INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Modèles temporels

Novembre 2017

François Vernadat
INSA-DGEI - LAAS-CNRS

vernadat@laas.fr

Université de Toulouse

Outline

Introduction

Systèmes Temporisés

Ensembles de Composants interagissent avec un environnement dans le but d'assurer un "service" satisfaisant des contraintes temporelles "dures".
Domaines : Systèmes de commande dans l'avionique, le nucléaire, le transport, les télécommunications, etc

Systèmes Critiques

Compte-tenu du domaine d'application, les " défaillances " (la non satisfaction du service attendu) peut avoir des coûts (humains ou économiques) prohibitifs ....

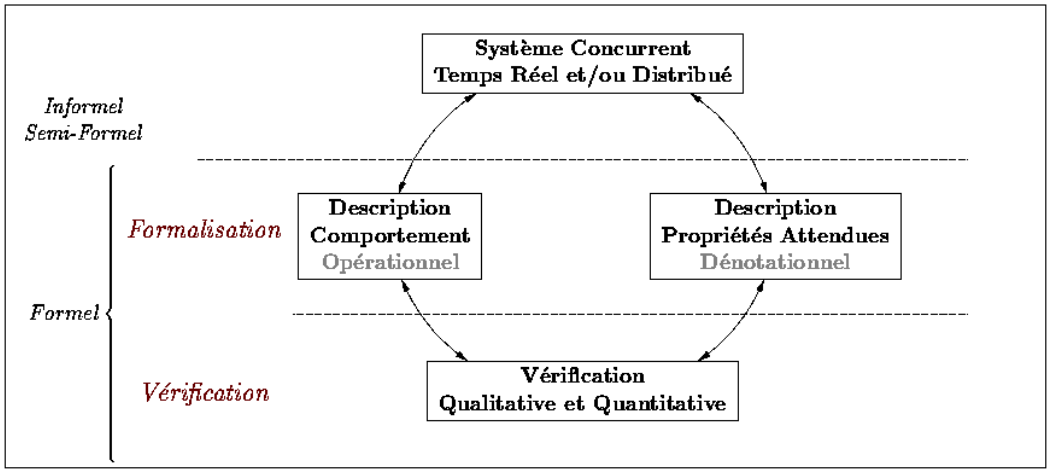
Criticité $\mapsto$ Formalisation/Vérification

Criticité $\mapsto$ Développer des logiciels "fiables"
Etre capable de "démontrer" leur "fiabilité"

Source importante de défaillance liée à des erreurs de conception

$\mapsto$ Méthodes Formelles (au plus tôt) dans le processus de développement

Modélisation Formelle



Description Opérationnelle

Description précise et sans ambiguïté de comment fonctionne opérationnellement le système

Propriétés attendues du Système

Spécification, Cahier des Charges formalisé, " Service " ....
Description précise et sans ambiguïté de ce que doit faire le système

Vérification

Permet de "mesurer " l'adéquation entre le service fourni / service attendu.
Le système conçu satisfait-il ses spécifications ?

Formalisation/Vérification

Ingrédients

De façon générale, toute méthode formelle nécessite des "langages dédiés" munis d'une sémantique précise, ainsi que des outils d'aide à la conception, la spécification et la vérification.
Dès la fin des années 80 : Techniques de Description Formelles (TDF)
Protocoles de Télécommunications, Circuits, Contrôle/Commande,

Modèles formels

Comportemental (description opérationnelle)

- **Modèles :** Réseaux de Petri, Automates Communicants, Algèbres de Processus
- **Langages normalisés :** Estelle, Lotos, SDL ....

Spécification (description des propriétés)

- **Modèles :** Logiques Temporelles, Eq. de Comportement, MSC ...

Mise en oeuvre

Algorithmes (efficaces) de Vérification
Outils supports (universitaires ou industriels)

Formalisation/Vérification Atemporelle

Existant Atemporel

Pas de prise en compte explicite du temps	Description Opérationnelle Spécifications des propriétés

Comportement Atemporel

- événements ne prennent pas de temps,
- pas de notion d'urgence, de latence
- (tous les événements sont équiprobables) etc

Spécifications Qualitatives : *Classification : Safety/Liveness*

Sûreté : *Rien de "mauvais" ne peut arriver*
Ex : Deux processus en même temps en section critique

Vivacité : *Quelque chose de "bon" doit arriver*
Ex : Un processus qui a envoyé une requête recevra inévitablement/potentiellement une réponse

Pour les exprimer $\mapsto$ Logiques **temporelles** caractérisant l'évolution temporelle d'un système mais avec un *Temps Logique*

De l'atemporel vers le temporel ($\equiv$ 90)

Développement du Critique : Aéronautique, Automobile, ...

Complexité accrue : Systèmes Temporisés encore + difficiles à Maîtriser
Concurrence (communication/asynchronisme/parallélisme/etc)
+ Contraintes temporelles

Besoin fort	Décrire des systèmes contraints temporellement Assurer leur "fiabilité "

Atouts

- Certains Modèles formels existaient déjà
  (Rdp Temporels/Temporisés ($>$ 74))
- Extensions des autres modèles possibles
  - Opérationnels : Alg de Proc, Automates Communicants
  - Spécifications : Logiques Temporelles "Temporisées "

Défis : **Analyse** - Beaucoup de pbs inécidables

- Descriptions Temporelles $\mapsto$ Espaces d'états infinis
$\mapsto$ Nécessité de trouver des représentations (finies)
symboliques qui préservent les possibilités d'analyse
(par ex : Réseaux temporels (74) .... Analyse (84))

Panorama du cours

Introduction à la modélisation des systèmes temporels

Comportement

Extensions des Réseaux de Petri & analyse

Extensions des Automates & analyse

Spécification (survol)

Logiques Temporelles / Temporisées

Formel/Pratique

- Réseaux de Petri Temporels (Tina)
- Automates temporisés (Kronos & Uppaal)
- Modélisation et Analyse

Organisation

Cours (10HCM) & TP (x3) (Tina & Uppaal)

Evaluation Exam écrit - Documents autorisés

Outline

Domaines Extensions Temporelles

Modèle RdP Initial Atemporel

$\mapsto$ Plusieurs extensions proposées pour prendre en compte le temps

Protocoles

Performances

Ordonnancement (temps-réel)

Introduction du temps : Certes mais quoi ? et où ?

Quoi Durées ? Délais ? Dates de tir ?

où Places ? Transitions ? Jetons ? Arcs ?

Début de l'histoire en 1974 .....

Etiquettes Temporelles

Etiquettes sur les Transitions :

t	durée associée à l'exécution d'une transition	
(ds, dt)	ds	Date de tir au plus tôt
	dt	Durée associée à l'exécution d'une transition
$[t_{min}, t_{max}]$	t_{min}	Date de tir au plus tôt
	t_{max}	Date de tir au plus tard

Etiquettes sur les Places :

t	Durée d'indisponibilité d'un jeton arrivant dans une place

Etiquettes sur les Arcs :

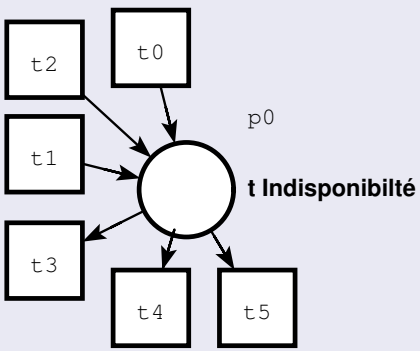
$[t_{min}, t_{max}]$	t_{min}	Date de tir au plus tôt
	t_{max}	Date de tir au plus tard

Quelques extensions

"Temporisé Transition" - Ramchandani (M.I.T) 1974

$\mapsto$ *Le tir d'une transition n'est plus instantané (de durée nulle) mais nécessite un certain temps t.*

"Temporisé Place" Sifakis (CNRS) 1977

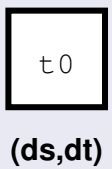


$\mapsto$ *t indique maintenant le temps pendant lequel un jeton déposé dans une place reste indisponible*

nb : Etiquette est associée à la place, mais l'information temporelle est associée au jeton.

Quelques extensions (suite)

"Temporisé Transition" - Razouk (U.C.L.A) 1983



$\mapsto$ (ds, dt)

- ds Date de tir au plus tôt
- dt Durée associée à l'exécution d'une transition

$\mapsto$
- Une transition doit rester sensibilisée (au moins) pendant un temps ds avant de pouvoir être tirée.
- Le tir commence et se poursuit durant dt.

"Temporels Transition" - Merlin (U.C.L.A) 1974

$\mapsto$ $[t_{min}, t_{max}]$

- t_{min} Date de tir au plus tôt
- t_{max} Date de tir au plus tard

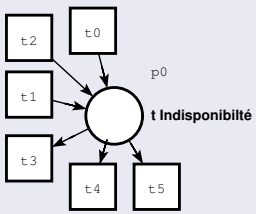
$\mapsto$
- Une transition doit rester sensibilisée durant t_{min} avant de pouvoir être tirée.
- Elle **ne peut** rester sensibilisée au dela de t_{max}

nb : Le tir d'une transition est de durée nulle !

Correspondances avec les Réseaux Place/Transitions

Durée

Ramchandani
$t = 0$
Durée de tir nulle



Sifakis
$t = 0$
Temps nul d'indispnonibilité du jeton

(ds,dt)

Razouk
$ds = 0$ durée mini de sensibilisation nulle
$dt = 0$ durée de tir nulle

[Tmin,Tmax]

Merlin
$t_{min} = 0$ durée mini de sensibilisation nulle
$t_{max} = \infty$ pas de contrainte sur
la date de tir au plus tard

Outline

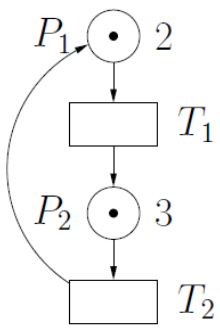
Réseaux **P**-Temporisés

Première extension permettant à un RdP de décrire
un système dont le fonctionnement dépend du temps.

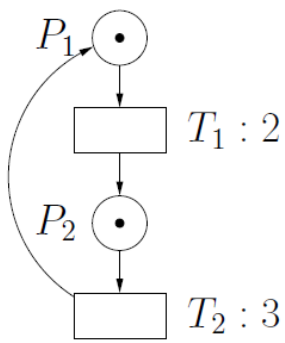
2 approches (équivalentes) suivies :

- L'exécution d'une transition n'est plus instantannée mais est caractérisée par une durée. $\mapsto$ Rdp T-Temporisés
- Les jetons sont munis d'un statut d'indisponibilité. Une transition produit des jetons qui ne seront pas immédiatement utilisables. La durée d'indisponibilité est définie par les places du réseau.
  $\mapsto$ Rdp P-Temporisés.

P-Temporisé



T-Temporisé



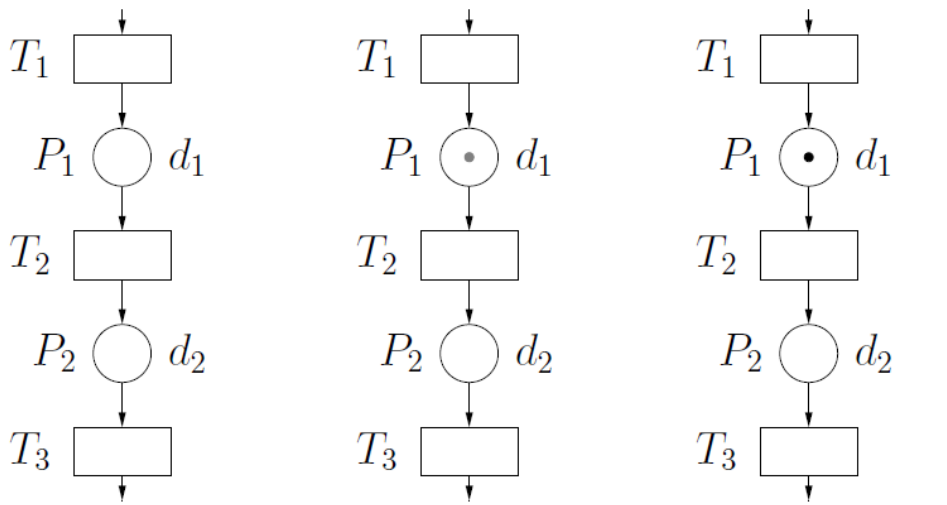
Réseaux **P**-Temporisés

Définitions : Réseau P-Temporisé

Rdp-PT est un couple $< R, \Theta >$ où :

$R = \ < P, T, Pre, Post >$ est un RdP place/transition (classique)

Θ : Application qui associe à chaque place un temps d'indisponibilté ($\Theta : P \mapsto R$)



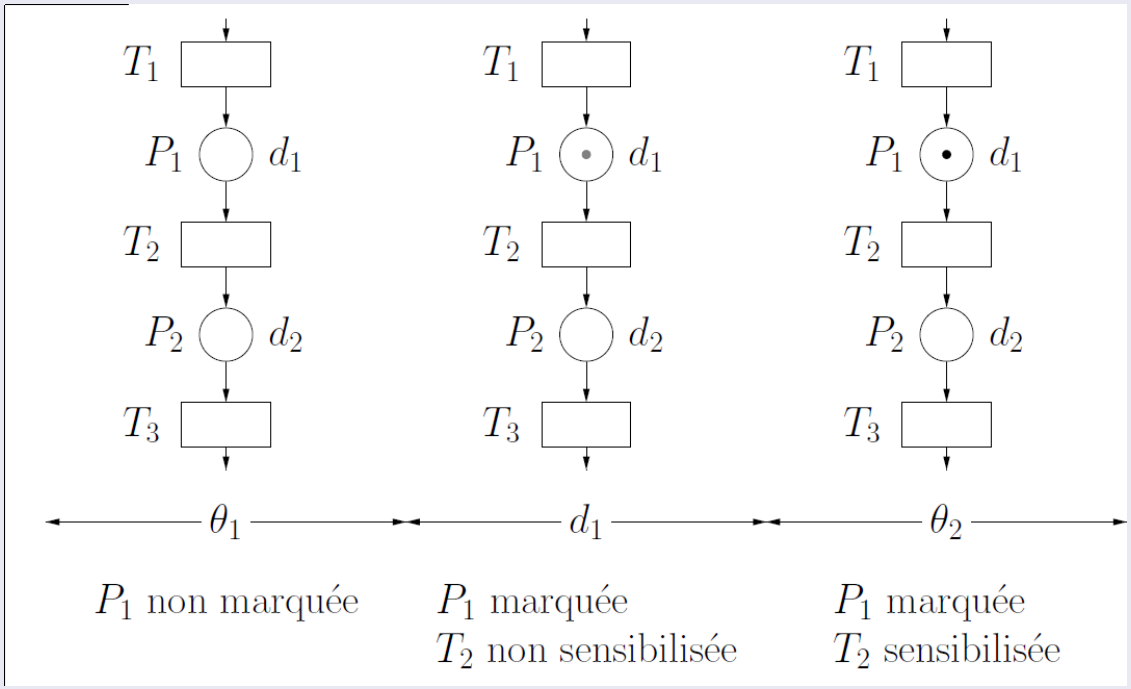
P_1 non marquée

P_1 marquée
Jeton indisponible
T_2 non sensibilisée

P_1 marquée
T_2 sensibilisée

Réseaux **P**-Temporisés : Sémantique inutive

Principe général de fonctionnement



d_1, d_2 déterminés par le réseau

$\theta_1, \theta_2, \theta_3$ délais arbitraires $\in [0, \infty[$

entre la sensibilisation d'une transition et son franchissement.

Principe général de fonctionnement

Lorsque $\theta_1 = \theta_2 = \theta_3 = 0 \mapsto$ Fonctionnement à "vitesse maximale" $\equiv$

dés qu'une transition est sensibilisée, on la franchit.

Réseaux **P**-Temporisés : Marquages temporisés

Marquage temporisé : (M, θ) où

M est un marquage et θ une Application qui associe à chaque place du réseau le nombre de jetons qu'elle contient et pour chacun de ceux-ci le temps d'indisponibilité qui lui est associé.

On note $\mathcal{M}_\theta$ l'ensemble de tous les marquages temporisés.

Représentation choisie pour un marquage temporisé

p_1	0
p_2	0

P_1 et P_2 marquées sont chacunes marquées avec un jeton immédiatement disponible

p_1	$\emptyset$
p_2	0.2

P_1 est non marquée et
P_2 est marquée par 2 jetons,
l'un immédiatement disponible,
l'autre ayant une indisponibilité de 2

Décalage temporel $\ominus$: Opération : $R \times R^* \mapsto R^*$

$$\ominus \text{ est définie par } \begin{cases} \emptyset \ominus t = \emptyset \\ \delta.\sigma \ominus t = Max(0, \delta - t).(\sigma \ominus t) \end{cases}$$

$0.2.5 \ominus 1 = 0.1.4, 0.2.5 \ominus 8 = 0.0.0$ etc etc

Réseaux **P**-Temporisés : Sémantique

Sensibilisation/Tir d'une transition à un instant θ donné

Sensibilisation

Analogue à celle des réseaux atemporels en se restreignant aux jetons disponibles

Tir d'une transition à un instant θ donné

3 différences wrt tir classique

1. Eeffectuer un décalage temporel de θ
2. Pour les places en pre-condition, retirer autant de jetons (disponibles) que ceux définis par Pre(p,t)
3. Pour les places en post-condition, ajouter autant de jetons que ceux définis par Post(p,t) affectés par l'indisponibilité définie par $\Theta(p)$ Sensibilisation

PB : ∞ instants de sensibilisation/tir

Comportement avec une infinité d'états et de transitions
$\mapsto$ Abstraction

Fonctionnement à vitesse Maximale

Vitesse Maximale

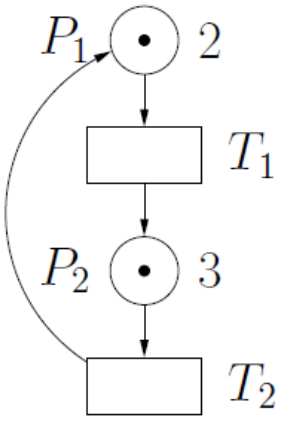
Objectif

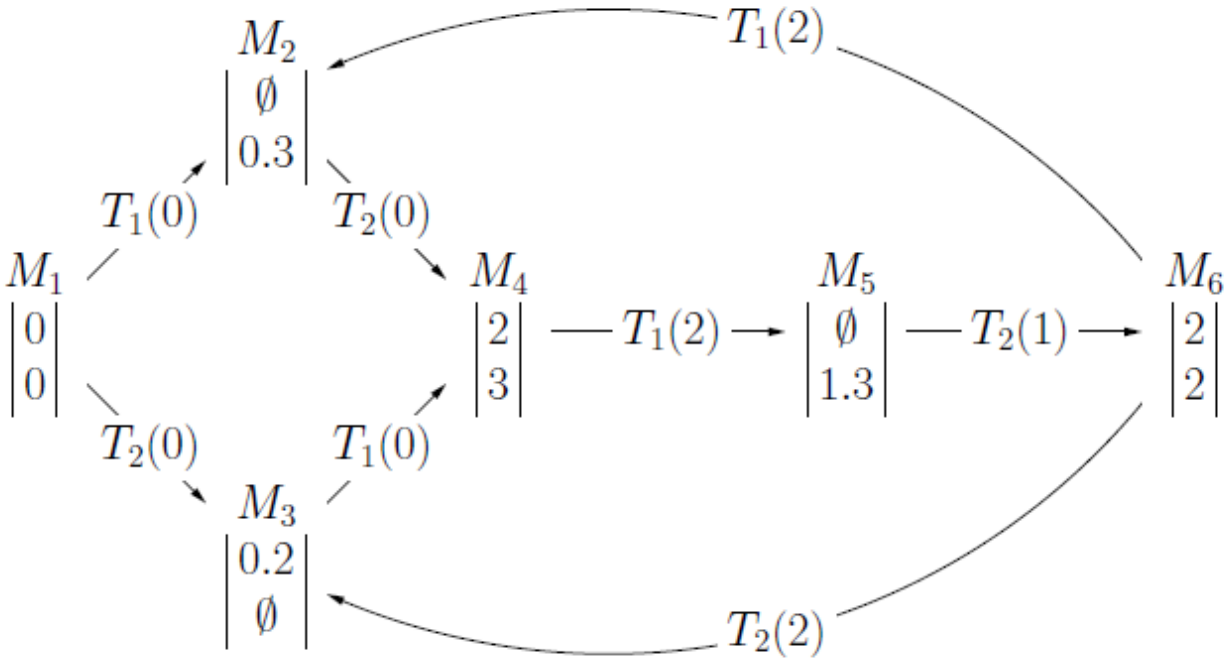
Abstraction pour pallier l'**infinité d'instants de sensibilisation/tir**

Principe

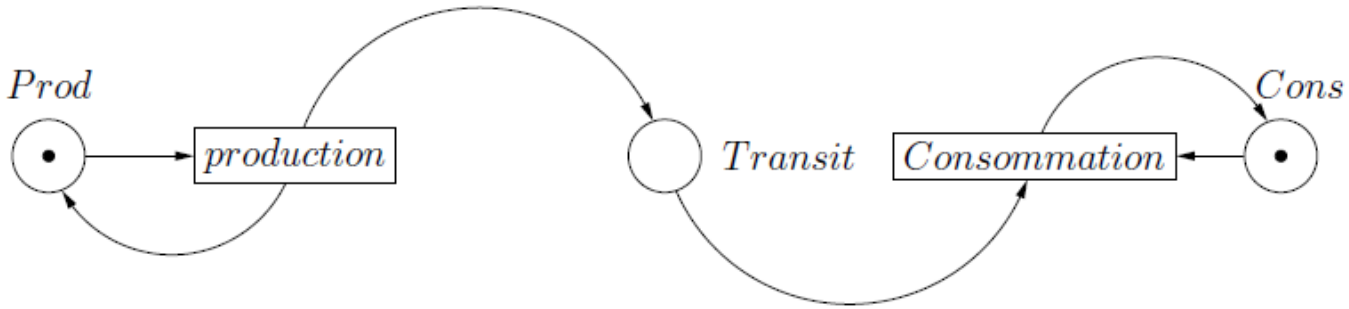
- Déterminer le plus petit instant de tir
- Explorer **uniquement** les transition sensibilisées **à cet instant**

Exemple





Exemple du Producteur/Consomateur

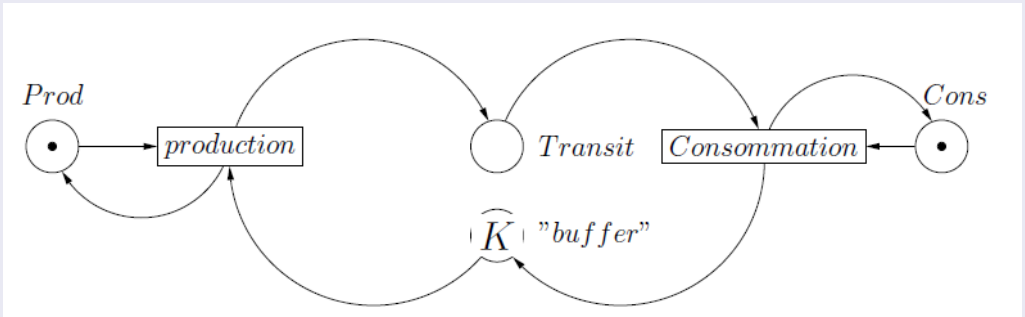


Arbre de Couverture

$$\left|\begin{array}{c}1\\0\\1\end{array}\right| \quad \textit{Production} \longrightarrow \quad \left|\begin{array}{c}1\\1\omega\\1\end{array}\right|$$

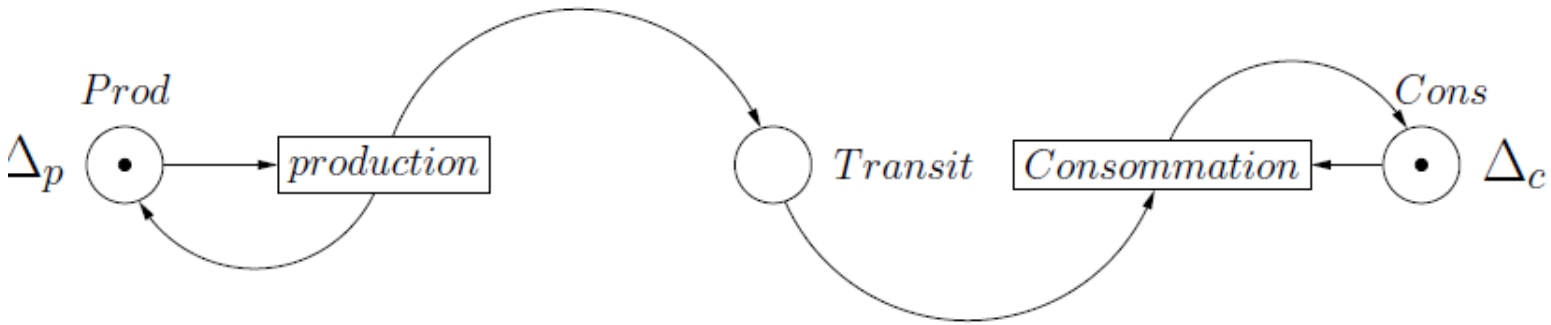
Production est une séquence répétitive croissante pour la place *transit* donc celle-ci est non-bornée.

