Automates temporisés

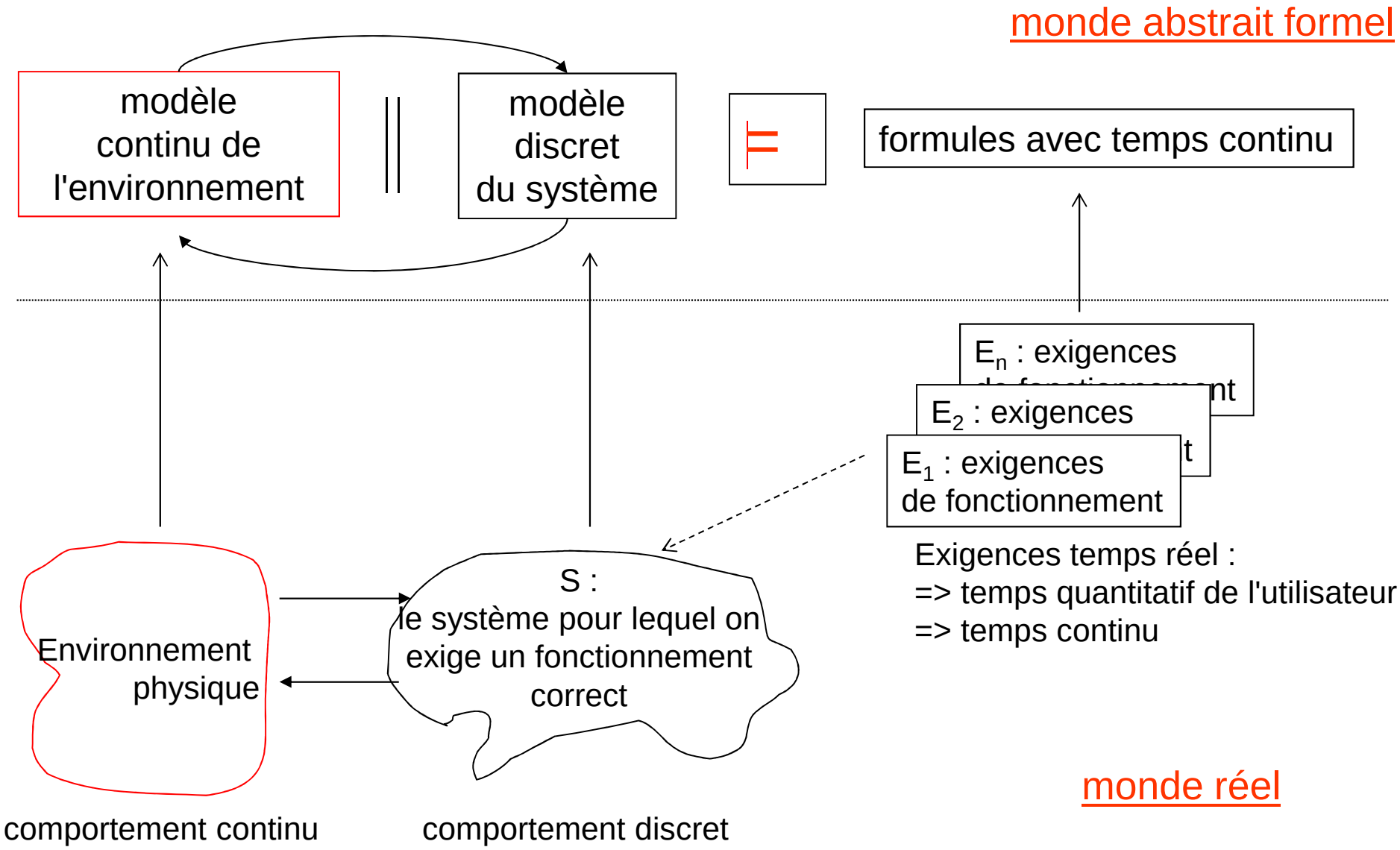
Amal El Fallah Seghrouchni

Amal.Elfallah@lip6.fr

Plan

- Introduction
- Définition d’un automate temporisé
- Composition d’automates temporisés
- Automates hybrides
- Conclusion

Le problème à résoudre



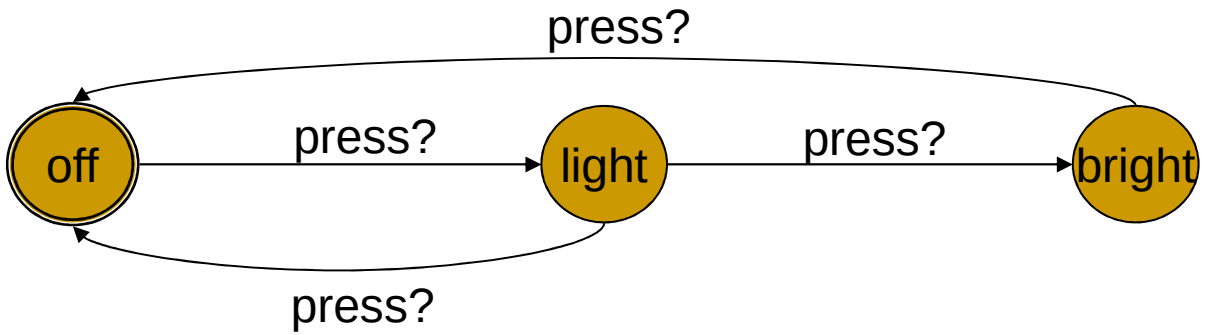
Besoins

- **La validation d'un système informatique** (ex. un protocole) **respectant des exigences temps réel requiert :**
  - La formalisation de ces exigences
  - La modélisation de l'environnement du système en prenant en compte le temps
- **Besoin d'un même formalisme pour les exigences et les modèles de systèmes ou d'environnements.**

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)
- lorsque la lampe est fortement lumineuse (bright) un appui sur "press" l'éteint (off)

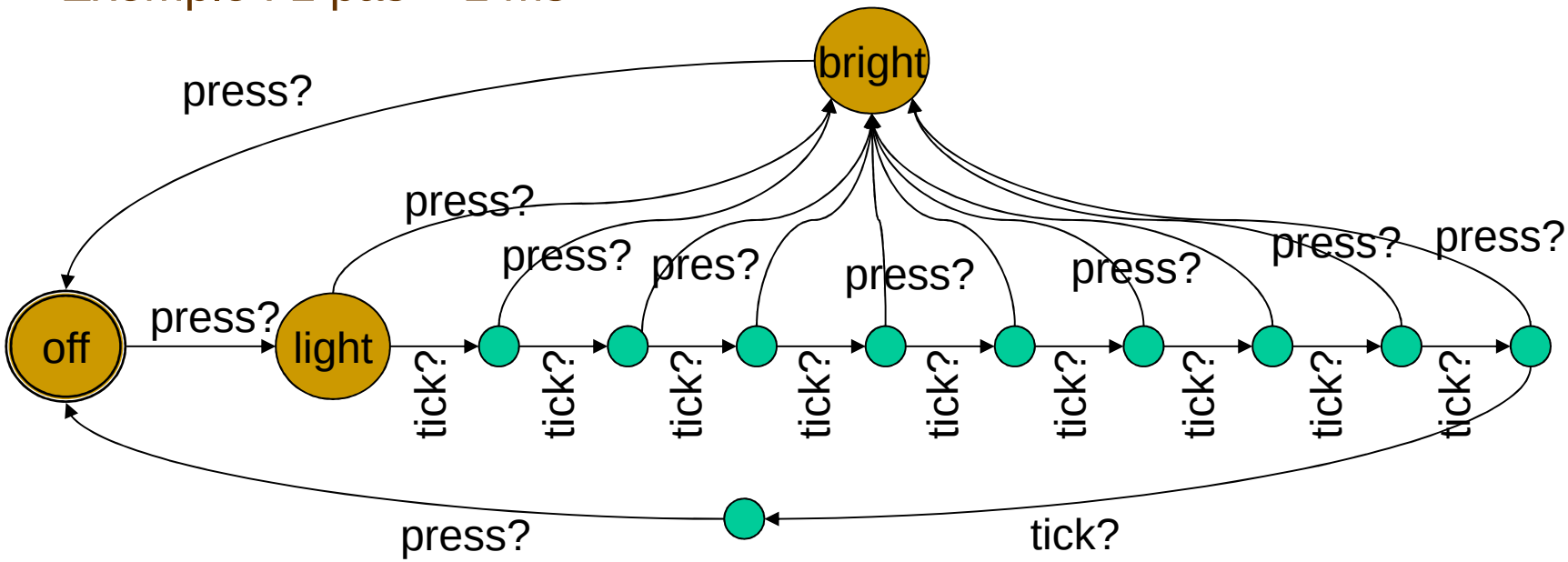
? double appui = deux impulsions en moins de 10ms



Il faut un formalisme pour représenter les durées des actions

Première solution : temps discret

Exemple : 1 pas = 1 ms



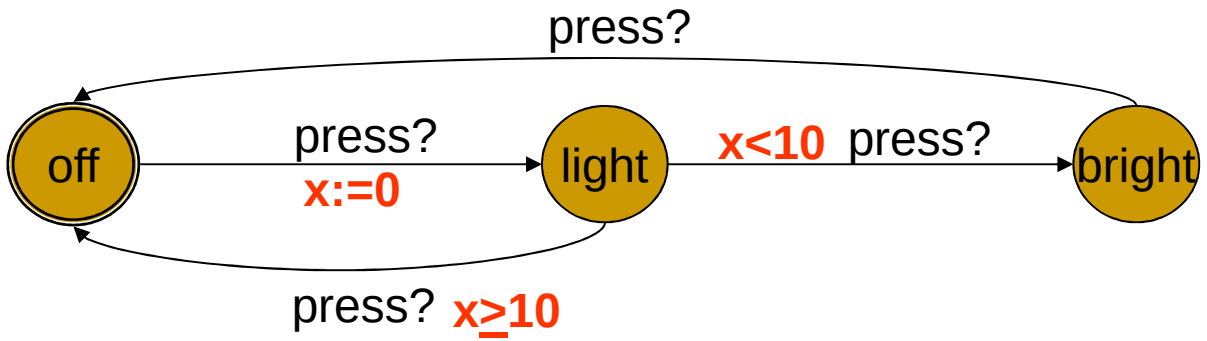
Mais :

