

# Modélisation des systèmes

**MODÉLISATION.  
EXEMPLES DE MODÈLES.  
CONCEPTS GÉNÉRAUX.  
AUTOMATES AVEC VARIABLES.  
AUTOMATES COMMUNICANTS.  
AUTOMATES TEMPORISÉES  
ABSTRACTION**





# MODÉLISATION



# Modéliser des systèmes

- ❑ Un modèle d'un système doit décrire fidèlement son comportement sans ambiguïté.
- ❑ Généralement, les systèmes sont décrits par des automates.





# Modéliser des systèmes

- ❑ Ces machines peuvent être :
  - ❑ simples comme les machines à états finis (automates) ou
  - ❑ plus complexes (pour les vrais programmes) comme les machines de Turing.
- ❑ Plus le formalisme utilisé est puissant, moins il peut être traité automatiquement.
- ❑ Le formalisme doit aussi être décidable.



# Automates

- ❑ Les systèmes qui s'apprêtent mieux au model-checking sont modélisables par des structures de kripke.
- ❑ Il est possible aussi de considérer des systèmes à états infinis, mais ceci sort du cadre du cours.
- ❑ Un automate est un modèle de machine évaluant d'état en état sous l'effet de transitions :
  - ❑ Les états sont représentés par des : 
  - ❑ Les transitions par des flèches :  $\longrightarrow$  ,
  - ❑ L'état initial est marqué :  $\rightarrow$  



# Automates

- Un automate est un terme générique utilisé pour désigner plusieurs concepts : système de transition étiqueté ou pas, structure de Kripke, etc..
- Un automate est en général un quadruplet  $M=(S, s_0, E, R, F)$  :
  - $S$  est un ensemble d'états,
  - $s_0 \in S$  est l'état initial.
  - $E$  est un ensemble d'étiquettes de transitions,
  - $R \subseteq S \times E \times S$  est un ensemble de transitions.
  - $F$  est un ensemble d'états finaux.

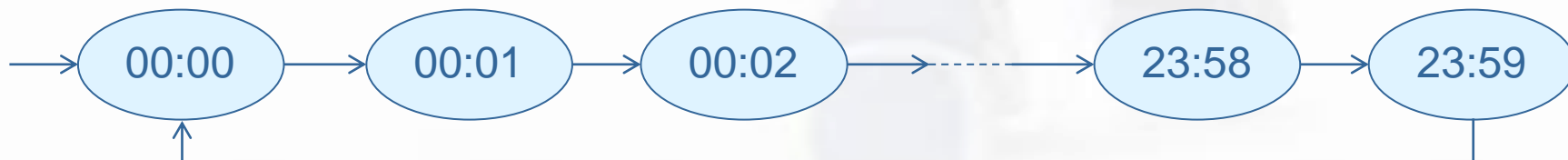


# EXEMPLES DE MODÈLES



# Exemple 1 : montre digitale

- Une montre digitale (qui indique seulement l'heure et la minute) peut être représentée par un automate :
- Chaque état donne l'heure et les minutes courantes  $\Rightarrow 24 \times 60$  états possibles.
- Une transition relie toute paire d'états distants d'une minute.







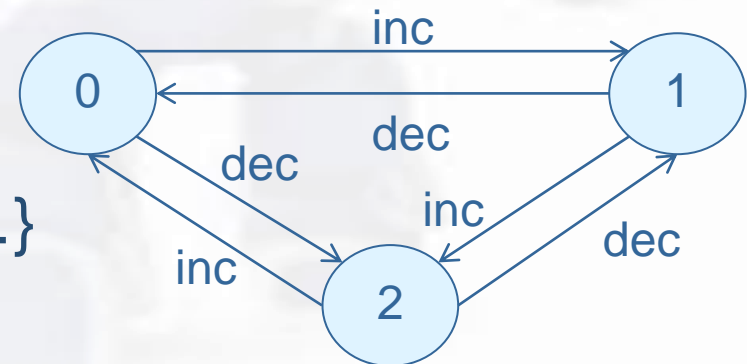
## Exemple 2 : compteur modulo 3 (M3)

- ❑ Les états correspondent aux valeurs du compteur.
- ❑ Les transitions traduisent les actions possibles sur le compteur : incrémentation ou décrémentation.

$M3 = (S, s_0, E, R)$

$S = \{0, 1, 2\}$ ,  $s_0 = 0$ ,  $E = \{\text{inc}, \text{dec}\}$

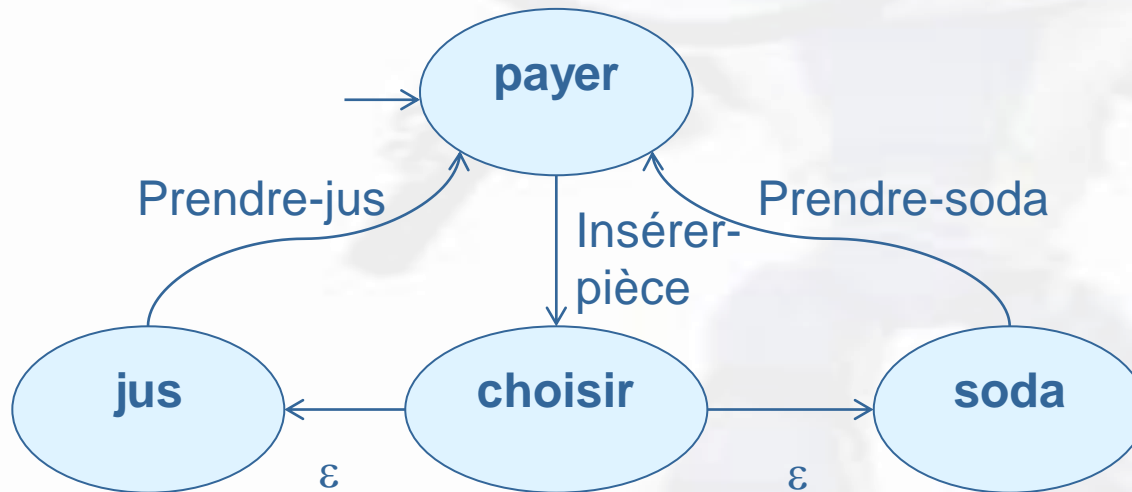
$R = \{(0, \text{inc}, 1), (1, \text{inc}, 2), \dots, (2, \text{dec}, 1), \dots\}$





# Exercice

□ Représentez une machine de distribution de boissons (jus et soda) par un automate.





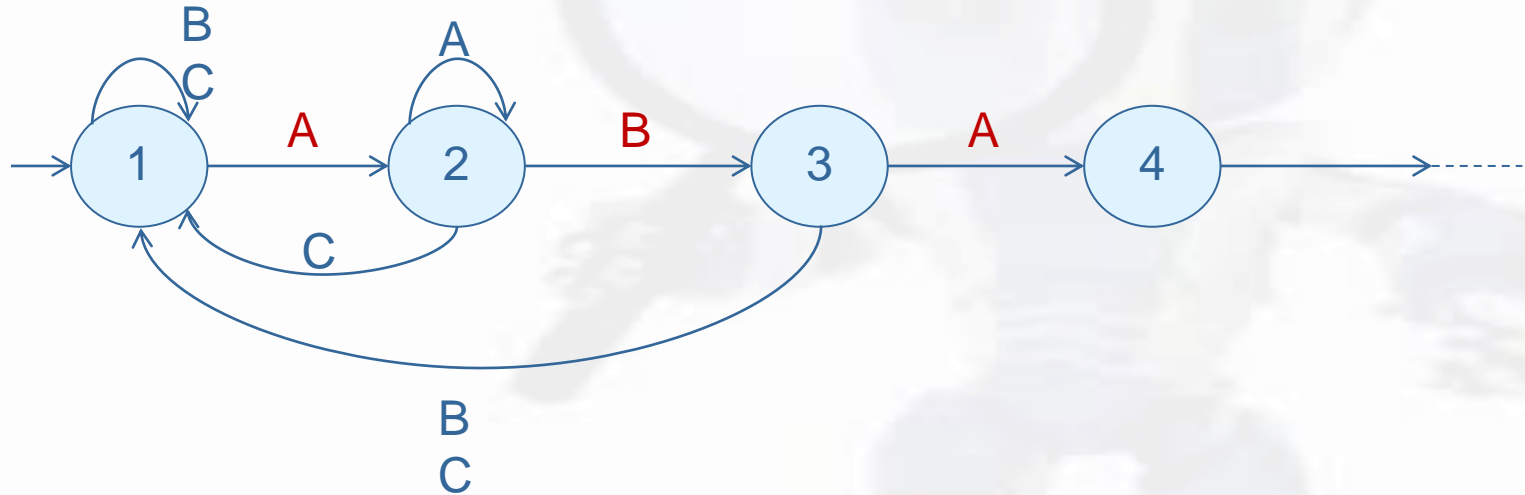
## Exemple 3 : Digicode

- ❑ Pour ouvrir la porte d'un immeuble par exemple.
- ❑ La porte s'ouvre dès qu'on a tapé la bonne combinaison (suite de caractères).
- ❑ Pour le but de l'exemple, on suppose que même si on commence par une mauvaise combinaison, il suffit de finir avec la bonne.
- ❑ 3 touches sont possibles : A, B, C.
- ❑ Le code est ABA.



# Modélisation

□ Automate à 4 états et 9 transitions :





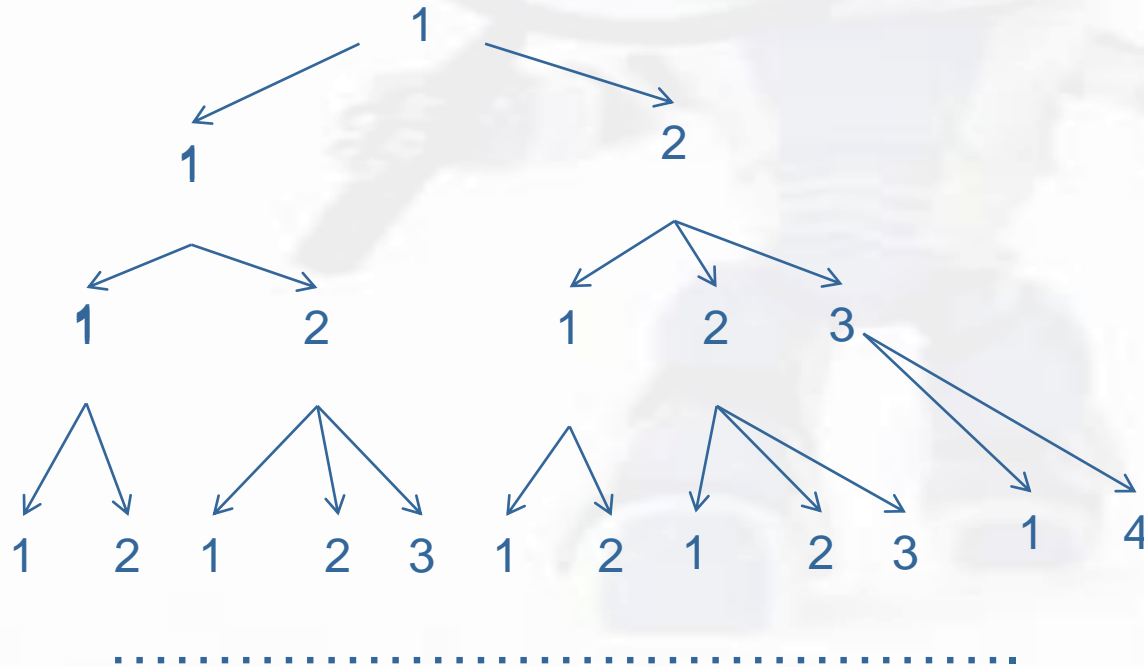
# Exécution

- ❑ Pour spécifier des propriétés, on s'intéresse moins aux transitions (mais possible) et on focalise sur l'exécution.
- ❑ Une **exécution** est une suite d'états décrivant une évolution du système : 11, 112, 1121, 12234, 112312234, ....
- ❑ Pour considérer toutes les exécutions possibles du système, on peut par exemple les ranger par ordre de longueur :
  - ❑ 11, 12
  - ❑ 111, 112, 121, 122, 123
  - ❑ ...



