



Vérification de modèles dans les systèmes dynamiques à événements discrets

Plan

- Logique Temporelle Linéaire
- Systèmes dynamiques à événements discrets
- Vérification formelle de LTS en LTL
- CTL et CTL*
- Ouverture : Automates temporisés

Pour aller plus loin (S2)

- **IAMSI** (IA et Manipulation Symbolique de l'Information)
 - Inférences / SAT
 - Logique non-monotones (ASP)
 - Planification
 - + apprentissage/fouille de données symboliques,
- Mais aussi
 - **DJ** (Décision et Jeux) : représentation de préférences, décision multi-critère...
 - **FoSyMa** (Fondements des Systèmes multiagents) : Architecture d'agents, Protocoles d'interactions, Planification. Système asynchrone avec gestion du temps



Pour aller plus loin (S3)

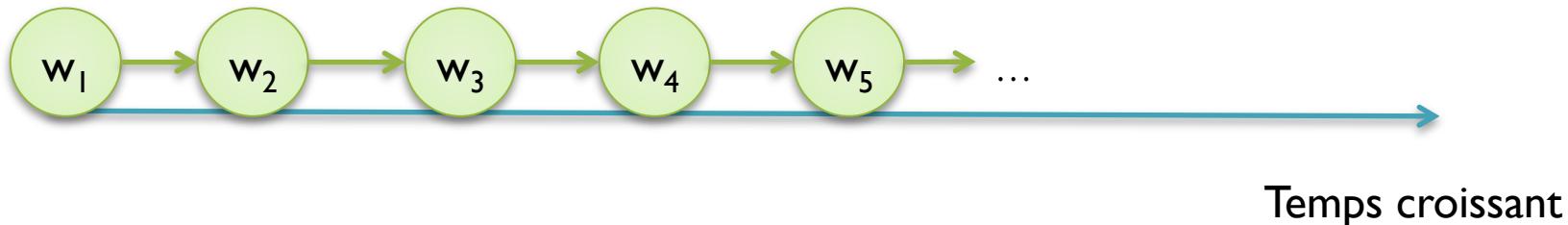
- **X-AI** : Explainable AI
 - Approches logiques d'explicabilité (logiques graduelles, abduction)
- **LODAS** : Linked Open Data et Apprentissage Symbolique
 - Ontologies : aspects pratiques des Logiques de Description (entre MLBDA et LRC)
 - Apprentissage Symbolique : utilisation de connaissances structurées dans l'apprentissage (prolog, inférences...)
- **COCOMA** Coordination et Consensus Multi Agent
 - SMA cognitifs/BDI
 - Argumentation
 - MDP



Logique Temporelle Linéaire (LTL)

Logique Temporelle Linéaire (LTL)

- Logique modale classique où chaque monde représente un instant et la relation d'accessibilité permet de passer d'un instant à l'instant suivant
 - Représentation linéaire : chaque monde a un et un seul successeur (ligne infinie de temps)
 - => Nec et Pos ont alors le même sens. C'est l'opérateur **O** (parfois aussi noté **X** pour neXt).

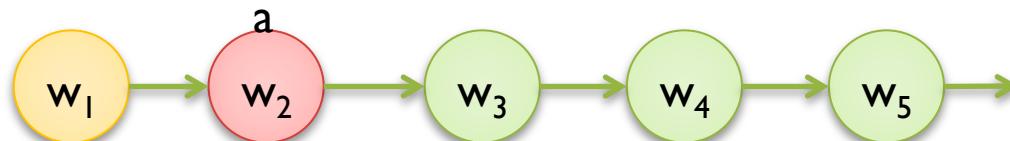


Logique Temporelle Linéaire

- Opérateurs

- Opérateur modal de base : **Oa**

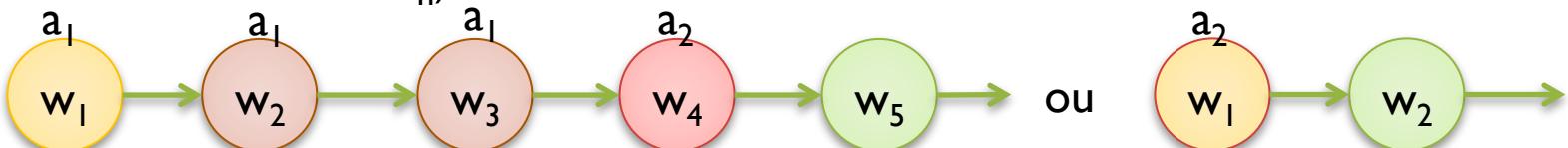
a est vrai au prochain instant



- Opérateur binaire \mathcal{U} (until) : $a_1 \mathcal{U} a_2$

a_1 vrai jusqu'à ce que a_2 le soit

- Il existe un instant présent ou futur w_n où a_2 sera vrai (potentiellement dès maintenant)
 - a_1 sera vrai dans tous les mondes depuis le monde courant w jusqu'à w_n exclus (donc peut ne pas être vrai du tout si $w=w_n$)

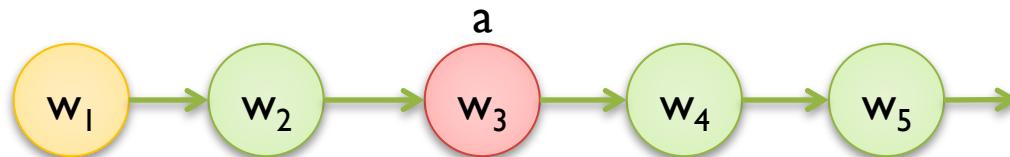


Logique Temporelle Linéaire

- Opérateurs

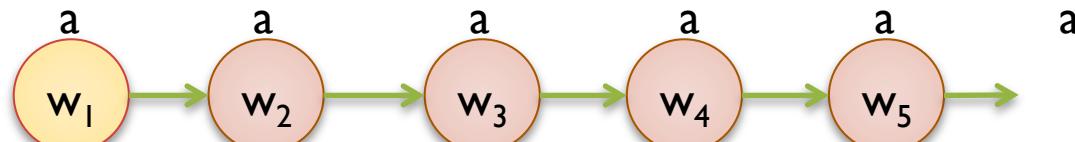
- Opérateur unaire F (Finally) : Fa

a sera vrai à au moins un moment (présent ou futur)



- Opérateur unaire G (Globally) : Ga

a est toujours vrai à partir de maintenant



Logique Temporelle Linéaire

- Syntaxe

- Grammaire

$$\varphi ::= p | \neg \varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \bigcirc \varphi | \varphi_1 \mathcal{U} \varphi_2 | F \varphi | G \varphi$$

- Opérateurs modaux

$$\varphi \mathcal{U} \psi = \bigvee_{k \geq 0} U_\varphi^k \psi \quad \text{où } U_\varphi \psi = \varphi \wedge \bigcirc \psi$$

$$F \varphi = \top \mathcal{U} \varphi = \bigvee_{k \geq 0} \bigcirc^k \varphi$$

$$G \varphi = \neg F \neg \varphi = \bigwedge_{k \geq 0} \bigcirc^k \varphi$$

Logique Temporelle Linéaire

- Modèle de Kripke linéaire

$$\mathcal{M} = \langle W, R, I \rangle (+ \mathcal{P})$$

$$W = \{w_i \mid i \in \mathbb{N}\} \qquad R = \{(w_i, w_{i+1}), i \in \mathbb{N}\}$$

$$I : \mathcal{P} \rightarrow 2^W$$

$\mathcal{M}, w_i \models p$ ssi $w_i \in I(p)$

$\mathcal{M}, w_i \models \neg\varphi$ ssi $\mathcal{M}, w_i \not\models \varphi$

$\mathcal{M}, w_i \models \varphi \wedge \psi$ ssi $\mathcal{M}, w_i \models \varphi$ et $\mathcal{M}, w_i \models \psi$

$\mathcal{M}, w_i \models \varphi \vee \psi$ ssi $\mathcal{M}, w_i \models \varphi$ ou $\mathcal{M}, w_i \models \psi$

$\mathcal{M}, w_i \models O\varphi$ ssi $\mathcal{M}, w_{i+1} \models \varphi$

$\mathcal{M}, w_i \models \varphi \mathcal{U} \psi$ ssi $\exists j, j \geq i, (\mathcal{M}, w_j \models \psi)$ et $\forall k \in \{i, \dots, j-1\}, (\mathcal{M}, w_k \models \varphi)$

$\mathcal{M}, w_i \models F\varphi$ ssi $\exists j, j \geq i, (\mathcal{M}, w_j \models \varphi)$

$\mathcal{M}, w_i \models G\varphi$ ssi $\forall j, j \geq i, (\mathcal{M}, w_j \models \varphi)$



Logique Temporelle Linéaire

- Comment considérer plus d'un scénario ?
 - Chaque ‘scénario’ correspond à un modèle de Kripke linéaire, ie, une séquence infinie de mondes que l'on appellera **chemin**
 - Au lieu de vérifier une formule dans un monde, on la vérifie sur un chemin (le début du chemin indiquant le monde duquel on part)
 - Il s'agit alors d'associer au système un ensemble de chemin : on définit un système de Kripke où un monde peut avoir plusieurs successeurs.

Logique Temporelle Linéaire

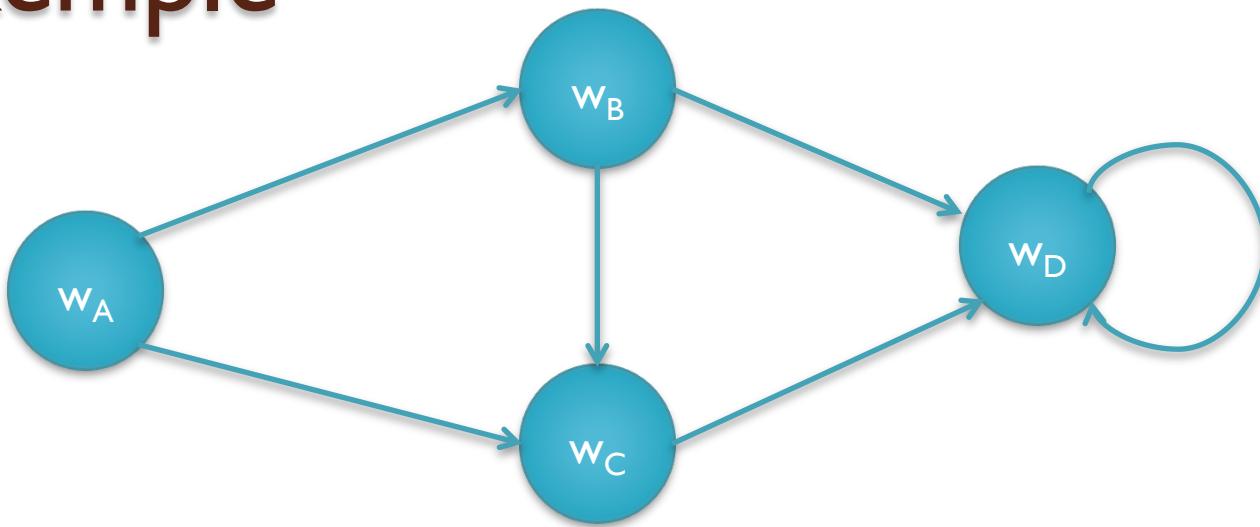
Soit $\mathcal{M} = \langle W, R, I \rangle$ (sur langage prop \mathcal{L}) un modèle de Kripke avec sérialité (chaque monde a au moins 1 successeur).

- A chaque monde, on peut associer l'ensemble des chemins commençant par ce monde

$$\Pi_w = \{w_0, w_1, \dots | w_0 = w, \forall i \in \mathbb{N}, w_i \in W \text{ et } (w_i, w_{i+1}) \in R\}$$

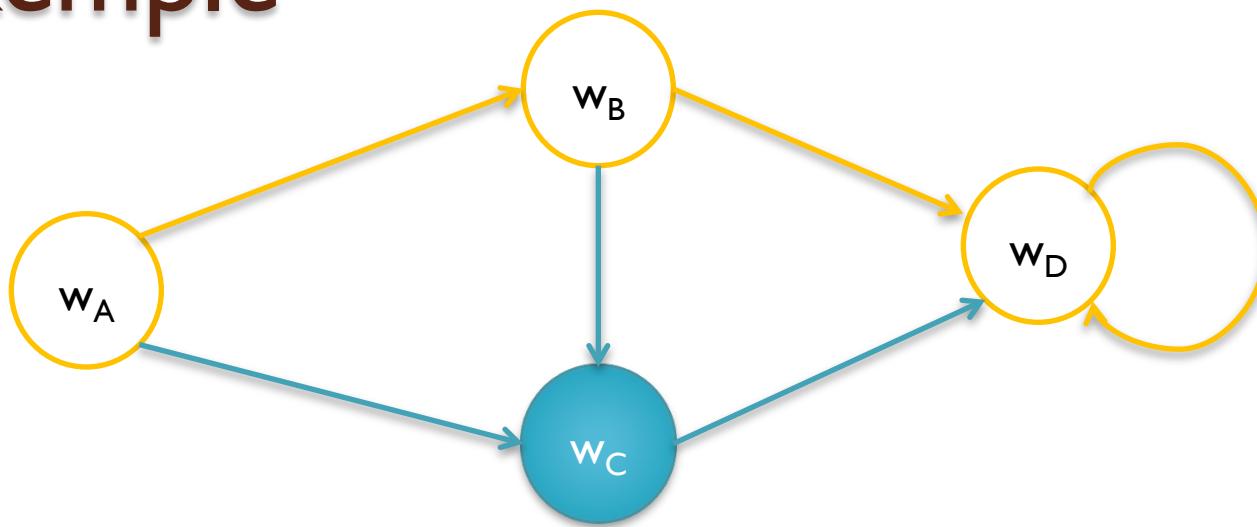
- Pour tout chemin $\pi = w_0, w_1, \dots$ on note
 - $\pi(i) = w_i$ (monde à l'étape i de π)
 - $\pi[i] = w_i, w_{i+1}, w_{i+2}, \dots$ (chemin décalé de i)

Exemple



- Π_{w_A} ne contient que 3 chemins
 -
 -
 -
 -
 -

Exemple



- Π_{w_A} ne contient que 3 chemins

- $\Pi_1 = w_A, w_B, w_D, w_D \dots$

-

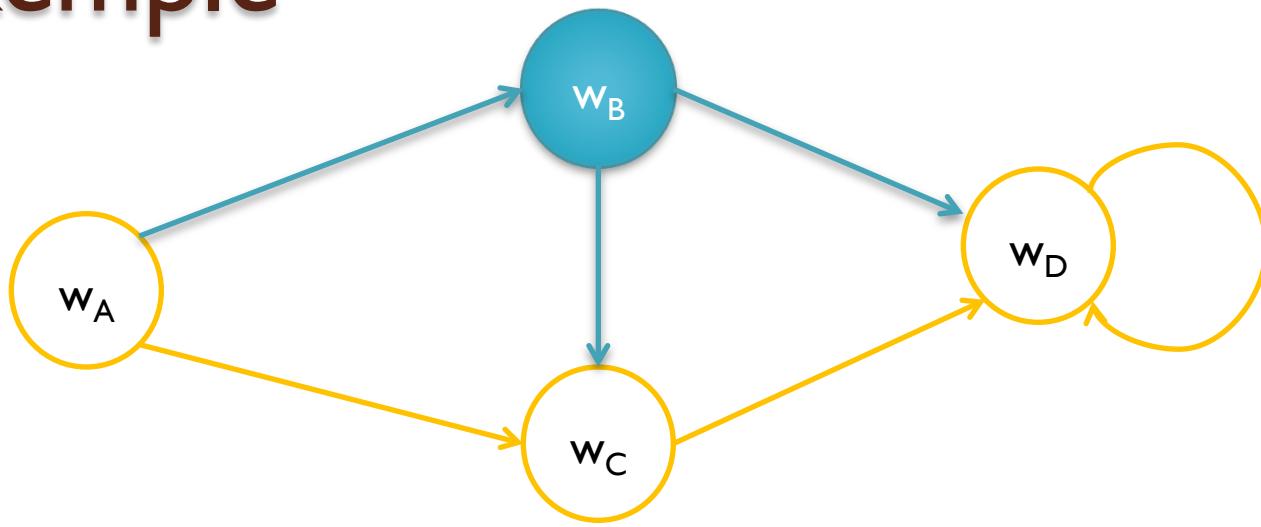
-

-

-

-

Exemple



- Π_{w_A} ne contient que 3 chemins

- $\Pi_1 = w_A, w_B, w_D, w_D \dots$

- $\Pi_2 = w_A, w_C, w_D, w_D \dots$

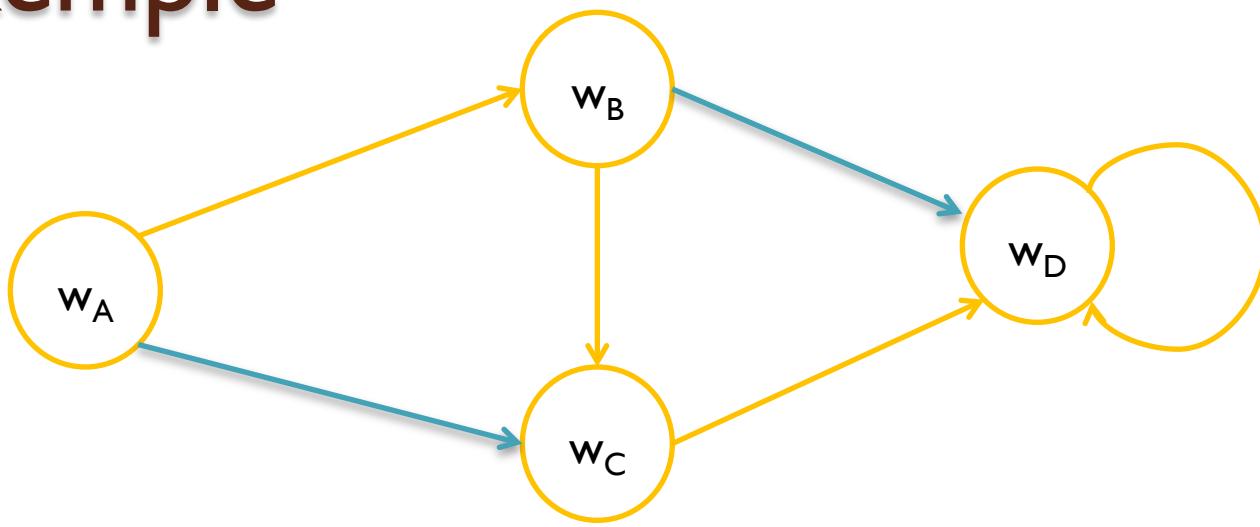
-

-

-

- ...

Exemple



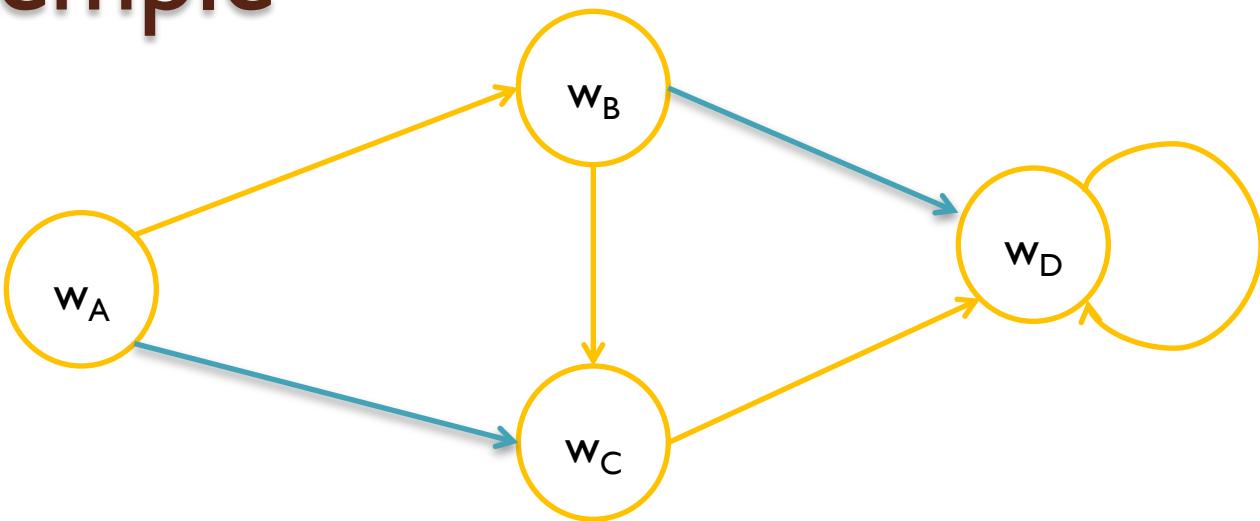
- Π_{w_A} ne contient que 3 chemins
 - $\Pi_1 = w_A, w_B, w_D, w_D \dots$
 - $\Pi_2 = w_A, w_C, w_D, w_D \dots$
 - $\Pi_3 = w_A, w_B, w_C, w_D, w_D \dots$

o

o

o

Exemple



- Π_{w_A} ne contient que 3 chemins

- $\Pi_1 = w_A, w_B, w_D, w_D \dots$
- $\Pi_2 = w_A, w_C, w_D, w_D \dots$
- $\Pi_3 = w_A, w_B, w_C, w_D, w_D \dots$

On a

- $\Pi_3(0) = w_A, \Pi_3(1) = w_B, \Pi_3(2) = w_C, \Pi_3(4+) = w_D$
- $\Pi_3^1 = w_B, w_C, w_D, w_D \dots$
- $\Pi_3^2 = w_C, w_D, w_D \dots$

Logique Temporelle Linéaire

- Sémantique des chemins

Soit $\mathcal{M} = \langle W, R, I \rangle$ (sur \mathcal{L}) un modèle de Kripke avec sérialité. L'ensemble de tous les chemins possibles basés sur ce modèle est $\Pi = \bigcup_{w \in W} \Pi_w$. Soit π un tel chemin :

$$\begin{array}{ll} \mathcal{M}, \pi \models p & \text{ssi } \pi(0) \in I(p) \\ \mathcal{M}, \pi \models X\varphi & \text{ssi } \mathcal{M}, \pi^1 \models \varphi \\ \mathcal{M}, \pi \models F\varphi & \text{ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi \\ \mathcal{M}, \pi \models G\varphi & \text{ssi } \forall i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi \\ \mathcal{M}, \pi \models \varphi U \psi & \text{ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \psi \text{ et} \\ & \quad \forall k \in \{0, \dots, i-1\}, \mathcal{M}, \pi^k \models \varphi \end{array}$$

F **valide** pour \mathcal{M} , ssi F vérifiée par tt chemin π de Π
F **satisfiable** pour \mathcal{M} ssi il existe un chemin π de Π vérifiant F (ie ssi $\neg F$ n'est pas valide pour pour \mathcal{M}).

Logique Temporelle Linéaire

- En pratique, il est courant de ne considérer que certains des chemins possibles (typiquement, seulement ceux qui commencent dans un ‘état initial’).
- On définit alors un ensemble de chemin Π ($\Pi \subseteq \bigcup_{w \in W} \Pi_w$) pour restreindre validité/sat.:
- F valide pour (\mathcal{M}, Π) ssi F vérifiée par tt chemin π de Π
- F satisfiable pour (\mathcal{M}, Π) ssi il existe un chemin π de Π vérifiant F (ie ssi $\neg F$ n'est pas valide pour pour (\mathcal{M}, Π)).

Logique Temporelle Linéaire

- Sémantique des chemins

Soit $\mathcal{M} = \langle W, R, I \rangle$ (sur \mathcal{P}) un modèle de Kripke avec sérialité et Π une ensemble de chemins basés sur ce modèle ($\Pi \subseteq \bigcup_{w \in W} \Pi_w$)

$$\mathcal{M}, \pi \models p \quad \text{ssi} \quad \pi(0) \in I(p)$$

$$\mathcal{M}, \pi \models X\varphi \quad \text{ssi} \quad \mathcal{M}, \pi^1 \models \varphi$$

$$\mathcal{M}, \pi \models F\varphi \quad \text{ssi} \quad \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models G\varphi \quad \text{ssi} \quad \forall i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models \varphi U \psi \quad \text{ssi} \quad \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \psi \text{ et} \\ \forall k \in \{0, \dots, i-1\}, \mathcal{M}, \pi^k \models \varphi$$

Note : dans ces définitions, comme π indique déjà ce qui est ou non accessible, seul I intervient directement

Logique Temporelle Linéaire

- Sémantique des chemins (ss modèle de Kripke)

Soit Π un ensemble de chemins basés formé à partir d'un vocabulaire V et I un mapping de \mathcal{P} vers V . On peut poser $\mathcal{M} = \langle V, \Pi, I \rangle$ et conserver les définitions telles quelles

$$\mathcal{M}, \pi \models p \quad \text{ssi } \pi(0) \in I(p)$$

$$\mathcal{M}, \pi \models X\varphi \quad \text{ssi } \mathcal{M}, \pi^1 \models \varphi$$

$$\mathcal{M}, \pi \models F\varphi \quad \text{ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models G\varphi \quad \text{ssi } \forall i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models \varphi U\psi \quad \text{ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \psi \text{ et}$$

$$\forall k \in \{0, \dots, i-1\}, \mathcal{M}, \pi^k \models \varphi$$

F **valide** pour (V, Π, I) ssi F vérifiée par tt chemin π de Π

F **satisfiable** pour (V, Π, I) ssi il existe un chemin π de Π vérifiant F (ie ssi $\neg F$ n'est pas valide pour pour (Π, I)).

Logique Temporelle Linéaire

- Etant donné un ensemble de chemin défini sur un vocabulaire $\langle V, \Pi, I \rangle$, on peut se ramener à un modèle de Kripke $\mathcal{M} = \langle W, R, I' \rangle$ (infini) sur chemins Π' comme suit:

$$W = \{w_{\pi, i} \mid \pi \in \Pi, i \in \mathbb{N}\}$$

$$R = \{(w_{\pi, i}, w_{\pi, i+1}) \mid \pi \in \Pi, i \in \mathbb{N}\}$$

$$I'(p) = \{w_{\pi, i} \mid \pi(i) \in I(p)\}$$

$$\Pi' = \bigcup_{\pi \in \Pi} \Pi_{w_{\pi, 0}}$$

Les formules valide pour $\langle V, \Pi, I \rangle$ sont exactement celles valides pour (\mathcal{M}, Π')

Logique Temporelle Linéaire

- Sémantique de mot
 - LTL souvent défini sur sémantique de mot (e.g. wikipedia)
 - Un ω -mot m est une séquence d'interpretations de \mathcal{P} : $m = a_0 a_1 \dots$ où chaque a_i est dans $2^{\mathcal{P}}$.
 - C'est un chemin de $\langle V, \sqcap, I \rangle$ avec $V = 2^{\mathcal{P}}$ et $I(p) = \{a \mid p \text{ dans } a\}$.
 - Langage : ensemble de mot \sim ensemble de chemins

$m \models p$ ssi $p \in m(0)$

$m \models X\varphi$ ssi $m^1 \models \varphi$

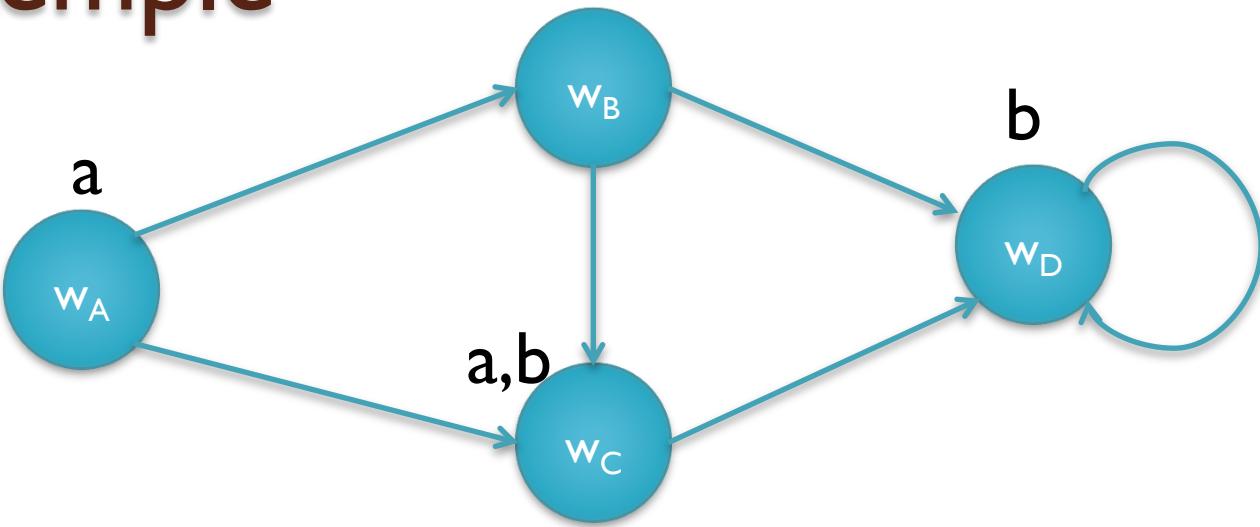
$m \models F\varphi$ ssi $\exists i \in \mathbb{N}, m^i \models \varphi$

$m \models G\varphi$ ssi $\forall i \in \mathbb{N}, m^i \models \varphi$

$m \models \varphi U \psi$ ssi $\exists i \in \mathbb{N}, m^i \models \psi$ et

$\forall k \in \{0, \dots, i-1\}, m^k \models \varphi$

Exemple



$$\mathcal{P} = \{a, b\}, I(a) = \{w_A, w_C\}, I(b) = \{w_C, w_D\}$$

- $(F G b)$ est valide dans \mathcal{M}
- $(G b)$ valide dans $(\mathcal{M}, \prod_{w_D})$ et $(\mathcal{M}, \prod_{w_C})$, insatisfiable dans $(\mathcal{M}, \prod_{w_A})$ et $(\mathcal{M}, \prod_{w_B})$
- $(X a)$ satisfiable dans $(\mathcal{M}, \prod_{w_A})$



Systèmes dynamiques à événements discrets

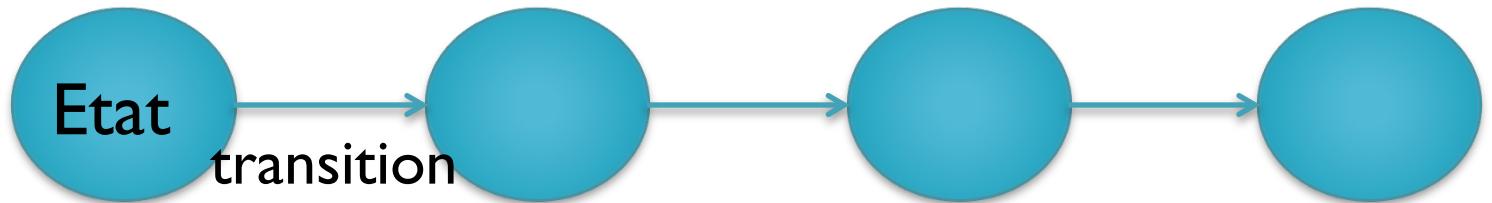


Systèmes dynamiques à évènements discrets

- Dynamique : système évolue avec le temps
- Evènements discrets
 - Par opposition à continu, temps modélisé de façon discrètes

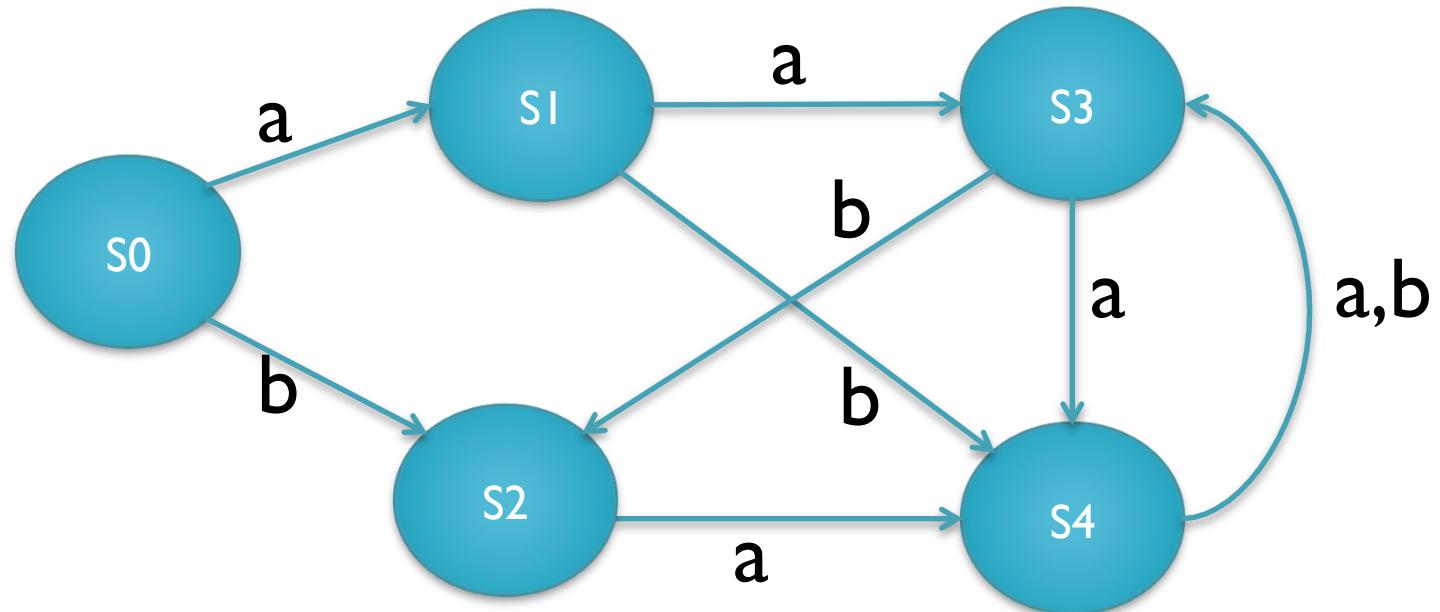
Modélisation en succession d'états et de transitions

Systèmes dynamiques à évènements discrets



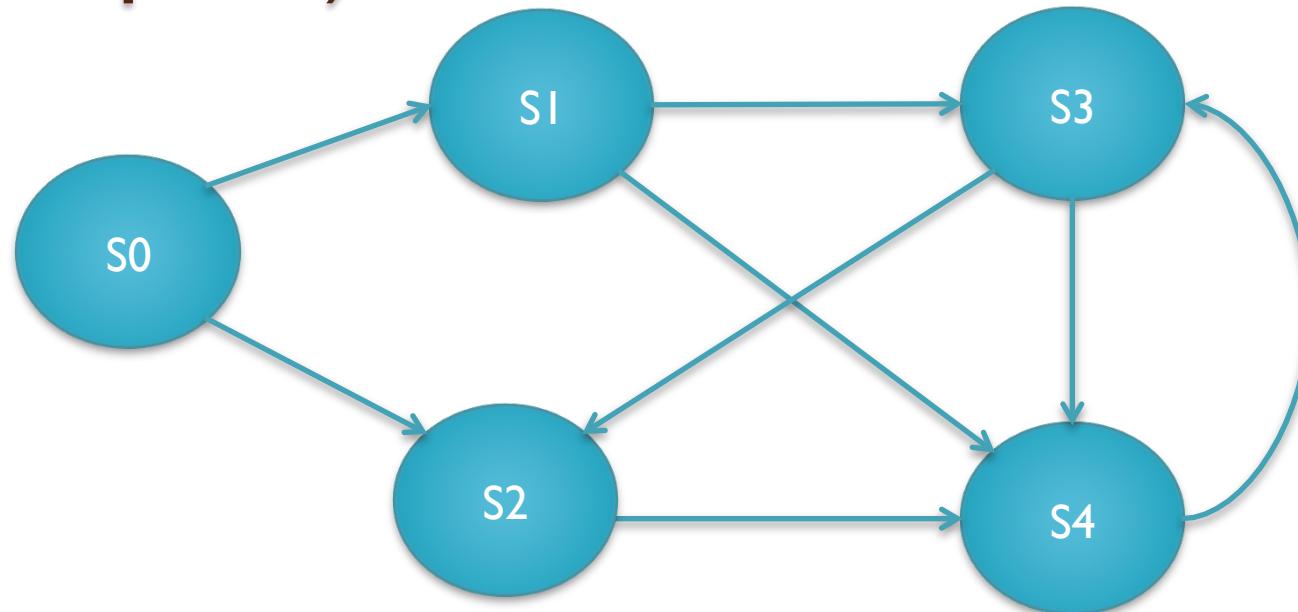
- Etat
 - Statut du monde à un instant donné
 - Constant entre 2 transition
- Transition = changement d'état
 - Actions, évènements
 - Label

Systèmes dynamiques à évènements discrets



- Typiquement, plusieurs transitions possible depuis un état : graphe
- Complet ? Déterministe ?