Contrôle de Flux Explicite



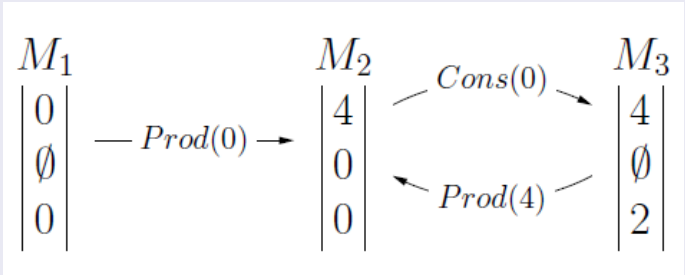
Analyse structurelle $\Rightarrow$ Transit est bornée par K
Contrôle de Flux nécessite des messages en $+$ $\Rightarrow$ Coût

Producteur/Consomateur temporisé

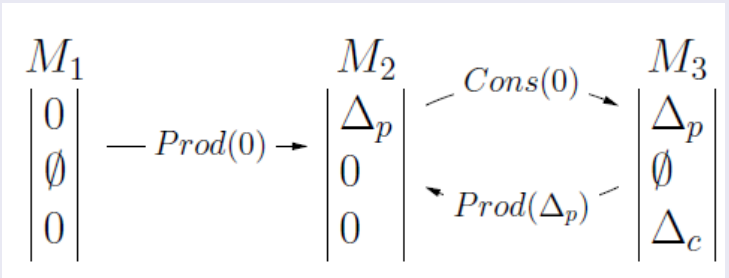


Différentes valeurs de temprosiations

Cas où
$\Delta_p = 4$
et $\Delta_c = 2$



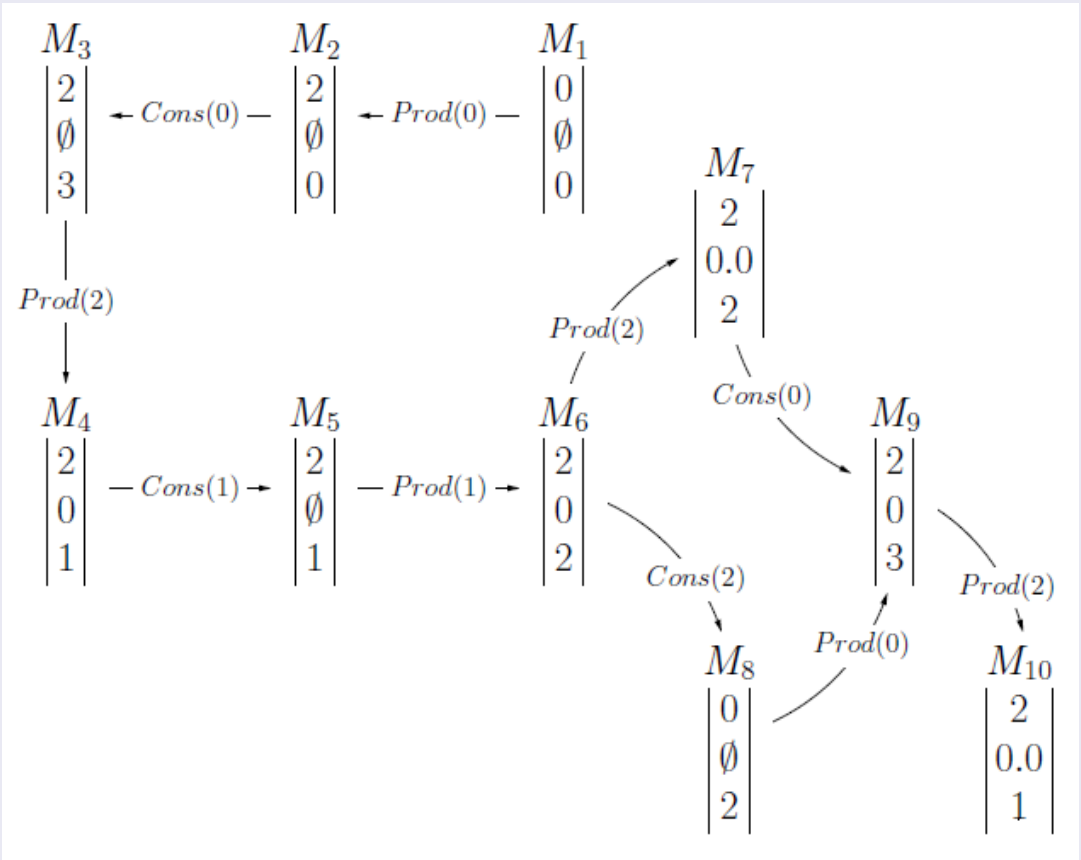
Cas où
$\Delta_p \geq \Delta_c$



Producteur/Consomateur temporisé

Cas où $\Delta_p < \Delta_c$

$\Delta_p = 2$ & $\Delta_c = 3$



Producteur/Consomateur temporisé : épilogue

Si $\Delta_p < \Delta_c$ Alors Transit est non bornée

"Analogie" avec les Rdp Place/Transitions :

$\exists$ Séquences Repétitives croissantes pour la place *transit*

(Intuitivement) Soient M_3 et M_9

$*$ atemporel : $M_3 \leq M_9$ (i.e $\begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} \leq \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}$)

$*$ temporel : M_3 et M_9 présentent les mêmes caractéristiques.

$\rightarrow$ M_3 et M_9 présentent les mêmes séquences

Plus Formellement

$\forall w \in \{0\}^* : M_w =_{Def} \begin{vmatrix} 2 \\ w \\ 3 \end{vmatrix}$

Soit $\sigma = Prod(2).Cons(1).Prod(2).Cons(0)$

On a $\forall w \in \{0\}^* : M_w[\sigma > \text{ et } M_w[\sigma > M_{w.0}$ $\quad (M_{w.0} = \begin{vmatrix} 2 \\ w.0 \\ 3 \end{vmatrix})$

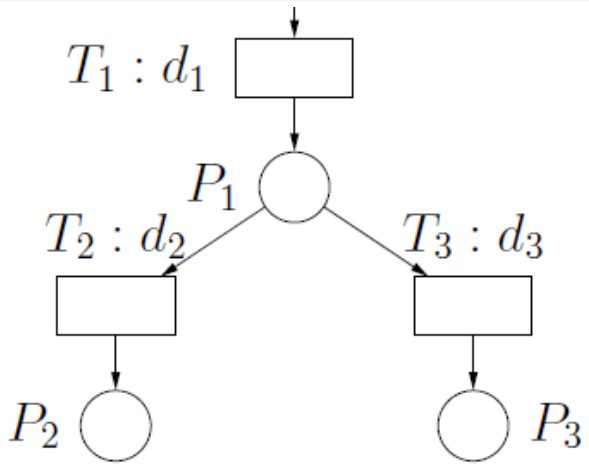
On a donc : $M_w[\sigma^\omega >$ et *transit* est non bornée

NB : Idem pour M_4 et M_{10}

Réseaux T-Temporisés (Ramchandani)

T-Temporisés

$\mapsto$ Etiquettes Temporelles sur les transitions
- Jetons immédiatement disponibles
- L'exécution d'une transition n'est plus instantanée mais est caractérisée par une durée : **Perte de l'atomicité**

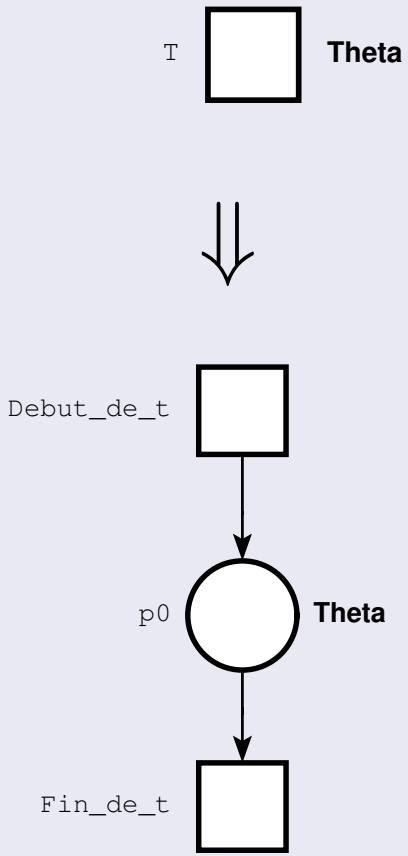


Comment représenter des transitions non atomiques ?

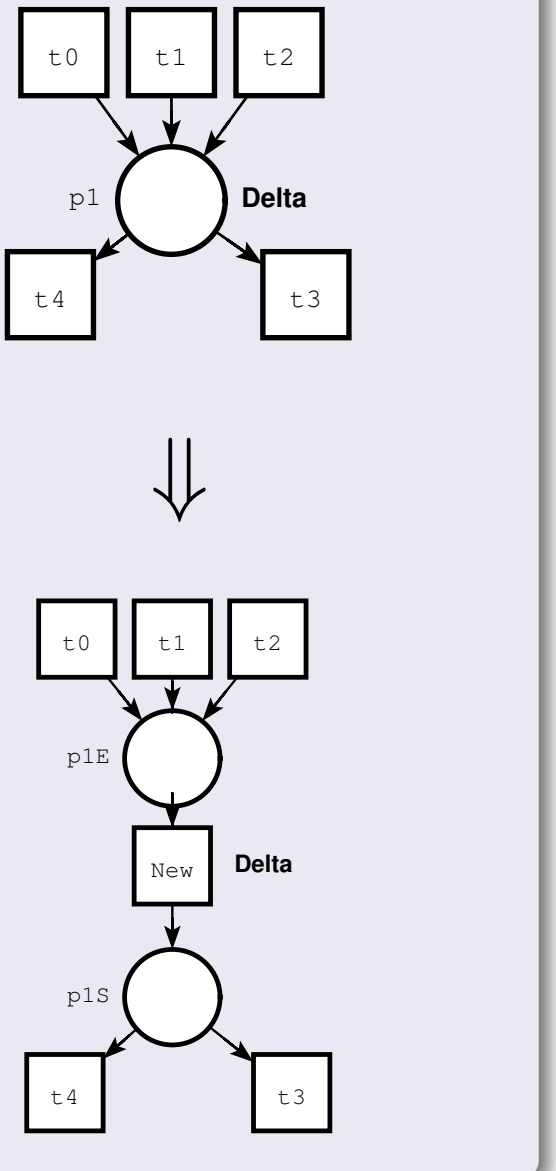
Pour "retrouver" l'atomicité des transitions dans le comportement, chaque transition est découpée en 2 transitions :
Debut_de_Tir & Fin_de_Tir

Equivalence des modèles Temporisés (Séquences)

T-temporisés ⊂ P-temporisés

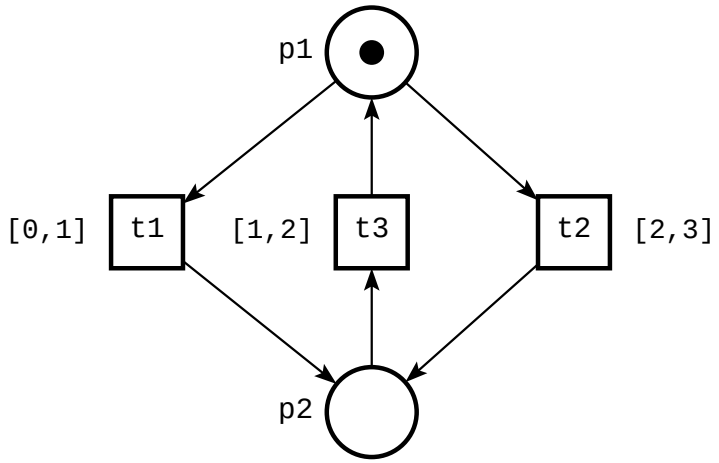


P-temporisés ⊂ T-temporisés



Outline

Réseaux Temporels (Merlin)



Réseau Temporel

est un couple $< R, I_0 >$ où :

$R =$ $< P, T, Pre, Post >$ est un RdP place/transition (classique)

I_s : Application Intervalle de Tir (statique) qui associe à chaque transition son intervalle de tir.

$I_s : \quad T \mapsto \mathbb{Q}^{+*} \times \mathbb{Q}^{+*}{}_{\infty}$

$t \mapsto [I_{min}(t), I_{max}(t)]$

$I =_{Not}$

[0,1]	t_1
[2,3]	t_2
[1,2]	t_3

Réseaux Temporels : Définitions

Marquage Temporel

est un couple $E =< M, I >$ où
$M : P \mapsto \mathbb{N}$ est un marquage (classique)
$I : T \mapsto \mathbb{Q}^{+*} \times \mathbb{Q}^{+*}{}_{\infty}$
NB I vérifie $(I(t) = \emptyset \Leftrightarrow M[t \not>)$ et $I(t) \subset I_s(t)$

Réseau Temporel Marqué

est un couple $< R, E_0 >$ où
R est un réseau temporel,
$E_0 =< M_0, I_0 >$ un marquage temporel
$M_0 : P \mapsto \mathbb{N}$ est le marquage initial
I_0 est définie par M_0 et I_s de la façon suivante :

$$\forall t \in T : I_0(t) = \begin{cases} I(t) & \text{si } M_0[t > \\ \emptyset & \text{sinon} \end{cases}$$

Ici $I_0 = \left| \begin{array}{c} [0,1] \\ [2,3] \\ \emptyset \end{array} \right|$

Sémantique

$E =< M, I >$ un marquage temporel, t une transition et $\theta \in \mathbb{Q}^{+*}{}_{\infty}$

Règle de sensibilisation

$E[t : \theta >=_{Not}$ | *La transition t est sensibilisée dans le marquage E pour l'instant θ*

$E[t : \theta >$ *ssi* $(a)\&\ \ (b)$

(a) $M[t >$

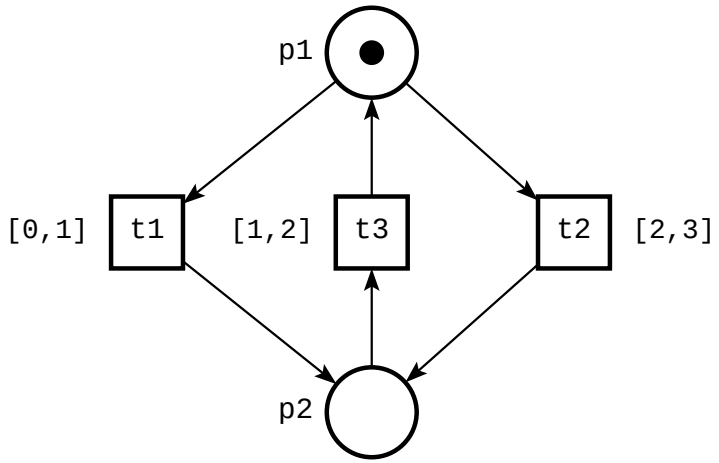
(b) $I_{min}(t) \leq \theta \leq Min(\Theta_{Max}(I, M))$ où

$\Theta_{Max}(I, M) = \{t_{max} : \exists t \in T$ tel que $M[t >$ et $I_{max}(t) = t_{max}\}$

(a) *la transition atemporelle est tirable*

(b) *L'instant θ est compris entre la date de tir au plus tôt de la transition t et la plus petite des dates de tir au plus tard des autres transitions sensibilisées*

Exemple de sensibilisation



- $\{t \in T : M_0[t >\} = \{t_1, t_2\}$
- $\Theta_{Max}(I_0, M_0) = \{I_{0\,max}(t_1), I_{0\,max}(t_2)\} = \{1, 3\}$
- $Min(\Theta_{Max}(I_0, M_0)) = 1$

Donc $E_0[t_1 : \theta >$ pour $\theta \in [0, 1]$
Mais $E_0[t_2 \not>$ car $I_{0\,min}(t_2) = 2 > Min(\Theta_{Max}(I_0, M_0))$

NB $E_0[t_2 \not>$ et $E_0[t_1 >$

Sémantique

$E =< M, I >$ un marquage temporel, t une transition et $\theta \in \mathbb{Q}^{+*}{}_{\infty}$

Règle de Tir

Si $E[t:\theta>$ Alors $E[t:\theta> E'$

$E[t:\theta> E' =_{Not}$ *Le tir de la transition à l'instant* θ *à partir de* E *mène dans l'état* E' *où*

$E' =< M', I' >$ avec $M' = (M - Pre(t)) + Post(t)$

I' est défini $(\forall t \in T)$ par les règles suivantes :

Si $M'[t \not>$ Alors $I'(t) = \emptyset$

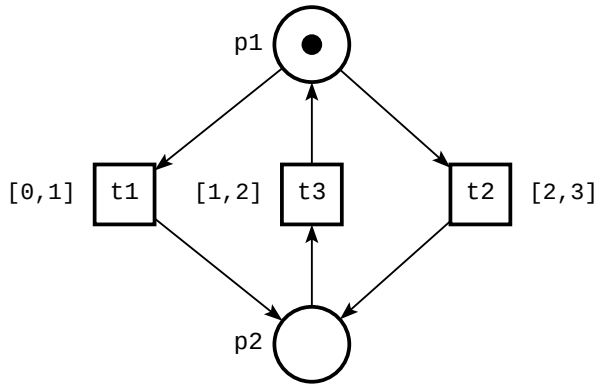
Sinon

Si $M[t>$ Alors $I'(t) = [Max(0, I_{min}(t) - \theta), I_{max}(t) - \theta]$

Sinon $I'(t) = I_0(t)$

NB $I_{max}(t) - \theta \geq 0$ car $\theta \leq Min(\Theta_{Max}(I, M))$

Exemple de franchissement



$E_0 =< M_0, I_0 >=< \begin{vmatrix} 1 \\ 0 \end{vmatrix}, \begin{vmatrix} [0,1] \\ [2,3] \\ \emptyset \end{vmatrix} >$

$E_0[t_1 : \theta_1 > E_1$

avec $E_1 =< M_1, I_1 >=< \begin{vmatrix} 0 \\ 1 \end{vmatrix}, \begin{vmatrix} \emptyset \\ \emptyset \\ [1,2] \end{vmatrix} >$

NB Comme $\{t : M_0[t >\} \cap \{t : M_1[t >\} = \emptyset$
on a E_1 indépendant de θ_1

Exo :

Trouver les couples (θ, E_θ) tels que
$E_1[t_3 : \theta >$
et $E_1[t_3 : \theta > E_\theta$

Espace des Etats Accessibles

$\mathcal{M}$ Ensemble des marquages temporels

Rappel : Marquage Temporel $= < M, I >$ où
$< M, I > \in \mathbb{N}^P \times (\mathbb{Q}^{+*} \times \mathbb{Q}^{+*}{}_\infty)^T$

Ensemble des Marquages accessibles $A(R, E_0)$

$A_0 = \{E_0\}$
$A_i = \{m \in \mathcal{M} / \exists m' \in A_{i-1}, \exists t \in T, \exists \theta \in \mathbb{Q}^{+*}{}_\infty$
$\quad : m'[t : \theta > m\}$
$A(R, E_0) =_{Def} \bigcup_{i \geq 0} A_i$

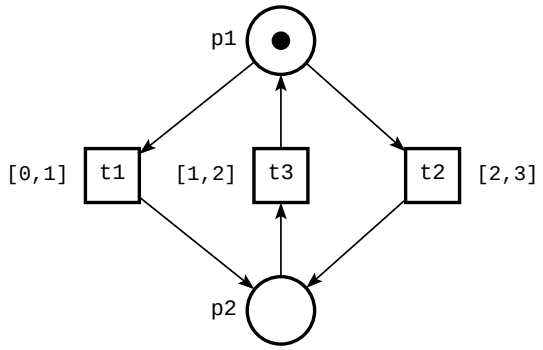
Graphe des Marquages accessibles $G(R, m_0)$

$G(R, m_0) =_{Def} < A(R, m_0), \rightarrow, L >$ **où**
$\rightarrow$ est le plus petit ensemble vérifiant :

$$(m_1, (t : \theta), m_2) \in \rightarrow \text{ ssi } \left| \begin{array}{l} m_1, m_2 \in A(R, m_0), \\ t \in T, t \in \mathbb{Q}^{+*}{}_\infty \\ \text{et } m_1[t : \theta > m_2 \end{array} \right.$$

$L \subset T \times \mathbb{Q}^{+*}{}_\infty$ défini par $(t : \theta) \in L$ ssi $(m_1, (t : \theta), m_2) \in \rightarrow$

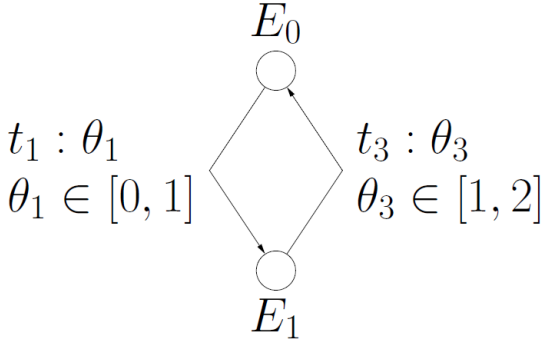
Exemple



$A_0 = \{< \left| \begin{matrix} 1 \\ 0 \end{matrix} \right|, \left| \begin{matrix} [0,1] \\ [2,3] \\ \emptyset \end{matrix} \right| >\}$ et $A_1 = \{< \left| \begin{matrix} 0 \\ 1 \end{matrix} \right|, \left| \begin{matrix} \emptyset \\ \emptyset \\ [1,2] \end{matrix} \right| >\}$

$A_2 = A_0$ donc $A(R, E_0) = \{E_0, E_1\}$

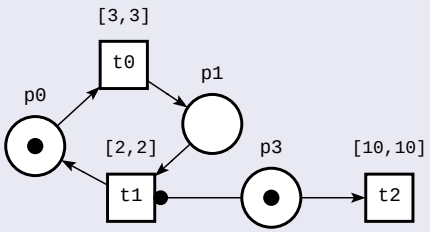
$$\rightarrow = \begin{matrix} \{(E_0, (t_1 : \theta), E_1) : \theta \in [0,1]\} \\ \cup \\ \{(E_1, (t_3 : \theta), E_0) : \theta \in [1,2]\} \end{matrix}$$

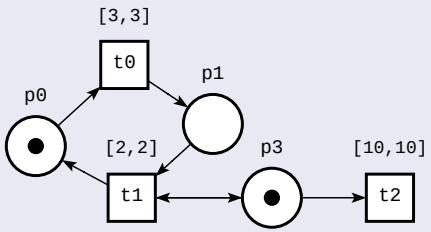


Ici $\rightsquigarrow$ Graphe ayant un nombre fini d'états et une infinité de transitions !

Read-Arcs

En atemporel : “read-arc” équivalent au test (arc entrant et sortant)





Retour sur la règle de tir :

On est dans le marquage p1p3, on franchit t1 et on arrive en p0p3. Que penser de t2 ?

+ t2 est-elle nouvellement sensibilisée en po.p3 ? i.e le tir de t1 a désensibilisé t2
+ t2 est-elle persitente ? i.e le tir de t1 n’a pas désensibilisé t2

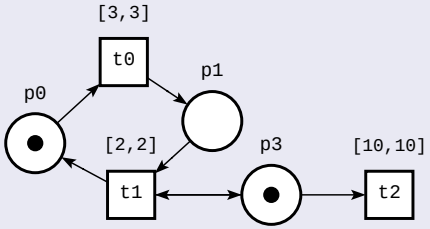
Comme les deux interprétations sont légitimes, on a recours au **read arc** pour pouvoir utiliser les deux.

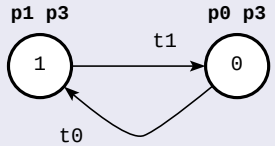
Cas du temporel

Avec read-arc : le tir de t1 ne désensibilise pas t2
Sans read-arc : le tir de t1 désensibilise t2

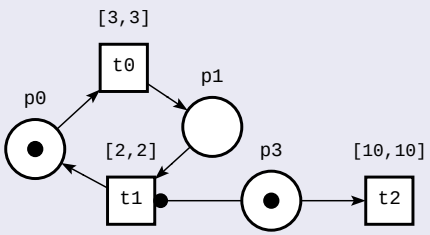
Exemple **avec** et **sans** read-arc

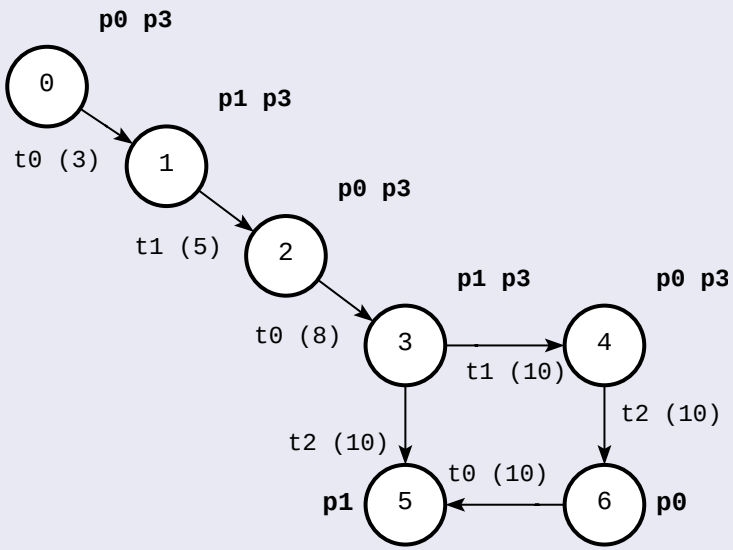
Sans Read-arc





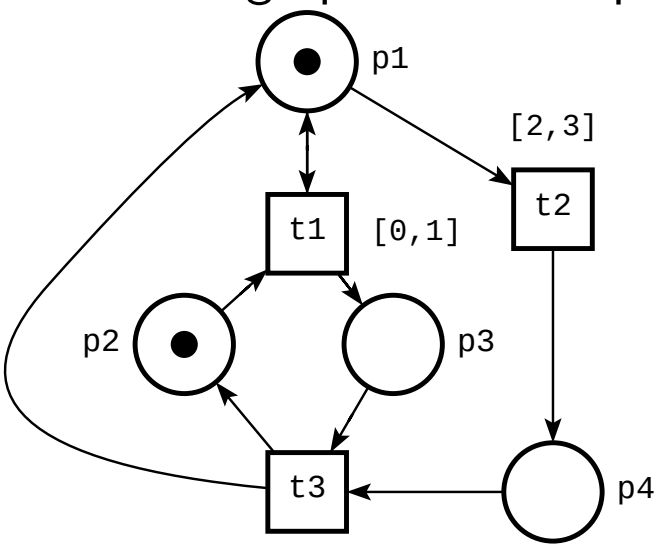
Avec Read-arc

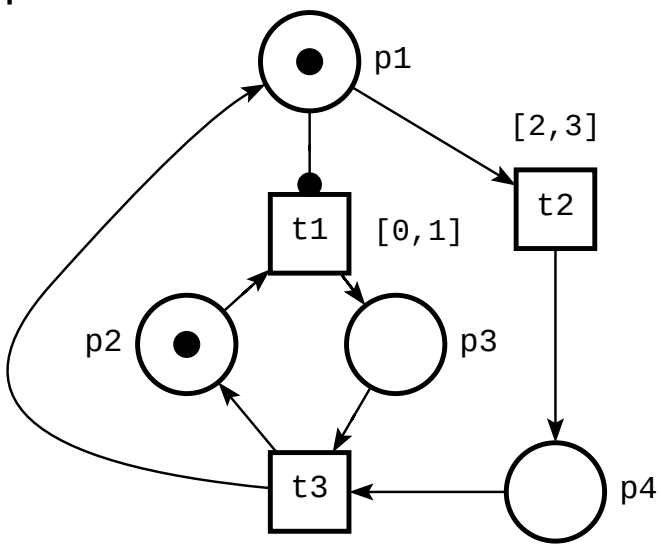




Exercice

Construire le graphe des marquages temporels des réseaux ci-dessous :