# Arbre d'exécution

- Toutes les exécutions du système peuvent être représentées par un arbre d'exécution (arbre infini en largeur et en profondeur).





# Propriétés élémentaires

- Pour vérifier des propriétés du système (modèle), on associe à chaque état de l'automate des propriétés élémentaires (satisfaites à cet état) modélisées par des propositions atomiques.
- Exemples :
  - « la porte est ouverte » est vraie dans l'état 4 et fausse dans les autres états (1,2,3).



# Propriétés vérifiables

- On peut ainsi vérifier des propriétés intéressantes telles que :
  - « *Si la porte s'ouvre (une exécution arrive dans l'état 4) alors les trois dernières lettres tapées sont dans l'ordre A,B,A* ».
  - « *Toute suite de lettres tapées finissant par ABA ouvre la porte (définit une exécution menant à l'état 4)* ».





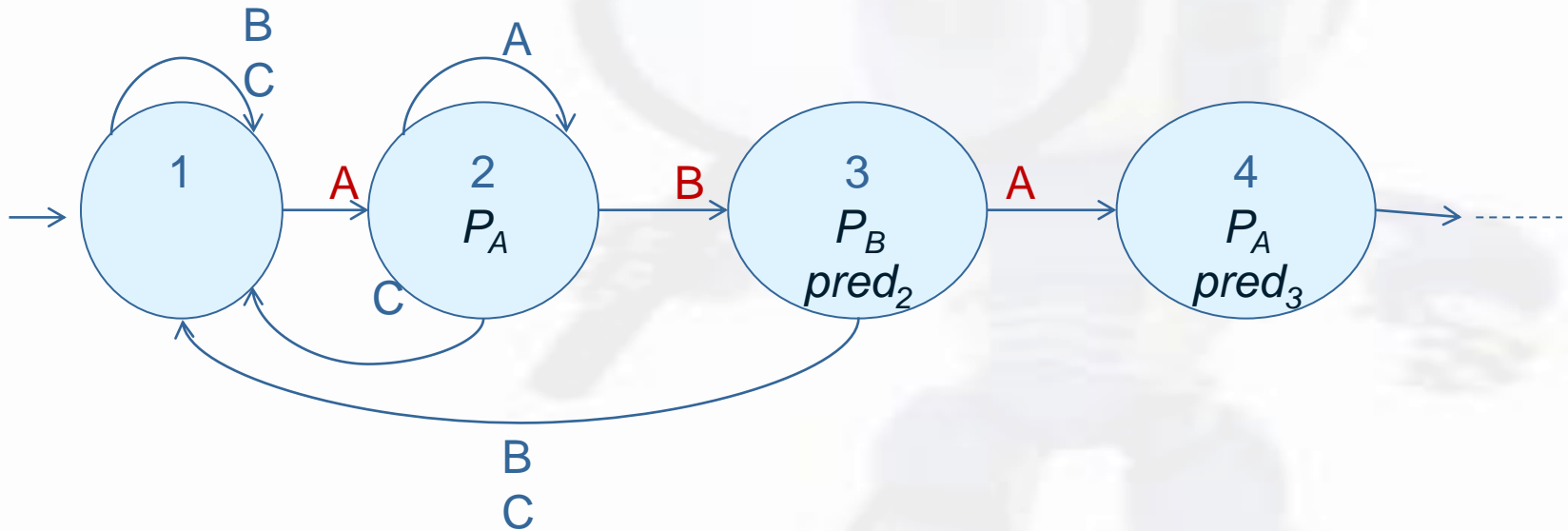
# Propositions atomiques pour le digicode

- $P_A$  : on vient nécessairement de taper un A (vraie dans les états 2 et 4).
- $P_B$  : on vient nécessairement de taper un B (vraie dans l'état 3 uniquement).
- $P_C$  : on vient nécessairement de taper un C (n'est vraie dans aucun état).
- $pred_2$  : l'état précédent dans une exécution est nécessairement un 2 (vraie dans l'état 3).
- $pred_3$  : l'état précédent dans une exécution est nécessairement un 3 (vraie dans l'état 4)



Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Propositions atomiques pour le digicode





## Exercice

□ Vérifier la propriété :

*« Si la porte s'ouvre (une exécution arrive dans l'état 4) alors les trois dernières lettres tapées sont dans l'ordre A,B,A ».*

□ C'est une propriété qui peut bien être vérifiée automatiquement par un model checker.



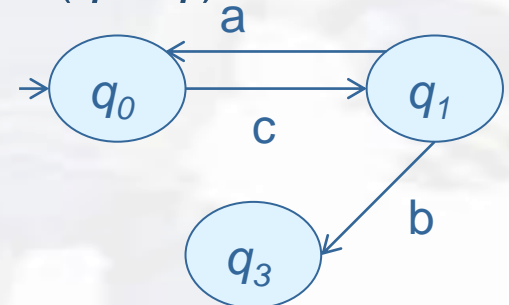
# CONCEPTS GÉNÉRAUX



# Systeme de transition étiqueté

□  $A=(Q,q_0,E,T)$  :

- $Q$  est un ensemble fini d'états.
- $E$  est un ensemble fini d'étiquettes de transition.
- $T \subseteq Q \times \{E \cup \{\varepsilon\}\} \times Q$  est l'ensemble de transitions, avec  $\varepsilon$  est une action interne non observable. Sachez que  $(q,e,q')$  est souvent notée  $q \xrightarrow{e} q'$ .
- $q_0$  est l'état initial.



□ Exemple.

□ Mais comment spécifier une propriété dans un état : dans l'état initial d'un robot aspirateur, le bac est vide.



# Systeme de Kripke

□  $A=(Q, q_0, E, T, Prop, I) :$

- $Q$  est un ensemble fini d'états.
- $E$  est un ensemble fini d'étiquettes de transition.
- $T \subseteq Q \times E \times Q$  est l'ensemble de transitions.
- $q_0$  est l'état initial.
- $Prop$  est un ensemble de propositions atomiques  $\{P_1, P_2, \dots, P_n\}$ .
- $I : Q \rightarrow 2^{Prop}$  application d'étiquetage des états par des propositions.(associe à tout état l'ensemble fini des propriétés élémentaires vérifiées dans l'état).



# Chemin

- Un **chemin**  $\pi$  de l'automate  $A$  est une suite  $\sigma$ , finie ou infinie, de transitions  $(q_i, e_i, q'_i)$  de  $A$  qui s'enchainent ( $q'_i = q_{i+1}$ ), et on note

$$q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} q_3 \xrightarrow{e_3} q_2 \dots$$

- Exemple :  $3 \xrightarrow{B} 1 \xrightarrow{A} 2 \xrightarrow{A} .2$
- La **longueur d'un chemin**  $\sigma$  (notée  $|\sigma|$ ) est le nombre de transitions qu'il contient ( $\omega$  s'il est infini).
- Le  $i^{\text{ème}}$  état de  $\sigma$  ( $\sigma(i)$ ) est l'état  $q_i$  atteint après  $i$  transitions (défini si  $i < |\sigma|$ ).



# Exécution

- Une **exécution** est une suite d'états décrivant une évolution du système.
- Une **exécution partielle** de A est un chemin partant de l'état initial  $q_0$  :  $1 \xrightarrow{B} 2 \xrightarrow{A} 2 \xrightarrow{B} 3$
- Une **exécution complète** est une exécution qu'on ne peut pas prolonger (infinie ou se termine dans un état sans successeur).





# Atteingabilité

- Un état est dit **atteignable** (accessible) s'il apparaît au moins une fois dans l'arbre d'exécution de l'automate.
- Le passage par un état peut être utilisé pour démontrer des propriétés du système :
  - Dans un système de commande d'une porte la vérification de l'atteingabilité de l'état *porte\_fermée* dans l'automate correspondant à démontrer la propriété « *la porte peut être fermée* ».



# **AUTOMATES AVEC VARIABLES**



# Automates avec variables d'états

- ❑ Introduire des données (variables) de contrôle.
- ❑ Souvent nécessaire pour modéliser des systèmes réels.
- ❑ Exemple : dans les protocoles de communication, il serait intéressant de compter le nombre de paquets reçus, émis, nombre d'erreurs, etc.
- ❑ Permettent aussi de réduire le nombre d'états de l'automate.



## Définition : Automate avec variables

- $A = (Q, q_0, E, T, Prop, Var, I) :$ 
  - $Q$  est un ensemble fini d'états.
  - $E$  est un ensemble fini d'étiquettes de transition.
  - $T \subseteq Q \times E \times Q$  est l'ensemble de transitions.
  - $q_0$  est l'état initial.
  - $Prop$  est un ensemble de propositions atomiques  $\{P_1, P_2, \dots, P_n\}$ .
  - $Var$  est un ensemble de variables  $\{v_1, \dots, v_n\}$ , chaque  $v_i$  peut prendre des valeurs un domaine  $Dv_i$ .
  - $I : Q \rightarrow 2^{Prop}$  application d'étiquetage des états.

