Modèles du parallélisme et les langages FDT

(Formal Description Technics)

Option: Systèmes parallèles et distribués

20 heures

Par Dr GUEZOULI Larbi

Modèles du parallélisme et les langages FDT (Formal description techniques)

- Chapitre I: Introduction
  - Modèles formels
  - Utilisation dans le parallélisme
- Chapitre II: Modèles du parallélisme
  - Les systèmes de transition d'états
  - Les automates à états fini
    - Définition, comportement, propriétés
  - Les réseaux de Petri
  - Les dérivés des réseaux de Petri
    - Rdp colorés;
    - Rdp temporises;
    - Rdp prédicats/transition;
  - Méthodes de réductions d'un RDP
- Chapitre III: Domaines d'application (des exposés)
  - Le langage ESTELLE
    - Spécification Estelle, structuration d'un module, en-tête de module...
  - LOTOS de base (Basic Language Of Temporal Ordering Specification)
  - LOTOS complet (Full LOTOS)
  - E-LOTOS (Enhancements to LOTOS - Révision du lotos standard -)
  - Introduction à CCS (Calculus of Communicating Systems)
  - Présentation de Act-one

Chapitre I

<u>Introduction</u>

I.1. Modèles formels

I.2. Utilisation dans le parallélisme

Exemple RAM, PRAM et BSP

I.1. Les modèles formels

- Le but d'un modèle formel:
  - spécifier
  - concevoir
  - vérifier
  - implémenter
  - et tester divers systèmes.
- Point fort:
  - La fiabilité

I.1. Les modèles formels

- Utilisation:
  - dans les systèmes critiques (vie humaine en jeux)
- Les modèles formels sont également utilisés comme support de raisonnement sur les propriétés et le comportement des systèmes.

I.1. Les modèles formels

- Approches essentielles:
  - Modèles basés sur les systèmes de transition d'états
  - Modèles d'interacteurs formels
  - …
- Pour notre cours: les modèles basés sur les systèmes de transition d'états.

I.1. Les modèles formels

<u>Les modèles basés sur les systèmes de transition d'états</u>

- Un système de transition décrit des comportements. Ces systèmes sont presque tous des extensions:
  - soit des automates à nombre fini d'états
  - soit des réseaux de Petri

I.1. Les modèles formels

<u>Introduction aux automates à états finis</u>

Les automates à nombre fini d'états sont représentés graphiquement par des *diagrammes de transition d'états*, graphes orientés dont les noeuds sont des ***états*** et les arcs des ***transitions***

I.1. Les modèles formels

Introduction aux automates à états finis (suite)

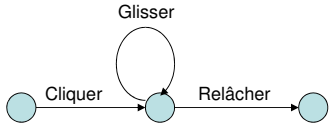


Diagramme de transition décrivant la gestion de l'opération cliquer-glisser d'une souris

- Un **état** est un ensemble de valeurs qui caractérise le système à un moment donné dans le temps
- Une **transition** d'état est une relation entre deux états indiquant un changement d'état possible

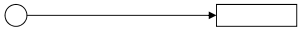
I.1. Les modèles formels

Introduction aux réseaux de Petri

Les *réseaux de Petri* sont une généralisation des automates à états.

Un réseau de Petri est un graphe biparti alterné qui possède deux types de noeuds: les *places* (cercles) et les *transitions* (rectangles)

Des *arcs* (flèches) relient les places aux états

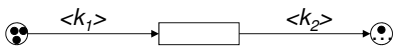


L'état du système, nommé *marquage*, est défini par la répartition de *jetons* (petits cercles foncés) dans les places

I.1. Les modèles formels

Introduction aux réseaux de Petri (suite)

- Une transition est ***franchissable*** sous certaines conditions, notamment lorsque suffisamment de jetons sont présents dans ses places d'entrée.
- Le franchissement d'une transition se traduit par une modification du marquage consistant la plupart du temps en un « déplacement » de jetons



Si <k_1> jetons existent dans la place de départ, la transition sera franchie et génère <k_2> jetons dans la place de sortie.

I.2. Utilisation dans le parallélisme

- But de la modélisation
  - Simplicité
  - Expressivité
  - Prédictibilité (prédire l'évolution du système)
  - Faisabilité
  - Portabilité
  - Efficacité
- Exemple RAM et PRAM

I.2. Utilisation dans le parallélisme

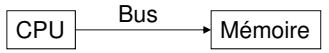
- Il existe 3 sous modèles
  - Modèle d'architecture (concerne le matériel. exp. gestion de l'utilisation de plusieurs bus en parallèle)
  - Modèle algorithmique (synchronisation)
  - Modèle de coût (ressources à accès multiple utilisable en parallèle)

I.2. Utilisation dans le parallélisme

- Exemple: Modèle RAM (non parallèle)

(Random Access Memory Machine) de Von Neumann 1950

- Sous-modèle architectural



- Sous-modèle algorithmique

Instructions classiques (addition, soustraction…)

- Sous-modèle de coût

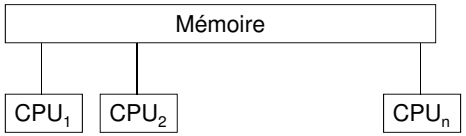
Le coût d'une opération est calculé à base du coût de l'addition qui est l'opération de base.

I.2. Utilisation dans le parallélisme

- Exemple: Modèle PRAM (parallèle)

Extension du modèle RAM pour le parallélisme 1982

- Sous-modèle architectural



I.2. Utilisation dans le parallélisme

- Sous-modèle algorithmique
  - Nombre arbitraire de processeurs;
  - Mémoire partagée par les processeurs;
  - Processeurs avec jeux habituel d'instructions;
  - Même horloge pour tous les processeurs.
- Sous-modèle de coût
  - Même coût pour toutes les instructions;
    (lecture, écriture, traitement)
  - Évaluation du nombre total d'instructions;
  - Évaluation du nombre total de cycle d'horloge.

I.2. Utilisation dans le parallélisme

- Exemple en pratique: OR parallèle

Soit un vecteur v[] de n valeurs booléennes stocké dans une mémoire PRAM

Est-ce qu'une de ces valeurs au moins est « true »

```
resultat ← false
pour tout i dans {0, … , n-1} faire en parallèle
     si v[i]=true alors resultat ← true
retourner resultat
```

Analyse:

Coût : O(n) n accès en parallèle à la mémoire

Temps: O(1) Le temps d'une opération

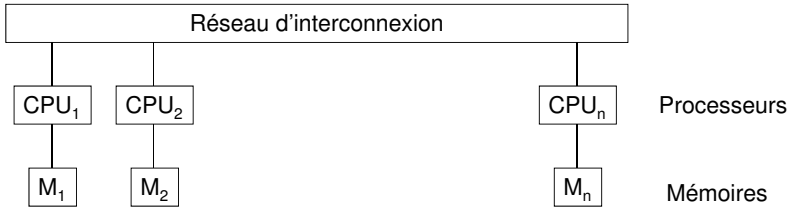
Problème: Accès simultané à la variable « resultat » !!!

I.2. Utilisation dans le parallélisme

- Exemple: Modèle à gros grains

Modèle BSP (Bulk Synchronous Parallel machine). Valiant 1990

Sous-modèle architectural:



I.2. Utilisation dans le parallélisme

Sous-modèle algorithmique:

Etapes:

1. Synchronisation des processus
2. Calcul local
3. Communication

Sous-modèle de coût:

$T_{total} = T_{local} + T_{comm} + T_{sync}$

Tel que:

T_{local} : Temps du calcul local

T_{comm} : Temps de communication = G*h + s

T_{sync} : Temps de synchronisation = temps d'un aller/retour sur le réseau

Avec s: temps d'initialisation d'une communication (socket, création de buffer…)

G: le débit du réseau

h: taille de la donnée à communiquer

Chapitre II

Modèles formels du parallélisme

II.1. Les systèmes de transition d'états

II.2. Les automates à états finis

Définition, comportement, propriétés

II.3. Les réseaux de Petri

II.4. Les dérivés des réseaux de Petri

II.4.1. Rdp colorés

II.4.2. Rdp temporises

II.4.3. Rdp prédicats/transition

II.5. Méthodes de réductions

II.1. Les systèmes de transition d'états

Les modèles formels qui nous **intéressent** sont ceux basés sur les systèmes de transition d'états

Les systèmes de transition d'états définissent un certain nombre de notations permettant de décrire les **différents comportements d'un système**.