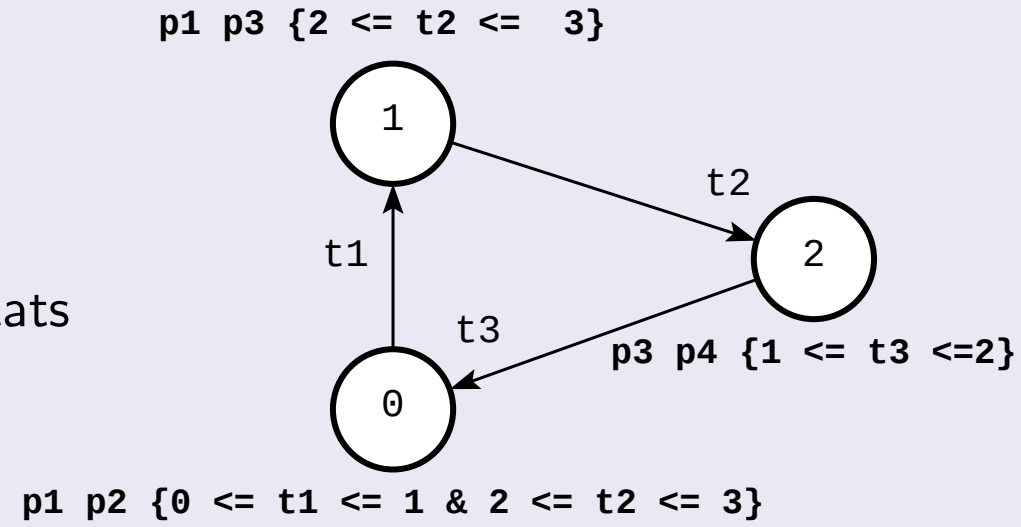




Solution

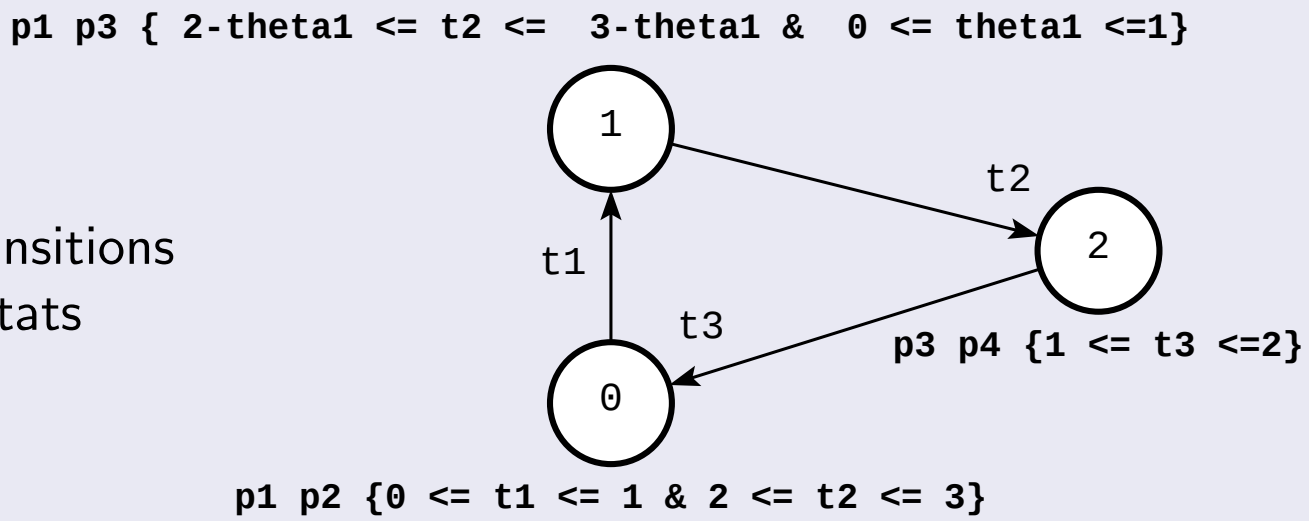
Sans Read-arc

Infinité de transitions
mais nombre fini d'états



Avec Read-arc

Infinité de transitions
et infinité d'états



“Conservation” des propriétés entre le modèle temporel et atemporel

Rdp P/T atemporel sous-jacent

On considère le RdP P/Tobtenu en oubliant les étiquettes temporelles.

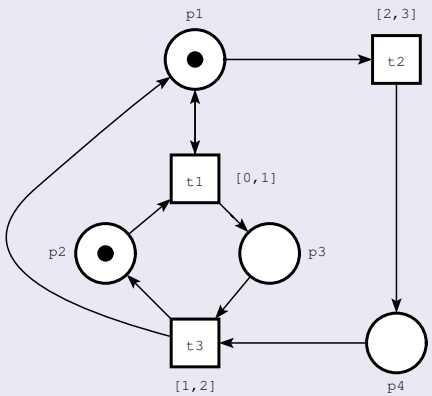
Propriétés

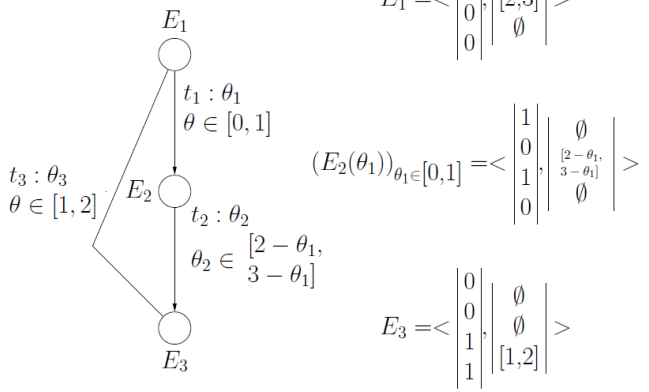
- Rdp P/T Borné $\Rightarrow$ Rdp Temporels Borné
- Rdp P/T sans blocage $\Rightarrow$ Rdp Temporels sans blocage
- Invariants de Place du réseau sous-jacent sont valides pour le RdP Temporel
- Rdp P/T Vivant ? Rdp Vivant (cf pages suivantes)

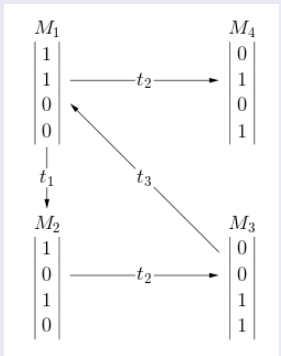
- Finitude du Graphe de Marquages est Indécidable
  (Réseaux temporels $\equiv$ RdP à Arcs inhibiteurs)

Vivacité

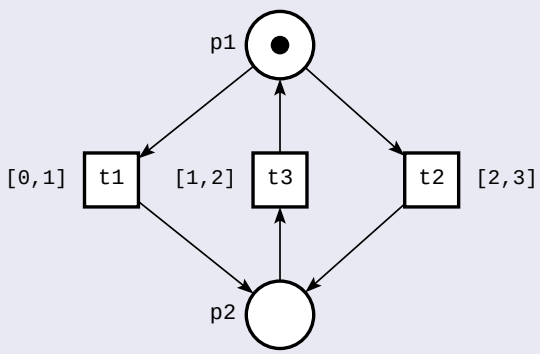
Rdp Temporel Vivant tandis que son Rdp sous-jacent est non Vivant

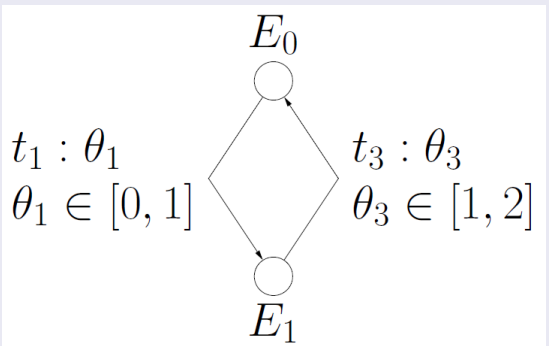


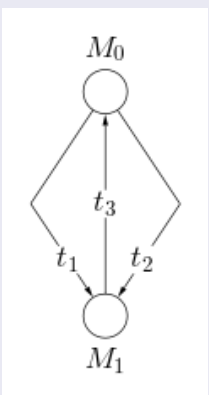




Rdp Temporel non Vivant tandis que son Rdp sous-jacent est Vivant







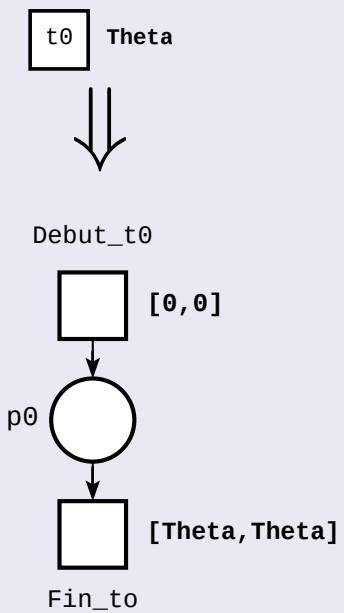
Expressivité* des 4 modèles (suite)

Comparaison wrt traces

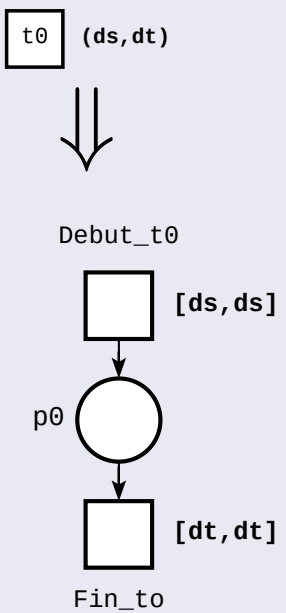
Ramchandani, Razouk, Sifakis $\subset$ Merlin

Ext $\subset$ Melin ssi $\exists$ Réseau temporel admettant les mêmes séquences de franchissement que Ext

Ramchandani $\subset$ Merlin



Razouk $\subset$ Merlin



Outline

Analyse d'un rdp temporel : $\mathcal{M} \subset \mathbb{N}^P \times (\mathbb{Q}^{+*} \times \mathbb{Q}^{+*}{}_\infty)^T$

PB / Solution

PB : Construction d'un graphe de marquages temporels peut genérer une infinité de marquages et de transitions y compris si le RdP sous-jacent est borné. Pb genéral à toute technique introduisant le temps !
Solution : Trouver une "abstraction" finie (de ce graphe infini) permettant de préserver "certaines" capacités d'analyse.

Différentes propriétés pertinentes

Atemporelles/Qualitatives états discrets, blocages, langage (séquences de transitions), LTL, branchement, CTL, ...

Temporelles/Quantitatives durées, wcet, bcet, ...

Apercu de deux abstractions

- Graphe(s) de Classes (dense) [Berthomieu & Menasche 1983],
- Essential Sates (discret) [Popova 1989] ...

1. Avec écoulement de temps : unitaire (-f1), abstrait (-f)
2. sans écoulement de temps (-d)

"Etats essentiels" avec écoulement (unitaire) du temps

Etat temporel : variante à base d'horloges

A chaque transition est associée une horloge. Cette horloge est initalisée à zéro lorsque la transition est sensibilisée ; elle compte le temps depuis lequel la transition est sensibilisée.
Par convention si t n'est pas sensibilisée en m alors $v(t) = \perp$
Etat temporel $=_{Def}$ (Marquage, Vecteur d'horloges)
$< m, v >$ où $m \in P \mapsto \mathbb{N}$ et $v \in T \mapsto \mathbb{R} \cup \{\perp\}$

Etat "essentiel"

Etat à vecteur d'horloges **discret** i.e $< m, v > \in \mathbb{N}^P \times \mathbb{N}_\perp^T$ Comportement abstrait réduit aux écoulements de temps discrets

Graphe d'états essentiels ("unitaires")

$G =< E_0, Es, \rightarrow >$ avec
$Es \subset \mathbb{N}^P \times \mathbb{N}_\perp^T$
$\rightarrow \subset Es \times T \cup \{1\} \times Es$
et $E_0 =< m_o, v_0 >$ où $m_o \in \mathbb{N}^P$ et
v_0 est défini par $v_0(t) = \begin{cases} 0 & \text{si } m_0[t > \\ \perp & \text{sinon} \end{cases}$

Graphe des états essentiels "unitaires" : construction

Comportement : Transitions discretes + Ecoulements de temps

Transitions discretes :

Soit $t \in T$, si $m[t>$ et $v(t) \in I(t)$ alors
$< m, v > [t >< m', v' >$ avec $m' = m \setminus^\bullet t + t^\bullet$

$$et\ v'(t') = \begin{cases} \perp & \text{si } m'[t' \not> \\ 0 & \text{si } m'[t' > \text{ et } m[t' \not> \\ v(t') & \text{sinon (i.e. } m[t' > \text{et} m'[t' >) \end{cases}$$

Transitions d'écoulement du temps :

Ecoulement maximal de temps : Soit $v \in \mathbb{N}_\perp^T$, on note $\theta_{Max}(v)$ l'écoulement maximal de temps autorisé par v i.e. $\theta_{Max}(v) = Min\{(I_{max}(t) - v(t)) : t \in T\}$

Si $\theta_{Max}(v) \geq 1$ alors $< m, v > [1 >< m, v' >$

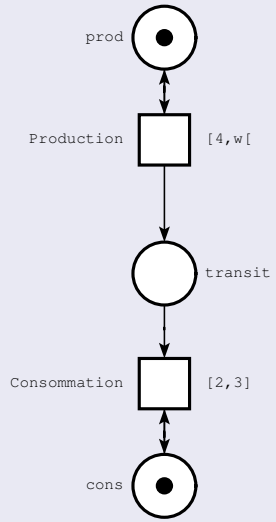
$$\text{avec } v'(t) = \begin{cases} \perp & \text{si } v(t) = \perp \\ v(t) + 1 & \text{si } I_{max}(t) \in \mathbb{N} \\ Min(v(t) + 1, I_{min}(t)) & \text{sinon (i.e. } I_{max}(t) = \infty) \end{cases}$$

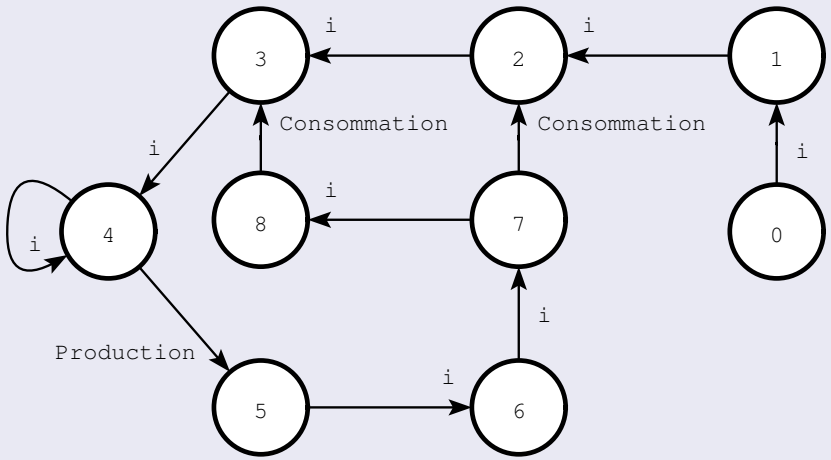
Propriété : Horloges bornées $\mapsto$ Ensemble fini de vecteurs d'horloges

$$\text{Pour } t \in T, v(t) \leq \begin{cases} I_{Max}(t) & \text{si } I_{Max}(t) \neq \infty \\ I_{Min}(t) & \text{sinon} \end{cases}$$

Exemple Producteur Consommateur

Reseau et Analyse (tina -F1)






```
9 classe(s), 11 transition(s)

state 0
    marking
cons prod
    clock vector
Production = 0
    firing domain
4 <= Production
```

```
state 1
    marking
cons prod
    clock vector
Production = 1
    firing domain
3<= Production
```

```
state 2
    marking
cons prod
    clock vector
Production = 2
    firing domain
2<=Production
```

Exemple Producteur Consommateur (suite)

Suite des états

```
state 3
    marking
cons prod
    clock vector
Production = 3
    firing domain
1<=Production
```

```
state 4
    marking
cons prod
    clock vector
Production = 4
    firing domain
0<=Production
```

```
state 5
    marking
cons prod transit
    clock vector
Consommation = 0
Production = 0
    firing domain
2<=Consommation<= 3
4<=Production
```

```
state 6
    marking
cons prod transit
    clock vector
Consommation = 1
Production = 1
    firing domain
1<=Consommation<= 2
3<= Production
```

```
state 7
    marking
cons prod transit
    clock vector
Consommation = 2
Production = 2
    firing domain
0<= Consommation<= 1
2<= Production
```

```
state 8
    marking
cons prod transit
    clock vector
Consommation = 3
Production = 3
    firing domain
0<=Consommation<= 0
1<=Production
```

Etats essentiels avec écoulement du temps (`tina -F`)

Objectif : réduire la taille de l'espace d'états du graphe "unitaire" (-F1)

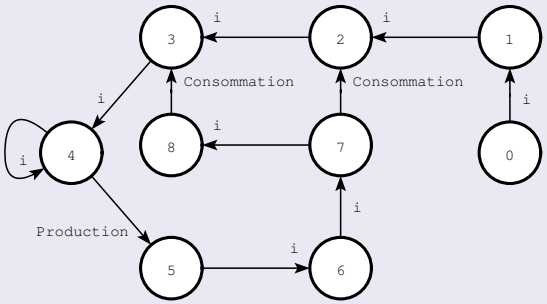
Principe : *On concatène "certaines" transitions d'écoulement de temps, les états intermédiaires sont oubliés ....* + formellement

(1) Soit $\equiv$, la relation d'équivalence sur les sommets du graphe, définie par : Si $s[i > s'$ et $s[t \not>$ et $s[t' \not>$ pour $t \in T$ alors $s \equiv s'$

(2) On quotiente le graphe "unitaire" par $\equiv$

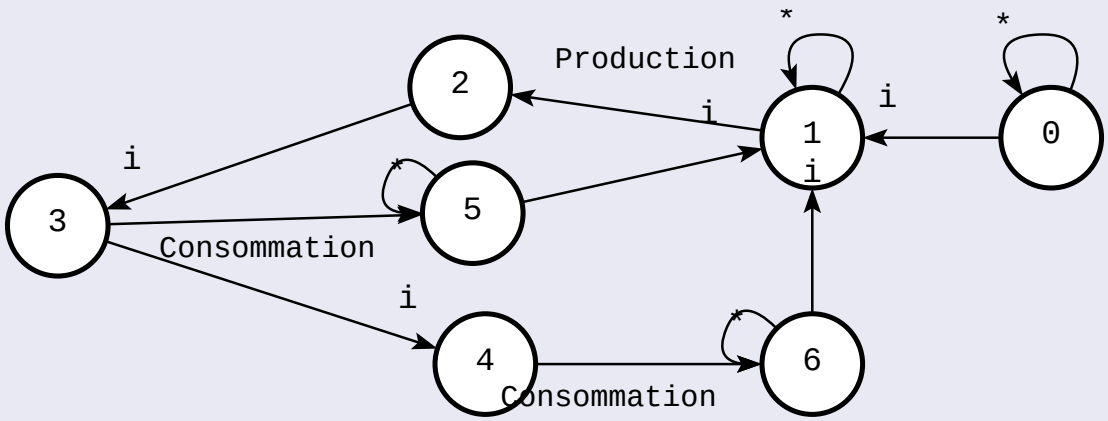
Exemple : Producteur/Consommateur (suite)

`tina -F1`



$0_{\equiv} = \{0, 1, 2, 3\}$

-F = Quotient de -F1 par $\equiv$



Graphes des états essentiels (sans écoulement de temps) `tina -D`)

Principe : suppression des transitions d'écoulement du temps

(Rappel) Automates à état finis $\mapsto$ Réduction des λ-transitions

On substitue la relation $\lambda^* \otimes t$ à la relation de transition initiale

où λ^* = fermeture réflexive et transitive de la relation λ

Sur un exemple :

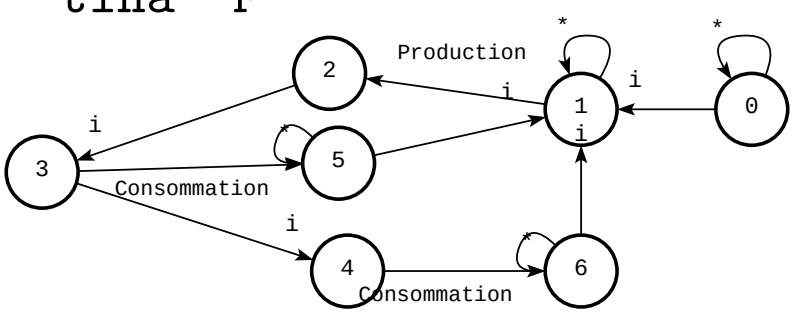
Calcul de 1^*

$0 \mapsto \{0, 1\}$
$1 \mapsto \{1\}$
$2 \mapsto \{2, 3, 4\}$
$3 \mapsto \{3, 4\}$
$4 \mapsto \{4\}$
$5 \mapsto \{5, 1\}$
$6 \mapsto \{6, 1\}$

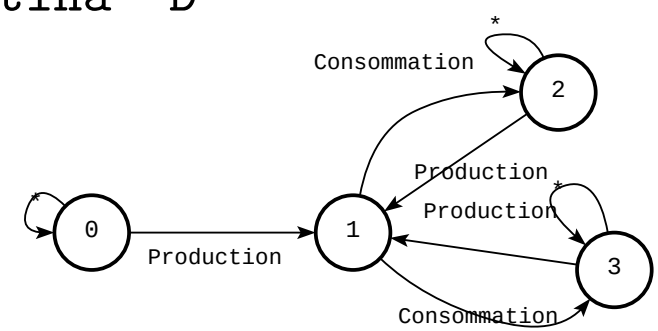
Calcul de $\lambda^* \otimes t$

0	*production*	$\mapsto \{2\}$
2	*consommation*	$\mapsto \{5, 6\}$
5	*production*	$\mapsto \{2\}$
6	*production*	$\mapsto \{2\}$

`tina -F`



`tina -D`



Etats essentiels bilan

Remarques

- Présentation successives des 3 constructions : mais elles peuvent être directement calculées à la volée.
- Graphes des états essentiels est non-déterministe. Dans le cas géreral, un automate non déterministe est en géneral plus compact que son correspondant déterministe ($n \mapsto 2^n$)

Propriétés préservées

Les 3 constructions présentées préservent les états discrets et les séquences, ainsi que les blocages et la divergence temporelle (LTL) Elles ne préservent pas le branchement ($\not CTL$) Des variantes permettent d'obtenir les temps extrémaux.

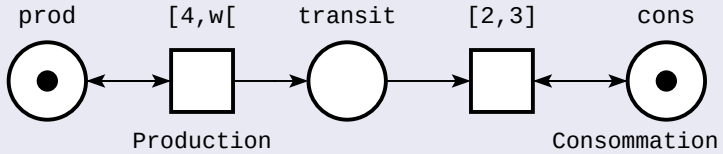
Avantages/Inconvenients

Méthodes simples et faciles à expliquer/programmer .... mais sensibles à l'échelle. .../...

Du discret au dense

Sensibilité au facteur d'échelle des approches discretes

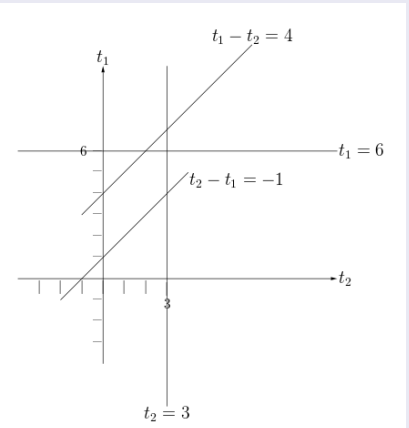
On dilate de façon uniforme les intervalles temporels, rien ne devrait changer au niveau du comportement et pourtant ...