- 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

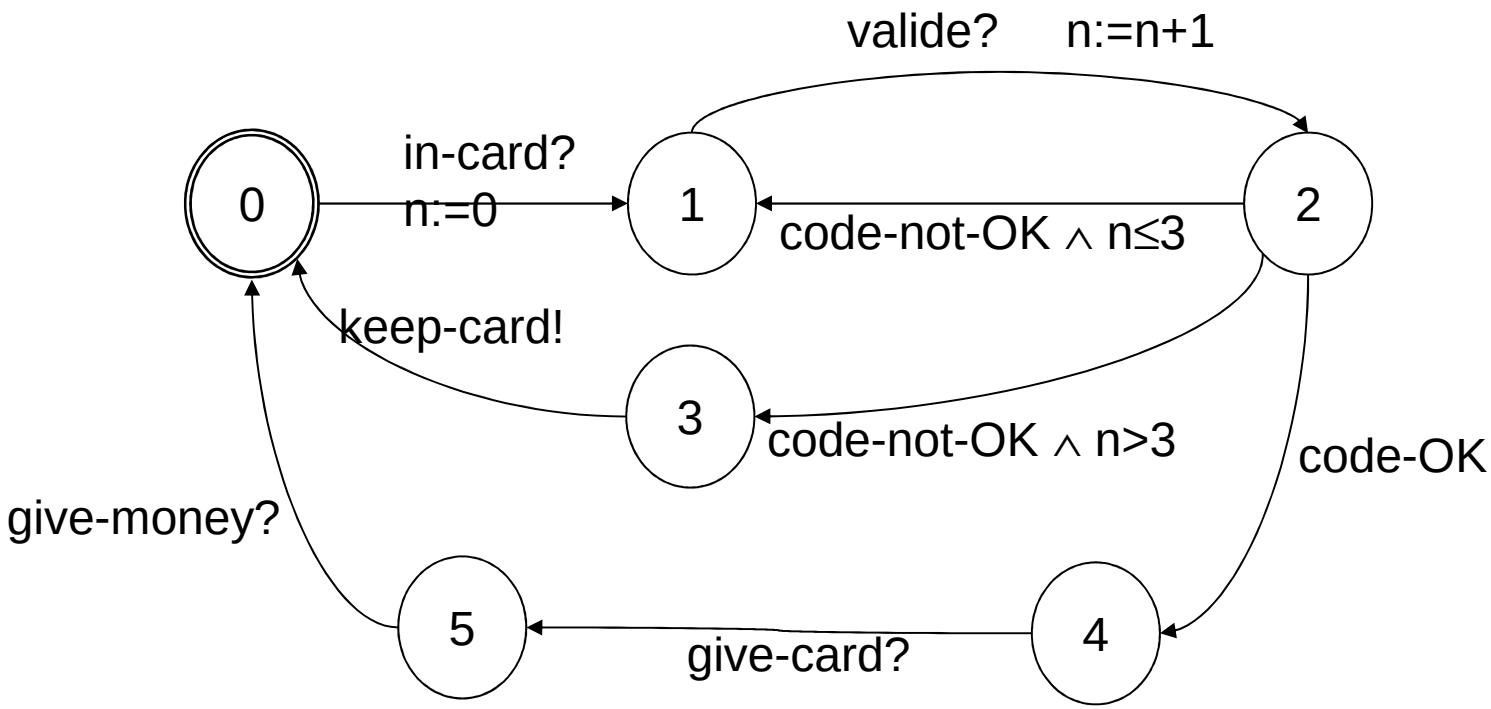
Introduire une variable réelle
modélisant l'écoulement du temps et exprimée en ms



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

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

Distributeur automatique de billets

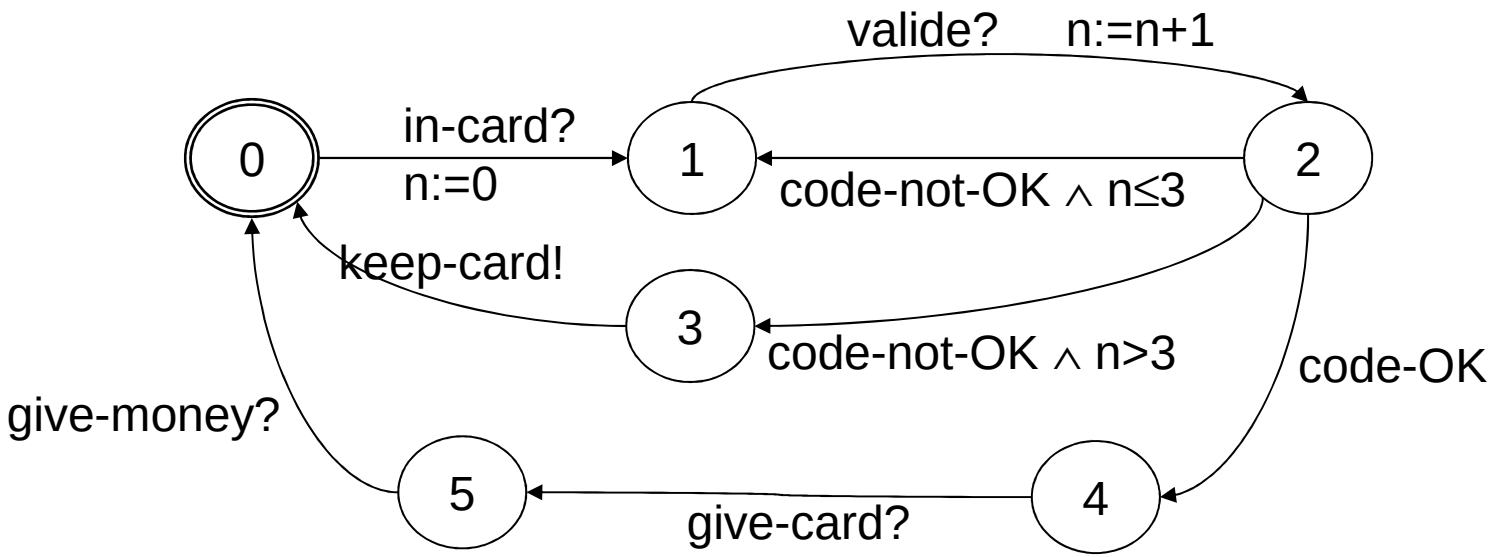


Exigences temporelles du DAB

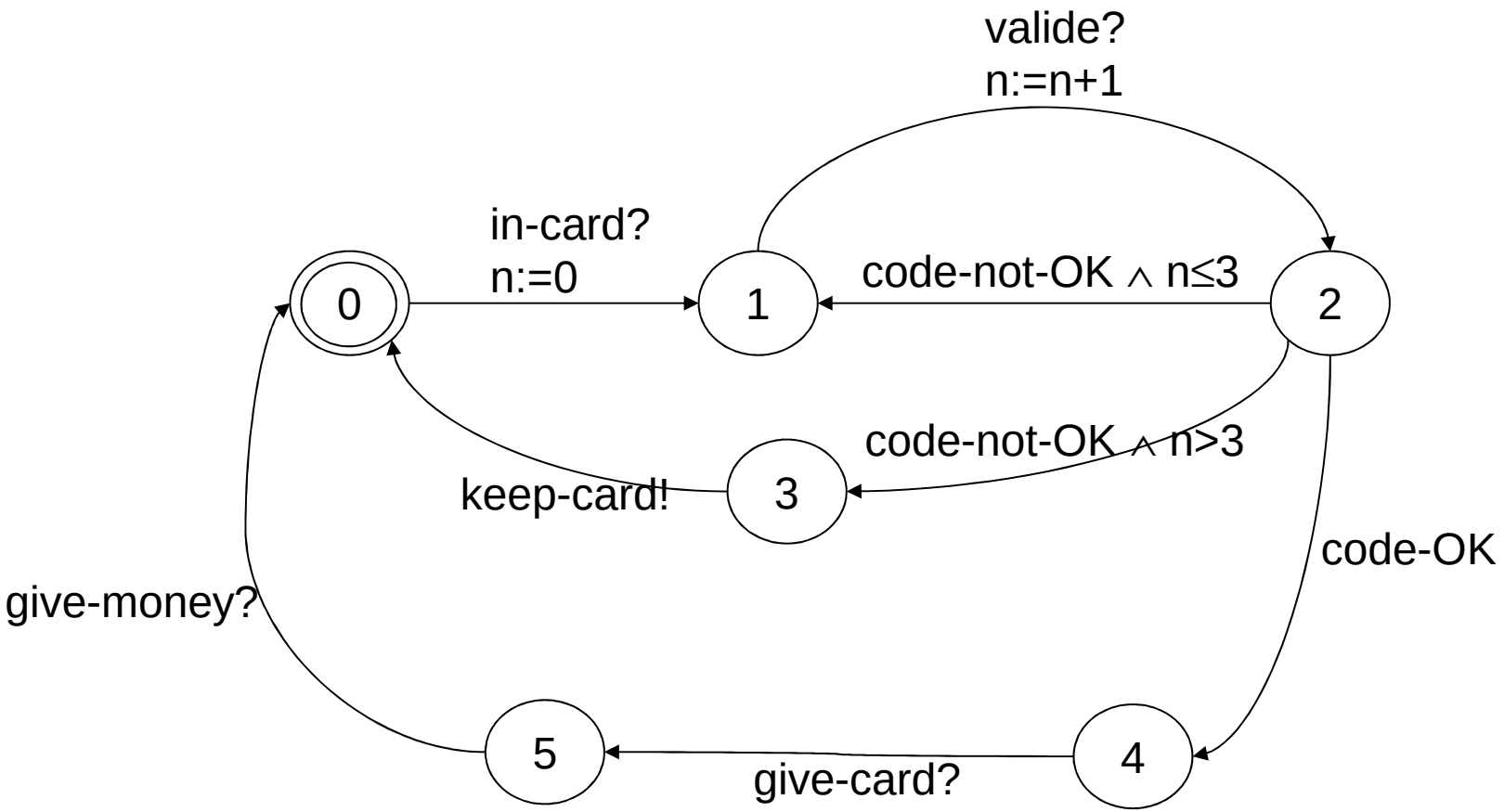
Comment prendre en compte des exigences temps réel telles que :