II.1. Les systèmes de transition d'états

Il existe un grand nombre de systèmes de transition d'états.

Ces systèmes sont **presque** tous des **extensions**:

- soit des automates à nombre fini d'états fini
- soit des réseaux de Petri

II.2. Les automates à états finis

<u>Définition formelle d'un automate à états fini</u>

Un automate à nombre fini d'états est:

$$A = <E, \sum , e_0, E_f, \delta >$$

Tel que:

E : Ensemble fini d'états

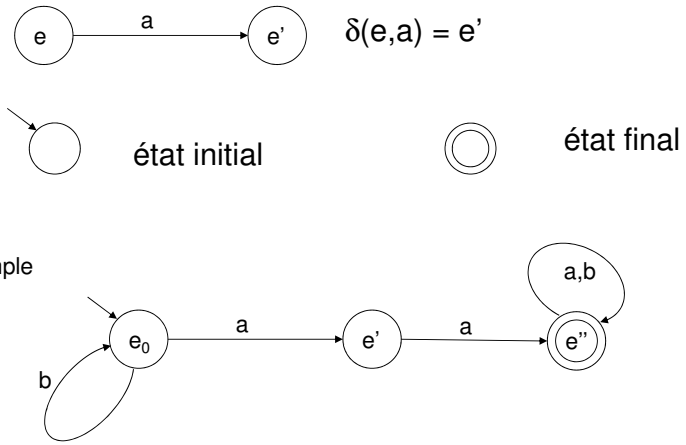
Σ : Ensemble fini et non vide d'alphabets (symboles)

$e_0 \in E \rightarrow$ L'état initial

$E_f \subseteq E \rightarrow$ L'ensemble des états finaux

δ fonction de transition de E x Σ dans E_f

II.2. Les automates à états finis

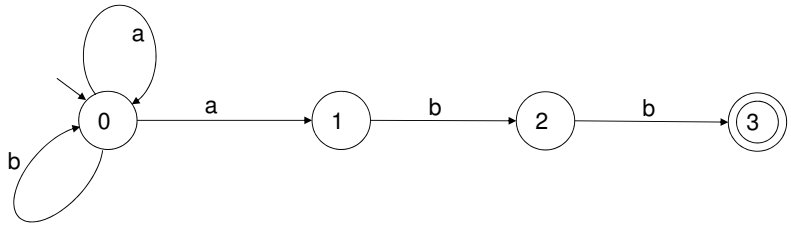


II.2. Les automates à états finis

- Exemple:

$E = \{0, 1, 2, 3\}$, $\sum = \{a, b\}$, $e_0 = 0$, $E_f = \{3\}$

Représentation graphique

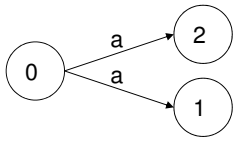


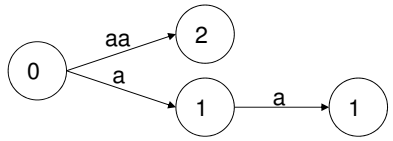
II.2. Les automates à états finis

- **Définitions:**

1) Un automate d'états fini est **déterministe** s'il n'y a pas à choisir entre 2 transitions.

Au contraire d'automate d'états fini **indéterministe**.





2) Le **langage reconnu par un automate** est l'ensemble des chaînes qui permettent de passer de l'état initial à un état terminal

3) L'automate de l'exemple précédant accepte le langage (l'expression régulière)

$$(a|b)^*abb$$

II.2. Les automates à états finis

- Élimination du non-déterminisme:
  - **Définition**:

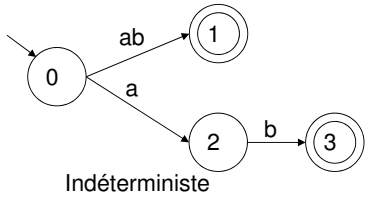
    Deux automates A_1 et A_2 sont équivalents s'ils acceptent le même langage, c'est-à-dire si $L(A_1) = L(A_2)$.

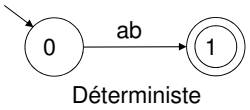
II.2. Les automates à états finis

- Élimination du non-déterminisme:
  - **Théorème (Rabin-Scott)**:

    Si le langage **L** est accepté par un automate fini non déterministe alors **L** est accepté par un automate fini déterministe.

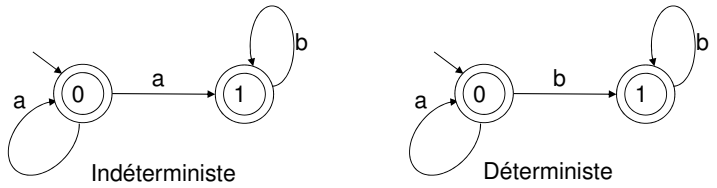
    Exp. 1





II.2. Les automates à états finis

Exp.2



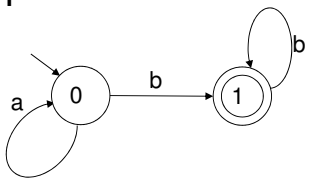
- Remarques:
  - E_f peut être **plein** $E_f = E$ et peut être **vide** $E_f = \varnothing$
  - E n'est **jamais vide** car il existe toujours l'état initial
  - δ est une fonction **partielle** parce qu'elle peut ne pas être définie pour tous les couples formés d'un état et d'un symbole

II.2. Les automates à états finis

Un mot x est **accepté ou reconnu** si l'état $\delta(e_0,x)$ est un état final.

Un langage accepté est constitué par un ensemble de mots accepté par l'automate

Exemple



Langage accepté
a*bb*

II.3. Les réseaux de Petri

Domaines d'application

- Systèmes de production (Évaluation des performances, Simulation)
- Validation des protocoles de communication
- Systèmes temps réel / distribués
- Modèles de raisonnement / planification

II.3. Les réseaux de Petri

Définition

Un réseau de Petri est un **ensemble d'automates** à états fini qui **communiquent**.

- communications **asynchrones** par échange de messages

- communication **synchrones** par rendez-vous, synchronisations, ressources partagées

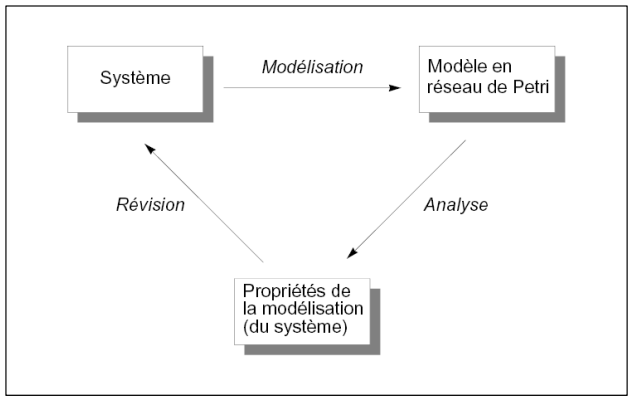
II.3. Les réseaux de Petri

Présentation informelle

- Un RDP est une **représentation mathématique** permettant la **modélisation** d'un système
- L'analyse d'un réseau de Petri peut révéler des caractéristiques concernant le **comportement dynamique** du système
- Les résultats de l'analyse d'un RDP permettent **d'évaluer** le système pour modification ou amélioration

II.3. Les réseaux de Petri

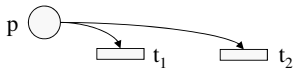
Démarche générale



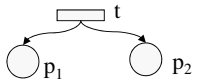
II.3. Les réseaux de Petri

Concepts de base

- Condition = Place
- Événement = Transition
- Pré-condition = arc Place → Transition



- Post-condition = arc Transition → Place



II.3. Les réseaux de Petri

Concepts de base (suite)

- Satisfaction d'une condition = Jeton dans une place

Remarque:

Une place peut contenir un nombre non borné de jetons

II.3. Les réseaux de Petri

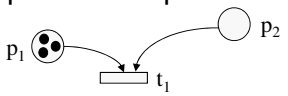
Concepts de base (suite)

- Condition
  - Une **condition** est une **description logique** (**prédicat**) d'un état du système.
  - Une condition est True ou False.
  - Un **état** du système peut être décrit comme un **ensemble de conditions**.
- Événement
  - Les **événements** sont des **actions** se déroulant dans le système.
  - Le **déclenchement** d'un événement **dépend** de l'état du système.
- Déclanchement, pré-condition, post-condition
  - Pour qu'un événement se **déclanche**, il faut que la **pré-condition** soit «true».
  - La **post-condition** sera vrais lorsque l'événement se déclanche. Les pré-conditions peuvent rester « true » comme elles peuvent être « false ».