Tina	$[4,\infty[-[2,3]$	$\times 10$	$\times 100$	$[4000,\infty[-[20,3000]$
$-D$	4/5	13/23	103/203	2983/5563
$-F$	7/9	25/36	205/306	5965/8946
$-F1$	9/11	72/83	702/830	7002/9983
$-W$	3/3	3/3	3/3	3/3

Abstraction dense et non plus discrète

Représentation des domaines de tir par des polyèdres



Abstractions denses : Graphes de Classes

Principe :

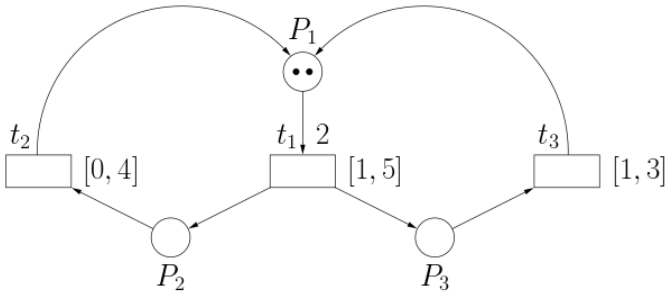
On ne construit plus un sous-graphe du graphe reel comme dans les approches discretes, on construit un graphe abstrait dont les sommets correspondent à des classes états temporels.
L'espace d'états est "recouvert" par des ensembles convexes d'états (classes d'états) de telle façon que $c \xrightarrow{t} c'$ ssi $\exists s \in c, \exists s', \exists \theta \in c' : s \xrightarrow{t:\theta} s'$

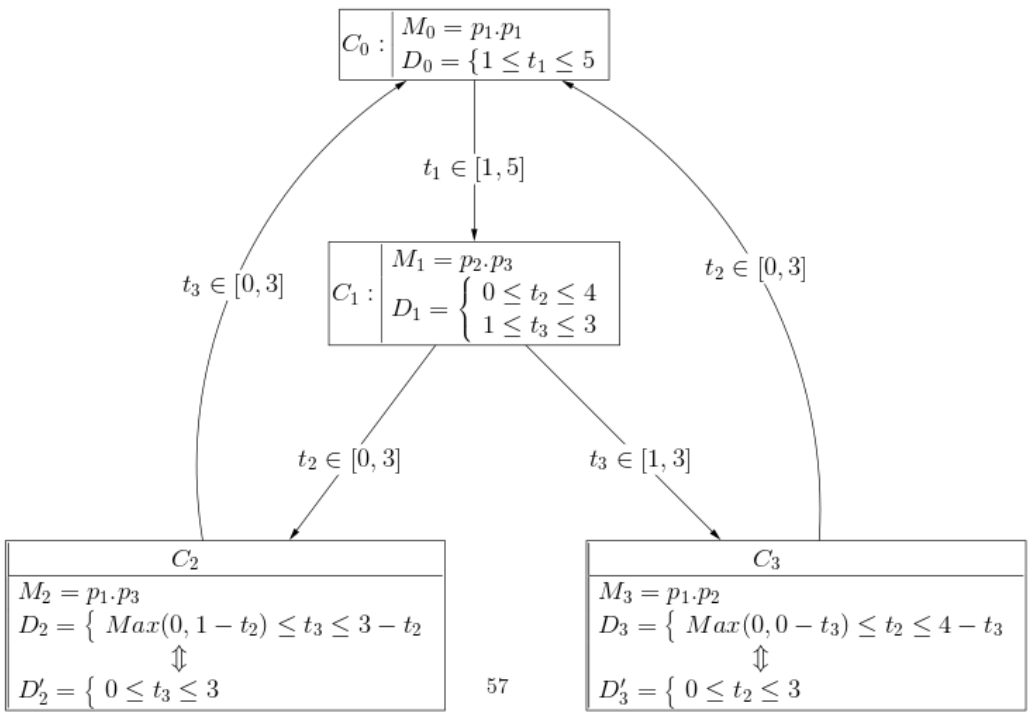
Equivalence de domaines temporels : I et I'

Opérationnellement : Comparer les espaces de solutions respectifs
$\mapsto I \equiv I'$ ssi $Sol(I) = Sol(I')$
(i.e, I et I' correspondent au même espace de solutions)
Impraticable
Dénotationnellement : Fournir une représentation **canonique** de l'espace des solutions. i.e $Sol(I) = Sol(I')$ ssi $F.C(I) = F.C(I')$
Ensuite $\mapsto I \equiv I'$ ssi $F.C(I) = F.C(I')$

$\rightarrow$ Avoir $\begin{cases} (1) \text{ une forme canonique} \\ (2) \text{ un algorithme de canonisation} \end{cases}$

Exemple de graphe de classes





57

Forme Canonique d'un domaine de tir

Forme **normale** d'un domaine de tir

Forme **Normale** $\begin{cases} a_i \leq t_i \leq b_i & \forall i \\ t_j - t_k \leq c_{jk} & \forall j,k \text{ avec } j \neq k \end{cases}$ où

t_i, t_j, t_k sont des transitions sensibilisées en M,

$a_i \in \mathbb{Q}^{+*}, b_i \in \mathbb{Q}^{+*}$ et $c_{jk} \in \mathbb{Q}^{+*} \cup \{\infty\}$

Forme **Canonique** d'un système sous Forme Normale

Forme **Canonique** $\begin{cases} a_i^* \leq t_i \leq b_i^* & \forall i \\ t_j - t_k \leq c_{jk}^* & \forall j,k \text{ avec } j \neq k \end{cases}$ où

a_i^* est la plus petite valeur possible pour t_i

b_i^* est la plus grande valeur possible pour t_i

c_{jk}^* est la plus grande différence possible pour les valeurs t_j et t_k

Propriété des domaines de tir

Tout domaine de tir (I) d'une classe d'états d'un PN $(< M, I >)$ peut-être exprimé sous forme **Normale** et **Canonique**

Construction du Graphe de Classes

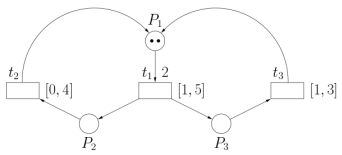
Soit $E =< M, I >$ un état Temporel, t une transition et θ un instant tel que $E[t : \theta > E'$ avec $E' =< M', I' >$

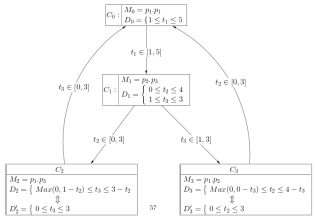
Construction de la forme canonique associée C' associée au domaine I'

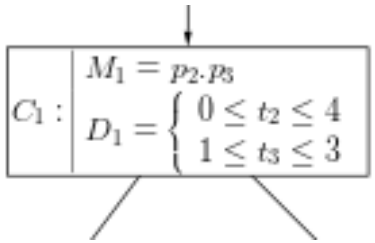
I' est obtenu en 5 étapes à partir de I :

1. Ajouter les conditions de Tir de t. Pour chaque "transition $t_i \in I$" (sensibilisée en E), ajouter $t \leq t_i$ à I.
   On note I_1 le domaine obtenu.
2. "Eliminer" de I_1 toutes les variables t_o associées aux transitions t_o qui étaient sensibilisées en M et ne le sont plus en M'.
   On note I_2 le domaine obtenu.
3. "Eliminer" t de I_2. On note I_3 le domaine obtenu.
4. Pour chaque transition t_n nouvellement sensibilisée en M', ajouter à I_3 l'inégalité $I_{0\,min}(t_n) \leq t_n \leq I_{0\,max}(t_n)$.
   On note I_4 le domaine obtenu.
5. "Canoniser" I_4 pour obtenir I'

Exemple de construction de graphe des marquages







On se place en C_1 et on tire t_2 :

1) $\begin{cases} 0 \le t_2 \le 4 \\ 1 \le t_3 \le 3 \end{cases}$ $\quad I_{1-1} \begin{cases} 0 \le t_2 \le 4 \\ 1 \le t_3 \le 3 \\ t_2 \le t_3 \end{cases}$

2) sans objet

3) Elimination de t_2 : a) Changement de variable $t_3 = t_2 + t'_3$

$I_{1-1} \equiv \begin{cases} 0 \le t_2 \le 4 \\ 1 \le t_2 + t'_3 \le 3 \\ t_2 \le t_2 + t'_3 \end{cases}$ $\quad I_{1-1} \equiv \begin{cases} 0 \le t_2 \le 4 \\ 1 - t'_3 \le t_2 \le 3 - t'_3 \\ 0 \le t'_3 \end{cases}$

b) Application des règles d'élimination de Fourier/Motzkin :

$\begin{cases} a_1 \le x \le b_1 \\ c_1 \le x \le d_1 \end{cases} \mapsto \begin{cases} a_1 \le b_1 \\ a_1 \le d_1 \\ c_1 \le b_1 \\ c_1 \le d_1 \end{cases}$ $\quad$ Ici $I_{1-1} \mapsto \begin{cases} 0 \le 4 \\ 0 \le 3 - t'_3 \\ 1 - t'_3 \le 4 \\ 1 - t'_3 \le 3 - t'_3 \\ 0 \le t'_3 \end{cases}$

$I_{1-2} \begin{cases} 0 \le t'_3 \le 3 \end{cases}$ $\quad$ Ici 4) et 5) sans objet donc terminé !

Autre exemple de normalisation

.... " on a tiré t_5", on élimine t_5

$$\begin{cases} 4 \leq t_1 \leq 6 \\ 2 \leq t_2 \leq 3 \\ 0 \leq t_5 \leq 3 \\ t_5 \leq t_1 \\ t_5 \leq t_2 \end{cases} \mapsto \begin{cases} 4 \leq t'_1 + t_5 \leq 6 \\ 2 \leq t'_2 + t_5 \leq 3 \\ 0 \leq t_5 \leq 3 \\ t_5 \leq t'_1 + t_5 \\ t_5 \leq t'_2 + t_5 \end{cases} \mapsto \begin{cases} 4 - t'_1 \leq t_5 \leq 6 - t'_1 \\ 2 - t'_2 \leq t_5 \leq 3 - t'_2 \\ 0 \leq t_5 \leq 3 \\ 0 \leq t'_1 \\ 0 \leq t'_2 \end{cases}$$

$$\mapsto \begin{cases} 4 - t'_1 \leq 6 - t'_1 & 4 \leq 6 \\ 4 - t'_1 \leq 3 - t'_2 & t'_2 - t'_1 \leq -1 \\ 4 - t'_1 \leq 3 & 1 \leq t'_1 \\ 2 - t'_2 \leq 6 - t'_1 & t'_1 - t'_2 \leq 4 \\ 2 - t'_2 \leq 3 - t'_2 & 2 \leq 3 \\ 2 - t'_2 \leq 3 & -1 \leq t'_2 \\ 0 \leq 6 - t'_1 & t'_1 \leq 6 \\ 0 \leq 3 - t'_2 & t'_2 \leq 3 \\ 0 \leq 3 & 0 \leq 3 \\ 0 \leq t'_1 & 0 \leq t'_1 \\ 0 \leq t'_2 & 0 \leq t'_2 \end{cases}$$

Forme Normale

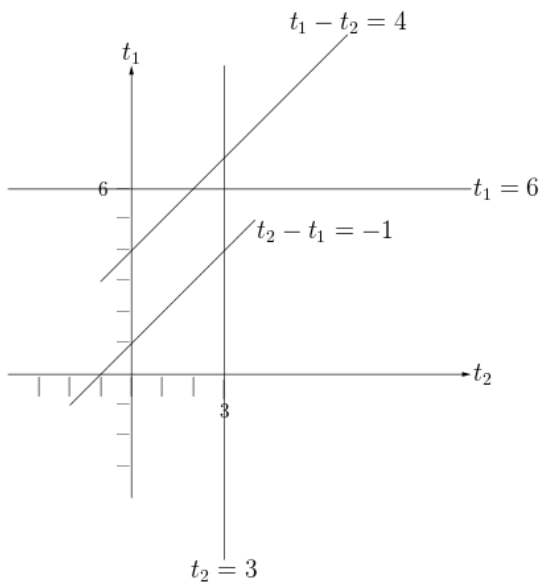
$$\mapsto \begin{cases} 1 \leq t'_1 \leq 6 \\ 0 \leq t'_2 \leq 3 \\ t'_2 - t'_1 \leq -1 \\ t'_1 - t'_2 \leq 4 \end{cases}$$

En fait elle est canonique !

Nécessité de la canonisation

On "décanonise" $IC \begin{cases} 1 \leq t'_1 \leq 6 \\ 0 \leq t'_2 \leq 3 \\ t'_2 - t'_1 \leq -1 \\ t'_1 - t'_2 \leq 4 \end{cases}$ par exemple $I \begin{cases} \mathbf{0} \leq t'_1 \leq 6 \\ 0 \leq t'_2 \leq 3 \\ t'_2 - t'_1 \leq -1 \\ t'_1 - t'_2 \leq 4 \end{cases}$

Graphiquement ce sont pourtant les mêmes domaines



Canonisation par élimination de Fourier/Motzkin

Application de Fourier/Motzkin

$$\begin{cases} 0 \leq t_1 \leq 6 \\ 0 \leq t_2 \leq 3 \\ t_2 - t_1 \leq -1 \\ t_1 - t_2 \leq 4 \end{cases}$$

On élimine t_1

$$\begin{cases} 0 \leq t_1 \leq 6 \\ 0 \leq t_2' + t_1 \leq 3 \\ t_2' + t_1 - t_1 \leq -1 \\ t_1 - t_2' - t_1 \leq 4 \end{cases} \qquad \begin{cases} 0 \leq t_1 \leq 6 & (a) \\ -t_1 \leq t_2' \leq 3 - t_1 & (b) \\ t_2' \leq -1 & (c) \\ \ldots\ldots\ldots \end{cases}$$

(b) & (c) $\Rightarrow t_1 \geq 1$

On recommence jusqu'à stabilisation ! (convergence dans $\mathbb{Q}^{+*}$)

Canonisation par calcul de plus courts chemins

Principe : Voir le système d'inéquations comme un graphe des coûts

Chaque variable du système d'inéquations correspond à un sommet et chaque inégalité de la forme $t_1 - t_2 \leq k$ correspond à un arc étiquetté $t_1 \xrightarrow{k} t_2$

On introduit une racine

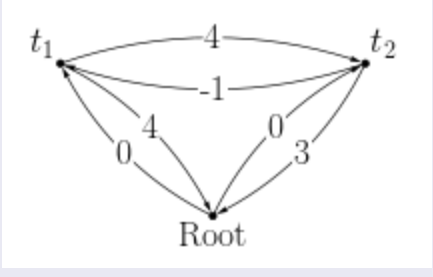
$$\begin{cases} 1 \leq t_1 \leq 6 \\ 0 \leq t_2 \leq 3 \\ t_2 - t_1 \leq -1 \\ t_1 - t_2 \leq 4 \end{cases} \mapsto \begin{cases} 1 \leq t_1 - Root \leq 6 \\ 0 \leq t_2 - Root \leq 3 \\ t_2 - t_1 \leq -1 \\ t_1 - t_2 \leq 4 \end{cases}$$

On transforme pour obtenir une clique à trois sommets : $t_1, t_2, Root$

$$\begin{cases} t_2 - t_1 \leq -1 \\ t_1 - t_2 \leq 4 \end{cases} \mapsto \begin{cases} t_2 \xrightarrow{-1} t_1 \\ t_1 \xrightarrow{4} t_2 \end{cases}$$

$$1 \leq t_1 - Root \leq 6 \mapsto \begin{cases} t_1 \xrightarrow{6} Root \\ Root \xrightarrow{-1} t_1 \end{cases}$$

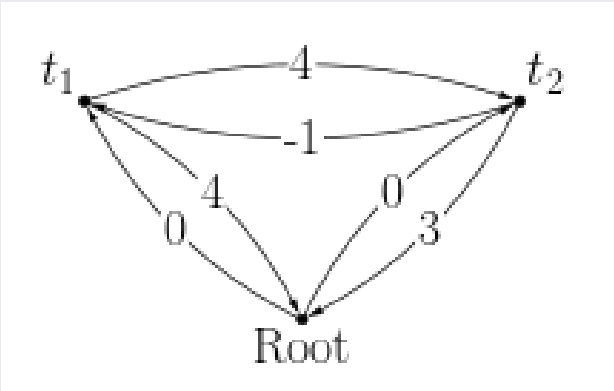
$$0 \leq t_2 - Root \leq 3 \mapsto \begin{cases} t_2 \xrightarrow{3} Root \\ Root \xrightarrow{0} t_2 \end{cases}$$



Canonisation par calcul de coûts minimaux (fin)

Calcul de coûts minimaux

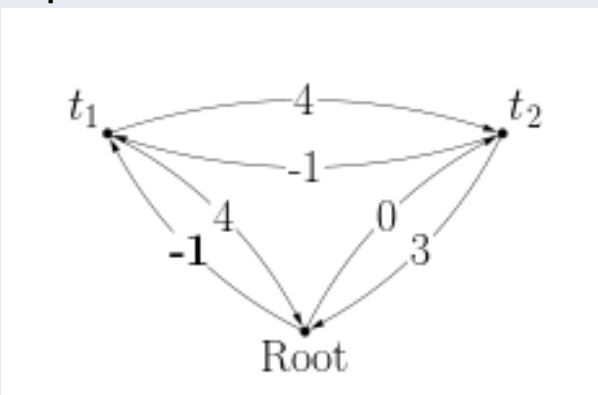
Graphe des Coûts



Forme Normale

$$\mapsto \begin{cases} 0 \leq t_1 \leq 6 \\ 0 \leq t_2 \leq 3 \\ t_2 - t_1 \leq -1 \\ t_1 - t_2 \leq 4 \end{cases}$$

Graphe des Coûts Minimaux



Forme Canonique

$$\mapsto \begin{cases} 1 \leq t_1 \leq 6 \\ 0 \leq t_2 \leq 3 \\ t_2 - t_1 \leq -1 \\ t_1 - t_2 \leq 4 \end{cases}$$

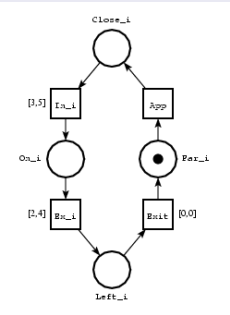
Calcul de plus courts chemins (avec des coûts négatifs) - Floyd-Warshall

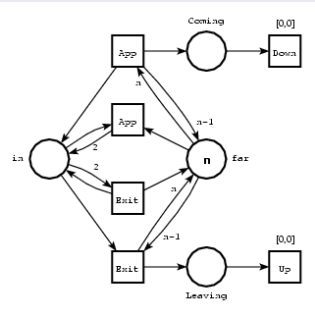
Diverses abstractions disponibles dans TINA

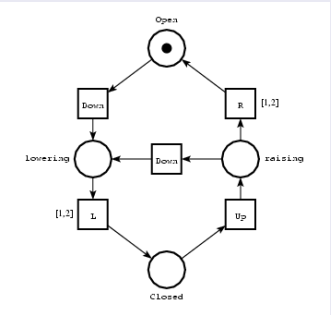
Objectif

Disposer de l'abstraction la plus **grossière** adéquate avec la classe de propriétés que l'on cherche à vérifier : marquages, séquences, LTL (séquences + divergence), CTL

Passage à niveau



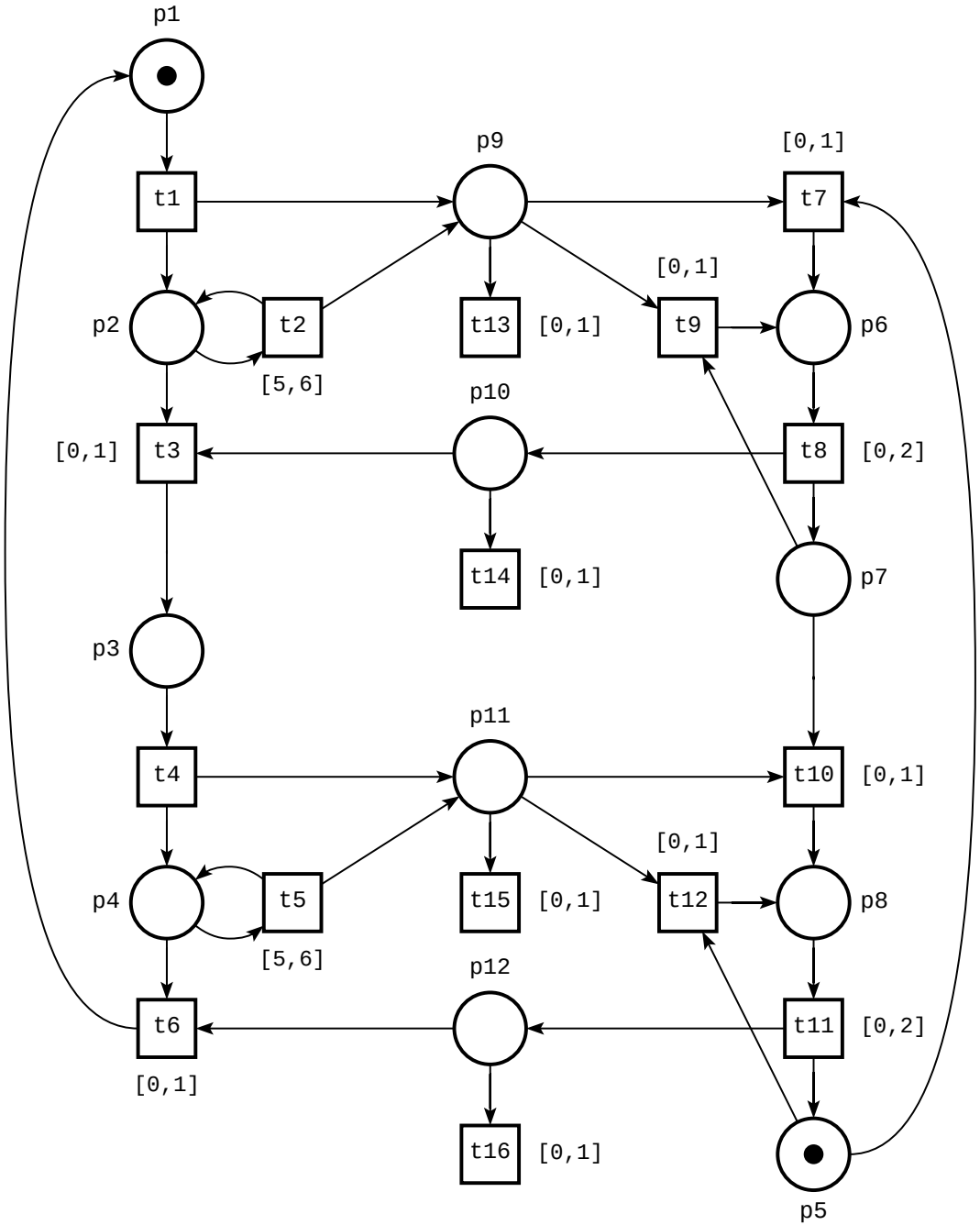




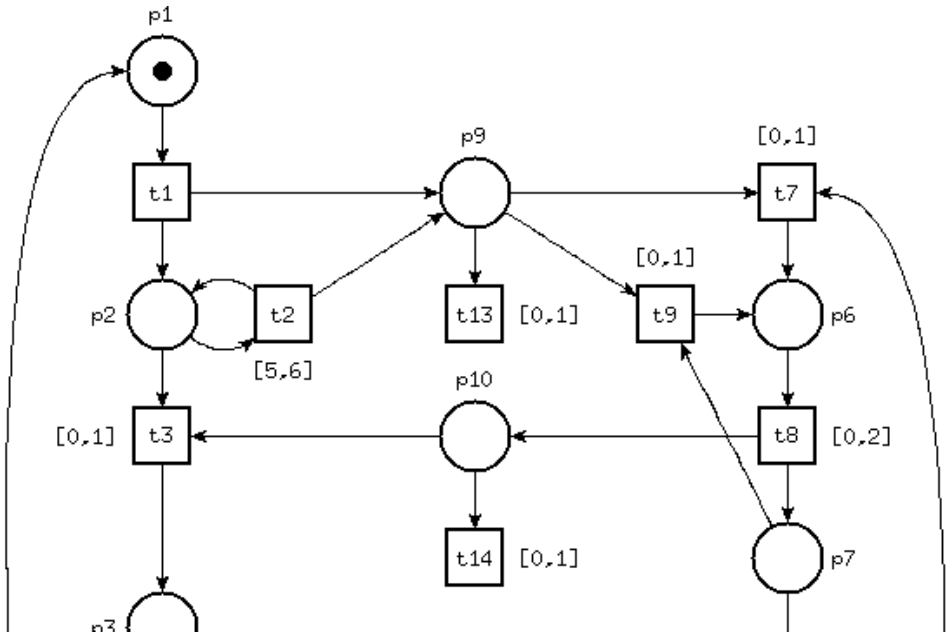
Illustration

		M tina -M	$M + LTL$ **tina -W**	E tina -E	$E + LTL$ tina -S	$E + CTL$ tina -A	$M + LTL$ (*discret*) **tina -D**	$M + LTL$ (*discret*) **tina -F**
	Cl	10	11	10	11	12	13	23
(1)	*Tr*	13	14	13	14	16	27	36
	Cl	37	123	41	141	195	116	382
(2)	*Tr*	74	218	82	254	849	198	373
	Cl	172	3101	232	5051	6973	1550	2280
(3)	*Tr*	492	7754	672	13019	49818	5823	5275
	Cl	1175	134501	1807	351271	356940	22268	28830
(4)	*Tr*	4534	436896	7062	1193376	1447835	91256	81077
	Cl	10972	855762	18052	35945411	23081275	313214	372264
(5)	*Tr*	53766	34337748	89166	151908273	279572133	1397517	1245355

Exemple du Bit Alterné



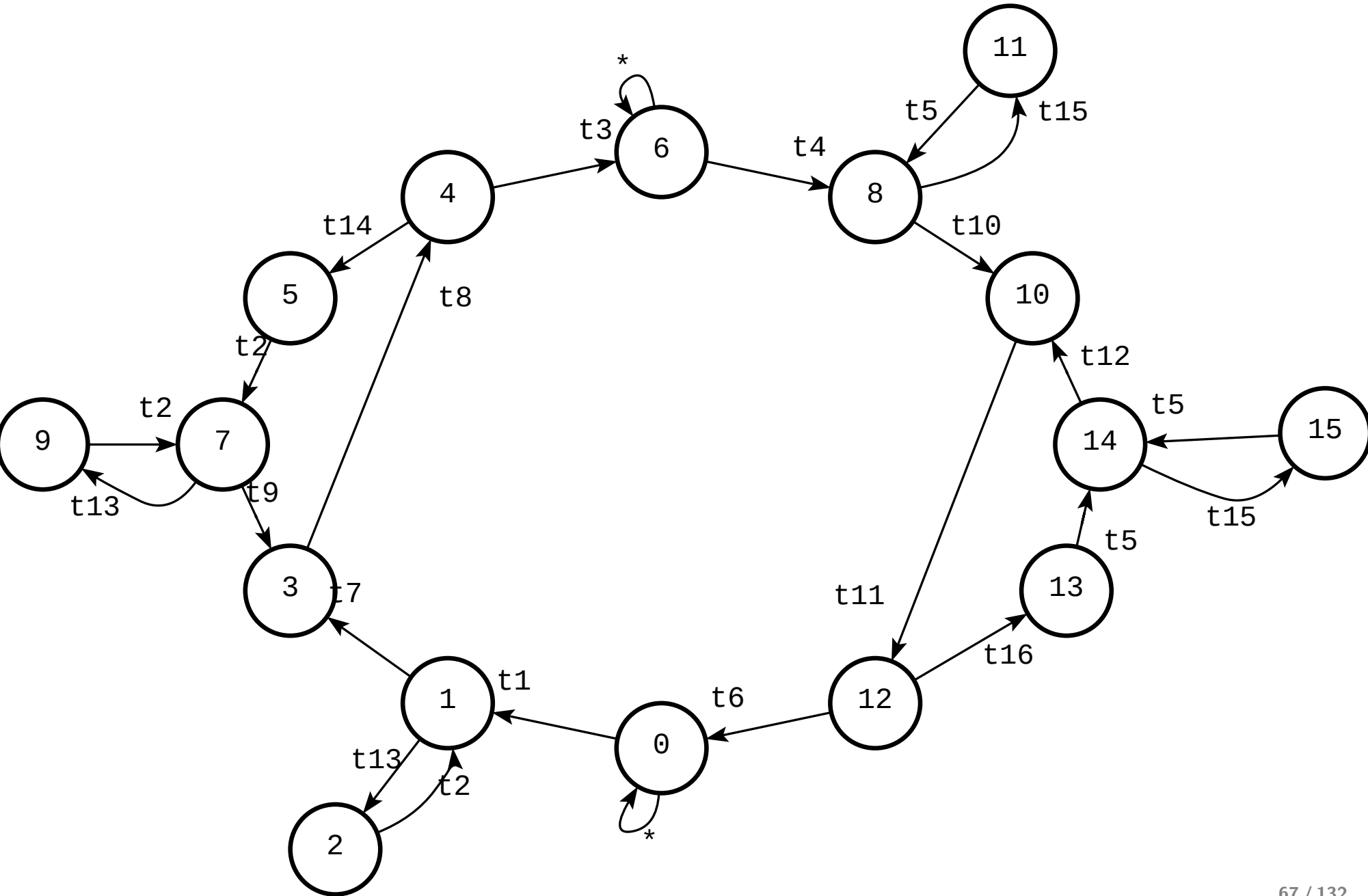
1/2 Bit Alterné : Légende



Bit 0

Transitions	Places
t1 émission du message	P1 Emetteur prèt à émettre
t2 timeout et retransmission	P2 Emetteur en attente Ack
t3 réception de l'Ack	P9 Message en transit
t13 perte du message	P5 Récepteur prèt à recevoir msg bit 0
t7 réception du message	P6 Récepteur prèt à envoyer Ack
t9 **re**-réception du message	P7 Récepteur prèt à recevoir msg bit 0 **et** bit 1
t8 émission de l'Ack	
t14 perte de l'Ack	

Graphe de classes associé



... et quelques classes

```
class 0
   marking
       p1 p5
  domain
      0 <= t1
```

```
class 1
   marking
p2 p5 p9
    domain
0 <= t13 <= 1
5 <= t2 <= 6
0 <= t7 <= 1
```

```
class 2
    marking
p2 p5
    domain
4 <= t2 <= 6
```

```
class 3
    marking
p2 p6
    domain
4 <= t2 <= 6
0 <= t8 <= 2
```

```
class 4
    marking
p10 p2 p7
    domain
0 <= t14 <= 1
2 <= t2 <= 6
0 <= t3 <= 1
```

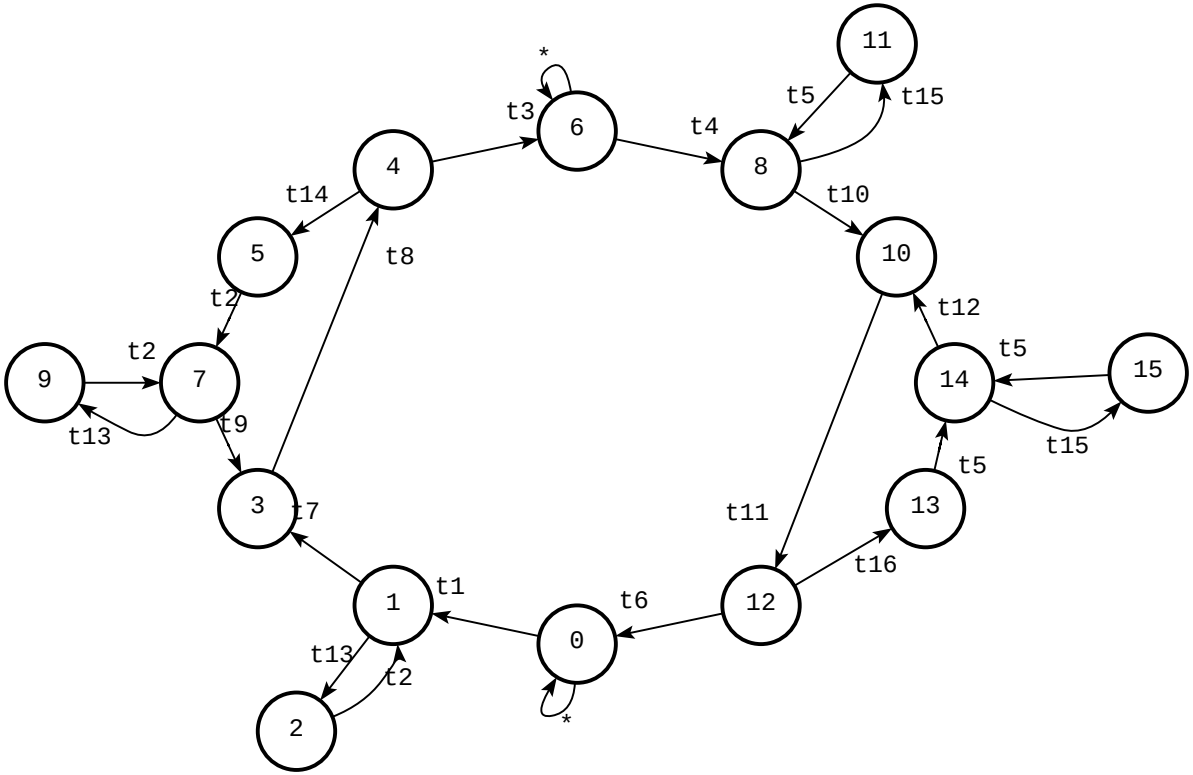
```
class 5
    marking
p2 p7
    domain
1 <= t2 <= 6
```

```
class 6
    marking
p2 p7 p9
    domain
0 <= t13 <= 1
5 <= t2 <= 6
0 <= t9 <= 1
```

```
class 7
    marking
p2 p7
    domain
4 <= t2 <= 6
```

```
class 8
    marking
p3 p7
    domain
0 <= t4
```

Quelques cycles



Commentaires

- t13-> t2-> ... : PerteMesg0-> timeout&!Mesg0

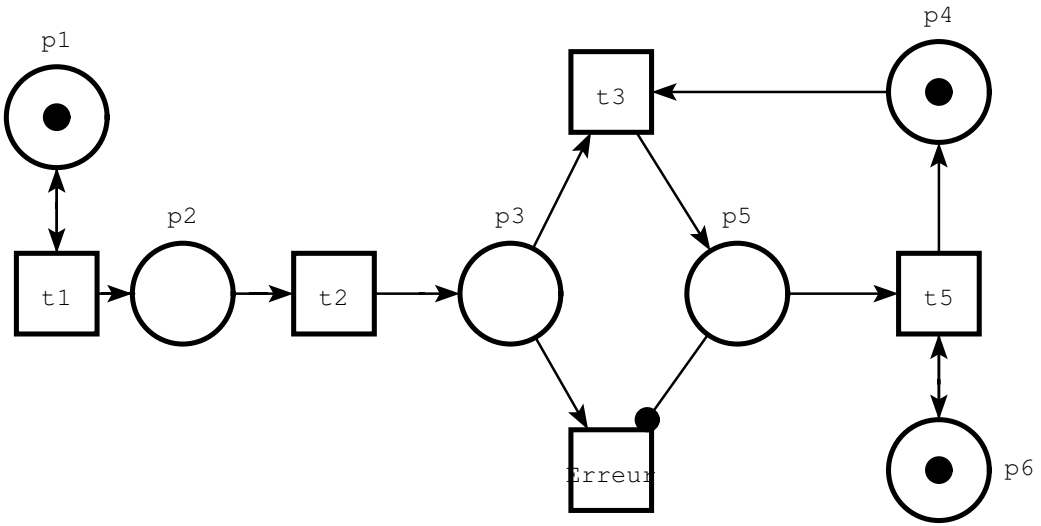
- t8-> t14-> t2-> t9-> ... :
!Ack0-> PerteAck0-> timeout&!Mesg0-> ?Mesg0

Fonct nominal : - t1-> t7-> t8-> t3-> t4-> t10-> t11-> t6-> ...
!Mesg0-> ?Mesg0-> !Ack0-> ?Ack0-> !Mesg1-> ?Mesg1-> !Ack1-> ?Ack1->

Exemple de vérification (LTL)

```
# Absence de blocage  -> Vrai
# Uppaal : A[] not deadlock
[] - dead;
# Un message emis est inevitablement recu  -> Faux
# Uppaal : p2 --> p6   ou "t1 --> t7"
[] (t1 => <> t7);
# Un message emis est inevitablement acquitte -> Faux
# Uppaal : "t1 --> t3"
[] (t1 => <> t3);
# Un message emis est inevitablement recu ou perdu  -> Vrai
[] (t1 => <> (t7 \/ t13));

# Place p2 du timeout ne peut pas être marquee en continu  -> Faux
# A la Uppaal : "E<>[] p2"
 - <> [] p2;
# Vivacité sous condition d'équité
(- [] <> (t13 \/ t14)) => ([] (t1 => <> t3));    ->  Vrai
Equité : S'il n'y a pas une infinité de pertes (mesg ou ack)
Si l'exécution est équitable alors
tout message émis sera inévitablement reçu
```

Calculs d'échéanciers : Producteur Consommateur



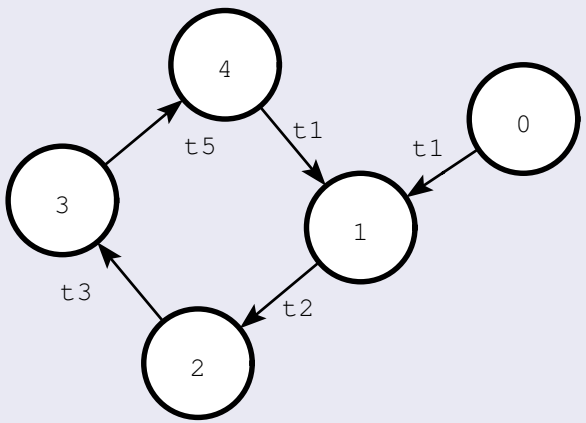
Légende

p2 : Message en transit,
p3 : Message arrivant dans le buffer du consommateur
p4/p5 : Buffer Vide/Plein (places complémentaires)
t_1 : Production
t_2 : Arrivée de la donnée chez le consommateur
t_3 : Bufferisation correcte (Buffer passe de vide à plein)
t_4 : Bufferisation Incorrecte (la donnée présente est écrasée)
t_5 : Réception par le producteur
Suivant les $\neq$ temporisations Bufferisation Correcte/Incorrecte

Version 1 : Producteur lent t1 [7,8]

Reseau et Graphe de Classes

```
tr Erreur [0,0] p3 p5?1 ->
tr t1 [7,8] p1 -> p1 p2
tr t2 [2,3] p2 -> p3
tr t3 [0,0] p3 p4 -> p5
tr t5 [0,2] p5 p6 -> p4 p6
pl p1 (1)
pl p4 (1)
pl p6 (1)
```

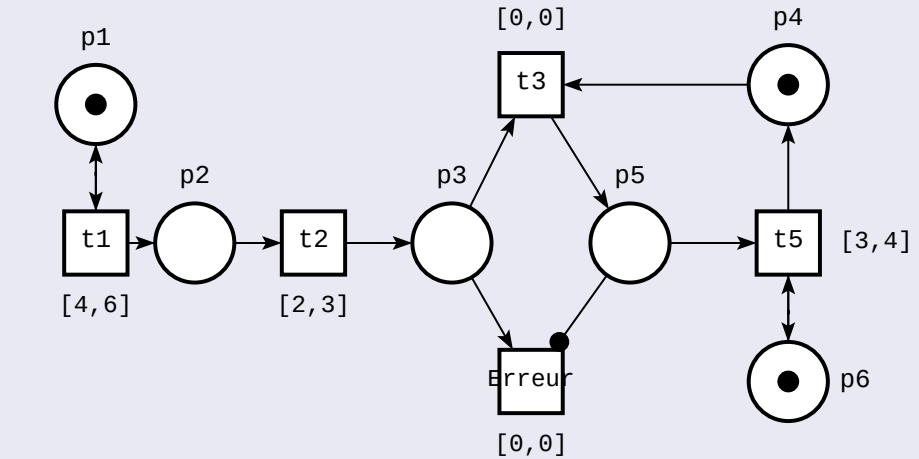


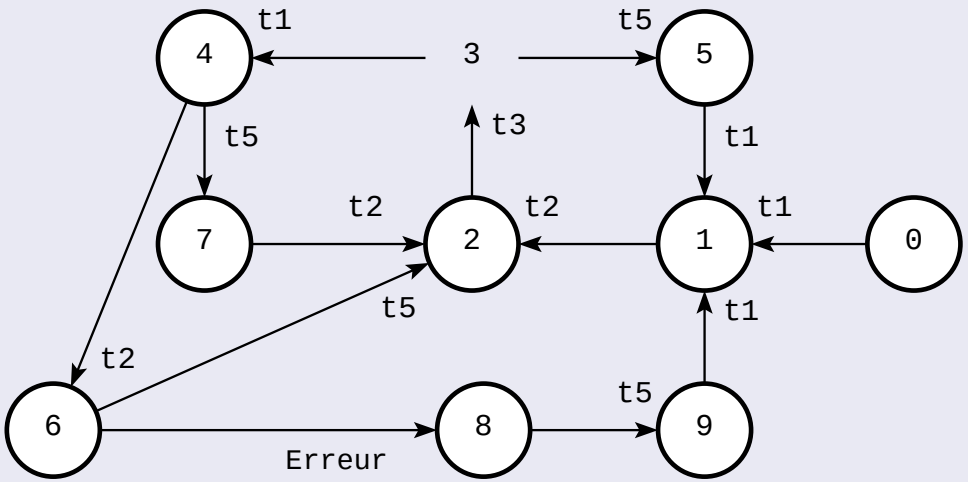
A la main : Erreur est morte (i.e. [] - Erreur $->$ True)

```
REACHABILITY ANALYSIS ---------------
bounded 5 classe(s), 5 transition(s)
class 0
    marking
        p1 p4 p6
    domain
        7 <= t1 <= 8
class 1
    marking
        p1 p2 p4 p6
    domain
        7 <= t1 <= 8
        2 <= t2 <= 3
class 2
    marking
        p1 p3 p4 p6
    domain
        4 <= t1 <= 6
        0 <= t3 <= 0

class 3
    marking
        p1 p5 p6
    domain
        4 <= t1 <= 6
        0 <= t5 <= 2
class 4
    marking
        p1 p4 p6
    domain
        2 <= t1 <= 6
```

Version 2 : Producteur t1 [4,6] & Consommateur t5 [3,4]

Reseau et Graphe de Classes

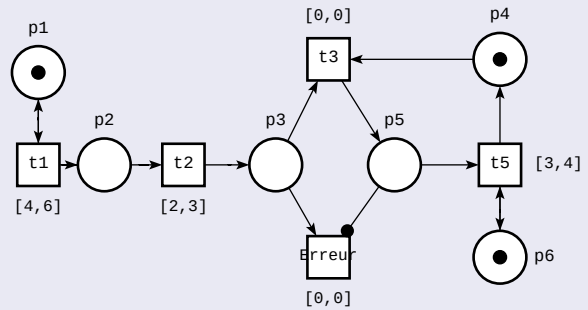


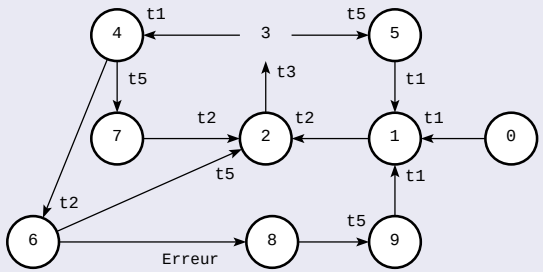



```
class 0
    marking
        p1 p4 p6
    domain
        4 <= t1 <= 6
class 1
    marking
        p1 p2 p4 p6
    domain
        4 <= t1 <= 6
        2 <= t2 <= 3
class 2
    marking
        p1 p3 p4 p6
    domain
        1 <= t1 <= 4
        0 <= t3 <= 0
class 3
    marking
        p1 p5 p6
    domain
        1 <= t1 <= 4
        3 <= t5 <= 4
class 4
    marking
        p1 p2 p5 p6
    domain
        4 <= t1 <= 6
        2 <= t2 <= 3
        0 <= t5 <= 3
class 5
    marking
        p1 p4 p6
    domain
        0 <= t1 <= 1
class 6
    marking
        p1 p3 p5 p6
    domain
        0 <= Erreur <= 0
        1 <= t1 <= 4
        0 <= t5 <= 1
        t5 - t1 <= -1
class 7
    marking
        p1 p2 p4 p6
    domain
        1 <= t1 <= 6
        0 <= t2 <= 3
        t1 - t2 <= 4
        t2 - t1 <= -1
class 8
    marking
        p1 p5 p6
    domain
        1 <= t1 <= 4
        0 <= t5 <= 1
        t5 - t1 <= -1
class 9
    marking
        p1 p4 p6
    domain
        1 <= t1 <= 4
```

Calcul d'échéanciers

Exemple





Invocation de Selt

```
selt pcbug.ktz -f
"[] - Erreur;" -p
FALSE
  state 0: p1 p4 p6
  -t1->
  state 1: p1 p2 p4 p6
  -t2->
  state 2: p1 p3 p4 p6
  -t3->
  state 3: p1 p5 p6
  -t1->
  state 5: p1 p2 p5 p6
  -t2->
  state 7: p1 p3 p5 p6
  -Erreur->
  state 10: p1 p5 p6
  [accepting all]
```

Selt : Détermine le scenario

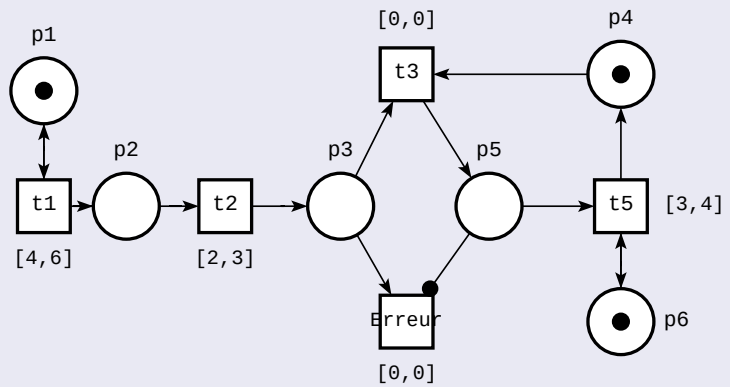
```
selt pcbug.ktz
-f "[] - Erreur"
-S erreur.scn

# [] - Erreur |- FALSE
t1 t2 t3 t1 t2 Erreur
```

Plan : Calcule l'échéancier (en absolu)

```
plan erreur.scn pcbug.ndr -S -abs
# -> Echeancier en dates absolues
t1$4 t2$6 t3$6 t1$8 t2$10 Erreur$10
```

Autres options de Plan

Réseau



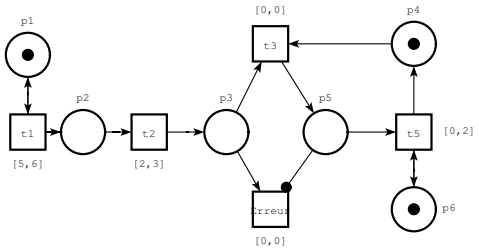
Echéancier en dates relatives

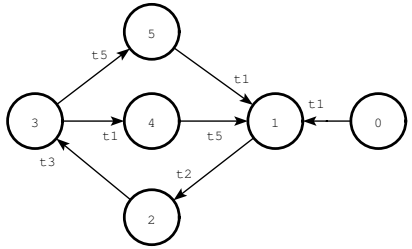
```
plan erreur.scn pcbug.ndr -S

t1@4 t2@3 t3@0 t1@1
     t2@2 Erreur@0
```

Ensemble de tous les échéanciers

```
plan cex.scn pc2.ndr

t1@x0 t2@x1 t3@x2 t1@x3
        t2@x4 Erreur@x5
where
4 <= x0 <= 6
----------------
2 <= x1 <= 3
2 <= x1 + x2 <= 3
0 <= x2 <= 0
4 <= x1 + x2 + x3 <= 6
1 <= x3 <= 4
0 <= x1 + x2 + x3 + x4
3 <= x3 + x4 <= 4
2 <= x4 <= 3
0 <= x1 + x2 + x3 + x4 + x5
3 <= x3 + x4 + x5 <= 4
```

Version 3 : Consommateur t5 [0,2] & Producteur t1 [3,4]





Vérification

Il peut y avoir deux productions (t1) consécutives sans qu'il y ait eu une consommation (t5) entre. 3 traductions : ϕ_0, ϕ_1 et ϕ_2.
ϕ_0 : $[](p2 => -(-t5 U t1))$
ϕ_1 : $[](t1 => -(()(-t5 U t1)))$
ϕ_2 : $[] - (p2 \wedge p5)$
Réponse attendue faux : le contre-exemple est une séquence comportant deux occurences de t1 sans occurence de t5 entre.

Script pour le vérifier

Génération de l'espace d'états : tina pcv3.ndr pcv3.ktz -s 0
– `"-s 0"` car le reseau atemporel sous-jacent est non borné
Vérification de la propriété et génération du contre-exemple.

```
selt pcv3.ktz -f "[](p2 => - (-t5 U t1))" -p -S scen2.scn
```

Génération de l'échéancier associé `plan scen2.scn pcv3.ndr -S -abs`

```
t1$5 t2$8 t3$8 t1$10
```

Outline

Introduction à la seconde Partie

Objectifs : Aborder des aspects non encore traités

1. Composition & Communication
2. Données (séquentiel)
3. Communication & Données (Concurrent & Séquentiel : pb associés)
4. Temporisation via des Horloges
   $\mapsto$ Alternative & Complément aux approches déja vues

Formalisme (Petri) vers un langage de Processus (UPPAAL)

Communication explicite : Synchronisation par Signal et Variables partagées
Manipulation des données : types de données, structures, ..
Communication avec passage de valeurs
Temporisation via des Horloges

Outil Support : UPPAAL (http ://www.uppaal.com/)

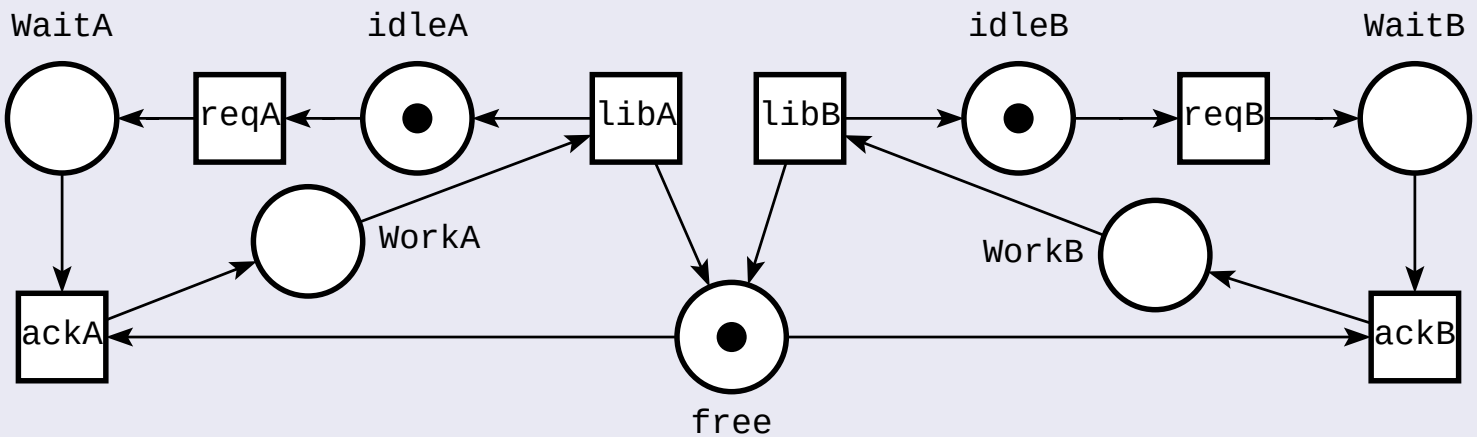
Environnement intégré pour la : modélisation, simulation, vérification de systèmes temporisés concurrents/réactifs

Exemple Introductif

Pb de Partage de ressources

2 clients partageant une ressource critique protégée par un sémaphore.

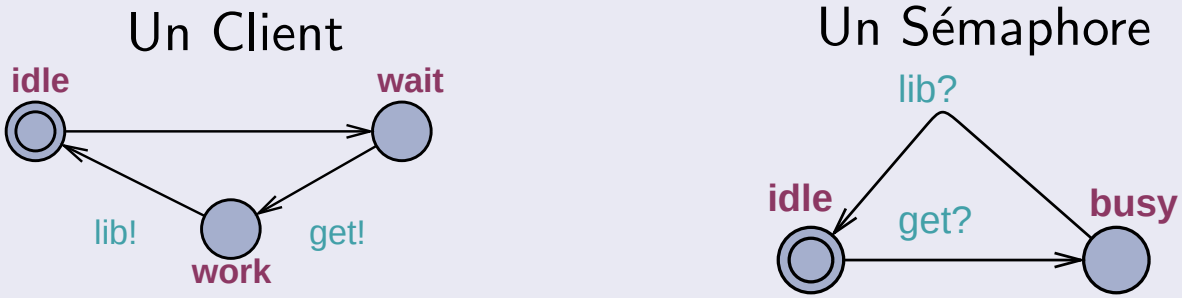
Modélisation **(standard)** en réseau de Petri



Modèle monolithique ! Où sont les clients, le sémaphore ?

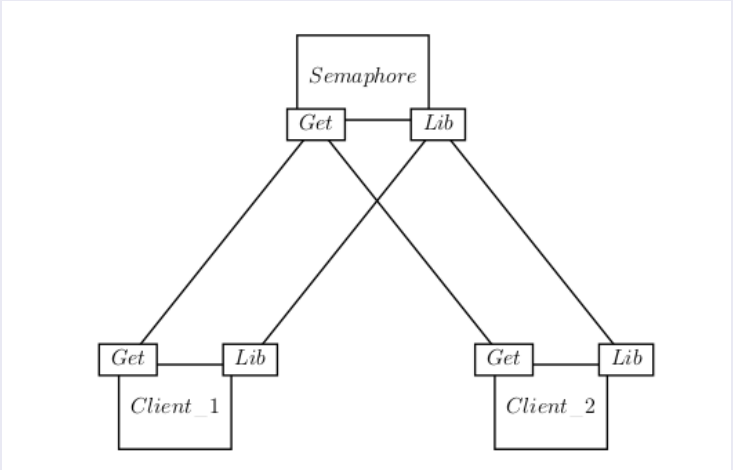
En langage de Processus (réseaux d'automates)

Approche modulaire : Un modèle par type de composant



Assemblage

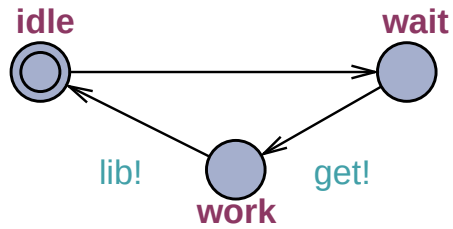
Système Global résulte de l'instanciation des différents composants et de leur mise en // (communication)

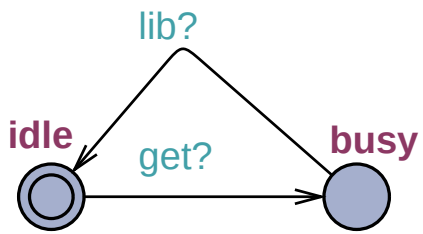


nb : On (re)verra aussi dans cette partie que l'on peut procéder à l'identique avec les Petri.

Version de base en UPPAAL sans paramètre formel

```
/* 2 canaux de communication */
chan   get, lib;

process Client(){
state idle, wait, work;
init idle;
trans idle -> wait{},
wait -> work{sync get!; },
work -> idle{sync lib!; }; }

process Semaphore(){
state idle, busy;
init idle;
trans idle -> busy{sync get?; },
busy -> idle{sync lib?; }; }
/* Instanciation des composants */
cl1 := Client();
cl2 := Client();
sem := Semaphore();
/* Declaration du systeme */
system  cl1,cl2,sem;
```





Version avec process paramétrés

2 clients partageant le même sémaphore

```
chan   get, lib;

process Client(chan &c1,chan &c2)
{ state idle, wait, work;
init idle;
trans idle -> wait{},
wait -> work{sync c1!;},
work -> idle{sync c2!;};}

process Semaphore(chan &c1,chan &c2
{ state idle, busy;
init idle;
trans idle -> busy{sync c1?; },
busy -> idle{sync c2?; };}

cl1 := Client(get,lib);
cl2 := Client(get,lib);
sem := Semaphore(get,lib);

system  cl1,cl2,sem;
```

3 clients, 2 sémaphores, ...

```
chan get1,lib1,get2,lib2;

process Client(...)
{Sans changement}

process Semaphore(...)
{Sans changement}

cl1 := Client(get1,lib1);
cl2 := Client(get2,lib2);
cl3 := Client(get1,lib1);
sem1 := Semaphore(get1,lib1);
sem2 := Semaphore(get2,lib2);

system cl1,cl2,sem1,cl3,sem2;
```

---

cl1 et cl3 partagent sem1 tandis que cl2 utilise seul sem2

system = ((cl1 * sem1 * cl3) // (cl2 * sem2))

Composition/Produits d'Automates

Définition Automate

nb : Automate au sens machine à états et non accepteur de langage

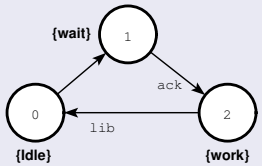
Un automate $=_{Def} < \mathcal{S}, \mathcal{L}, \mathcal{T} >$ où

1. $\mathcal{S}$ est un ensemble fini d'états ("locations"). état initial : s_{init}
2. $\mathcal{L}$ est un ensemble fini de labels
3. $\mathcal{T}$ est un ensemble fini de transitions
   Chaque transition $t \in \mathcal{T}$ est définie par un tuple $< s, l, s' >$ où :
   1. s ($resp$ s') $\in \mathcal{S}$: représentent la source et la cible de la transition.
   2. l est le label de la transition (une transition peut ne pas avoir de label)

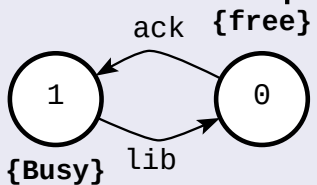
En Option et à des fins de vérification : $\mathcal{P}$ est une fonction qui associe à chaque état un ensemble de variables propositionnelles.

Exemples

Automate Client



Automate Sémaphore



Synchronisation sur labels communs

Définition : Produit synchrone par fusion de transitions de même label

Soient $\mathcal{M}_i =_{Def} < \mathcal{S}_i, \mathcal{L}_i, \mathcal{T}_i >$, $i \in [1,2]$, deux automates avec $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$

$\mathcal{M} =_{Def} \mathcal{M}_1 \times \mathcal{M}_2$, l'automate produit ($\mathcal{M} = < \mathcal{S}, \mathcal{L}, \mathcal{T} >$) est défini ainsi :

1. $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$
2. $(s_{01}, s_{02}) \in \mathcal{S} \subset \mathcal{S}_1 \times \mathcal{S}_2$,
3. $\mathcal{S}$ et $\mathcal{T}$ sont les $+$ petits ensembles vérifiant :

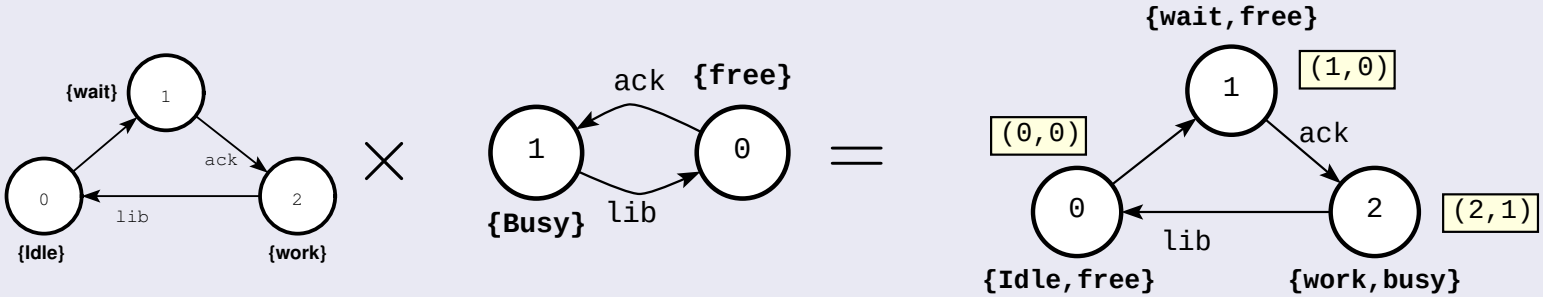
$$\text{Si} \begin{cases} (s_1, s_2) \in \mathcal{S} \\ \text{et } (s_1, l, s_1') \in \mathcal{T}_1 \\ \text{avec } l \notin \mathcal{L}_1 \cap \mathcal{L}_2 \end{cases} \quad \text{Alors} \begin{cases} ((s_1, s_2), l, (s_1', s2)) \in \mathcal{T} \\ \text{et } (s_1', s2) \in \mathcal{S} \end{cases}$$

$$\text{Si} \begin{cases} (s_1, s_2) \in \mathcal{S} \\ \text{et } (s_2, l, s_2') \in \mathcal{T}_2 \\ \text{avec } l \notin \mathcal{L}_1 \cap \mathcal{L}_2 \end{cases} \quad \text{Alors} \begin{cases} ((s_1, s_2), l, (s_1, s_2')) \in \mathcal{T} \\ \text{et } (s_1, s_2') \in \mathcal{S} \end{cases}$$

$$\text{Si} \begin{cases} (s_1, s_2) \in \mathcal{S} \\ \text{et } (s_1, l, s_1') \in \mathcal{T}_1 \\ \text{et } (s_2, l, s_2') \in \mathcal{T}_2 \\ (l \in \mathcal{L}_1 \cap \mathcal{L}_2) \end{cases} \quad \text{Alors} \begin{cases} ((s_1, s_2), l, (s_1', s_2')) \in \mathcal{T} \\ \text{et } (s_1', s_2') \in \mathcal{S} \end{cases}$$

Produit Synchrone (suite)

Exemple #1 : Client × Sémaphore



Propriété : Le produit de deux automates est un automate

nb : Le produit d'automates est "exploratoire" tandis que celui des Petri était analytique

Exemple #2 : Passage à niveau

Spécification

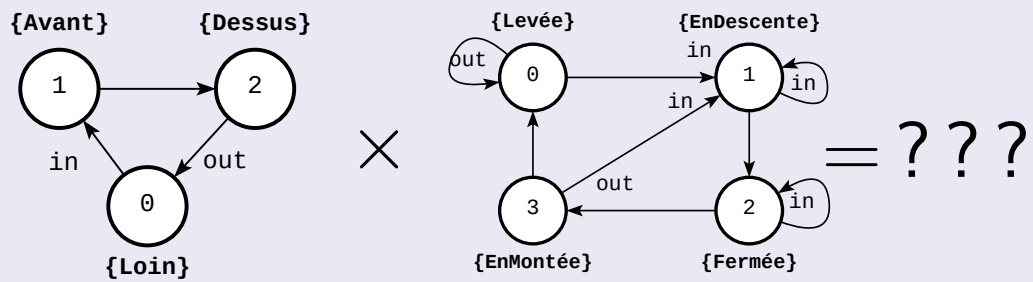
Train : Alternativement loin, avant et sur le passage à niveau
Barrière : Alternativement levée, en descente, fermée, en montée

Synchronisation entre le train et la barrière via deux portes (canaux, signaux) : **in** et **out**

in lorsque le train approche, la barrière commence à descendre, ...

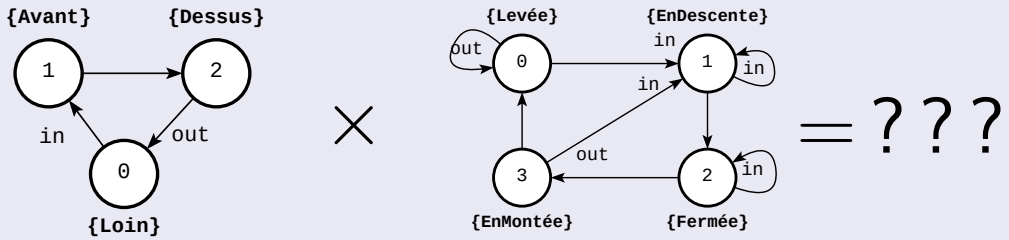
out lorsque le train quitte le passage à niveau, la barrière commence à se lever, ...

Composants : Train & Barrière

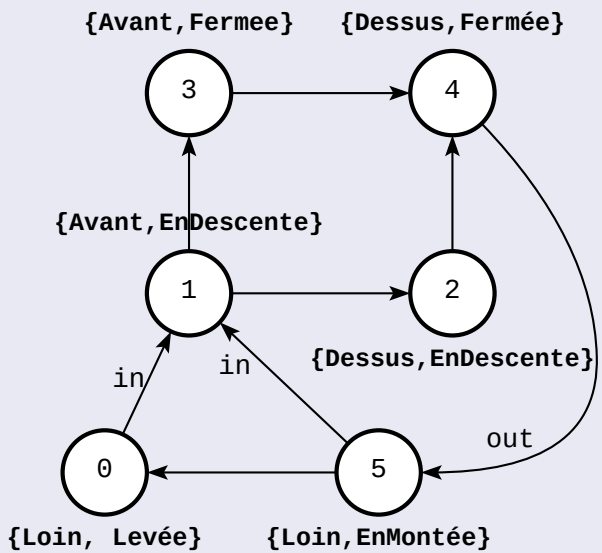


Exemple #2 : Passage à niveau (fin)

Composants : Train & Barrière



Automate Produit



Analyse

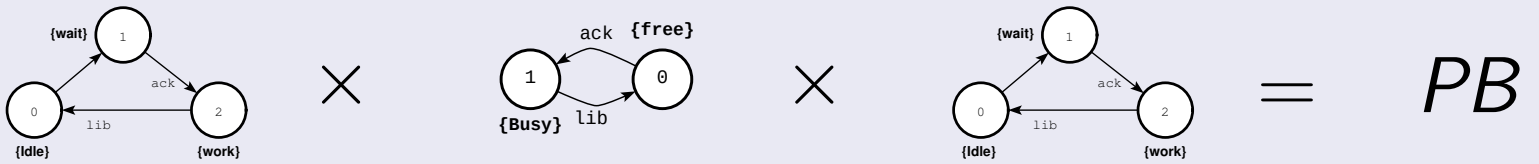
La possibilité d'atteindre un état où le train est Dessus et la barrière En Descente est problématique !

nb : On ne peut guère faire mieux sans modéliser le temps ! (cf cours Modèles temporels)

Dans ce même état, le train doit attendre que la barrière soit fermée pour qu'il puisse quitter le passage à niveau ! Dans la barrière, il manque une transition entre 1 et 3 avec une synchro sur out

Produit Synchrone : Synchro sur labels complémentaires

Limitation des ports non orientés

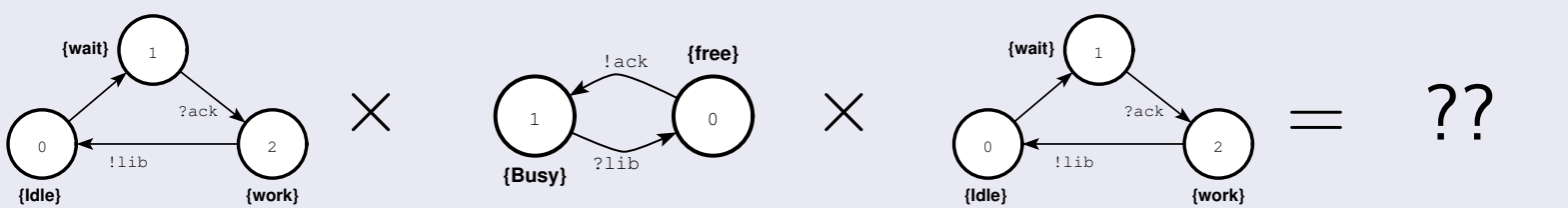


Labels (ports, portes, canaux, ...) orientés

Labels de synchronisation : $l \in$ **Sync** ssi $l = ?\ g$ *ou* $l = !\ g$

Labels complémentaires (#)
$l_1 \# l_2$ ssi $(l_1 = ?\ g$ *et* $l_2 = !\ g)$ *ou* $(l_1 = ?\ g$ *et* $l_2 = !\ g)$

Avec des ports orientés



Produit Synchrone sur labels complémentaires

Produit (généralisé) associé

Soient k automates, $\mathcal{M}_i =_{Def} < \mathcal{S}_i, \mathcal{L}_i, \mathcal{T}_i >$, pour $i \in [1, k]$,

On note $\mathcal{M} =_{Def} \mathcal{M}_1 \times \mathcal{M}_2 \times \ldots \times \mathcal{M}_k$ l'automate produit
où $\mathcal{M} = < \mathcal{S}, \mathcal{L}, \mathcal{T} >$ est défini ainsi :

1. $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_k \cup \{\tau\}$
2. $(s_{01}, \ldots, s_{0k}) \in \mathcal{S} \subset \mathcal{S}_1 \times \ldots \times \mathcal{S}_k$,
3. $\mathcal{S}$ et $\mathcal{T}$ sont les $+$ petits ensembles vérifiant :

Evolution // (indépendante, asynchrone)

$$\text{Si } \begin{cases} (.., s_i, ..) \in \mathcal{S} \\ \text{et } (s_i, l, s_i') \in \mathcal{T}_i \\ \text{avec } l \notin Sync \end{cases} \quad \text{Alors } \begin{cases} ((.., s_i, ..), l, (.., s_i', ..)) \in \mathcal{T} \\ \text{et } (.., s_i', ..) \in \mathcal{S} \end{cases}$$

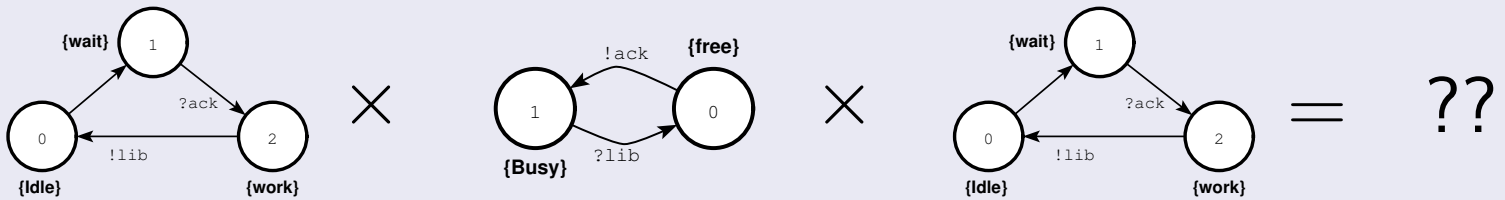
Evolution synchronisée

$$\text{Si } \begin{cases} (.., s_i, .., s_j, ...) \in \mathcal{S} \\ \text{et } (s_i, l_i, s_i') \in \mathcal{T}_i \\ \text{et } (s_j, l_j, s_j') \in \mathcal{T}_j \\ \text{avec } l_i \# l_j \end{cases} \quad \text{Alors } \begin{cases} (..., s_i', ..., s_j', ...) \in \mathcal{S} \\ \text{et} \\ ((., s_i, ., s_j, .), \tau, (., s_i', ., s_j', .)) \in \mathcal{T} \end{cases}$$

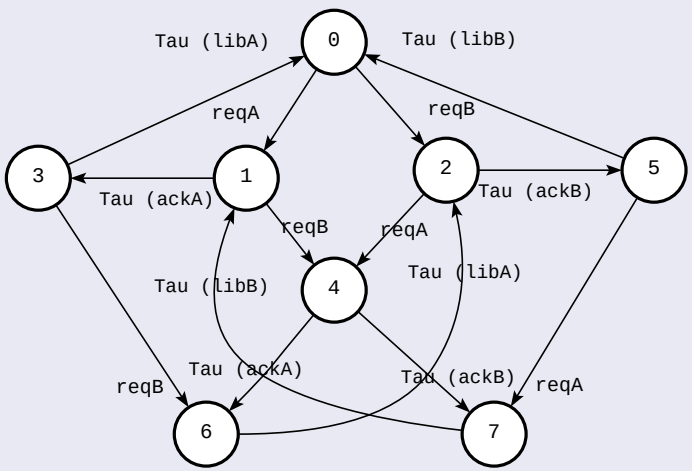
Les transitions issues d'une synchronisation sont "renommées" par τ.

Application sur l'exemple des 2 clients

Spécification du Système



Automate produit et Propositions d'états associés



0 $< IdleA, Free, IdleB >$
1 $< WaitA, Free, IdleB >$
2 $< IdleA, Free, WaitB >$
3 $< WorkA, Busy, IdleB >$
4 $< WaitA, Free, WaitB >$
5 $< IdleA, Busy, WorkB >$
6 $< WorkA, Busy, WaitB >$
7 $< WaitA, Busy, WorkB >$

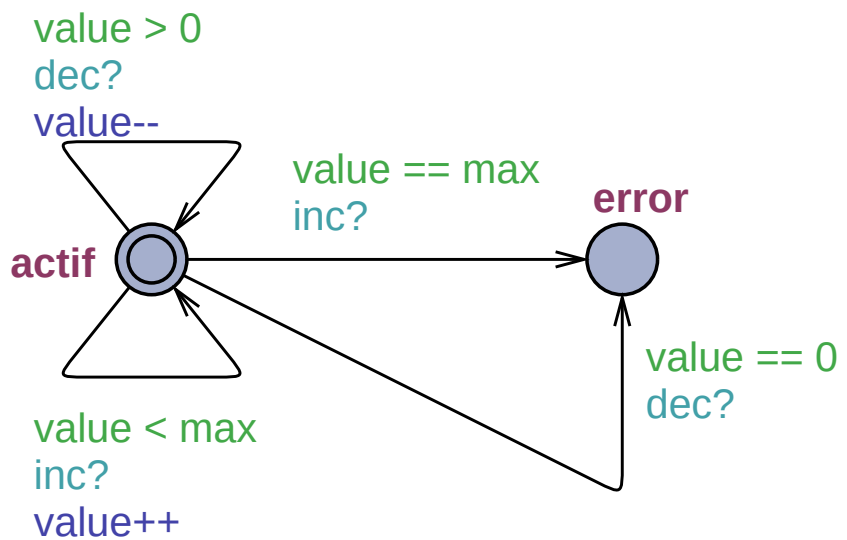
Prise en compte des Données

Plan

1. Intérêt/Nécessité de prendre en compte les données
   Automates $\mapsto$ Automates étendus
   (Petri Nets $\mapsto$ High-Level PN (RdP Colorés, RdP Prédicat/Transition, ...)
   1. données locales
   2. données partagées
2. Communication par signal + données partagées $\mapsto$ Communication avec passage de valeurs

Compteur avec données en Uppaal

```
const int cptmax =  5; /* Constante Globale */
chan   inc, dec;

process compteur(const int max, chan &plus, chan &moins){
int value:=0; /* variable locale */
state actif, error;
init actif;
trans  actif -> actif {guard value < max; sync inc?; assign value++; },
       actif -> actif{guard value > 0; sync dec?; assign value--; },
       actif -> error{guard value == max; sync inc?; },
       actif -> error{guard value == 0; sync dec?; }; }

process client(chan &c1, chan &c2)
{TBD}

cpt := compteur(cptmax,inc,dec);
cl := client(inc,dec);

system cpt,cl;
```



Variables partagées & Synchronisations : Sémantique UPPAAL

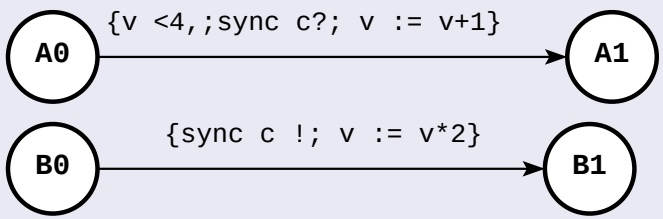
Sémantique UPPAAL

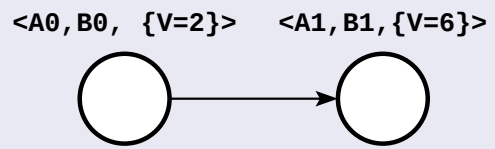
Dans le cas de transitions synchronisées,

1. On évalue en premier les gardes des deux transitions (l'ordre d'évaluation n'a pas d'importance)
2. On effectue les assignations de la transition correspondant à l'émission !
3. **Ensuite** les assignations de la transition correspondant à la réception ?

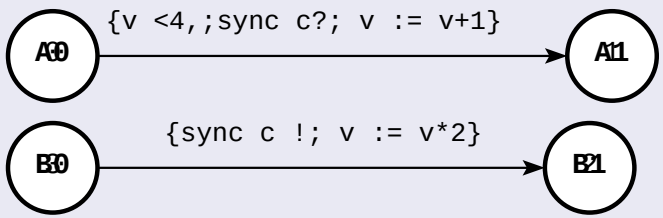
Exemples d'application

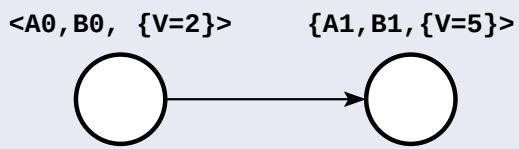
Ex #1





Ex #2

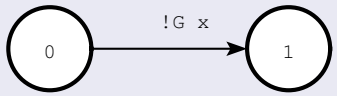




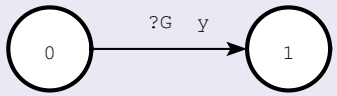
Communication Synchrone avec passage de valeurs

Objectif

Etre capable de communiquer en synchrone directement par valeurs et non seulement par signal : *Emetteur envoi une donnée X sur la porte G, le récepteur recoit cette donnée et la stocke dans une variable locale Y.*



Emetteur

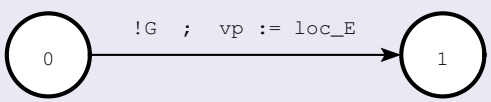


Récepteur

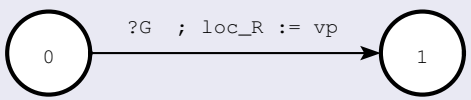
Codage en UPPAAL

Non directement expressible en UPPAAL mais la communication synchrone avec passage de valeurs est "simulable" "dès" que l'on dispose de la synchronisation pure et des variables partagées.

Principe : Une variable partagée vp. Deux variables locales (Loc_E et Loc_R) respectivement situées chez l'émetteur et chez le récepteur.



Emetteur



Récepteur

Avantage : *L'émetteur écrit dans la variable partagée, le récepteur la lit. Le transfert est fait sous le contrôle d'une synchronisation :* **garantie** *que la valeur lue est bien la* **bonne** *et qu'elle nous est* **bien adressée** *!*

Compteur amélioré

```
int shared_value := 0;
const int  cptmax  := 6;
chan  g;

process compteur(const int max, chan &in){
int value:=0;
state actif, frozen, error;

init actif;
trans actif -> frozen{sync in?; assign value := value + shared_value;},
frozen -> error{guard not ((value > 0) and (value < max)); },
frozen -> actif{guard ((value > 0) and (value < max)); };}

process client(const int dec,const int inc, chan &out){
state actif;
init actif;
trans actif -> actif{sync out!; assign shared_value := inc; },
      actif -> actif{sync out!; assign shared_value := dec; };}

cpt := compteur(cptmax,g);
cl := client(-1,2,g);
system cpt,cl;
```

Outline

Automates Temporisés

Extension temporelle des automates [Alur et Dill 1994]

Information quantitative
sur l'écoulement du temps $\mapsto$ **Horloges explicites**
dans les automates

Par la suite, extensions des Algèbres de Processus
En particulier CCS, LOTOS (ET-Lotos, RT-Lotos, ...)

Modèle "compositionnel"

Automates communicants temporisés
Wrt RdP souvent considéré (à tort) comme monolithique
mais difficile de temporiser les interactions.