- 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

…



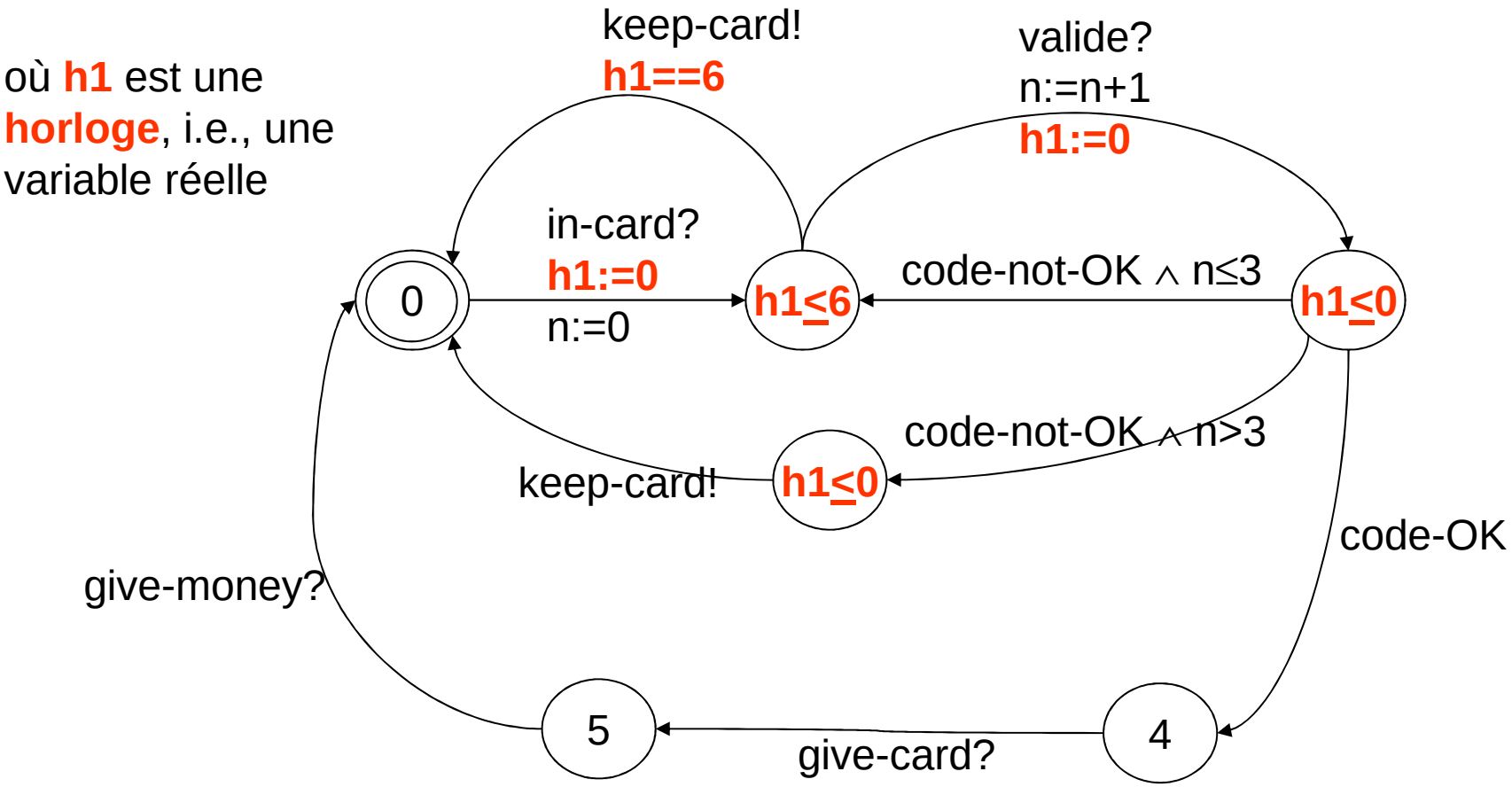
Suite du DAB


valide?
n:=n+1
in-card?
n:=0
code-not-OK ∧ n≤3
0
1
2
code-not-OK ∧ n>3
keep-card!
3
code-OK
give-money?
5
4
give-card?

Suite du DAB

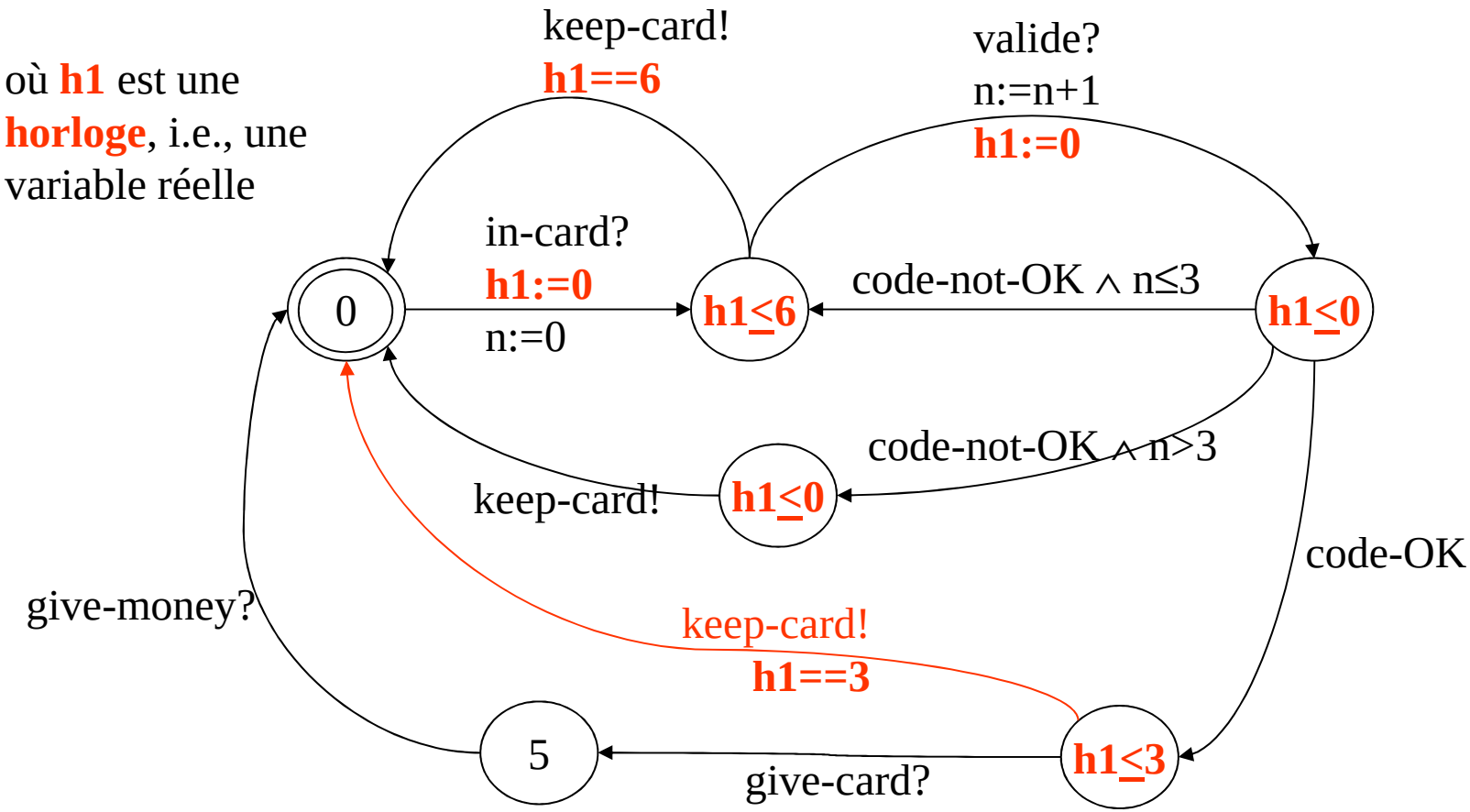
- lors de chaque essai, le client doit valider son code en moins de 6 secondes, sinon la carte est conservée

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



Suite du DAB …

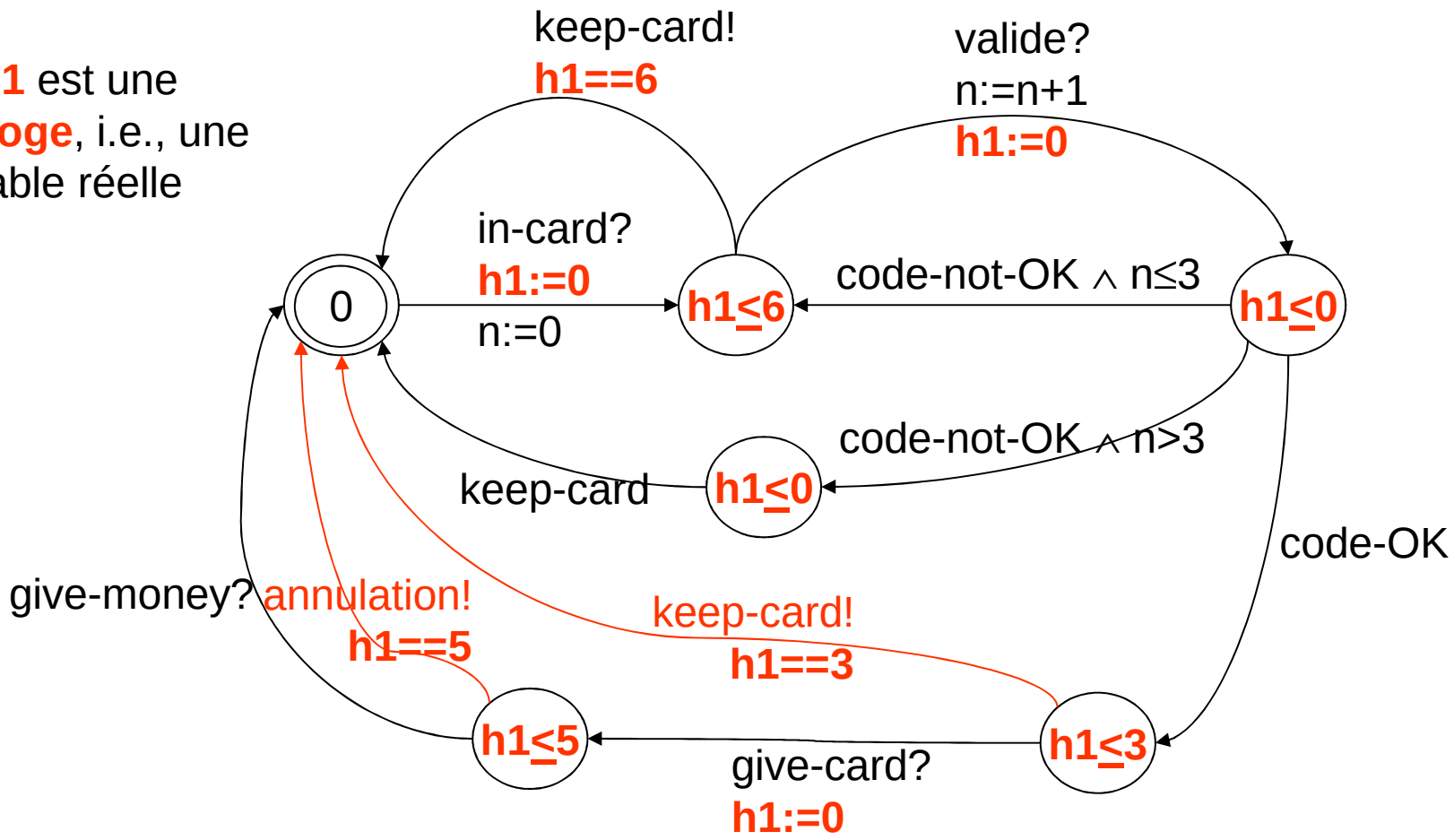
- lorsque la transaction réussit, le client doit récupérer sa carte en moins de 3 secondes, sinon celle-ci est conservée