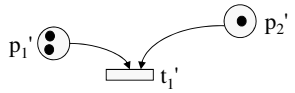
II.3. Les réseaux de Petri

Concepts de base (suite)

- Transition **franchissable** = satisfaction de toutes les places des pré-conditions de la transition

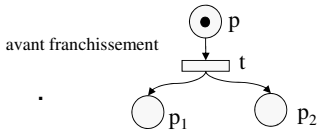


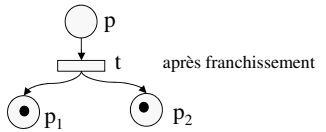
non franchissable



franchissable

- Effet du franchissement d'une transition = satisfaction de toutes les places post-conditions de la transition





II.3. Les réseaux de Petri

Modélisation avec ressources

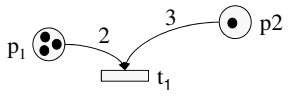
Modélisation avec ressources

- Dans la plus part des cas, **un jeton = une ressource**
- Le nombre de jeton dans une place = le nombre de ressources qu'elle possède
- Ces ressources sont consommées et produites par les événements du système
- Un arc peut être **valué** par un entier ≠ 0.
  - Dans la pré-condition, une **valuation** = nombre de jetons qui doivent être présent dans la place qui précède la transition
  - Dans la post-condition, une **valuation** = nombre de jeton qui seront produits après le franchissement de la transition.
  - Par défaut, les arcs sont valués par 1.

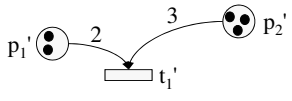
II.3. Les réseaux de Petri

Modélisation avec ressources

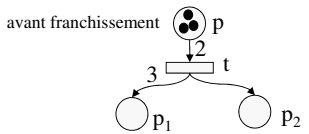
- Exemple

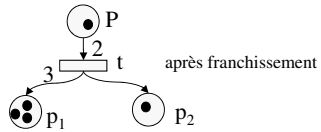


non franchissable



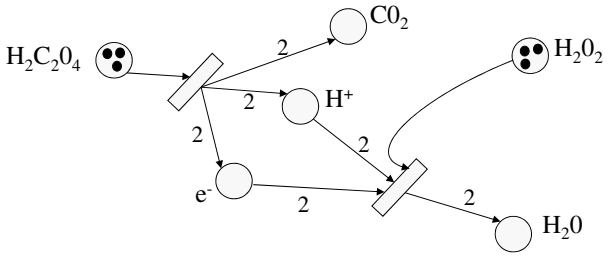
franchissable





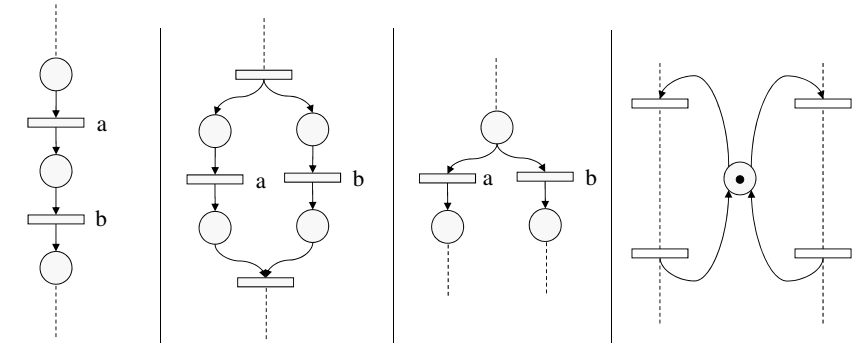
II.3. Les réseaux de Petri
Modélisation avec ressources

– Exemple d'une réaction chimique



II.3. Les réseaux de Petri
Modélisation avec ressources

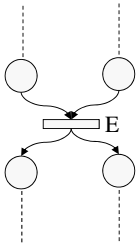
– Schémas particuliers



séquence ab | indépendance alb | indétermisme a ou b | partage de ressource (conflit)

II.3. Les réseaux de Petri
Modélisation avec ressources

– Schémas particuliers



synchonisation a=b,
Envoi synchrone
d'un message E

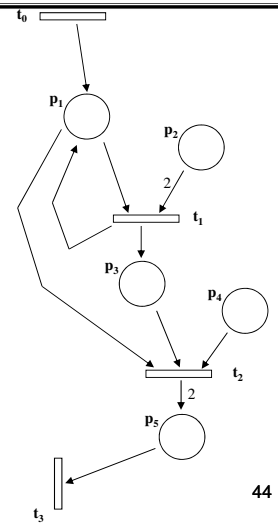
II.3. Les réseaux de Petri

Notions

- Une **transition-puit** est une transition ayant une sortie vide.
- Une **transition-source** est une transition ayant une entrée vide.
- Une **boucle** est un circuit constitué d'une seule place et d'une seule transition.
- Un RdP sans boucle est dit **pur**

Exemple :

- t_0 est une transition-source.
- t_3 est une transition-puit.
- (p_1,t_1) une boucle



II.3. Les réseaux de Petri
Formalisation (Définition formelle)

Réseau de Petri: R = <P, T, Pre, Post>

- P = ensemble de places
- T = ensemble de transitions

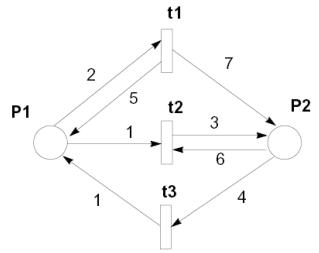
- Pre = PxT → N conditions sur les places précédentes
  Pre(p, t) = nombre de jeton nécessaire dans la place p pour le franchissement de la transition t

- Post = PxT → N conditions sur les places suivantes
  Post(p, t) = nombre de jeton produits dans la place p lors du franchissement de la transition t

=> C = Post - Pre matrice d'incidence

II.3. Les réseaux de Petri
Formalisation

- Exemple d'une représentation matricielle



Entrée

	t1	t2	t3
P1	2	1	0
P2	0	6	4

Pre

Sortie

	t1	t2	t3
P1	5	0	1
P2	7	3	0

Post

II.3. Les réseaux de Petri
Formalisation

- Le marquage M d'un RDP R=(P, T, Pre, Post) est représenté par son état.

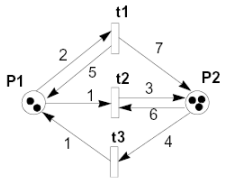
M : P → N

Ça permet de donner à chaque place le nombre de jetons qu'elle contient.

M_0 est le marquage initial.

M_0

P1	2
P2	3



.

II.3. Les réseaux de Petri
Formalisation (Propriétés dynamiques)

- Dynamique (sémantique) d'un RdP.
  - Une transition **t** est **franchissable** ssi, pour toute place **p** en entrée de **t**,

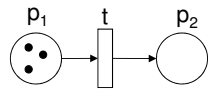
    M(p) ≥ Pre(p, t)
  - Si une transition **t** est **franchissable** à partir du marquage **M**, alors le nouveau marquage de toute place **p** en entrée de **t** est

    M'(p) = M(p) - Pre(p, t) + Post(p, t)
    = M(p) + C(p, t)
    avec C = Post - Pre (matrice d'incidence)

on note M → t → M'

Exp:

$M'(p_1) = M(p_1) - Pre(p_1, t) + Post(p_1, t)$
= 3-1+0 = 2



II.3. Les réseaux de Petri

Formalisation (Propriétés dynamiques)

- Exemple 1
  - t1 est franchissable car

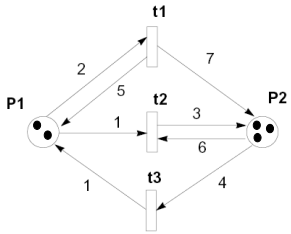
$$Pre(., t1) = \begin{bmatrix}2\\0\end{bmatrix} \leq M0$$

  - après le franchissement de t1

$$M = M0 - Pre(., t1) + Post(., t1)$$

$$= \begin{bmatrix}2\\3\end{bmatrix} - \begin{bmatrix}2 & 1 & 0\\0 & 6 & 4\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}5 & 0 & 1\\7 & 3 & 0\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}$$

$$= \begin{bmatrix}2\\3\end{bmatrix} - \begin{bmatrix}2\\0\end{bmatrix} + \begin{bmatrix}5\\7\end{bmatrix} = \begin{bmatrix}5\\10\end{bmatrix}$$