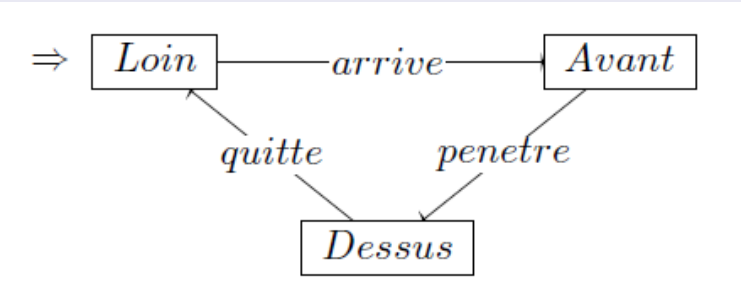
Outils

UPPAAL `www.uppaal.org`
KRONOS `http://www-verimag.imag.fr`

Ingédients : Automates, Horloges

Automate

Base : Etats/Locations & Transitions
+ Données & Gardes sur les transitions
+ Etiquettes pour la composition



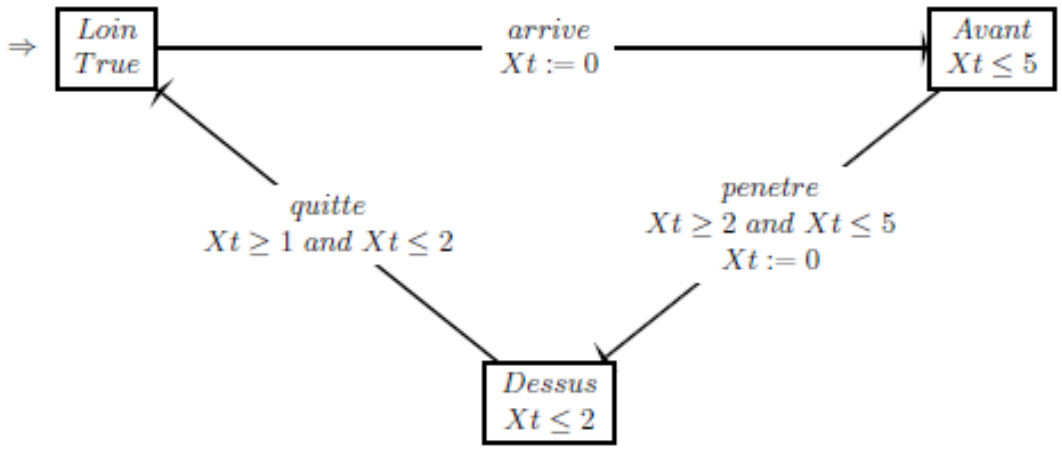
Horloges

$\mathcal{X}$ un ensemble fini d'horloges. Une valuation v est une fonction qui assigne une valeur $\in \mathbb{R}^*$ à chaque horloge. On note $\nu_{\mathcal{X}}$, l'ensemble des valuations d'horloge.
Pour $\delta \in \mathbb{R}^*, v \in \nu_{\mathcal{X}}$, on note $v' =_{Not} v + \delta$, la valuation v'
où : $v'(x) =_{Def} v(x) + \delta, \forall x \in \mathcal{X}$

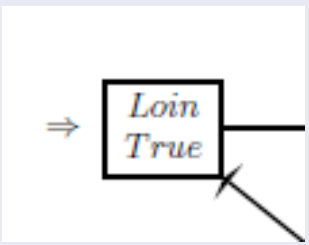
Contraintes et Opérations sur les horloges

Contraintes : On note $\Psi_{\mathcal{X}}$ l'ensemble des prédicats sur $\mathcal{X}$ défini comme une conjonction d'atomes de la forme $x \# c$ ou $x - y \# c$
avec $\# \in \{<, \leq, >, \geq, =\}$ et c une constante entière.
Opérations : Mise à zéro (affectation par une constante)

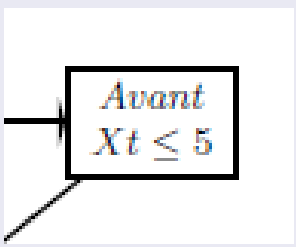
Exemple d'automate temporisé à une horloge : Xt



Etats + Invariants temporels

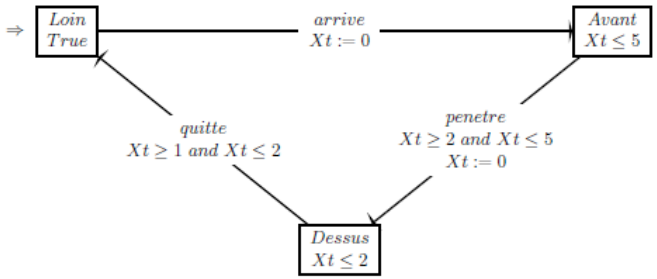


Etat initial ($\Leftarrow$)
Loin : Proposition d'état
True : Invariant temporel
On peut rester indéfiniment longtemps Loin

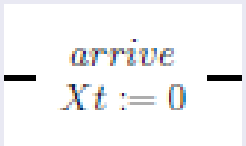


Avant : Proposition d'état
$Xt \leq 5$: Invariant temporel
On ne peut pas être Avant au-delà de 5

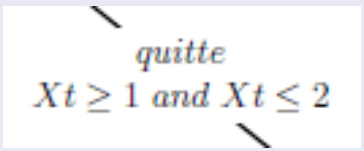
Exemple d'automate temporisé à une horloge : Xt



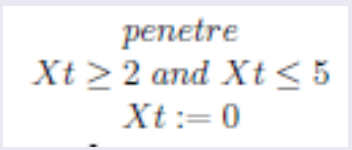
Transitions + Gardes temporelles et Remises à Zéro des horloges



arrive : label (synchronisation)
Remise à zéro de l'horloge Xt



quitte : label (synchronisation)
$XT \geq 1$ *and* $Xt \leq 2$: garde temporelle
La transition n'est franchissable que pour des valeurs de $Xt \in [1, 2]$



penetre : label (synchronisation)
$XT \geq 2$ *and* $Xt \leq 5$: garde temporelle
La transition n'est franchissable que pour des valeurs de $Xt \in [2, 5]$
Xt est remise à zéro lors du tir

Bien sûr possible d'avoir des transitions sans aucune caractéristique temporelle

Définition : Automate Temporisé

Automate temporisé : $\mathcal{M} =_{Def} < \mathcal{S}, \mathcal{X}, \mathcal{L}, \mathcal{T}, \mathcal{I}, \mathcal{P} >$ où

1. $\mathcal{S}$ est un ensemble fini de "locations". Location initiale : s_{init}
2. $\mathcal{X}$ est un ensemble fini d'horloges
3. $\mathcal{L}$ est un ensemble fini de labels
4. $\mathcal{T}$ est un ensemble fini de transitions

   Chaque transition $t \in \mathcal{T}$ est définie par un tuple $< s, l, \psi, \rho, s' >$ où :

   1. s ($resp$ s') $\in \mathcal{S}$: représentent la source et la cible de la transition.
   2. l est le label de la transition
   3. $\psi \in \Psi_{\mathcal{X}}$ représente la garde (données et temporelle $\in \Psi_{\mathcal{X}}$) de sensibilisation de la transition
   4. ρ est l'assignation permettant de spécifier les réinitialisations d'horloges consécutives au tir de la transition.
5. $\mathcal{I}$ est une fonction qui associe un invariant temporel à chaque location s, i.e., une condition $\mathcal{I}(s) \in \Psi_{\mathcal{X}}$.
6. $\mathcal{P}$ est une fonction qui associe à chaque location un ensemble de variables propositionnelles.

Sémantique

Etat d'un automate temporisé

Vecteurs d'horloges : $v : \mathcal{X} \mapsto \mathbb{R}^*$
$\nu_{\mathcal{X}} = \mathbb{R}^{*\mathcal{X}}$ ensemble de tous les vecteurs d'horloges

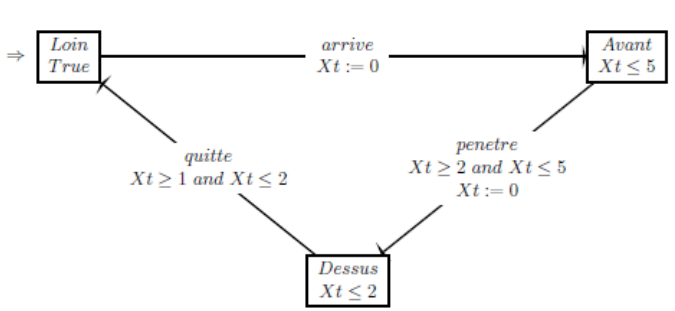
Etat temporel $\mathcal{M}$: $(s, v) \in S \times \nu_{\mathcal{X}}$ tq $v \models \mathcal{I}(s)$.
On note $\mathcal{Q}$, l'ensemble de tous les états de $\mathcal{M}$
Location intiale (s_{init}, v_{init}) avec $v_{init}(x) = 0, \forall x \in \mathcal{X}$.

Transitions d'un automate temporisé

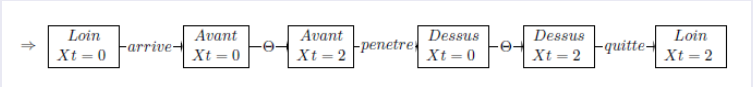
Soit $t =< s, l, \psi, \rho, s' > \in \mathcal{T}$
2 types de transitions sont considérées :

- Transitions **discrètes** : changement de location et mise à jour de certaines horloges : $(s, v) \xrightarrow{l} (s', v')$ si $v \models \psi$ et $v' = v[\rho]$
- Transitions d'**écoulement du temps** : Location inchangée, incrémentation uniformes des horloges. $(s, v) \rightarrow_\theta (s, v + \theta)$
  Ecoulement de temps régi par l'invariant temporel associé à l'état : $(s, v) \rightarrow_\theta (s, v + \theta)$ ssi $\forall \theta' \leq \theta, v + \theta' \models \mathcal{I}(s)$.

Exemples d'exécution

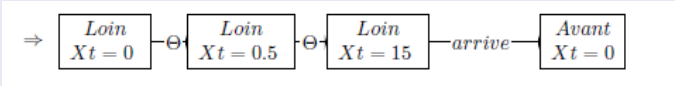


"Vitesse Maximale"



Exécution correspondant à un fonctionnement "au plus vite" de l'automate : dès qu'une transition discrète est possible alors elle doit être tirée - les écoulements de temps sont minimaux.

Délais arbitraires



Exécution avec des écoulements de temps "quelconques".

Espace d'états généralement infini !

Analyse nécessite de disposer d'une abstratcion finie.
Nb Abstractions discrètes vues pour les TPn fonctionnent
Graphe de Régions : Abstraction dense introduite spécifiquement pour les T.A

Compositon d'automates (rappel du cadre atemporel)

Définition du produit d'automates

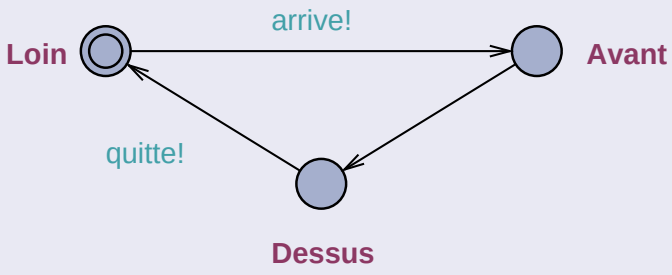
Soient $\mathcal{M}_i =_{Def} < \mathcal{S}_i, \mathcal{L}_i, \mathcal{T}_i, \mathcal{P}_i >$, pour $i \in [1,2]$,
deux automates atemporels avec : $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$

On note $\mathcal{M} =_{Def} \mathcal{M}_1 \times \mathcal{M}_2$ l'automate produit
où $\mathcal{M} = < \mathcal{S}, \mathcal{X}, \mathcal{L}, \mathcal{T}, \mathcal{I}, \mathcal{P} >$ est défini par :

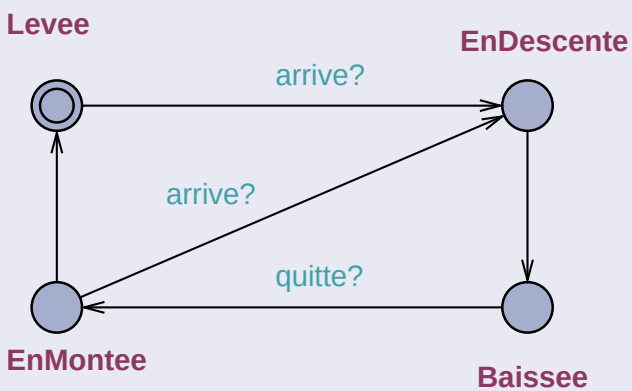
- $\mathcal{S} \subset \mathcal{S}_1 \times \mathcal{S}_2$, $\mathcal{L} \subset \mathcal{L}_1 \cup \mathcal{L}_2$
- $\mathcal{S}$ et $\mathcal{T}$ sont les + petits ensembles vérifiant $(s_{10}, s_{20}) \in \mathcal{S}$
  et pour $(s_1, s_2) \in \mathcal{S}$ et $< s_i, l_i, s_i' > \in \mathcal{T}_i \quad (i \in \{1,2\})$ alors
  1. Si $l_1 = l_2$ alors $< (s_1, s_2), l_i, (s_1', s_2') > \in \mathcal{T}$ et $(s_1', s_2') \in \mathcal{S}$
  2. Si $l_1 \neq l_2$ alors
     $< (s_1, s_2), l_1, (s_1', s_2) > \in \mathcal{T}$ et $(s_1', s_2) \in \mathcal{S}$
     $< (s_1, s_2), l_2, (s_1, s_2') > \in \mathcal{T}$ et $(s_1, s_2') \in \mathcal{S}$
- $\mathcal{P}((s_1, s_2)) =_{Def} \mathcal{P}(s_1) \cup \mathcal{P}(s_2)$

Passage à niveau (simplifié) : Composition train + barrière

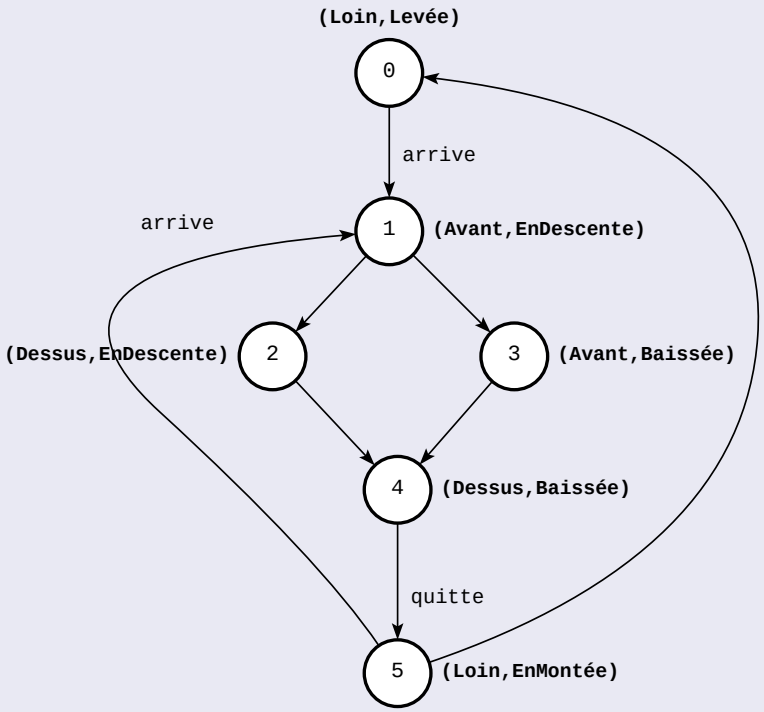
Train



Barrière



Composition



PB : Le train peut-etre engagé alors que la barrière est encore en descente

Compositons d'automates temporisés

Définition du produit d'automates temporisés

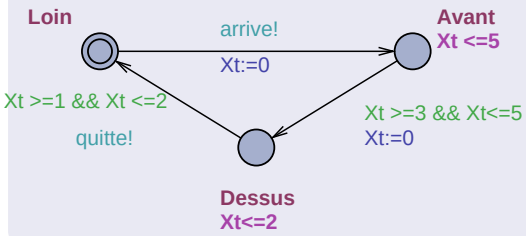
Soient $\mathcal{M}_i =_{Def} < \mathcal{S}_i, \mathcal{X}_i, \mathcal{L}_i, \mathcal{T}_i, \mathcal{I}_i, \mathcal{P}_i >$, pour $i \in [1,2]$,
deux automates temporisés avec :$\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset, \mathcal{X}_i \cap \mathcal{X}_j = \emptyset$

On note $\mathcal{M} =_{Def} \mathcal{M}_1 \times \mathcal{M}_2$ avec $\mathcal{M} = < \mathcal{S}, \mathcal{X}, \mathcal{L}, \mathcal{T}, \mathcal{I}, \mathcal{P} >$ défini par :

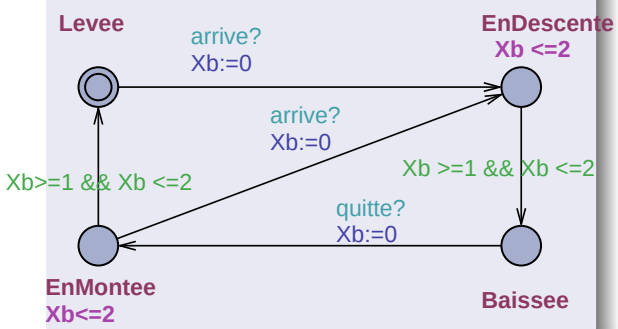
- $\mathcal{S} \subset \mathcal{S}_1 \times \mathcal{S}_2$, $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$, $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$
- $\mathcal{S}$ et $\mathcal{T}$ sont les $+$ petits ensembles vérifiant $(s_{10}, s_{20}) \in \mathcal{S}$
  et pour $(s_1, s_2) \in \mathcal{S}$ pour $< s_i, l_i, \psi_i, X_i, s'_i > \in \mathcal{T}_i \quad (i \in \{1,2\})$ alors
  1. Si $l_1 = l_2$ alors
     $< (s_1, s_2), l_i, \psi_1 \wedge \psi_2, X_1 \cup X_2, (s'_1, s'_2) > \in \mathcal{T}$ et $(s'_1, s'_2) \in \mathcal{S}$
  2. Si $l_1 \neq l_2$ alors
     $< (s_1, s_2), l_1, \psi_1, X_1, (s'_1, s_2) > \in \mathcal{T}$ et $(s'_1, s_2) \in \mathcal{S}$
     ET $< (s_1, s_2), l_2, \psi_2, X_2, (s_1, s'_2) > \in \mathcal{T}$ et $(s_1, s'_2) \in \mathcal{S}$
- $\mathcal{I}((s_1, s_2)) =_{Def} \mathcal{I}(s_1) \wedge \mathcal{I}(s_2)$
- $\mathcal{P}((s_1, s_2)) =_{Def} \mathcal{P}(s_1) \cup \mathcal{P}(s_2)$

Passage à niveau : version temporisée

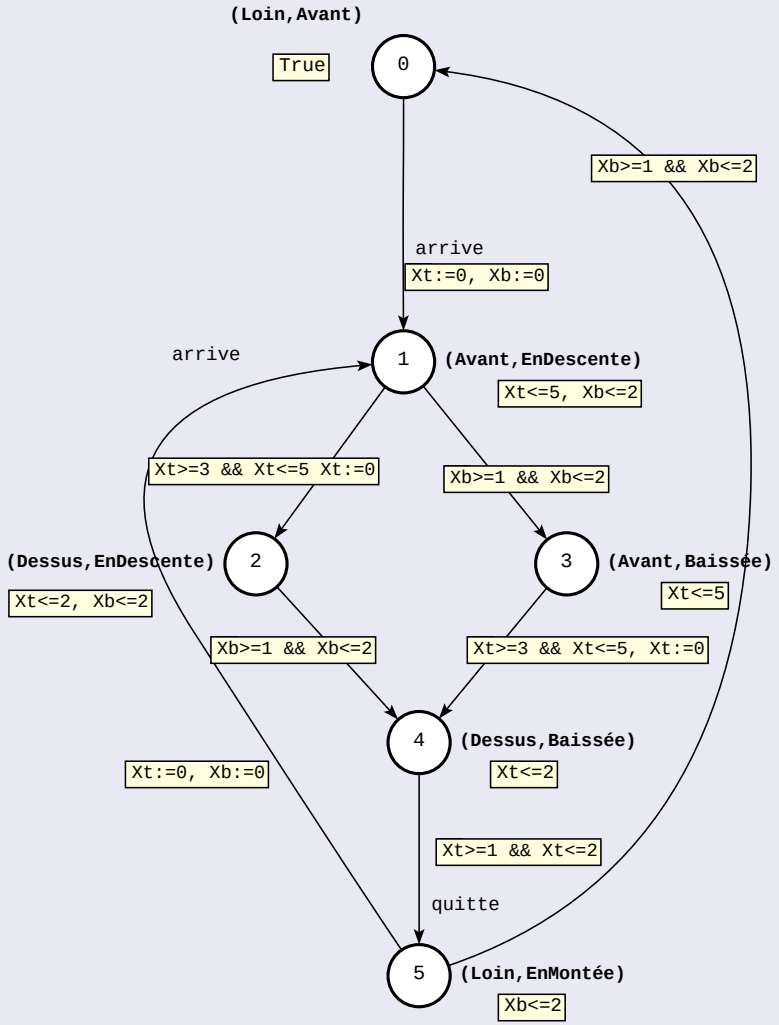
Train



Barrière



Composition



Pb résolu : en 1, $Xt \leq 2$ donc le train ne peut pas s'engager alors que la barrière n'est pas encore fermée.

Passage à niveau en Uppaal

Forme textuelle xta (1/2)

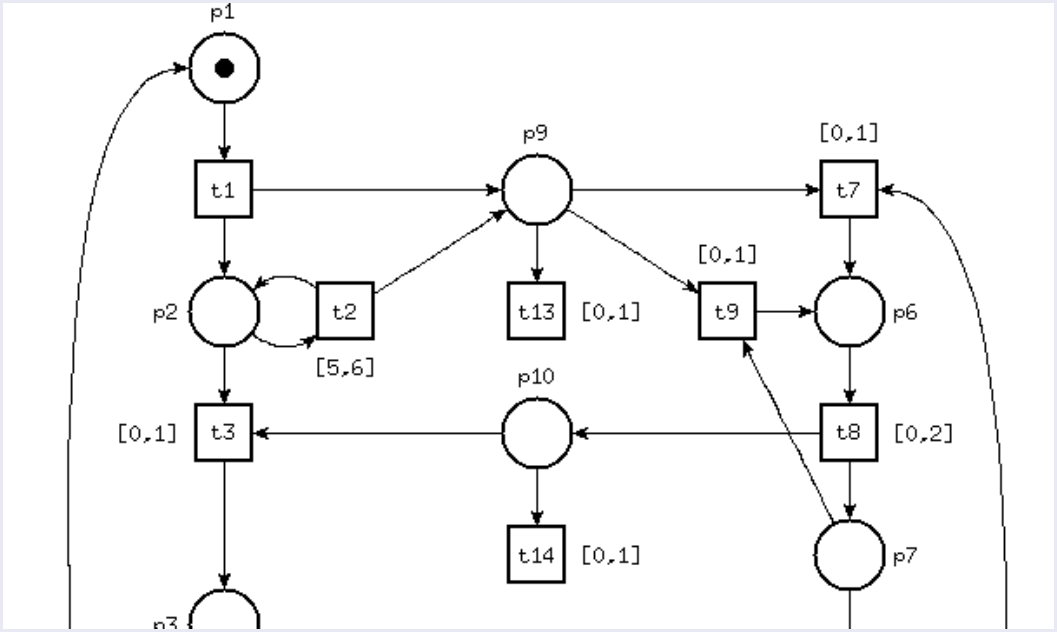
```
// Place global declarations here.
chan arrive,quitte;

process train() {
// Place local declarations here.
clock Xt;
state
    Dessus {Xt<=2},
    Avant {Xt <=5},
    Loin;
init Loin;
trans
    Loin -> Avant { sync arrive!; assign Xt:=0;  },
    Dessus -> Loin { guard Xt >=1 && Xt <=2; sync quitte!;  },
    /** Rentre */
    Avant -> Dessus { guard Xt >=3 && Xt<=5; assign Xt:=0;  };
}

.../...
```

Forme textuelle xta (2/2)

```
process barriere() {
clock Xb;
state
    EnMontee {Xb<=2},
    Baissee,
    EnDescente {Xb <=2},
    Levee;
init Levee;
trans
    EnMontee -> EnDescente { sync arrive?; assign Xb:=0;  },
    EnMontee -> Levee { guard Xb>=1 && Xb <=2;  },
    Baissee -> EnMontee { sync quitte?; assign Xb:=0;  },
    EnDescente -> Baissee { guard Xb >=1 && Xb <=2;  },
    Levee -> EnDescente { sync arrive?; assign Xb:=0;  };
}
// Place template instantiations here.
tr = train();
br = barriere();
// List one or more processes to be composed into a system.
system tr,br;
```

Protocole du Bit Alterné par des automates temporisés

1/2 Bit Alterné (rappel)



Transitions

t1 émission du message
t2 timeout et retransmission
t3 réception de l'Ack
t13 perte du message
t7 réception du message
t9 **re**-réception du message
t8 émission de l'Ack
t14 perte de l'Ack

Places

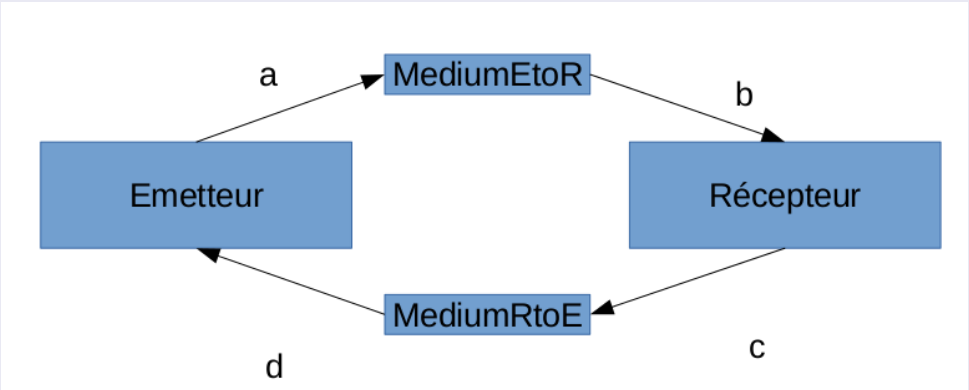
P1 Emetteur prèt émettre
P2 Emetteur en attente Ack
P9 Message en transit
P5 Récepteur prèt à recevoir msg bit 0
P6 Récepteur prèt à envoyer Ack
P7 Récepteur prèt à recevoir msg bit 0 **et** bit 1

Aternating Bit Proptocol en Uppaal (1/)

Architecture

3 Classes :	Recepteur, Emetteur, Medium
4 Instances :	1 émetteur, 1 récepteur, 2 instances de medium
4 canaux :	a, b, c,d (deux ports par composant)
4 Horloges :	1 pour chaque instance (donc pas d'horloge globale)

Topologie :



Alternating Bit Protocol en Uppaal (2/)

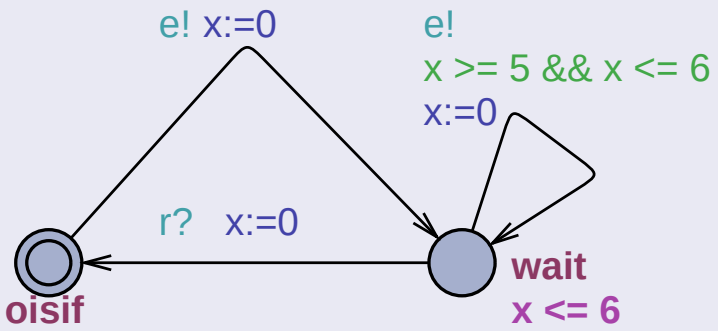
Déclarations, Classes et Instances

```
// Declarations globales : les canaux seulement.
// Les horloges seront declarees dans les process

chan ia,oa,ir,ob;

process sender(chan &e, chan &r){TBD}
process receiver(chan &r, chan &e){TBD}
process medium(chan &i, chan &o, const int fiable ){TBD}

// Declarations (et connexion) des instances

em = sender(ia,ob);
medemrec = medium(ia,oa,0);
rec  = receiver(oa,ir);
medrecem = medium(ir,ob,1);

// Declaration du systeme

system em,medemrec,medrecem,rec;
```

Alternating Bit Protocol en Uppaal (3/)