Suite du DAB

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

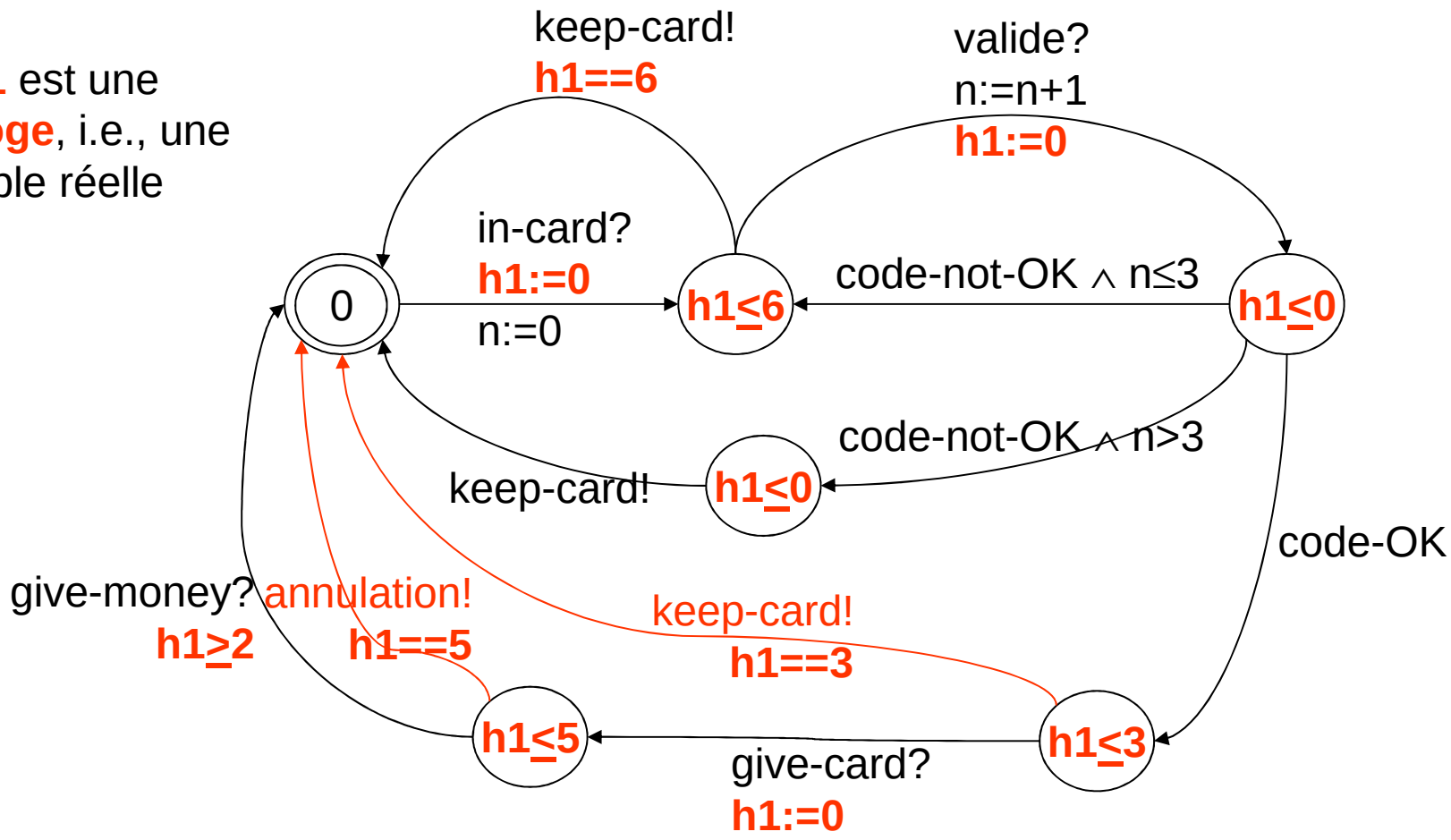
où **h1** est une **horloge**, i.e., une variable réelle



Suite du DAB

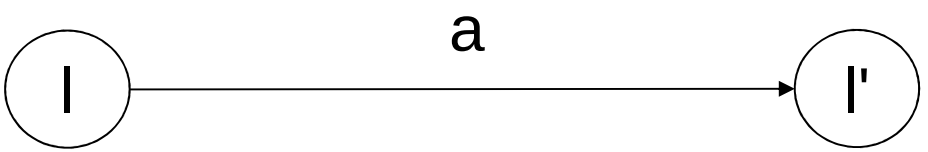
- les billets ne peuvent être récupérés par le client qu'au plus tôt 2 secondes après avoir récupéré sa carte

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



Automate temporisé

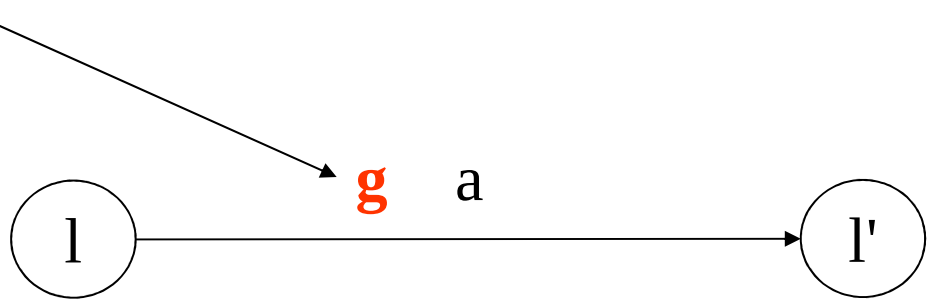
- **Généralisation : un automate temporisé, c'est :**
  - des nœuds et des transitions entre ces nœuds étiquetées par des actions



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 ~ k ou x ~ k
avec x et y deux horloges
avec k une constante entière
et ~ ∈ {==, <, >, ≤, ≥}

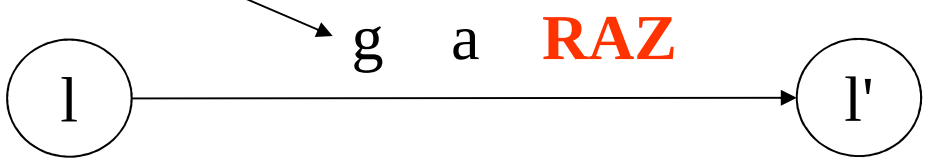


Automate temporisé

- **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 ~ k ou x ~ k
avec x et y deux horloges
avec k une constante entière
et ~ ∈ {==, <, >, ≤, ≥}

Remise à zéro de toutes les horloges de RAZ

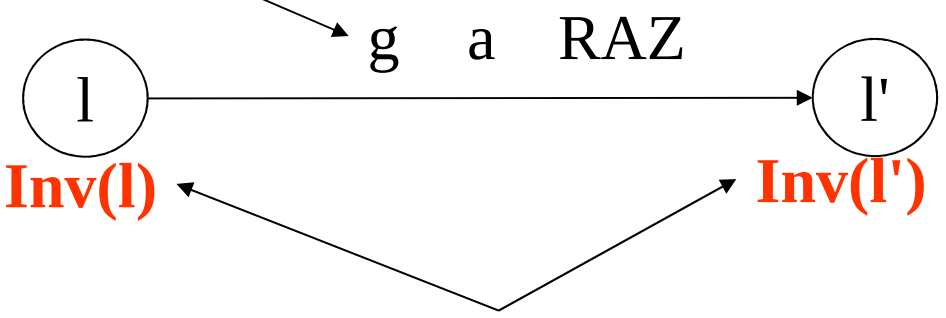


Automate temporisé

- **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 ~ k
avec x et y deux horloges
avec k une constante entière
et ~ ∈ {==, <, >, ≤, ≥}

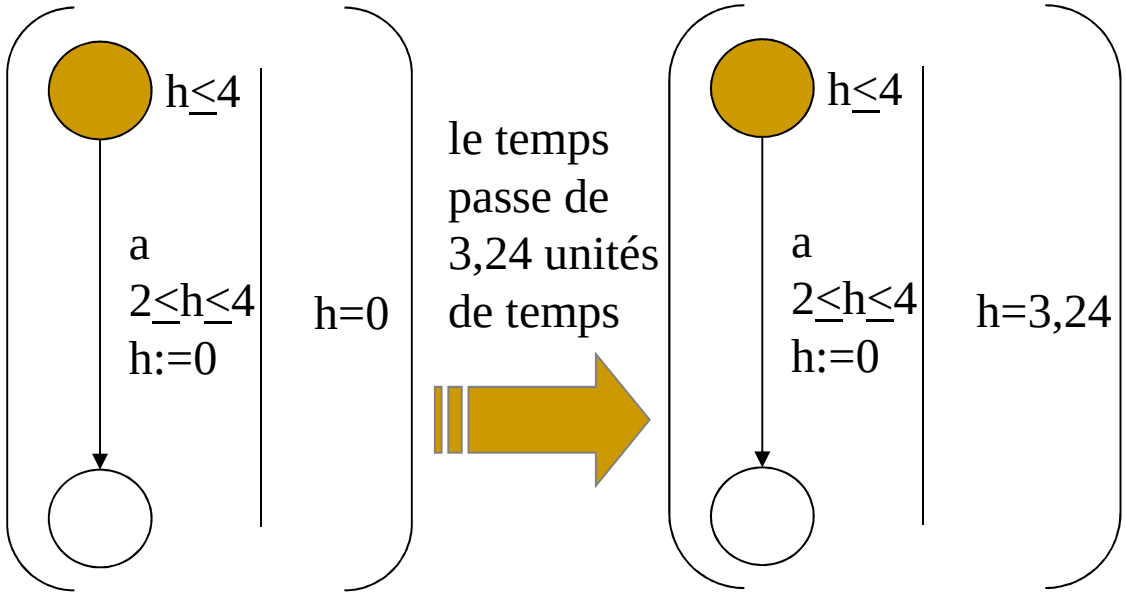
Remise à zéro de toutes
les horloges de RAZ



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.

