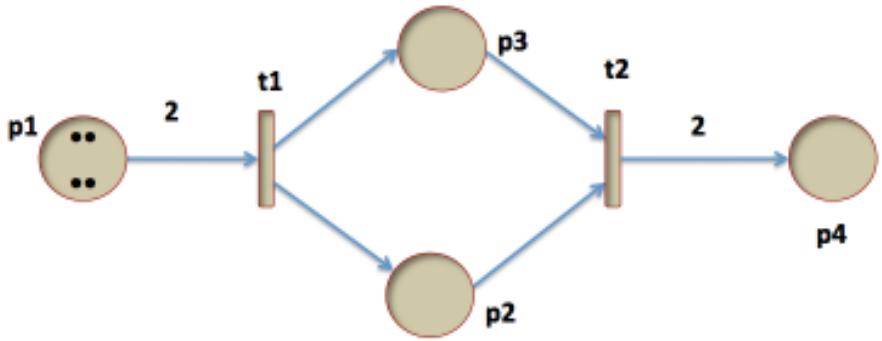
$M[\sigma > M_1 \Rightarrow M_1 = M + W\vec{\sigma}$

Sequence de franchissement



Soit $\sigma = t_1 t_2 t_1$ calculer M tq. $M_0[\sigma > M$

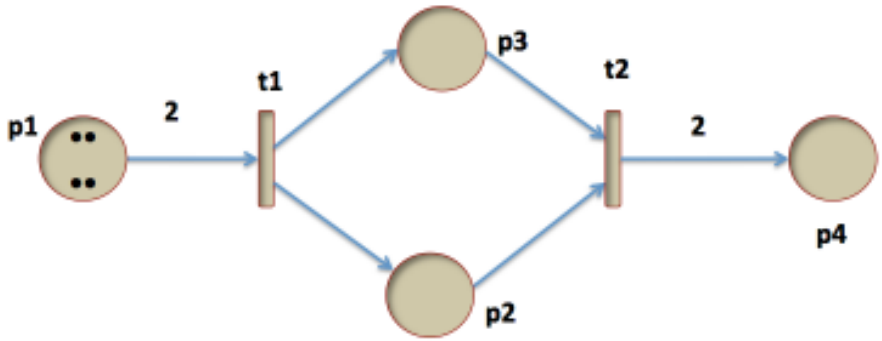
Sequence de franchissement



Soit $\sigma = t_1 t_2 t_1$ calculer M tq. $M_0[\sigma > M$

$$M_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

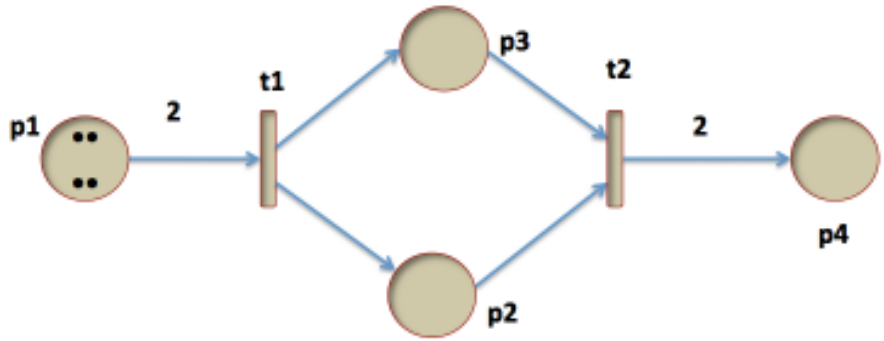
Sequence de franchissement



Soit $\sigma = t_1 t_2 t_1$ calculer M tq. $M_0[\sigma > M$

$$M_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{\sigma} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

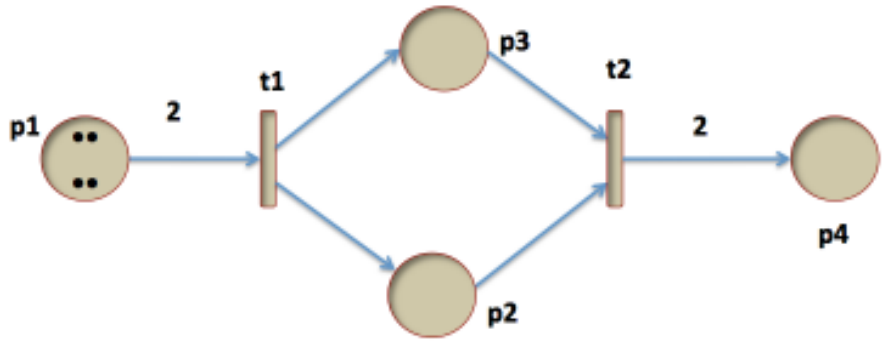
Sequence de franchissement



Soit $\sigma = t_1 t_2 t_1$ calculer M tq. $M_0[\sigma > M$

$$M_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{\sigma} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{Pre} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{Post} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 2 \end{bmatrix}$$