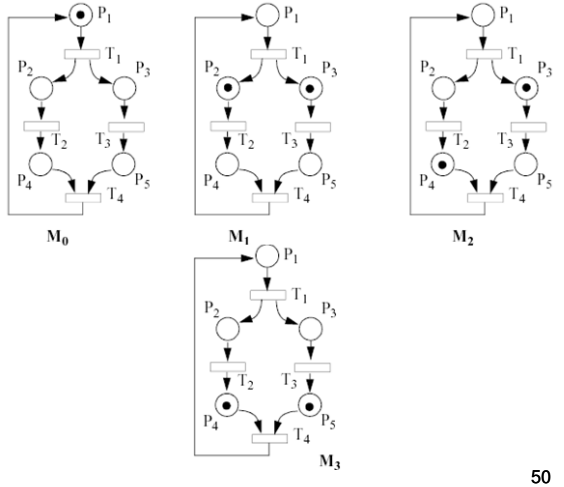
II.3. Les réseaux de Petri

Formalisation (Séquence de transitions franchissable)

Exemple

Séquence de transitions

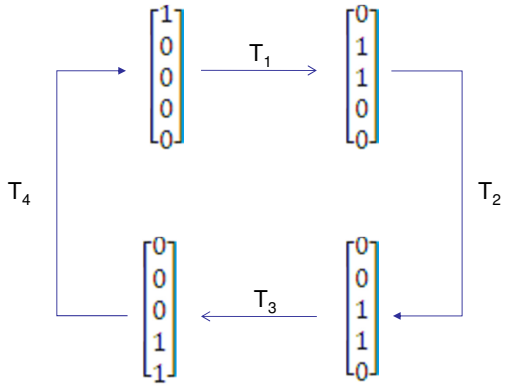
T1 T2 T3 T4
est une séquence de transitions franchissables



II.3. Les réseaux de Petri

Formalisation (Séquence de transitions franchissable)

- Suite exemple

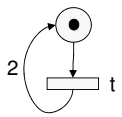


II.3. Les réseaux de Petri

Formalisation (Propriétés dynamiques)

Remarque: Le graphe de marquage peut être infini ➔ cas d'un RdP non borné

- Exemple



(1) —t→ (2) —t→ (3) —t→ (4) —t→ …

II.3. Les réseaux de Petri
Formalisation (Propriétés dynamiques)

- Propriétés des RdP
  - Place et RdP k-bornés
    - Une **place** est dite **k-bornée** pour un marquage initial si sa marque ne dépasse jamais k (binaire si k=1)
    - Un **RdP** est **k-borné** si toutes les places le sont.
  - Transition et RdP vivants
    - Une **transition** est dite **quasi-vivante** pour un marquage M s'il existe un marquage M' accessible à partir de M permettant de la franchir.
    - Une **transition** est dite **vivante** si elle est **quasi-vivante** pour tout marquage accessible à partir de M_0.
    - Un **RdP** est dit **vivant** si toutes ses transitions le sont.
  - Un RdP ne contient pas de **blocage** s'il peut continuellement évoluer
  - Un RdP est dit **réinitialisable** si M_0 est accessible à partir de tout marquage

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Déterminer les propriétés d'un RdP en se basant sur sa structure (indépendamment du marquage)

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Composante conservatrice positive

Soit un RdP R=(P, T, Pre, Post)

Soit V_p un vecteur de N^p avec p = taille de P

V_p est appelé **composante conservatrice positive** ssi

$$\mathbf{V_p^T \cdot C = 0}$$

Remarque:

Une composante conservatrice positive est un **ensemble de places** dans lequel le **nombre de jetons est borné** quelque soit les transitions franchies

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Exemple 1

En examinant la colonne t_1 nous trouvons que:
On peut remplacer la ligne P_1 par P_1+P_3
Puis supprimer P_3 et t_1
En examinant la colonne t_2 nous trouvons que:
On peut remplacer la ligne P_1+P_3 par $P_1+P_3+P_2$
Puis supprimer P_2 , t_2 et t_3
En examinant la colonne t_4 nous trouvons que:
On peut remplacer la ligne P_4 par P_4+P_5
Puis supprimer P_5 et t_4
En examinant la colonne t_5 nous trouvons que:
On peut remplacer la ligne P_4+P_5 par $P_4+P_5+P_6$
Puis supprimer P_6 et t_4
Et la matrice devienne comme suit:
Donc nous concluons la matrice de base des C.C.P comme suit:

$$T = (\ t_1,\ t_2,\ t_3,\ t_4,\ t_5,\ t_6)$$

$$P = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix} \quad C = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

$$T = (\ t_6)$$

$$P = \begin{pmatrix} P_1+P_2+P_3 \\ P_4+P_5+P_6 \end{pmatrix} \quad C = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Exemple 1 (suite)

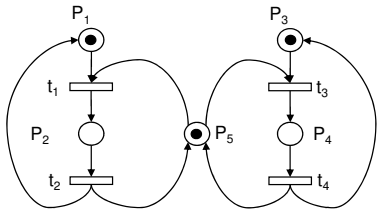
Donc nous concluons la matrice de base des C.C.P comme suit:

Chaque colonne de la base représente une composante conservatrice positive.

$$P = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{pmatrix} \quad B = \begin{array}{c} (V_p1,\ V_p2) \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \end{array}$$

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Exemple 2



$$T = (t_1,\ t_2,\ t_3,\ t_4) \qquad P = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{pmatrix} \quad C = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

En examinant la colonne t_1 nous trouvons que:
On peut remplacer la ligne P_1 par P_1+P_2 et la ligne P_5 par P_2+P_5
Puis supprimer la ligne P_2 et les colonnes t_1 et t_2
En examinant la colonne t_3 nous trouvons que:
On peut remplacer la ligne P_3 par P_3+P_4 et la ligne P_2+P_5 par $P_2+P_4+P_5$
Puis supprimer la ligne P_4

$$T = (t_3,\ t_4) \qquad P = \begin{pmatrix} P_1+P_2 \\ P_3+P_4 \\ P_2+P_4+P_5 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Exemple 2 (suite)

Donc nous concluons la matrice de base des C.C.P comme suit:

$$P = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{pmatrix} \quad B = \begin{array}{c} (V_p1,\ V_p2,\ V_p3) \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

et les **composantes conservatrices positives** sont représentées par les **colonnes** de la matrice de base des C.C.P.

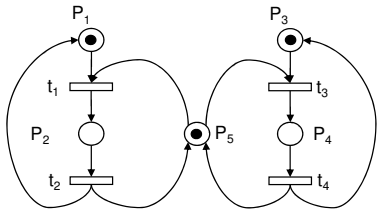
II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

Propriétés d'une composante conservatrice positive

- Les **places** d'une composante conservatrice positive sont **k-bornées**
- Si toutes les places d'un RdP appartiennent à des composantes conservatrices positives, alors le **réseau** est **k-borné** (réciproque fausse)

II.3. Les réseaux de Petri
Formalisation (Propriétés structurelles)

- Exemple



V1 = (P_1 + P_2)

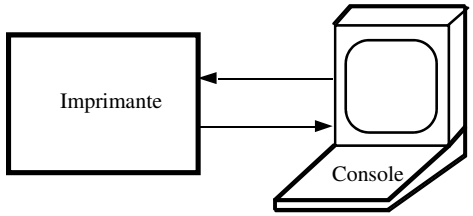
V2 = (P_3 + P_4)

V3 = (P_2 + P_4 + P_5)

– Les trois composantes V1, V2 et V3 couvrent l'ensemble du réseau donc le réseau est k-borné
– Il ne peut pas y avoir 1 jeton à la fois dans P_2 et P_4

II.3. Les réseaux de Petri
Exemple Imprimante-Console

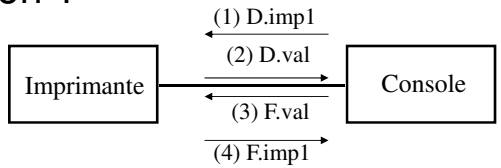
- Exemple Imprimante-Console



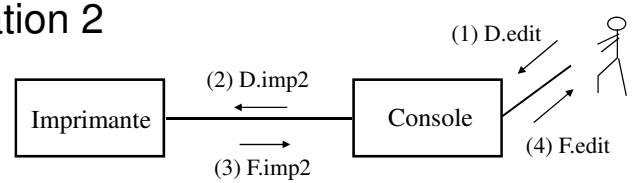
– Utilisation 1
  • Impression d'un texte « Imp1 »
  • Validation de l'impression « Val »
– Utilisation 2
  • Saisie d'un texte « Edit »
  • Impression du texte saisie « Imp2 »