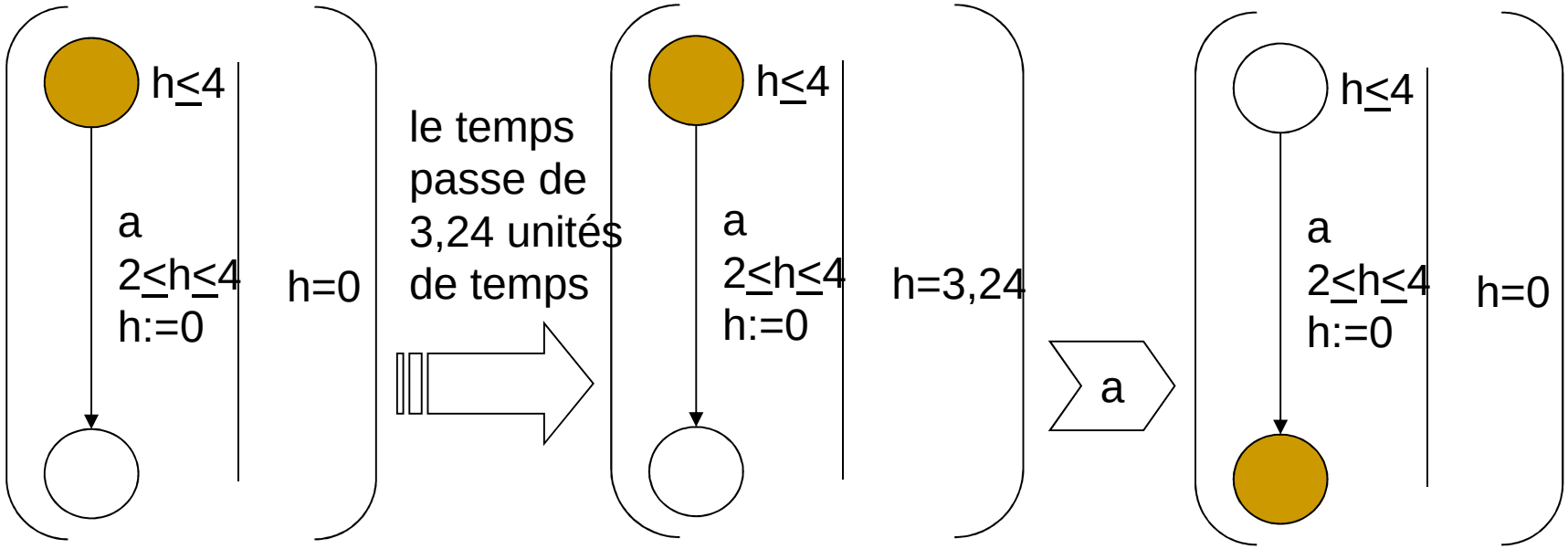
- **un automate temporisé évolue**
  - soit en laissant passer le temps sans changer de nœud
  - soit en tirant une transition sans laisser passer le temps

=> 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



Syntaxe d'un A.T.

Un automate temporisé est (AT) défini par :

- un ensemble fini d'états Q
- un ensemble fini d'horloges X = {x, y, z ..}
- un alphabet fini $\sum$= {a,b, c..}
- un ensemble Δ de transitions dont chacune est formée :
  - d'une garde g : une condition c (conjonction) sur les horloges x dans {>, <, =, ≥, ≤},
  - d'une lettre de l'alphabet de $\sum$,
  - de remises à zéro r : x:=0, y :=0, .. (parfois noté comme le sous-ensemble de raz noté {x, y, z..})
  - d'invariants Inv(q) dans les états (invariants)
- q0 : nœud initial
- d'états finaux

➢ *On peut également mettre des variables booléennes dans les gardes*

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 :

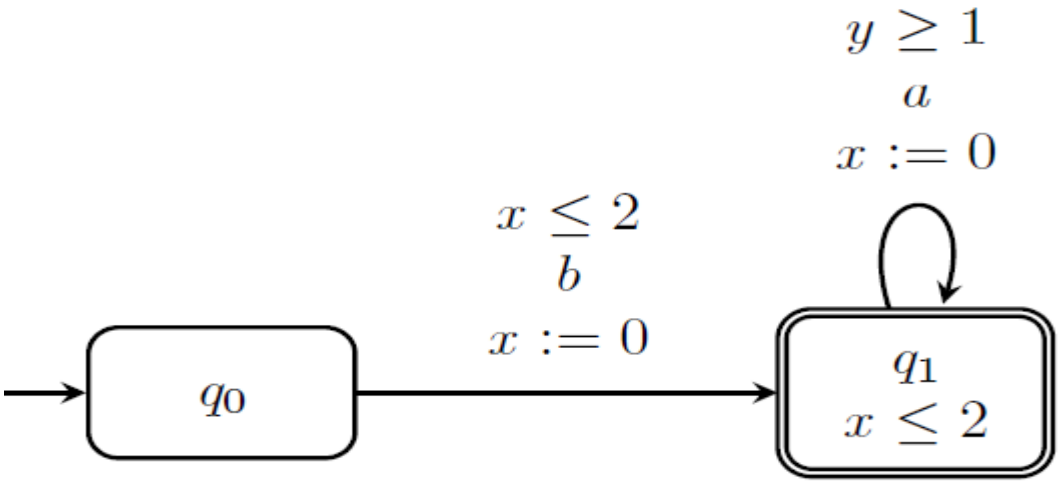
    Si $q \xrightarrow{g;a;r} q' \in \Delta$, et , $v \Rightarrow g$ et $v[r:= 0] \Rightarrow Inv(q')$

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

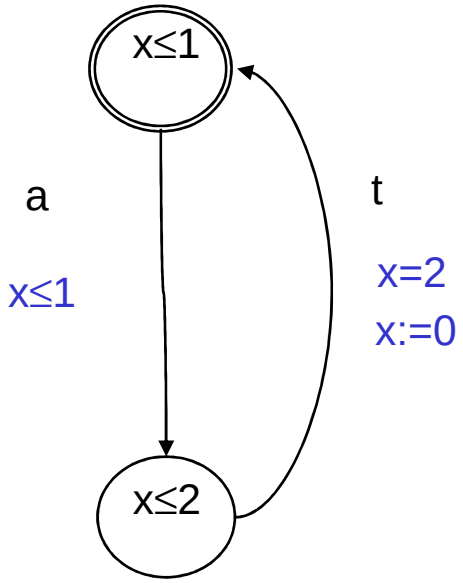
Exécution d'au A.T.

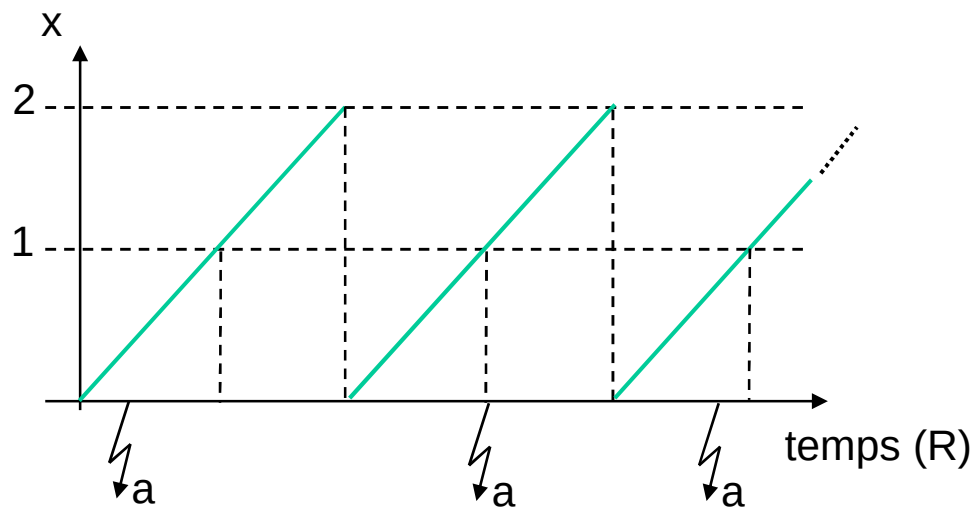
- **Un exécution est un chemin dans le système de transitions temporisé où s'alternent les pas de temps et les transitions discrètes**
  - (q0; (0;0)) $\xrightarrow{1,2}$ (q0; (1,2;1,2)) $\xrightarrow{x\leq 2,b,x:=0}$ (q1; (0;1,2)) $\xrightarrow{0,37}$ (q1; (0,37;1,57)) $\xrightarrow{y\geq 1,a,x:=0}$ (q1; (0;1,57))

- **Une exécution génère un mot temporisé: une suite d'actions couplées**
  - (b;1,2) (a; 1,57)