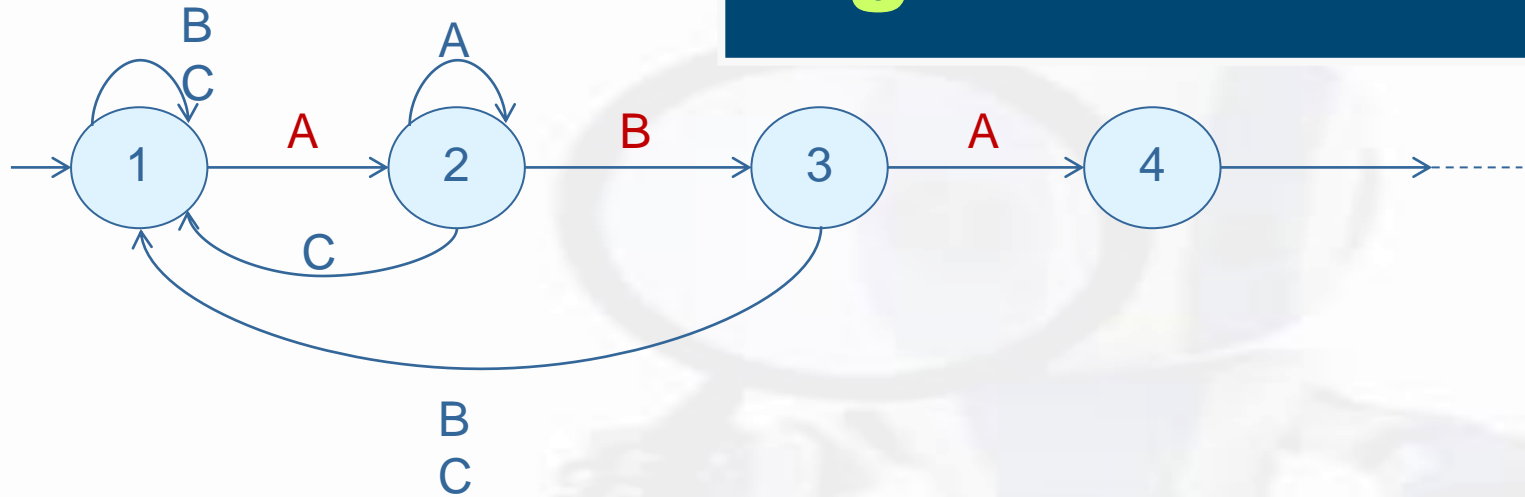


# Manipulation des variables par l'automate

- Le lien entre l'automate et les variables peut être :
  - D'**affectation** : une transition peut modifier la valeur d'une ou plusieurs variables.
  - De **garde** : une transition peut être gardée par une condition sur les variables. Ainsi, pour franchir la transition il faut que la condition sur les variables soit vérifiée.



## Exemple : Digicode

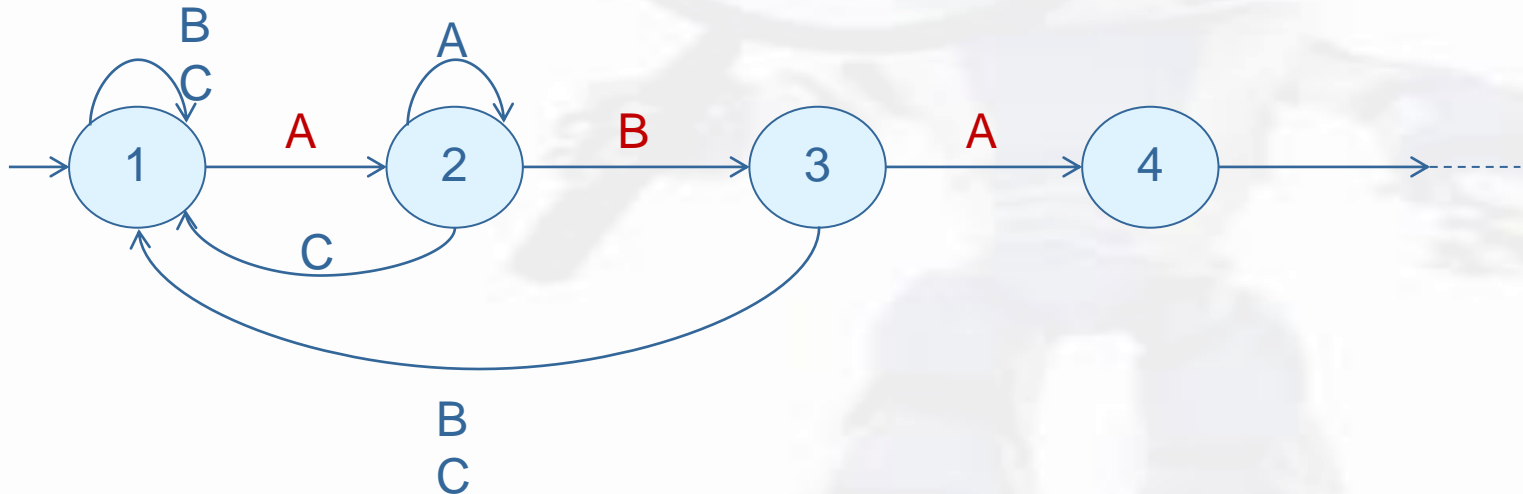


- ❑ Cette modélisation du digicode n'est pas réaliste, c'est ainsi que nous l'améliorons en ajoutant un compteur de nombre d'erreurs commises par l'utilisateur et n'autoriser que 3 erreurs.
- ❑ Nous ajoutons donc une variable entière *cpt*, initialisée à 0.



## Lien entre l'automate et *cpt*

- Affectation : les transitions correspondantes à une erreur incrémentent le compteur.
- Qu'elles sont ces transitions ?

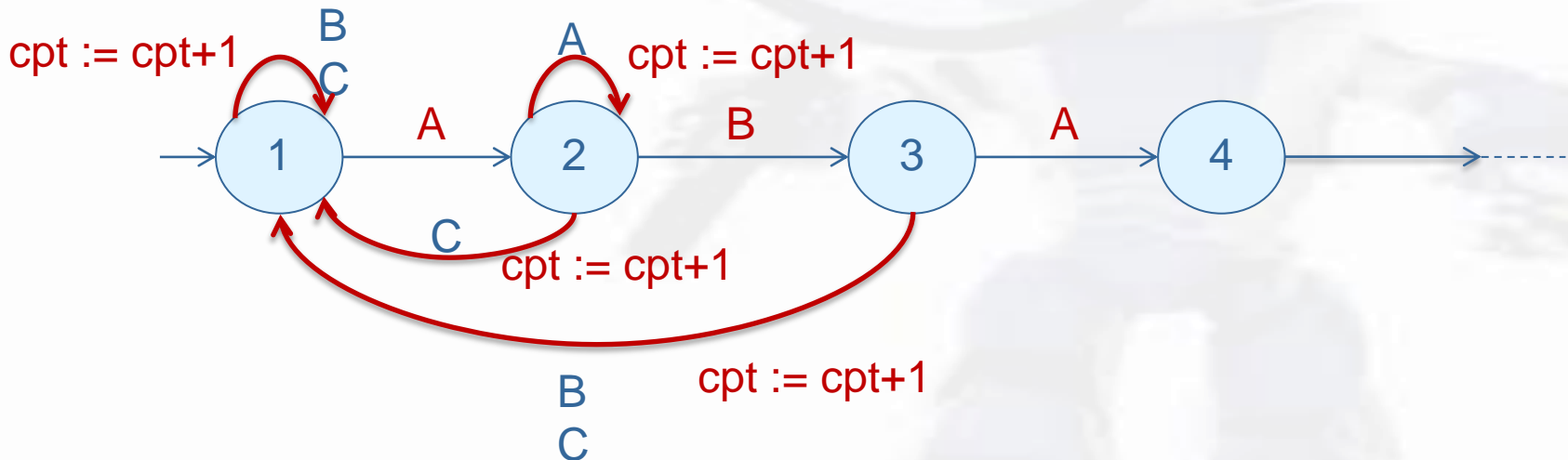




## Lien entre l'automate et *cpt*

□ **Affectation** : les transitions correspondantes à une erreur incrémentent le compteur.

□ Qu'elles sont ces transitions ?



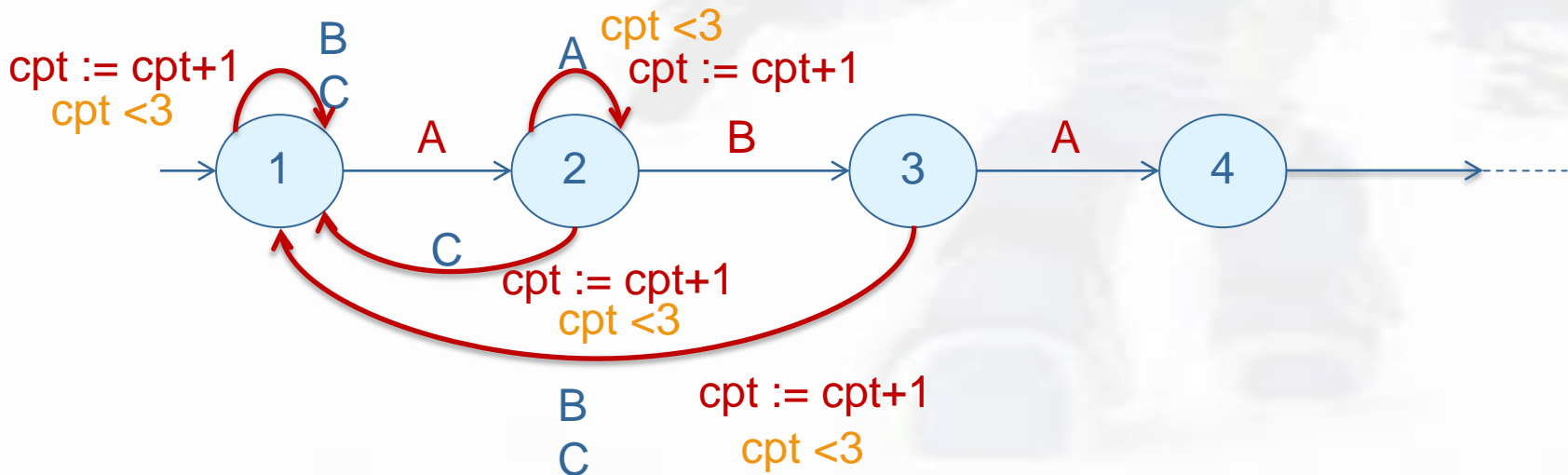
□ Au fait toutes les transitions sauf (1,A,2), (2,B,3) et (3,A,4) menant à l'ouverture de la porte.





## Lien entre l'automate et *cpt*

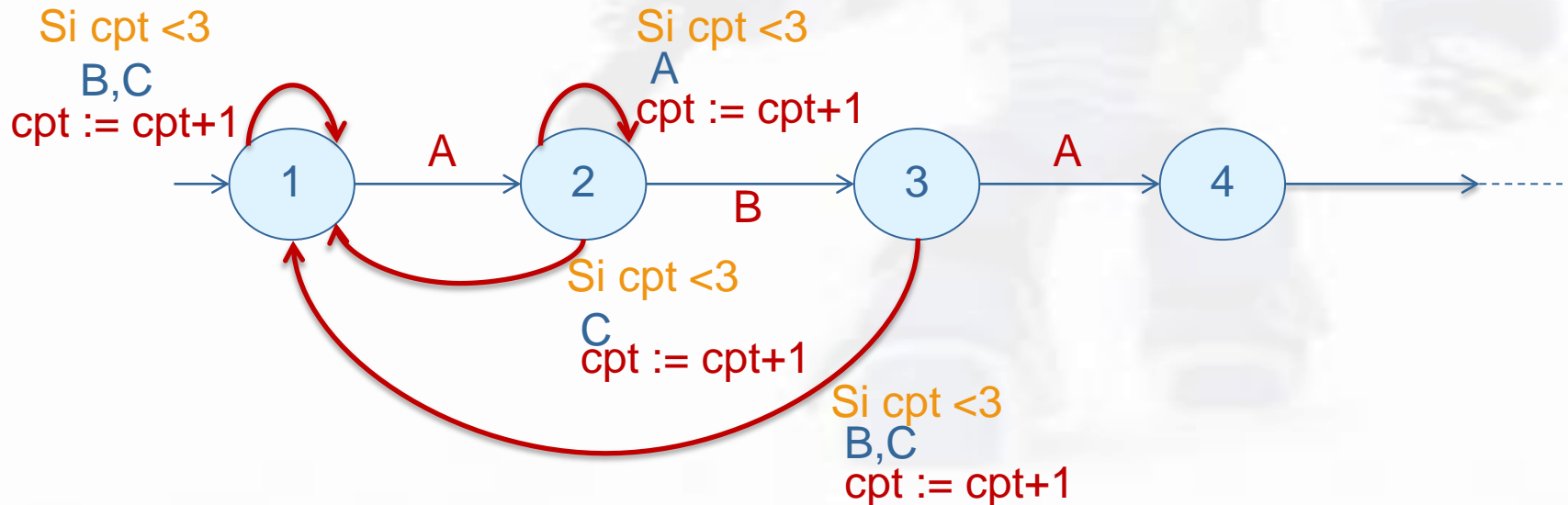
□ Garde : pour tolérer au maximum 3 erreurs, il faut marquer les transitions correspondant à une erreur par une condition de garde :  $cpt < 3$ .