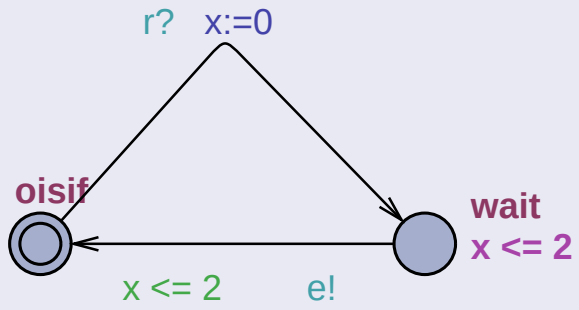
Emetteur

```
sender(chan &e,chan &r)
{clock x; ...
```



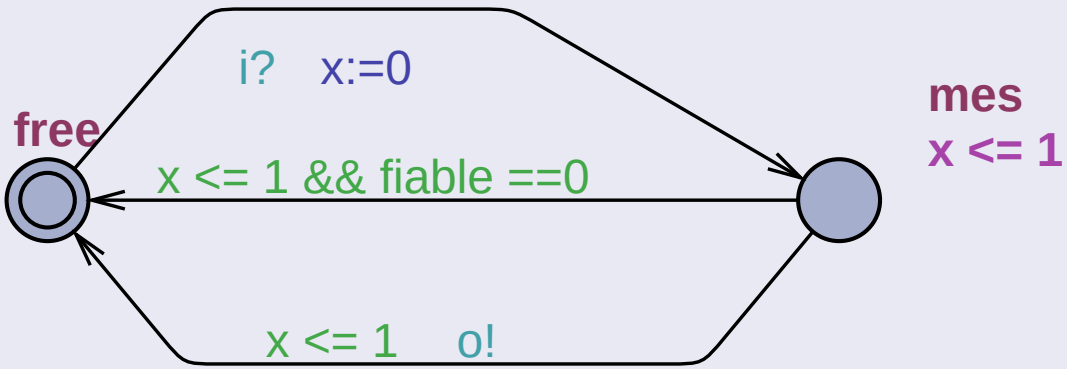
Récepteur

```
receiver(chan &r,chan &e)
{clock x;...
```



Medium

```
medium(chan &i,chan &o,const int fiable)
{clock x;...
```

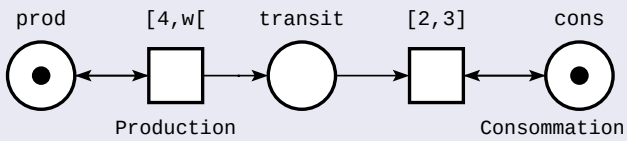


Alternating Bit Protocol : code xta

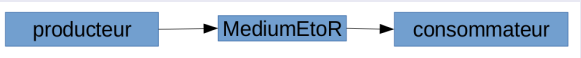
```
chan ia,oa,ir,ob;
process sender(chan &e, chan &r){
clock x;
state
    oisif,
    wait {x <= 6};
init oisif;
trans
    oisif -> wait { sync e!; assign x:=0;},
    wait -> wait { guard x >= 5 && x <= 6; sync e!; assign x:=0;},
    wait -> oisif { sync r?; assign x:=0;};
}
process receiver(chan &r, chan &e) {
clock x;
state
    oisif,
    wait {x <= 2};
init oisif;
trans
    oisif -> wait { sync r?; assign x:=0;  },
    wait -> oisif { guard x <= 2; sync e!;  };
}
process medium(chan &i, chan &o, const int fiable){
clock x;
state
    free,
    mes {x <= 1};
init free;
trans
    free -> mes { sync i?; assign x:=0;  },
    mes -> free { guard x <= 1 && fiable ==0;},
    mes -> free { guard x <= 1; sync o!;  };
}
em = sender(ia,ob);
medemrec = medium(ia,oa,0);
rec  = receiver(oa,ir);
medrecem = medium(ir,ob,1);
system em,medemrec,medrecem,rec;
```

Exercice : Modélisation du Producteur/Consommateur

Modèle Petri



Architecture

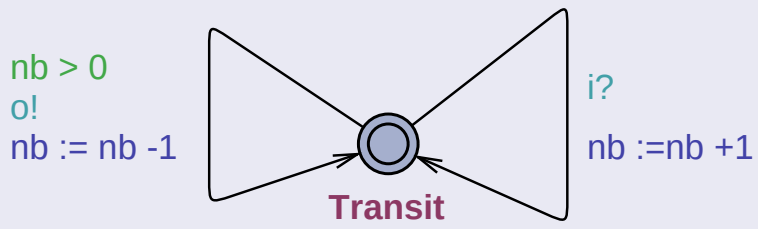
3 Classes :	Producteur, Medium , Consommateur
3 Instances :	1 producteur, 1 medium, 1 consommateur
2 canaux :	p, c (un port par composant sauf 2 pour le medium)
2 Horloges :	producteur et consommateur
Topologie :	

Déclarations, Classes et Instances

```
chan prod,cons ;
process Producteur(chan &p){clock Xc; ...}
process Consommateur(chan &c){...}
process medium(chan &i, chan &o){...}
P = Producteur(prod);
m = medium(prod,cons);
C = Consommateur(cons);
system P, m, C;
```

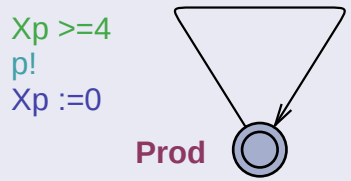
Exercice : Modélisation du Producteur/Consommateur

medium(chan &i,chan &o)



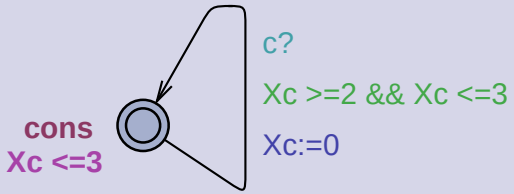
Producteur(chan &p)

{clock Xp; ...



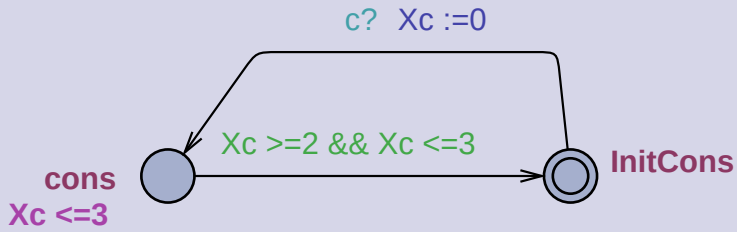
Quid du consommateur ?

V_0 : Conso(chan &c)



$\mapsto$ Blocage temporel !

V_1 : Conso(chan &c)



$\mapsto$ nb non bornée !

V_3 : Conso(urgent chan &c)

Comportement du consommateur inchangé

Canal urgent : Si une communication est "urgente" alors le temps ne peut pas s'écouler.

$\mapsto$ Comportement identique au Petri initial

Bilan partiel de la temporisation des automates

Modèle initial Alur et Dill

Horloges et Gardes temporelles

Urgence et blocage temporels

Invariants temporels

Temporisation des Interactions

Canaux urgents (Uppaal)

Graphe de Régions : une abstraction (dense) finie du comportement d'un automate temporisé

Restriction aux constantes d'horloges entières (ou rationnelles)

Restriction équivalente à celle utilisée dans les TPNs

Notations

- Pour $r \in \mathbb{R}^*$, on pose $r = \lfloor r \rfloor + \langle r \rangle$ où $\lfloor r \rfloor$ dénote sa partie entière (par défaut) et $\langle r \rangle$ sa partie fractionnaire.
- Pour un automate $\mathcal{A}$ et chaque horloge $x \in \mathcal{X}$, on note M_x la plus grande constante d'horloge pour x dans $\mathcal{A}$

Propriétés de base

1. Deux états $(s, v), (s, v')$ tels que v et v' ont la même partie entière sur chaque horloge et tels que leurs parties fractionnaires sont dans le même ordre présentent le même comportement.
2. Soit (s, v) un état et une horloge $x \in \mathcal{X}$ pour laquelle $v(x) > M_x$ alors pour tout v' tel que $v(c) = v'(c)$ si $c \neq x$ et $v'(x) > v(x)$, (s, v) et (s, v') présentent le même comportement.

Régions temporelles : Equivalence de vecteurs d'horloges

Région : Equivalence de vecteurs d'horloges

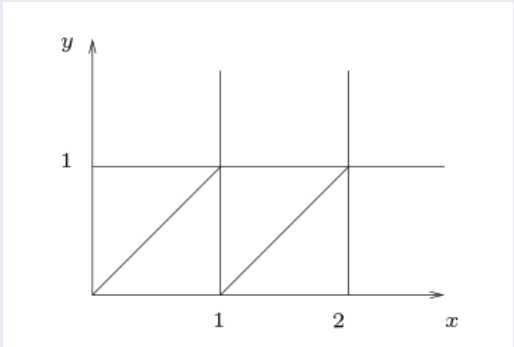
$v_1 \sim v_2$ ssi (1), (2) et (3)

1. $\forall x \in \mathcal{X} : (v_1(x) > M_x$ et $v_2(x) > M_x)$ ou $(\lfloor v_1(x) \rfloor = \lfloor v_1(x) \rfloor)$
2. Si $v(x) \leq M_x$ alors $(\langle v_1(x) \rangle = 0$ ssi $\langle v_1(x) \rangle = 0)$ $\forall x \in \mathcal{X}$
3. Si $v_1(x) \leq M_x$ et $v_1(y) \leq M_y$ alors $(\langle v_1(x) \rangle \leq \langle v_1(y) \rangle)$ ssi $(\langle v_2(x) \rangle \leq \langle v_2(y) \rangle)$ $\forall x, y \in \mathcal{X}$

Propriétés :

On appelle Région, les classes d'équivalence de $\sim$
$\sim$ est d'index fini, le nombre de régions est borné par $2^{|\mathcal{X}|}.|\mathcal{X}|!.(2M.2)^{|X|}$
Graphe de régions Quotient $(\nu_{\mathcal{X}} / \sim)$ est fini **explosif**
Le problème de l'accessibilité est donc décidable

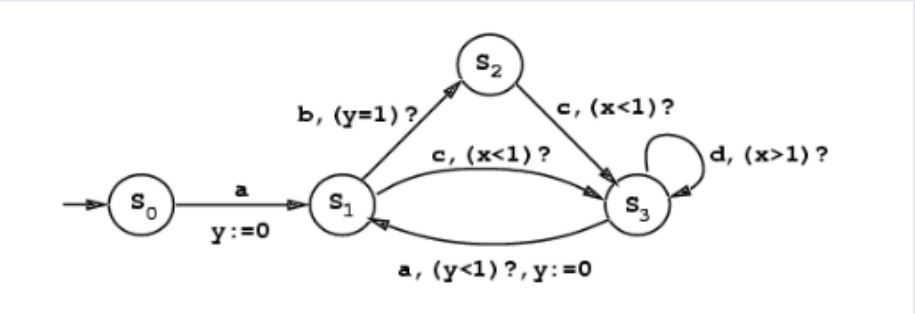
Exercice : Trouver les 28 régions pour $M_x = 2$ et $M_y = 1$

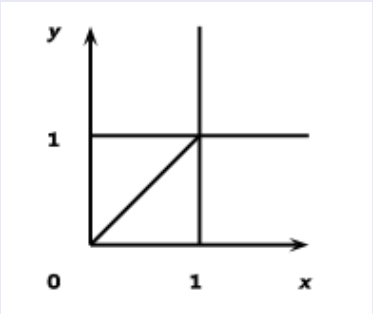


- Régions "ponctuelles" : 6
(ex (0.0))
- Segments ouverts : 14
(ex $0 < x = y < 1$)
- Régions ouvertes : 8
(ex $0 < x < y < 1$)

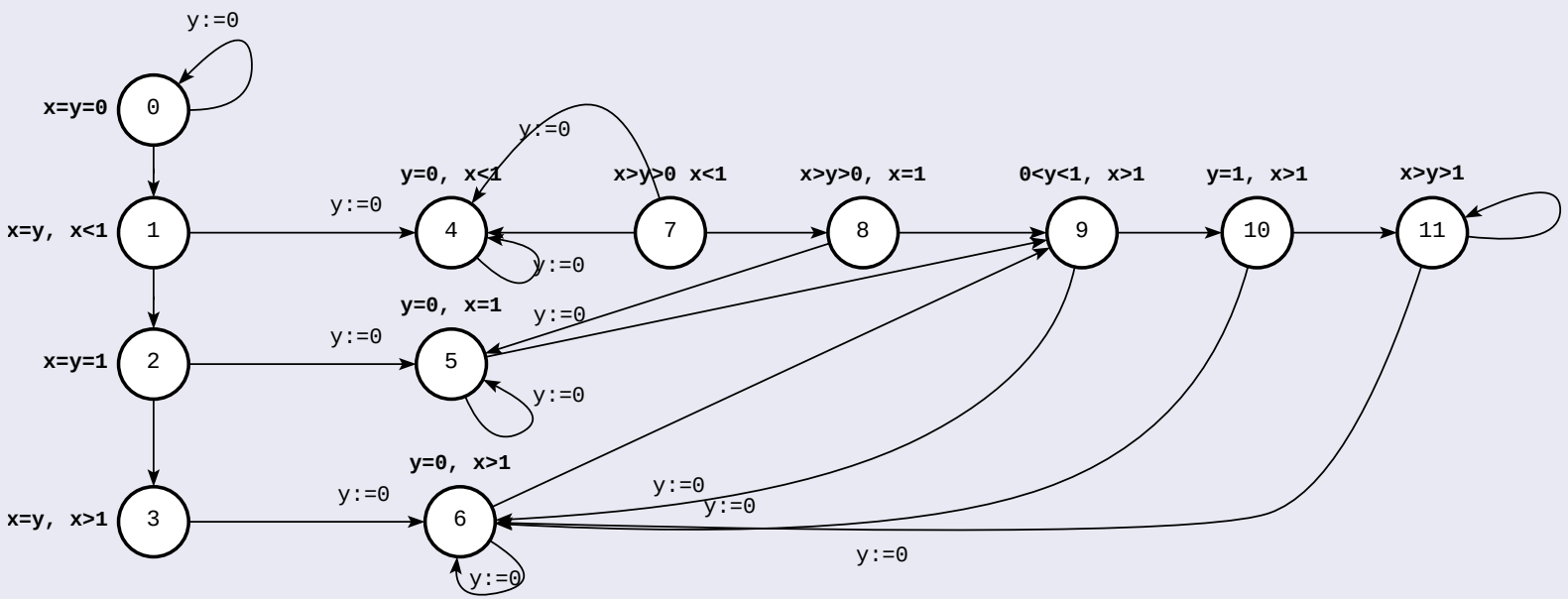
Automate des Régions (principe)

Automate exemple et régions géométriques



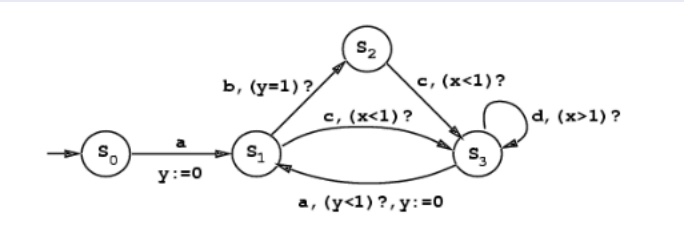


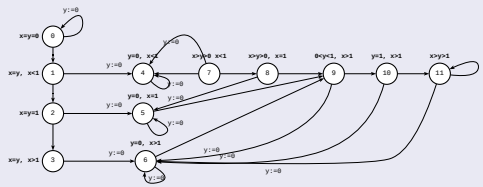
Automate des Régions



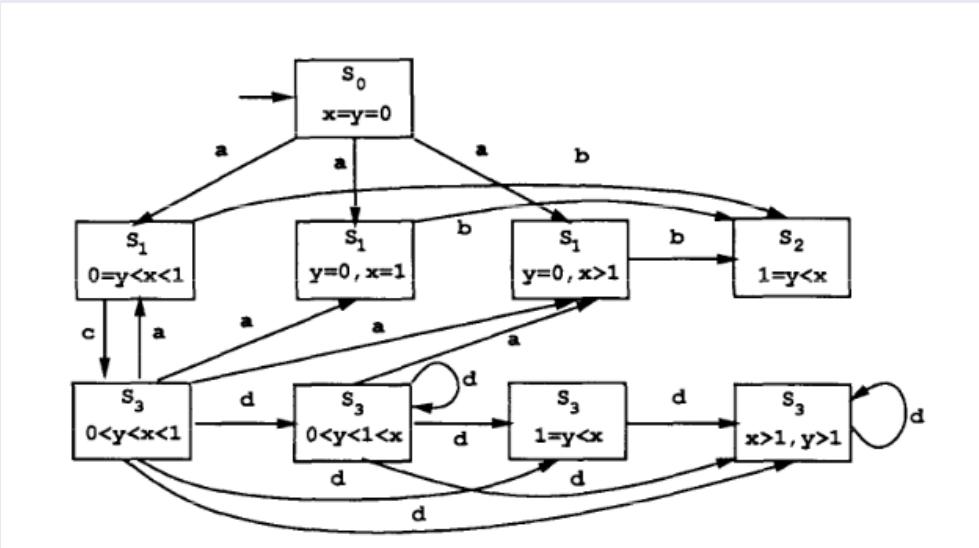
Graphe des Régions (exemple Alur Dill 1994)

Graphe des régions : Automate ⊗ Automate des régions





Graphe des régions



Remaques

Construction du produit possible à la volée

Même ainsi le résultat est souvent trop gros

Graphe des Zones (exemple Alur Dill 1994)

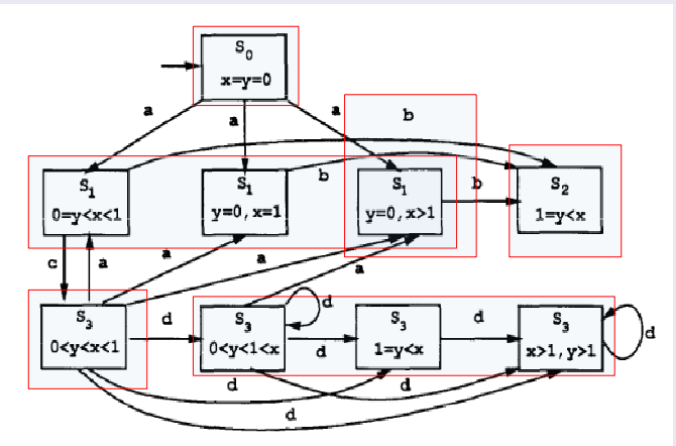
Graphe des Zones

Minimisation du graphe des régions en fusionnant des zones convexes.

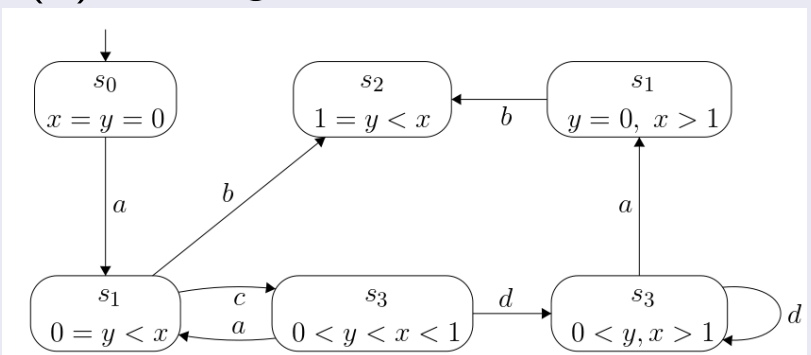
Zones constituent un *recouvrement* de l'ensemble des régions

Des régions aux zones

(a) Recouvrement des régions



(b) Passage au "Quotient"



En vrai, construction à la volée à base de DBM (cf domaines de tir des TPns)

Propriétés du graphe de zones

Propriétés

Graphe de zones Vs Automates temporisés :

- Pour chaque transition discrète :

  Si $(s, v) \xrightarrow{l} (s', v')$ Alors $(s, z) \xrightarrow{l} (s', z')$ avec $v \in z$ et $v' \in z'$

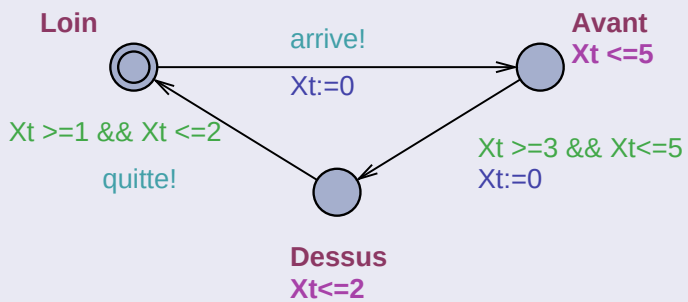
  Si $(s, z) \xrightarrow{l} (s', z')$ Alors $\exists v \in z, \exists v' \in z'$ tq $(s, v) \xrightarrow{l} (s', v')$

- Pour chaque transition d'écoulement du temps :

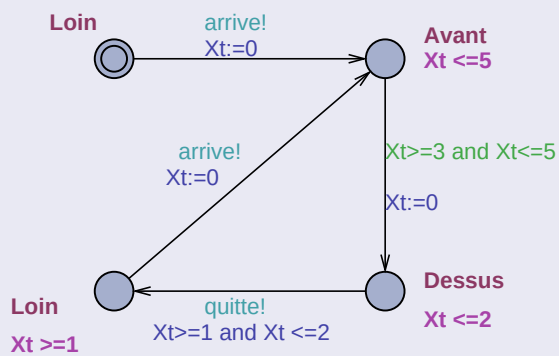
  Si $(s, v) \rightarrow_\theta (s, v + \theta)$
  Alors $(s, z) \rightarrow_\theta (s, z')$ avec $v \in z$ et $v + \theta \in z'$

Représentation d'un graphe de zones par un automate temporisé

Automate temporisé du train



Graphe des zones du train représenté par un automate temporisé



Outline

Compléments

Comparaison des modèles : Petri (borné) & Automates

Equivalence Langage

TPN $\equiv$ TA 2006

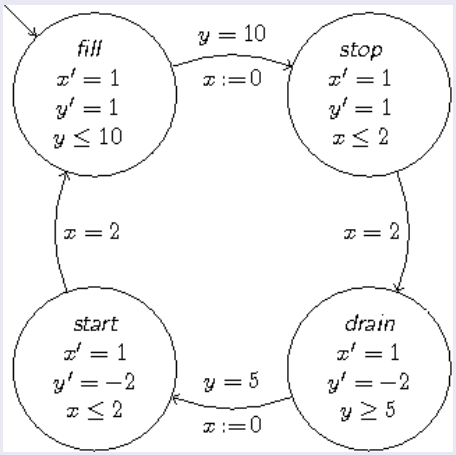
Bisimulation Faible

TPN $\leq$ TA 2004
PrTPN $\equiv$ TA 2006

Du temporisé à l'hybride (PN & Automates)

Temporel : variables continues (horloges) qui évoluent toutes avec la même dérivée

Hybride : les dérivées des variables continues sont distinctes et peuvent varier.
Accessibilité indécidable



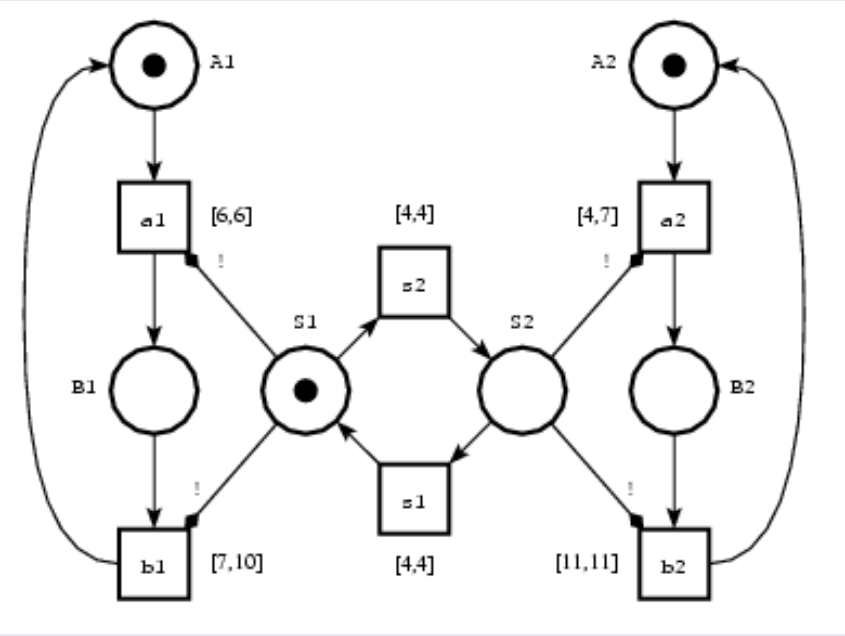
Modèles chronométrés (stopwatch)

$$\text{Temporel} < \text{Chronométré} \leq \text{Hybride}$$

TPN/AT chronomètres : horloges $\mapsto$ chronomètres dont la dérivée $\in \{0, 1\}$
Possibilité de modéliser la *préemption temporelle*
Ordonnancements préemptifs (ex EDF)

Modèles chronométrés (hybride linéaire)

Round-Robin en SwTPN (Stopwatch TPN)



Une transition sensibilisée peut être Active ou Suspendue

Caractéristiques

- Analyse indécidable (y compris pour les réseaux bornés)
- "Graphe de classes" adaptable mais terminaison non garantie
- Algorithme par sur-approximation (proprétés de sûreté)
  situation analogue pour les Automates à chronomètres

Bibliographie

Vérification de logiciels : techniques et outils du model-checking.
Ph. Schnoebelen, B. Bérard, M. Bidoit,
F. Laroussinie and A. Petit.
Vuibert, 1999.

Les Réseaux de Petri. Modèles fondamentaux,
M. Diaz Editeur
Hermes Science, 2001
Traité IC2 Information-Commande-Communication,

Petri Nets: Fundamental Models,
Verification and Applications
M. Diaz Editeur
ISTE-Wiley 2008

Modeling and Verification of Real-time Systems
S. Merz and N. Navet editors
ISTE-Wiley 2008

Time and Petri Nets
L Popova-Zeugmann
Springer-Verlag 2013

Outils

The tool Kronos
http://www-verimag.imag.fr/TEMPORISE/kronos/

Uppaal: A tool suite for verification of real-time systems
http://www.docs.uu.se/docs/rtmv/uppaal

Oris Tool - Analysis of timed and stochastic Petri nets
www.oris-tool.org/

Roméo - A tool for Time Petri Nets analysis
http://romeo.rts-software.org/

TINA: TIme Petri Net Analyzer
http://www.laas.fr/tina/

Hytech: The hybrid technology tool
http://www-cad.eecs.berkeley.edu/~tah/hytech

Table des Matières

Table des Matières

Table des Matières

Table des Matières