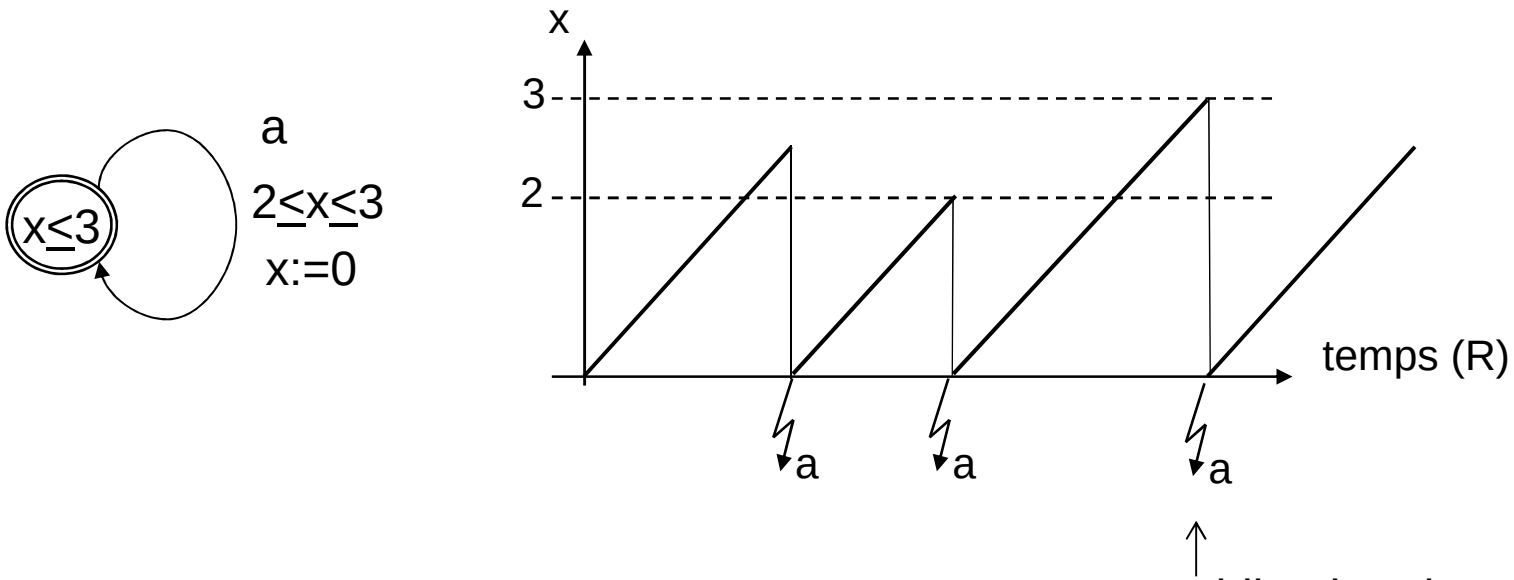


Exemple





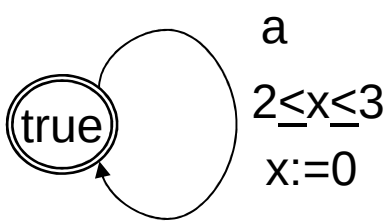
Exemple

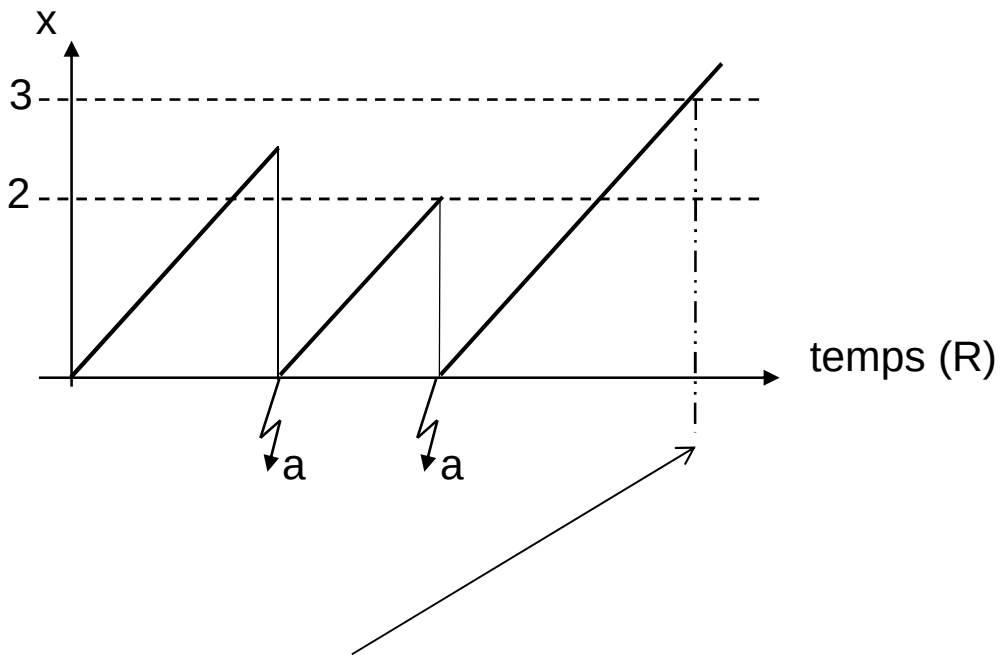


=> l'invariant interdit de rester dans le nœud lorsque x devient supérieur à 3

=> franchissement obligatoire de la transition avant que x dépasse 3

Exemple



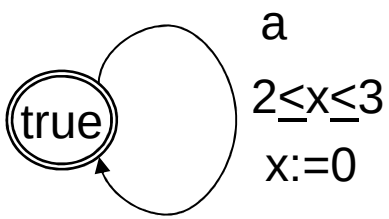


à partir de cet instant, la transition n'est plus franchissable

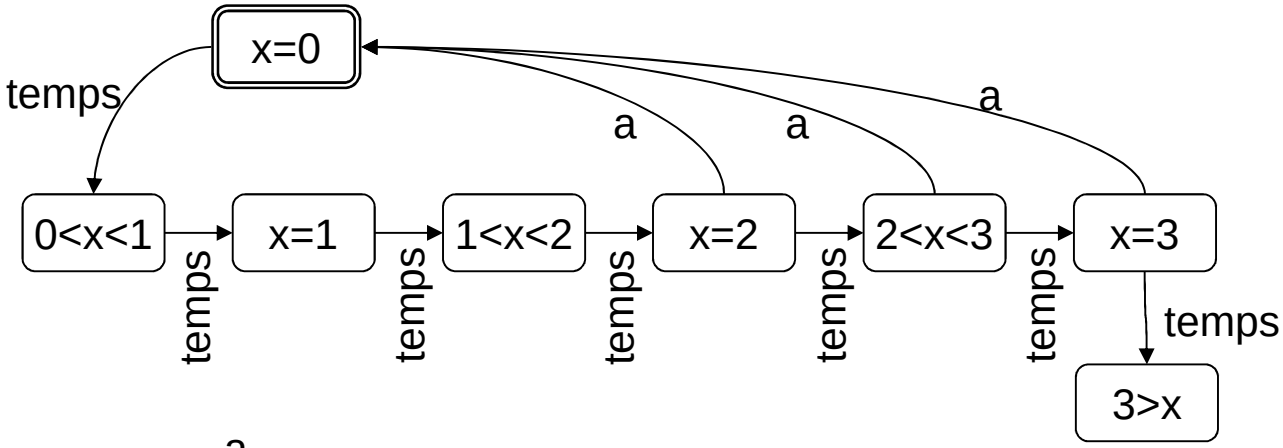
=> rien ne force la transition à être franchie entre 2 et 3.

Exemple

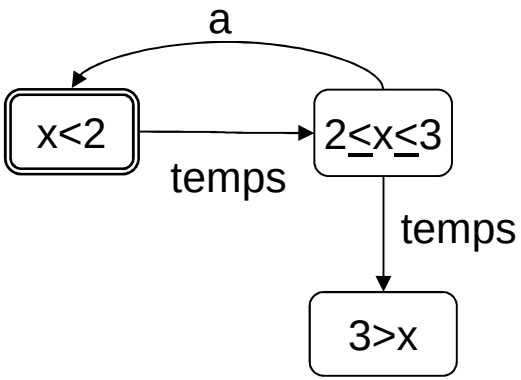
Remarque : il existe un automate discret équivalent :



équivalent à

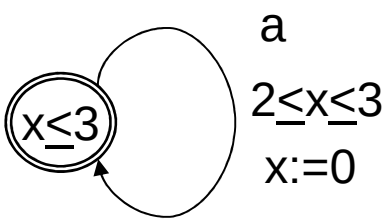


équivalent à

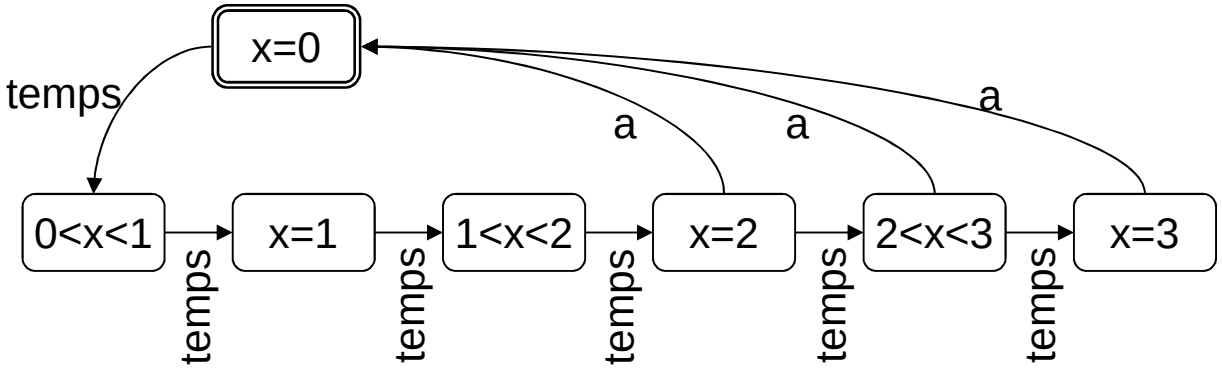


Exemple

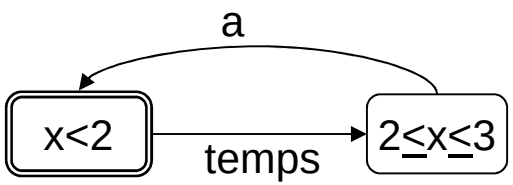
Remarque : il existe un automate discret équivalent :