# Conventions

- ❑ Les gardes sont les premières à spécifier et sont précédées par le mot clé si.
- ❑ Ils sont suivis des étiquettes des actions.
- ❑ Suivis des affectations.



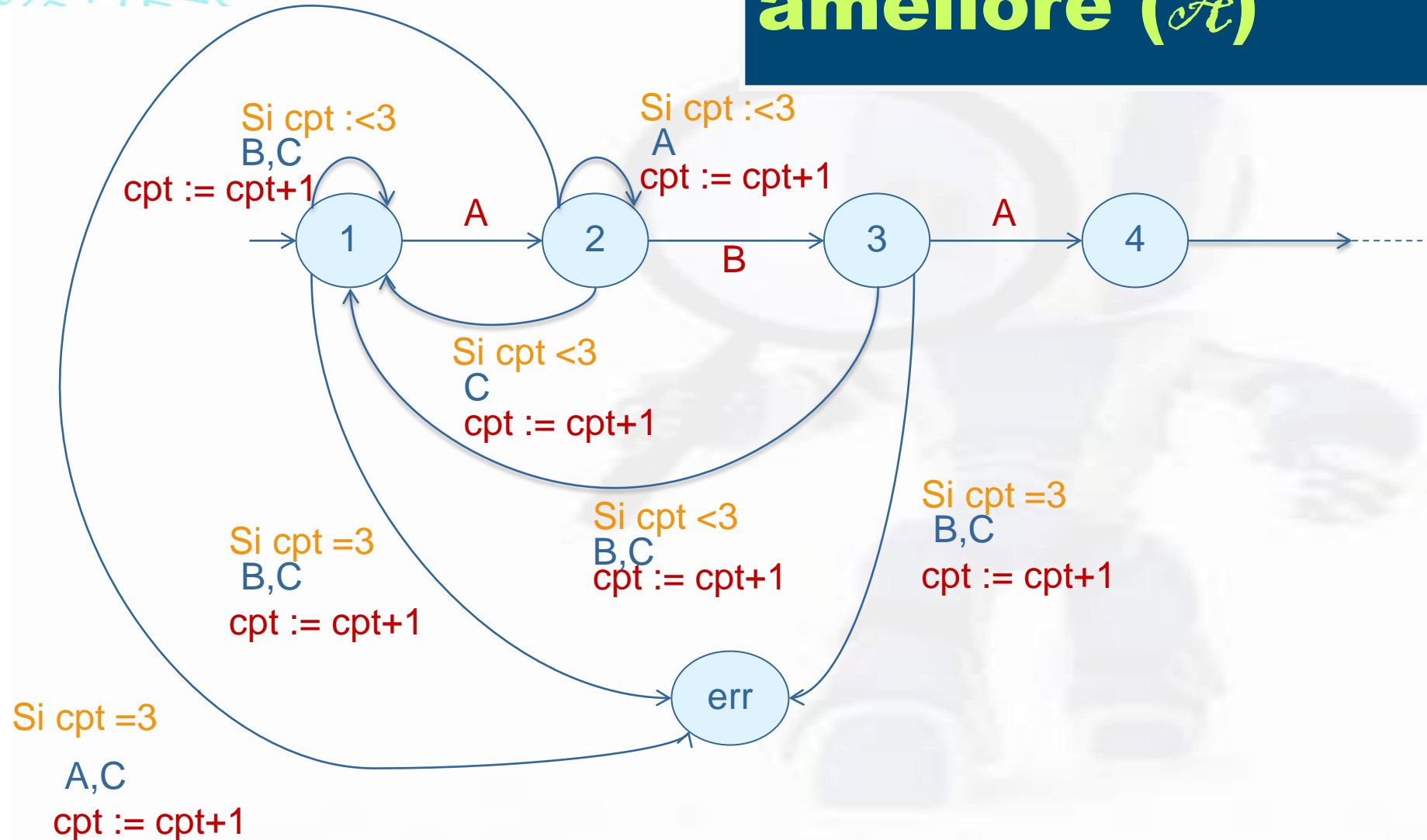


# Amélioration du Digicode

- ❑ Déclencher une alarme si 4 erreurs sont commises.
- ❑ Ajouter un état err et 3 transitions entrantes issues des trois états : 1,2,3 et gardées par *si cpt=3*.



# Digicode amélioré ( $\mathcal{A}$ )





# Dépliage de l'automate

- ❑ Pour appliquer des méthodes de vérification telles que le model checking, il est nécessaire de travailler sur des modèles dépliés.
- ❑ Uniquement les transitions possibles sont présentes.
- ❑ L'automate déplié est un graphe d'états appelé souvent *système de transitions*.



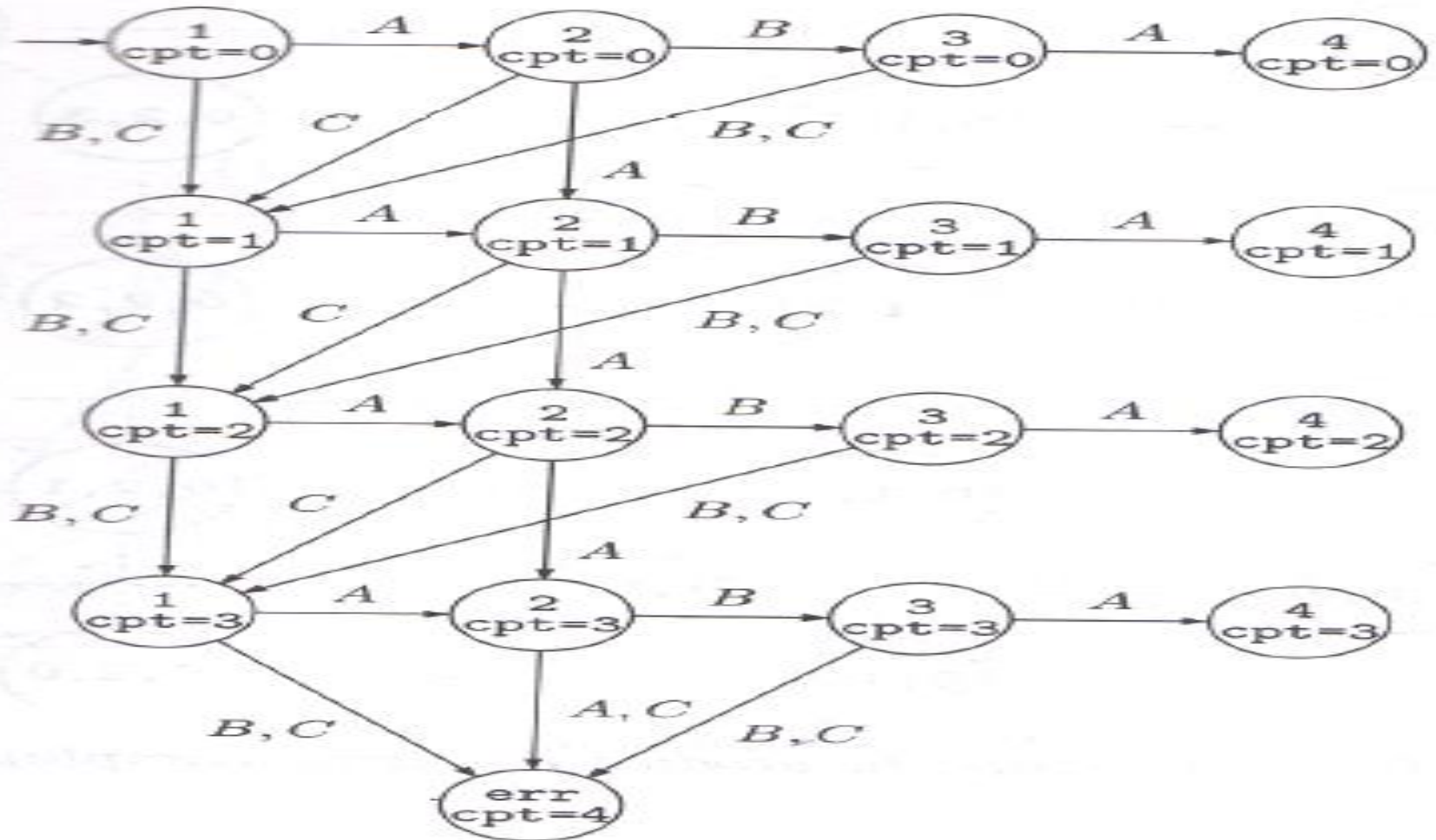
# Construction de l'automate déplié

- ❑ Construction des états : pour chaque état de  $\mathcal{A}$ , créer autant d'états (appelés globaux) qu'il y a de valeurs des variables utilisées.
- ❑ Construction des transitions :
  - ❑ déplier les transitions entre les occurrences d'état si la garde est vraie,
  - ❑ les transitions ne sont plus gardées car on connaît les valeurs des variables des états globaux,
  - ❑ supprimer les affectations des variables, car les états contiennent directement les valeurs de variables.
- ❑ Supprimer les états non atteignables.



Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Digicode avec comptage des erreurs





# **AUTOMATES COMMUNICANTS**





# Compositions de structures de Kripke

- ❑ Les modèles de Kripke complexes sont obtenus généralement par composition d'autres modèles : produits de systèmes.
- ❑ On parle de produit cartésien pour les systèmes indépendants.
- ❑ On parle de produit synchronisé pour les systèmes dépendants.



# Modèles Composés

- ❑ Permettent de modéliser des systèmes découpés en modules ou sous-systèmes.
- ❑ Ils sont adéquats pour abstraire les systèmes parallèles et/ou distribués.
- ❑ Consiste à modéliser chaque composante du système, ensuite de les faire coopérer.



# Composants sans interaction

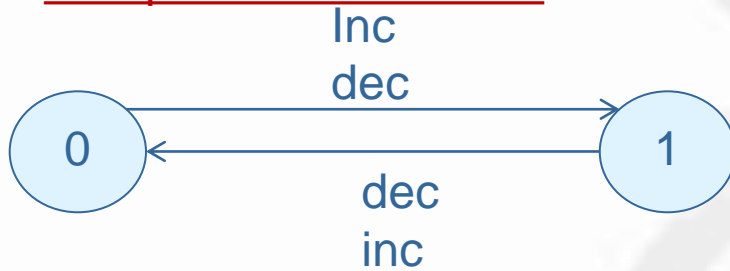
- Il suffit de représenter chaque composant par une structure de Kripke.
- Construire la structure qui soit le **produit cartésien** des structures modélisant les composants.
- Un état global n'est autre qu'un vecteur composé des différents états (locaux) des composants.