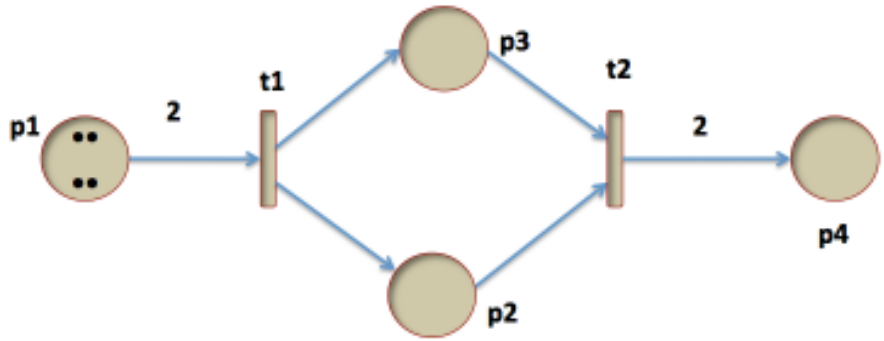
Sequence de franchissement



Soit $\sigma = t_1 t_2 t_1$ calculer M tq. $M_0[\sigma > M$

$$M_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{\sigma} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{Pre} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{Post} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 2 \end{bmatrix} \quad W = \begin{bmatrix} -2 & 0 \\ 1 & -1 \\ 1 & -1 \\ 0 & 2 \end{bmatrix}$$