Systèmes à transitions d'états (non-étiqueté)



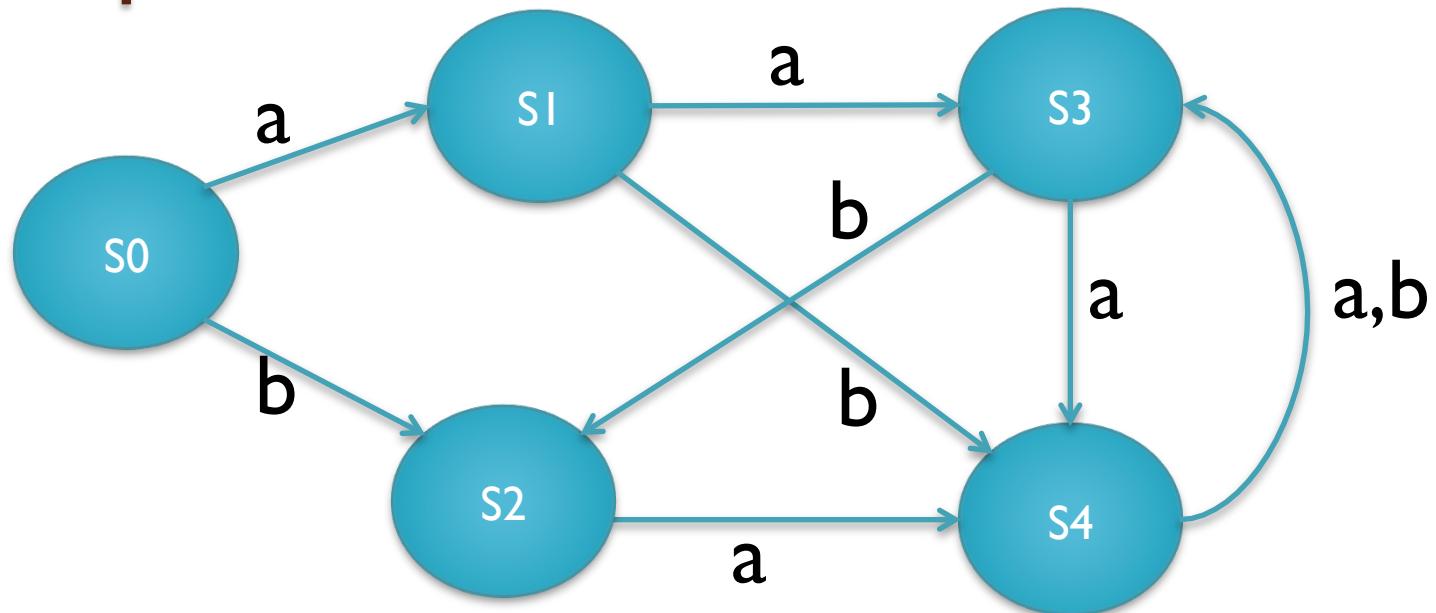
- Formalisme général (\sim graphe dirigé)

$$\langle S, T \rangle$$

S ensemble d'états

$T \subseteq S \times S$ relation binaire sur S (transitions)

Systèmes à transitions d'états étiqueté

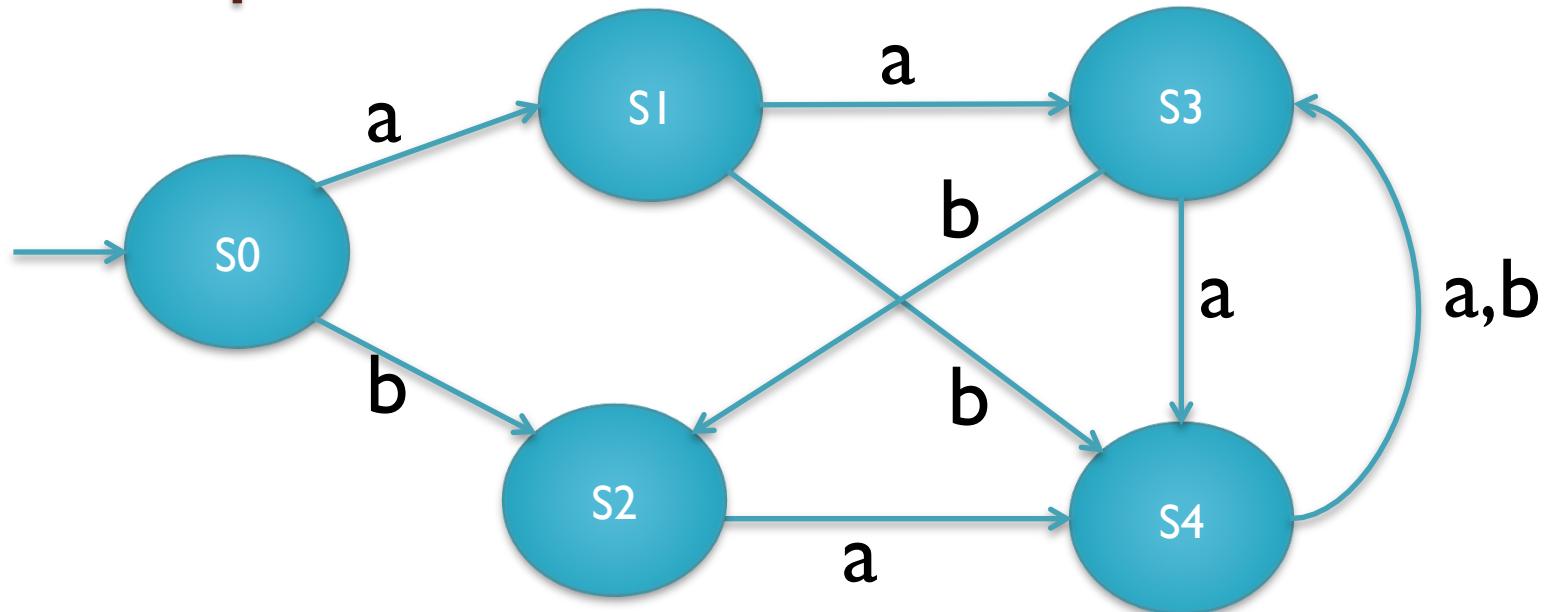


- LTS (Labeled Transition System)

$$\langle S, L, T \rangle$$

S ensemble d'états ; L ensemble de labels
 $T \subseteq S \times S$ relation binaire sur S (transitions)

Systèmes à transitions d'états étiqueté

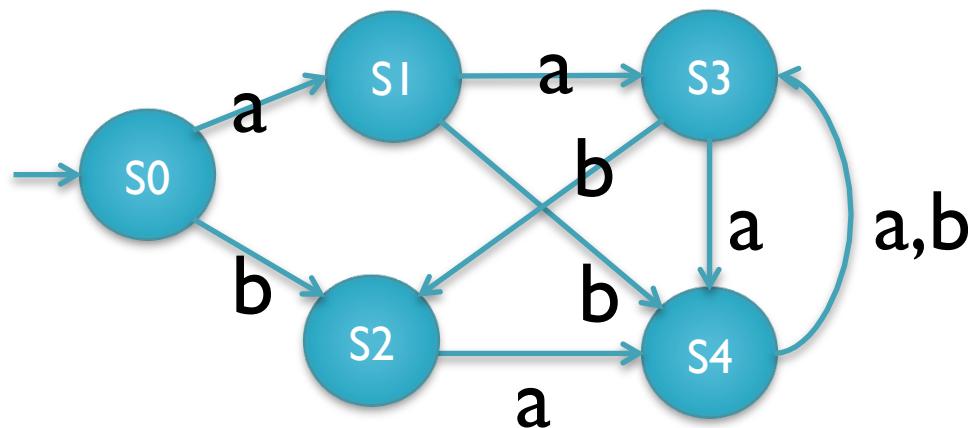


- LTS (Labeled Transition System)

$$\langle S, L, T \rangle + s_0, \text{état initial}$$

S ensemble d'états ; L ensemble de labels
 $T \subseteq S \times L \times S$ (transitions étiquetées)

LTS with initial state $\langle S, s_0, L, T \rangle$



- Exécution

$$\rho = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_4 \xrightarrow{b} s_3 \xrightarrow{a} s_4 \dots$$

- Mot lu par l'exécution (transitions)

$$w = a, b, b, a$$

- (dual : s-mot)

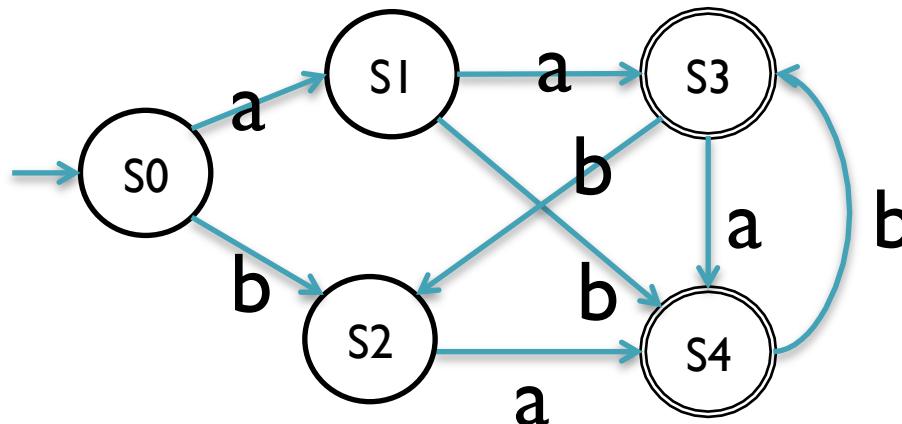
$$w_s = s_0, s_1, s_4, s_3, s_4, \dots$$

Systèmes à transitions d'états

- Exemples explicites ou dérivés
 - Réseaux de Pétri
 - Modèles de Kripke (exp. wrt LTL, CTL)
 - Automates finis
 - Processus de décision Markovien (MDP)
 - Planification : formalismes STRIPS
- Définir l'espace d'états (liste ou définition implicite, possiblement infini) et les transitions (conditions pour que $(s1, s2)$ soit dans T)

Automate finis

- LTS avec état initial et états de sortie



$\langle S, s_0, L, T, F \rangle$

Mot accepté : mot lu par exécution finie se terminant dans un état final

$$\rho = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_4 \xrightarrow{b} s_3 \xrightarrow{a} s_4$$

Langage reconnu : ensemble des mots acceptés (abba reconnu, aab ou bb non)

Réseau de Pétri comme LTS

$\langle S, L, T \rangle$

- Réseau de Petri ($P, T, Pre, Post$)
 - Etats : ensemble des marquages $M : N^P$ (infini)
 - Labels : transitions ($L = T$)
 - Transitions : (M_1, t, M_2) dans T ssi
 - t est déclencheable dans M_1
 - $M_2 = M_1 - Pre(t) + Post(t)$
- Réseau de Pétri marqué
 - Etat initial $s_0 =$ marquage initial M_0
 - Le graphe des marquages accessibles coincide avec LTS (si borné)

Modèles de Kripke et LTS

- Cadre de Kripke $\langle W, R \rangle$: système de transition $\langle S, T \rangle$
- Modèle de Kripke : $\langle W, R, V \rangle$ où V valuation des mondes (2^{PA}) où PA est l'ensemble des var. prop.
 - Etats étiquetés (ou considérer état = monde + sa valuation)
- En LTL : R correspond bien au sens des transitions d'un pas. (idem en CTL).
- Non étiqueté : quand on considère ce que lit une exécution, on utilise s-mot (basé sur monde), voire ω -mot : S_0, S_1, \dots où chaque S_i est dans 2^{PA}



Vérification formelle de LTS en LTL

Vérification formelle

- Vérifier des propriétés sur un modèle
- Propriétés = formules de logique modale temporelle
 - Sur quoi portent-elle ? (États ou transition)
 - Typiquement, nécessité de pouvoir raisonner sur les transitions aussi
- Modèle = description des évolutions possibles du système (ici système dynamique à événement discret)
 - LTS et variantes
 - Réseau de Pétri

Vérification formelle

- Modèles (LTS ou Petri) permettent de simuler des scénario
 - Exécution
$$\rho = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_4 \xrightarrow{b} s_3 \xrightarrow{a} s_4 \dots$$
a,b : transition. s_i : états (ou marquages)
- Modèles de Kripke : transition non étiquetées
 - On perd l'information sur transitions. Les chemin : s_0, s_1, \dots correspondent aux s-mot (ou ω -mot) plutôt qu'aux mots formés sur les transitions
 - Chemins forcément infinis (alors que certains système modélisent des exécutions finies)

Vérification formelle

- Problème I : chemins sur états vs transition
- Idée : associer à chaque état la transition qu'on prend pour en sortir (démultiplie les états)

Ainsi, l'exécution

$$\rho = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_4 \xrightarrow{b} s_3 \xrightarrow{a} s_4 \dots$$

Correspond au chemin

$$\pi = (s_0, a), (s_1, b), (s_4, b), (s_3, a), (s_4, \dots)$$

$\nabla = W$ défini dans ExT

Vérification formelle

- Problème 2 : prise en compte de chemin finis (e.g. marquage puits)
- Idée : prolonger les chemins finis avec infinité d'éléments ‘vides’ (état Fin / transition vide μ)

Ainsi, l'exécution finie

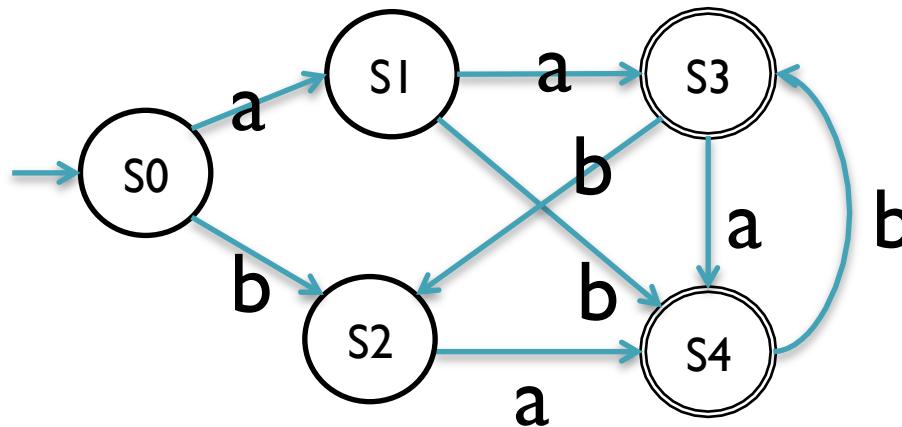
$$\rho = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_4 \xrightarrow{b} s_3 \xrightarrow{a} s_4$$

Correspond au chemin infini

$$\pi = (s_0, a), (s_1, b), (s_4, b), (s_3, a), (s_4, \mu), (Fin, \mu), \dots$$

Exemple

- Automate fini $\langle S, s_0, L, T, F \rangle$



- Modèle de Kripke $\langle W, R, I_L \rangle$ avec chemins Π

$$W = \{(s, l) \in S \times L \mid \exists s' \in S. (s, l, s') \in T\}$$

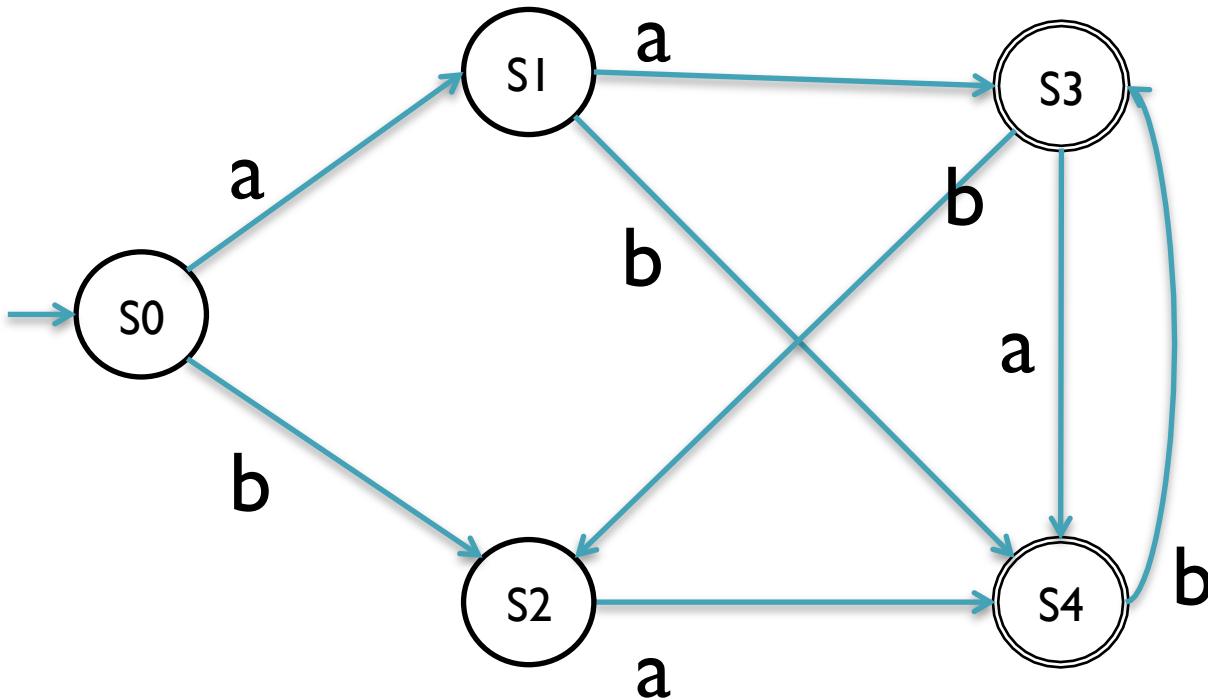
$$\cup \{(s, \mu) \mid s \in F\} \cup \{(Fin, \mu)\}$$

$$R = \{((s, l), (s', l')) \mid (s, l, s') \in T, (s', l') \in W\}$$

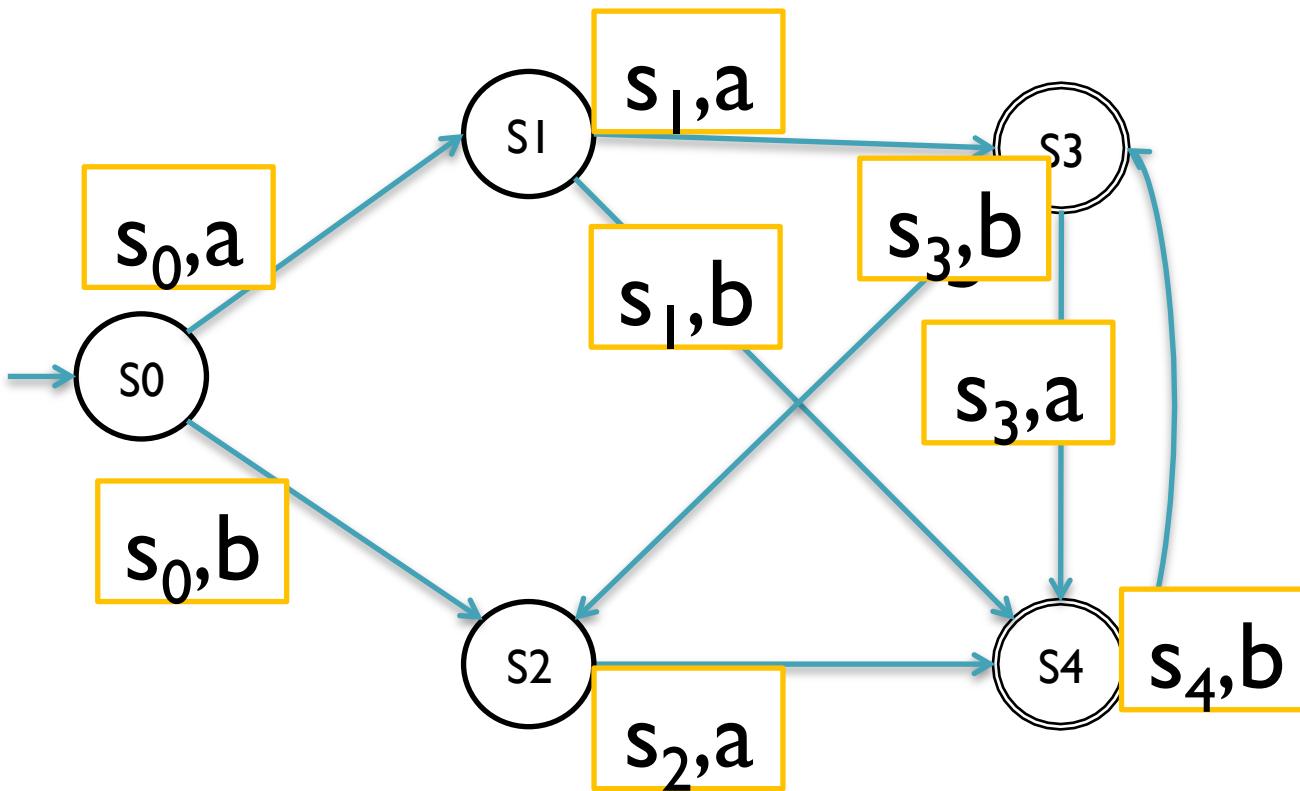
$$\cup \{((s, \mu), (Fin, \mu)) \mid s \in F\}$$

$$I_L(p) = \{(s, l) \in W \mid l = p\} \quad \text{et} \quad \Pi = \bigcup_{(s_0, l) \in W} \Pi_{(s_0, l)}$$

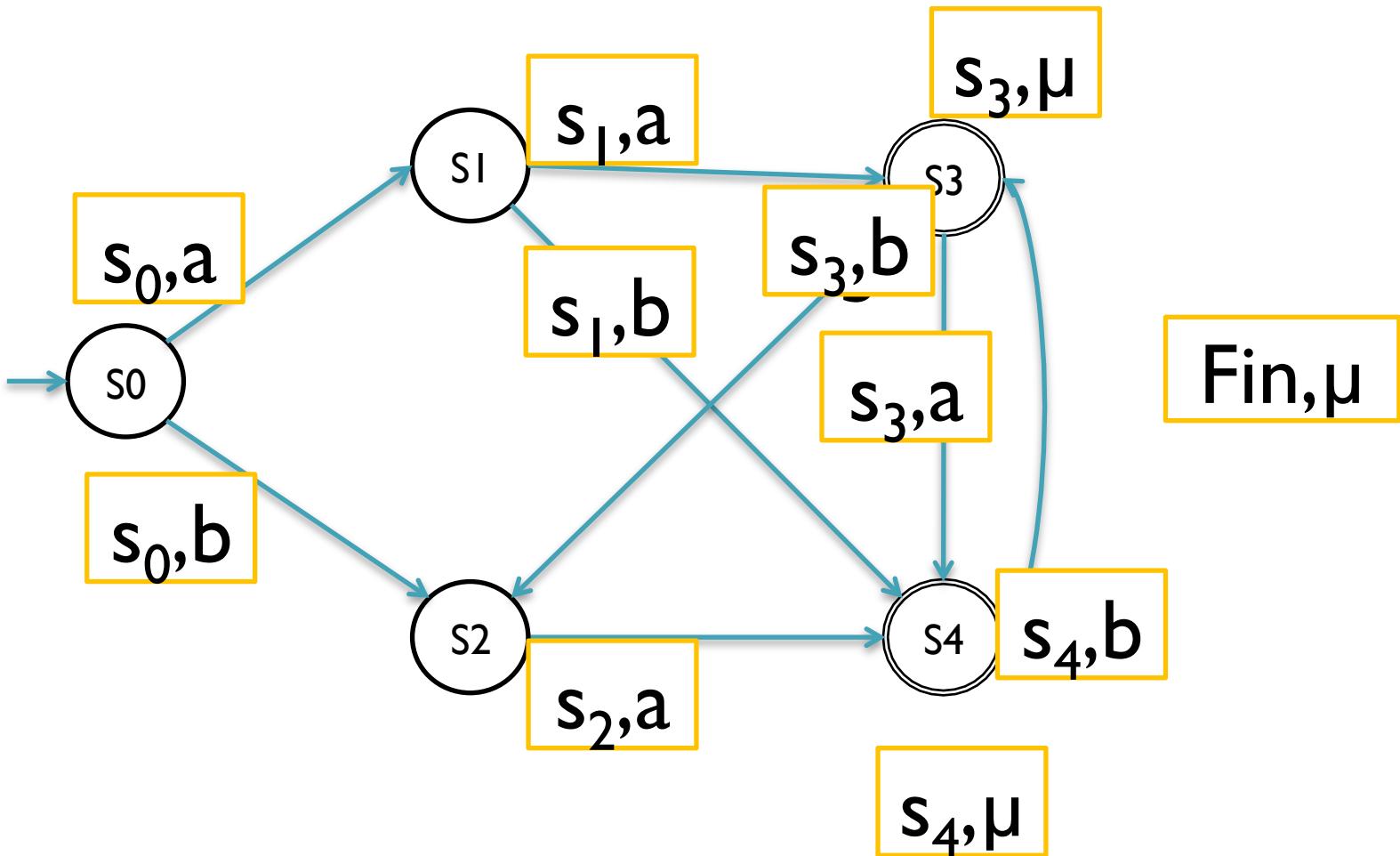
Exemple



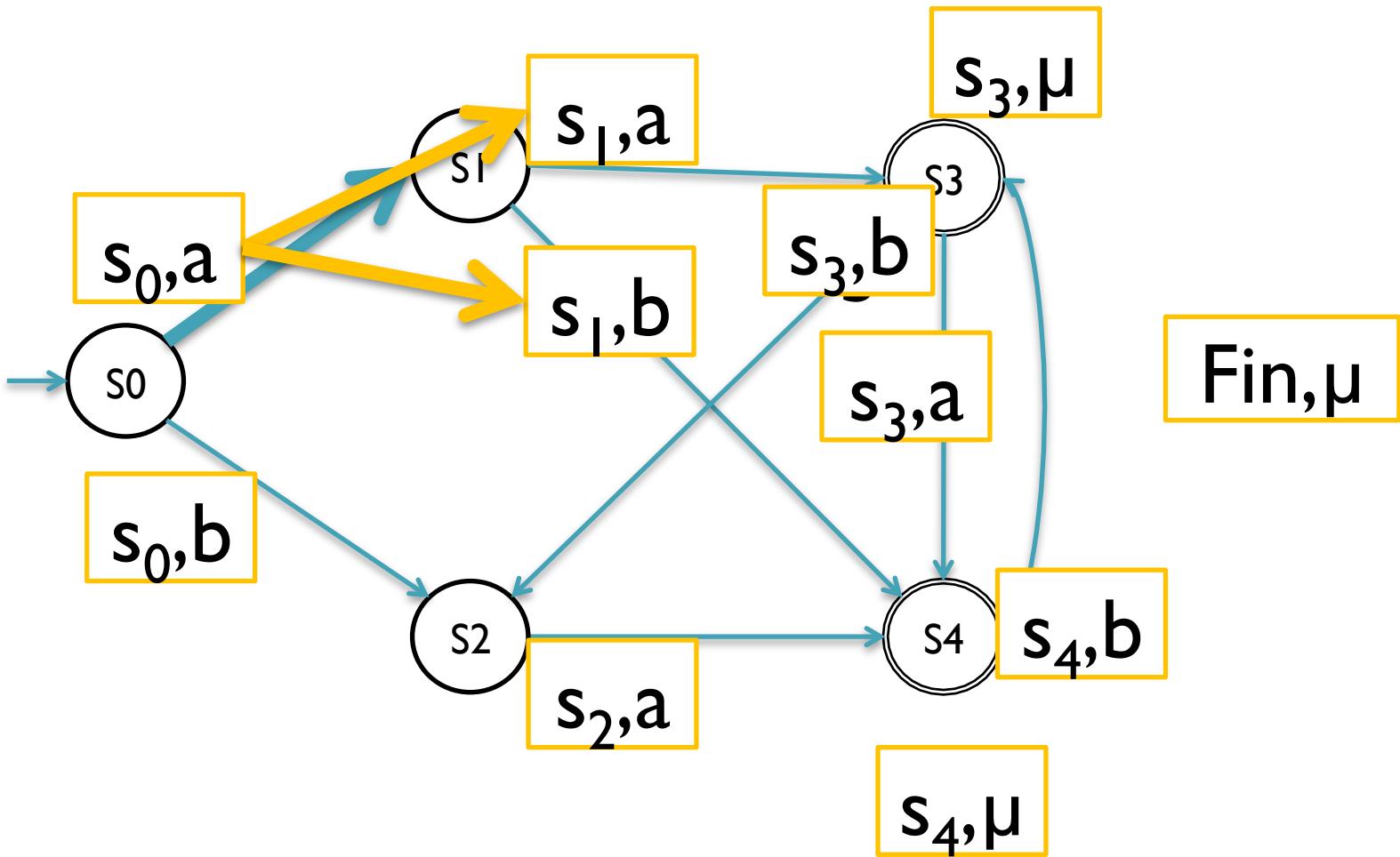
Exemple



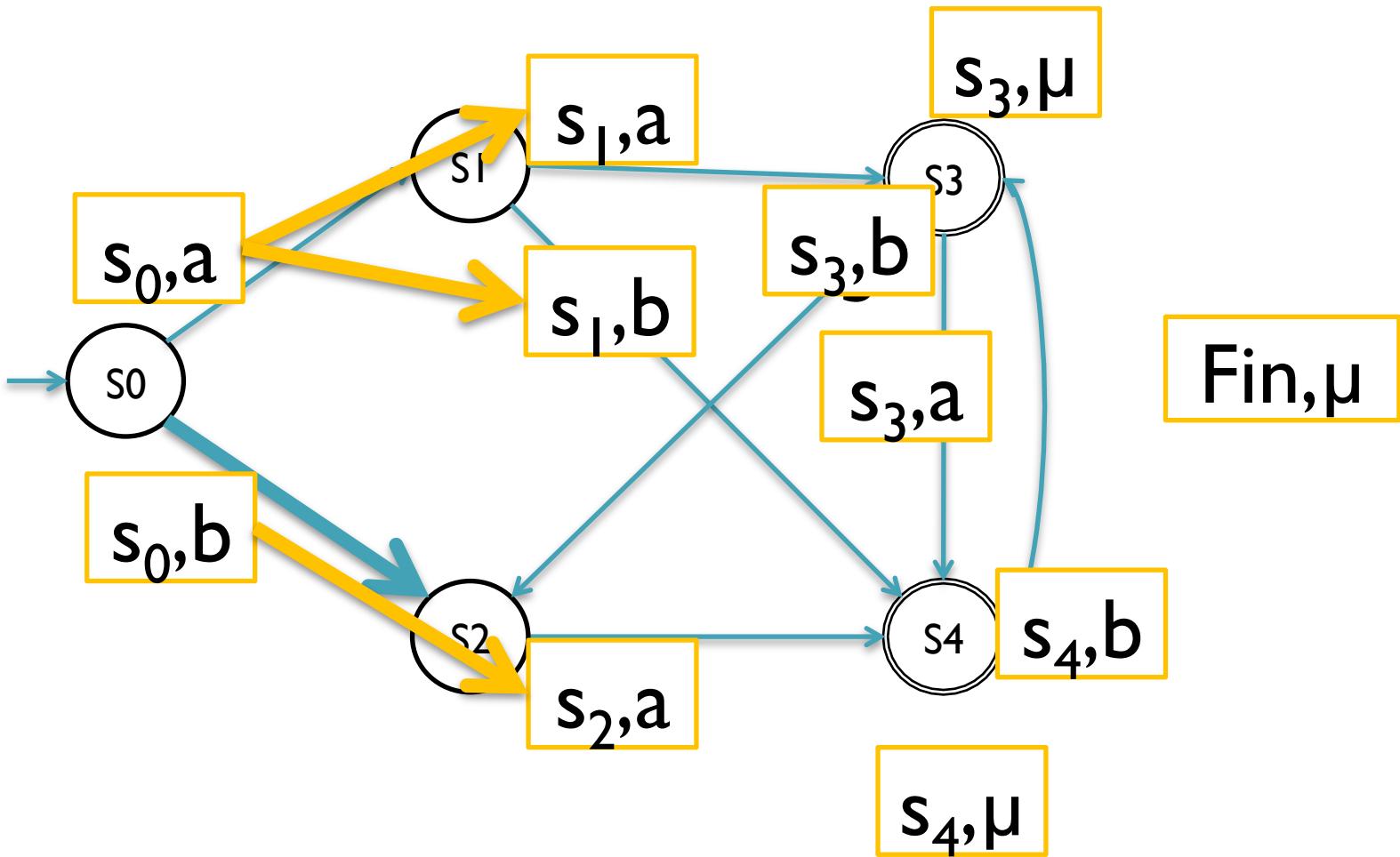
Exemple



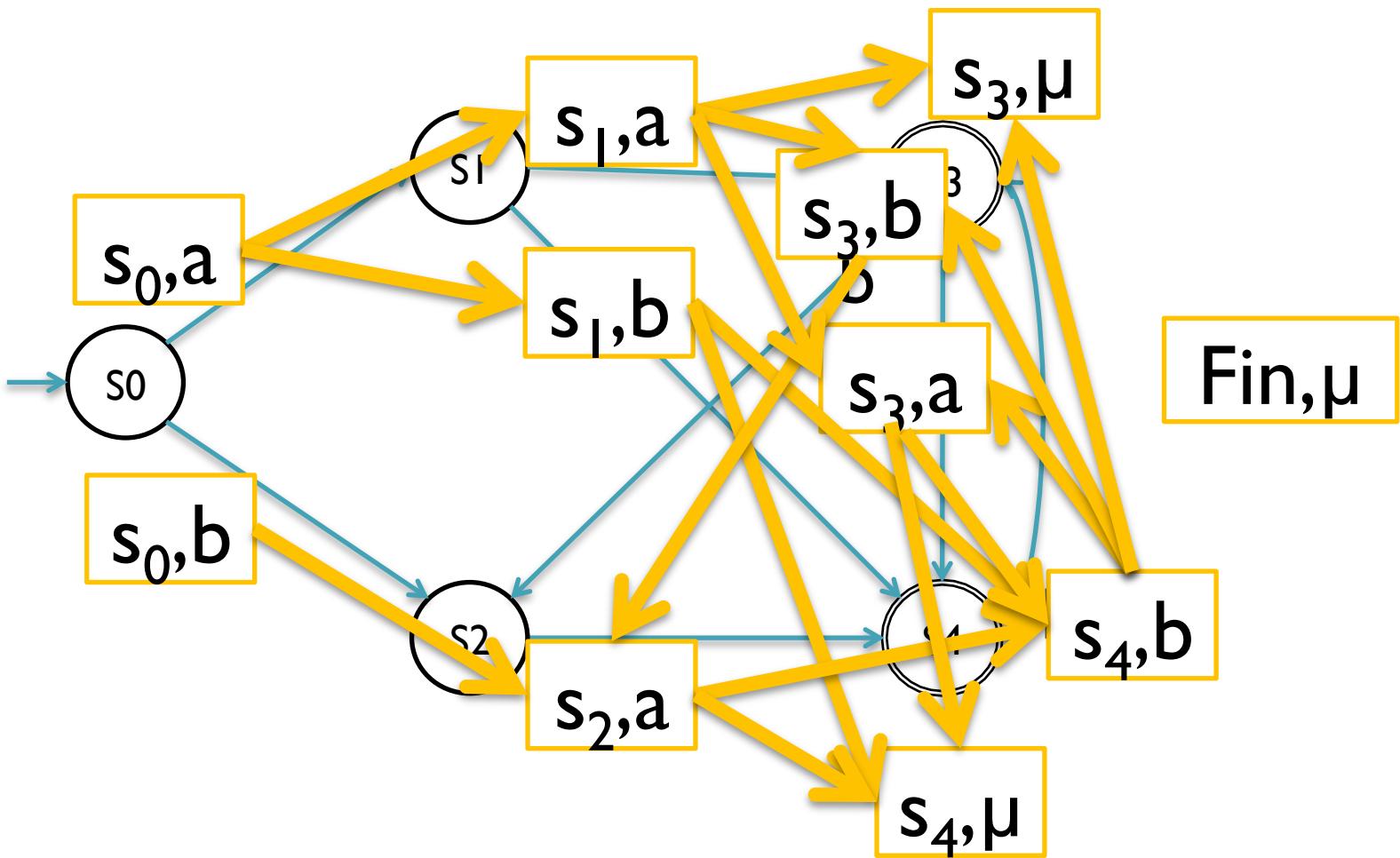
Exemple



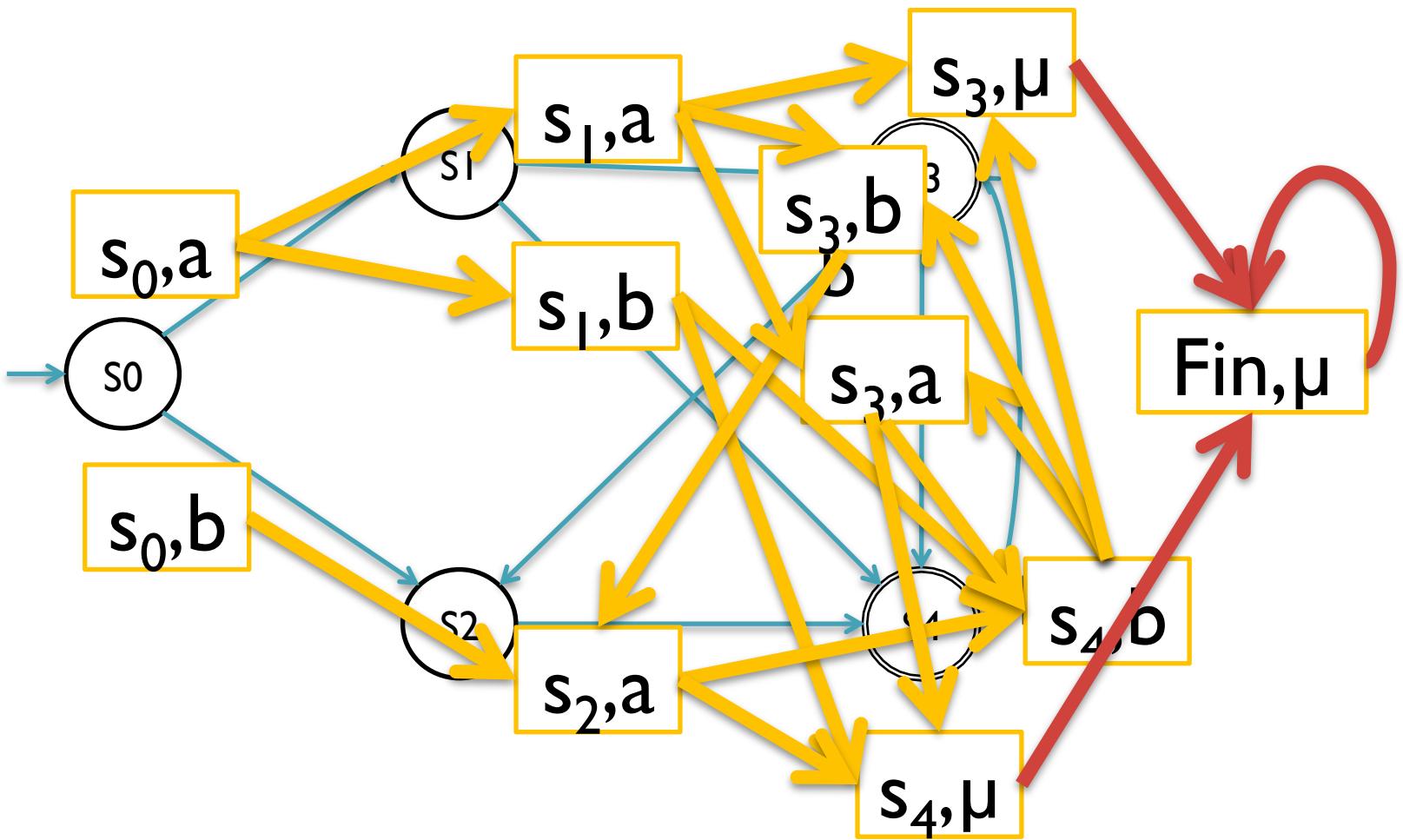
Exemple



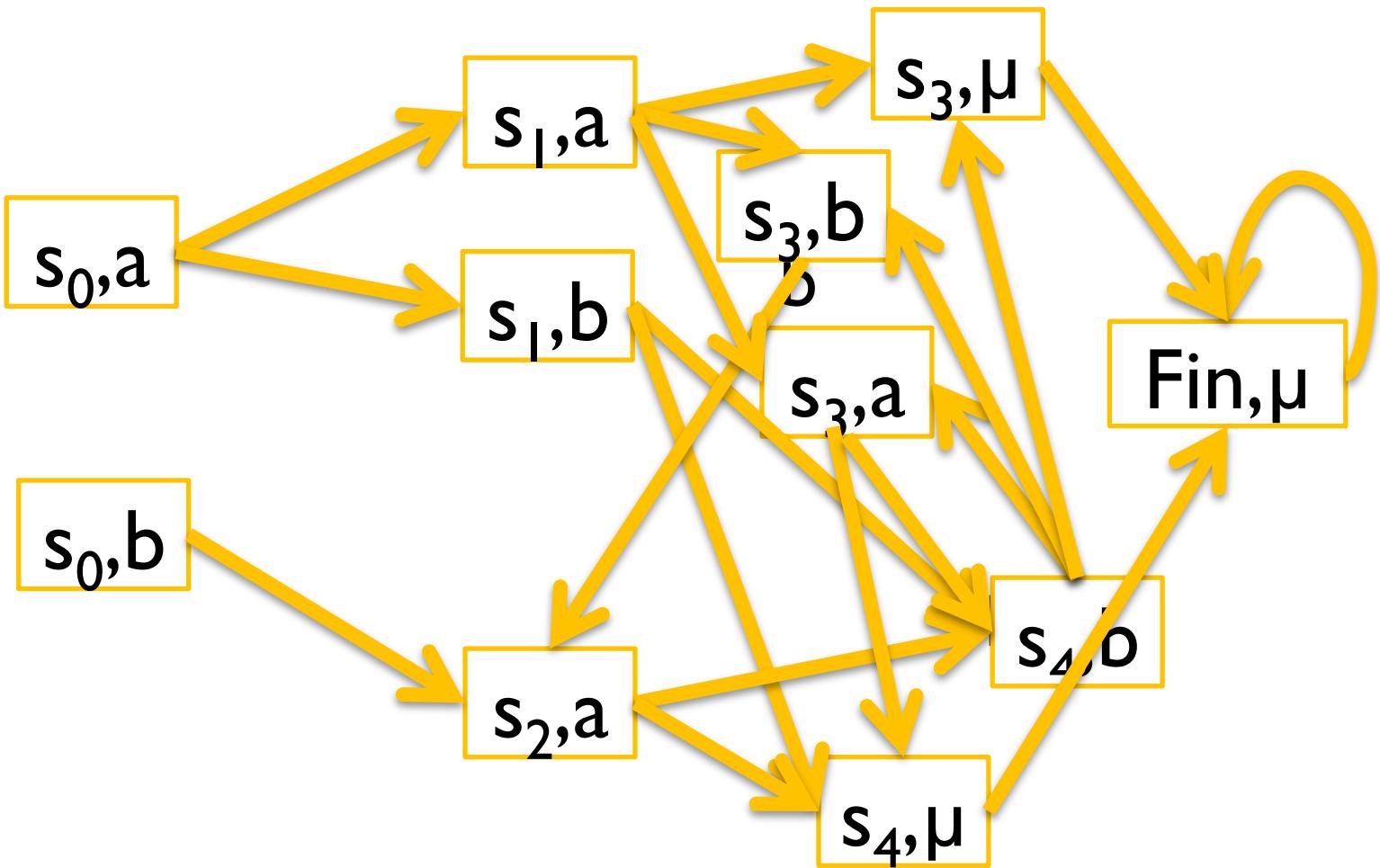
Exemple



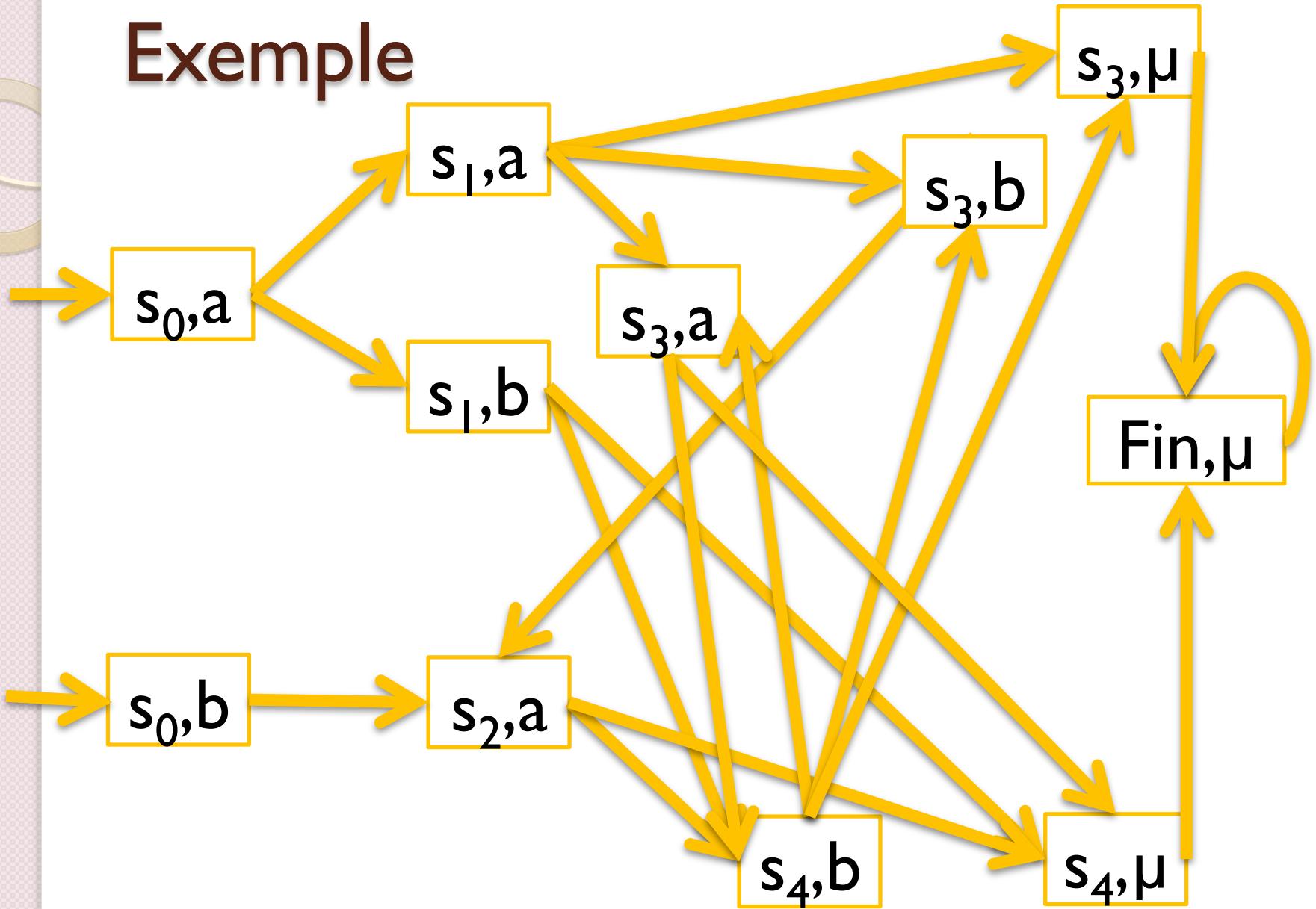
Exemple



Exemple



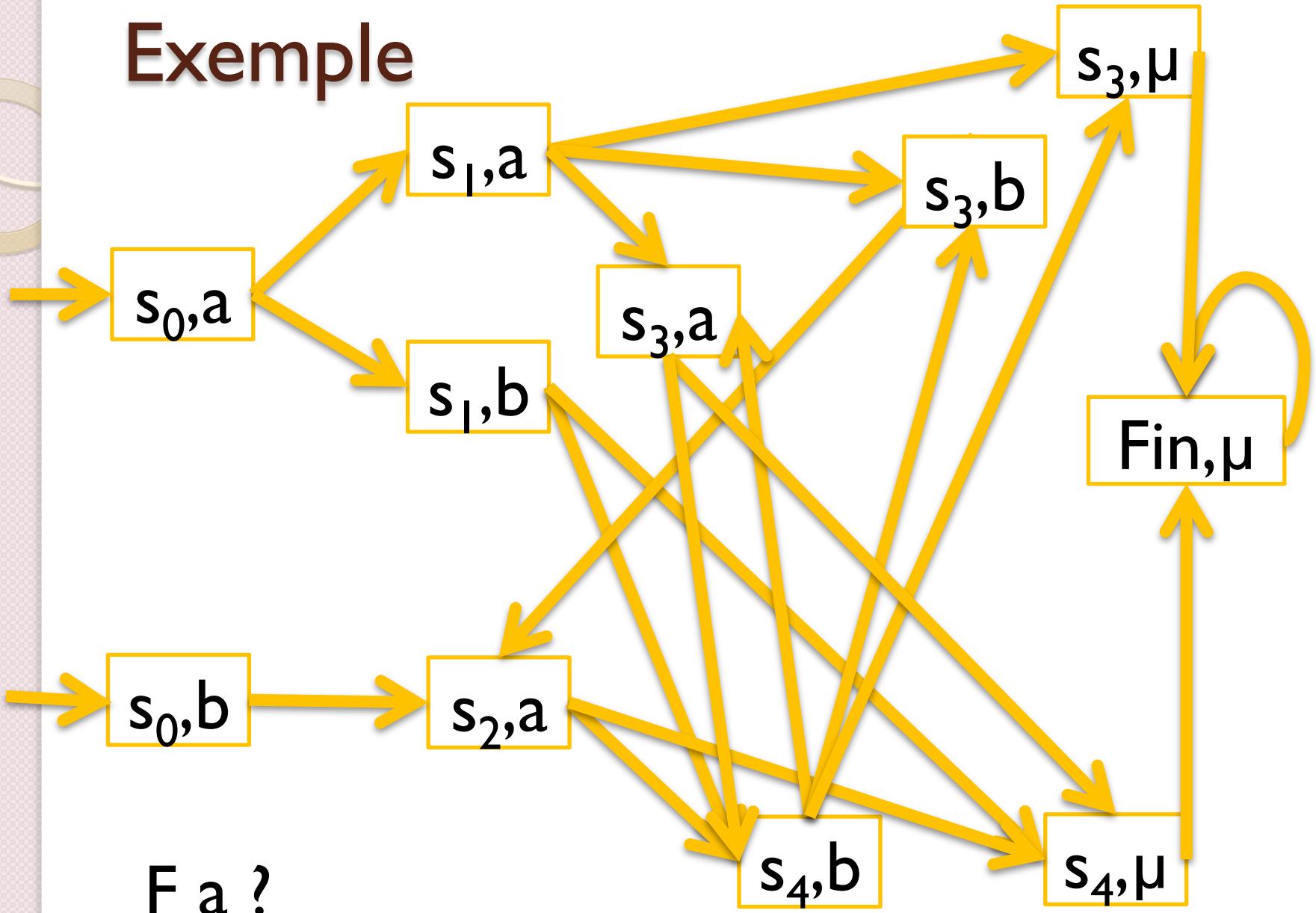
Exemple



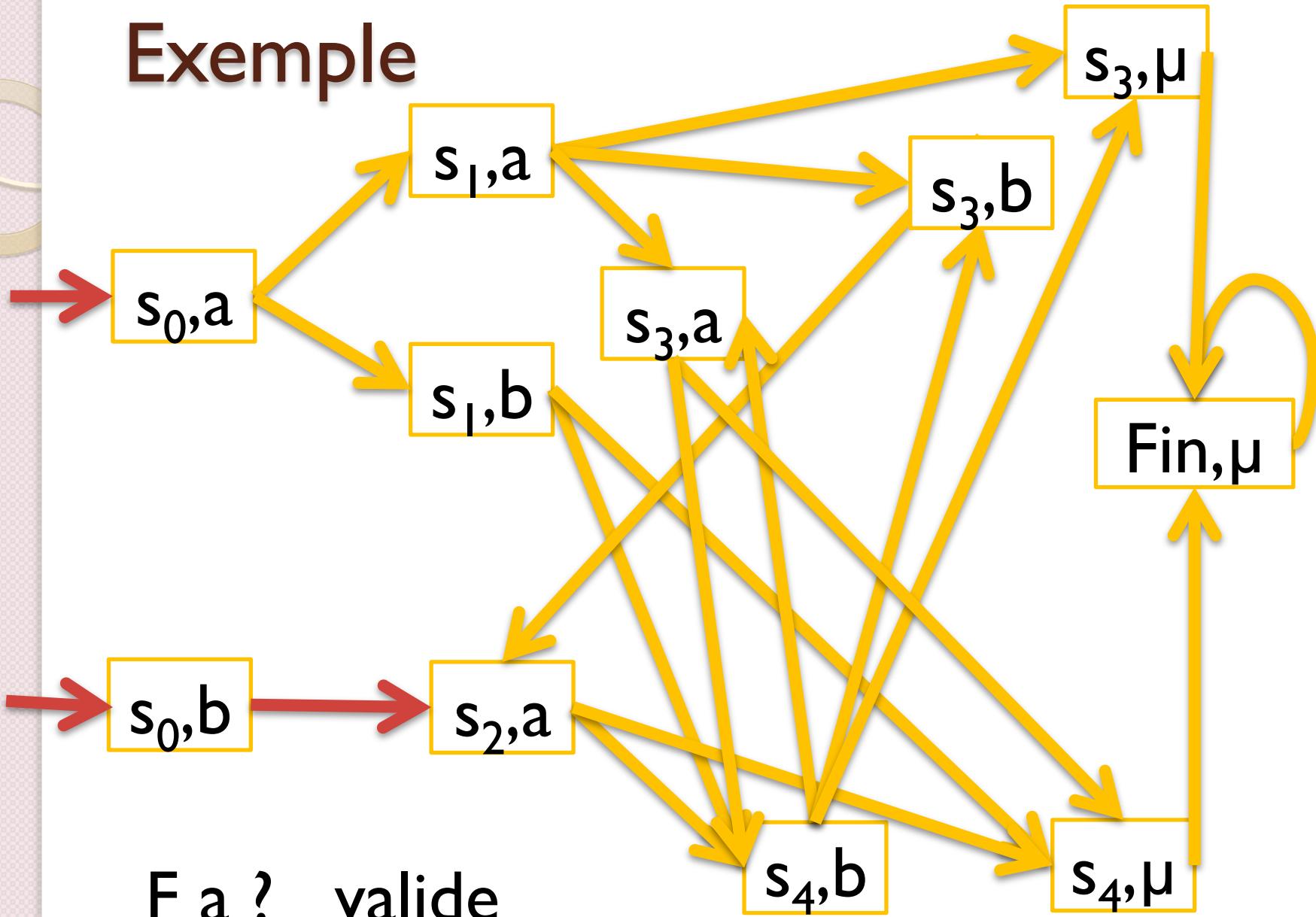
Verification sur un LTS

- En pratique, on ne dessine pas le modèle de Kripke explicitement
- On peut vérifier les chemins directement sur le LTS

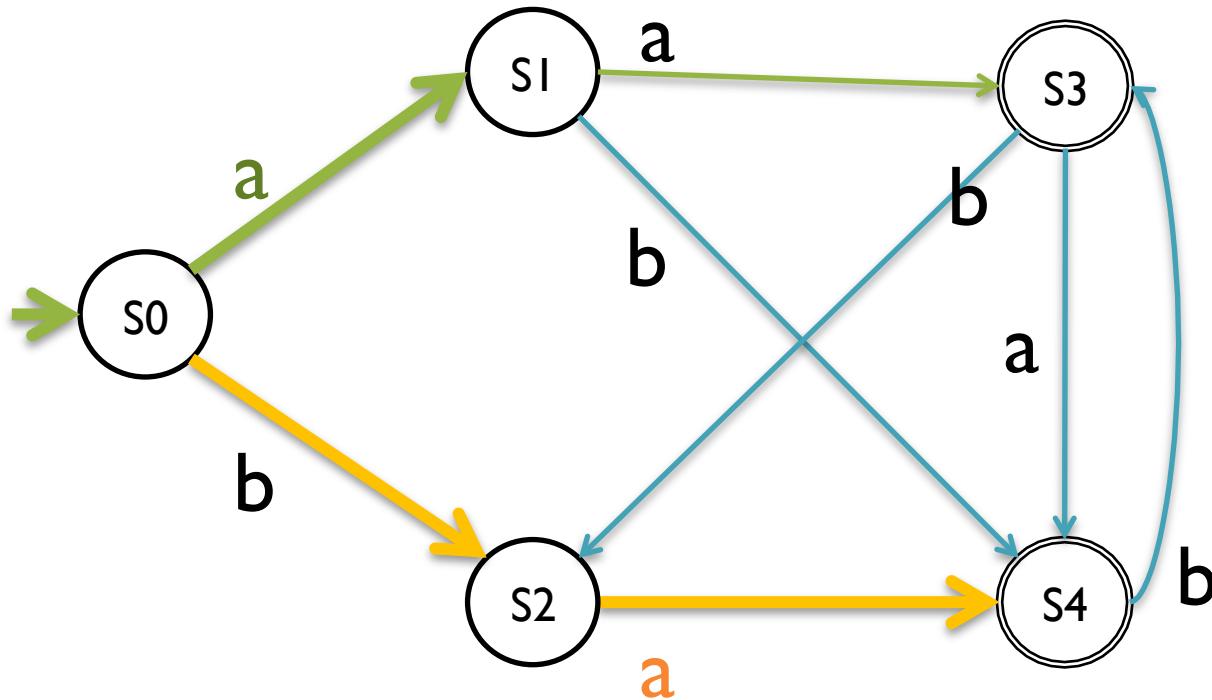
Exemple



Exemple

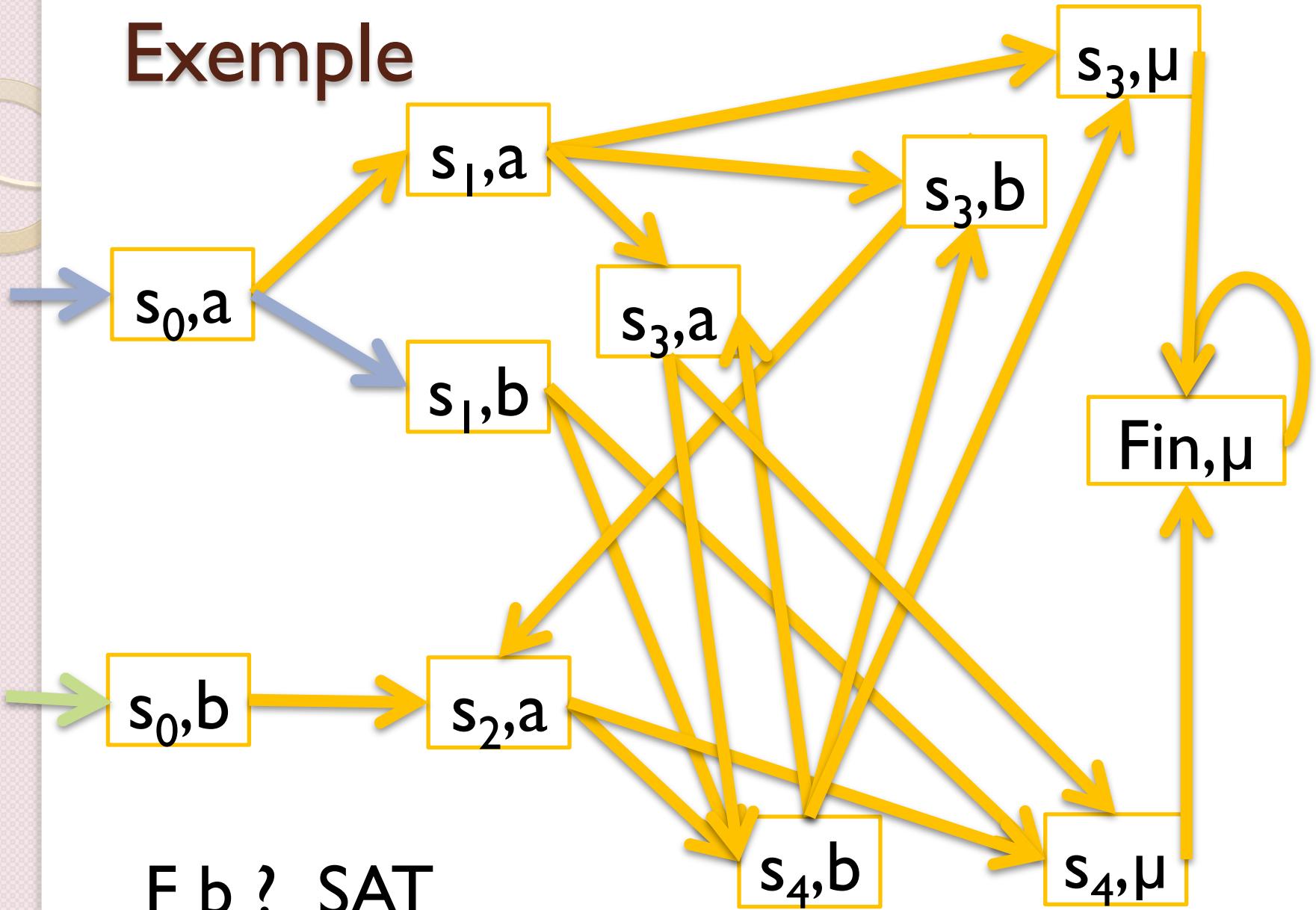


Exemple

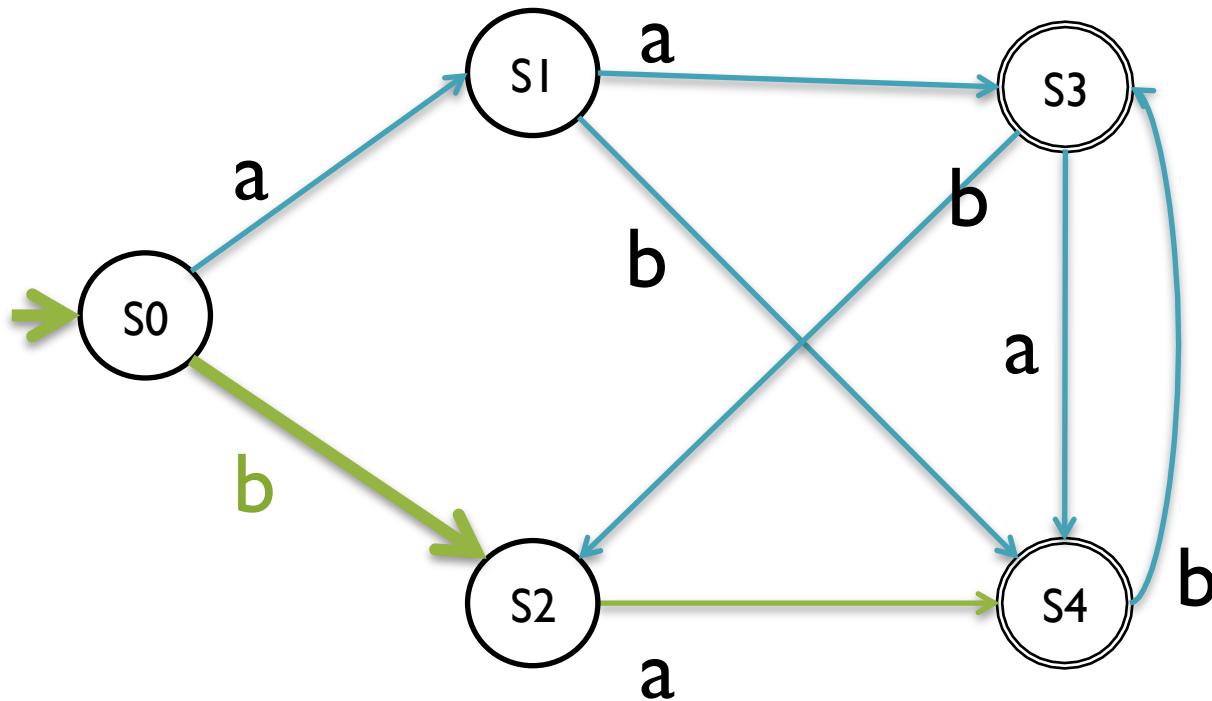


F a ? valide

Exemple

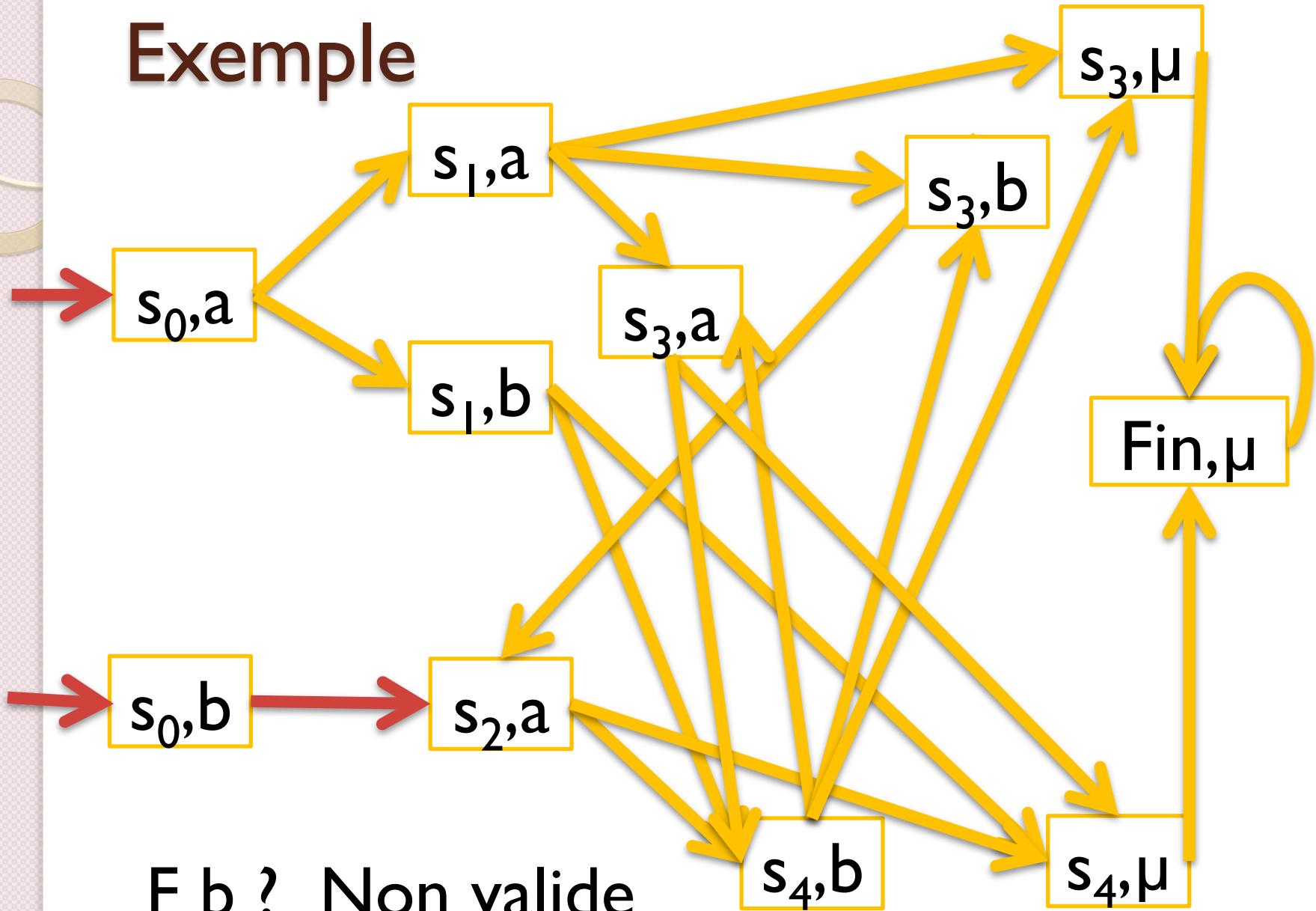


Exemple

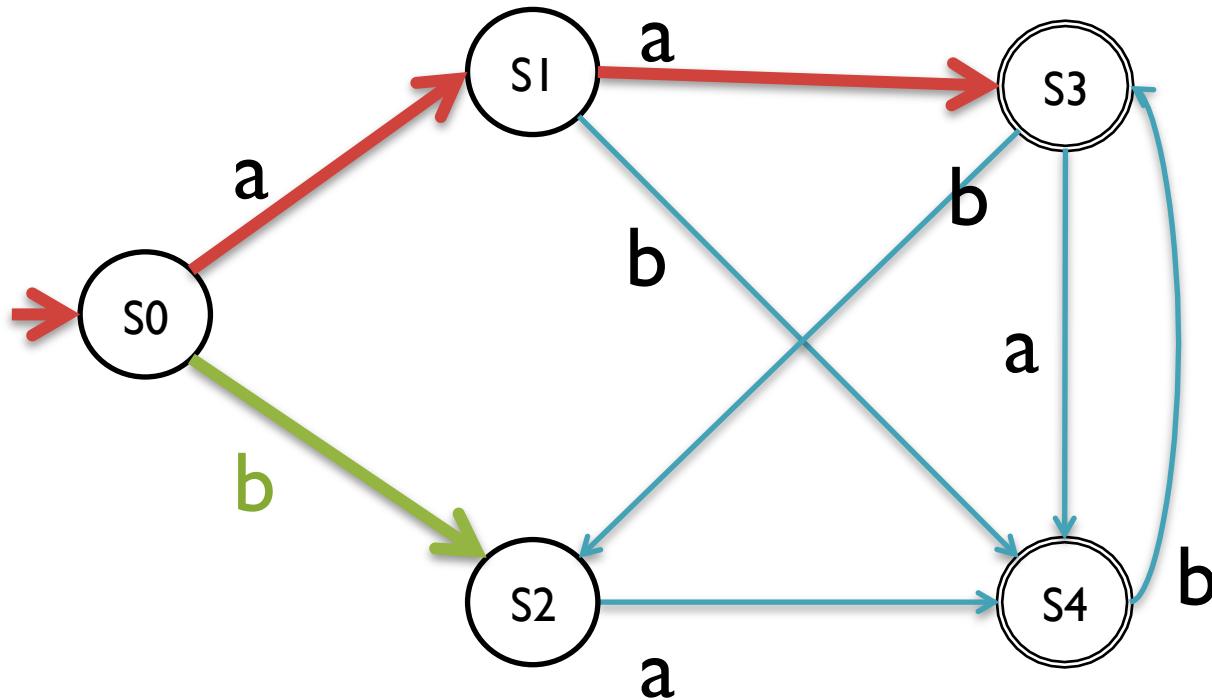


F b ? SAT

Exemple



Exemple



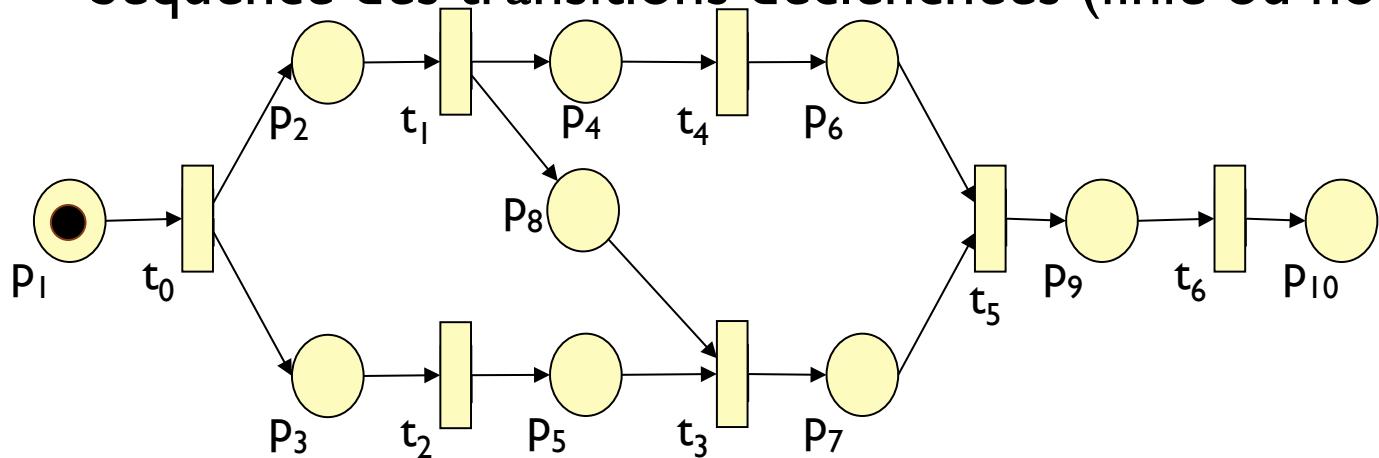
F b ? Non valide

Vérification dans les RdP

- Équivalent à vérification dans le graphe des marquages accessibles (LTS)
- Chemin dans ce graphe correspond à une séquence de transitions franchissable depuis le marquage initial.

LTL et réseaux de Pétri

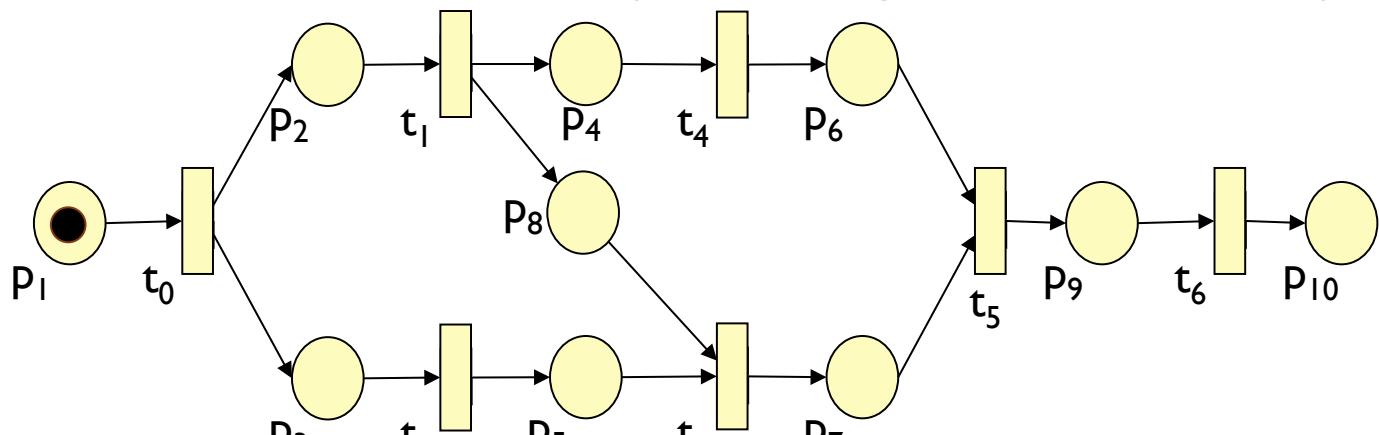
- Trace d'une simulation dans un RdP :
 - Séquence des transitions déclenchées (finie ou non)



- Exemple de Trace possible : $t_0 t_1 t_4 t_2 t_3 t_5 t_6$
- On peut y associer un ensemble de chemins (de longueur infinie) \sqcap définit sur le vocabulaire $V=\{t_0, \dots, t_{10}, \mu\}$ en prolongeant toute trace finie d'un infinité de μ .
- On pose $P=\{t_0, \dots, t_{10}\}$, et $I(t_i)=\{t_i\}$

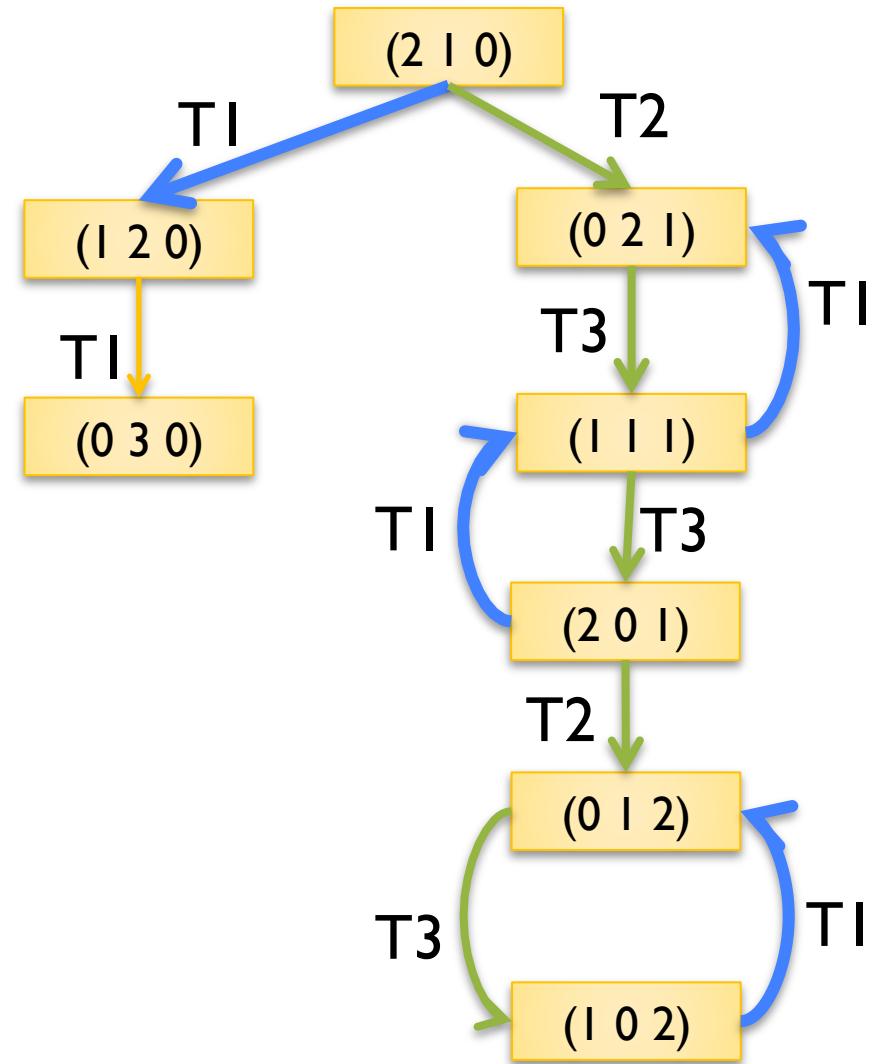
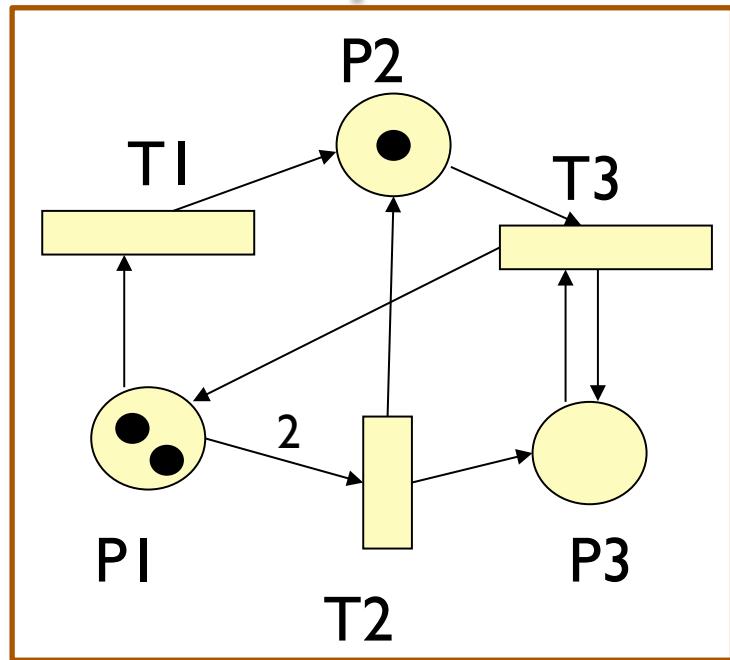
LTL et réseaux de Pétri

- En considérant cela, on peut donc écrire des formules de LTL vérifiables sur des chemins (traces complétées de μ à l'infini).