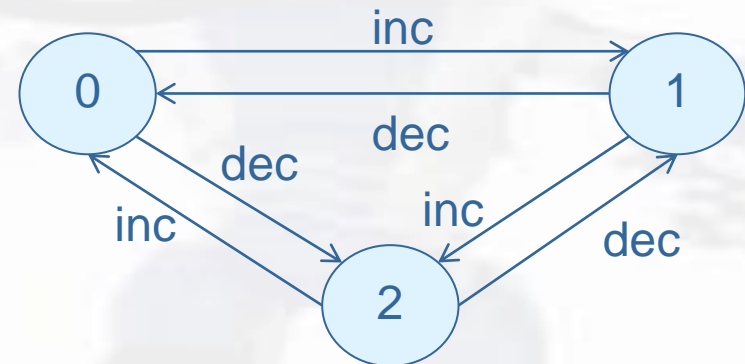
# Exemple : compteur

□ Un système composé de 3 compteurs :  
modulo 2, modulo 3 et modulo 4.

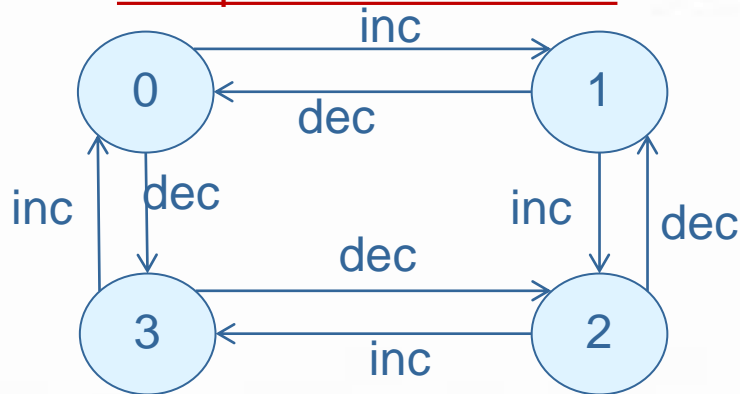
Compteur modulo 2 **M2**



Compteur modulo 3 **M3**



Compteur modulo 4 **M4**





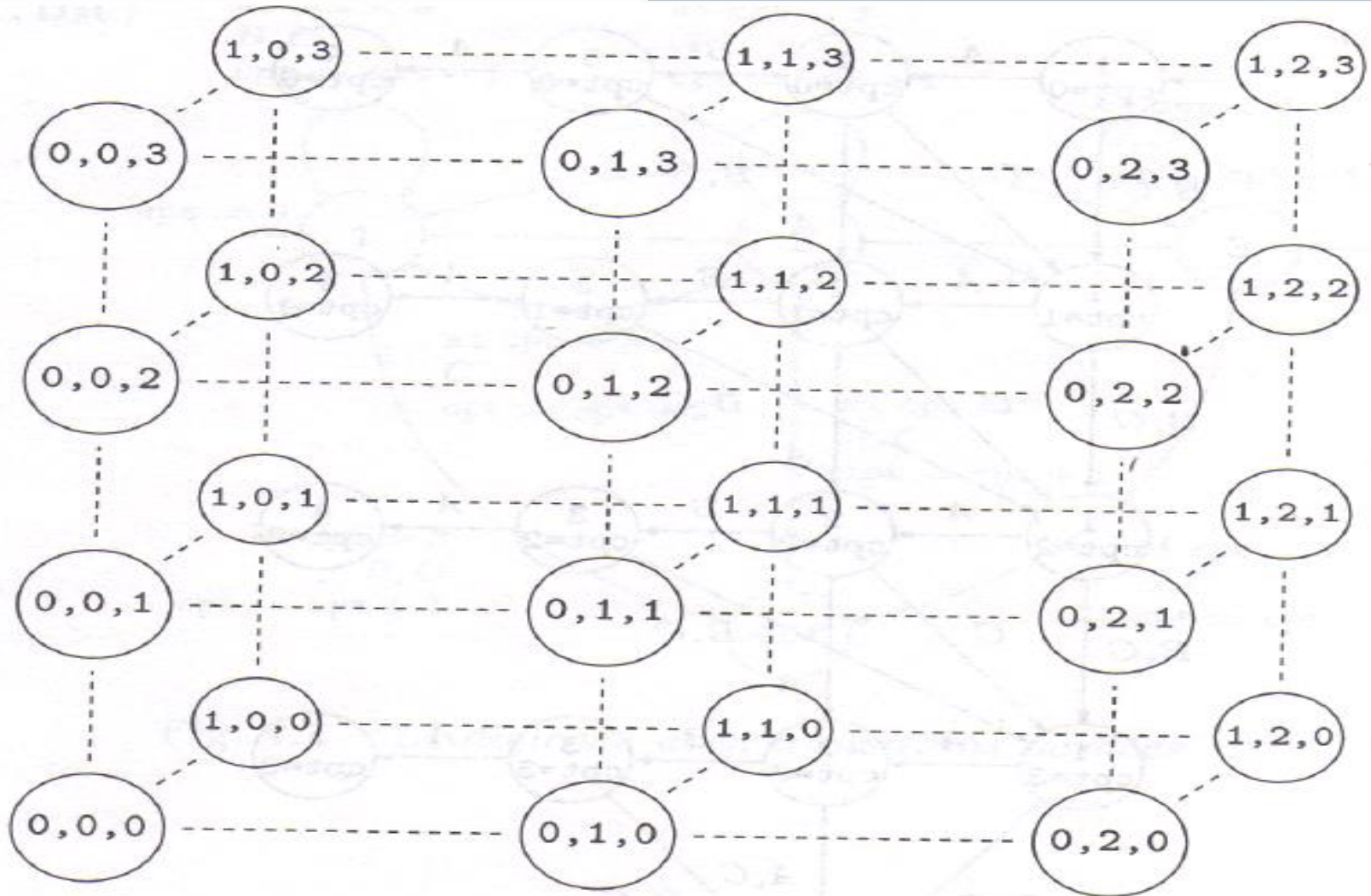
# Les états de l'automate composé

- L'automate  $M_{234}$  (produit cartésien des automates  $M_2$ ,  $M_3$  et  $M_4$ ) contient  $2 \times 3 \times 4 = 24$  états.
- Chaque état est étiqueté par un vecteur de trois éléments chacune pouvant être : 0, 1, 2 ou 3.



Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Etats du produit des 3 compteurs





# Transitions de l'automate composé

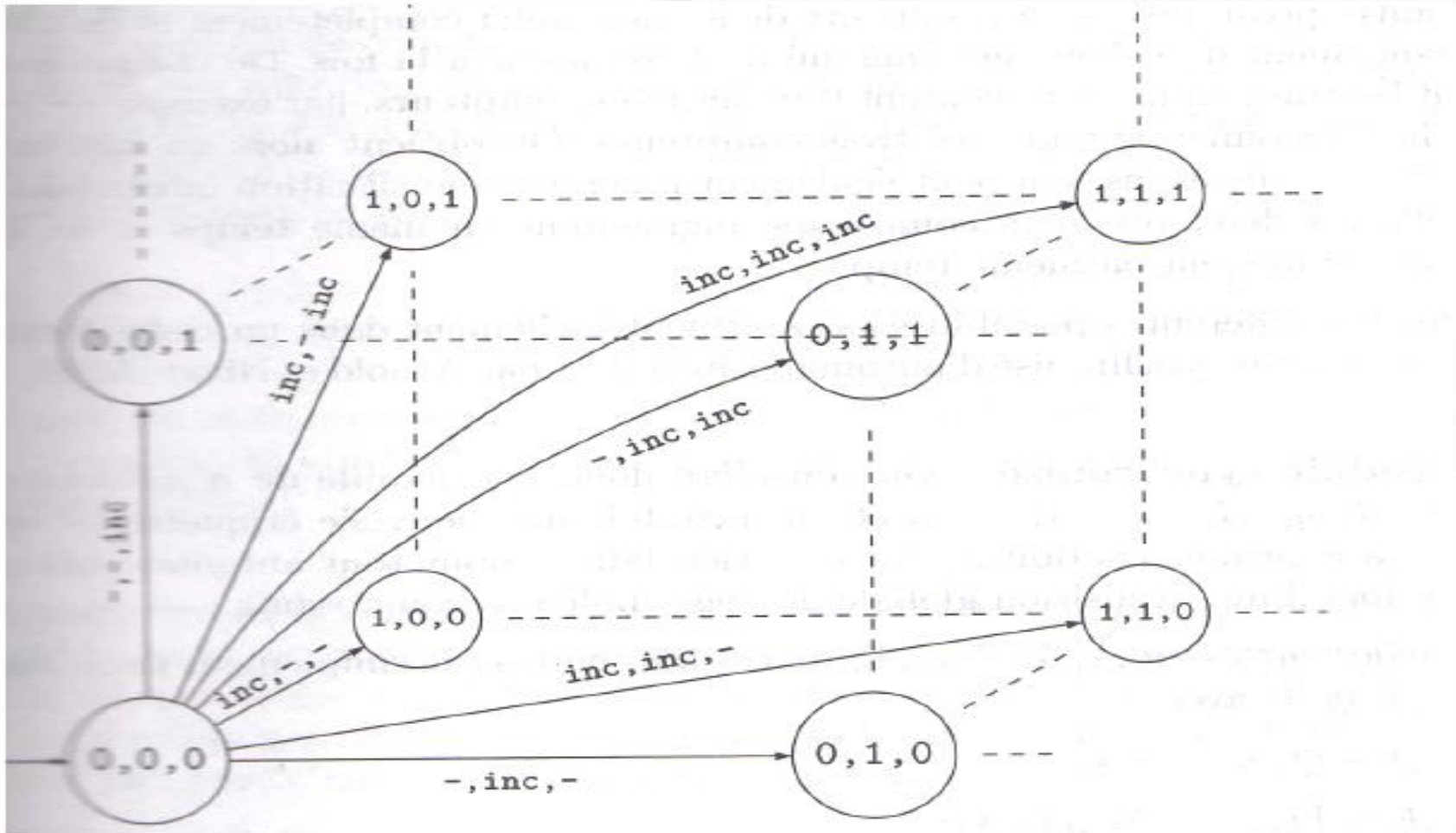
- Il n'y a pas de synchronisation, chaque transition permet l'incrémentement ou la décrémentation de un ou plusieurs compteurs individuels :  $3 \times 3 \times 3 = 27$  choix.
- Il est possible qu'un compteur reste stable, ceci sera modélisé par le symbole « \_ ».
- Notez que ce symbole n'existait pas dans les compteurs individuels. Pourquoi ?
- Il y a  $27 - 1 = 26$  (on élimine celle où aucun compteur n'évolue) transitions possibles dans chaque état  $\Rightarrow 24 \times 26 = 624$  transitions.





Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Quelques transitions de M234







# Produit synchronisé

- ❑ Effectué pour les systèmes dépendants.
- ❑ Nous désirons obtenir le produit de  $n$  automates :  $\mathcal{A}_i = \langle Q_i, E_i, T_i, q_{0,i}, l_i \rangle$ ,  $i=1, \dots, n$ .
- ❑ Nous avons besoin d'un symbole spécial « - » pour exprimer la stabilité d'un composant.
- ❑ Le produit cartésien  $\mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_n$  est l'automate  $\mathcal{A} = \langle Q, E, T, q_0, l \rangle$  et est noté  $\mathcal{A}_1 \parallel \mathcal{A}_2 \parallel \dots \parallel \mathcal{A}_n$ , calculons le.



# Produit synchronisé

- Le produit cartésien  $\mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_n$  est l'automate  $\mathcal{A} = \langle Q, E, T, q_0, l \rangle$  avec :
  - $Q = Q_1 \times Q_2 \times \dots \times Q_n$
  - $E = \prod_{1 \leq i \leq n} (E_i \cup \{-\})$ ,
  - $T = \{((q_1, q_2, \dots, q_n), (e_1, e_2, \dots, e_n), (q'_1, q'_2, \dots, q'_n)) \mid \forall i : e_i = '-' \text{ et } q'_i = q_i, \text{ ou } e_i \neq '-' \text{ et } (q_i, e_i, q'_i) \in T_i\}$
  - $q_0 = (q_{01}, q_{02}, \dots, q_{0n})$ ,
  - $l((q_1, q_2, \dots, q_n)) = \bigcup_{1 \leq i \leq n} l_i(q_i)$ .
- Pour synchroniser les composants, nous restreignons les transitions autorisées :
  - $\text{Sync} \subseteq \prod_{1 \leq i \leq n} (E_i \cup \{-\})$ .



# Produit synchronisé

- ❑ Les composants évoluent selon une fonction de synchronisation.
- ❑ Exemples :
  - ❑ Exiger que les 3 compteurs évoluent en même temps et au même rythme : étiquettes possibles « inc,inc,inc » et « dec,dec,dec ».
  - ❑ Un seul compteur peut évoluer à la fois.
  - ❑ Les compteurs évoluent en couple



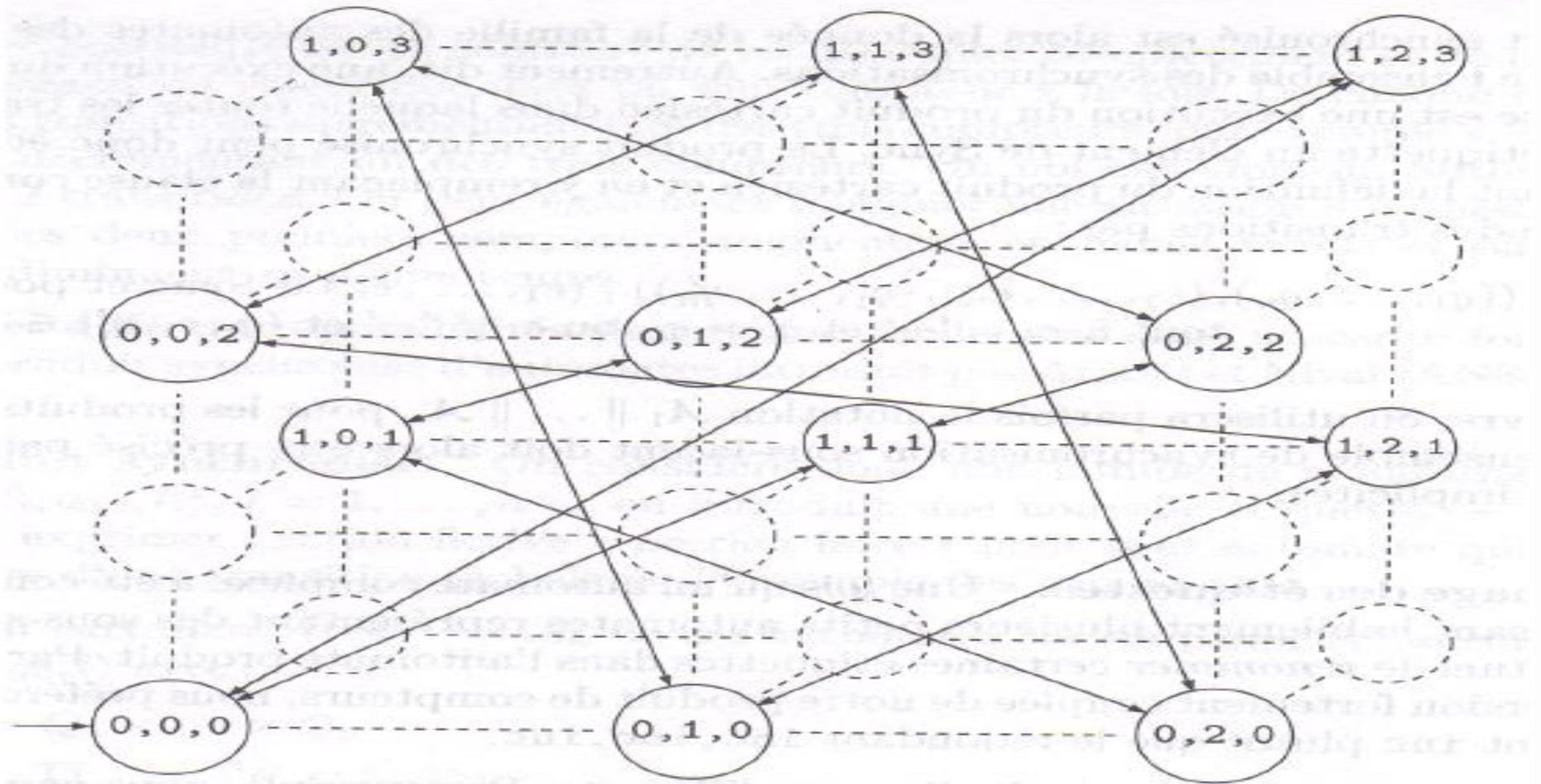
# Produit synchronisé

- Pour avoir un automate composé qui couple fortement les 3 compteurs ( $M234c$ ) Sync sera définie par :
  - $\text{Sync} = \{(\text{inc}, \text{inc}, \text{inc}), (\text{dec}, \text{dec}, \text{dec})\}$ .
- La fonction  $T$  doit donc être redéfinie :
  - $T = \{((q_1, q_2, \dots, q_n), (e_1, e_2, \dots, e_n), (q'_1, q'_2, \dots, q'_n)) \mid (e_1, e_2, \dots, e_n) \in \text{Sync} \text{ et } \forall i : e_i = '-' \text{ et } q'_i = q_i, \text{ ou } e_i \neq '-' \text{ et } (q_i, e_i, q'_i) \in T_i\}$



Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

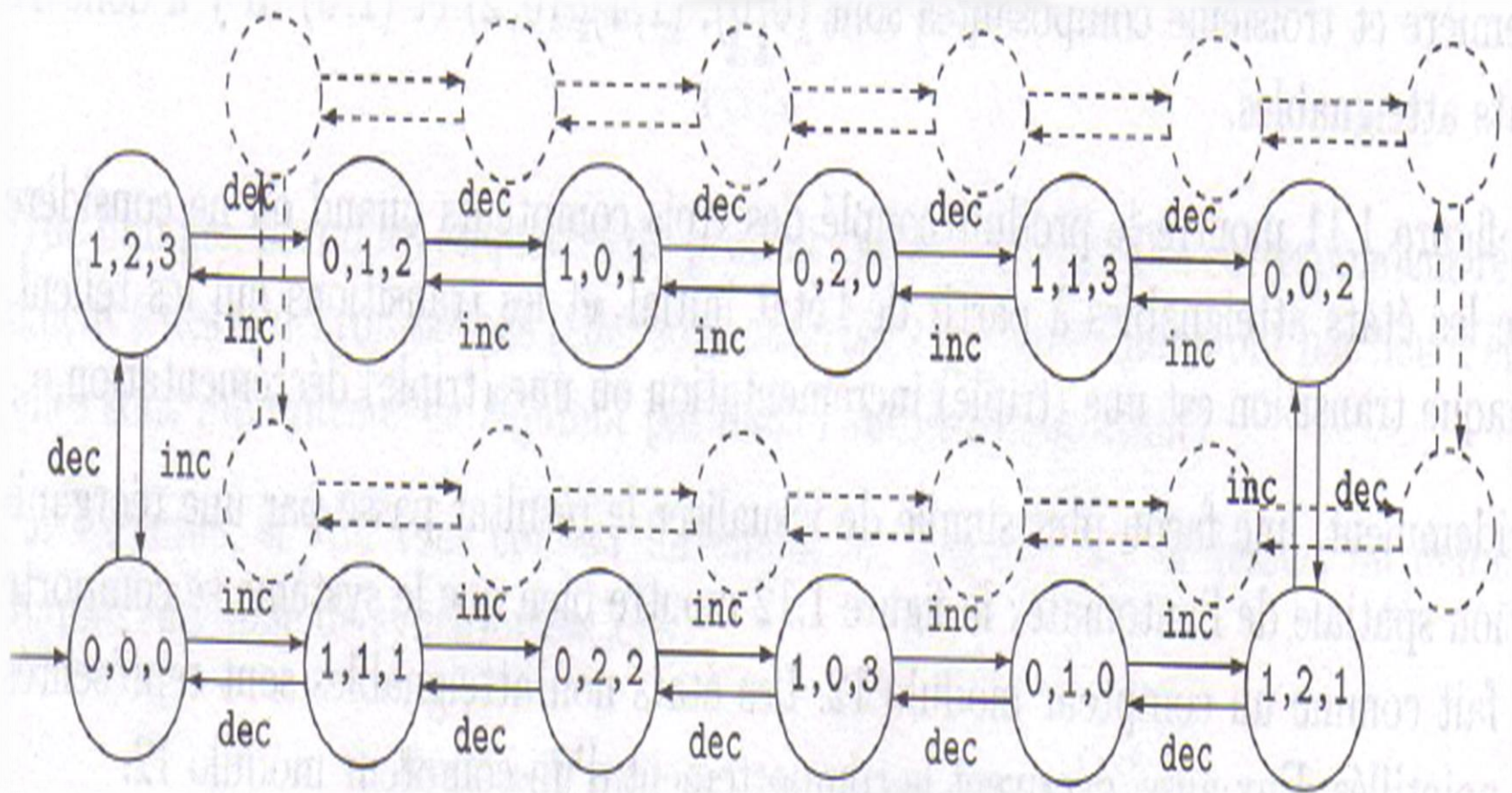
# ***M234c avec les états atteignables***







# M234c Nettoyé





# Problème d'explosion

- ❑ Trouver l'ensemble des états atteignables dans un produit synchronisé est un problème difficile.
- ❑ Heureusement que cette procédure est généralement automatisable.
- ❑ Elle consiste à explorer tous les états du produit synchronisé.
- ❑ Le produit synchronisé des automates  $A_1, A_2, \dots, A_p$  ayant respectivement les états  $n_1, n_2, \dots, n_p$ , aura  $n_1 \times n_2 \times \dots \times n_p$  états  $\Rightarrow$  nombre exponentiel en  $p$  : problème de l'explosion du nombre d'états.