Sequence de franchissement

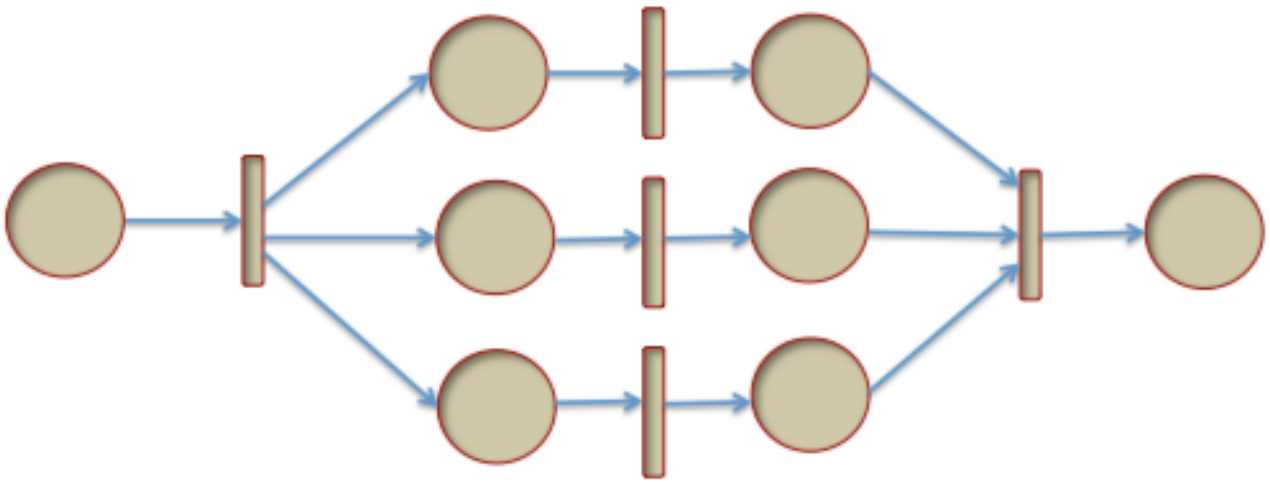


Soit $\sigma = t_1 t_2 t_1$ calculer M tq. $M_0[\sigma > M$

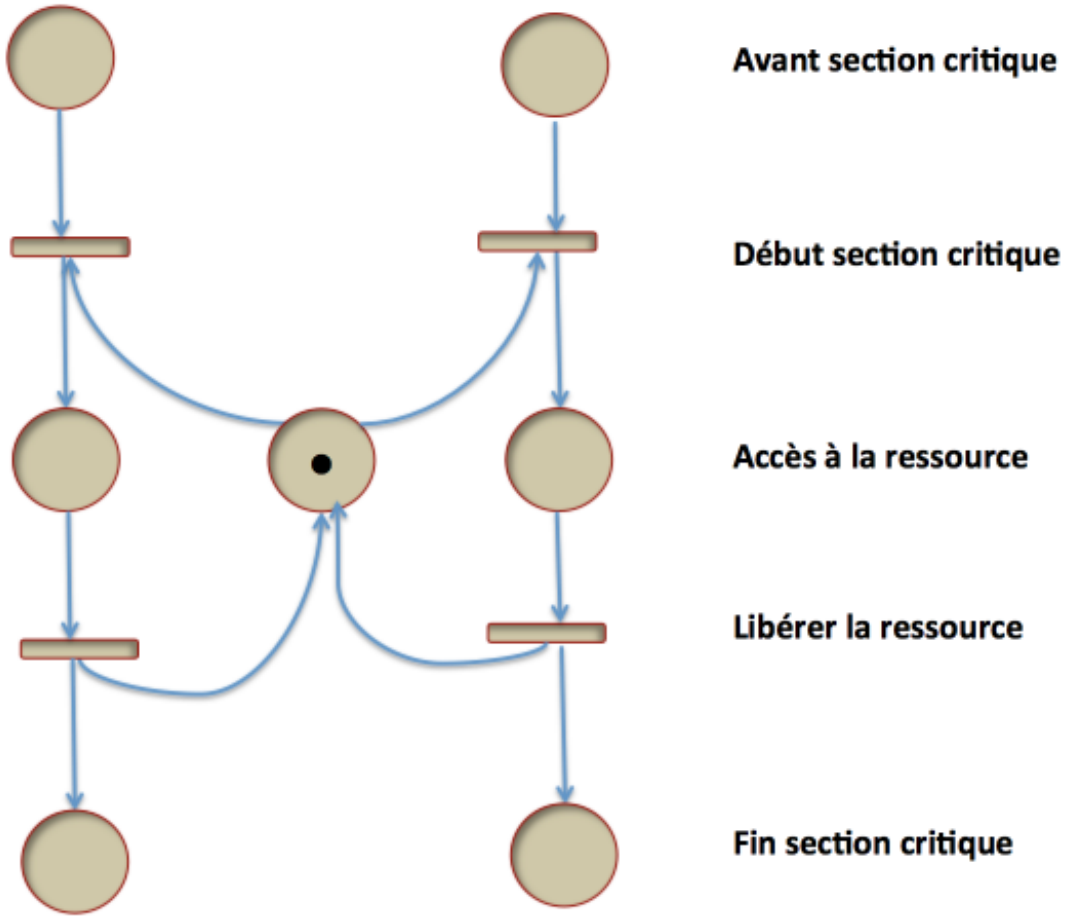
$$M_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \vec{\sigma} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{Pre} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \text{Post} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 2 \end{bmatrix} \quad W = \begin{bmatrix} -2 & 0 \\ 1 & -1 \\ 1 & -1 \\ 0 & 2 \end{bmatrix}$$

$$M = M_0 + W\vec{\sigma} = M_0 = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 0 \\ 1 & -1 \\ 1 & -1 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -4 \\ 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}$$