II.3. Les réseaux de Petri
Exemple Imprimante-Console

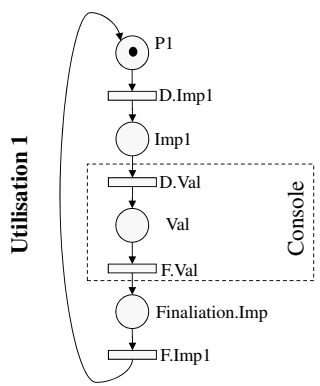
Utilisation 1

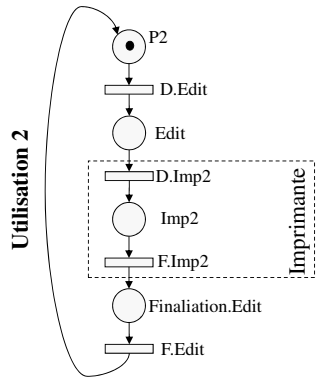


Utilisation 2

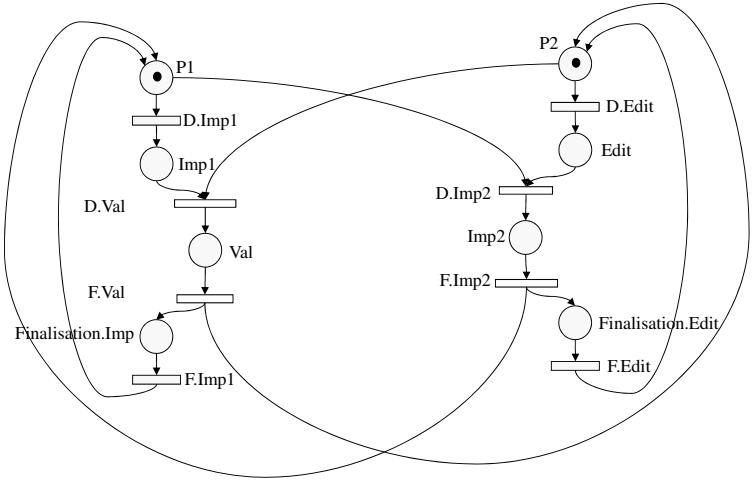


II.3. Les réseaux de Petri
Exemple Imprimante-Console





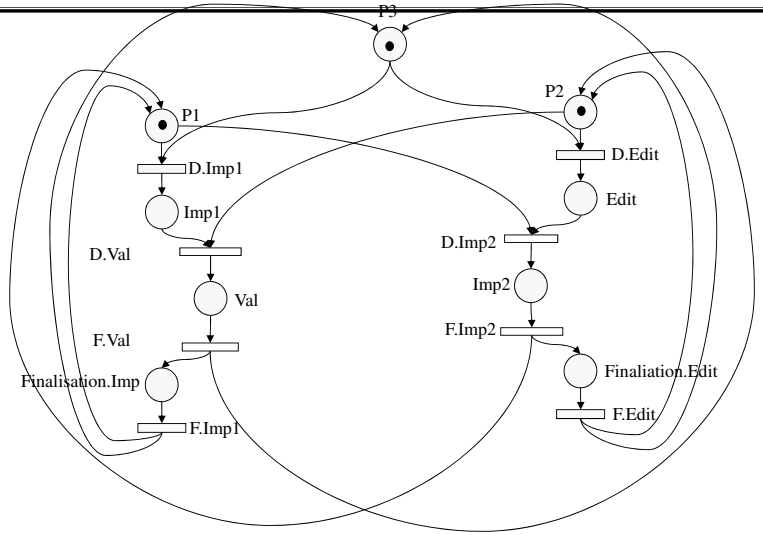
II.3. Les réseaux de Petri
Exemple Imprimante-Console



II.3. Les réseaux de Petri
Exemple Imprimante-Console

- Analyse
  - L'arrivée de D.Imp1 et D.Edit en même temps crée un blocage
- Solution
  - Utilisation d'un sémaphore pour empêcher l'arrivée des deux événements en même temps

II.3. Les réseaux de Petri
Exemple Imprimante-Console



II.3. Les réseaux de Petri
Exemple Imprimante-Console

- Commentaires
  - Le blocage dépend:
    - de la structure du réseau de Petri
    - du marquage initial
  - Le blocage est un problème critique
  - Difficile de prouver l'absence du blocage

II.3. Les réseaux de Petri

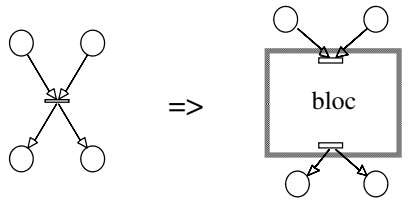
Raffinement et composition

- Le raffinement et la composition dans les réseaux de Petri permettent de **modéliser un système complexe**.
- Exp. Modélisation des différentes parties du système séparément, puis composition du modèle final à partir des autres parties.

II.3. Les réseaux de Petri

Raffinement et composition

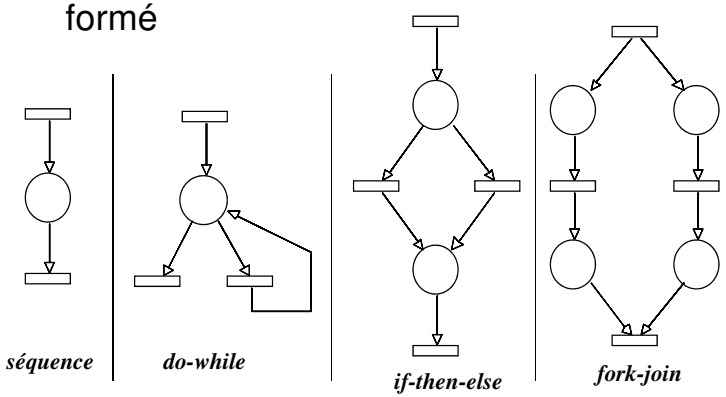
- Raffinement
  - Substitution d'une transition par bloc bien formé



II.3. Les réseaux de Petri

Raffinement et composition

- Raffinement (Blocs bien formés standard)
  - Substitution d'une transition par bloc bien formé

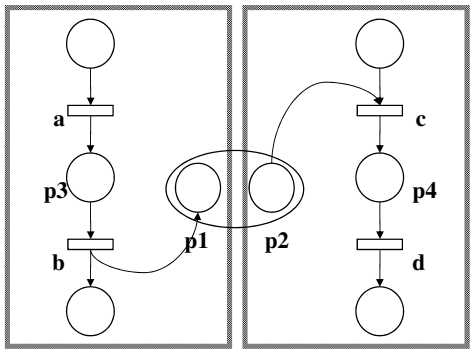


II.3. Les réseaux de Petri

Raffinement et composition

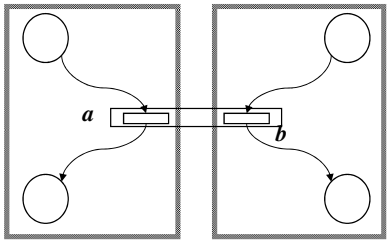
- Composition asynchrone

  (fusion de places)



II.3. Les réseaux de Petri
Raffinement et composition

- Composition synchrone
  (fusion de transitions)



II.4. Les dérivés des réseaux de Petri

II.4.1. Rdp colorés
II.4.2. Rdp temporels
II.4.3. Rdp prédicats/transition

II.4. Les dérivés des réseaux de Petri
II.4.1. RdP colorés

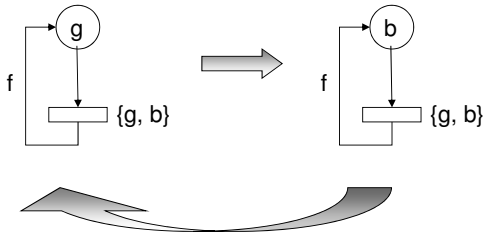
II.4.1. RdP colorés

- Les jetons sont « **typés** » par des couleurs.
- Le nombre de couleurs est **fini**.
- Représentation **compacte** des systèmes par rapport aux réseaux usuels.
- **Plusieurs couleurs** de franchissement sont associées à une transition.
- Une **fonction** est associée à un arc

II.4. Les dérivés des réseaux de Petri
II.4.1. RdP colorés

Exemple

f(g) = b
f(b) = g



Définition formelle

Un réseau de Petri coloré est défini:

Rc = <P, T, Coul, Csec>

avec:

P: Ensemble fini de **places**.