# Problème d'explosion

- ❑ L'utilisation de variables peut aussi causer un problème d'explosion d'états.
- ❑ Lors du dépliage, les valeurs prises par une variable peuvent être infinies.
- ❑ Une solution serait l'utilisation des *réseaux de Petri* (adaptés pour les systèmes parallèles).





## Variantes du produit synchronisé

- ❑ Synchronisation par envoie de message.
- ❑ Synchronisation par variables partagées.



# Synchronisation par message

- Il s'agit de la synchronisation par envoie/réception de messages.
- Il existe deux sortes d'étiquettes pour les transitions :
  - Emission d'un message : *!m*.
  - Réception d'un message : *?m*.
- Dans le produit synchronisé par message, seulement les transitions où l'émission est accompagnée de la réception correspondantes sont autorisées.



Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

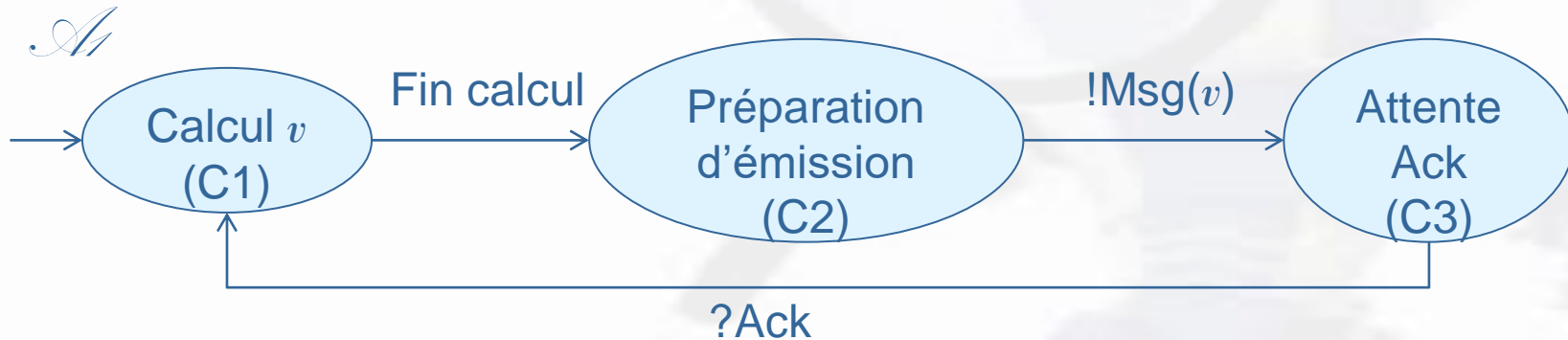
## Exemple1 : envoie et réception de messages

- Partage d'une donnée  $v$  entre deux systèmes  $\mathcal{A}_1$  et  $\mathcal{A}_2$ : calcul et traitement.



## Exemple1 : envoie et réception de messages

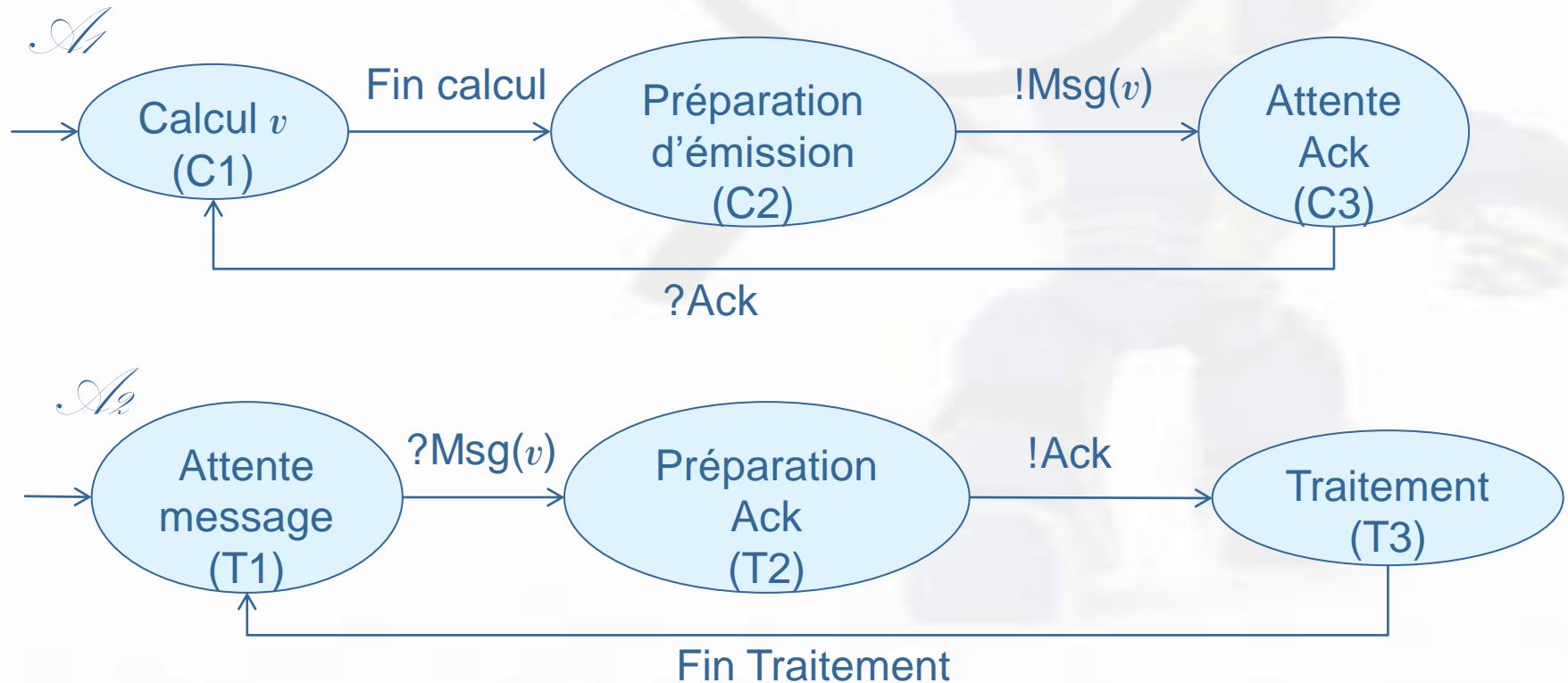
□ Partage d'une donnée  $v$  entre deux systèmes  $\mathcal{A}_1$  et  $\mathcal{A}_2$ : calcul et traitement.





# Exemple1 : envoie et réception de messages

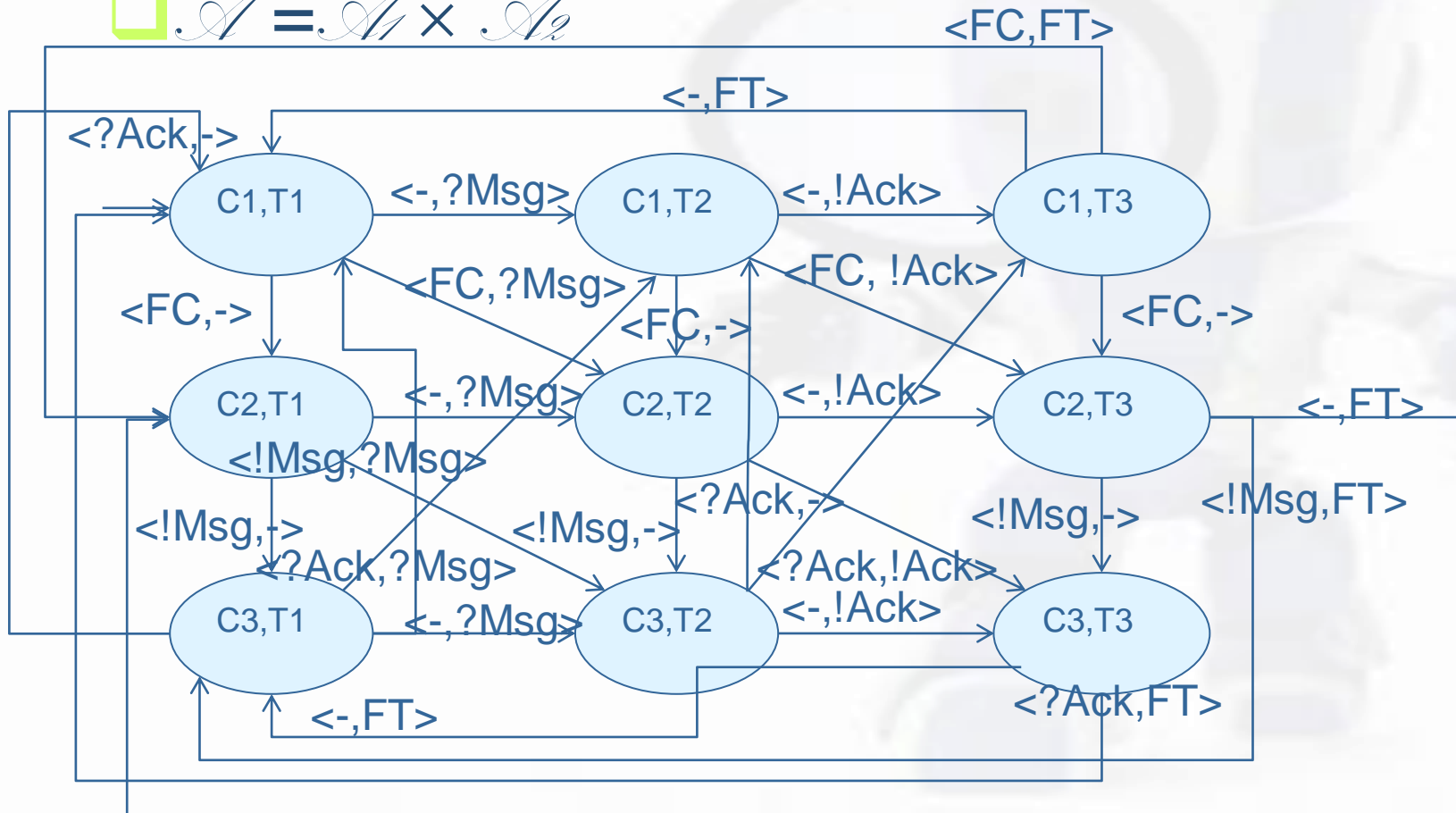
□ Partage d'une donnée  $v$  entre deux systèmes  $\mathcal{A}_1$  et  $\mathcal{A}_2$ : calcul et traitement.