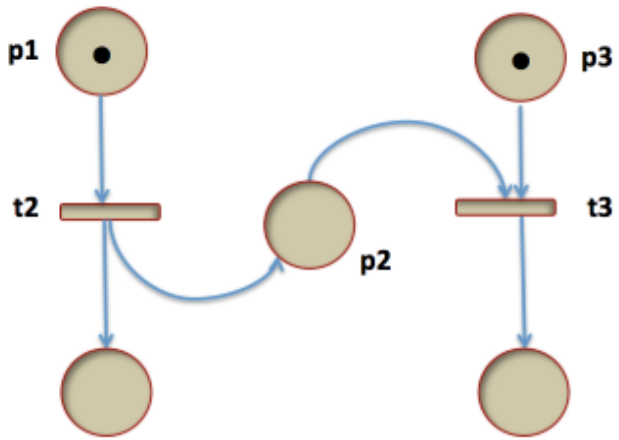
Parallèlisme et synchronisation



Partage de ressources



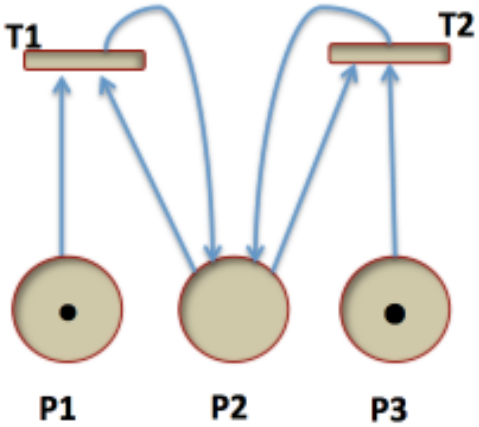
Communication asynchrone



Blocage

Definition : blocage

Un blocage est un marquage pour lequel il n'y a pas de transition valide.
Un rdP est dit sans blocage pour M_0 si il n'y a pas de marquage dans $*M_0$ qui soit un blocage.



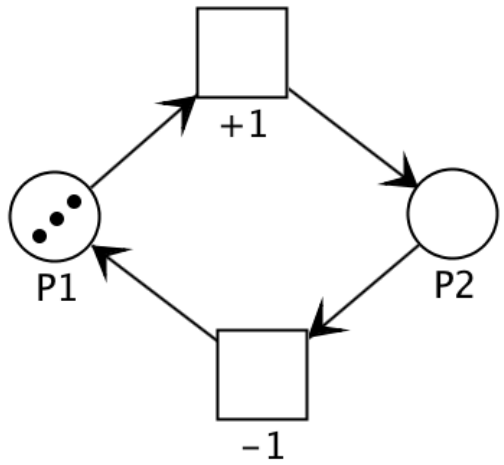
$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

Exemples - Un ascenseur

Le nombre de jetons représente le numéro d'étage.

Exemples - Un ascenseur

Le nombre de jetons représente le numéro d'étage.

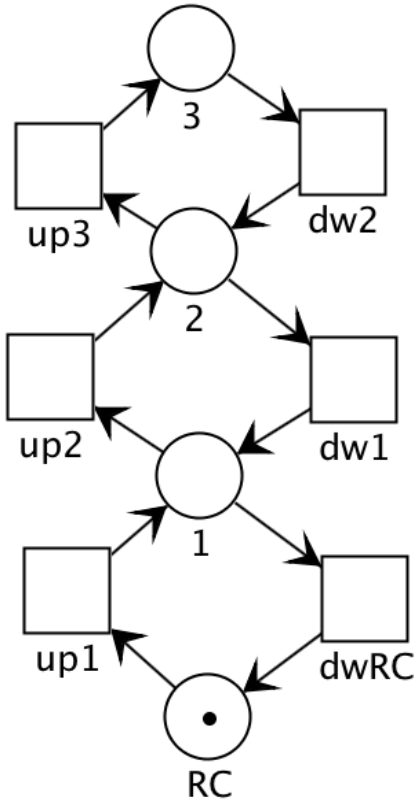


Un ascenseur (2)

Une place représente un étage et le jeton indique si l'ascenseur est présent ou non.

Un ascenseur (2)

Une place représente un étage et le jeton indique si l'ascenseur est présent ou non.

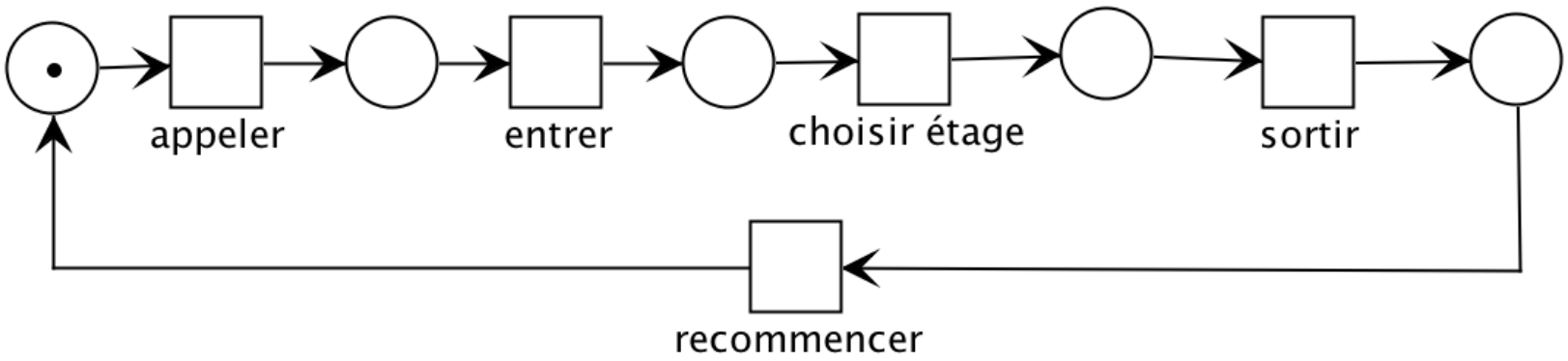


Un ascenseur (3)