T: Ensemble fini de **transitions**.

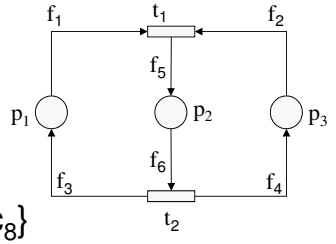
Coul: Ensemble fini de **couleurs**.

Csec: *fonction* $P \cup T \rightarrow$ CoulPos

CoulPos: Ensemble de **couleurs possible** pour une transition ou une place (**CoulPos** $\subseteq$ **Coul**)

Exemple

Soit un RdP coloré:



Rc = <P, T, Coul, Csec>

P = {p_1, p_2, p_3}

T = {t_1, t_2}

Coul = {c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8}

Csec(p_1) = {c_1, c_2} ; Csec(p_2) = {c_3, c_4}

Csec(p_3) = {c_5, c_6, c_7, c_8}

Csec(t_1) = {c_1, c_2, c_5, c_6, c_7, c_8}

Csec(t_2) = {c_3, c_4}

Suite exemple

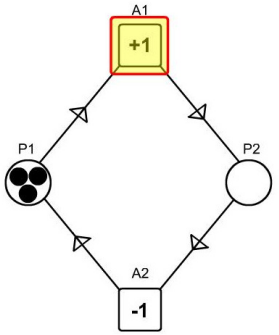
Marquage initial

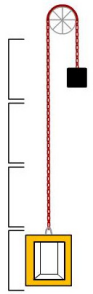
$$M_0 = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Exemple 2 (Ascenseur)

Différentes modélisations pour un même problème

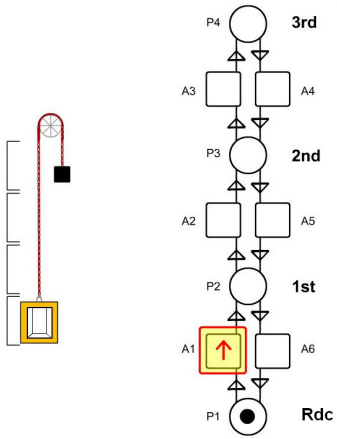
Les jetons représentent les étages





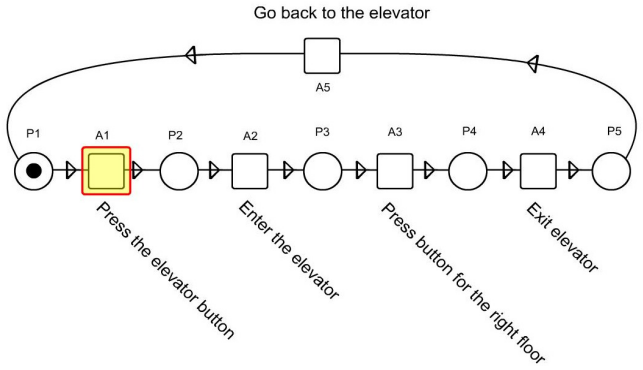
Suite exemple 2 (Ascenseur)

Le jeton représente l'ascenseur



Suite exemple 2 (Ascenseur)

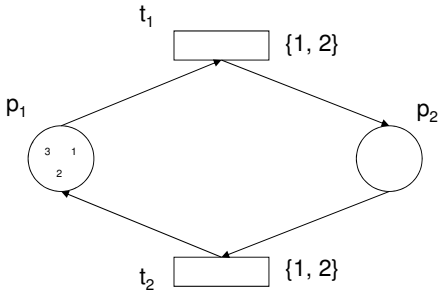
Le jeton représente l'utilisateur de l'ascenseur



Suite exemple 2 (Ascenseur)

L'ascenseur ne monte pas au 3ème étage suite à une panne

Pour résoudre ce problème nous utilisons un RdP coloré.



Exemple 3 (Location de voiture)

Le RdP suivant modélise le fonctionnement d'une entreprise de location de voitures qui a deux types de clients.

P_0 correspond au nombre de voitures prêtes à être louées

P_1 (respectivement P_2) indique le nombre de clients de type 1 (resp. 2) attendant pour louer une voiture

Le franchissement de la transition t_1 (resp. t_2) correspond au début de la location d'une voiture par un client de type 1 (resp. 2).

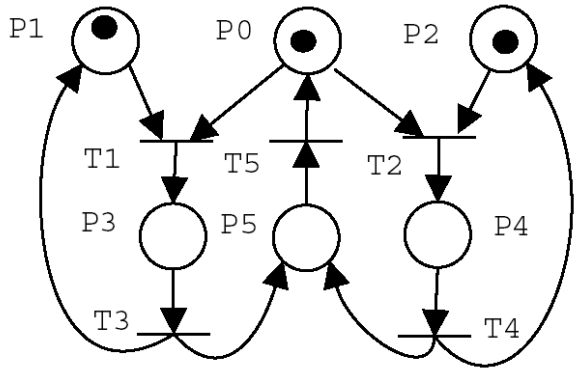
Le nombre de marques dans la place P_3 (resp. P_4) correspond au nombre de voitures louées par les clients de type 1 (resp. 2).

Le franchissement de la transition t_3 (resp. t_4) correspond à la fin de location par un client de type 1 (resp. 2).

La voiture est alors au garage pour entretien: Le nombre de jetons dans P_5 indique le nombre de voitures au garage.

Le franchissement de la transition t_5 correspond à la fin de l'entretien.

Suite exemple 3 (Location de voiture)



Suite exemple 3 (Location de voiture)

Question:

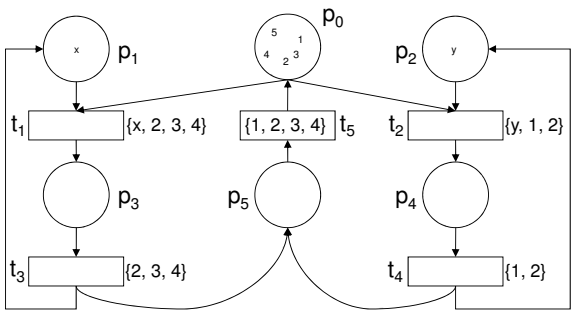
L'entreprise possède 5 voitures $\{v_1, v_2, v_3, v_4, v_5\}$, les clients de type 1 ont le droit de louer $\{v_1, v_2\}$, et les clients de type 2 ont le droit de louer $\{v_2, v_3, v_4\}$

Modéliser ce problème.

Suite exemple 3 (Location de voiture)

Solution:

On utilise un RdP coloré



- Deux principales familles
  - Les réseaux de Petri temporisés
  - Les réseaux de Petri temporels

II.4. Les dérivés des réseaux de Petri
II.4.2. RdP temporels

- Les réseaux de Petri temporisés
  - Temporisations associées aux **transitions** (**t-temporisés**)
  - Temporisations associées aux **places** (**p-temporisés**)

Temporisation = **durée minimale** de **franchissement** d'une **transition**, ou la **durée minimale** qu'un **jeton** passe dans une **place**.