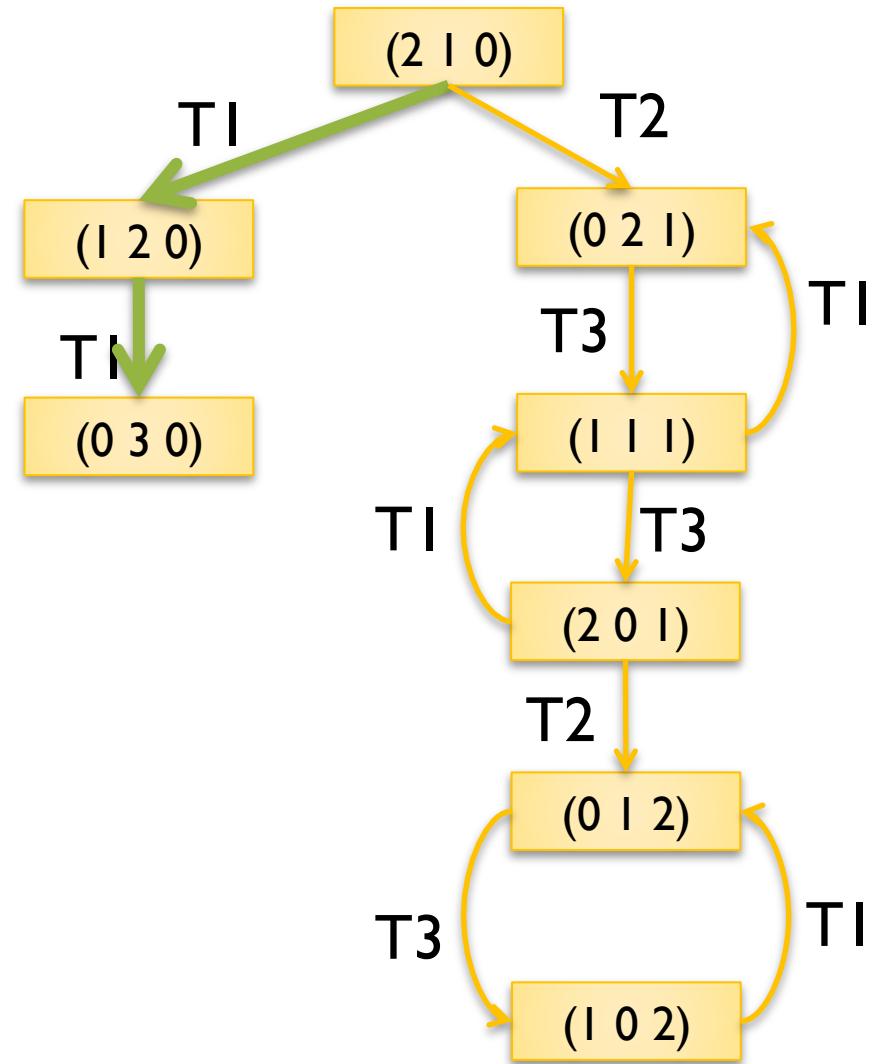
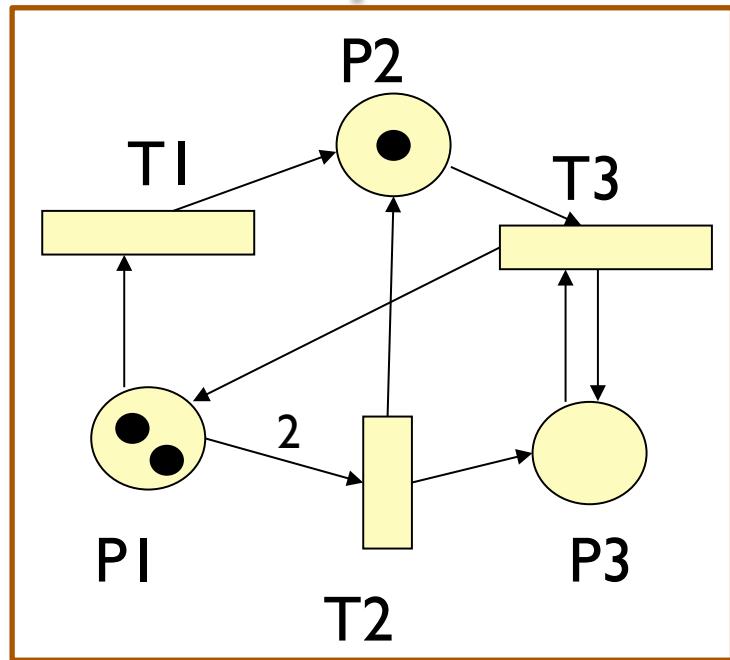
- Exemple $\Pi = t_0 t_1 t_4 t_2 t_3 t_5 t_6 \mu \dots$ vérifie les formules
 - $t_0, \text{OO}t_4, F(t_3 \wedge O t_5), \neg t_3 \not\vee t_1, \neg t_3 \not\vee t_4, G(t_5 \rightarrow O t_6)$
- On dira que le RdP vérifie φ ssi tt chemin formé à partir de ces traces vérifie φ (ie φ valide pour \prod, I)
 - Ex : ce RdP vérifie $t_0, \neg t_3 \not\vee t_1, G(t_5 \rightarrow O t_6)$ mais il ne vérifie pas $\neg t_3 \not\vee t_4, \text{OO}t_4, F(t_3 \wedge O t_5)$ (e.g. trace $t_0 t_1 t_2 t_3 t_4 t_5 t_6$)

Exemple



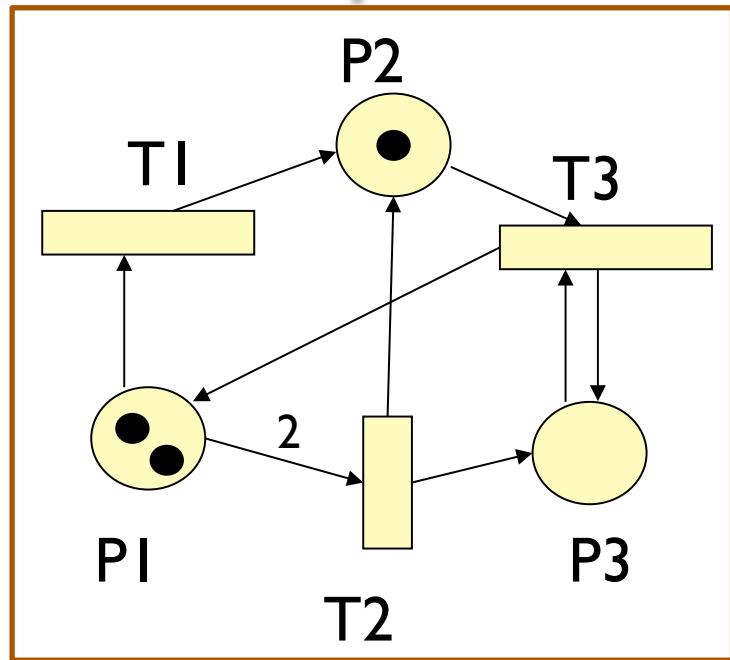
$F T_1$ Valide

Exemple



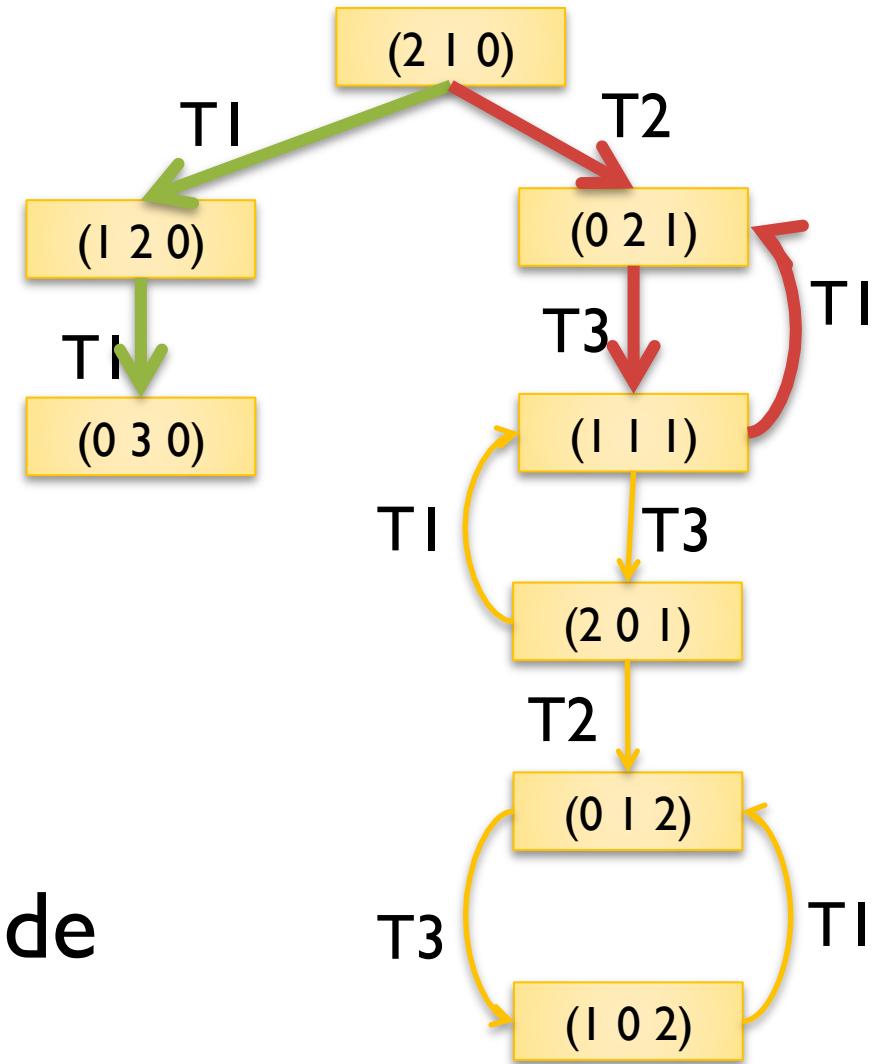
$\text{FG} \neg T_1 \text{ SAT}$

Exemple



$\text{FG} \neg T_1 \text{ SAT}$

$\text{FG} \neg T_1 \text{ non valide}$



Limites de LTL

- LTL implicitement quantifiée universellement sur tous les chemins

RdP marqué sans blocage ssi il vérifie

$$G \bigvee_{t \in T} t$$

- Quantification existentielle sur les chemins ?
Simulable via négation

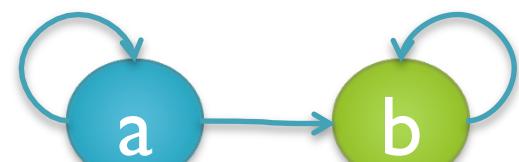
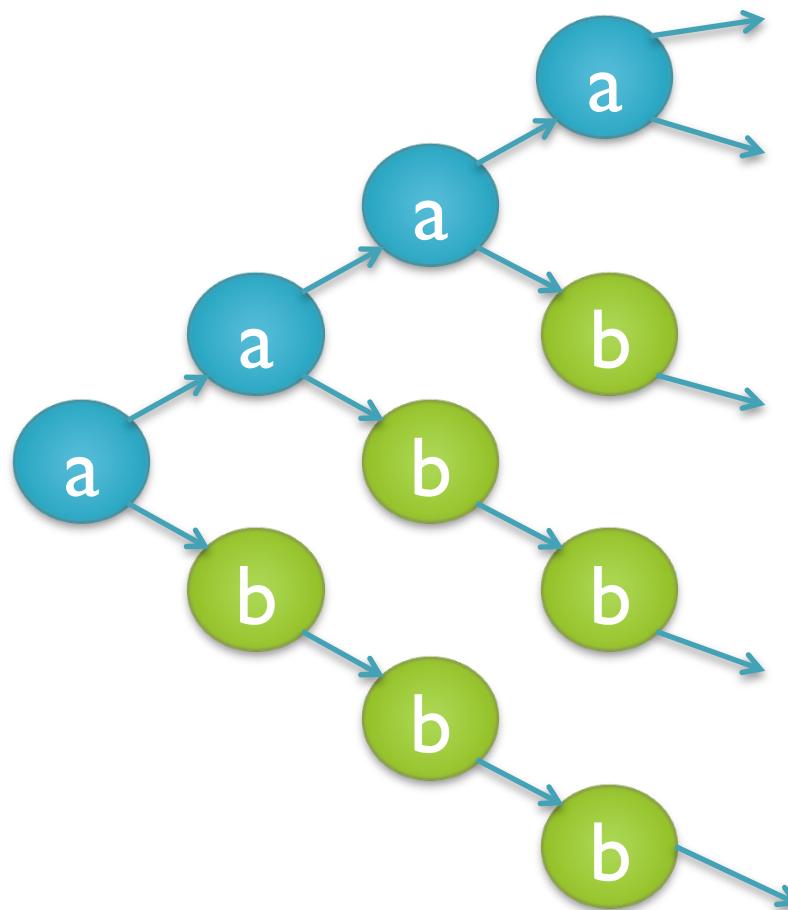
RdP marqué : t quasi vivant ssi il ne vérifie pas $G\neg t$

- Mais quantifications seulement possible en début
Typiquement, on ne peut pas exprimer qu'un RdP est vivant en LTL

CTL et CTL*

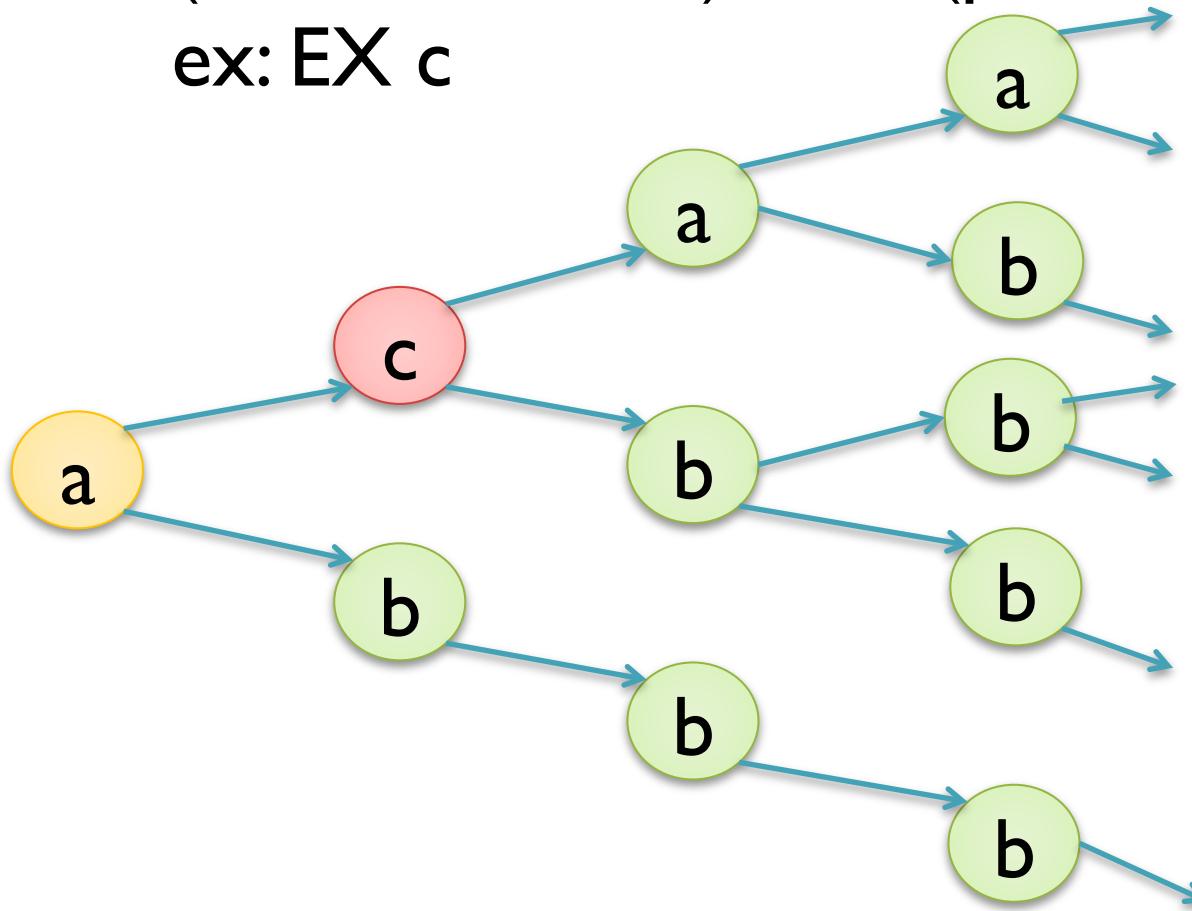
Computation Tree Logic (CTL)

- Idée : au lieu d'un ligne temporel, on considère un arbre de possible



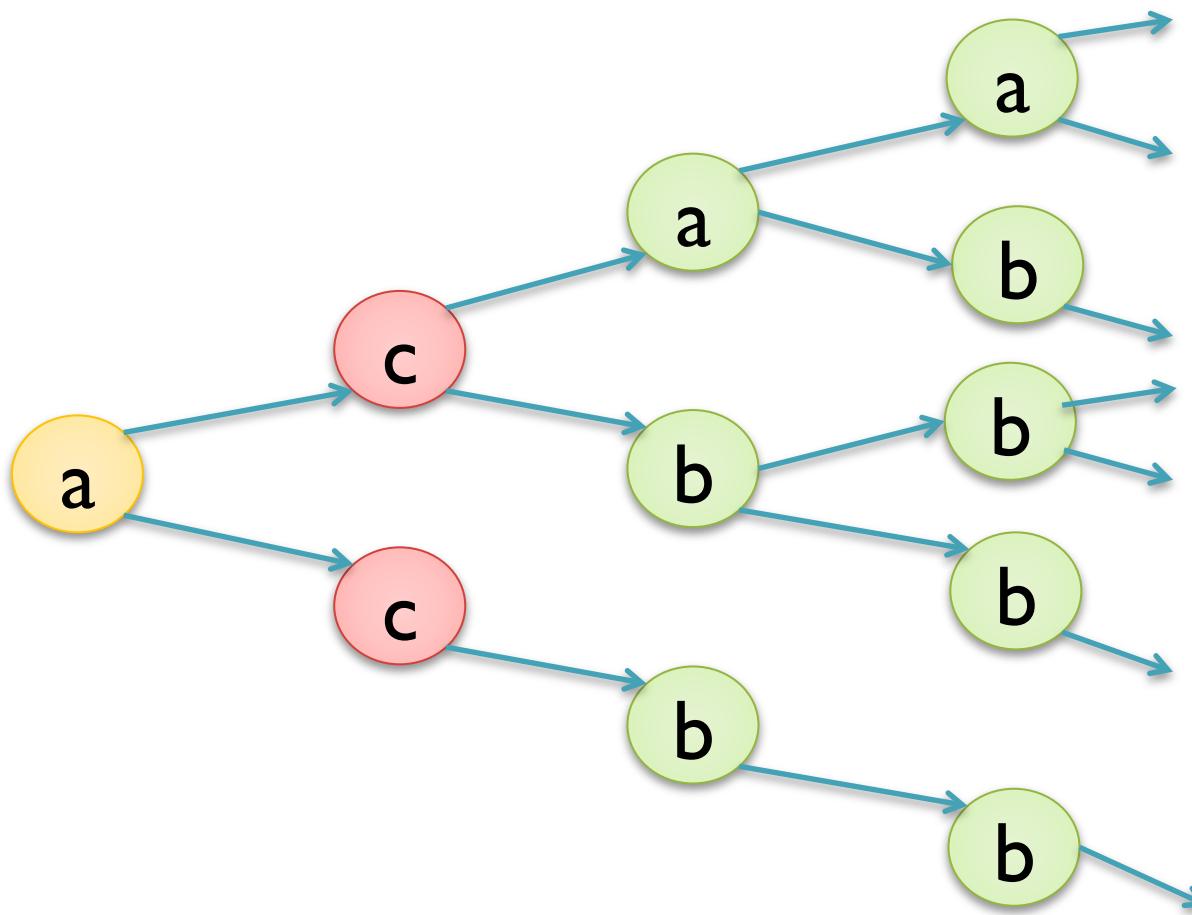
Computation Tree Logic (CTL)

- Opérateurs (X, U, G, F) quantifiés par A (inévitablement) ou E (possiblement) :
ex: $EX\ c$



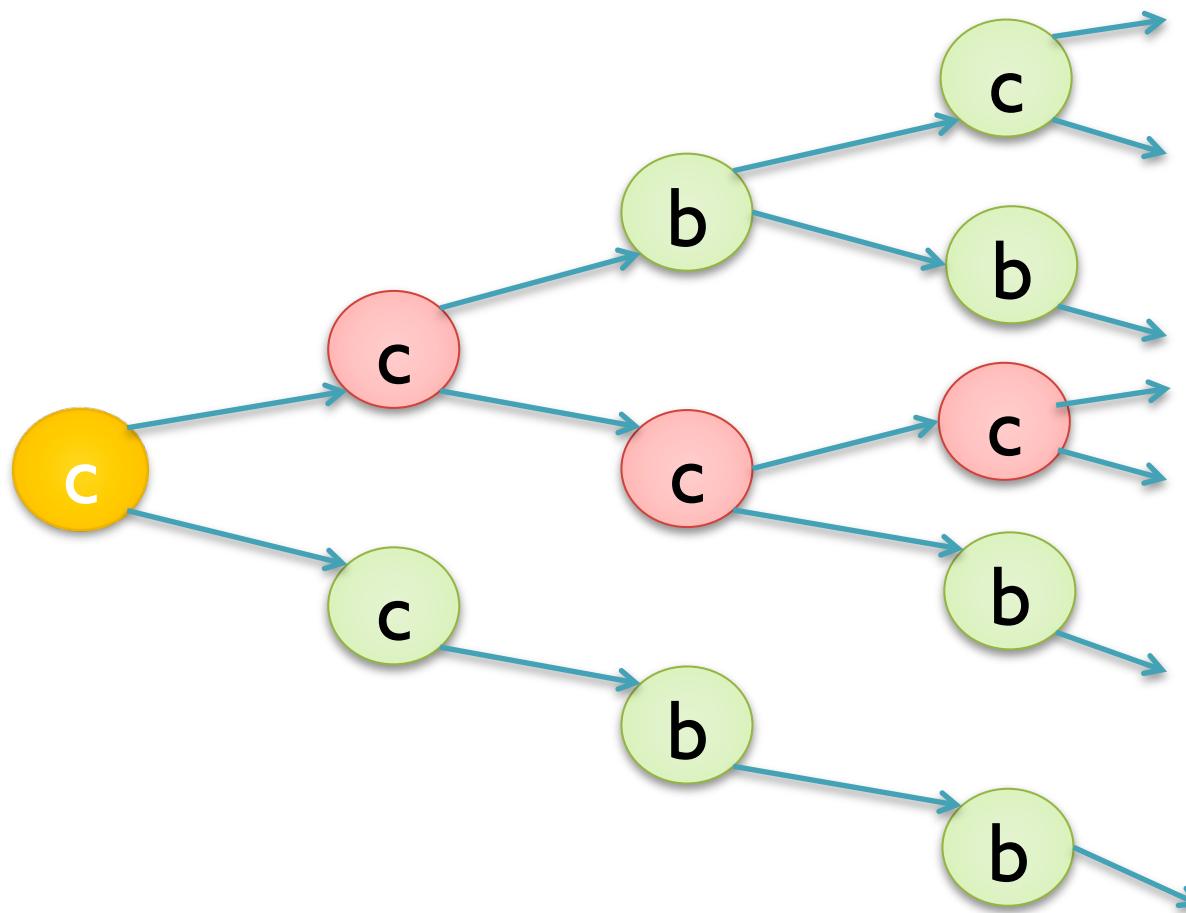
Computation Tree Logic (CTL)

- Opérateur : AX c



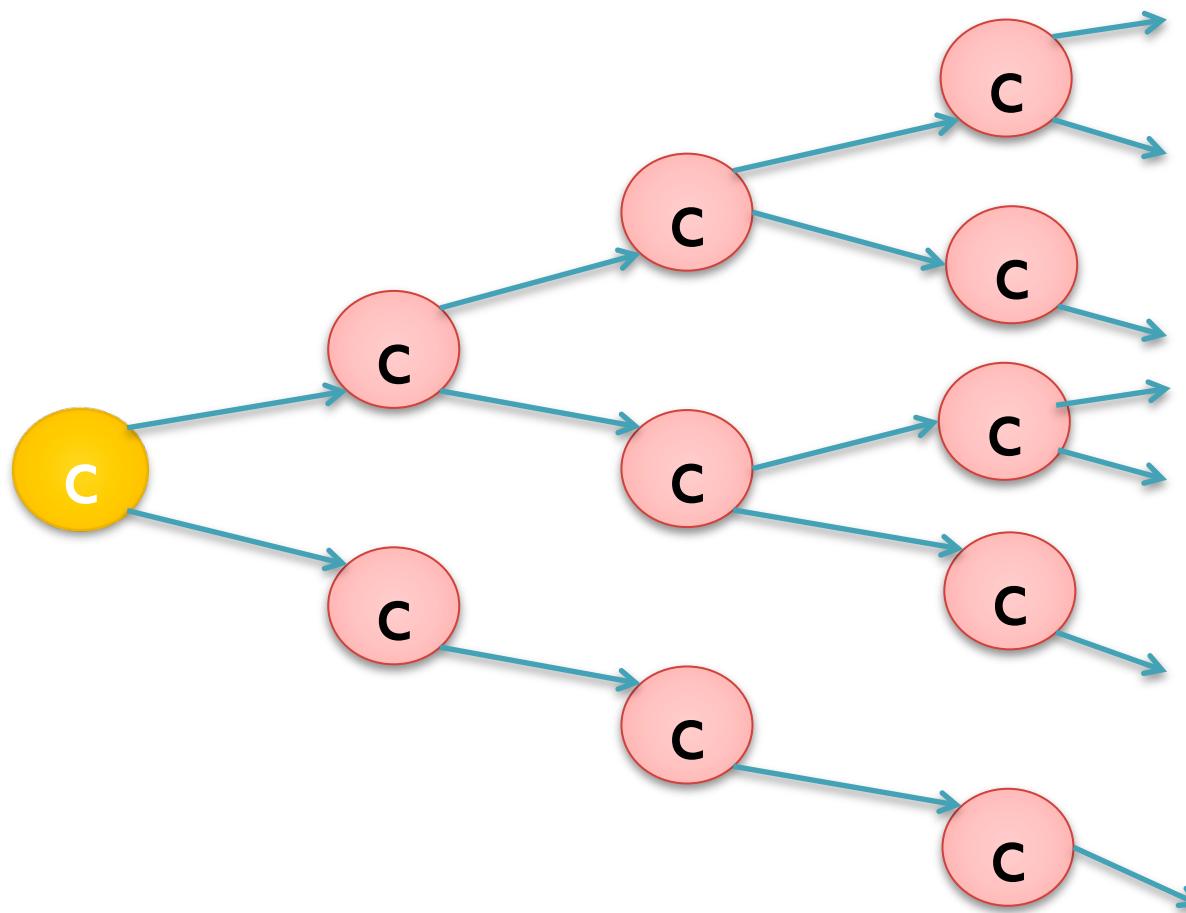
Computation Tree Logic (CTL)

- Opérateur : EG c



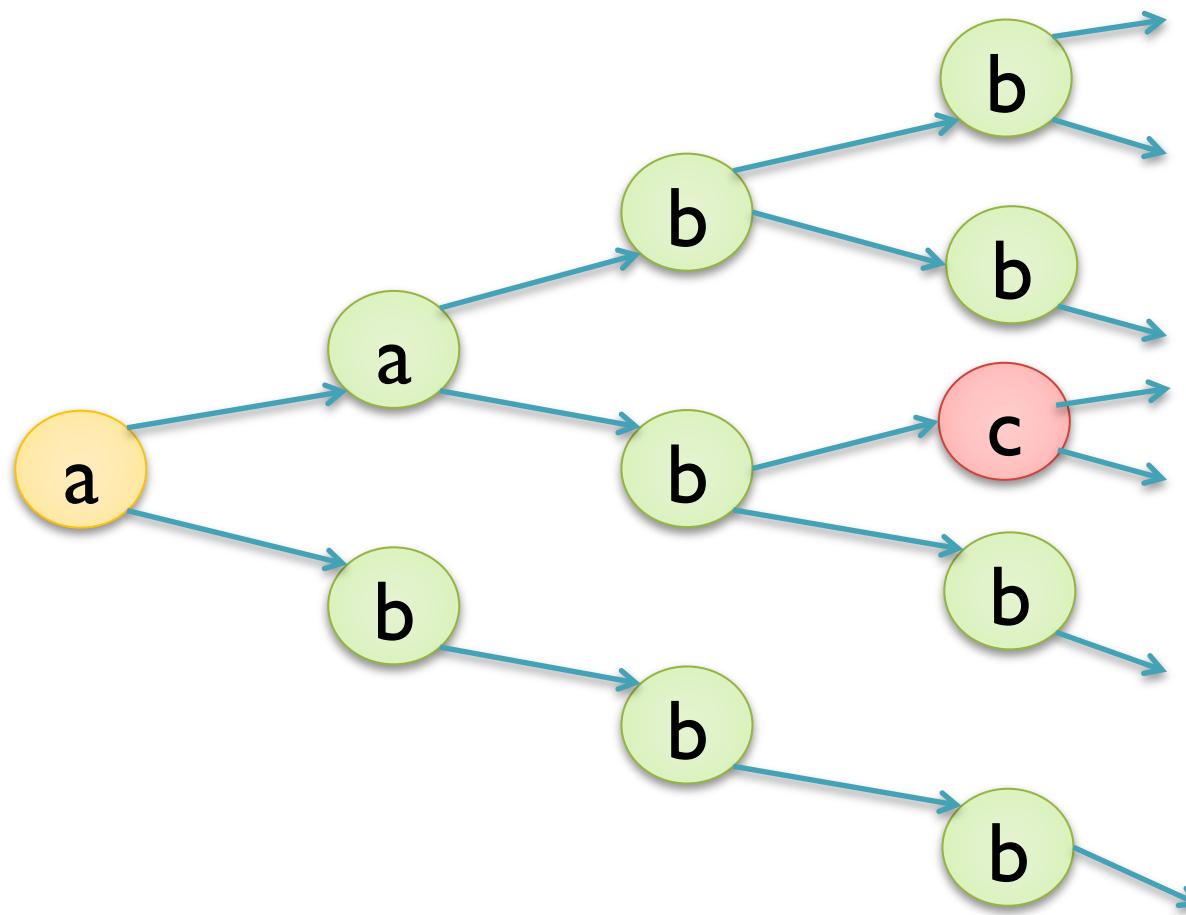
Computation Tree Logic (CTL)

- Opérateur : AG c



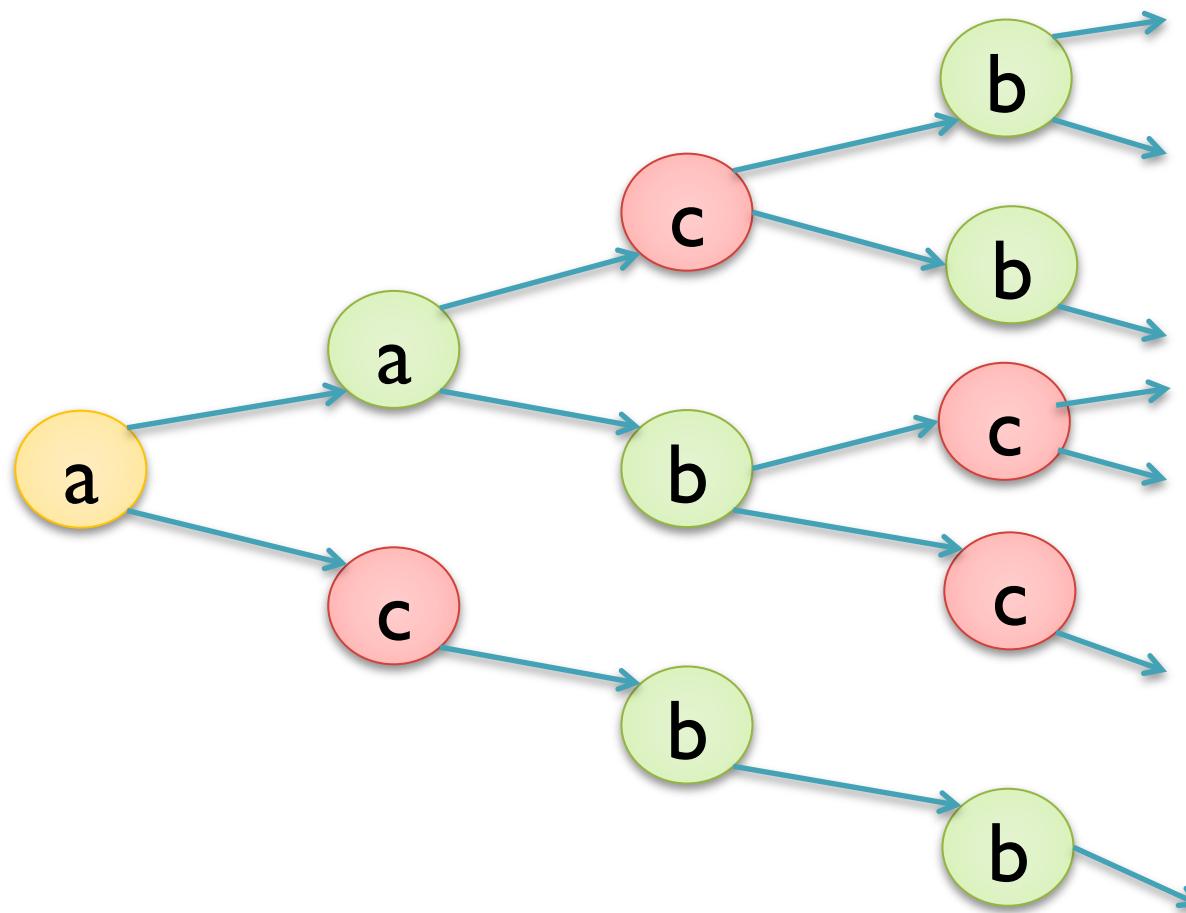
Computation Tree Logic (CTL)

- Opérateur : EF c



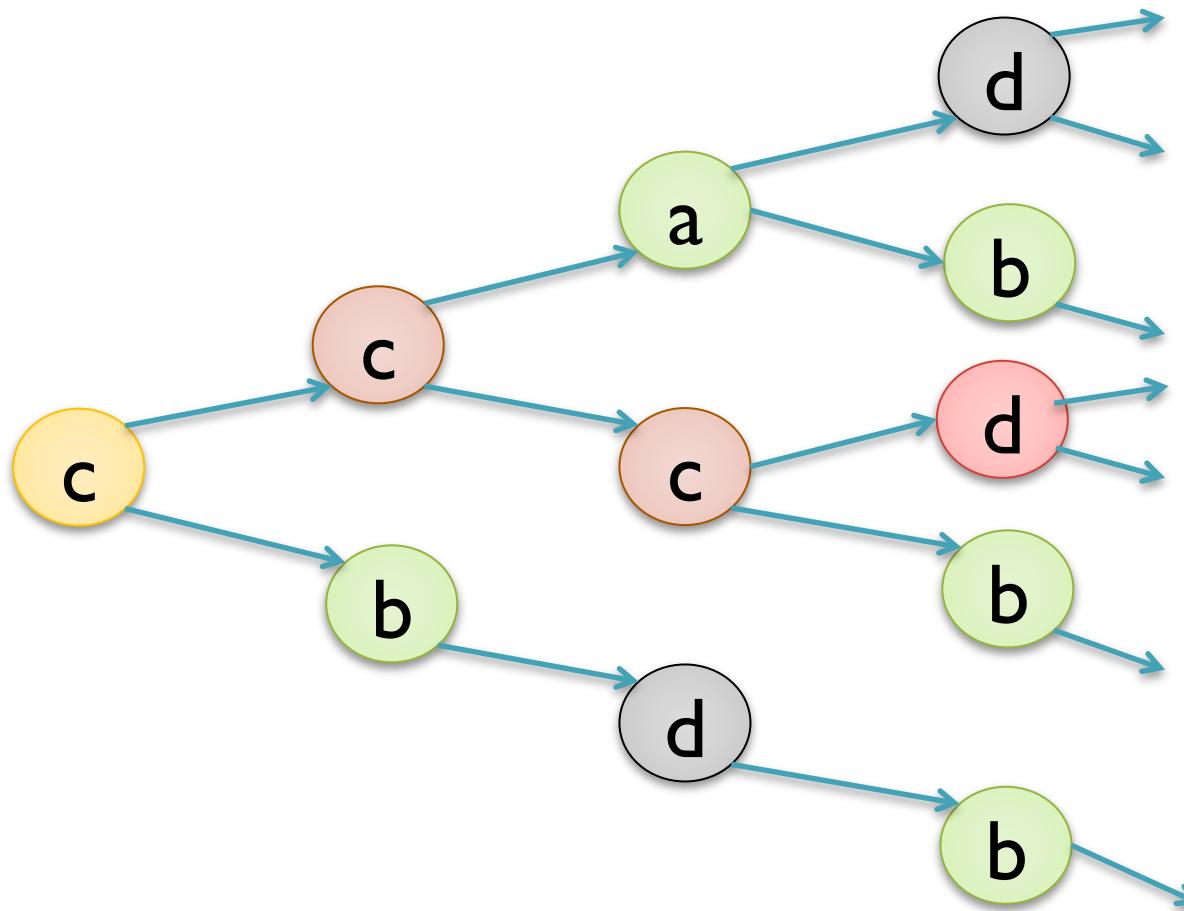
Computation Tree Logic (CTL)