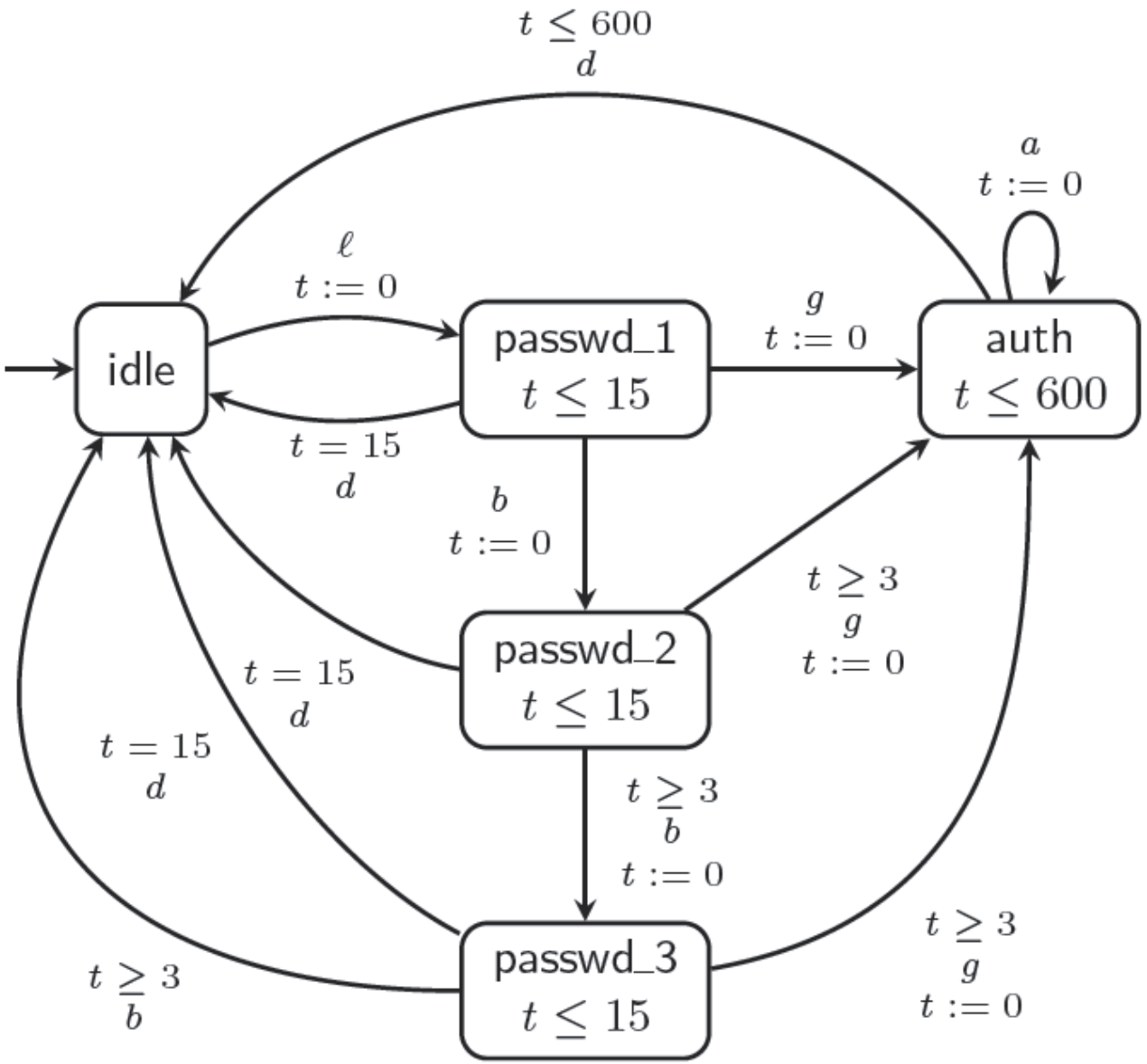
équivalent à



équivalent à



Exemple : un système de webmail



$\ell =$ login entré,

$g =$ mot de passe correct,

$b =$ mot de passe incorrect,

$a =$ actions diverses,

$d =$ déconnexion.

Une exécution

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

$$(\mathsf{idle}\,;\,0) \xrightarrow{2,71} (\mathsf{idle}\,;\,2,71) \xrightarrow{\ell} (\mathsf{passwd\_1}\,;\,0) \xrightarrow{12,34} (\mathsf{passwd\_1}\,;\,12,34)$$
$$\ldots \xrightarrow{b} (\mathsf{passwd\_2}\,;\,0) \xrightarrow{5,73} (\mathsf{passwd\_2}\,;\,5,73) \xrightarrow{g} (\mathsf{auth}\,;\,0) \xrightarrow{215,21}$$
$$\ldots (\mathsf{auth}\,;\,215,21) \xrightarrow{a} (\mathsf{auth}\,;\,0) \xrightarrow{600} (\mathsf{auth}\,;\,600) \xrightarrow{d} (\mathsf{idle}\,;\,600)$$

$\rightsquigarrow$ Le mot temporisé généré par cette exécution :

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

Besoin de modularité
Composition d’ A.T.

- **Vérifier qu'un système fonctionne correctement nécessite la modélisation :**
  - du système
  - de son environnement
  - de l'utilisateur exprimant des exigences de bon fonctionnement (scénarii attendus ou situations indésirables)
- **Description du système (complet) sous la forme d'un ensemble d'automates temporisés interagissant**
  - interaction par synchronisation des automates sur des actions communes
  - même temps (et donc même taux de progression des horloges) pour tous les automates
- **Notion de produit synchronisé d'automates temporisés !**

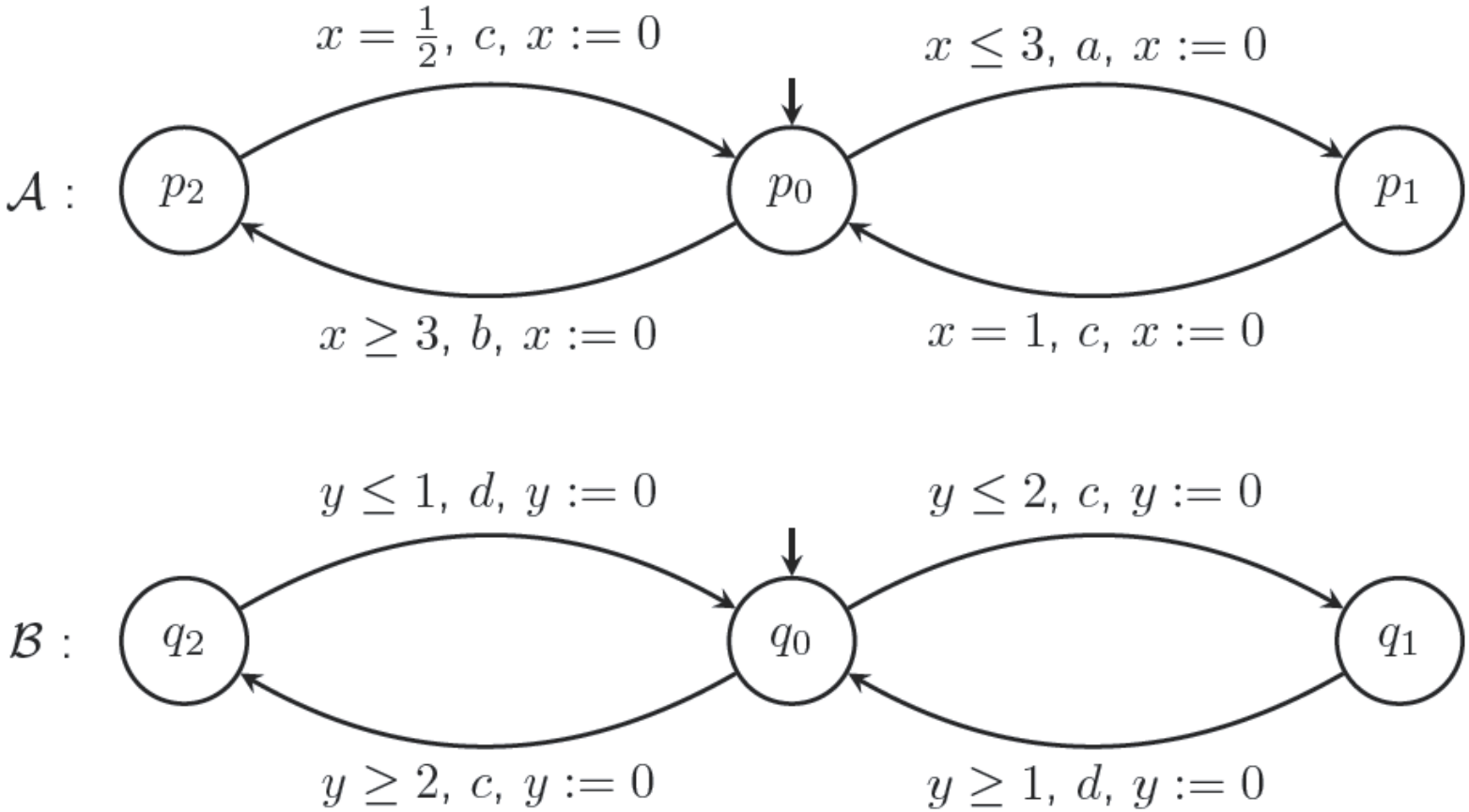
Composition d'automates

Si $\mathcal{A}_1 = \langle Q_1, q_1^i, \Sigma_1, X_1, \Delta_1, Inv_1 \rangle$ et $\mathcal{A}_2 = \langle Q_2, q_2^i, \Sigma_2, X_2, \Delta_2, Inv_2 \rangle$ sont des automates temporisés avec $X_1 \cap X_2 = \emptyset$, la synchronisation de $\mathcal{A}_1$ et $\mathcal{A}_2$ est l'automate temporisé $\mathcal{A}_1 || \mathcal{A}_2 = \langle Q, q^i, \Sigma, X, \Delta, Inv \rangle$ où