II.4. Les dérivés des réseaux de Petri
II.4.2. RdP temporels

- Les réseaux de Petri temporels
  - Intervalles associés aux **transitions** (**t-temporels**)
  - Intervalles associés aux **places** (**p-temporels**)

II.4. Les dérivés des réseaux de Petri
II.4.2. RdP temporels

- Les réseaux temporels sont obtenus depuis les réseaux de Petri ordinaires en associant deux dates ***min*** et ***max*** à chaque transition (intervalle) « **t-temporel** »
- Nous notons ce type de réseaux: **RdPT**

II.4. Les dérivés des réseaux de Petri
II.4.2. RdP temporels

- Définition formelle:

Un réseau temporel (ou Time Petri net) est un tuple

<P, T, Pre, Post, I_s>

tel que,

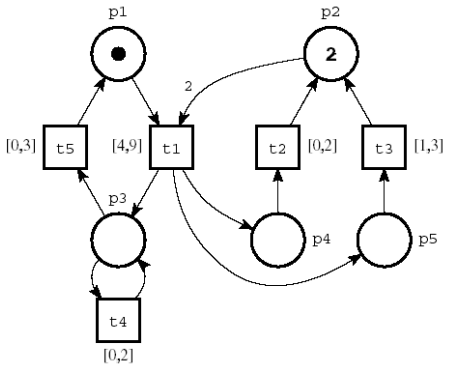
I_s : $T \rightarrow I^+$ est une fonction appelée **Intervalle statique**

et I^+ est l'ensemble des intervalles réels fermés non vides à bornes positives.

exp: $I_s(t_1) = [0, 5]$

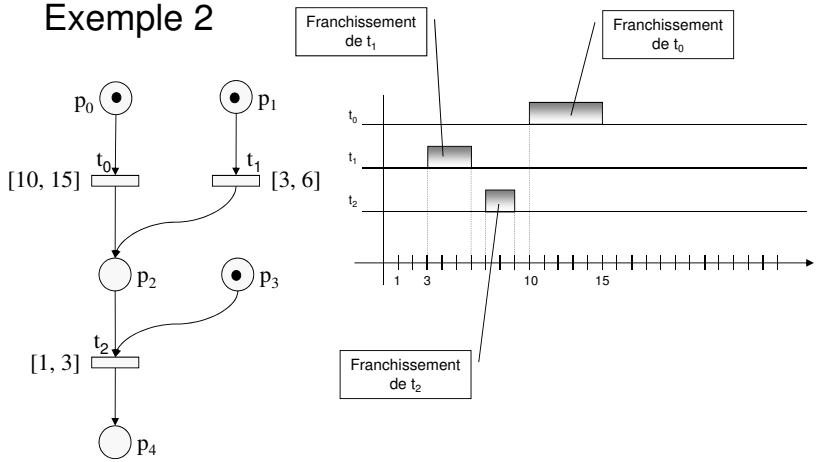
II.4. Les dérivés des réseaux de Petri
II.4.2. RdP temporels

- Exemple 1 d'un RdPT



II.4. Les dérivés des réseaux de Petri
II.4.2. RdP temporels

- Exemple 2



II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

- Introduction

  Dans les RdP colorés on a remplacer les valuations des arcs par des fonctions (ou matrices) ce qui a permis d'augmenter la puissance du réseau (compacter la représentation)

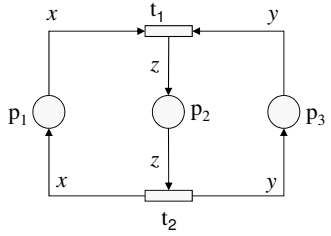
  Dans les **RdP Prédicats-Transitions** on remplace les **transitions** par des **règles** dans une logique de premier ordre.

  Une **règle** (transition) représente une **famille d'événements** au lieu d'un seul événement.

  On associe à un **arc** une **variable** au lieu d'une fonction dans le RdP coloré.

II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

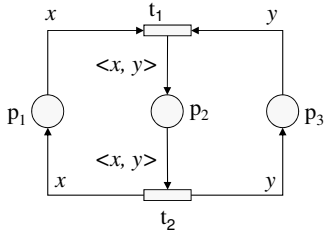
- Exemple



II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

- Notion de n-uplet de constantes et de variables

Dans l'exemple précédant, le franchissement de la transition t_1 produit deux constantes, une issue du contenu de la variable x, et une autre issue du contenu de la variable y se qui représente un 2-uplet.



II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

- Définition formelle

Un réseau de Petri Prédicats-Transitions est un triplet:

$$R_{pt} = <R, A_n, M_0>$$

Tel que :

R = <P, T, Pré, Post> Réseau de Petri ordinaire **sous-jacent** de R_{pt}.

A_n: **L'annotation** de R, $A_n = <C_{onst}, V, A_{tc}, A_{ta}, A_c>$

avec:

C_{onst}: Ensemble de **constantes**

V: Ensemble de **variables** formelles

A_{tc}: $T \rightarrow L_c(C_{onst}, V)$ associant à chaque **transition** une **condition** sous la forme d'un prédicat en fonction de C_{onst} et V

A_{ta}: $T \rightarrow L_a(C_{onst}, V)$ associant à chaque **transition** une **action** sous la forme d'une suite d'affectation de valeurs à des variables formelles.

A_c: une application associant à chaque **arc** une **somme formelle** de n-uplets d'éléments de V, tel que $||A_c(p, t)|| = C(p, t)$

avec C(p, t) poids de l'arc dans la matrice d'incidence.

M_0: Marquage initial associant à chaque place une somme formelle de n-uplets de C_{onst}.

II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

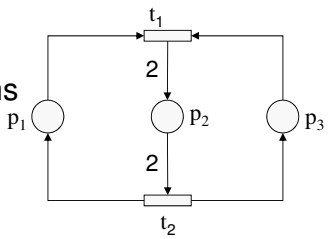
- Exemple

Dans l'exemple précédant, le réseau sous-jacent R est schématisé comme suit:

C_{onst}= {c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8}

V = {x, y}

Pas de conditions, et pas d'actions



$$A_c = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{bmatrix} -<x> & <x> \\ <x,y> & -<x,y> \\ -<y> & <y> \end{bmatrix} \end{array}$$

$$M_0 = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{bmatrix} <c_1> + <c_2> \\ 0 \\ <c_3> + <c_4> \end{bmatrix}$$

II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

- Système de règles

On associe des places aux prédicats utilisés.

Exp. $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow B$

Cette **clause** représente une **transition** t, et les A_i représentent les **places d'entrée** à t, et B représente la place de **sortie** de t.

Les places des A_i peuvent prendre des valeurs de C_{onst} rendant ce prédicat vrai.

Donc la transition t est franchissable si le prédicat associé est vrai.

- Exemple

Ensemble de clauses (de règles)

r_1 : parent(a, b) ←

r_2 : parent(b, c) ←

r_3 : ancêtre(x, y) ← parent(x, y)

r_4 : ancêtre(x, z) ← parent(x, y), ancêtre(y, z)

r_5 : ← ancêtre(x, c)

La règle r_1 signifie que « ***a*** est parent de ***b*** »

r_2 signifie que « ***b*** est parent de ***c*** »

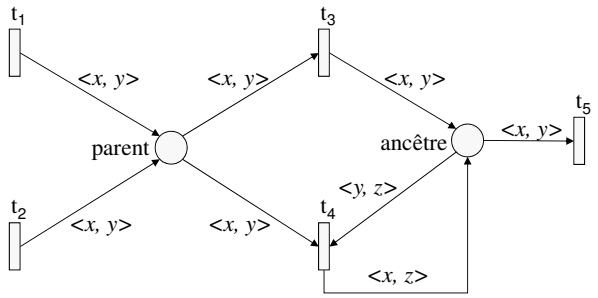
r_3 signifie que « si ***x*** est parent de ***y*** alors il est ancêtre de ***y*** »

r_4 signifie que « ***x*** est parent de ***y*** et ***y*** est ancêtre de ***z*** alors ***x*** est un ancêtre de ***z*** »

r_5 signifie que « ***x*** est ancêtre de ***c*** »

c.a.d on cherche à la sortie du réseau le parent de ***c***.

- Réseau de Petri dérivant des clauses est:



On ajoute aux transitions d'entrée et de sortie du réseau t_1, t_2 et t_5 les conditions suivantes:

$A_{tc}(t_1) = (\boldsymbol{x}==\boldsymbol{a}) \wedge (\boldsymbol{y}==\boldsymbol{b})$

$A_{tc}(t_2) = (\boldsymbol{x}==\boldsymbol{b}) \wedge (\boldsymbol{y}==\boldsymbol{c})$

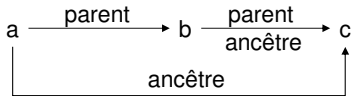
$A_{tc}(t_5) = (\boldsymbol{y}==\boldsymbol{c})$

On a pas besoin d'actions (A_{ta})

II.4. Les dérivés des réseaux de Petri
II.4.3. RdP Prédicats-Transitions

La séquence t_2,t_3,t_5 produit « ***b*** est ancêtre de ***c*** »

La séquence t_2,t_3,t_1,t_4,t_5 produit « ***a*** est ancêtre de ***c*** »



II.5. Méthodes de réductions

Nous verrons dans ce cours 4 méthodes de réductions

II.5.1. Suppression de place

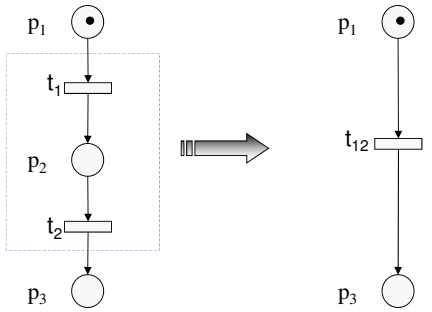
Une place p peut être supprimée si:

- Les **transitions en sortie** de p n'ont pas d'autres places en entrée que p
- Il n'existe pas de transition t qui soit à la fois en entrée et en sortie de p
- Au moins une transition en sortie de p n'est pas une transition puit.