- Opérateur : AF c



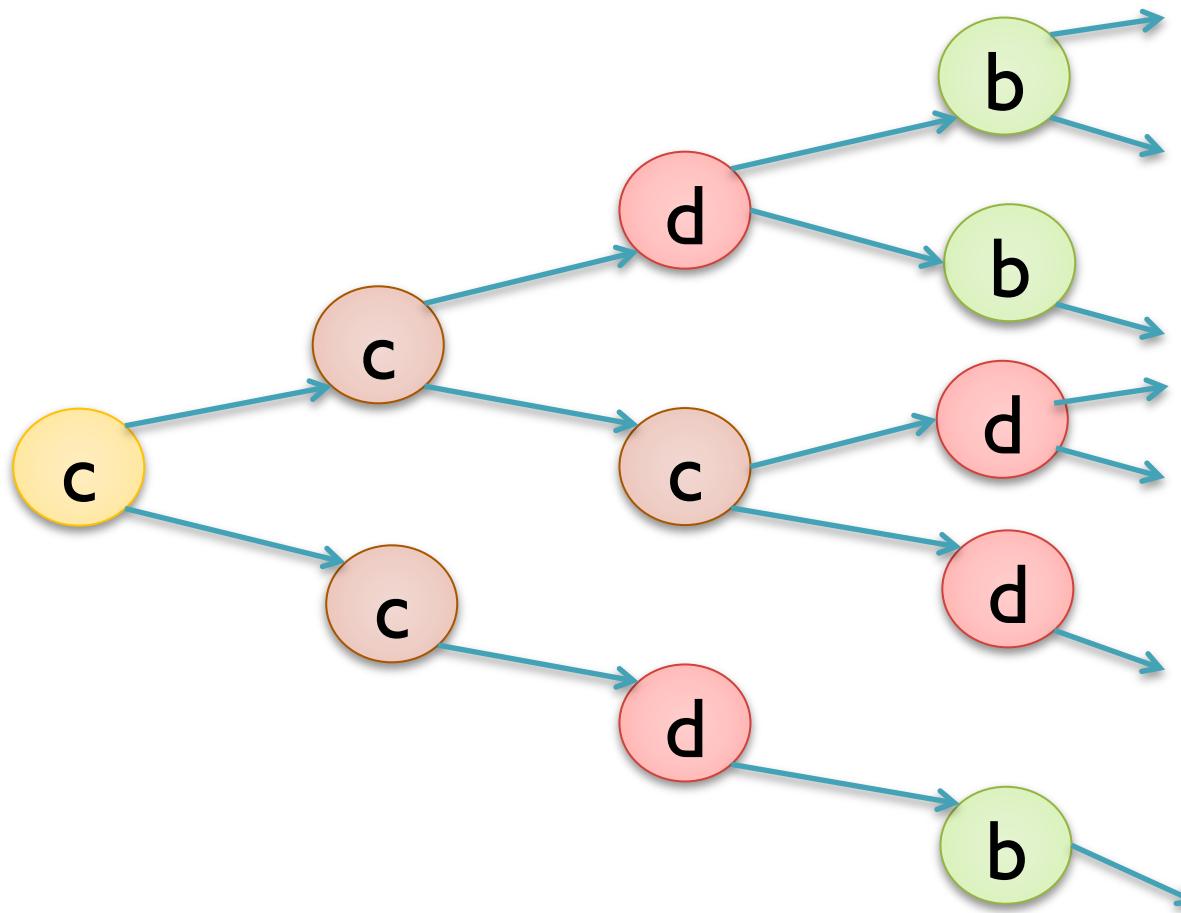
Computation Tree Logic (CTL)

- Opérateur : $E(cUd)$



Computation Tree Logic (CTL)

- Opérateur : $A(c \cup d)$



CTL

- Syntaxe

$$\varphi ::= p | \neg \varphi | \varphi \wedge \varphi | \varphi \vee \varphi | \dots$$
$$| EX\varphi | AX\varphi | EG\varphi | AG\varphi | EF\varphi | AF\varphi | E(\varphi U \varphi) | A(\varphi U \varphi)$$

- Opérateurs minimaux :

$$\{\neg, \wedge, EX, EG, EU\}$$

- $EF\phi == E(true \cup \phi)$
- $AX\phi == \neg EX\neg\phi$
- $AG\phi == \neg EF(\neg\phi)$
- $AF\phi == \neg EG(\neg\phi)$
- $A[\phi \cup \psi] == \neg(E[(\neg\psi) \cup \neg(\phi \vee \psi)] \vee EG\neg\psi)$

CTL

- Sémantique

Modèle de Kripke $\langle M, R, I \rangle$ (avec sérialité)

Chemins depuis w : $\prod_w = \{w_0, w_1, w_2, \dots | w_0 = w\text{, et pour tt } i, w_i R w_{i+1}\}$

$M, w \models p$ ssi $w \in I(p)$

$M, w \models \neg\varphi$ ssi $M, w \not\models \varphi$

$M, w \models \varphi \wedge \psi$ ssi $M, w \models \varphi$ et $M, w \models \psi$

...

$M, w \models EX\varphi$ ssi $\exists \pi \in \Pi_w, (M, \pi(1) \models \varphi)$

$M, w \models EF\varphi$ ssi $\exists \pi \in \Pi_w, \exists i \in \mathbb{N}, (M, \pi(i) \models \varphi)$

$M, w \models EG\varphi$ ssi $\exists \pi \in \Pi_w, \forall i \in \mathbb{N}, (M, \pi(i) \models \varphi)$

$M, w \models E(\varphi U \psi)$ ssi $\exists \pi \in \Pi_w, \exists i \in \mathbb{N}, (M, \pi(i) \models \psi)$
et $\forall k \in \{0, \dots, i-1\}, (M, \pi(k) \models \varphi)$

$M, w \models AX\varphi$ ssi $\forall \pi \in \Pi_w, (M, \pi(1) \models \varphi)$

$M, w \models AF\varphi$ ssi $\forall \pi \in \Pi_w, \exists i \in \mathbb{N}, (M, \pi(i) \models \varphi)$

$M, w \models AG\varphi$ ssi $\forall \pi \in \Pi_w, \forall i \in \mathbb{N}, (M, \pi(i) \models \varphi)$

$M, w \models A(\varphi U \psi)$ ssi $\forall \pi \in \Pi_w, \exists i \in \mathbb{N}, (M, \pi(i) \models \psi)$
et $\forall k \in \{0, \dots, i-1\}, (M, \pi(k) \models \varphi)$

CTL

- Sémantique

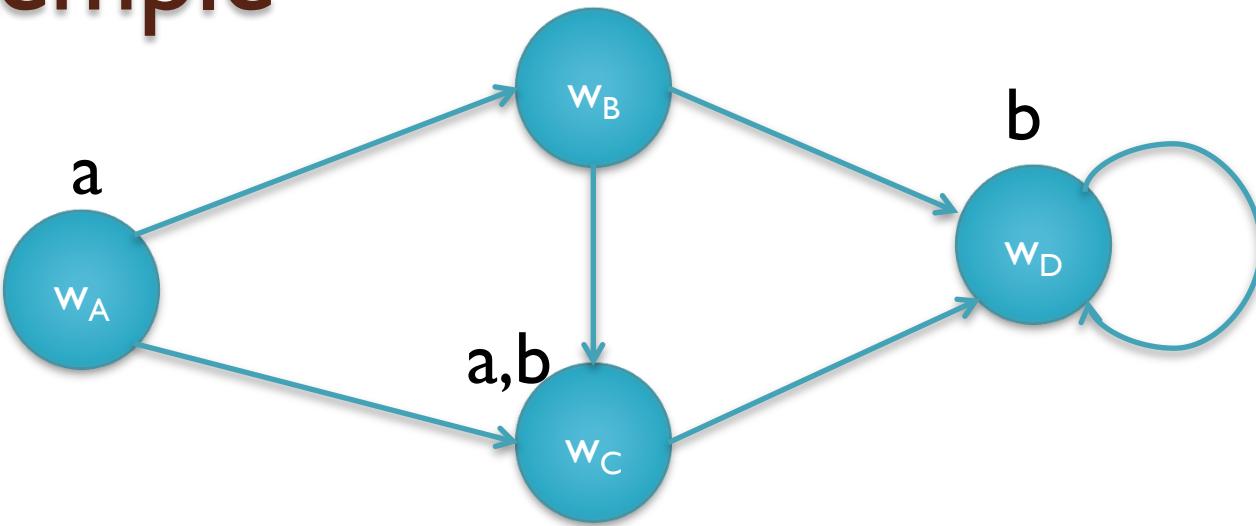
Modèle de Kripke $\langle M, R, I \rangle$ (avec sérialité)

Chemins depuis w : $\prod_w = \{w_0, w_1, w_2, \dots | w_0 = w,$
et pour tt i , $w_i R w_{i+1}\}$

- Validité/satisfiabilité (comme logique épistémique)

- Formule valide pour M ssi elle est vérifiée dans tous les mondes
- Formule satisfiable pour M ssi elle est vérifiée dans au moins un monde

Exemple



$$\mathcal{P} = \{a, b\}. I(a) = \{w_A, w_C\}, I(b) = \{w_C, w_D\}$$

- (AF AG b) est valide dans \mathcal{M}
- (AG b) SAT et non valide : vérifiée dans w_D et w_C , mais pas dans w_A ni w_B
- (EX a) vérifiée dans w_A .
- (AX a) non vérifiée dans w_A .

CTL et Réseau de Pétri

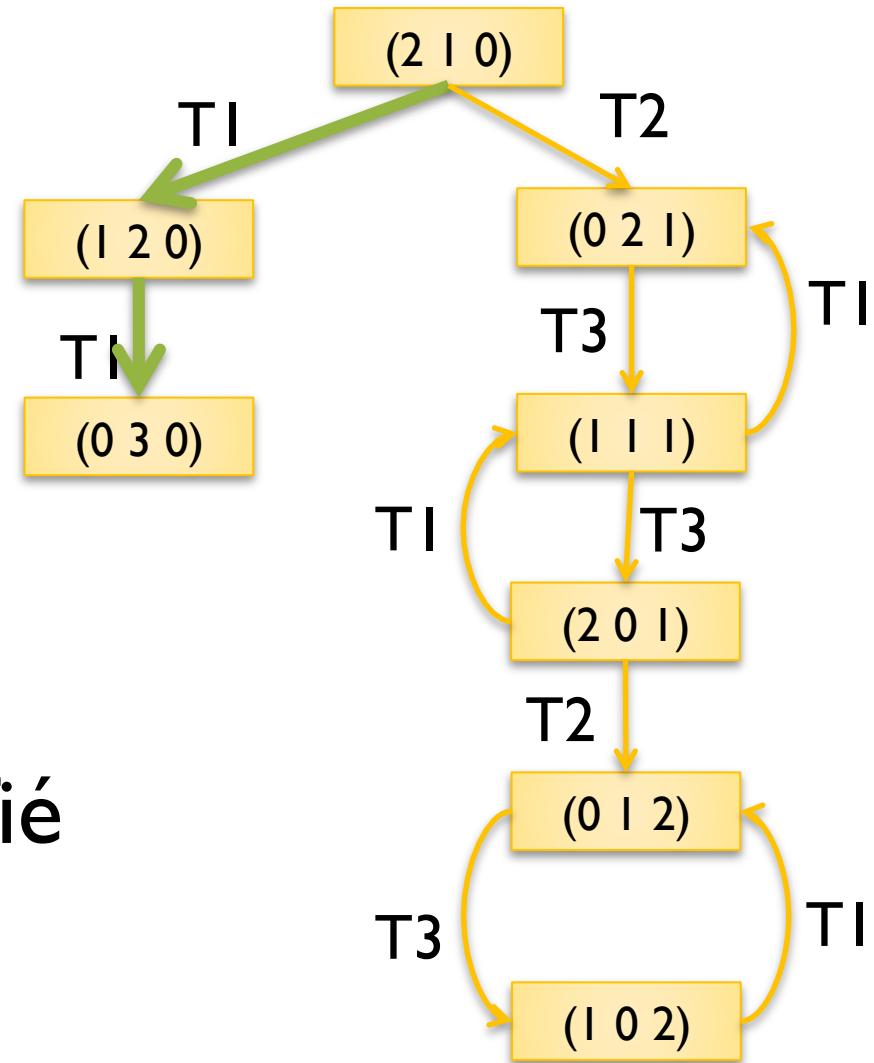
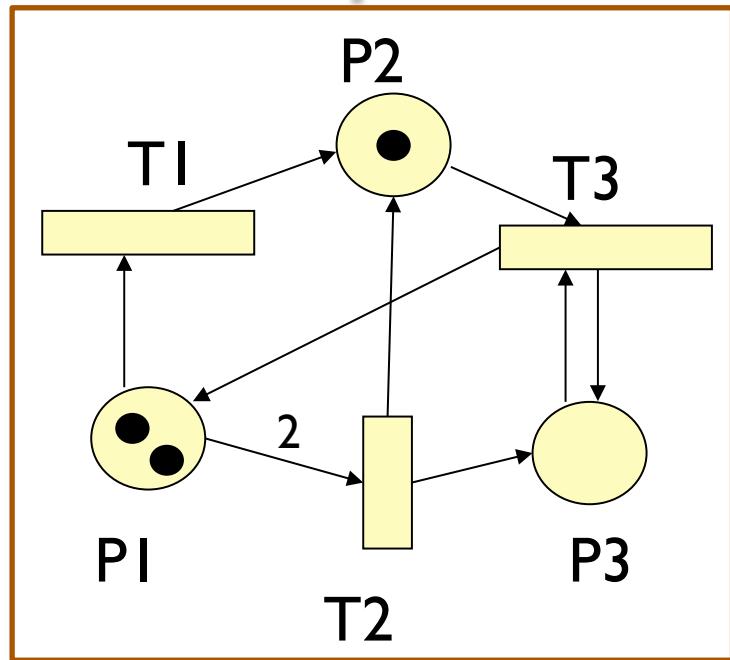
- Sémantique

Modèle de Kripke $\langle M, R, I \rangle$ (avec sérialité)

Chemins depuis w : $\prod_w = \{w_0, w_1, w_2, \dots | w_0 = w,$
et pour tt i , $w_i R w_{i+1}\}$

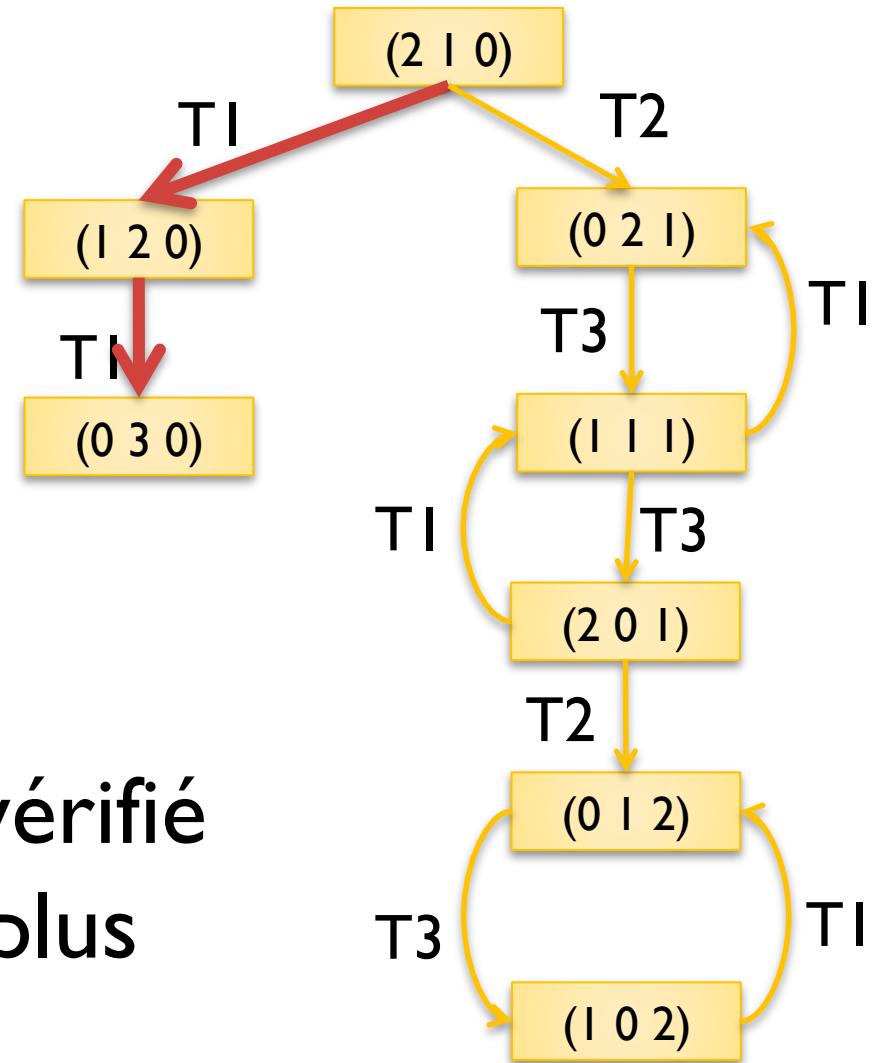
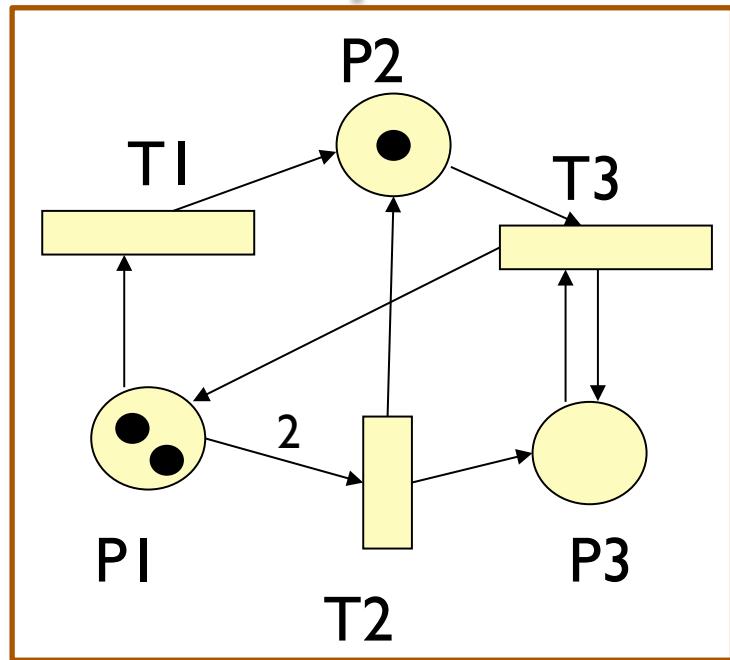
- Un RdP marqué vérifie une formule ssi cette formule est vérifiée dans le marquage initiale (ie, dans tout monde de la forme $(M_0, ?)$ de la structure de Kripke correspondante)

Exemple



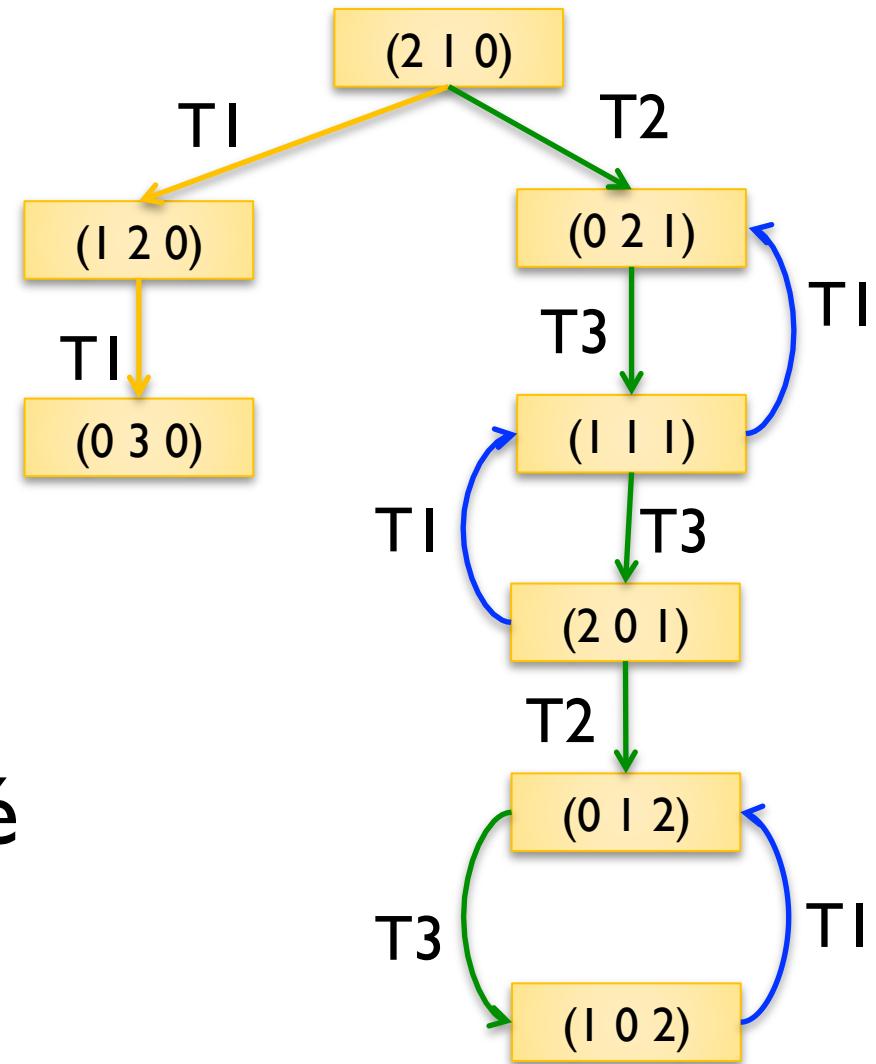
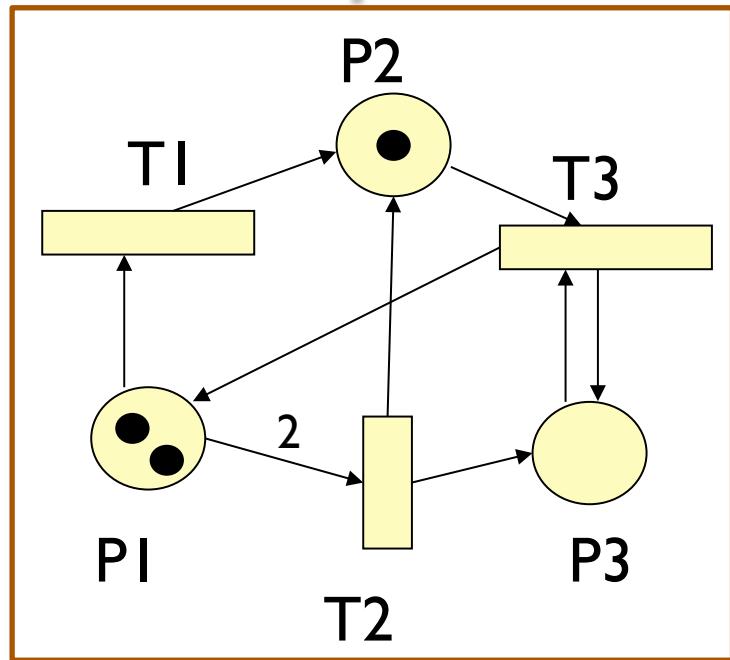
$\text{EF AG} \neg T_1$ vérifié

Exemple



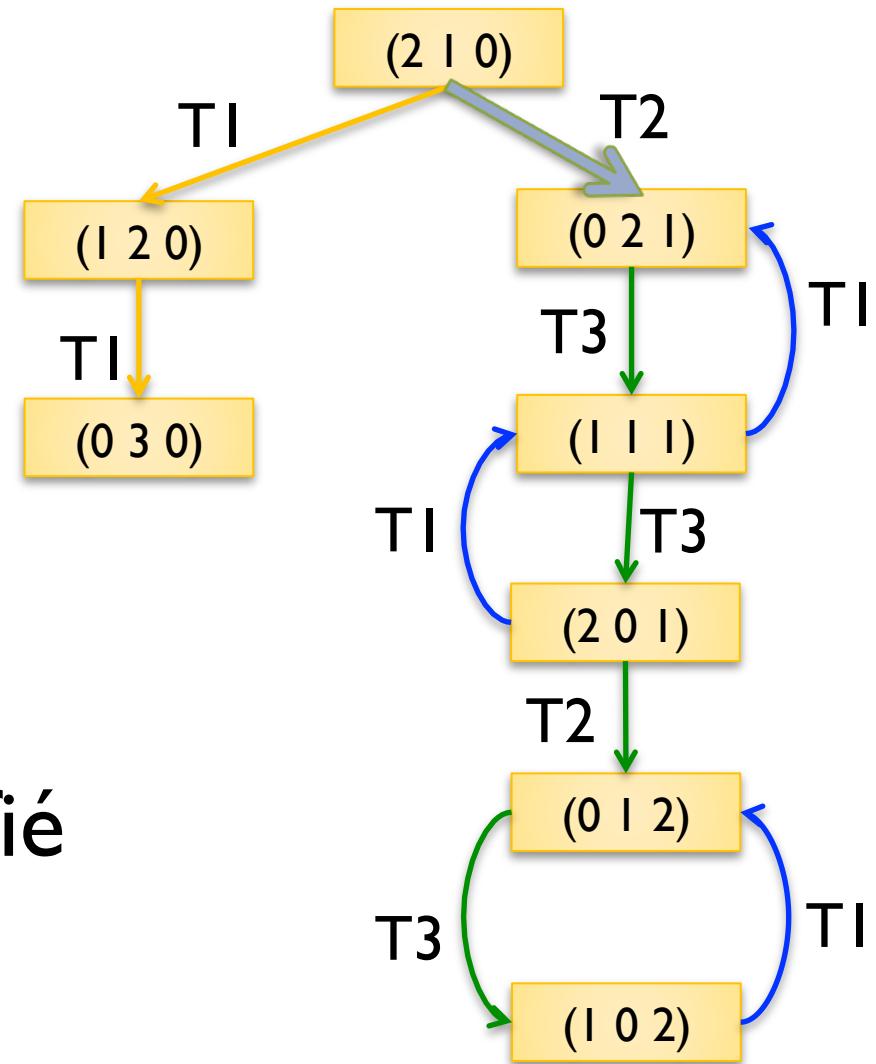
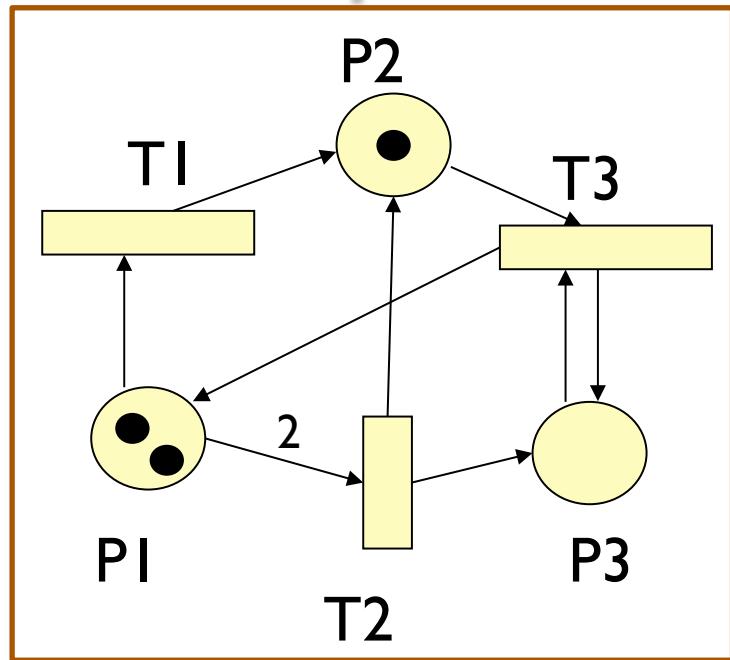
AG EFT₁ non vérifié
AG AFT₁ non plus

Exemple



EG EFT₁ vérifié

Exemple



EX AG AFT₁ vérifié

CTL

- Négations : $\neg A\Theta = E\neg\Theta$ (où $\neg G\neg=F$ et $\neg X\neg=X$)
 - $\neg AX p (= E\neg X p) = EX \neg p,$
 - $\neg AG p (= E\neg G p) = EF \neg p$
 - $\neg AF p = EG \neg p$
- Expansion
 - $AG p = p \wedge AX AG p$
 - $EG p = p \wedge EX EG p$
 - $AF p = p \vee AX AF p$
 - $EF p = p \vee EX EG p$

CTL et LTL

- CTL exprime choses inexprimable en LTL

(typiquement quand mélange de E et A, ou E ailleurs qu'en première position)

Ex : transition t vivante : AG EF t

(on prend la conjonction sur tous les t pour que le RdP marqué soit vivant)

- LTL peut toutefois aussi exprimer choses inexprimable en CTL :

(typiquement si sélection de chemins avec une propriété)

Ex : GF t₁ → GF t₂

- Intersection non vide :

$$G\neg p \quad < - > \quad AG \neg p$$

$$F p \quad < - > \quad AF p$$

$$G(p \rightarrow F q) \quad < - > \quad AG(p \rightarrow AF q)$$

$$GF p \quad < - > \quad AG AF p$$

CTL*

- Unifie LTL et CTL
- Idée : séparer A,E et X,G,F,U
- 2 type de formules entrelacées (via opérateur E,A)
 - Formules d'état (state formulae) vérifiées sur des mondes : ce sont les formules de base
 - Formules de chemins (path formulae) vérifiées sur des chemins, toujours précédées d'un E ou d'un A

CTL* - Syntaxe

- Sur langage prop P, on définit

$$\Phi ::= \perp | \top | p | \neg \Phi | \Phi \wedge \Phi | \Phi \vee \Phi | A\varphi | E\varphi$$
$$\varphi ::= \Phi | \neg \varphi | \varphi \wedge \varphi | \varphi \vee \varphi | X\varphi | \varphi U \varphi | F\varphi | G\varphi$$

- Où

- p formule atomique (dans P)
- Φ formule d'état (=formule de CTL*)
- φ formule de chemin

CTL* - Syntaxe

- Sur langage prop P, on définit

$$\Phi ::= \perp | \top | p | \neg \Phi | \Phi \wedge \Phi | \Phi \vee \Phi | A\varphi | E\varphi$$
$$\varphi ::= \Phi | \neg \varphi | \varphi \wedge \varphi | \varphi \vee \varphi | X\varphi | \varphi U \varphi | F\varphi | G\varphi$$

- Opérateurs d'état :

- Opérateur classique (\neg, \vee, \dots)

- Opérateur $A\varphi$ et $E\varphi$: quantificateur universelles et existentielle sur des chemins

- Opérateurs de chemins :

- Opérateurs classiques

- Opérateurs temporel modaux définis sur un chemin : X, F, G, U (comme en LTL)

CTL* - Sémantique

- Formules d'états

$$\mathcal{M}, w \models p \text{ ssi } w \in I(p)$$

$$\mathcal{M}, w \models \neg\Phi \text{ ssi } \mathcal{M}, w \not\models \Phi$$

...

$$\mathcal{M}, w \models A\varphi \text{ ssi } \forall \pi \in \Pi_w, \mathcal{M}, \pi \models \varphi$$

$$\mathcal{M}, w \models E\varphi \text{ ssi } \exists \pi \in \Pi_w, \mathcal{M}, \pi \models \varphi$$

- Formule de chemin

$$\mathcal{M}, \pi \models \Phi \text{ ssi } \mathcal{M}, \pi(0) \models \Phi$$

$$\mathcal{M}, \pi \models \neg\varphi \text{ ssi } \mathcal{M}, \pi \not\models \varphi \dots$$

$$\mathcal{M}, \pi \models X\varphi \text{ ssi } \mathcal{M}, \pi^1 \models \varphi$$

$$\mathcal{M}, \pi \models F\varphi \text{ ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models G\varphi \text{ ssi } \forall i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models \varphi U \psi \text{ ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \psi \text{ et}$$

$$\forall k \in \{0, \dots, i-1\}, \mathcal{M}, \pi^k \models \varphi$$

CTL* - Sémantique

- Formules d'états

$$\mathcal{M}, w \models p \text{ ssi } w \in I(p)$$

$$\mathcal{M}, w \models \neg\Phi \text{ ssi } \mathcal{M}, w \not\models \Phi$$

...

$$\mathcal{M}, w \models A\varphi \text{ ssi } \forall \pi \in \Pi_w, \mathcal{M}, \pi \models \varphi$$

$$\mathcal{M}, w \models E\varphi \text{ ssi } \exists \pi \in \Pi_w, \mathcal{M}, \pi \models \varphi$$

- Formule de chemin

$$\mathcal{M}, \pi \models \Phi \text{ ssi } \mathcal{M}, \pi(0) \models \Phi$$

$$\mathcal{M}, \pi \models \neg\varphi \text{ ssi } \mathcal{M}, \pi \not\models \varphi \dots$$

$$\mathcal{M}, \pi \models X\varphi \text{ ssi } \mathcal{M}, \pi^1 \models \varphi$$

$$\mathcal{M}, \pi \models F\varphi \text{ ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models G\varphi \text{ ssi } \forall i \in \mathbb{N}, \mathcal{M}, \pi^i \models \varphi$$

$$\mathcal{M}, \pi \models \varphi U \psi \text{ ssi } \exists i \in \mathbb{N}, \mathcal{M}, \pi^i \models \psi \text{ et}$$

$$\forall k \in \{0, \dots, i-1\}, \mathcal{M}, \pi^k \models \varphi$$



CTL* et Rdp

- Comme pour CTL, un réseau de Pétri marqué vérifie une formule de CTL* ssi cette formule est vérifié dans son marquage initiale (ie dans tout monde formé à partir de M_0)

CTL* et Rdp

- Rdp marqué $R = \langle P, T, Pre, Post, M_0 \rangle$

R sans blocage ssi R vérifie $AG \bigvee_{t \in T} t$

$t \in T$ quasi-vivante ssi R vérifie EFt

R quasi-vivant ssi R vérifie $\bigwedge_{t \in T} EFt$

$t \in T$ vivante ssi R vérifie $AGEFt$

R vivant ssi R vivant ssi R vérifie $AG \bigwedge_{t \in T} EFt$

CTL* et Rdp

- Rdp marqué $R = \langle P, T, Pre, Post, M_0 \rangle$

R sans blocage ssi R vérifie $AG \bigvee_{t \in T} t$

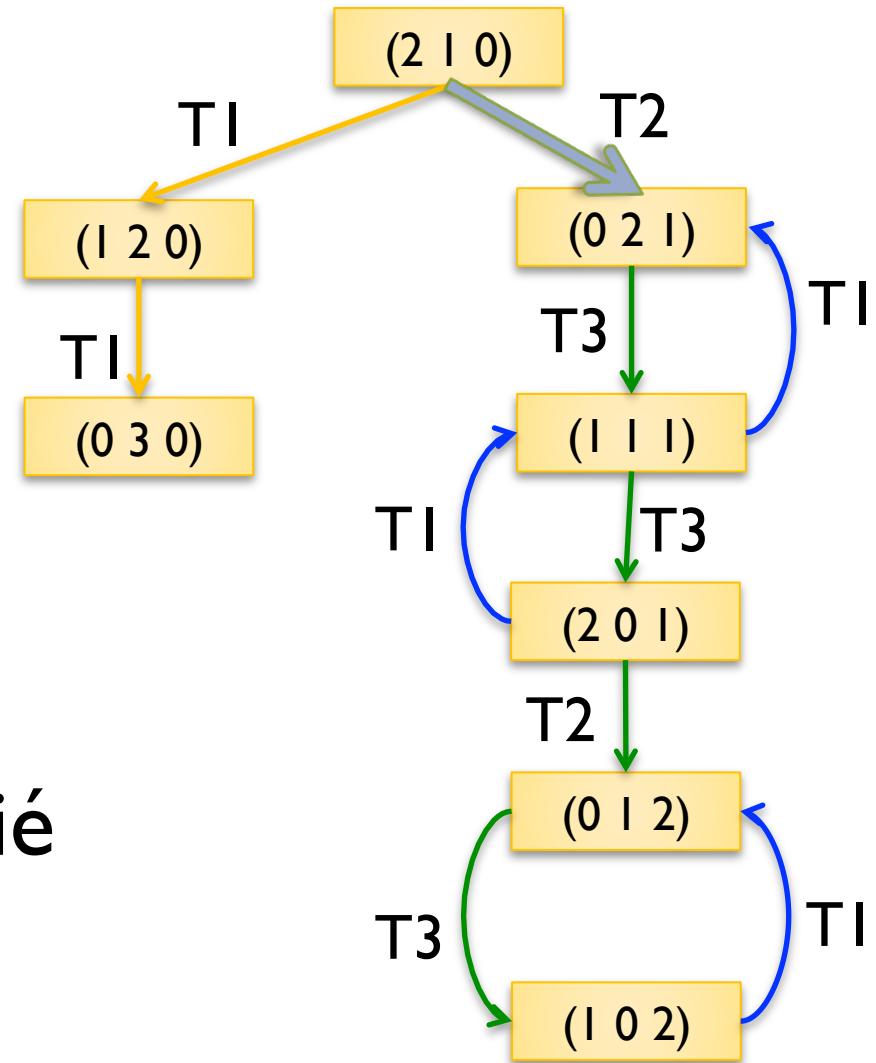
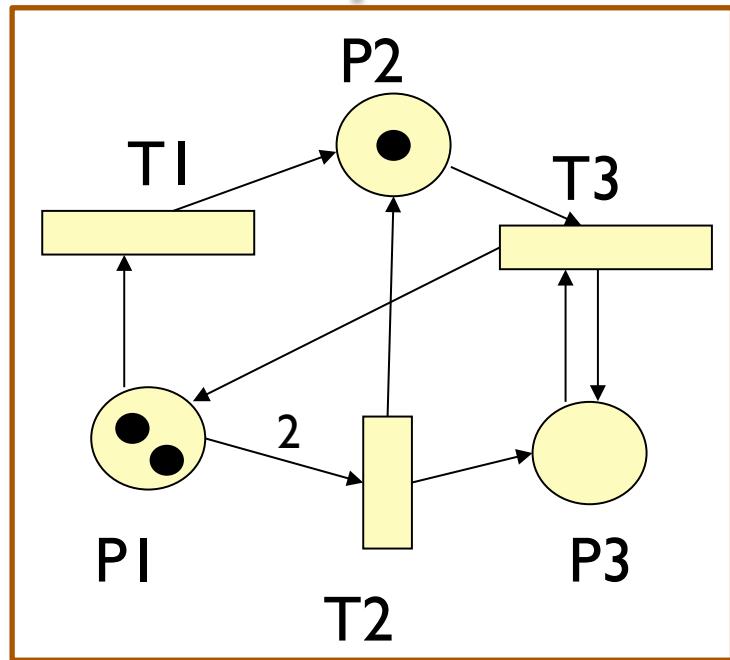
$t \in T$ quasi-vivante ssi R vérifie EFt

R quasi-vivant ssi R vérifie $\bigwedge_{t \in T} EFt$

$t \in T$ vivante ssi R vérifie $AGEFt$

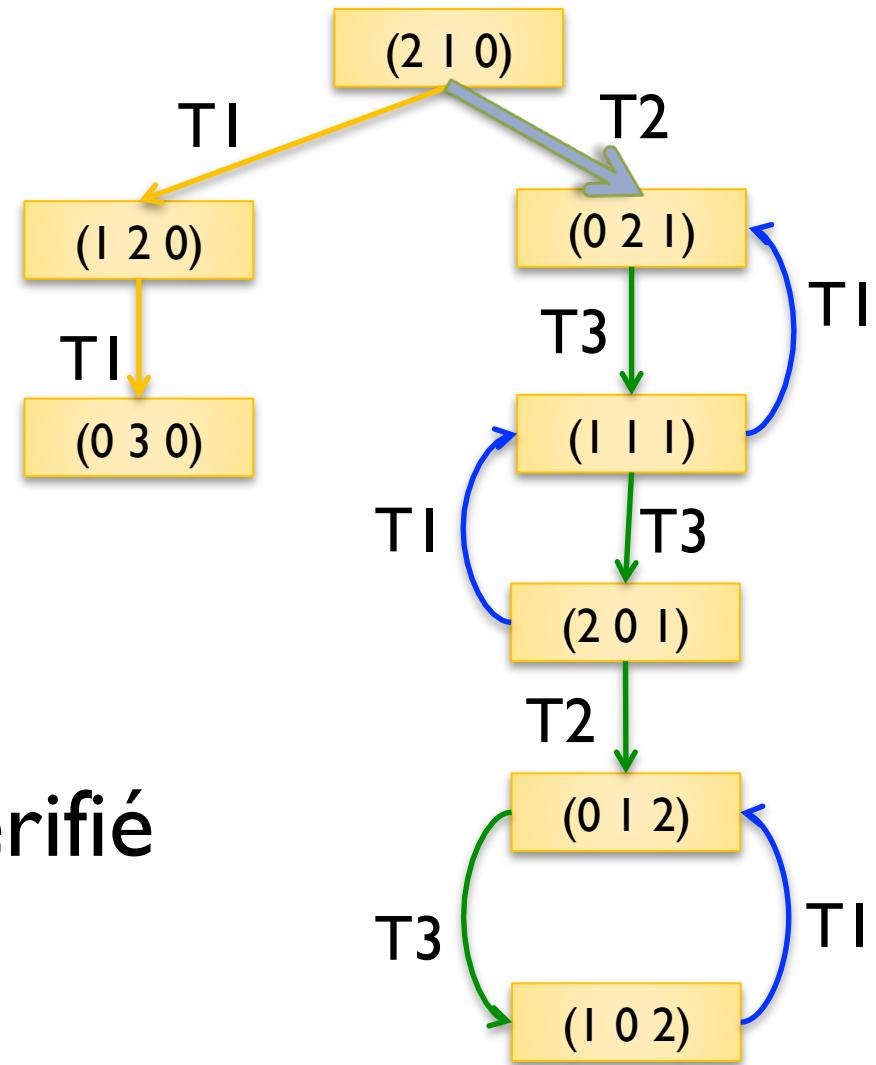
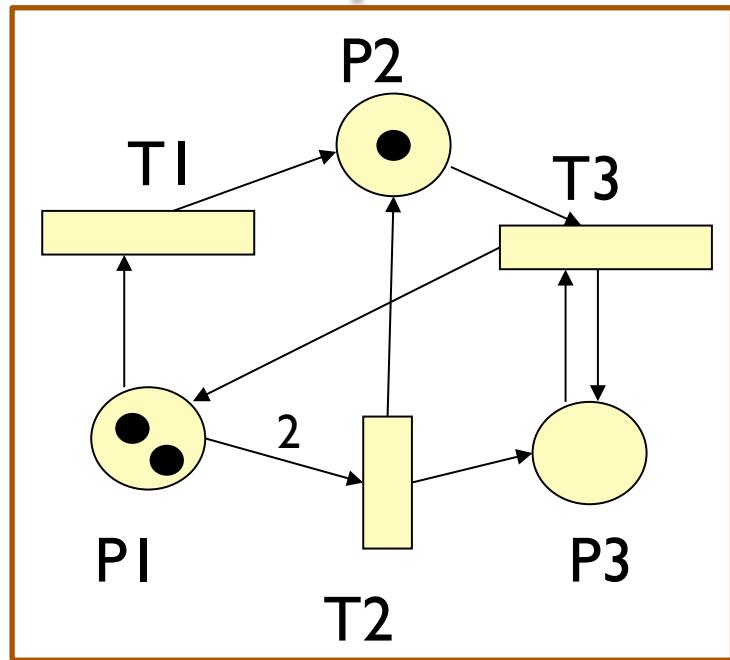
R vivant ssi R vivant ssi R vérifie $AG \bigwedge_{t \in T} EFt$

Exemple



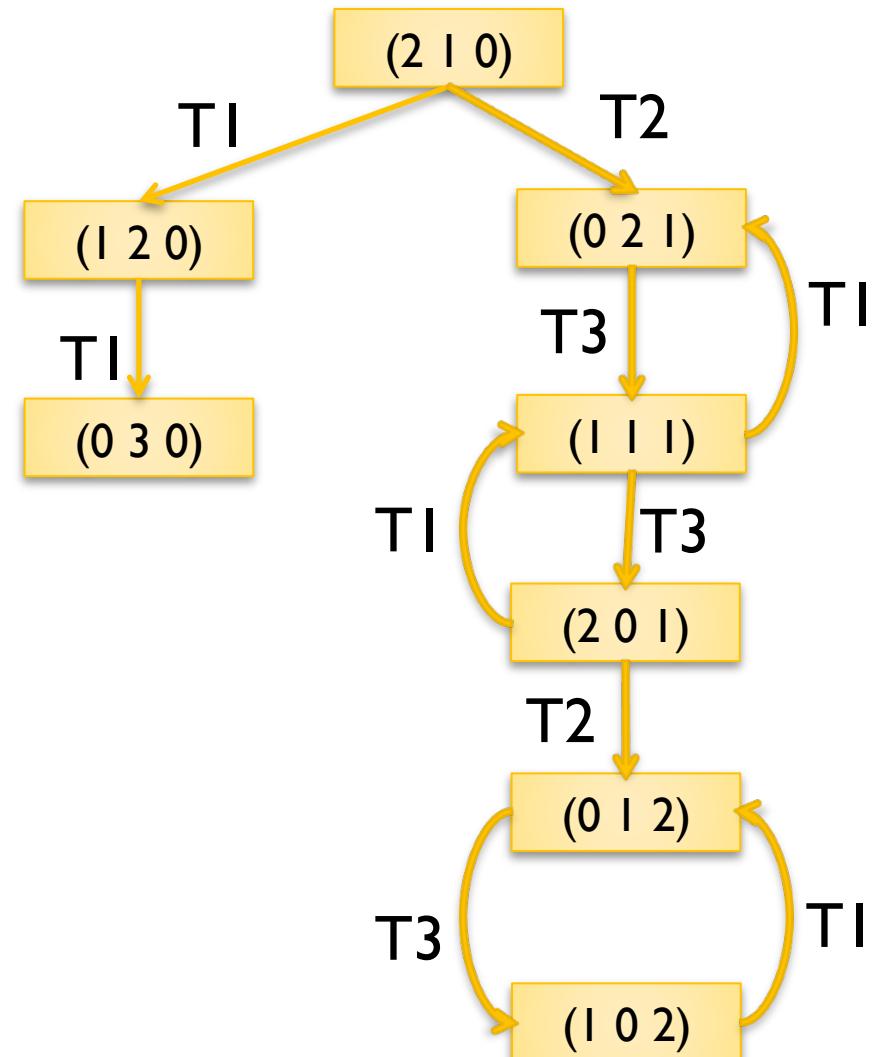
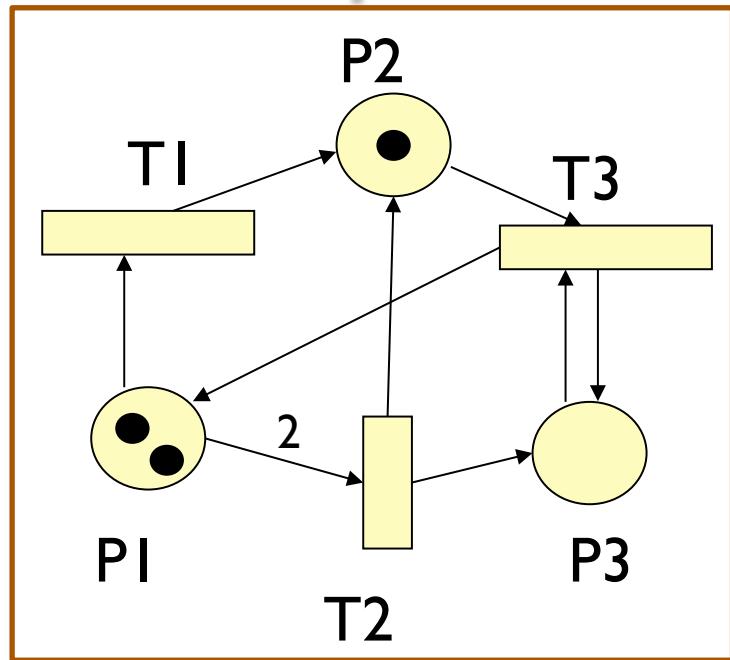
EXG AGFT_I vérifié

Exemple



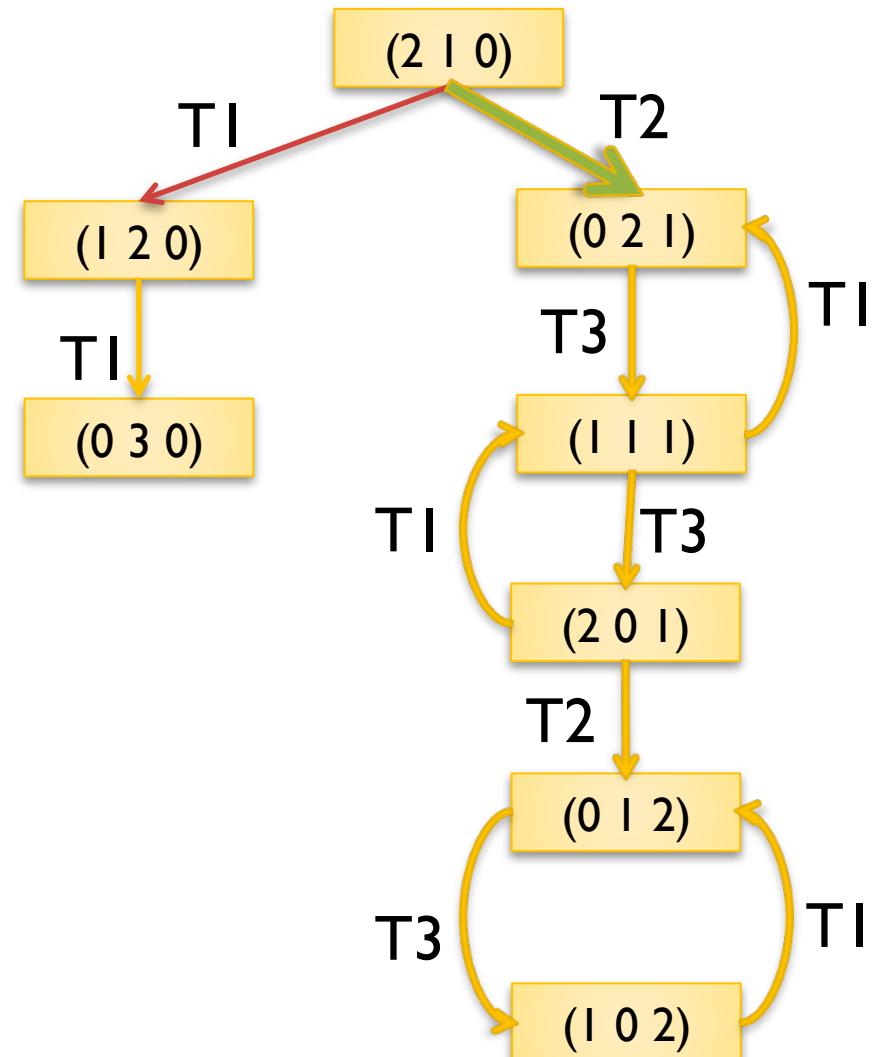
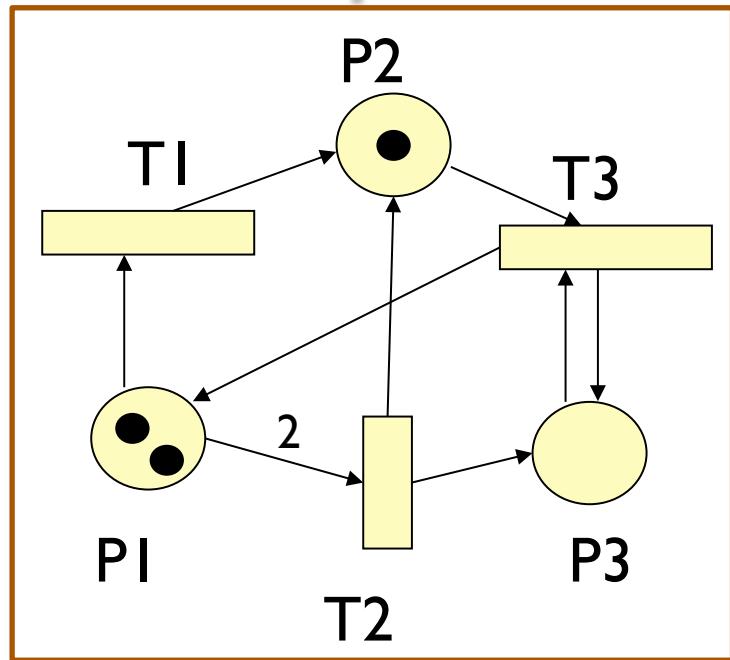
$A(T_2 \rightarrow GF T_1)$ vérifié

Exemple



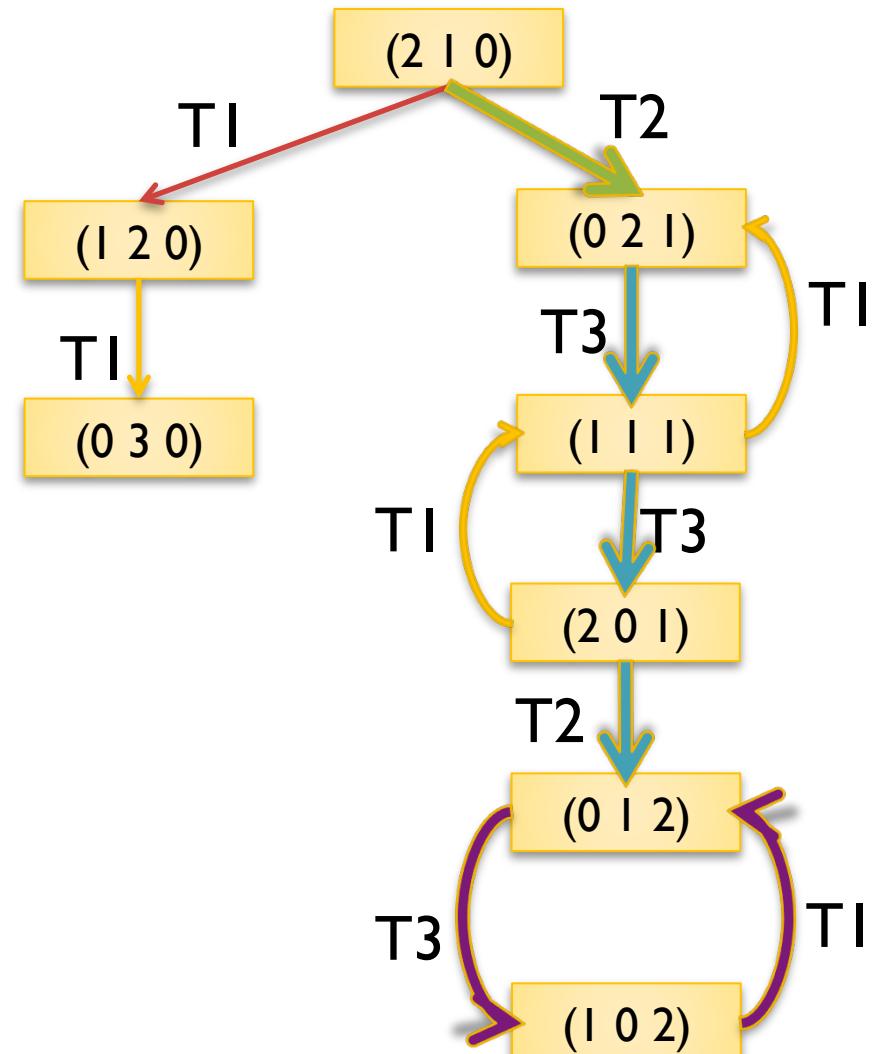
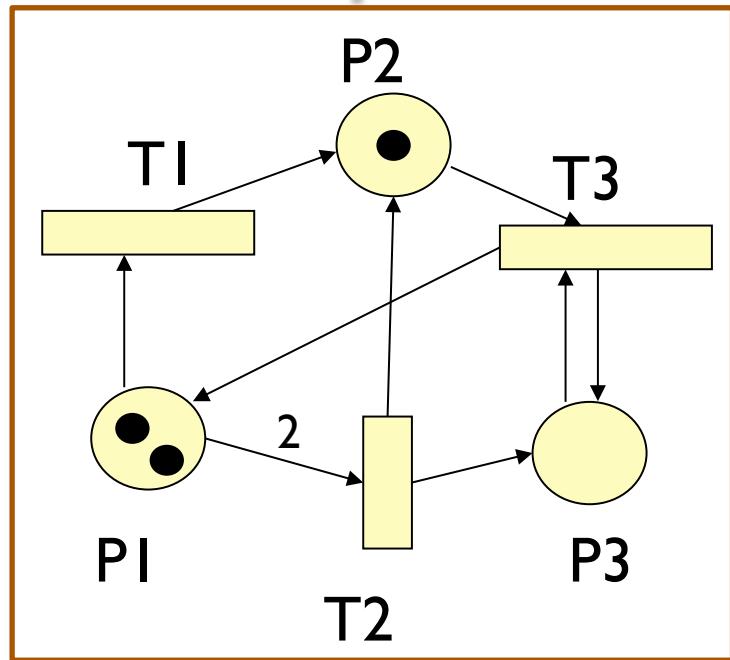
$A(T_2 \rightarrow \text{EXXXAG}((T_1 \wedge \neg T_3) \vee (T_3 \wedge \neg T_1)))$

Exemple



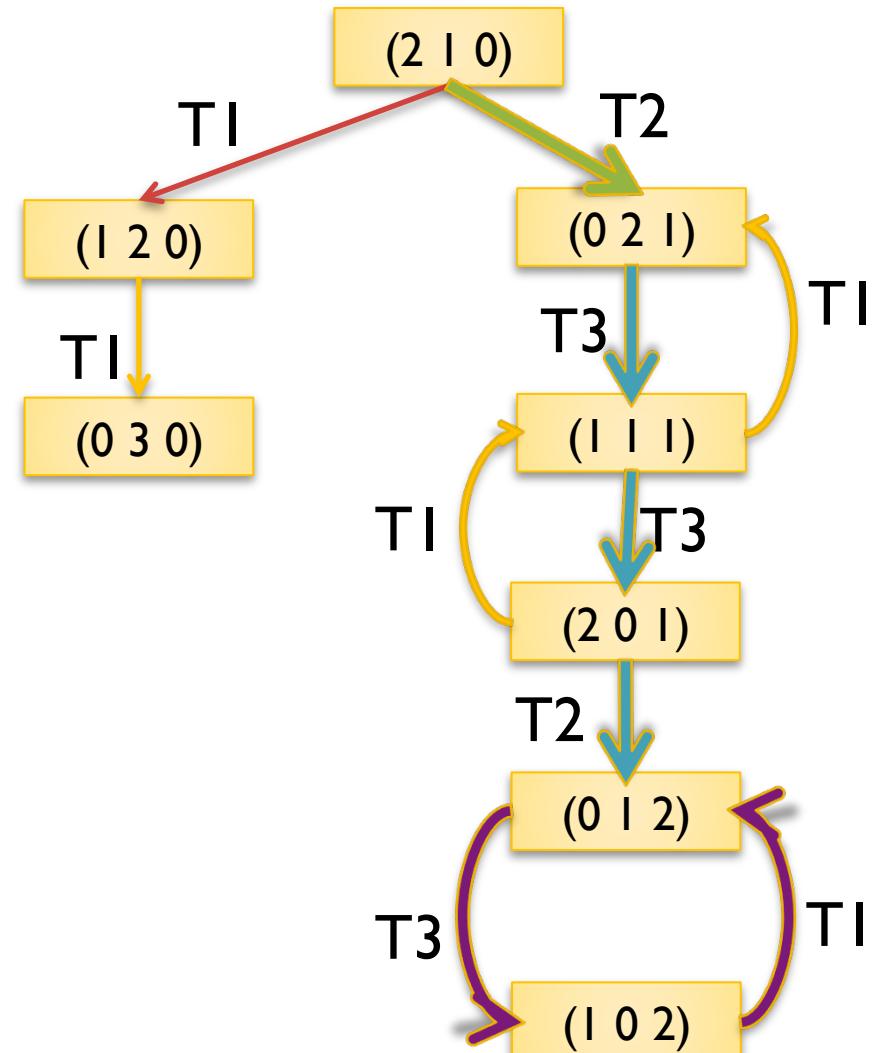
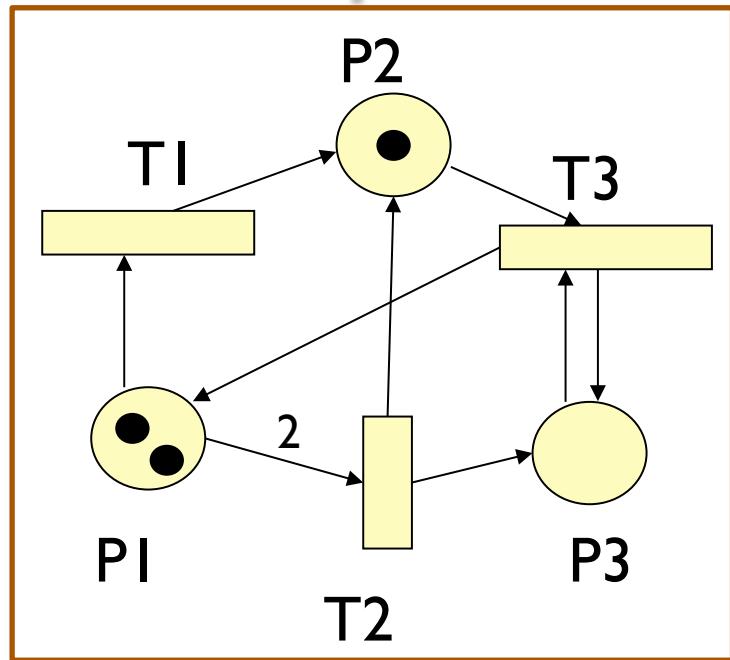
$A(T_2 \rightarrow \text{EXXXAG}((T_1 \wedge \neg T_3) \vee (T_3 \wedge \neg T_1)))$

Exemple



$A(T_2 \rightarrow \text{EXXXAG}((T_1 \wedge \neg T_3) \vee (T_3 \wedge \neg T_1)))$

Exemple



Vérifié

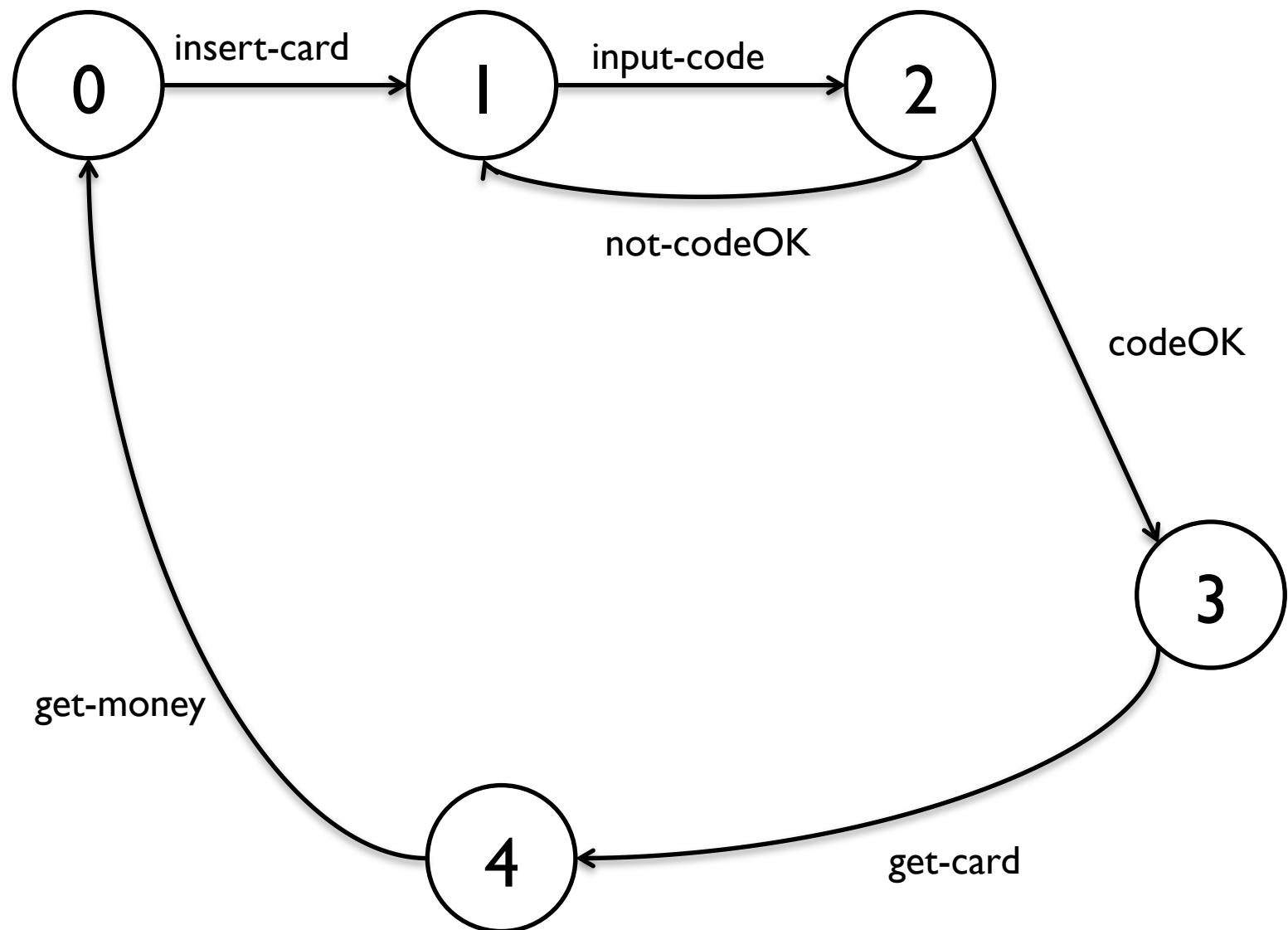
$A(T_2 \rightarrow \text{EXXXAG}((T_1 \wedge \neg T_3) \vee (T_3 \wedge \neg T_1)))$



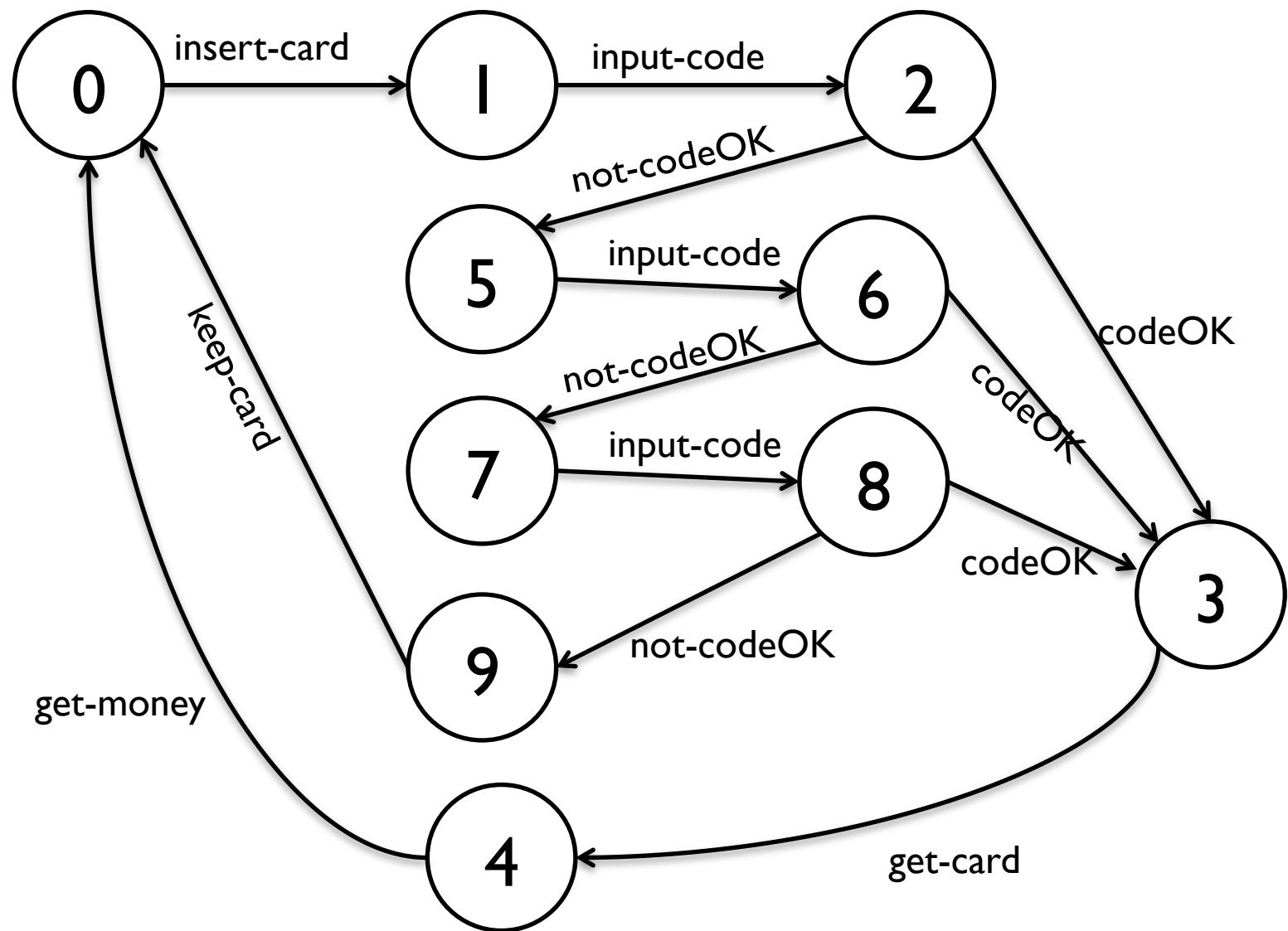
Ouverture

Gestion du temps continu : automates temporisés

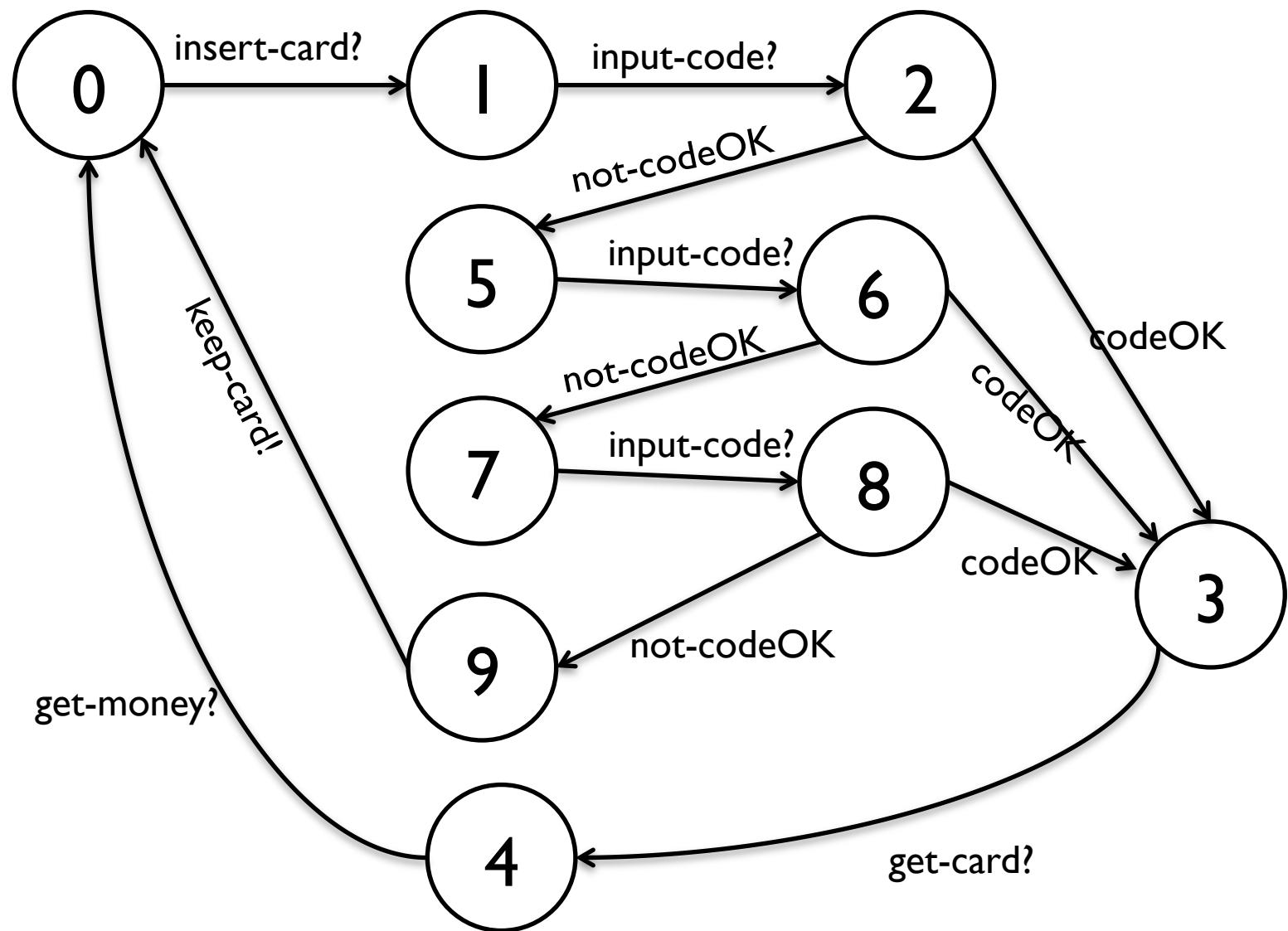
Exemple : DAB



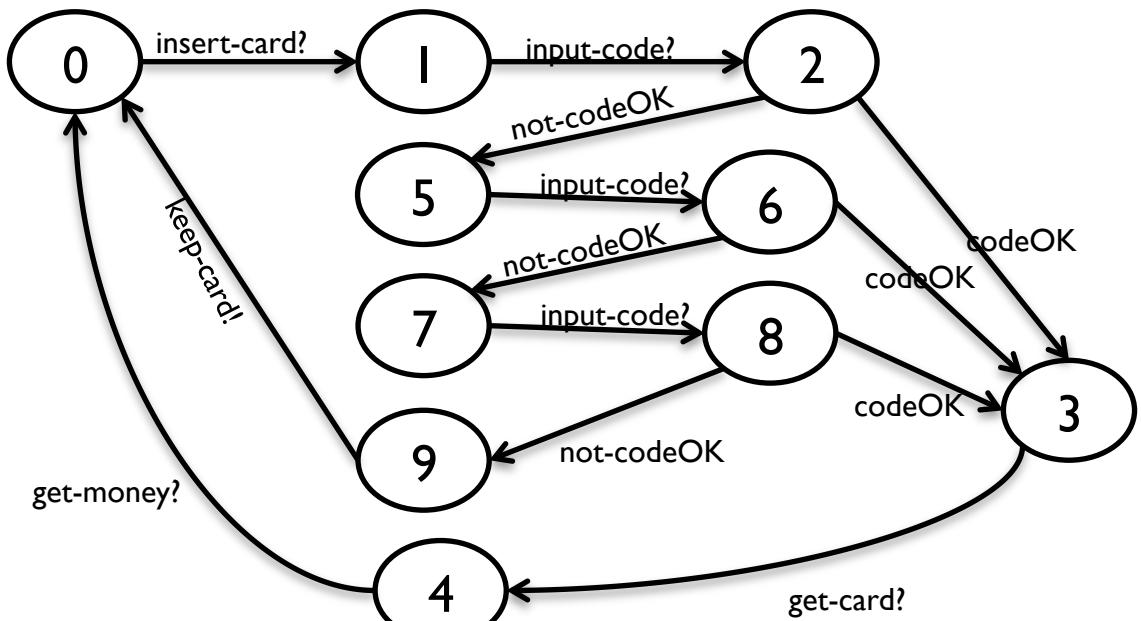
Exemple : DAB



Exemple : DAB input-output



Exemple : DAB input-output



- Input : insert-card? Input-code? get-card?
=> external action, action from env, receiving message
- Output : keep-card! (release-card/money!)
=> action towards env, sending message
- Interne : codeOk, notCodeOk
=> internal check and/or transitions