Le jeton est une personne utilisant l'ascenseur

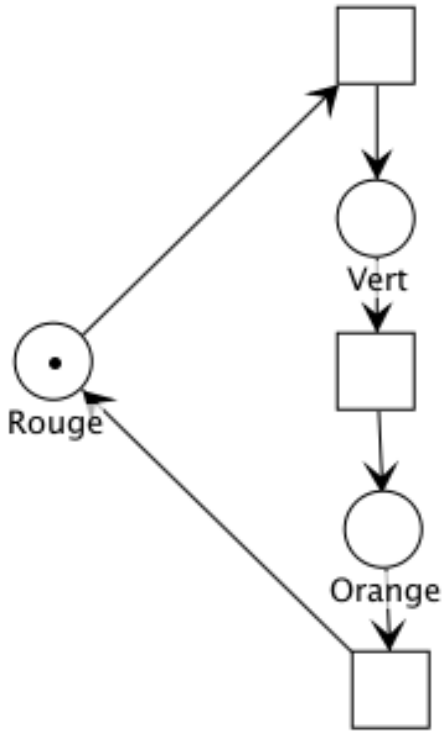
Un ascenseur (3)

Le jeton est une personne utilisant l'ascenseur



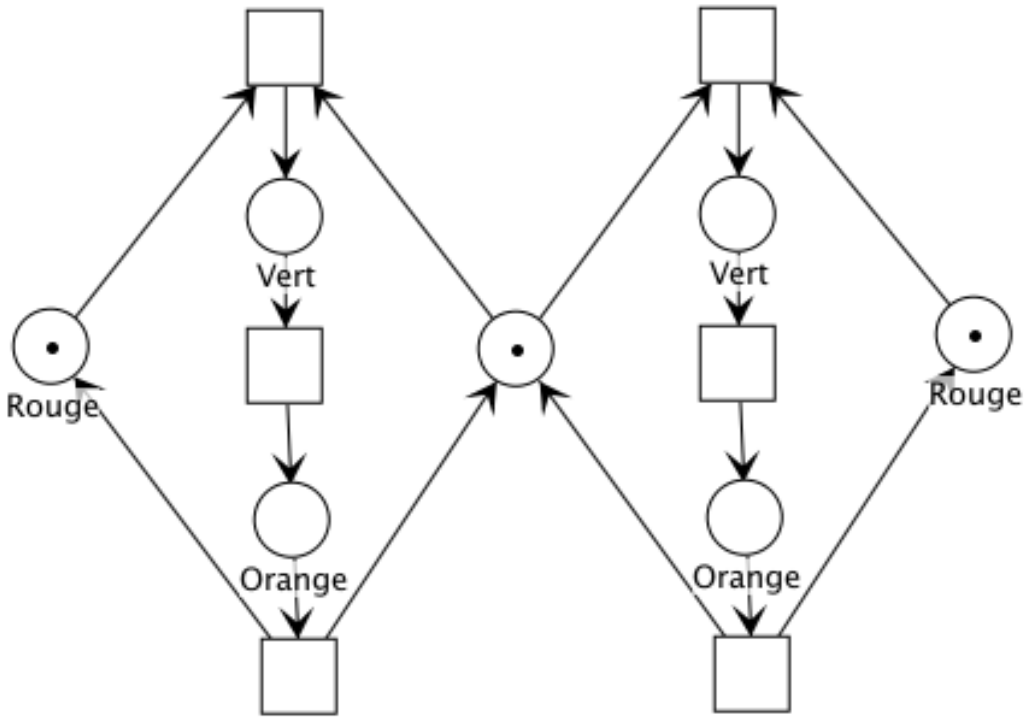
Un feu rouge

Un feu rouge



Deux feux rouges

Deux feux rouges



Références

Petri Nets

- Slides de Vincent Augusto ESM Saint-Etienne
- Les Réseaux de Petri. Notes de cours par N. Bennis
- Plein de pages sur le net, en particulier
  `www.informatik.uni-hamburg.de/TGI/PetriNets/`