Si toutes ces conditions sont remplies alors p peut être supprimée.

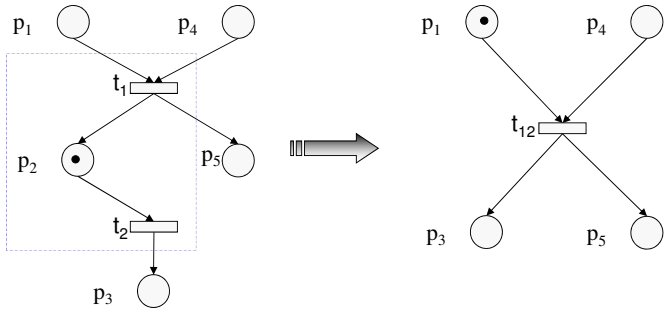
II.5. Méthodes de réductions

Exemple 1:



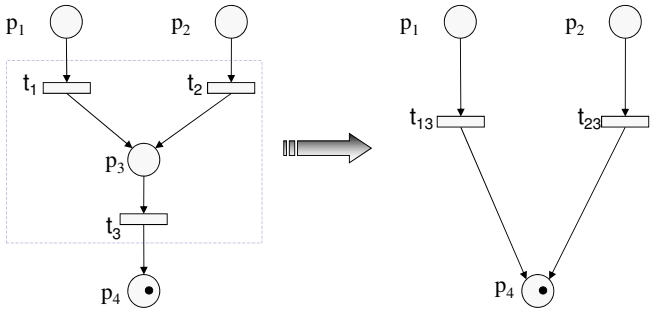
II.5. Méthodes de réductions

Exemple 2:



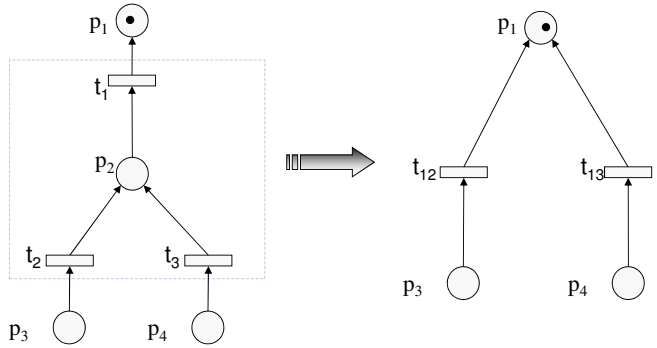
II.5. Méthodes de réductions

Exemple 3:



II.5. Méthodes de réductions

Exemple 3:



II.5. Méthodes de réductions

II.5.2. Suppression des places implicites

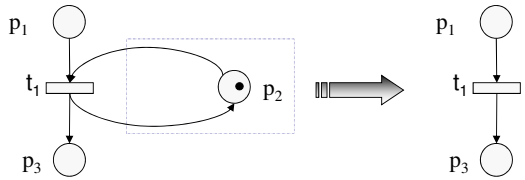
Une place p est implicite si:

- Son **marquage** n'a **aucun impact** sur le **franchissement** des transitions en sortie

Dans ce cas là on peut supprimer la place p.

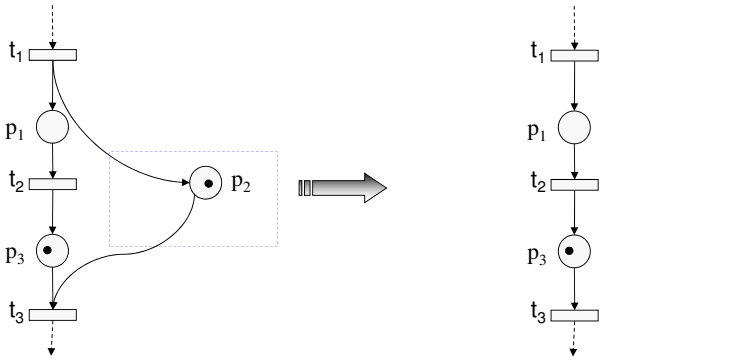
II.5. Méthodes de réductions

Exemple 1:



II.5. Méthodes de réductions

Exemple 2:



II.5. Méthodes de réductions

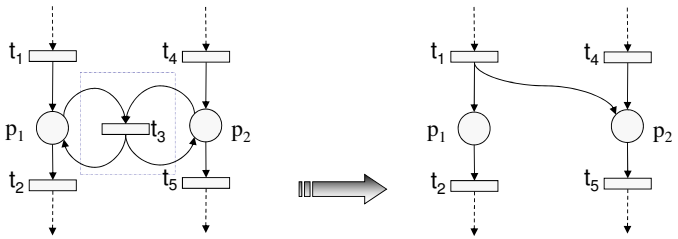
II.5.3. Suppression des transitions neutres

Une transition t est neutre si les **places en entrée** sont aussi des **places en sortie**.

Dans ce cas là on peut supprimer t.

II.5. Méthodes de réductions

Exemple:



II.5. Méthodes de réductions

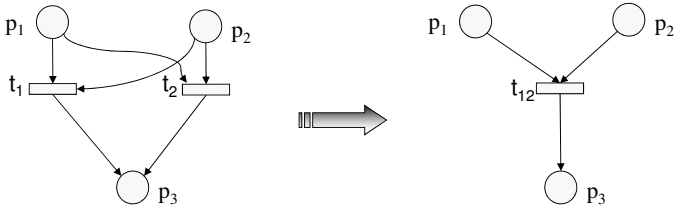
II.5.4. Suppression des transitions identiques

Deux transitions t_i et t_j sont identiques si elles ont les **mêmes positions en entrée** et les **même positions en sortie**.

Dans ce cas là on peut supprimer ces transitions en les remplaçant par une seule transition.

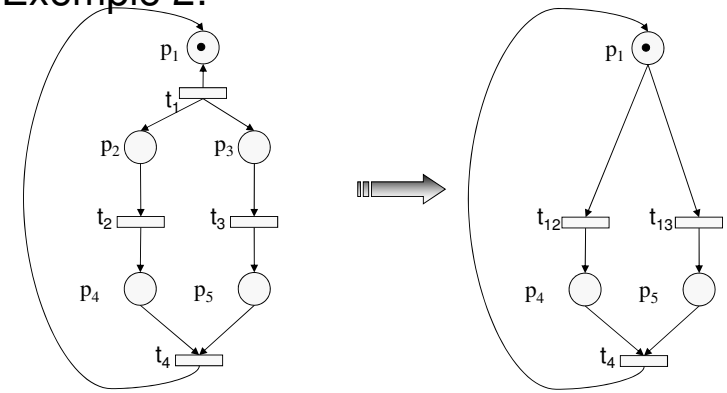
II.5. Méthodes de réductions

Exemple 1:



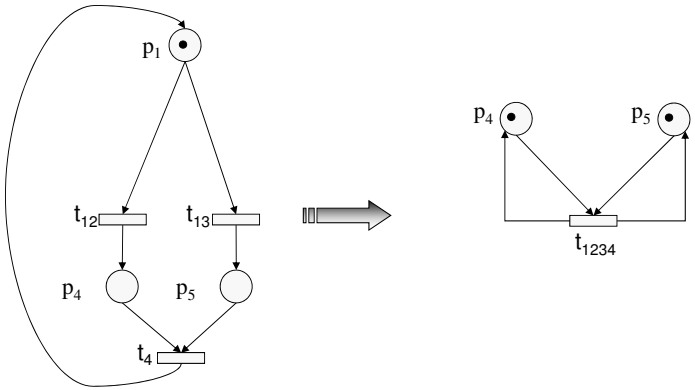
II.5. Méthodes de réductions

Exemple 2:



II.5. Méthodes de réductions

Exemple 2:



II.5. Méthodes de réductions

Exemple 2: