F. Vernadat 2008

TPs: http://www.laas.fr/~francois/POLYS/ENSEIGNEMENT/PN/
Support de cours : http://www.laas.fr/~francois/POLYS/ENSEIGNEMENT/rdpfv.pdf
Polycopie de Robert Valette : http://www.laas.fr/~francois/POLYS/ENSEIGNEMENT/rv.pdf

Historique

1962 Thèse de Carl Adam Petri (RFA)

1975 Travail important (France & RFA)

1980 Conférence Annuelle (Springer-Verlag)

Diffusion

"Mondiale" : Universitaire & Industrielle

Outils Disponibles [89-92]

31 Outils USA, Japon, CEE, Pays de l'Est

10 Industriels (3 Francais (2 Toulousains))

Site où sont répertoriés les $\neq$ outils:

http://www.daimi.aau.dk/PetriNets/tools/

Applets Java: http://www.daimi.aau.dk/PetriNets/tools/java/

Intérêt

Formalisme Rigoureux (sémantique précise)

Expressivité

Compacité (machines à états)

Concepts: //, Coopération, Compétition, Synchronisation, ...

Dualité Etat/Evénement

Représentation Graphique

Possibilité d'Analyse Structurelle, Exhaustive

1977 Grafcet Commande Séquentielle (API)

1980 Evaluation de Performances

1980 Protocoles de Communication

1986 Systèmes d'Information, IHM

1989 Modèles de Raisonnement

1989 Systèmes Coopératifs

$\neq$ *Applications* $\leftrightarrow \neq$ *Modeles*

G.W Brams (ouvrage collectif)

Réseaux de Petri: Théorie et Pratique (2 tomes)

Masson -1980 (Epuisé DGE)

W. Reisig

Petri Nets. An Introduction

Springer-Verlag EATCS 1985

C. Reutenauer

Aspect Mathématiques des Réseaux de Petri

Etudes et recherches en informatique

Masson - 1989

K. Jensen

Coloured Petri Nets (2 tomes)

Springer-Verlag EATCS 1992

W. Reisig

Elements od Distributed Algortithms:
Modeling and Analysis with Petri Nets

Springer-Verlag 1998

M. Diaz (Editeur)

Les Réseaux de Petri. Modèles fondamentaux,

Hermes Science, Traité IC2, NISBN 2-7462-0250-6, 2001

M. Diaz (Editeur)

Vérification et mise en oeuvre des réseaux de Petri

Hermes Science, Traité IC2, NISBN ... , 2003

Sites WEB

World of Petri Nets:

http://www.daimi.aau.dk/PetriNets/

FAQ: Frequently Asked Questions
Petri Nets Standard
Tools etc ....

Groupe Réseaux de Petri

http://www.ec-lille.fr/ rdp/

Réseaux Place/Transition

- Définitions & Concepts de base, Enumération, Propriétés
- Analyse Structurelle : Invariants et Réduction
- Décidabilité de la finitude de l'espace d'états accessibles

Extensions

- Réseaux Prédicat/Transitions (colorés)
- Réseaux temporels, temporisés

Nombreux Autres Modèles

- Systèmes de transitions (automates),
- Algèbres de Processus,
- Acteurs, State-Charts, Grammaires de Graphes, ...

1 Définitions de Base

1.1 Réseau Place/Transition

$R =< P, T, Pr, Post >$ où

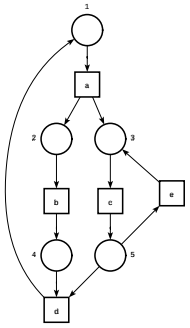
P est un ensemble fini de *Places*

T est un ensemble fini de *Transitions* $\quad P \cap T = \emptyset$

$Pre : P \times T \mapsto IN$ *incidence avant*

$Post : P \times T \mapsto IN$ *incidence arrière*

1.2 Représentation Graphique



1.3 Réseau Place/Transition Marqué

Marquage:

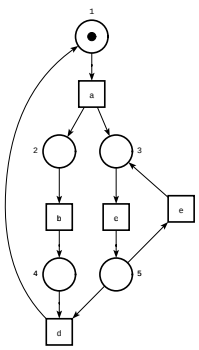
$M : P \mapsto IN$

$M(p)$ dénote le marquage pour M de la place p

Réseau marqué

Couple (réseau, marquage) $N =< R, M >$

Représentation Graphique



1.4 Notations Matricielles

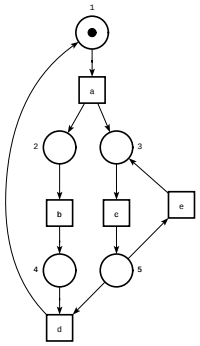
Réprésenter matriciellement M, Pre & $Post$

$Pre : P \times T \mapsto IN$ $\quad Post : P \times T \mapsto IN$ $\quad M : P \mapsto IN$

Pre	a	b	c	d	e
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	1	1

Post	a	b	c	d	e
1	0	0	0	1	0
2	1	0	0	0	0
3	1	0	0	0	1
4	0	1	0	0	0
5	0	0	1	0	0

M_0
1
0
0
0
0



≡ Evolution des marquages

2.1 Transition sensibilisée

Une transition t est sensibilisée pour un marquage M ssi

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

On le note $M[\ t\ >$

2.2 Règle de Tir d'une transition sensibilisée

Soit t une transition et M un marquage tel que $M[\ t\ >$

alors le tir de t à partir de M conduit en M' défini par:

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

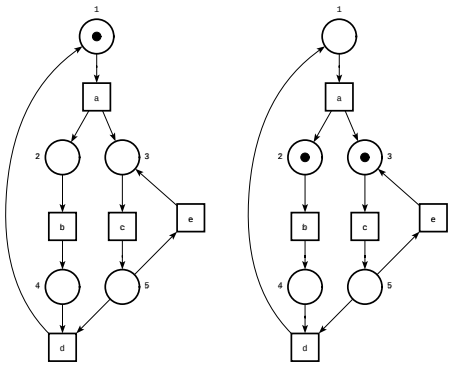
(ou aussi $M' = M - Pre(.,t) + Post(.,t)$)

On le note $M[\ t\ > M'$

$M[\ t\ >$ ssi $\forall p \in P : M(p) \geq Pre(p,t)$

Si $M[\ t\ > M'$ alors $\forall p \in P : M'(p) = M(p) - Pre(p,t) + Post(p,t)$

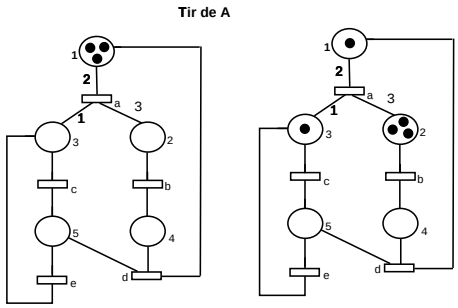
Tir de la transition a

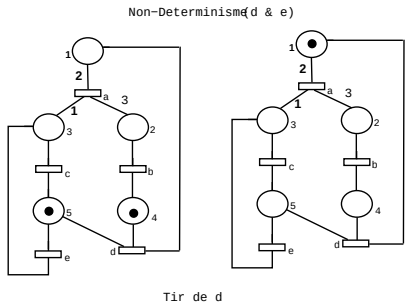


Remarques:

- $Pre(.,t)$ est le marquage minimum permettant de tirer t
- $Post(.,t)$ est le marquage minimum que l'on peut atteindre après le tir de t
- L'évolution des marquages est indépendante du marquage de départ

Exemple (suite)





3 Espace des Etats Accessibles

3.1 Ensemble des Marquages accessibles $A(R, m_0)$

$A_0 = \{m_0\}$

$A_i = \{m \in IN^p / \exists m' \in A_{i-1}, \exists t \in T : m'[\ t\ > m\}$

$A(R, m_0) = \cup_{i \geq 0} A_i$

3.2 Graphe des Marquages accessibles $G(R, m_0)$

Graphe $\mathcal{G} =< S, \rightarrow, L >$ où

S est un ensemble de sommets,
L est un ensemble de labels de transitions
et $\rightarrow$ la relation de transition ($\subset S \times L \times S$)

Notations: $s \stackrel{t}{\longrightarrow} s'$ au lieu de $(s,t,s') \in \rightarrow$

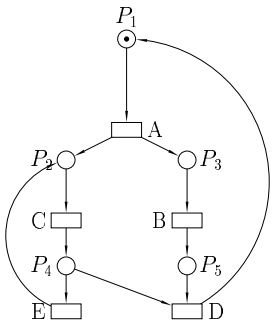
Si $\omega = t_1 \ldots t_n$ on note $s \stackrel{\omega}{\longrightarrow} s'$ ssi $s \stackrel{t_1}{\longrightarrow} s_1, \ldots s_n \stackrel{t_n}{\longrightarrow} s_{n+1}$ $(s' = s_{n+1})$

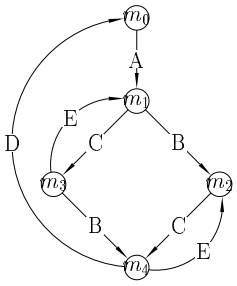
$G(R, m_0) =< A(R, m_0), \rightarrow, L >$ **où**

- $\rightarrow$ est le plus petit ensemble vérifiant:

  $(m_1, t, m_2) \in \rightarrow$ ssi $m_1, m_2 \in A(R, m_0), t \in T$ et $m_1[\ t\ >\ m_2$

- $L \subset T$ défini par $t \in L$ ssi $(m_1, t, m_2) \in \rightarrow$





$$m_o = \begin{vmatrix}1\\0\\0\\0\\0\end{vmatrix} \quad m_1 = \begin{vmatrix}0\\1\\1\\0\\0\end{vmatrix} \quad m_2 = \begin{vmatrix}0\\0\\1\\1\\0\end{vmatrix} \quad m_3 = \begin{vmatrix}0\\1\\0\\0\\1\end{vmatrix} \quad m_4 = \begin{vmatrix}0\\0\\0\\1\\1\end{vmatrix}$$

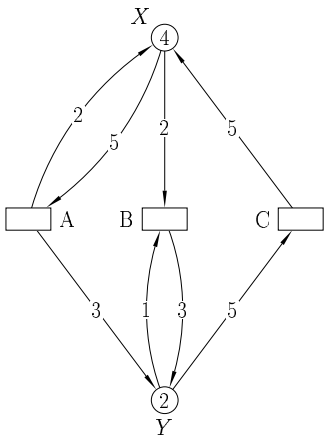
$L(R, m_0) = (a.(ce)^*.(cb + bc)^*.(ec)^*.d)^*$

1. Initialisation: $Stack$ is empty ;
   push m_0 into $Stack$
   $A(R, m_0)$ is empty ;
   enter m_0 in $A(R, m_0)$

2. Boucle : while $Stack \neq \emptyset$
   loop {
   pop(q) from stack
   $T \leftarrow \{t \in T : q\ [\ t\ >\}$;
   $\forall t \in\ T\ do$
   $\{q\ [\ t\ > q'$;
   $if\ q' \notin A(R, m_0)\ then\ \{enter\ q'\ in\ A(R, m_0); put\ q'\ onto\ Stack\}$
   $enter\ (q, t, q')in\ \rightarrow$
   }
   }

"Pseudo" car rien sa terminaison n'est pas assurée

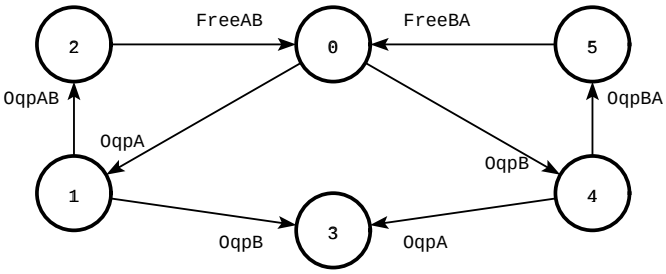
3.4 Exercice



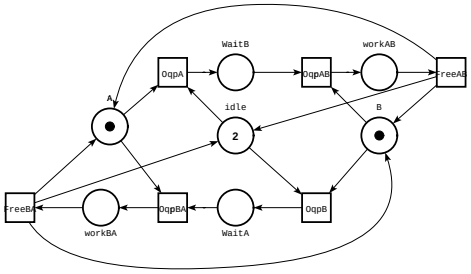
.../...

3.5.1 Partage de Ressources

Pb: Deux processus (banalisés) partagent deux ressources (A et B). L'un des processus se procure d'abord A (OqpA), puis se procure B (OqpAB), il travaille (WorkAB) puis il retourne (en même temps) au repos et libère (FreeAB) les ressources A et B. L'autre processus procède dans l'ordre inverse: obtention de B puis de A.



Réseau Associé



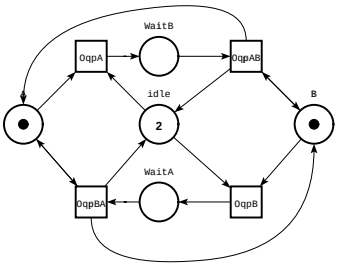
Marquages

```
MARKINGS:

0 : A B idle*2
1 : B WaitB idle
2 : idle workAB
3 : WaitA WaitB
4 : A WaitA idle
5 : idle workBA
```

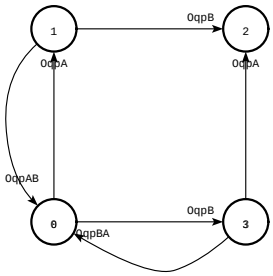

3.5.2 Partage de Ressources (variante)

Pb: Deux processus (banalisés) partagent deux ressources (A et B). L'un des processus se procure d'abord A (OqpA), puis se procure B (OqpAB) et retourne (en même temps) au repos et libère les ressources A et B. L'autre processus procède dans l'ordre inverse: obtention de B puis de A.