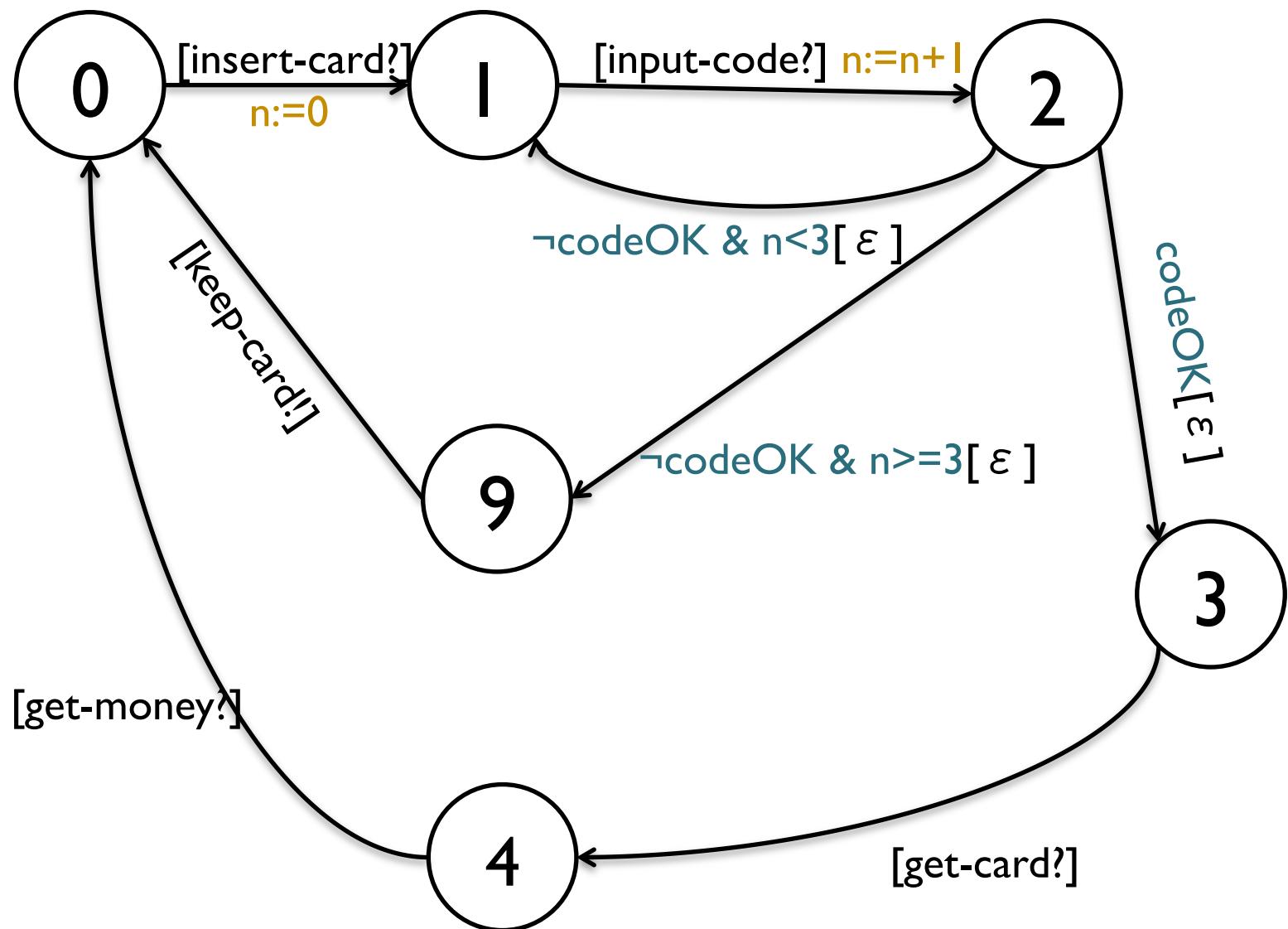


IOSTS

- Input-Output Symbolic Transition System
- Ajout de variables et garde
 - Variables : booléenne ou entières
 - Garde : formule à vérifier pour passer transition
 - Mise à jour : update des variables
 - Etiquette :
 - Canaux
 - Transition interne
 - Epsilon-transitions : label vide

DAB : IOSTS

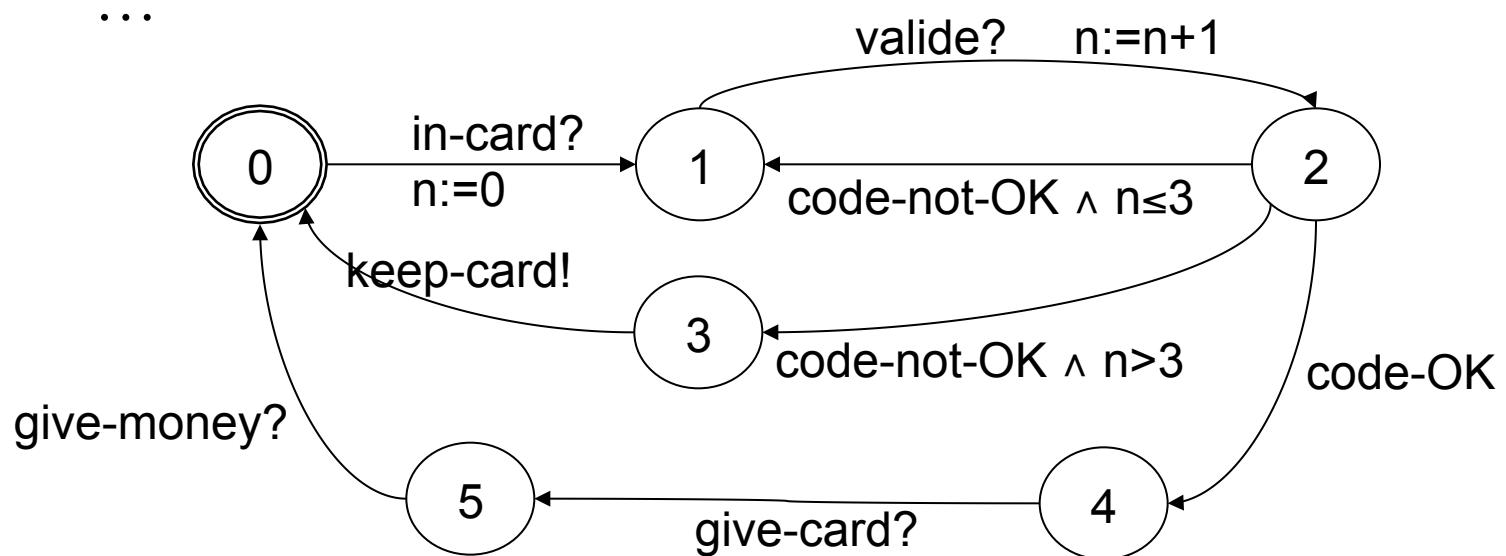
Garde Update



Exigences temporelles du DAB

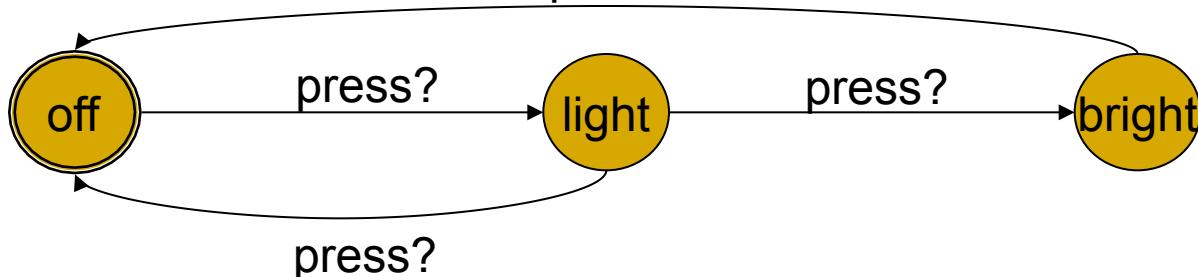
Comment prendre en compte des exigences temps réel telles que :

- lors de chaque essai, le client doit valider son code en moins de 6 secondes, sinon la carte est conservée
- lorsque la transaction réussit, le client doit récupérer sa carte en moins de 3 secondes, sinon celle-ci est conservée



Lampe à trois états

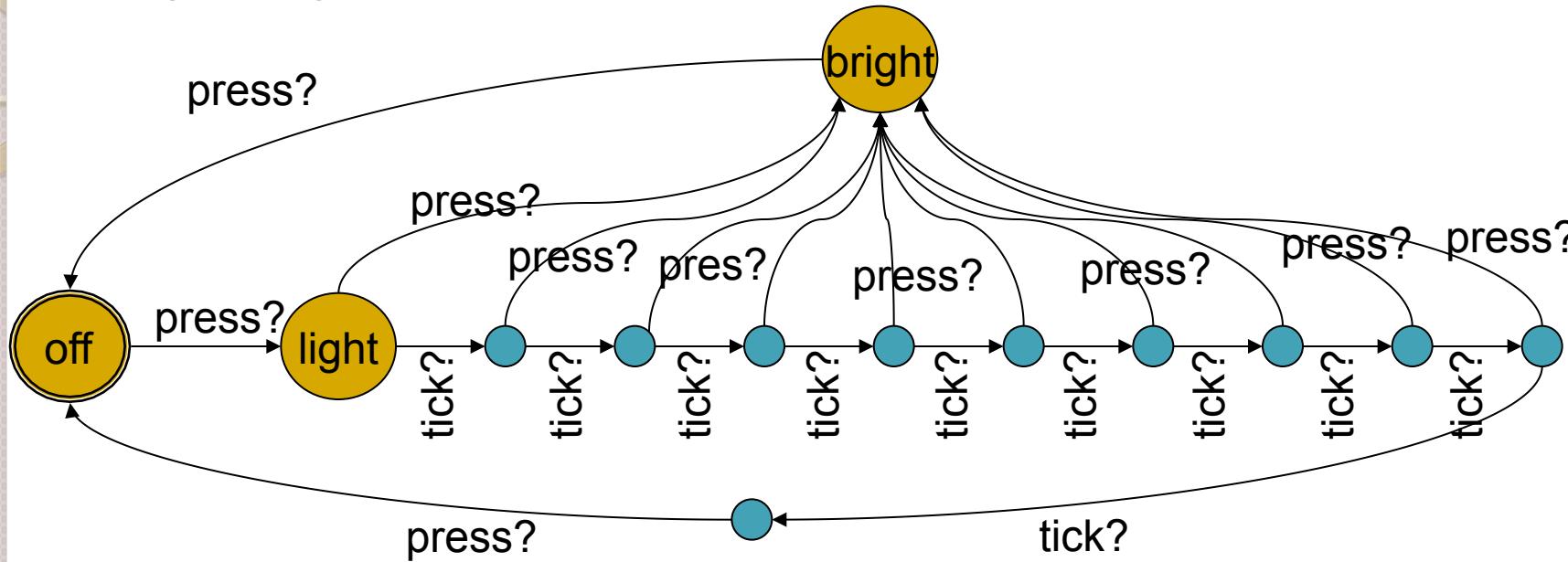
- lorsque la lampe est éteinte (off), un appui simple sur "press" l'allume (light)
- lorsque la lampe est éteinte, un double appui sur "press" la rend fortement lumineuse (bright)
- lorsque la lampe est allumée (light) un appui sur "press" l'éteint (off)
? double appui = deux impulsions en moins de 10ms
press?
- lorsque la lampe est fortement lumineuse (bright) un appui sur "press" l'éteint (off)



Il faut un formalisme pour représenter les durées des actions

Première solution : temps discret

Exemple : 1 pas = 1 ms



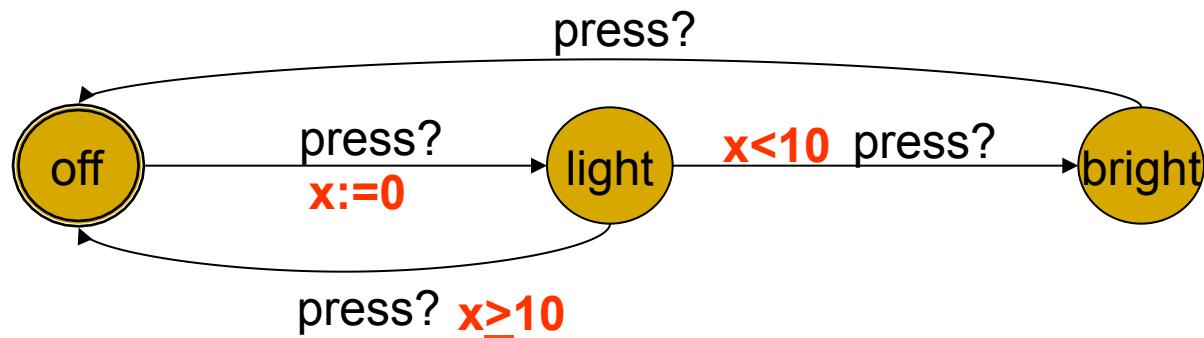
Mais :

- modélisation dépendante du pas d'échantillonnage
- modélisation peu précise :
 - peut rejeter 2 impulsions successives de "press" séparées par 9,1 ms et
 - accepter 2 impulsions successives de "press" séparées par 9,9 ms

Autre solution : la prise en compte d'un temps continu !

2ème solution

Introduire une variable réelle modélisant l'écoulement du temps et exprimée en ms



Avec $x \in \mathbb{R}^+$; x est appelée *horloge* ;

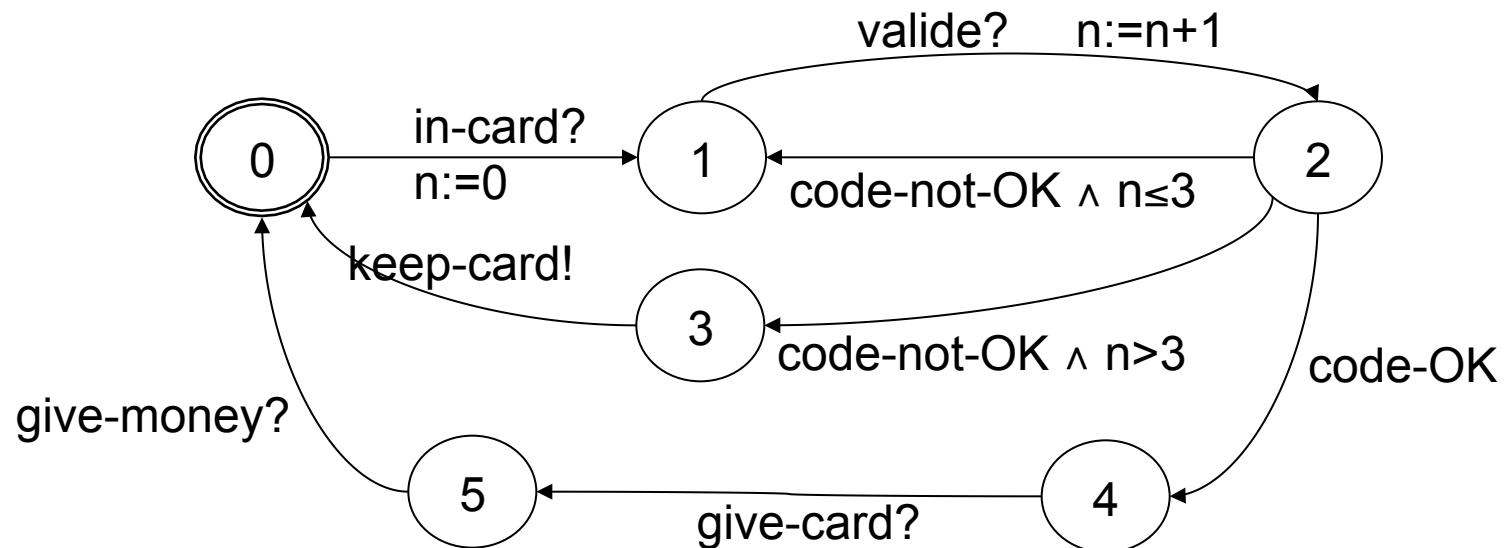
- on peut la réinitialiser ($x:=0$)
- on peut la comparer à des constantes ($x<10$, $x>10$)

Exigences temporelles du DAB

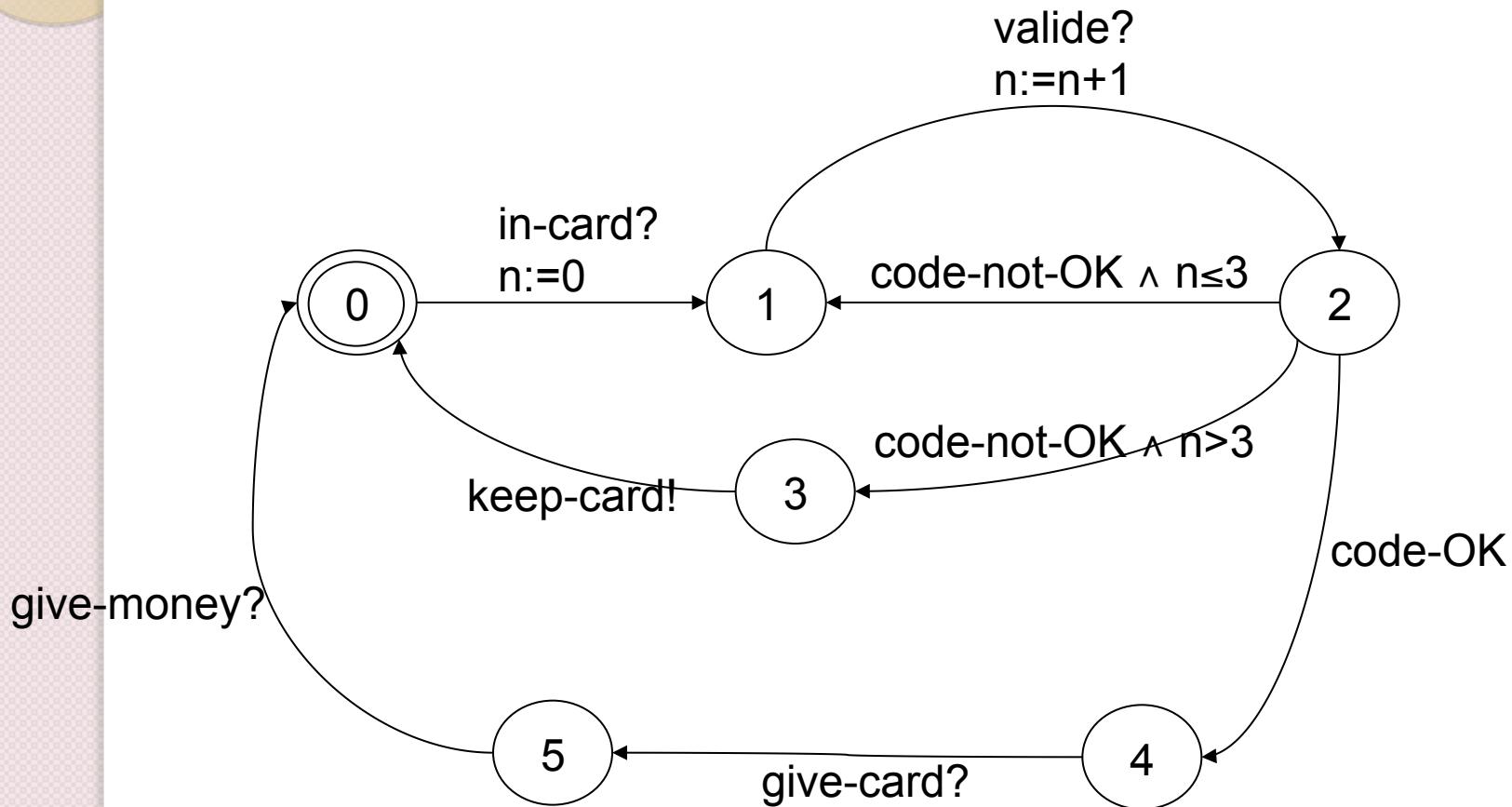
Comment prendre en compte des exigences temps réel telles que :

- lors de chaque essai, le client doit valider son code en moins de 6 secondes, sinon la carte est conservée
- lorsque la transaction réussit, le client doit récupérer sa carte en moins de 3 secondes, sinon celle-ci est conservée

...



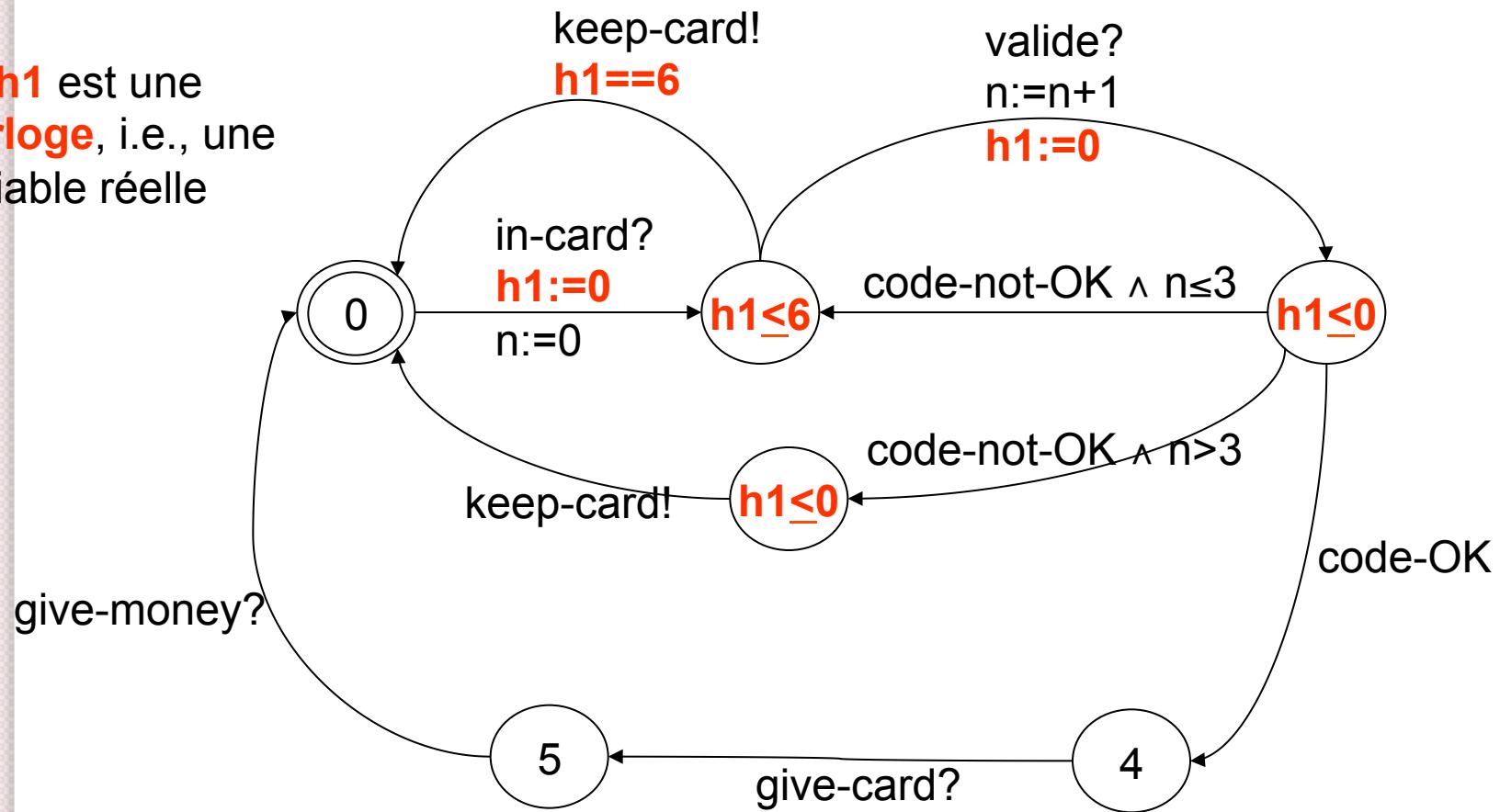
Suite du DAB



Suite du DAB

- lors de chaque essai, le client doit valider son code en moins de 6 secondes, sinon la carte est conservée

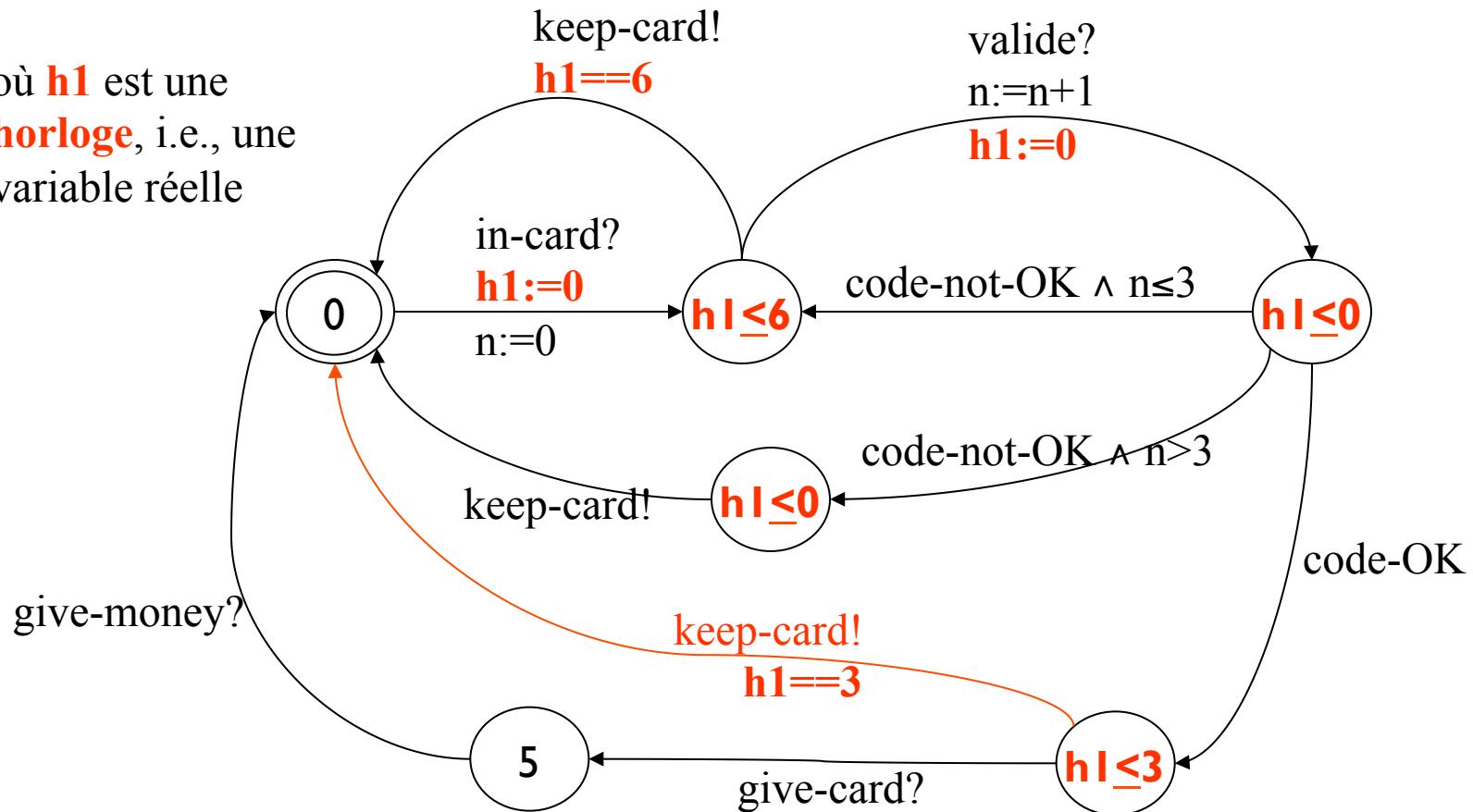
où **h1** est une horloge, i.e., une variable réelle



Suite du DAB ...

- lorsque la transaction réussit, le client doit récupérer sa carte en moins de 3 secondes, sinon celle-ci est conservée

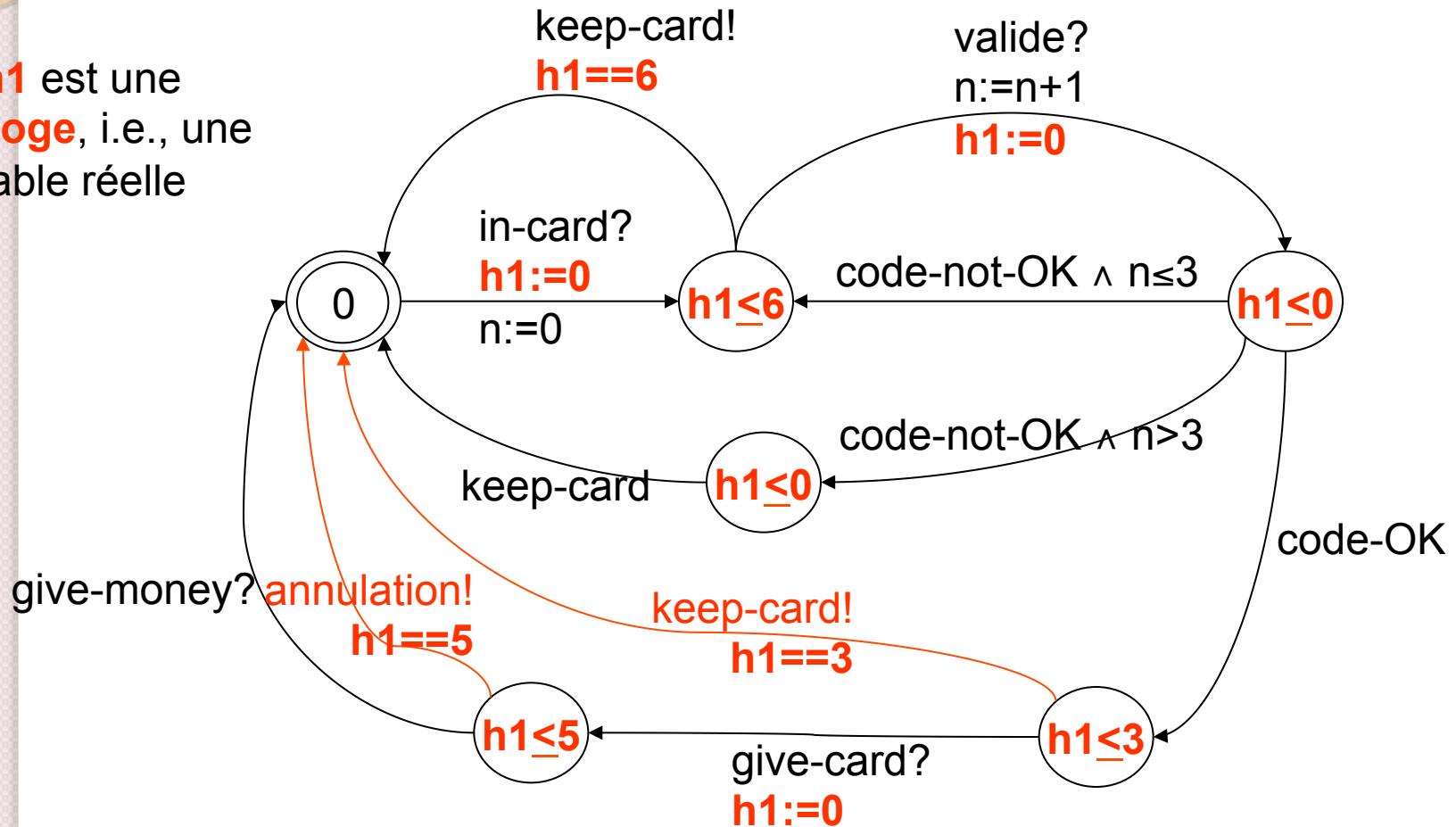
où **h1** est une **horloge**, i.e., une variable réelle



Suite du DAB

→ le client doit récupérer ses billets en moins de 5 secondes, sinon la transaction est annulée

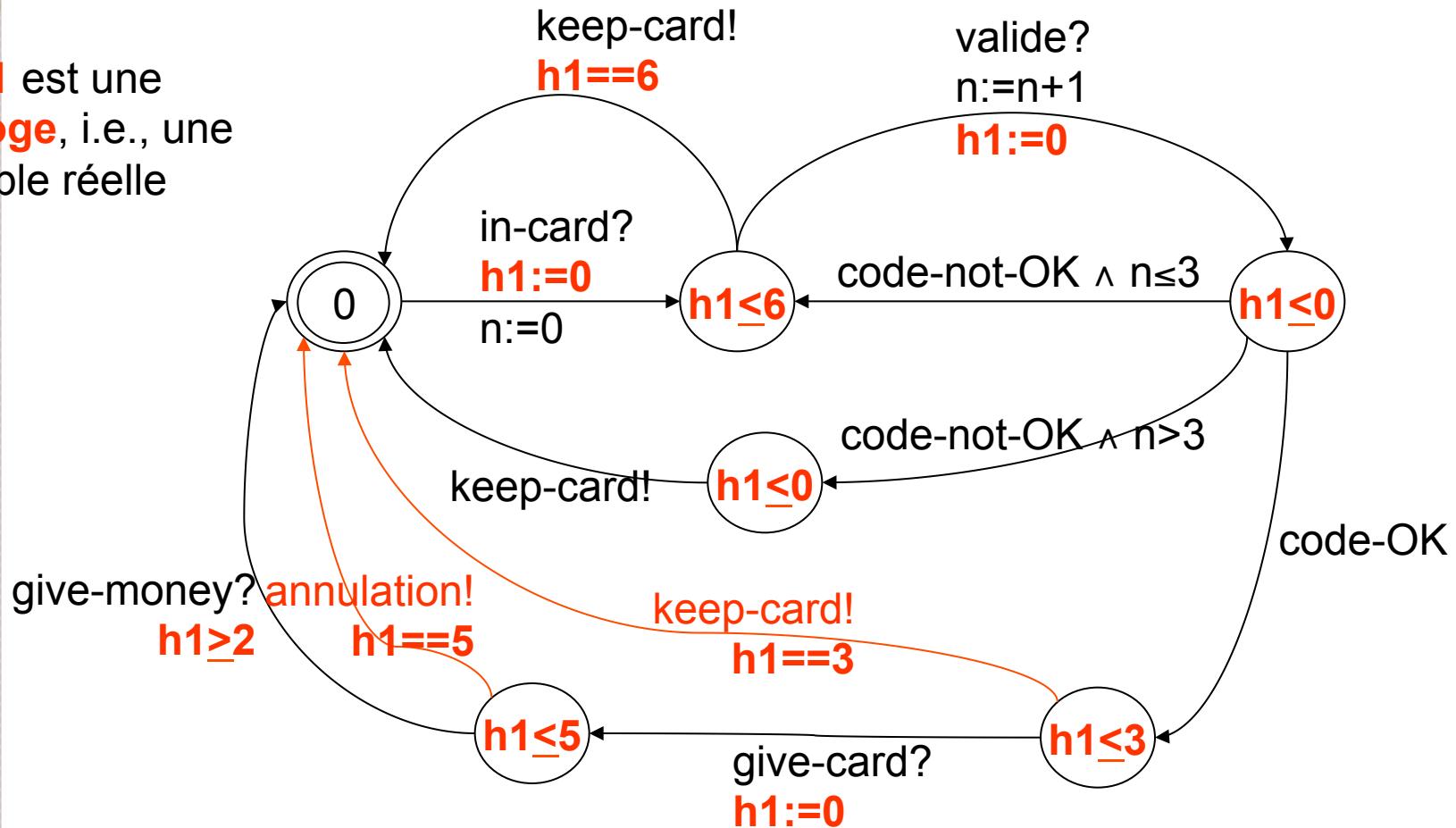
où **h1** est une **horloge**, i.e., une variable réelle



Suite du DAB

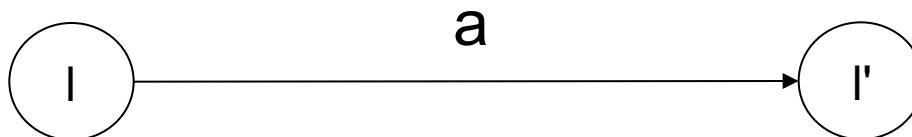
- les billets ne peuvent être récupérés par le client qu'au plus tôt 2 secondes après avoir récupéré sa carte

où $h1$ est une **horloge**, i.e., une variable réelle



Automate temporisé

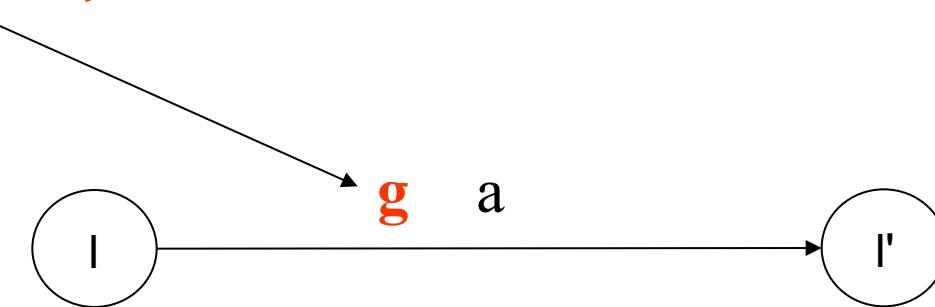
- Généralisation : un automate temporisé, c'est :
 - des nœuds et des transitions entre ces nœuds étiquetées par des actions