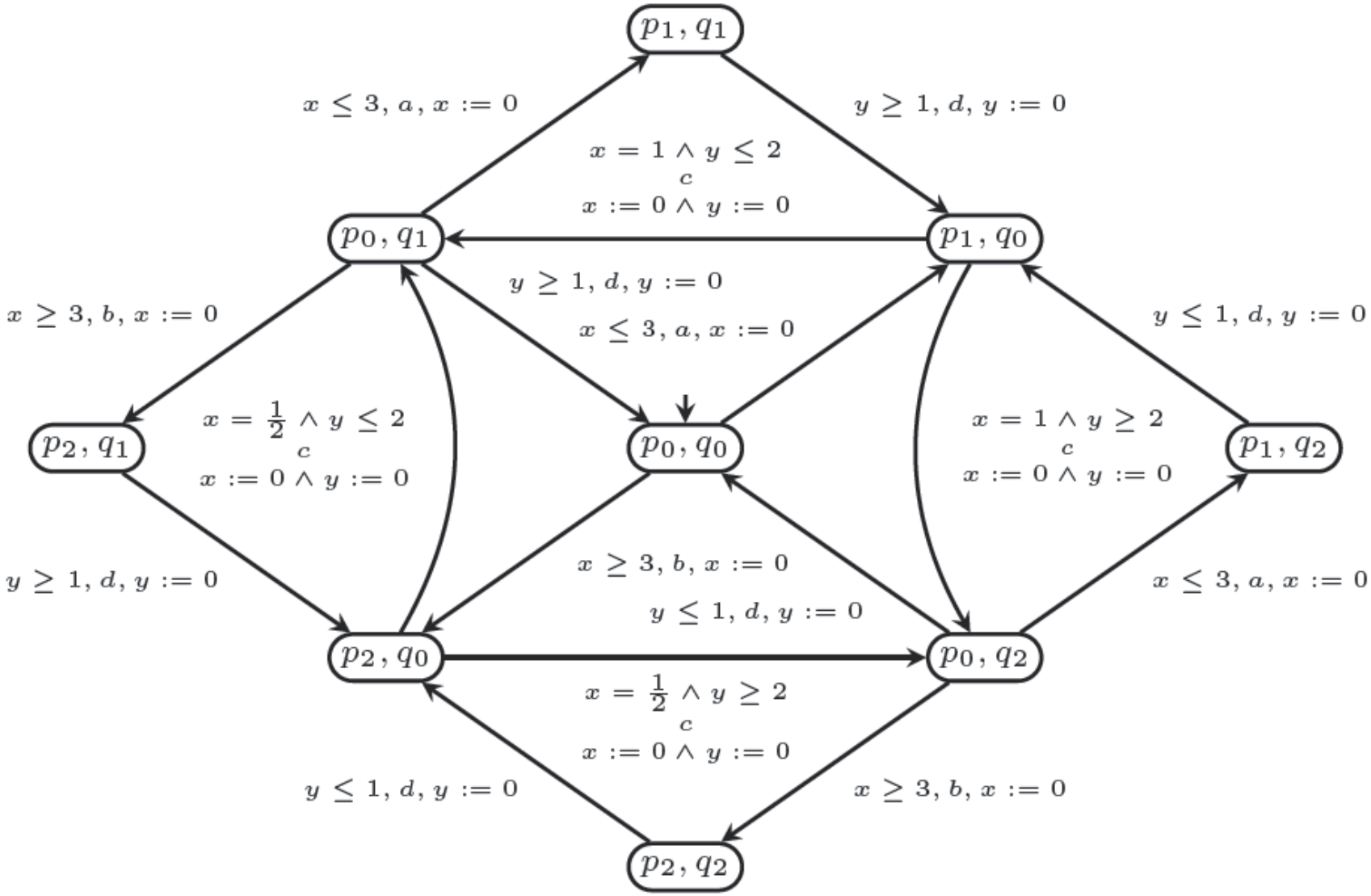
- $Q = Q_1 \times Q_2$
- $q^i = (q_1^i, q_2^i)$
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $X = X_1 \cup X_2$
- Supposons que $(q_1, g_1, a_1, r_1, q_1') \in \Delta_1$ et $(q_2, g_2, a_2, r_2, q_2') \in \Delta_2$.
  - si $a_1 = a_2 = a \in \Sigma_1 \cap \Sigma_2$, on se synchronise : $((q_1, q_2), g_1 \wedge g_2, a, r_1 \cup r_2, (q_1', q_2')) \in \Delta$
  - si $a_1 \in \Sigma_1 \setminus \Sigma_2$, seul $\mathcal{A}_1$ évolue : $((q_1, q_2), g_1, a_1, r_1, (q_1', q_2)) \in \Delta$
  - si $a_2 \in \Sigma_2 \setminus \Sigma_1$, seul $\mathcal{A}_2$ évolue : $((q_1, q_2), g_2, a_2, r_2, (q_1, q_2')) \in \Delta$
- $\forall (q_1, q_2) \in Q_1 \times Q_2,\ Inv(q_1, q_2) = Inv_1(q_1) \wedge Inv_2(q_2)$

Exemple

Automates originaux, synchronisation sur c



Composition

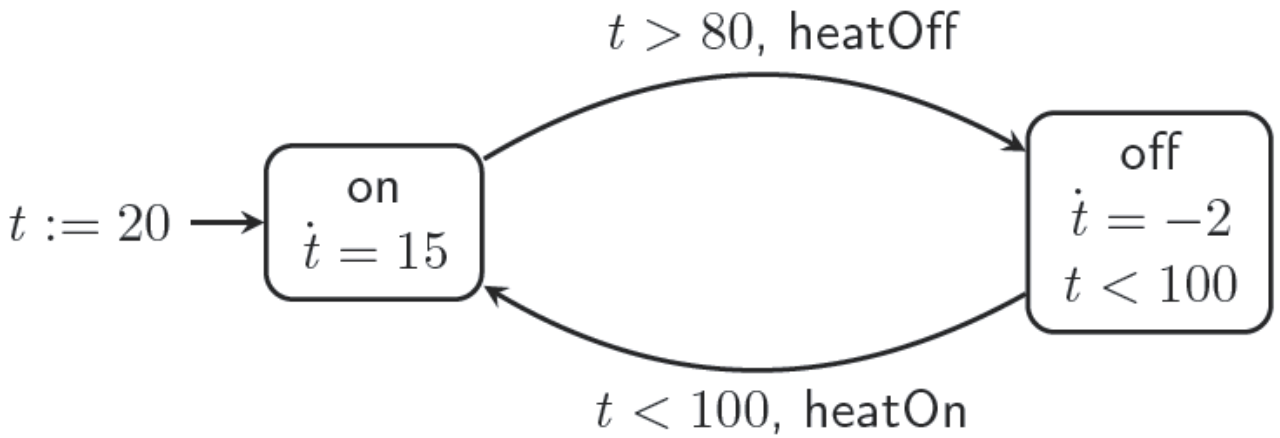


Automates hybrides

- **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) :**
  - I Les horloges sont remplacées par des variable.
  - 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 ($\dot{t} = 15$), et décroit de 2°C par minute lorsque le chauffe-eau est éteint ($\dot{t} = -2$)
  - 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 les manipuler en TME 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

Conclusion

- **Les automates temporisés permettent d'associer des durées aux actions**
- **Les réseaux de Petri temporisés permettent d'associer un intervalle durant lequel une action sera exécuté**
- **Les intervalles d'Allen permettent d'associer des intervalles où l'action peut se produire.**

- **Ces trois formalismes sont très utilisés en planification mono-agent et aussi multi-agents car ils permettent la synchronisation et/ou la concurrence.**

Exemples supplémentaires

Modélisation de la communication

Besoin : modéliser une communication synchrone par ***envoi/réception*** de messages

Idée :

- étiquette de transition m! = envoi de m
- étiquette de transition m? = réception de m
- synchronisation de m! et m? par une contrainte de synchronisation

Conséquence :

- nécessité d'une ***action vide*** (qui ne fait rien) = •
- pour permettre la synchronisation de m! et m? uniquement (pas de synchronisation avec les actions des autres AT du produit synchronisé)

$\Rightarrow$ extension de l'alphabet des actions $\sum' = \sum \cup \{\bullet\}$

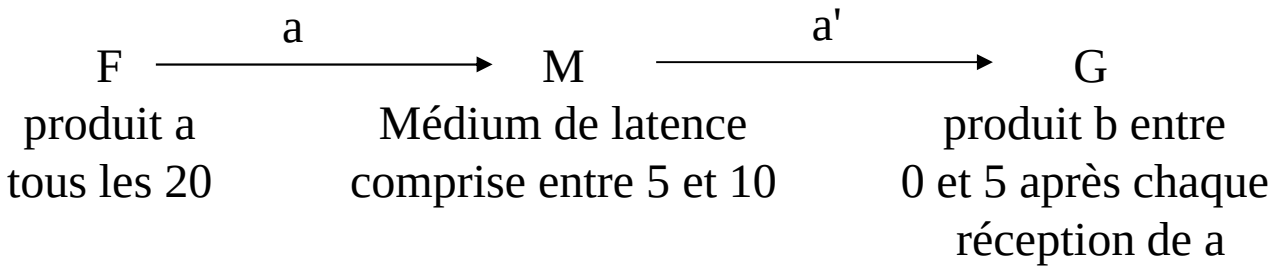
$\Rightarrow$ extension de la définition des AT par ajout de la transition vide :

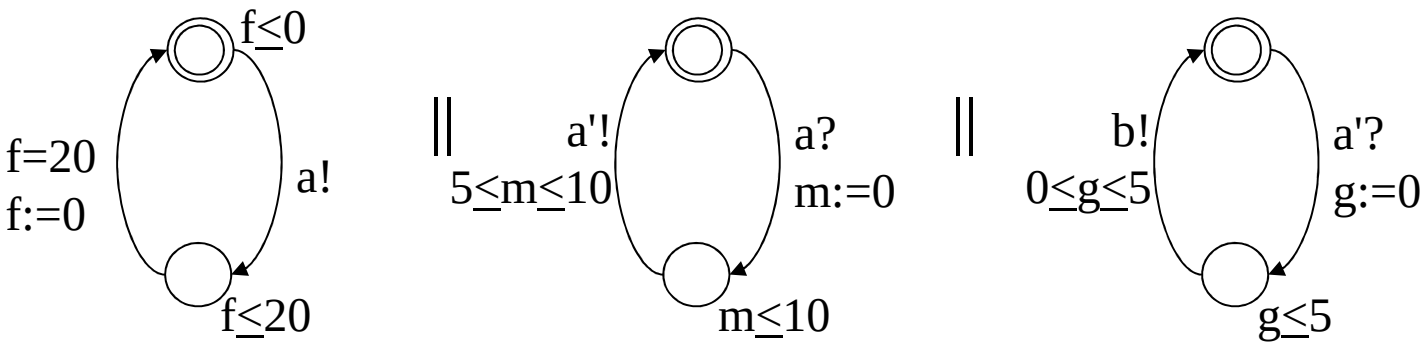
$$\forall q \in Q, (q, true, \bullet, \varnothing, q) \in \Delta$$

$\Rightarrow$ synchronisation de m! et m? uniquement par une contrainte du type

$$I=(\bullet, \ldots, m!, \bullet, \ldots, \bullet, m?, \bullet, \ldots, \bullet)$$

Emission / Réception synchronisées





sync={(a!, a?, •), (•, a'!, a'?), (•, •, b!)}