Réseau Associé



Graphe de marquages associé



Différentes notations pour représenter des marquages

Matricielle

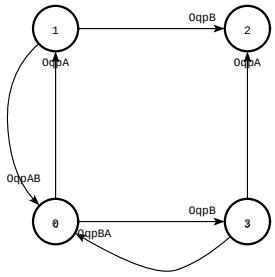
	0	1	2	3
m_A	1	0	0	1
m_B	1	1	0	0
mI	2	1	0	1
mWA	0	0	1	1
mWB	0	1	1	0

Vectorielle (combinaison linéaire)

$\vec{A}$	$\vec{B}$	$\vec{I}$	$\vec{WA}$	$\vec{WB}$
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

$\vec{0} = \vec{A} + \vec{B} + 2.\vec{I}$, $\vec{1} = \vec{B} + \vec{I} + \vec{WB}$, $\vec{2} = \vec{WA} + \vec{WB}$, etc ...

ou 0 : A B idle*2 , 1 : B WaitB idle , 2 : WaitA WaitB , 3 : A WaitA idle



Le graphe des marquages est fini (4 états) $\mapsto$ *réseau borné*

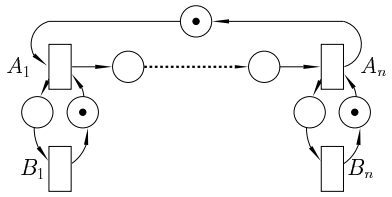
Le réseau (son graphe de marquages) admet la séquence infinie $(OqpA.OqpAB)^{\omega}$
$\mapsto$ *réseau infiniment actif*

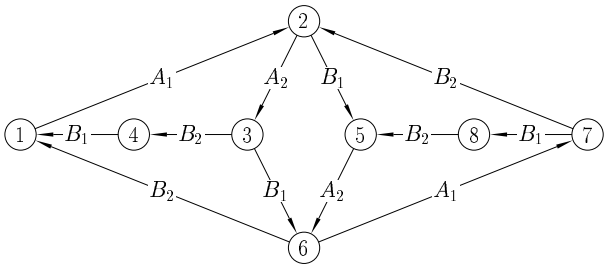
La séquence $OqpA.OqpAB.OqpB.OqpBA$ est possible à partir de 0.
Toutes les transitions du réseau sont possiblement sensibilisées
$\mapsto$ *réseau quasi-vivant*

La séquence $OqpA.OqpB$ conduit à un marquage mort
$\mapsto$ *réseau bloquant*

Il n'est pas toujours possible de revenir à l'état initial
$\mapsto$ *réseau non réinitialisable*

pb: n sites $(S_k : \ k \in [1, n])$ exécutent cycliquement A_k et B_K.
Un "ordonnanceur" donne la main cycliquement à chacun des sites.





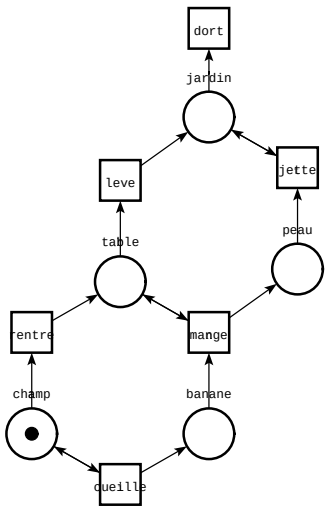
Borné, Propre, Vivant

Explosion Combinatoire: $n \times 2^n$ états et $(n^2 + n) \times 2^{n-1}$ arcs

3.5.4 La vie d'un planteur de bananes

(S. Haddad [Diaz 01]) Le planteur est initialement au champ où il cueille des bananes (en quantité supposé infinie). Il cesse son travail et se met à table pour manger quelques bananes. A la fin du repas, il passe au jardin et jette quelques unes des peaux de bananes mangées. Il part ensuite dormir.

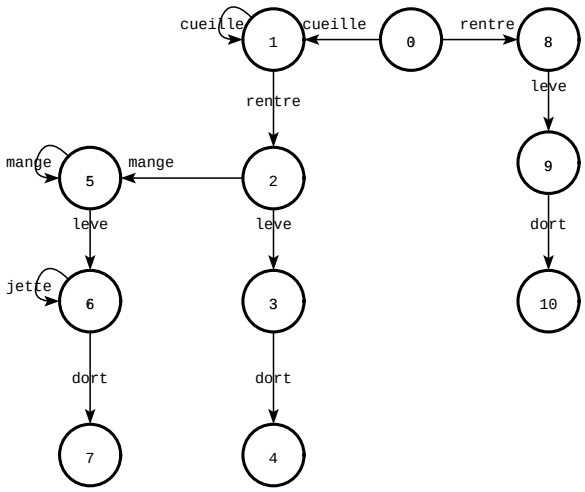
Réseau Associé



Réseau Bloquant non borné !

Impossible de construire le graphe des marquages.

Différentes journées du planteur



Comment arriver à synthétiser cette information ?

Monoïde E^* (Etoile de Kleene)

$< E, . >$ une loi de composition interne et associative
et ϵ l'élément neutre pour .

$$E^* = \bigcup_{n \geq 0} E^n \; et \; E^+ = \bigcup_{n > 0} E^n$$

où $E^0 = \{\epsilon\}$ et $E^k = \{e.\sigma | e \in E \text{ et } \sigma \in E^{k-1}\}$

$E = \{e_1, e_2\}$
$E^1 = E$, $E^2 = \{e_1.e_1, e_1.e_2, e_2.e_1, e_2.e_2\}$ ...

NB: E^* est infini dès que $E \neq \emptyset$

Langage Rationnel

Tout langage obtenu à partir des langages singletons (à un seul mot) par un nombre fini d'unions, de produits et d'étoiles.

Exemple: $\{a\}^*\{b\}.\{b\}.(\{a| \cup \{b\})^*$ dénoté par $a^*b.b(a+b)^*$

Expression rationnelle ($lettre, ., +, *$) (cf egrep/sed, Tcl/Tk, Perl ,,,)

Par analogie, on peut définir un Langage associé à un Rdp

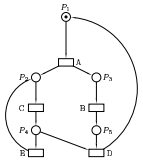
$L(R, m_0) = \{\omega \in T^* : m_0 \xrightarrow{\omega}\}$

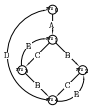
"Langage associé à un RdP est non rationnel"

$N =< R, m_0 > \& G(R, m_0)$

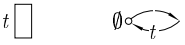
4.1 N est sans blocage

$\forall m \in G(R, m_0), \exists t \in T : m[\ t\ >$

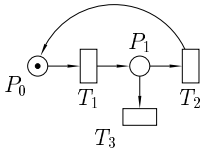


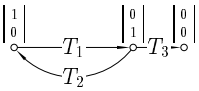


Plus petit réseau sans blocage



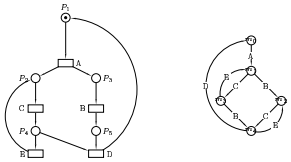
Réseau avec un état de blocage



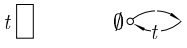


4.2 N est Réinitialisable (propre)

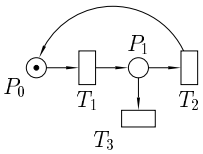
$\forall m \in G(R, m_0), \exists \omega \in T^+ : m[\ \omega\ >\ m_0$

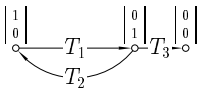


Plus petit réseau réinitialisable (propre)



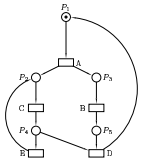
Réseau non réinitialisable

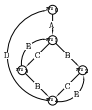




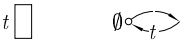
4.3 N est Quasi-Vivant

$\forall t \in T, \exists m \in G(R, m_0) : m[\ t\ >$

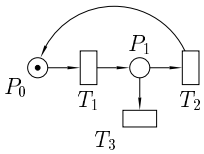


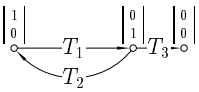


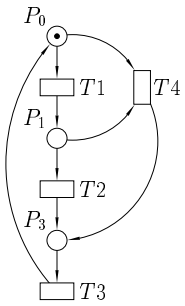
Plus petit réseau quasi-vivant

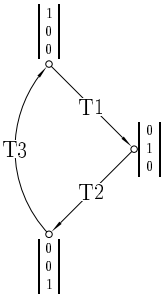


Réseau quasi-vivant



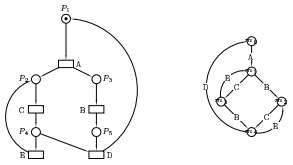






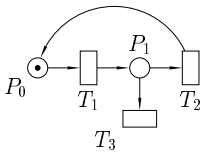
$L(R, m_0) = (T1.T2.T3)^*$

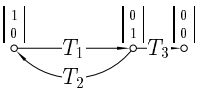
$\forall m \in G(R, m_0), \forall t \in T, \exists \omega \in T^* : m[\ \omega.t\ >$



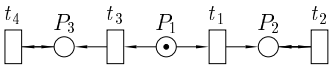
Plus petit réseau vivant t 

Réseau quasi-vivant et non vivant

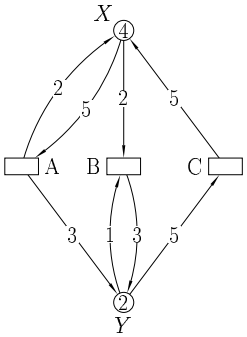


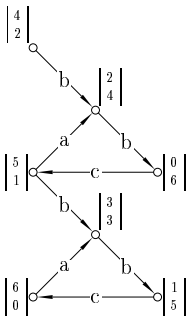


Réseau non bloquant sans aucune transition vivante



4.4.1 Vivant et non propre





$L(R, m_0) = bbc.(abc)^*.b.(bca)^*$

4.5 Relations entre les propriétés

Réinitialisable $\Rightarrow$ Sans blocage

Vivant $\Rightarrow$ Sans blocage

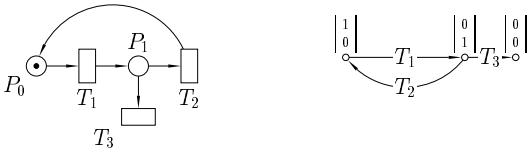
Réinitialisable $\not\Leftarrow$ Vivant

Réinitialisable + Quasi-Vivant $\Rightarrow$ Vivant

$< R, m_0 >$ est infiment actif ssi $\exists \sigma \in T^\omega \cap L(R, m_0)$

Plus petit réseau infiniment actif ($L = t^\omega$)

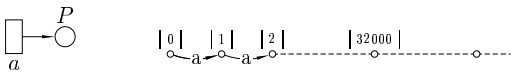
Réseau bloquant infiniment actif ($L = (T_1.T_2)^\omega$)



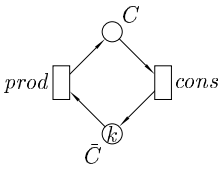
$\exists b \in IN : \forall m \in G(R, m_0), \forall p \in P : m(p) \leq b$

Plus petit réseau borné ($L = t^*$)

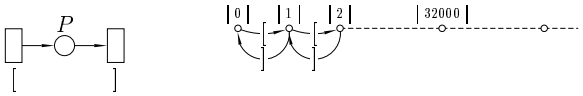
Plus petit réseau non borné ($L = a^*$)



Réseau k-Borné

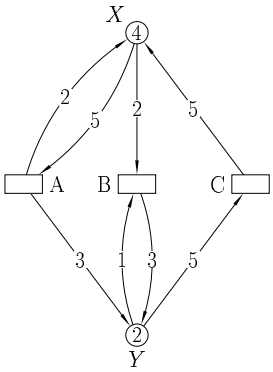


Non borné et à langage non rationnel

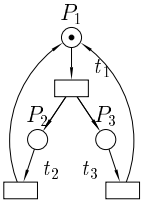


(Langage de Dick: expressions bien parenthésées)

Réseau 6-Borné



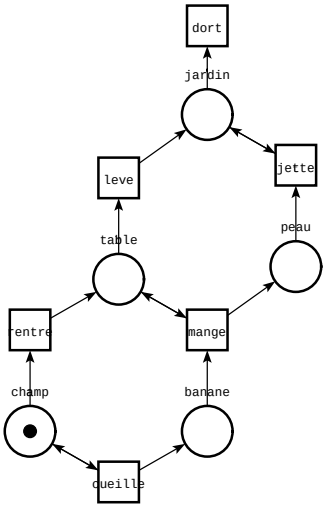
Réseau non borné
sans aucune
place bornée

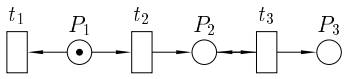


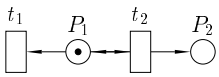
Propriété: Non Borné ⇒ Infiniment Actif (réciproque fausse)

4.7.1 Borné versus Bloquage

Réseau Planteur

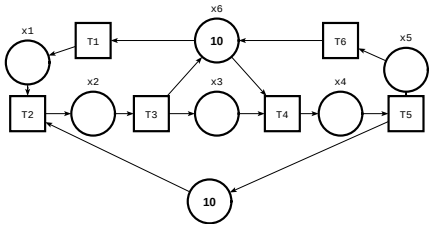






5.1 Piscine

pb: Une piscine comporte c cabines pour se changer et p paniers pour déposer ses vétements.
- On n'entre dans le piscine que si une cabine est libre. On attends un panier pour se changer et déposer ses vètements. On libère la cabine et on pénètre dans le bassin.
- On ne quitte le bassin que si une cabine est libre. On se change et on restitue cabine et panier. On quitte la piscine.



Légende des transitions
T1: le client entre à la piscine
(si ∃ cabine libre (x6))
T2: le client se déshabille
(si ∃ panier libre (x7))
T3: le client pénètre dans le bassin
(et restitue la cabine (x6))
T4: le client quitte le bassin
(si ∃ cabine libre (x6))
T5: le client s'habille
(et restitue le panier (x7))
T6: le client quitte la piscine.

Légende des places
x1: client en attente d'un panier
x2: client se déshabille
x3: client dans le bassin
x4: client s'habille
x5: client habillé prèt à sortir
x6: Compteur de cabines libres
x7: Compteur de paniers libres

Cas où $c = p = 2$: Scénario menant à un blocage

2 clients pénètrent dans la piscine (T1 puis T1)
plus de cabine

les 2 clients se déshabillent (T2 puis T2)
plus de panier

L'un des clients en maillot pénètre dans le bassin (T3)
un cabine libre

Un 3^{eme} client pénètre dans la piscine (T1)
plus de cabine

Le second client en maillot pénètre dans le bassin (T3)
un cabine libre

Un 3^{eme} client pénètre dans la piscine (T1)
plus de cabine

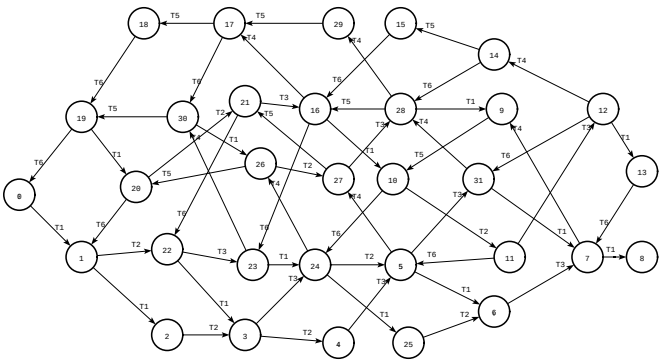
Bilan: 4 clients: 2 en attente et 2 dans le bassin
Plus de cabine, plus de panier

Marquages	Marquages
0 : x6*2 x7*2	0 ↦ T1/1
1 : x1 x6 x7*2	1 ↦ T1/2
2 : x1*2 x7*2	2 ↦ T2/3
3 : x1 x2 x7	3 ↦ T2/4
4 : x2*2	4 ↦ T3/5
5 : x2 x3 x6	5 ↦ T1/6
6 : x1 x2 x3	6 ↦ T3/7
7 : x1 x3*2 x6	7 ↦ T1/8
8 : x1*2 x3*2	8 ↦

Graphe des marquages pour $c = p = 2$

Taille: 32 marquages, 57 transitions

Blocage: 8 : x1(2) x3(2)



Pour $n = k = 10 \mapsto$ 7006 marquages, 28885 transitions

Pour $n = k = 15 \mapsto$ 38759 marquages 178703 transitions

remarque: Le réseau est bloqué pour tout marquage vérifiant $x6 = x7 = 0$
où x6: compteur de cabines et x7: compteur de paniers

Si $p < c$

On considère la séquence $T_1^c.T_2^p$

	T_1^c	T_2^p
$(x6)$ *cabines*	0	0
$(x7)$ *paniers*	p	0

Si $c = p$

On considère la séquence $T_1^c.T_2^c.T_3^c.T_1^c$

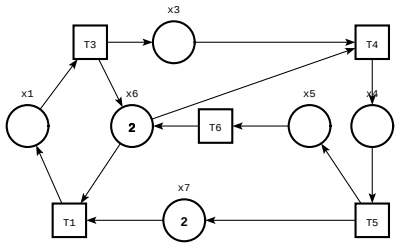
	T_1^c	T_2^c	T_3^c	T_1^c
$(x6)$ *cabines*	0	0	c	0
$(x7)$ *paniers*	c	0	0	0

Si $c < p$

On considère la séquence $T_1^c.T_2^c.T_3^c.T_1^c.T_2^{p-c}$

	T_1^c	T_2^c	T_3^c	T_1^c	T_2^{p-c}
$(x6)$ *cabines*	0	0	c	0	0
$(x7)$ *paniers*	c	$p-c$	$p-c$	$p-c$	0

On n'entre dans la piscine que si une cabine et un panier sont libres.

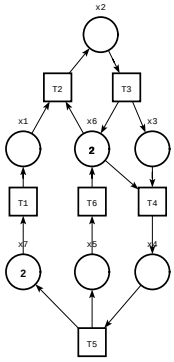


Pour $c = p = 2 \mapsto 19$ marquages, 33 transitions

Borné, Propre, Vivant

Une autre solution

On n'entre dans la piscine que si un panier est libre.

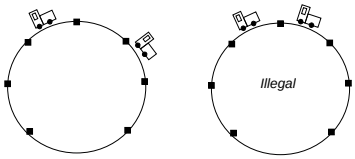


Pour $c = p = 2 \mapsto 32$ marquages, 62 transitions

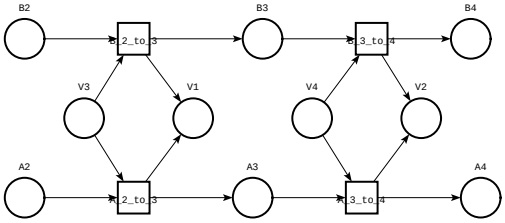
Borné, Propre, Vivant

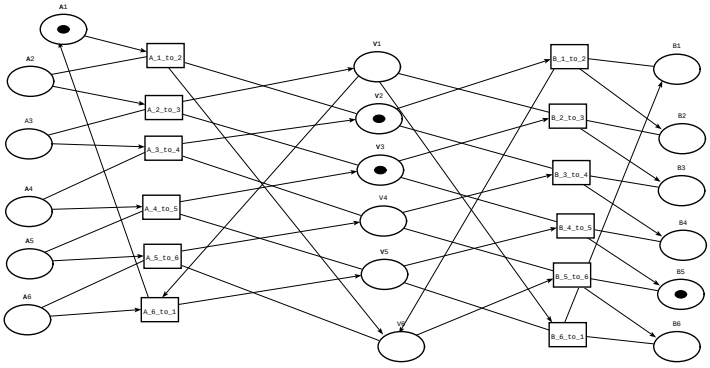
5.2 Trains de Genrich

pb: n (2) trains circulent (dans le même sens) sur une voie ferrée circulaire. La voie est découpé en sections. Par mesure de sécurité, il doit y avoir une section libre entre deux trains.



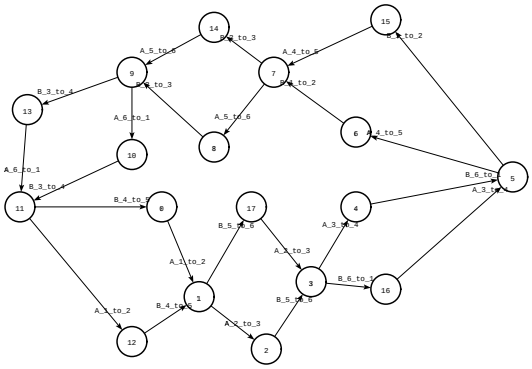
Simplification: un bus de trains





Taille: 18 marquages, 24 transitions

Borné, Propre, Vivant



L'anneau de trains en textuel

```
net {trains 6}
tr A_2_to_3 A2 V3 -> A3 V1
tr B_2_to_3 B2 V3 -> B3 V1
tr A_3_to_4 A3 V4 -> A4 V2
tr B_3_to_4 B3 V4 -> B4 V2
tr A_4_to_5 A4 V5 -> A5 V3
tr B_4_to_5 B4 V5 -> B5 V3
tr A_5_to_6 A5 V6 -> A6 V4
tr B_5_to_6 B5 V6 -> B6 V4
tr A_6_to_1 A6 V1 -> A1 V5
tr A_1_to_2 A1 V2  -> A2 V6
tr B_6_to_1 B6 V1 -> B1 V5
tr B_1_to_2 B1 V2  -> B2 V6
```

Le script pour le générer

```
#!/bin/sh
# \
exec tclsh "$0" -- "$@"
# script Tcl/TK pour generer le reseau des trains
proc main {} {
    global argv
    if {"" == [set n [lindex $argv 1]]} {set n 4}
    puts "net \{trains $n\}"
    for {set i 2} {$i < $n} {incr i} {
    puts "tr A_${i}_to_[expr $i + 1] A${i} V[expr $i + 1] -> A[expr $i + 1] V[expr $i - 1]"
    puts "tr B_${i}_to_[expr $i + 1] B${i} V[expr $i + 1] -> B[expr $i + 1] V[expr $i - 1]"
    }
    puts "tr A_${n}_to_1 A${n} V1 -> A1 V[expr $i - 1]"
    puts "tr A_1_to_2 A1 V2  -> A2 V${i}"
    puts "tr B_${n}_to_1 B${n} V1 -> B1 V[expr $i - 1]"
    puts "tr B_1_to_2 B1 V2  -> B2 V${i}"
}
main
```


5.3 Une histoire de pont

Un groupe de 4 personnes (A,B,C,D) se situe sur la rive gauche d'un fleuve et doit se rendre sur la rive droite.

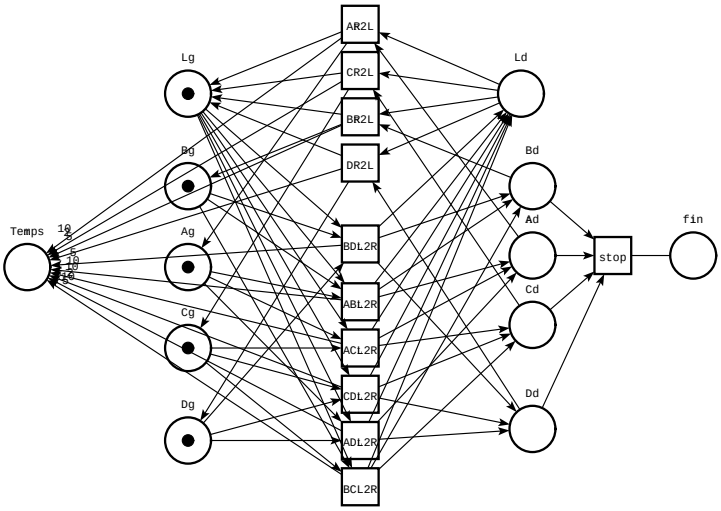
Pour ce faire, ils doivent emprunter un pont mal éclairé qui ne peut supporter qu'une charge de deux personnes. Le groupe dispose d'une seule lampe de poche. Chaque traversée du pont nécessite la possession de la lampe. Il est donc nécessaire de ramener la lampe de l'autre coté pour permettre une nouvelle traversée.

Les 4 personnes marchent à une vitesse différente. Les temps de traversée pour chacun des individus est respectivement de 10 min pour A, 5 min pour B, 2 min pour C et 1 min pour D.

Quel est le temps minimal pour faire passer les 4 personnes sur la rive droite ?

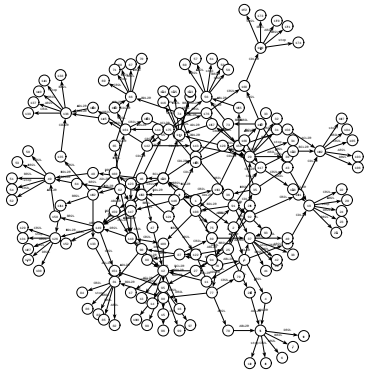
Solution:

Passage de C et D	2 min
Retour de C	4 min
Passage de A et B	14 min
Retour de D	15 min
Passage de C et D	17 min



Taille: 177 marquages, 237 transitions

Borné, Bloquant (et c'est tant mieux!)



Analye

Tous les états de blocage marquent la place FIN
$\Rightarrow$ Les états de blocage correspondent à la fin du "calcul"

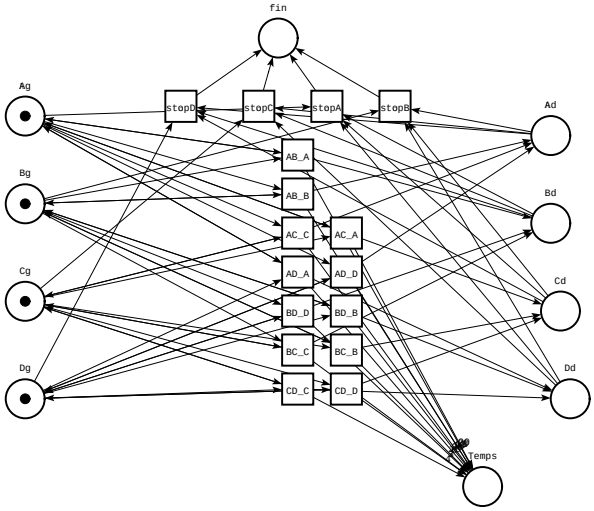
Une analyse des "états terminaux" permet de récupérer les $\neq$ temps de "transit"

Temps d'exécution possibles: 17, 19, 20 , 21, 23, 24, 26,
27, 30, 33, 34, 36, 37, 40, 50

Une mauvaise solution

principe: Chaque transition représente le passage de gauche à droite de deux personnes et le retour à gauche de l'une d'entre-elles. Plus besoin de matérialiser la lampe.

nb: XY_X matérialise le passage de gauche à droite de X et de Y et le retour à gauche de X.



Le même en textuel (plus clair)

```
tr AB_A Ag Bg -> Ag Bd Temps*20
tr AB_B Ag Bg -> Ad Bg Temps*15
tr AC_A Ag Cg -> Ag Cd Temps*20
tr AC_C Ag Cg -> Ad Cg Temps*12
tr AD_A Ag Dg -> Ag Dd Temps*20
tr AD_D Ag Dg -> Ad Dg Temps*11
tr BC_B Bg Cg -> Bg Cd Temps*10
tr BC_C Bg Cg -> Bd Cg Temps*7
tr BD_B Bg Dg -> Bg Dd Temps*10
tr BD_D Bg Dg -> Bd Dg Temps*6
tr CD_C Cg Dg -> Cg Dd Temps*4
tr CD_D Cg Dg -> Cd Dg Temps*3
tr stopA Ag Bd Cd Dd -> fin
tr stopB Ad Bg Cd Dd -> fin
tr stopC Ad Bd Cg Dd -> fin
tr stopD Ad Bd Cd Dg -> fin
pl Ag (1)
pl Bg (1)
pl Cg (1)
pl Dg (1)
```

Analye

Tous les états de blocage marquent la place FIN

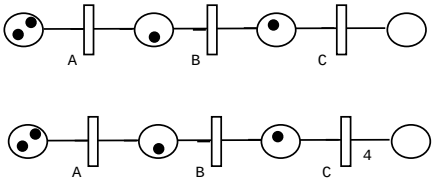
Plus petit temps d'exécution possible: 19 !!

Il manque donc des solutions !

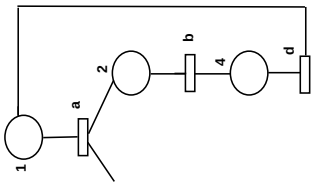
- **Séquentialité** *Dépendance causale*
- **Concurrence** *Indépendance causale*
- **Non-déterminisme** *Une cause peut avoir plusieurs effets* $\neq$
- **Synchronisation** *Plusieurs causes doivent s'être produites avant que l'effet puisse avoir lieu*
- **Compétition**
- **Communication** *Transfert d'informations*

Processus Séquentiels:

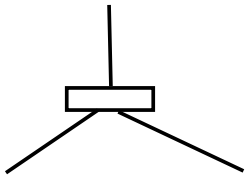
Plusieurs processus qui exécutent le même code (dans un état d'exécution possiblement différent)



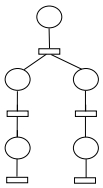
Répétition



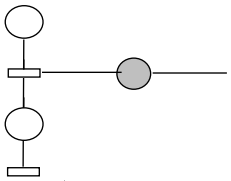
6.2 Fork



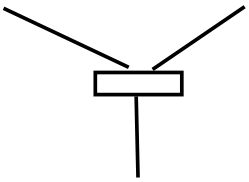
Création de Processus, Lancement de tâches en //



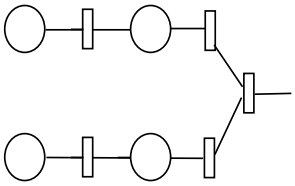
Communication: Envoi de Messages



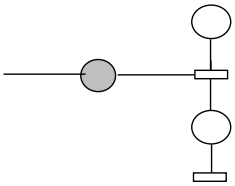
6.3 Join



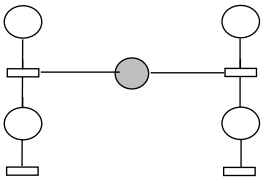
Synchronisation



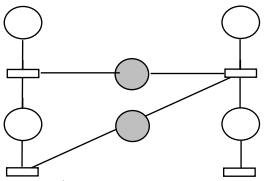
Communication: Réception de Messages



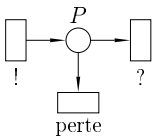
Appel/Réponse



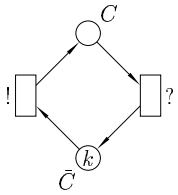
Variante

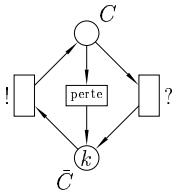


Avec Perte et Sans contrôle de Flux

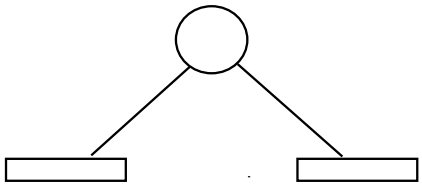


Avec contrôle de Flux

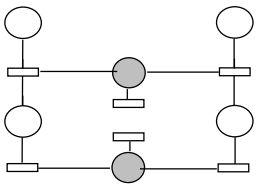




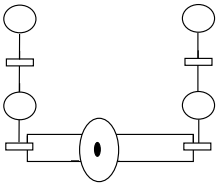
6.5 Choix



Non-Déterminisme



Conflit



6.6 Conflit & Indépendance

Conflit Structurel

t_1 et t_2 sont en conflit structurel ssi

$\exists p \in P : Pre(p, t_1) \times Pre(p, t_2) \neq 0$

Conflit Effectif

t_1 et t_2 sont en conflit effectif pour un marquage M ssi

(1) t_1 et t_2 sont en conflit structurel
(2) $M[\ t_1\ > \ \&\ M[\ t_2\ >$
(3) $\exists p : M(P) < Pre(p, t_1) + Pre(p, t_2)$

Indépendance Structurelle

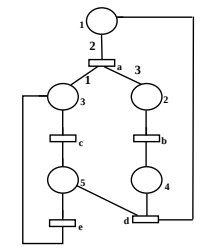
t_1 et t_2 sont structurellement indépendantes ssi

$\forall p \in P : Pre(p, t_1) \times Pre(p, t_2) = 0$

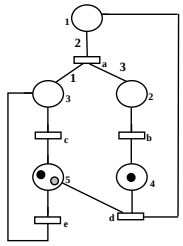
Indépendance Effective

t_1 et t_2 sont effectivement indépendantes pour un marquage M ssi

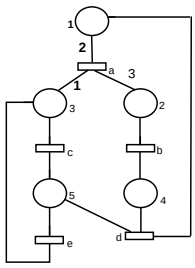
(1) t_1 et t_2 sont structurellement indépendantes
(2) $M[\ t_1\ > \&\ M[\ t_2\ >$



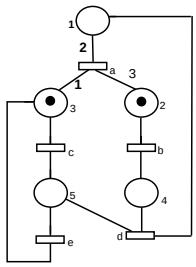
Structurel



Effectif ●
Non−Effectif ● ○



Structurelle



Effective

7.1 Matrices d'un réseau

Réseau est défini via deux matrices: $Pre, Post : P \times T \mapsto IN$

$\mapsto$ Matrice d'Incidence $C : P \times T \mapsto \mathbb{Z}$

C = Post - Pre

$(\forall p \in P, \forall t \in T : C(p,t) = Post(p,t) - Pre(p,t))$

Post	a	b	c	d	e
1	0	0	0	1	0
2	1	0	0	0	0
3	1	0	0	0	1
4	0	1	0	0	0
5	0	0	1	0	0

-

Pre	a	b	c	d	e
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	1	1

=

C	a	b	c	d	e
1	-1	0	0	1	0
2	1	-1	0	0	0
3	1	0	-1	0	1
4	0	1	0	-1	0
5	0	0	1	-1	-1

7.2 Réseau est Pur

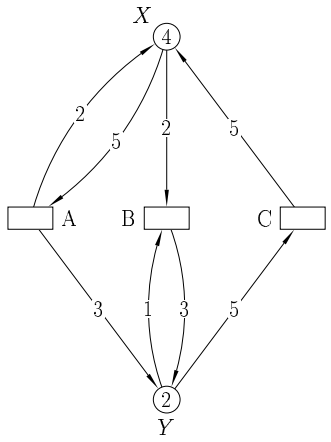
$\forall p \in P, \forall t \in T : Pre(p,t) \times Post(p,t) = 0$

Propriété des réseaux Purs $C \leftrightarrow Pre, Post$

Pre(p,t) = - Max (0, - C(p,t))

Post(p,t) = Max (0, C(p,t))

Exemple



Pre	*a*	*b*	*c*
	5	2	0
	0	1	5

Post	*a*	*b*	*c*
	2	0	5
	3	3	0

C	*a*	*b*	*c*
	−3	−2	5
	3	2	−5

7.3 Image commutative d'une séquence de transitions

$^- : T^* \mapsto IN^t$ définie par $\bar{\sigma}(t) = |\sigma|_t$

où $|\ |_\ : T^* \times T \mapsto IN$ est définie par

$$|\epsilon|_t = 0 \qquad \text{et} \qquad |u.\sigma|_t = \begin{cases} 1 + |\sigma|_t & \text{si } u = t \\ |\sigma|_t & \text{sinon} \end{cases}$$

Exemple $T = \{t1, t2, t3, t4, t5\}$

$\sigma_1 = t1.t2.t2.t3.t5$ $\quad |\sigma_1|_{t1} = |\sigma_1|_{t3} = |\sigma_1|_{t5} = 1,\ |\sigma_1|_{t2} = 2,\ |\sigma_1|_{t4} = 0$ $\quad \bar{\sigma_1} = \begin{vmatrix} 1 \\ 2 \\ 1 \\ 0 \\ 1 \end{vmatrix}$

$\sigma_2 = t4.t2.t1.t3.t5$ $\quad |\sigma_1|_t = 1 \ \ \forall t \in T$ $\quad \bar{\sigma_1} = \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix}$

$\sigma_3 = t5.t3.t2.t1.t2$ $\quad |\sigma_3|_{t1} = |\sigma_1|_{t3} = |\sigma_1|_{t5} = 1,\ |\sigma_1|_{t2} = 2,\ |\sigma_1|_{t4} = 0$ $\quad \bar{\sigma_3} = \begin{vmatrix} 1 \\ 2 \\ 1 \\ 0 \\ 1 \end{vmatrix}$

NB $^-$ n'est pas injective ($\sigma_3 \neq \sigma_1$ ET $\bar{\sigma_3} = \bar{\sigma_1}$)

Soient $m_1, m_2 \in AR(R, m_0), \sigma \in T^*$

Si $m_1 [\sigma > m_2$ **Alors** $m_2 = m_1 + C.\bar{\sigma}$

Preuve: par récurrence sur $|\sigma|$

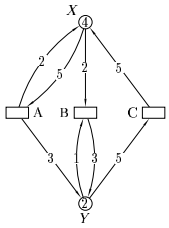
La Réciproque est **FAUSSE** car

- (1) on travaille avec $\bar{\sigma}$ et non σ
- (2) on raisonne avec C et non avec Pre & $Post$

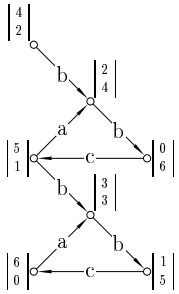
Corollaire

Soient $m_1, m_2 \in A(R, m_0)$ et $\sigma \in T^*$
tels que $m_2 = m_1 + C.\bar{\sigma}$

Si $\exists p \in P : m_2(p) < 0$ Alors NON $(m_1 [\sigma >)$



C	a	b	c
	-3	-2	5
	3	2	-5



Si $S = BBCA$ alors $\bar{S} = \begin{vmatrix} 1 \\ 2 \\ 1 \end{vmatrix}$ et $C.\bar{S} = \begin{vmatrix} -3 & -2 & 5 \\ 3 & 2 & -5 \end{vmatrix} \otimes \begin{vmatrix} 1 \\ 2 \\ 1 \end{vmatrix} = \begin{vmatrix} -2 \\ 2 \end{vmatrix}$

Rappel: **Si** $m_1 [\sigma > m_2$ **Alors** $m_2 = m_1 + C.\bar{\sigma}$

Ici $\begin{vmatrix} 4 \\ 2 \end{vmatrix} [BBCA > M'$ avec $M' = \begin{vmatrix} 4 \\ 2 \end{vmatrix} + \begin{vmatrix} -2 \\ 2 \end{vmatrix} = \begin{vmatrix} 2 \\ 4 \end{vmatrix}$

De même, $\begin{vmatrix} 5 \\ 1 \end{vmatrix} [BBCA > M''$ avec $M'' = \begin{vmatrix} 5 \\ 1 \end{vmatrix} + \begin{vmatrix} -2 \\ 2 \end{vmatrix} = \begin{vmatrix} 3 \\ 3 \end{vmatrix}$

8 Algorithme de Décision de la bornitude

Problème: Etre capable de savoir si un réseau de Petri est borné ou non.

nb: le "pseudo"-algorithme de construction de graphe des marquages n'apporte de réponse que si le réseau est borné .....

Grandes étapes pour y arriver ....

Propriété de Monotonie

Caractérisation des réseaux infiniment Actifs

Caractérisation des réseaux non Bornés

Algorithme de Décision

* Arbre de Couverture
* Graphe de Couverture

8.1 Propriété de Monotonie

Règles de Sensibilisation et de Tir (rappels)

- Une transition t est sensibilisée pour un marquage M
$M [t >$ ssi $\forall p \in P : M(p) \geq Pre(p, t)$

- *Soit t une transition et M un marquage tel que $M [t >$ alors $M [t > M'$*

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

(ou aussi $M' = M - Pre(., t) + Post(., t)$)

Remarques:

- $Pre(., t)$ est le marquage minimum permettant de tirer t
- $Post(., t)$ est le marquage minimum que l'on peut atteindre après le tir de t
- L'évolution des marquages est indépendante du marquage de départ $M' - M = Post(., t) - Pre(., t))$

- $\forall M, M' \in G(R, m_0)$ tels que $M' \geq M$

  $\forall t \in T : \quad M\ [t > \Rightarrow M'\ [t >$

  $\forall \sigma \in T^* : \quad M\ [\sigma > \Rightarrow M'\ [\sigma >$

  (par récurrence sur $|\sigma|$)

- $\forall M_1, M_2 \in G(R, m_0), \forall \sigma \in T^*$: tels que $M_1\ [\sigma > M_2$

  Si $M_1' > M_1$ alors $M_1'\ [\sigma > M_2'$ avec

  $M_2' - M_1' = M_2 - M_1 = C.\bar{\sigma}$

  $M_1' - M_1 = M_2' - M_2$

Corollaire (monotonie)

$$\forall m_0, m_0' \in IN^P : m_0 < m_0' \Rightarrow \begin{array}{l} |A(R, m_0)| \leq |A(R, m_0')| \\ |G(R, m_0)| \leq |G(R, m_0')| \\ |L(R, m_0)| \leq |L(R, m_0')| \end{array}$$

$< R, m_0 >$ est infiment actif ssi

il existe une séquence infinie $\exists \sigma \in T^\omega \cap L(R, m_0)$

Non borné $\Rightarrow$ Infiniment Actif (réciproque fausse)

Séquence Répétitive

Soit $\sigma \in T^*, \sigma$ est Répétitive ssi:

$\exists M \in G(R, M_0) : M\ [\sigma > \ \&\ M'\ [\sigma > M" \Rightarrow M"\ [\sigma >$

$$\Leftrightarrow \begin{cases} \exists v \in T^* : v.\sigma \in L(R, M_0) \\ \forall v \in T^* : v.\sigma \in L(R, M_0) \Rightarrow v.(\sigma)^i \in L(R, M_0) \quad \forall i \geq 0 \end{cases}$$

Caractérisation des séquences répétitives

Soit σ une séquence de **franchissement** pour $< R, m_0 >$

$\exists M \in G(R, m_0) : M\ [\sigma >$ alors $(i) \Leftrightarrow (2) \Leftrightarrow (3)$

(1) $\exists M, M' \in G(R, m_0) : M\ [\sigma > M'$ et $M' \geq M$

(2) $\forall M, M' \in G(R, m_0)$ tels que $M\ [\sigma > M'$ Alors $M' \geq M$

(3) σ est une séquence répétitive

8.2.1 Caractérisation des Réseaux Infiniment Actifs

Propriété Un réseau $< R, m_0 >$ est infiniment actif ssi $G(R, m_0)$ possède une séquence répétitive.

$\Leftarrow$ (évident)

$G(R, m_0)$ possède une séquence répétitive σ

$\equiv_{def} \exists v \in T^*, \forall i \geq 0 : v.\sigma^i \in L(R, M_0)$

Donc $L(R, M_0)$ possède une séquence infinie
$\equiv_{def} < R, m_0 >$ est infiniment actif

$\Rightarrow$ (plus subtil)

$< R, m_0 >$ est infiniment actif $\equiv_{def} \exists \sigma \in T^\omega \cap L(R, m_0)$

A partir de σ on peut construire $v_\sigma = v_0, v_1, \ldots v_k \ldots$ une suite infinie de marquages

TH: $U = u_0, u_1, \ldots u_k \ldots$ une suite infinie d'éléments $\in IN^m$
Alors on peut extraire de U une sous-suite infinie croissante:
$u_{i_0} \leq u_{i1} \leq \ldots \leq u_{ik} \ldots$

$\ldots \sigma \mapsto v_\sigma = v_0, v_1, \ldots v_k \ldots$ une suite infinie $\in IN^p$

TH $\mapsto$ sous-suite infinie $u_{i_0} \leq u_{i1} \leq \ldots \leq u_{ik} \ldots$

On considère u_{i_0} et u_{i_1}

Par construction de U,
$\exists s_1, s_2 \in T^* : m_0\ [s_1 > u_{i_0}$ et $u_{i_0}\ [s_2 > u_{i_1}$

Comme $u_{i_0} \leq u_{i1}, s_2$ est une **séquence répétitive**

(cf caractérisation des séquences répétitives)

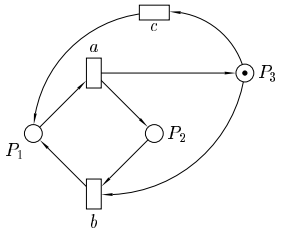
cqfd (modulo TH)

TH découle de $(< IN, <>)$ est un bon ordre

$< E, <>$ est un **Bon Ordre** ssi
$\forall P \subset E : P \neq \emptyset \Rightarrow \exists p \in P : \forall p' \in P : p' = p$ ou $p < p'$

Rappel Réseau $< R, m_0 >$ est borné ssi
$\forall p \in P, \exists k \in IN : \forall m \in G(R, m_0) \;\; m(p) \geq k$

Def: Séquence σ est Répétitive croissante pour une place p ssi $\forall m, m' \in G(R, m_0) : m\,[\sigma > m' \Rightarrow m' \geq m \,\mathrm{ET}\, m'(p) > m(p)$



$$\left|\begin{matrix}1\\0\\0\\\circ\end{matrix}\right| \quad \left|\begin{matrix}1\\1\\0\\\circ\end{matrix}\right| \quad \left|\begin{matrix}1\\2\\0\\\circ\end{matrix}\right|$$

$$\left|\begin{matrix}\circ\\0\\0\\1\end{matrix}\right| \quad \left|\begin{matrix}\circ\\0\\1\\1\end{matrix}\right| \quad \left|\begin{matrix}\circ\\0\\2\\1\end{matrix}\right| \quad \left|\begin{matrix}\circ\\0\\3\\1\end{matrix}\right|$$

Un réseau $< R, m_0 >$ est **Non borné** ssi
$\exists$ séquence σ répétitive croissante pour une place p
et un marquage $m \in G(R, m_0)$ tel que $m\,[\sigma >$

$(\Leftarrow)$ (évident)

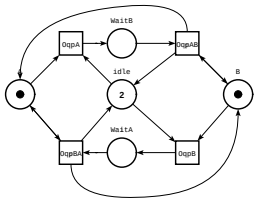
$(\Rightarrow)$

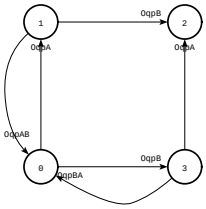
8.4 Arbre de Couverture

Détecter toutes les places non bornées
$\Updownarrow$
Recherche des séquences répétitives croissantes

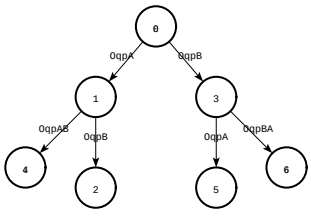
$\mapsto$ 2 aspects: Arbre/Graphe & Couverture

Arbre versus Graphe





0 : A B idle*2 , 1 : B WaitB idle , 2 : WaitA WaitB , 3 : A WaitA idle



Couverture

$< R, m_0 > \longmapsto AC(R, m_0)$ (Arbre fini)

dont l'ensemble des sommets "couvre" $A(R, m_0)$

(E "*couvre*" F ssi $\forall f \in F, \exists e \in E : e \geq f$)

IN_ω Complétion de IN

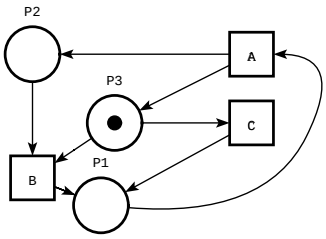
$IN_\omega = IN \cup \{\omega\}$

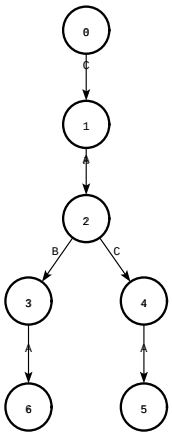
- $\forall n \in IN : n < \omega$
- $n + \omega = \omega + n = \omega$
  $\omega - n = \omega$ ($n - \omega$ n'est pas défini)

On étend de la même manière $+, -, <$ à IN_ω^p

Au lieu de considèrer des marquages dans IN^p,

on va considérer des marquages dans IN_ω^p





0 : P3
1 : P1
2 : P2*w P3
3 : P1 P2*w
4 : P2*w P3
5 : P2*w P3

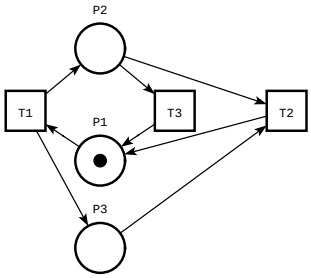
Soit q un sommet étiqueté par m

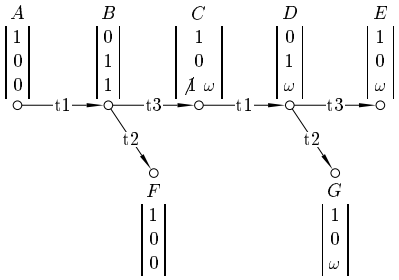
Si q admet un ancêtre p ayant la même étiquette
Alors q n'a pas de fils.
Sinon
Pour chaque transition t sensibilisée en m
q admet un fils q_t,
l'arc reliant q à q_t est étiqueté par t
chaque sommet q_t est étiqueté par $\Omega(m_t)$
avec m_t tel que $m[t > m_t$

Définition de $\Omega(m_t)(p)$

Pour chaque place $p \in P$
Si q_t admet un ancêtre a étiquetté par m_a
avec $m_a \geq m_t$ et $m_a(p) > m_t(p)$
Alors $\Omega(m_t)(p) = \infty$ Sinon $\Omega(m_t)(p) = m_t(p)$

8.5.1 Exemple





8.5.2 Propriétés de l'arbre de Couverture

[Karp-Miller 69]

- $\forall < R, m_0 >$: $AC(R, m_0)$ est fini

- Un réseau $< R, m_0 >$ est **non borné** ssi
  $\exists\, q \in AC(R, m_0), \exists p \in P$ tel que $q(p) = \omega$

- Pour un réseau $< R, m_0 >$, une place p est **non bornée**
  ssi $\exists q \in AC(R, m_0)$ tel que $q(p) = \omega$

- Pour un réseau $< R, m_0 >$, une place bornée p
  a pour borne $Max(\{q(p) : q \in AC(R, m_0)\})$

rappel: $AC(R, m_0) =< S, \rightarrow, L >$ avec $S = IN \times A(R, m_0)$

Soit $\equiv \subset S \times S$ défini par $(n, m) \equiv (n', m')$ ssi $m = m'$

$\equiv$ est une relation d'équivalence

$$GC(R, m_0) = AC(R, m_0)/_{\equiv}$$

$GC(R, m_0) =< S/_{\equiv}, \rightarrow_{\equiv}, L >$

où $S/_{\equiv}$ l'ensemble quotient de S par $\equiv$

et $\rightarrow_{\equiv}$, la plus petite relation vérifiant:

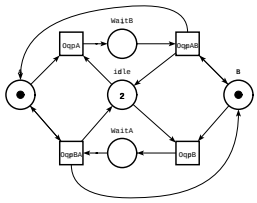
Si $s \rightarrow q$ avec $s \in s_{\equiv}$ et $q \in q_{\equiv}$ Alors $s_{\equiv} \rightarrow_{\equiv} q_{\equiv}$

8.6.1 Propriétés du graphe de couverture

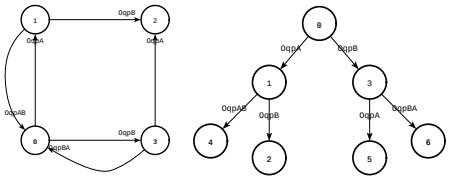
- Si $< R, m_0 >$ est borné alors $GC(R, m_0) = G(R, m_0)$
- Si $m_0 \,[x > m$ alors dans $GC(R, m_0)$ aussi $m_0 \xrightarrow{x} m$

Réciproque fausse: *L'introduction des ω-marquages lors de la construction de l'arbre de couverture constitue une majoration un peu brutale qui se traduit par l'apparition sur le graphe de couverture de séquences de franchissement qui n'existent pas en réalité.*

Cas des réseaux Bornés

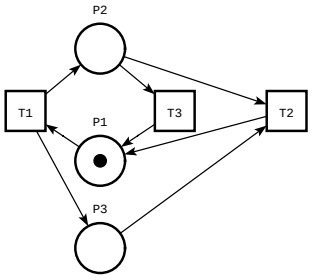


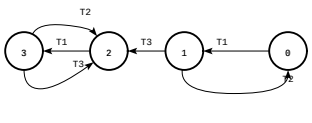
Graphe de couverture = Graphe des marquages

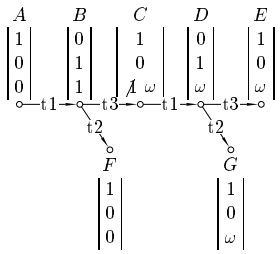


0 : A B idle*2 , 1 : B WaitB idle , 2 : WaitA WaitB , 3 : A WaitA idle

Cas général

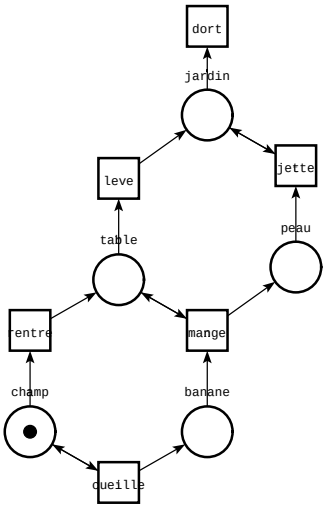


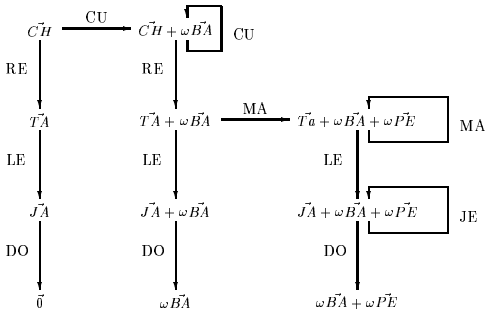


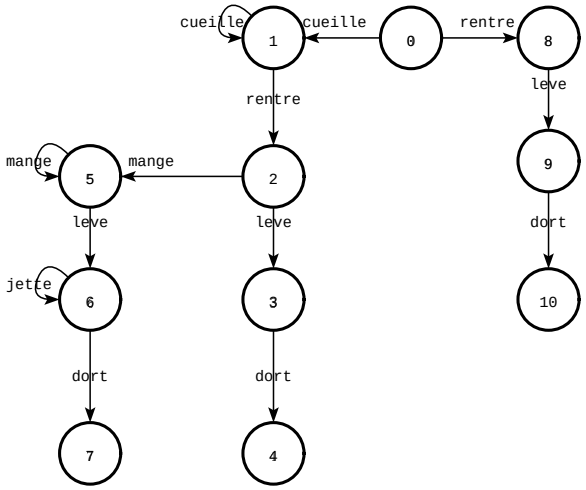


Retour sur le réseau planteur

Réseau Associé





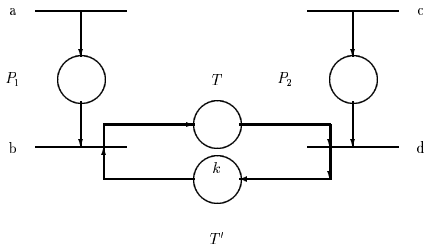


La séquence $Cueille.Rentre.Mange.Mange.Mange$ existe dans le graphe de couverture sans être pour autant une séquence valide de franchissement : le planteur ne peut manger plus de bananes qu'il en a cueilli.

9 Analyse Structurelle

9.1 Intuition

Producteur/Consommateur



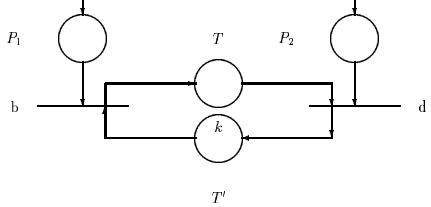
On considère les places T et T'

• a et c ne sont pas connectées aux places T et T'

a et b n'affectent pas les marquages de T et de T'

Pour $x \in \{a, c\}$, Si $m\ [x > m'$ alors

$m'(T) = m(T)$ et $m'(T') = m(T')$



T'

- Si $m\ [b > m'$ alors
$m'(T) = m(T) + 1$ et $m'(T') = m(T') - 1$

- Si $m\ [d > m'$ alors
$m'(T) = m(T) - 1$ et $m'(T') = m(T') + 1$

Remarque pour $t \in \{b, d\}$
Si $m\ [t > m'$ alors $m(T) + m(T') = m'(T) + m'(T')$

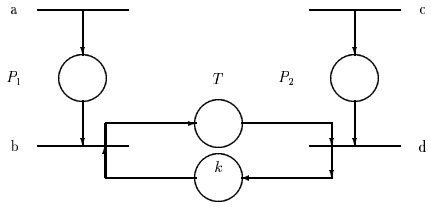
Bilan

1) a ou c laisse $m(T)$ et $m(T')$ invariants

2) b ou d laisse la **Somme** $m(T) + m(T')$ invariante

(1) & (2) $\Rightarrow m(T) + m(T')$ est invariante

$$\forall m \in G(R, m_0) : m(T) + m(T') = m_0(T) + m_0(T')$$



T'

Bilan Si $m\ [a.b.c.d > m'$ Alors $m = m'$

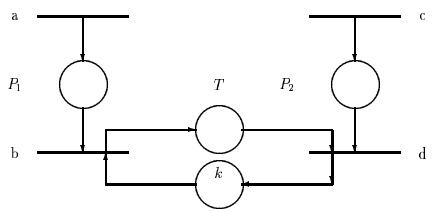
$$\begin{vmatrix} P_1 \\ P_2 \\ T \\ T' \end{vmatrix} \xrightarrow{a.b.c.d} \begin{vmatrix} (P_1+1)-1 \\ (P_2+1)-1 \\ (T+1)-1 \\ (T'-1)+1 \end{vmatrix} = \begin{vmatrix} P_1 \\ P_2 \\ T \\ T' \end{vmatrix}$$

Vérification possible

$m_1\ [\sigma > m_2 \Rightarrow m_2 = m_1 + C.\bar{\sigma}$

C	a	b	c	d
P_1	1	−1	0	0
P_2	0	0	1	−1
T	0	1	0	−1
T'	0	−1	0	−1

$$\times \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$



T'

- On considère a et b et m tel que $m\ [a.b > m'$

$$\begin{vmatrix} P_1 \\ P_2 \\ T \\ T' \end{vmatrix} \xrightarrow{a} \begin{vmatrix} P_1+1 \\ P_2 \\ T \\ T' \end{vmatrix} \xrightarrow{b} \begin{vmatrix} (P_1+1)-1 \\ P_2 \\ T+1 \\ T'-1 \end{vmatrix} = \begin{vmatrix} P_1 \\ P_2 \\ T+1 \\ T'-1 \end{vmatrix}$$

- Pour c et d et m tel que $m\ [c.d > m'$

$$\begin{vmatrix} P_1 \\ P_2 \\ T \\ T' \end{vmatrix} \xrightarrow{c} \begin{vmatrix} P_1 \\ P_2+1 \\ T \\ T' \end{vmatrix} \xrightarrow{d} \begin{vmatrix} P_1 \\ (P_2+1)-1 \\ T-1 \\ T'+1 \end{vmatrix} = \begin{vmatrix} P_1 \\ P_2 \\ T-1 \\ T'+1 \end{vmatrix}$$

9.2 Analyse Structurelle (principe)

Exploiter l'équation Fondamentale des RdP

Eq Fond Rdp: Soient $m_1, m_2 \in G(R, m_0), \sigma \in T^*$

Si $m_1\ [\sigma > m_2$ **Alors** $m_2 = m_1 + C.\bar{\sigma}$

Corollaire de l'équation fondamentale:

$$\forall m \in G(R, m_0), \exists \sigma \in T^*\ m = m_0 + C.\bar{\sigma}$$

Analyse structurelle $\equiv$ **"éliminer"** $C.\bar{\sigma}$

- "Attaquer" C à gauche : Trouver F telle que $f.C = 0$
$\longmapsto F.m = F.m_0$

- Chercher les σ vérifiant $C.\bar{\sigma} = 0$
$\longmapsto m_0\ [\sigma > m$ avec $m_0 = m$

Soit F telle que $f.C = 0 \quad \longmapsto F.m = F.m_0 \quad (E)$

$(P = \{p_1, p_2 \ldots p_n\})$

$$\sum_{j=1}^{j=p} f_j \times m(p_j) = \sum_{j=1}^{j=p} f_j \times m_0(p_j) = Cste(m_0) \qquad (E)$$

$(E) \Rightarrow \forall i \in [1, p]$

$$f_i \times m(p_i) = Cste(m_0) - (\sum_{j \in [1,p] \setminus \{i\}} f_j \times m(p_j))$$

Si $f_i \neq 0$ Alors

$$m(p_i) = 1/f_i \times (Cste(m_0) - (\sum_{j \in [1,p] \setminus \{i\}} f_j \times m(p_j))$$

Si **de plus** $F > 0$ (i.e $\forall i : f_i > 0$)

Alors $m(p_i) \leq Cste(m_0)/f_i$**Et** p_i **est bornée**

$B \subseteq P$ est un invariant de Place ssi

$\exists F \geq 0$ tq $||F|| = B$ et $f.C = 0$

où $||F|| = \{i \in [1, Card(P)] : F_i \neq 0\}$

Propriété des Invariants de Place

Si B est un invariant de place, alors
toute place p de B est bornée

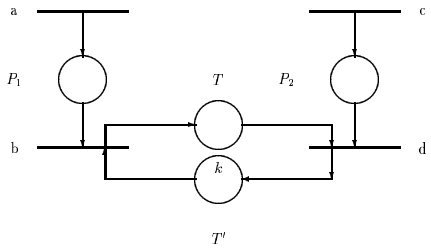
Réciproque fausse

$R =< P, T, Pre, Post >$ est Conservatif ssi
T est un invariant de Place

Si R est Conservatif alors R est borné

Réciproque fausse

Retour sur l'exemple du Producteur/Consommateur



T'

$$\left| F_{P_1} \;\; F_{P_2} \;\; F_T \;\; F_{T'} \right| \times \begin{vmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & -1 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

ssi $\left| F_{P_1} \;\; F_{P_2} \;\; F_T \;\; F_{T'} \right| = \lambda . \left| 0 \;\; 0 \;\; 1 \;\; 1 \right|$

Comme F_T & $F_{T'} \neq 0$ on a T et T' bornées

- Si $m_0 = \begin{vmatrix} 0 \\ 0 \\ 0 \\ k \end{vmatrix}$ Alors $F.m_0 = \lambda \times k$

$F.m = F.m_0 \Leftrightarrow \lambda \times (m(T) + m(T')) = \lambda \times k$

Finalement $m(T) \leq K$ & $m(T') \leq K$ $(\forall m \in G(R, m_0))$

9.4 Composantes Repétitives (Invariants de Transition)

Invariant de Transition

$S \subseteq T$ est un **invariant de Transition** *ssi*
$\exists s \in T^*$ telle que $||\sigma|| = S$ et $C.\bar{\sigma} = 0$

$R =< P, T, Pre, Post >$ est répétitif ssi
T est un invariant de transitions

Si R est vivant et borné est Alors R est répétitif

Problème: Lecteurs et écrivains en concurrence pour accéder à une (abstraction) de base de données. Modéliser un arbitre (contrôleur) permettant de synchroniser les lecteurs et les écrivains.

Conditions de synchronisation entre lecteurs et écrivains

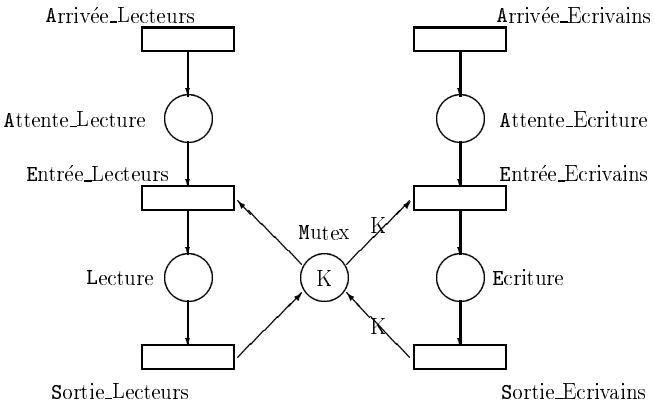
C_1 : Pas plus de k lectures simultanées
C_2 : Pas plus d'un écrivain dans la base
C_3 : Pas de lecture et d'écriture simultanément dans la base

Autre propriétés:

- Vivacité/"Absence de Famine": Toute requète (de lecture ou d'écriture) est satisfaite au bout d'un temps fini.
- Maintien de la cohérence des données (si une valeur est lue alors elle correspond à la dernière valeur écrite).

Modèle de la base de données

"générique" : entier positif k



Le système de base de données est constitué par:

- deux salles d'attente: l'une pour les lecteurs (Attente_Lecture) et l'autre pour les écrivains (Attente_Ecriture),
- une salle de lecture (Lecture) et d'une salle d'écriture (Ecriture).

"Composants du réseau"

- partie gauche du réseau décrit le traitement associé aux lecteurs
- partie droite concerne le traitement des écrivains.
- partie centrale (place Mutex) assure la synchronisation entre lecteurs et écrivains.

Modélisation générique:

- la capacité de la salle de lecture est représentée par un paramètre, l'entier positif k.
- k apparaît dans le marquage initial du réseau (place Mutex) et comme valuation des arcs reliant la place Mutex aux transitions Sortie_Lecteurs et Entrée_Lecteurs.

9.5.2 Vérification des propriétés de synchronisation

Reformuler les propriétés de synchronisation déja exprimées
en utilisant les "éléments de langage" fournis par le réseau proposé.

$\forall m \in A(R, m_0)$

C_1 : $m(Lecture) \leq k$
Pas plus de k lectures simultanées

C_2 : $m(Ecriture) \leq 1$
Pas plus d'un écrivain dans la base

C_3 : $m(Ecriture) \times m(Lecture) = 0$
Pas de lecture et d'écriture simultanément dans la base

Pour compacter les notations qui vont suivre, les transitions et les places ont été renommées.

p_1 : Lecture	t_1 : Arrive_Ecrivains
p_2 : Mutex	t_2 : Arrive_Lecteurs
p_3 : Ecriture	t_3 : Entre_Lecteurs
p_4 : Attente_Lecture	t_4 : Sortie_Lecteurs
p_5 : Attente_Ecriture	t_5 : Entre_Ecrivains
	t_6 : Sortie_Ecrivains

Matrice d'incidence associée au réseau

Comme le modèle est générique, cette matrice sera paramétrée par k.

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{vmatrix} 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -k & k \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \end{vmatrix} \end{array}$$

La recherche des P-semiflot conduit à résoudre le système suivant:

$$(v_1, v_2, v_3, v_4, v_5) \times \begin{vmatrix} 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & -k & k \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

Dans ce cas, les P-semiflot s'écrivent sous la forme $v = \lambda.(1, 1, k, 0, 0)$ pour tout entier naturel positif λ

On considère maintenant l'invariant linéaire suivant:

$$\forall m \in A(R, m_0), v^t.m = v^t.m_0$$

En remplaçant dans l'équation précédente m_0 et v par leurs valeurs, on obtient:

$$(1, 1, k, 0, 0).\begin{vmatrix} m(p_1) \\ m(p_2) \\ m(p_3) \\ m(p_4) \\ m(p_5) \end{vmatrix} = (1, 1, n, 0, 0).\begin{vmatrix} 0 \\ k \\ 0 \\ 0 \\ 0 \end{vmatrix} = k$$

On obtient donc: $\forall M \in G(R, m_0), m(p_1) + m(p_2) + k \times m(p_3) = n$

C-à-d $m(\text{Lecture}) + m(\text{Mutex}) + k \times m(\text{Ecriture}) = k$

En isolant $m(\text{Lecture})$ dans l'invariant, on obtient:

$$m(\text{Lecture}) = n - (m(\text{Mutex}) + n \times m(\text{Ecriture}))$$

Comme un marquage est une application à valeurs dans les entiers naturels, on en déduit

$$C_1 : m(\text{Lecture}) \leq k$$

.

$C_2 : m(\text{Ecriture}) \leq 1$ peut être obtenue de façon identique en isolant $m(\text{Ecriture})$ dans l'invariant et en se souvenant que k est strictement positif.

C_3 peut être obtenue en raisonnant par l'absurde.

En prenant la négation de C_3, on obtient

$$m(\text{Lecture}) > 0 \text{ et } m(\text{Ecriture}) > 0$$

On a donc

$$m(\text{Lecture}) + k \times m(\text{Ecriture}) \geq k + 1$$

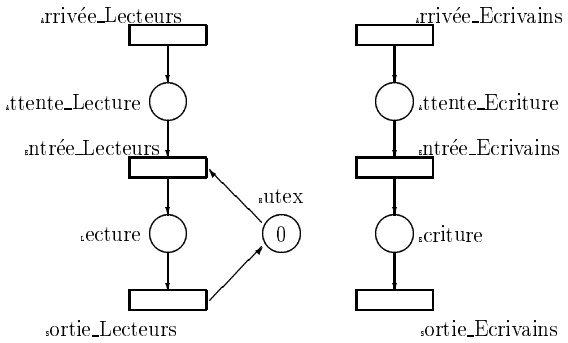
D'un autre coté, l'invariant nous assure que :

$$m(\text{Lecture}) + k \times m(\text{Ecriture}) \leq k$$

Comme $k > 0$, on aboutit ainsi à la contradiction et C_3 est validée.

Cas où $k = 0$

L'analyse structurelle ne permets pas d'assurer C_2 et C_3.



La place utex n'est plus une pré-condition de la transition ntrée_Lecteurs (l'arc correspondant disparaît car il est valué par 0)
et le nombre d'écritures simultanées devient non borné. La lecture devient impossible puisque le marquage initial de la place utex est nul.

Analyse paramétrée

Possibilité de pouvoir travailler avec l'analyse structurelle sur un réseau paramétré.

Les propriétés recherchées ont été établies non pas pour un réseau marqué mais pour toute une famille indexée par $k > 0$ de réseaux marqués.

Une approche exhaustive (construction du graphe d'accessibilité, arbre de couverture de Karp et Miller) n'aurait pu être conduite puisque nous aurions du auparavant fixer une valeur pour le paramètre k.

9.6.1 Vivacité/Sûreté

L'analyse structurelle est une approche séduisante pour la vérification des propriétés dites de "Sûreté"

"Rien de mauvais ne peut arriver"

L'analyse structurelle n'appporte pas de réponse pour la vérification des propriétés dites de "Vivacité"

"Quelque chose de bon doit arriver"

Un lecteur potentiel ne restera pas indéfiniment en attente dans la salle de lecture, ou en d'autres termes, que toute requête d'entrée dans la salle de lecture sera honorée au bout d'un temps fini.

Analogie avec le problème des Philosophes:

Attente infinie $\Rightarrow$ "Présence de Famine"

9.6.2 "Possibilité de Famine"

Lorsqu'un lecteur est en attente de lecture, $\mathcal{E}_1$ est possible:

$$\mathcal{E}_1 = (\texttt{A}rrive\_Ecrivains)^\omega$$

il est donc possible de faire entrer une infinité d'écrivains laissant le lecteur indéfiniment en attente

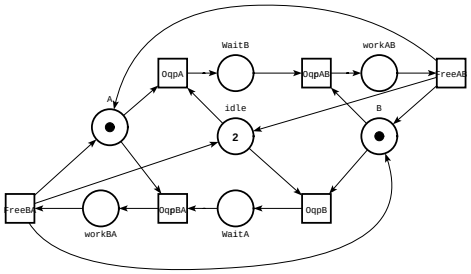
De même, la séquence infinie

$\mathcal{E}_2 = (\texttt{E}ntre\_Ecrivains.\texttt{S}ortie\_Ecrivains.\texttt{A}rrive\_Ecrivains)^\omega$

laisse de nouveau le lecteur indéfiniment en attente

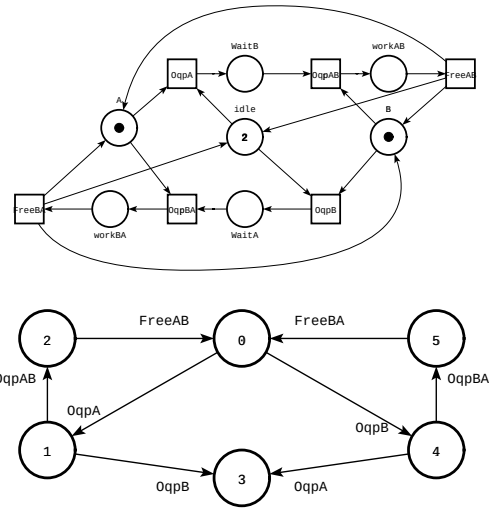
9.7 Analyse structurelle avec TINA

9.7.1 Invariants de Place (P-Semi-Flows)

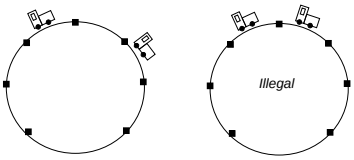


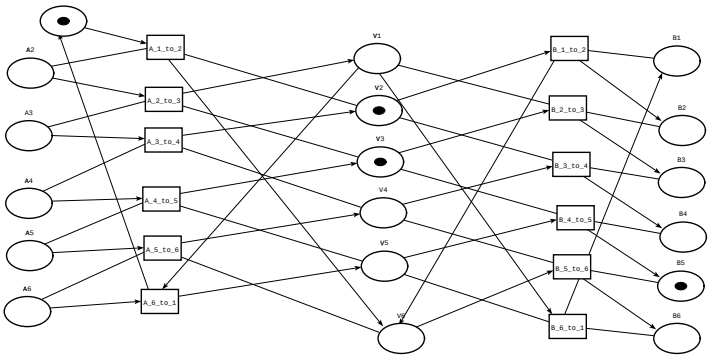

```
MARKINGS:

0 : A B idle*2
1 : B WaitB idle
2 : idle workAB
3 : WaitA WaitB
4 : A WaitA idle
5 : idle workBA

P-SEMI-FLOWS GENERATING SET -----------------

invariant
A WaitB workAB workBA
WaitA WaitB idle workAB workBA
B WaitA workAB workBA
```


9.7.2 Invariants de Transition (T-Semi-Flows)

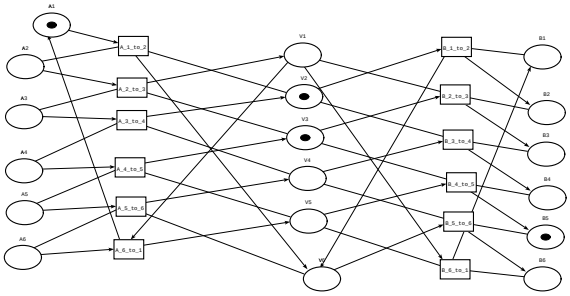



```
T-SEMI-FLOWS GENERATING SET ------------------

consistent

FreeAB OqpA OqpAB
FreeBA OqpB OqpBA
```

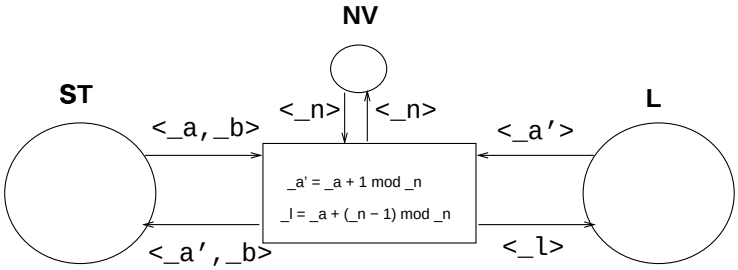




Limite du modèle de base / Limite du graphique
Intérêt/Nécessité d'un modèle plus expressif
Pour obtenir des descriptions plus compactes



Version prédicat/transition



10.1 Termes de Base

Données Initiales:

- $Const \leftarrow$ Ensemble de Constantes

- $Var \leftarrow$ Ensemble de Variables

- $\Pi \leftarrow$ Ensemble de Symboles de Prédicats
(Arite : $\Pi \mapsto IN^+$)

$\mapsto L(Const, Var, \Pi)$ Ensemble des prédicats

$=_{def}$ plus petit ensemble vérifiant 1) 2) et 3)

1) $Const \subset L(Const, Var, \Pi)$

2) $Var \subset L(Const, Var, \Pi)$

3) Pour chaque $\pi \in \Pi$ tel que $arite(\pi) = k$

et $t_1, t_2, \ldots t_k \in L(Const, Var, \Pi)$

Alors $\pi(t_1, t_2, \ldots t_k) \in L(Const, Var, \Pi)$

Exemple du Train:

$Const = \{train\_a, train\_b, 0, 1, \ldots 6\}$

$Var = \{$ _a,_b, ... _z , _a', _b', ... _z' $\}$

$\Pi = \{ST, NV, L\}$

Arité	Instance	"Sémantique"
ST/2 :	ST(0,train_a)	"le train a est en 0"
NV/1 :	NV(7)	"la section comporte 7 voies"
L/1 :	L(2) :	"la portion 2 est libre "

$$\text{Termes} = \begin{cases} Const \cup Var \\ \cup \{NV(x) : x \in Const \cup Var\} \\ \cup \{L(x) : x \in Const \cup Var\} \\ \cup \{ST(x,y) : (x,y) \in (Const \cup Var)^2\} \\ \cup \ldots\ldots infini \end{cases}$$

Soit σ une application de $Var \mapsto L(Const, Var, \Pi)$

σ peut-être canoniquement étendue en une substitution $\bar{\sigma}$

$\bar{\sigma} : L(Const, Var, \Pi) \mapsto L(Const, Var, \Pi)$

de la façon suivante:

$$\begin{array}{ll} \bar{\sigma}(c) = c & \text{pour } c \in Const \\ \bar{\sigma}(v) = \sigma(v) & \text{pour } v \in Var \\ \bar{\sigma}(\pi(t_1, t_2 \ldots t_k)) = \pi(\bar{\sigma}(t_1), \bar{\sigma}(t_2), \ldots \bar{\sigma}(t_k)) & \text{sinon} \end{array}$$

Extension aux ensembles de Termes

Pour $E \subset L(Const, Var, \Pi)$ et σ une substitution:

$\bar{\sigma}(E) =_{def} \{\bar{\sigma}(e) : e \in E\}$

$\bar{\sigma}$ préserve le cardinal de E ssi $Card(\bar{\sigma}(E)) = Card(E)$

Terme Clos

Un terme clos est un terme sans aucune variable
(invariant pour toute substitution)

On note $L_G(Const, Var, \Pi)$,
l'ensemble des termes **clos** de $L(Const, Var, \Pi)$

$M = \{ST(\_x, \_y), ST(\_z, \_t), NV(\_V), L(\_a), L(\_b), L(\_c)\}$

$\sigma =_{def} (\_x/0, \_y/a, \_z/2, \_t/b, \_a/3, \_b/4, \_c/5, \_v/7\}$

$\bar{\sigma}(M) = \{ST(0, a), ST(2, b), NV(7), L(3), L(4), L(5)\}$

nb: $Card(\bar{\sigma}(M)) = Card(M)$

$\sigma' =_{def} (\_x/0, \_y/a, \_z/2, \_t/b, \_a/3, \_b/5, \_c/5, \_v/5)$

$Card(\bar{\sigma'}(M)) < Card(M)$

10.3 Marquage d'un Réseau de Prédicats

Ensemble fini de Prédicats Clos

$M = \mathcal{P}_{\mathcal{F}}(L_G(Const, Var, \Pi))$

$\mathcal{P}_{\mathcal{F}}(E)$ ensembles des parties finies de E

10.4 Transition d'un Réseau de Prédicats

Quadruplet $< Label, Pre, Post, Cond >$ où

$Pre, Post \in \mathcal{P}_{\mathcal{F}}(L_G(Const, Var, \Pi) \setminus Var)$

$Label \in L(Const, Var, \Pi)$

$Cond \in L_{fonct}(Const, Var, Fonct)$
où $Fonct$ un ensemble de symboles fonctionnels .....

Exemple

$$\begin{array}{ll} < \ depart\_train(\_a, \_b), & \text{“Label”} \\ \quad \{ST(\_a, \_b), L(\_a'), NV(\_n)\} & \text{“Pre-conditions”} \\ \quad \{ST(\_a', \_b), L(\_l), NV(\_n)\} & \text{“Post-conditions”} \\ \quad [\_a' = \_a + 1 \ \ mod\_n, & \text{“Conditions} \\ \quad\ \ \_l = \_a + (\_n - 1) \ \ mod\_n] & \text{sur les variables”} \\ > & \end{array}$$

10.5 Réseaux de Prédicats

construit sur $< Const, Var, \Pi, Fonct >$

Ensemble fini de Transitions

$$\begin{array}{l} \mathcal{P}_{\mathcal{F}}(\\ \qquad L(Const, Var, \Pi) \\ \quad \times \ \mathcal{P}_{\mathcal{F}}(L(Const, Var, \Pi) \setminus Var) \\ \quad \times \ \mathcal{P}_{\mathcal{F}}(L(Const, Var, \Pi) \setminus Var) \\ \quad \times \ L_{fonct}(Const, Var, Fonct) \\ ) \end{array}$$

Marquage: $\mathcal{P}_{\mathcal{F}}(L_G(Const, Var, \Pi))$

$< R, M >$ un réseau Pr/Tr marqué,
t une transition de R $\quad t =< label, pre, post, cond >$
et σ une application de $Var \mapsto L(Const, Var, \Pi)$

$M[\bar{\sigma}(t) >$ "M sensibilise l'instance de t définie par σ" ssi

1. $\bar{\sigma}(pre) \subset M$ & $Card(\bar{\sigma}(pre)) = Card(pre)$
2. $\bar{\sigma}(cond)$ est "vraie"
3. $Card(\bar{\sigma}(post)) = Card(post)$
4. $[\bar{\sigma}(post) \setminus \bar{\sigma}(pre)] \cap M = \emptyset$
   où $A \setminus B =_{def} \{x \in A \text{ et } x \notin B\}$

$M_0 = \{ST(0,a), ST(2,b), NV(7), L(3), L(4), L(5)\}$

$$< \ depart\_train(\_a, \_b), \{ST(\_a, \_b), L(\_a'), NV(\_n)\} \{ST(\_a', \_b), L(\_l), NV(\_n)\} [\_a' = \_a + 1 \mod \_n, \ \_l = \_a + (\_n - 1) \mod \_n] >$$

$\sigma =_{def} (\_a/2, \_b/b, \_a'/3, \_n/7, \_l/1\}$

Pb: A-t-on $M_0[\bar{\sigma}(t) >$

$$\begin{array}{ll} & \bar{\sigma}(pre) = \bar{\sigma}(\{ST(\_a, \_b), L(\_a'), NV(\_n)\}) \\ (1) & = \{\bar{\sigma}(ST(\_a, \_b)), \bar{\sigma}(L(\_a')), \bar{\sigma}(NV(\_n))\} \\ & = \{ST(2,b), L(3), NV(7)\} \end{array}$$
$Card(\bar{\sigma}(pre)) = Card(pre)$

$$\begin{array}{ll} & \bar{\sigma}(cond) = \bar{\sigma}([\_a' = \_a + 1 \mod \_n \\ & \quad et \ \ \_l = \_a + (\_n - 1) \mod \_n]) \\ (2) & = [(3 = 2 + 1 \mod 7) \\ & \quad et \ \ (1 = 2 + (7 - 1) \mod 7)] \end{array}$$
$\bar{\sigma}(cond)$ est vraie

$$\begin{array}{ll} & \bar{\sigma}(post) = \bar{\sigma}(\{ST(\_a', \_b), L(\_l), NV(\_n)\}) \\ (3) & = \{ST(3,b), L(1), NV(7)\} \end{array}$$
$Card(\bar{\sigma}(post)) = Card(post)$

$$\begin{array}{ll} & \bar{\sigma}(post) \setminus \bar{\sigma}(pre) = [\{ST(3,b), L(1), NV(7)\} \\ (4) & \qquad \setminus \{ST(2,b), L(3), NV(7)\}] \\ & = \{ST(3,b), L(1)\} \end{array}$$

$$\begin{array}{ll} [\bar{\sigma}(post) \setminus \bar{\sigma}(pre)] \cap M_0 & = \{ST(3,b), L(1)\} \\ & \quad \cap \{ST(0,a), ST(2,b), \\ & \qquad NV(7), L(3), L(4), L(5)\} \\ & = \emptyset \end{array}$$

(1),(2),(3),(4) $\Rightarrow M_0[\bar{\sigma}(t) >$

10.7 Tir d'un transition

Si $M[\bar{\sigma}(t) >$ alors $\exists M' : M[\bar{\sigma}(t) > M'$

$M' = (M \setminus \bar{\sigma}(pre)) \cup \bar{\sigma}(post)$

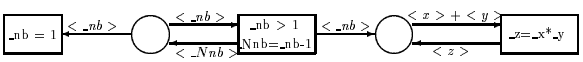
Exemple

On a $M_0[\bar{\sigma}(t) >$

Avec $\begin{array}{ll} M' & = \{ST(3,b), L(1), NV(7) \\ & \quad ST(0,a), L(4), L(5)\} \end{array}$

10.8 Exercice

Γ-Calcul



10.9 Quelques exemples

```
INITIALIZE
TO idle([p1,p2,p3]), rec(a,1), rec(b,1),rec(c,1)
END;

TRANS   take(_x,_r)
FROM    idle([_x|_tail]),rec(_r,_n)
TO      idle(_tail),rec(_r,_nn),oqp(_x,_r)
PROVIDED positif(_n),minus(_n,_nn)
END;

TRANS   take(_x,_q)
FROM    oqp(_x,_r),rec(_q,_n)
TO      work(_x,_r,_q),rec(_q,_nn)
PROVIDED positif(_n),minus(_n,_nn),other(_r,_q)
END;

TRANS   release(_x,_r,_q)
FROM    work(_x,_r,_q),idle(_list),rec(_q,_xq),rec(_r,_xr)
TO      idle([_x|_list]),rec(_q,_nxq),rec(_r,_nxr)
PROVIDED plus(_xr,_nxr),plus(_xq,_nxq)
END;

PROLOG-CODE ;
other(_a,_b) :- \+ _a == _b .
minus(_x,_nx) :- _nx is _x - 1 .
plus(_x,_nx) :- _nx is _x + 1 .
positif(_x) :- _x > 0 .
```



```
PrT MODEL;
INITIALIZE
TO nb(5),
   oisif(0),   oisif(1),   oisif(2),   oisif(3),   oisif(4),
   fork(0),    fork(1),    fork(2),    fork(3),    fork(4)
END;

TRANS demande(_ident)
FROM oisif(_ident)
TO   attente(_ident)
END;

TRANS mange(_ident)
FROM attente(_ident),nb(_nb),fork(_left),fork(_right)
PROVIDED voisins(_ident,_nb,_left,_right)
TO   mange(_ident),nb(_nb)
END;

TRANS pense(_ident)
FROM attente(_ident),nb(_nb)
PROVIDED voisins(_ident,_nb,_left,_right)
TO   mange(_ident),fork(_left),fork(_right),nb(_nb)
END;

PROLOG-CODE;
voisins(_ident,_nb,_ident,_right) :-
_right is ((_ident + 1 ) mod _nb) .
```

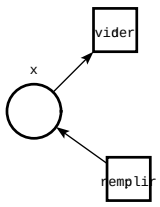

10.10 Réseau Place/Transition et Prédicat/Transition

Propriété: Tout réseau place/transition peut se traduire en réseau Prédicat/Transition.

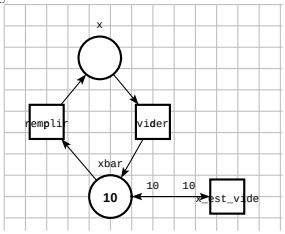
Réciproque fausse: Un réseau Prédicat/Transition permet de représenter le "test à zéro" (i.e arcs inhibiteurs)

```
PrT MODEL;

INITIALIZE
TO x(0)
END;

TRANS remplir_x
FROM     x(_x)
TO       x(_nx)
PROVIDED
     _nx is (_x + 1)
END;

TRANS vider_x
FROM     x(_x)
TO       x(_nx)
PROVIDED _x > 0,
     _nx is (_x - 1)
END;

TRANS  x_est_vide
FROM     x(0)
TO       x(0)
END;
```

Pas de solution dans le cas général

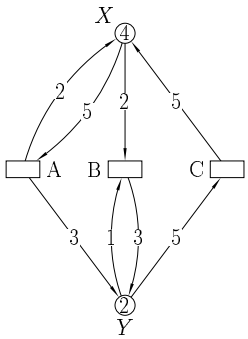


Si on connaît la borne de la place alors on peut utiliser d'une place complémentaire



10.10.1 Principe de la traduction

```
PrT MODEL;    (* Première Version *)

INITIALIZE
TO x(4), y(2)
END;

TRANS a
FROM     x(_x), y(_y)
TO       x(_nx), y(_ny)
PROVIDED _x >= 5,
         _nx is ( _x - 5 ) + 2,
         _ny is _y + 3
END;

TRANS    b
FROM     x(_x) , y(_y)
TO       x(_nx), y(_ny)
PROVIDED _x >= 2, _y >= 1,
         _nx is _x - 2 ,
         _ny is ( _y - 1 ) + 3
END;

TRANS c
FROM     x(_x) , y(_y)
TO       x(_nx), y(_ny)
PROVIDED _y >= 5,  _nx is _x + 5 ,
         _ny is _y - 5
END;
```



```
PrT MODEL;

INITIALIZE
TO x(4), y(2)
END;


TRANS     a
FROM      x(_x), y(_y)
TO        x(_nx), y(_ny)
PROVIDED  a_tirable(_x,_y),
          tir_a(_x,_y,_nx,_ny)
END;

TRANS     b
FROM      x(_x) , y(_y)
TO        x(_nx), y(_ny)
PROVIDED  b_tirable(_x,_y),
          tir_b(_x,_y,_nx,_ny)
END;

TRANS     c
FROM      x(_x) , y(_y)
TO        x(_nx), y(_ny)
PROVIDED  c_tirable(_x,_y),
          tir_c(_x,_y,_nx,_ny)
END;
```

```
PROLOG-CODE;

a_tirable(_x,_y) :-
           _x >= 5 .
b_tirable(_x,_y) :-
          _x >= 2,_y >= 1 .
c_tirable(_x,_y) :-
          _y >= 5 .


tir_a(_x,_y,_nx,_ny) :-
           _nx is (_x - 5)
           _ny is _y + 3 .

tir_b(_x,_y,_nx,_ny) :-
           _nx is  _x - 2,
           _ny is (_y - 1)

tir_c(_x,_y,_nx,_ny) :-
           _nx is  _x + 5 ,
          _ny is _y - 5  .
```



```
PrT MODEL;
INITIALIZE
TO x(4), y(2)
END;

TRANS    _nom
FROM     x(_x), y(_y)
TO       x(_nx), y(_ny)
PROVIDED est_tirable(_nom,_x,_y),
         tir_de(_nom,_x,_y,_nx,_ny)
END;

PROLOG-CODE;
est_tirable(a,_x,_y) :- _x >= 5 .
est_tirable(b,_x,_y) :- _x >= 2 , _y >= 1 .
est_tirable(c,_x,_y) :- _y >= 5 .


tir_de(a,_x,_y,_nx,_ny) :- _nx is (_x - 5) + 2 , _ny is _y + 3 .
tir_de(b,_x,_y,_nx,_ny) :- _nx is _x - 2 , _ny is (_y - 1 ) + 3 .
tir_de(c,_x,_y,_nx,_ny) :- _nx is _x + 5 ,  _ny is _y - 5 .
```

Propriété Tout réseau place-transition peut se ramener à un réseau prédicat-transition possédant une seule transition.

Contents