Automate temporisé

- Généralisation : un automate temporisé, c'est
 - des nœuds et des transitions entre ces nœuds étiquetées par des actions
 - des gardes temporelles sur les transitions

Garde : $x - y \sim k$ ou $x \sim k$
avec x et y deux horloges
avec k une constante entière
et $\sim \in \{==, <, >, \leq, \geq\}$

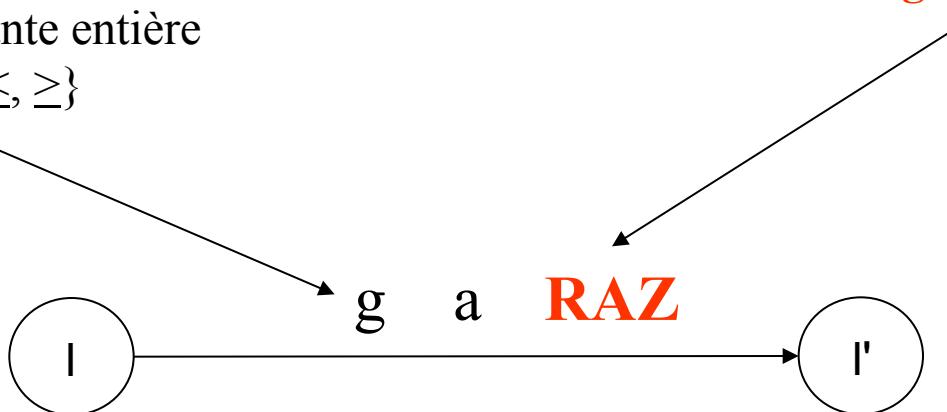


Automate temporisé

- Généralisation : un automate temporisé, c'est
 - des nœuds et des transitions entre ces nœuds étiquetées par des actions
 - des gardes temporelles sur les transitions
 - des remises à zéro d'horloges

Garde : $x - y \sim k$ ou $x \sim k$
avec x et y deux horloges
avec k une constante entière
et $\sim \in \{==, <, >, \leq, \geq\}$

Remise à zéro de toutes les horloges de RAZ

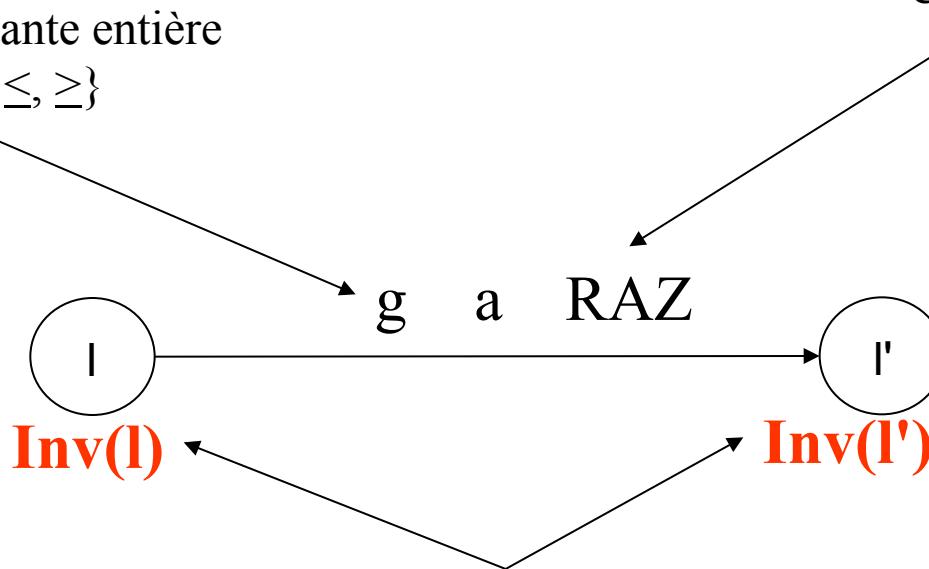


Automate temporisé

- Généralisation : un automate temporisé, c'est
 - des nœuds et des transitions entre ces nœuds étiquetées par des actions
 - des gardes temporelles sur les transitions
 - des remises à zéro d'horloges
 - des invariants dans chaque nœud pour forcer le mouvement

Garde : $x - y \sim k$
avec x et y deux horloges
avec k une constante entière
et $\sim \in \{==, <, >, \leq, \geq\}$

Remise à zéro de toutes
les horloges de RAZ

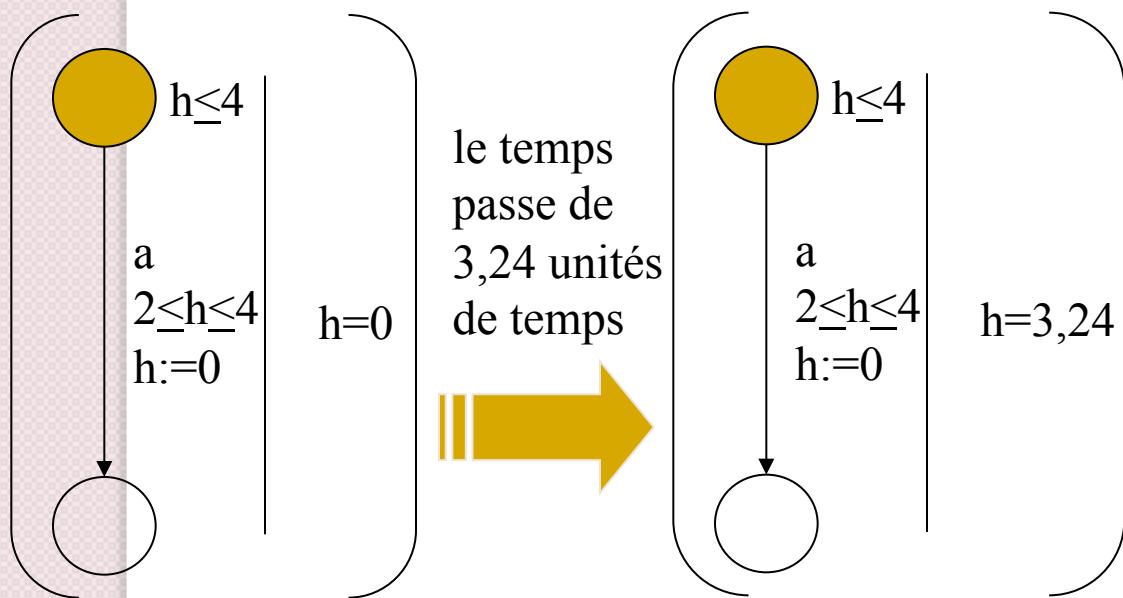


**Invariant : formule qui doit être vraie
tant que le nœud est occupé**

Comportement d'un automate temporisé

➤ Un automate temporisé évolue

- soit en laissant passer le temps sans changer de nœud



Comportement d'un A.T.

➤ un automate temporisé évolue

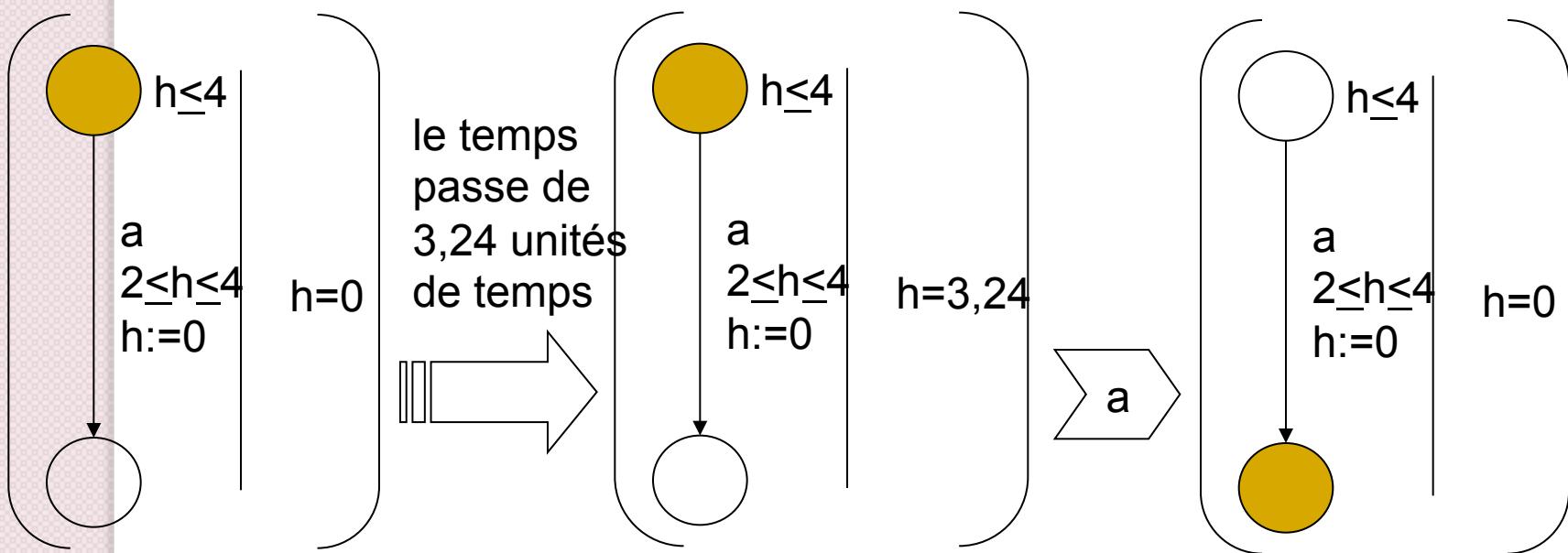
- soit en laissant passer le temps sans changer de nœud
- soit en tirant une transition sans laisser passer le temps

=> le tir d'une transition est instantané

=> une transition ne peut être tirée que si sa garde est vraie

=> une transition dont la garde est vraie peut ne pas être tirée

=> on ne peut rester dans un nœud dont l'invariant devient faux



Définition - préliminaires

Soit Σ un ensemble d'**actions**.

- **Suite temporelle** : suite finie ou infinie non décroissante de réels positifs
- **Mot temporisé** : suite finie ou infinie de paires (a_i, t_i) : $(a_1, t_1)(a_2, t_2, \dots (a_p, t_p) \dots$ telle que chaque a_i soit dans Σ et que $(t_i)_{i>0}$ soit une suite temporelle

Soit X un ensemble d'horloges (var. réelles)

- **Valuation** (d'une horloge): mapping
 $v:X \rightarrow \mathbb{R}_{\geq 0}$

Valuations

Soit X un ensemble d'**horloges** (var. réelles)

- **Valuation** (d'une horloge): mapping $v:X \rightarrow \mathbb{R}_{\geq 0}$
- $\mathbb{R}_{\geq 0}^X$: ensemble de toutes les valuations.
- $\mathbf{0}_X$: valuation qui assigne 0 à ttes les horloges
- Soit v une valuation et t un réel pos.
 $v+t$ défini pour tt x par $v+t(x)=v(x)+t$
- Soit v valuation et r subset de X
 $v[r]$ est la valuation obtenue à partir de v en remettant à 0 tte horloge de r
 $v[r](x)=0$ si x dans r et $v[r](x)=x$ si x dans $X-r$

Contraintes d'horloge

Une **contrainte d'horloge** est une conjonction d'éléments de la forme $(x \sim k)$ où x dans X , k entier relatif et \sim dans $\{<, \leq, =, \geq, >\}$

$\Phi(X)$: ensemble des contraintes d'horloge

Soit φ dans $\Phi(X)$ et v dans $R_{\geq 0}^X$

v satisfait φ ($v \models \varphi$) ssi v satisfait chaque $(x \sim k)$ de la conjonction (ie $v(x) \sim k$)

Extensions

- Contrainte diagonale : $(x-y \sim k)$
- Ajout de variable booléennes

Syntaxe d'un A.T.

Un automate temporisé est (AT) défini par :

$$\langle Q, q_0, X, \Sigma, Inv, \Delta, F \rangle$$

- Q : ensemble fini d'états, avec un état initial q_0
- X : ensemble fini d'horloges $X = \{x, y, z \dots\}$
- Σ : un alphabet fini $\Sigma = \{a, b, c \dots\}$
- Inv : mapping d'invariants $X \rightarrow \Phi(X)$
- Δ un ensemble Δ de transitions, avec $\Delta \subseteq Q \times \Phi(X) \times \sigma \times 2^X \times Q$
chaque transition $\partial = (q_s, g, l, r, q_d)$ est notée $q_s \xrightarrow{g,l,r} q_d$ et contient:
 - les états source q_s et destination q_d
 - une garde g dans $\Phi(X)$: contrainte d'horloge
 - une lettre l de l'alphabet de Σ ,
 - des remises à zéro $r : x := 0, y := 0, \dots$ (noté comme le sous-ensemble de $X \{x, y, z \dots\}$)
- F : ensemble d'états finaux

Sémantique d'un A.T.

- Une **valuation** est un élément de $R^X_{\geq 0}$ c.à.d une fonction qui associe à chaque horloge sa valeur :

$$v(x) = 1,2 ; v(y) = 3,24 ; v(z) = \sqrt{2}$$

- Une **configuration** est un élément de $Q \times R^X_{\geq 0}$ c.à.d un état et une valuation :

$$(q_0 ; (1, 2 ; 3,24 ; \sqrt{2}))$$

- Une valuation **satisfait** une garde g si pour toute horloge $x \in X$ et toute contrainte c d'horloge x , on a $v(x)$ satisfait c .
- Pour $d \in R_{\geq 0}$, $v + d$ est la valuation $x \rightarrow v(x) + d$.
- Pour $r \subseteq X$, $v[r := 0]$ est la valuation $x \rightarrow 0$ si $x \in r$, $v(x)$ sinon.

Sémantique d'un A.T.

- Un A.T. génère un système de transition temporisé infini :

- La configuration initiale est $(q_0 ; 0)$: état initial, valuation nulle.
- Partant d'une configuration $(q ; v)$, deux types de transition :

Ecoulement du temps :

Pour $d \in R_{\geq 0}$, $v + d \Rightarrow Inv(q)$ alors $(q; v) \xrightarrow{d} (q; v + d)$.

transition discrète :

Si $q \xrightarrow{g;a;r} q' \in \Delta$, et, $v \Rightarrow g$ et $v[r:=0] \Rightarrow Inv(q')$

alors $(q, v) \xrightarrow{a} (q', v[r:=0])$

Sémantique d'un AT

La sémantique opérationnel d'un A.T. A est donné par le système de transition (temporisé) (ie LTS temporisé)
[|A|] = **<S,s0,L,T>**) suivant :

$$S = \{(q, v) \in Q \times \mathbb{R}_{\leq 0}^X \mid v \models Inv(q)\}$$

$$s_0 = (q_0, \mathbf{0}_X) \quad L = \mathbb{R}_{\leq 0} \times \Sigma$$

$$T = \{(q, v) \xrightarrow{d,a} (q', v') \mid \forall d' \in [0, d] : v + d' \models Inv(q) \\ \text{et } \exists (q \xrightarrow{g,a,r} q') \in \Delta : v + d \models g, \text{ et } v' = (v + d)[r]\}\}$$

- Exécution de A définie comme exécution de [|A|] :

$$(q_0, v_0) \xrightarrow{d_1,a_1} (q_1, v_1) \xrightarrow{d_2,a_2} \dots$$

- Mot temporisé :

$$w = (a_1, t_1)(a_2, t_2) \dots \text{ où } \forall i, t_i = \sum_{j \leq i} d_j$$

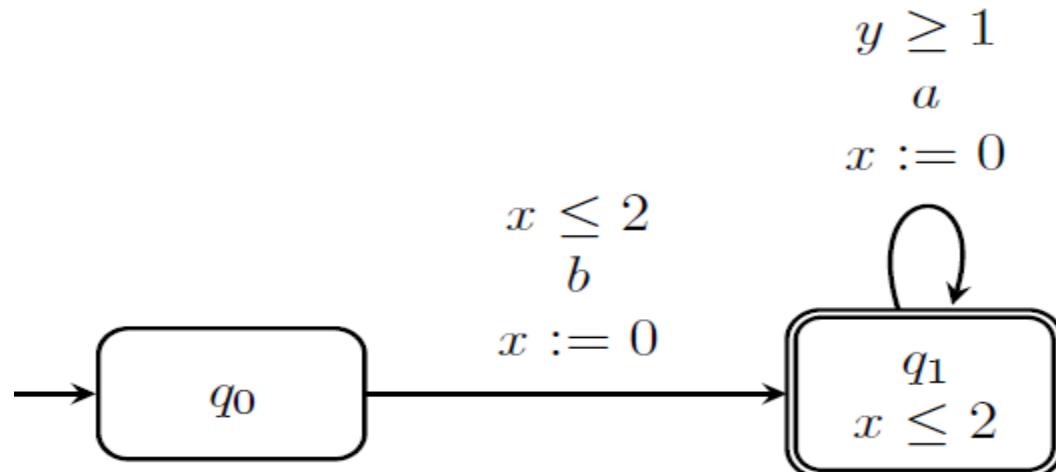
Exécution d'un A.T.

- Une exécution est un chemin dans le système de transitions temporisé où s'alternent les pas de temps et les transitions discrètes

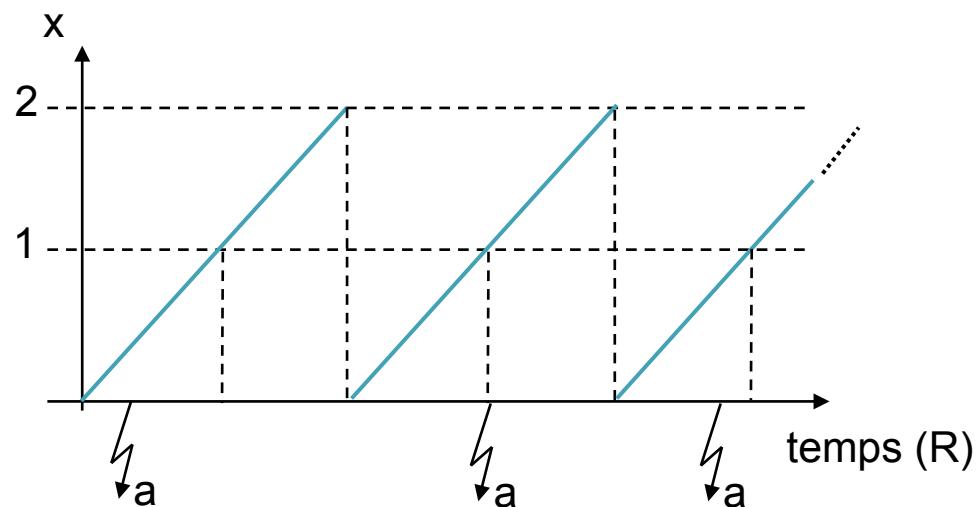
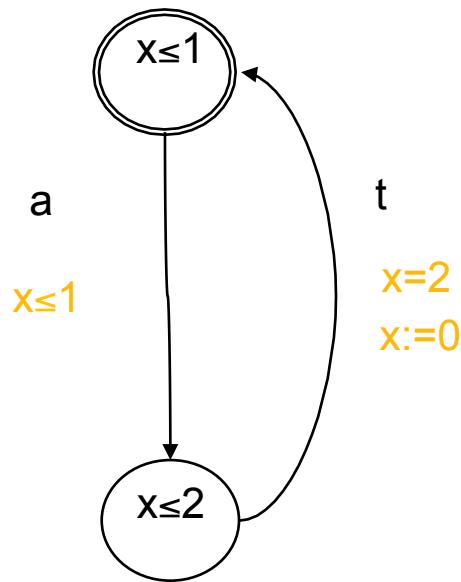
$$(q_0, (0, 0)) \xrightarrow{1.2} (q_0, (1.2, 1.2)) \xrightarrow{b} (q_1, (0, 1.2)) \xrightarrow{0.37} (q_1, (0.37, 1.57)) \xrightarrow{a} (q_1, (0, 1.57))$$

ou $(q_0, (0, 0)) \xrightarrow{1.2, b} (q_1, (0, 1.2)) \xrightarrow{0.37, a} (q_1, (0, 1.57))$

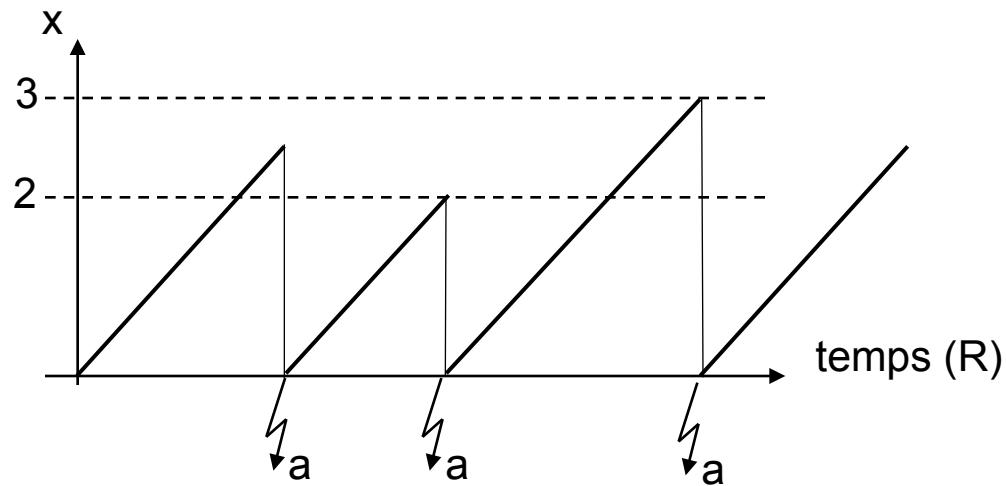
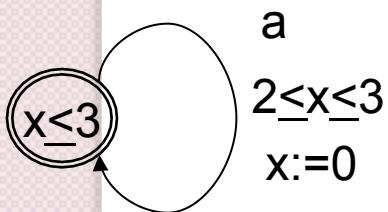
- Mot temporisé : $w = (b, 1.2)(a, 1.57)$



Exemple



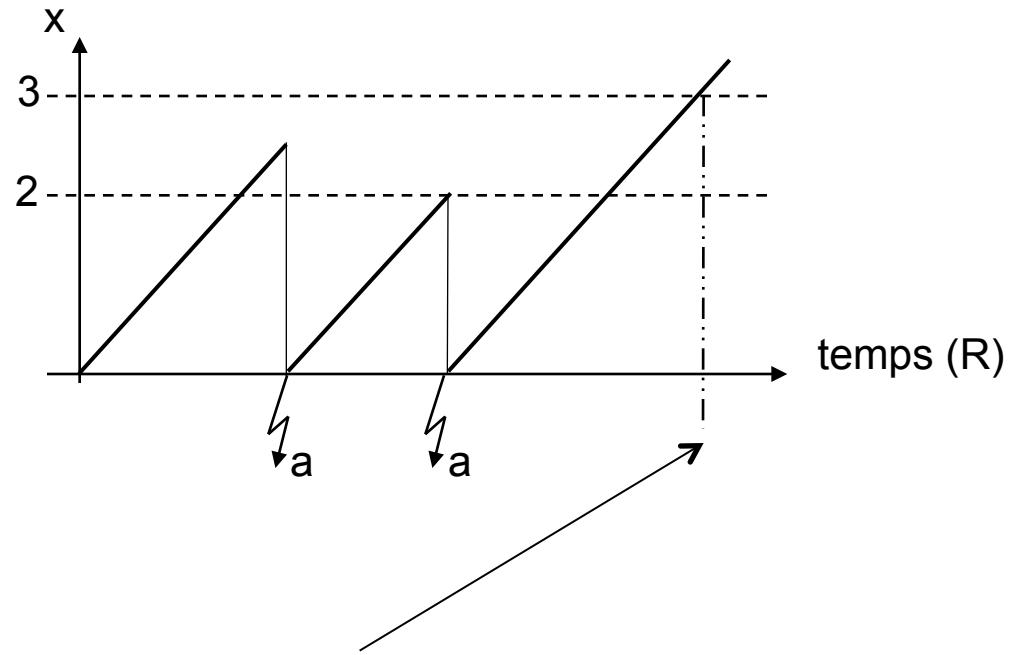
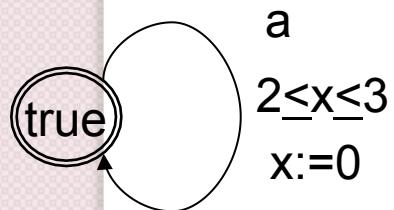
Exemple



- => l'invariant interdit de rester dans le nœud lorsque x devient supérieur à 3
- => franchissement obligatoire de la transition avant que x dépasse 3

↑
obligation de
franchissement de la
transition

Exemple

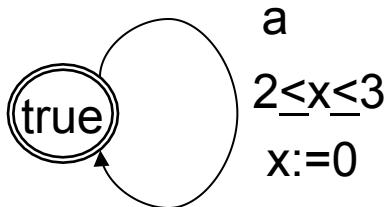


à partir de cet instant, la transition n'est plus franchissable

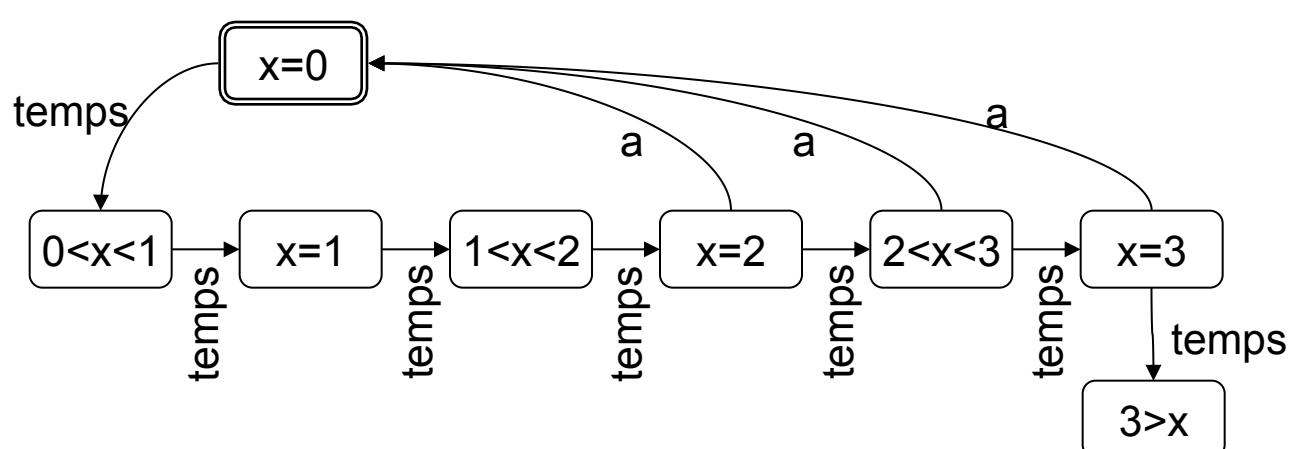
=> rien ne force la transition à être franchie entre 2 et 3.

Exemple

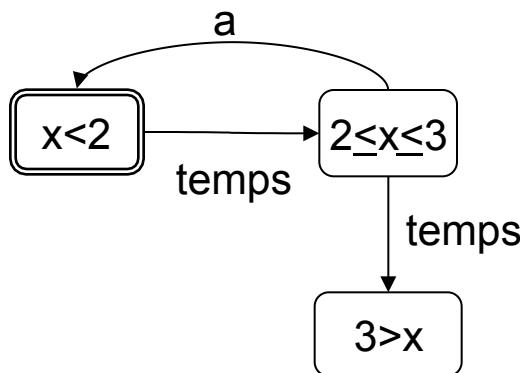
Remarque : il existe un automate discret équivalent :



équivalent à

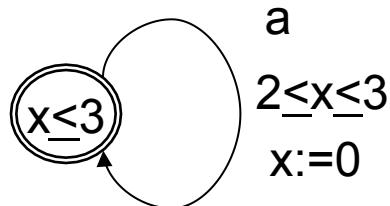


équivalent à

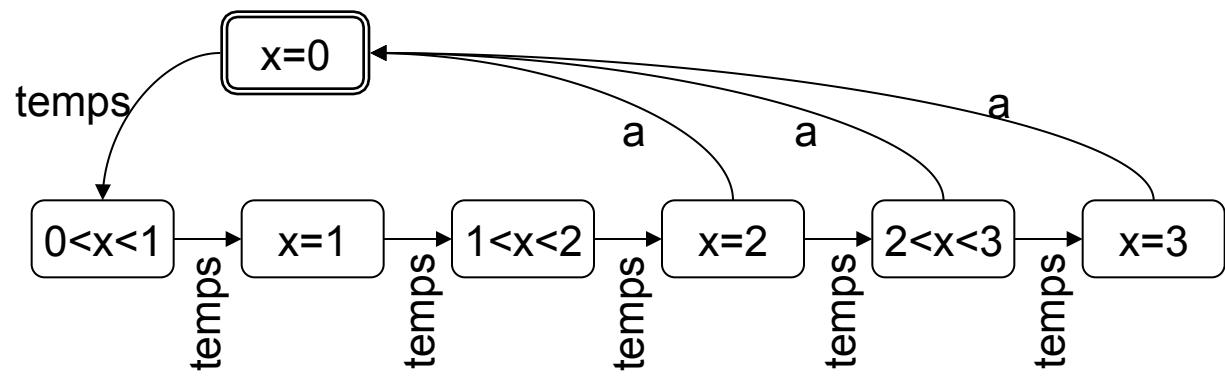


Exemple

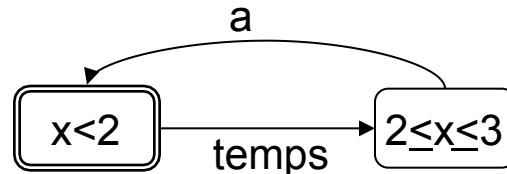
Remarque : il existe un automate discret équivalent :



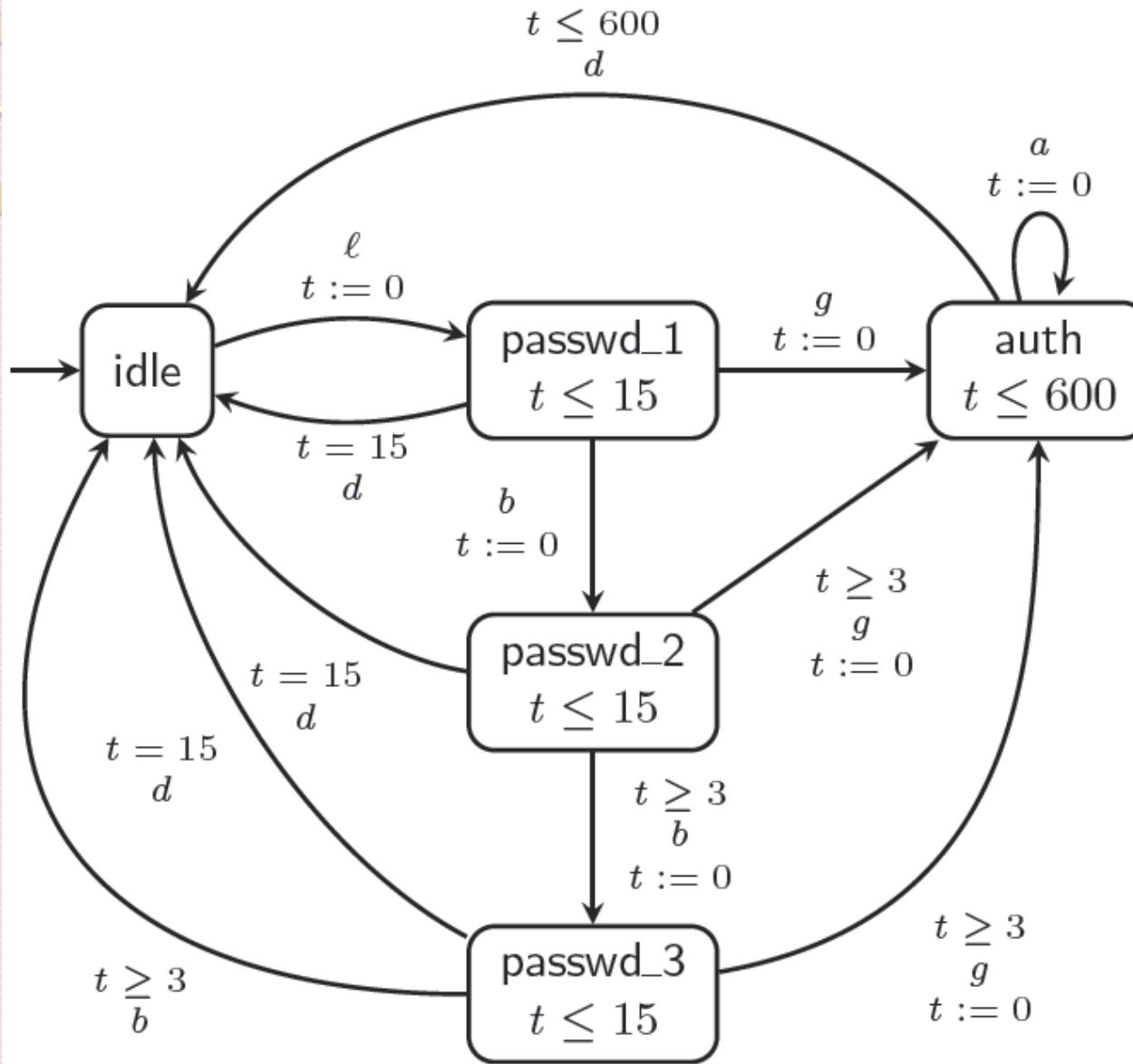
équivalent à



équivalent à



Exemple : un système de webmail



ℓ = login entré,

g = mot de passe correct,

b = mot de passe incorrect,

a = actions diverses,

d = déconnexion.

Une exécution

~~~ Un premier essai de mot de passe infructueux, puis une action, avant d'être déconnecté par *timeout* :

$$\begin{aligned} (\text{idle}; 0) &\xrightarrow{2,71} (\text{idle}; 2, 71) \xrightarrow{\ell} (\text{passwd\_1}; 0) \xrightarrow{12,34} (\text{passwd\_1}; 12, 34) \\ \dots &\xrightarrow{b} (\text{passwd\_2}; 0) \xrightarrow{5,73} (\text{passwd\_2}; 5, 73) \xrightarrow{g} (\text{auth}; 0) \xrightarrow{215,21} \\ &\dots (\text{auth}; 215, 21) \xrightarrow{a} (\text{auth}; 0) \xrightarrow{600} (\text{auth}; 600) \xrightarrow{d} (\text{idle}; 600) \end{aligned}$$

~~~ Le mot temporisé généré par cette exécution :

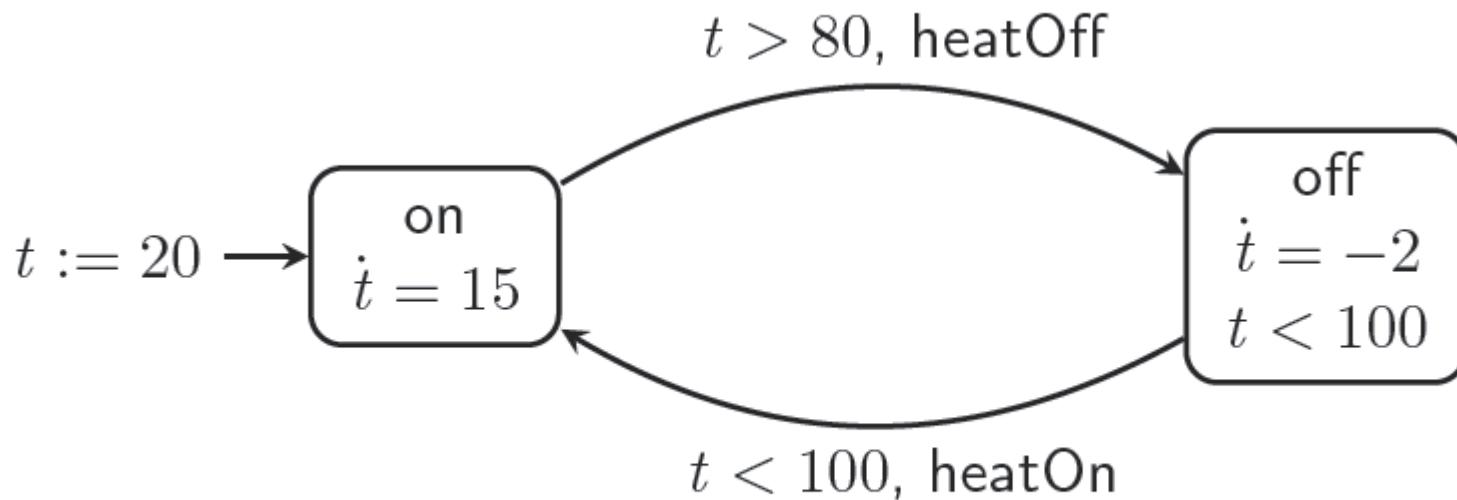
$$(\ell; 2, 71)(b; 15, 05)(g; 20, 78)(a; 235, 99)(d; 835, 99)$$

Automates hybrides

- Les automates temporisés sont limités :
 - Toutes les horloges ont la même vitesse.
 - N'autorise que des comparaisons simples et les remises à zéro
 - Modélise le temps, mais n'est pas adaptable à la modélisation d'autres modèles continus
- Les automates hybrides (HA) :
 - Les horloges sont remplacées par des variables.
 - Dans chaque état, chaque variable varie linéairement: $x = -1/2$
 - Les comparaisons peuvent faire intervenir des combinaisons linéaires de variables : $x + 2y \geq 3/2$
 - Les mises à jour peuvent aussi être plus complexes : $x := y + 1/3$

Exemple d'un automate hybride

- Un chauffe-eau est soit allumé e soit éteint.
- La température de l'eau est modélisée par la variable t .
- Elle croit de 15°C par minute lorsque le chauffe-eau est allumé ($t = 15$), et décroît de 2°C par minute lorsque le chauffe-eau est éteint ($t = -2$)
- Initialement, l'eau est à 20°C , et elle doit être maintenue entre 80 et 100°C



Comparaison entre les AT et les AH

- Les automates hybrides sont plus expressifs : on peut modéliser plein de choses avec.
 - On peut les manipuler avec HyTech.
- Par contre il est plus difficile de vérifier des choses sur les automates hybrides.
- La vérification est possible sur les automates temporisés.
 - En particulier le problème d'accessibilité est décidable sur les A.T