# Synchronisation de $\mathcal{A}_1$ et $\mathcal{A}_2$

$\square .\mathcal{A} = .\mathcal{A}_1 \times .\mathcal{A}_2$





# Contrainte de synchronisation

- La contrainte de synchronisation entre ces deux automates se réduit à la simultanéité des émissions/réceptions de messages :
  - Transitions de la forme  $(!t, ?t) : !t \in \mathcal{A}_1, ?t \in \mathcal{A}_2$ , ou inversement.
  - Transitions ne contenant aucune émission et/ou réception de messages.
- $\text{Sync} = \{ \langle !\text{Msg}, ?\text{Msg} \rangle, \langle !\text{Ack}, ?\text{Ack} \rangle, \langle \text{FC}, \text{FT} \rangle, \langle -, \text{FT} \rangle, \langle \text{FC}, - \rangle \}$

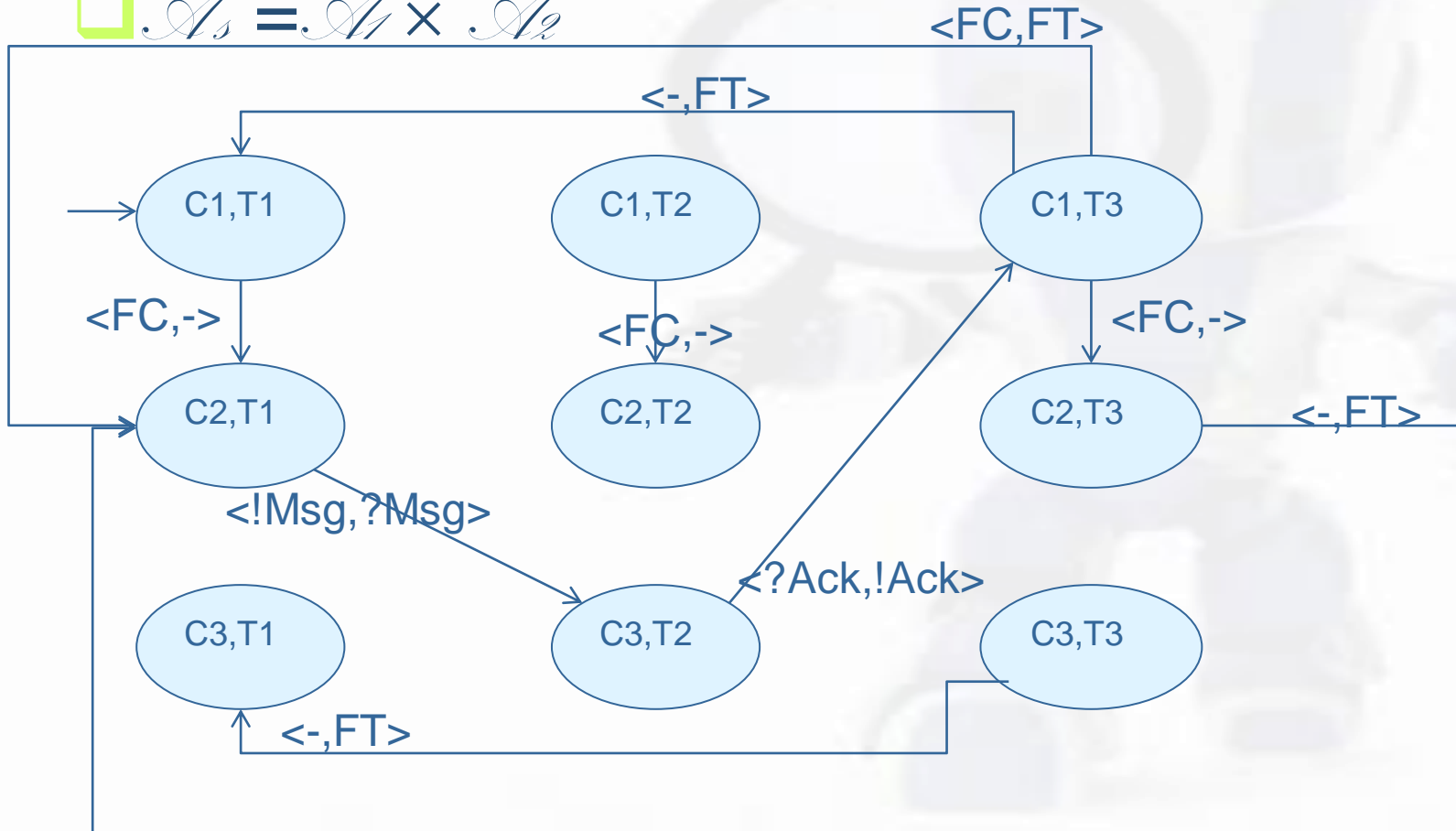


Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

$\mathcal{A}_1 \times \mathcal{A}_2$

# Synchronisé

□  $\mathcal{A}_s = \mathcal{A}_1 \times \mathcal{A}_2$







## Exemple 2 : Ascenseur

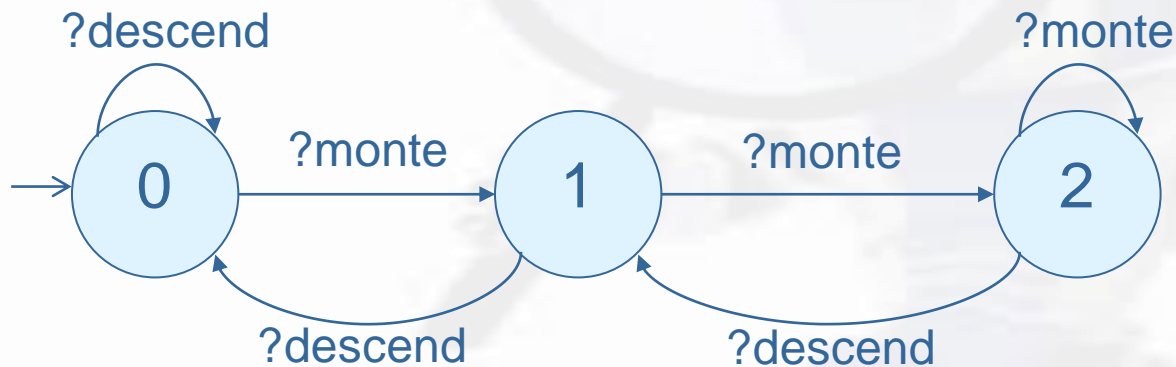
- ❑ Ascenseur à trois étages (rez-de-chaussée, 1<sup>er</sup>, 2<sup>ème</sup>)
- ❑ Cinq composants :
  - ❑ Une Cabine : qui monte et descend,
  - ❑ Trois portes (une par étage) : qui s'ouvrent et se ferment,
  - ❑ Un contrôleur : contrôle les 3 portes et la cabine par émission d'ordre.
- ❑ Nous négligeons le monde extérieur au système (appels à l'ascenseur)



Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Modélisation des composants

## □ Cabine

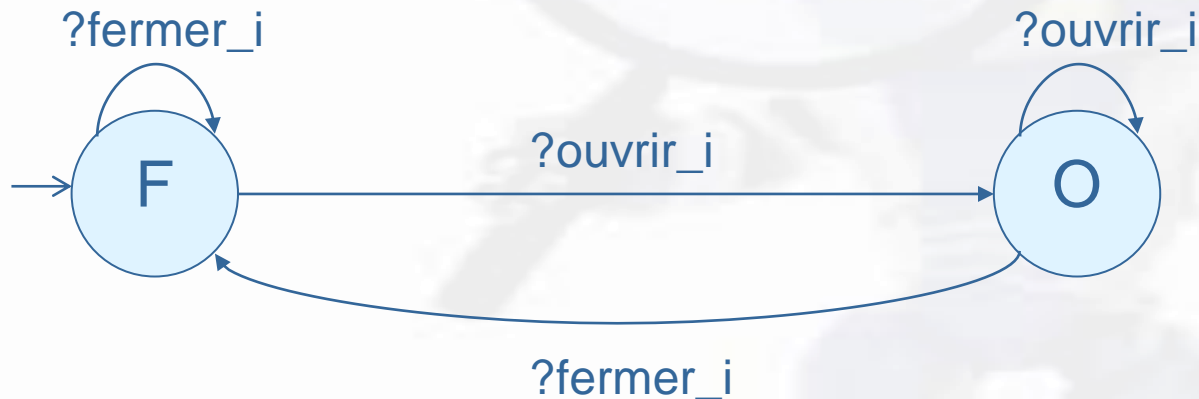




Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Modélisation des composants

## □ Porte numéro i

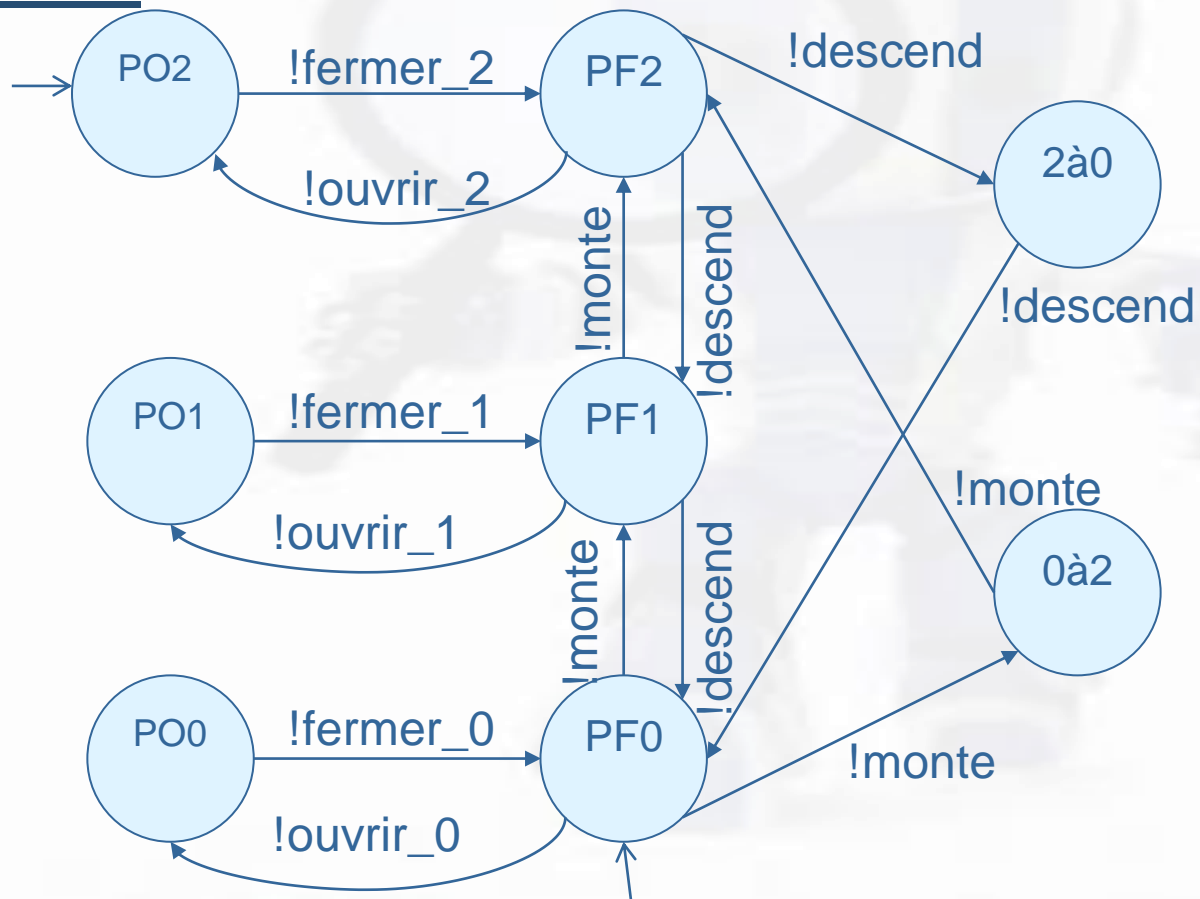




Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Modélisation des composants

## □ Contrôleur





# Synchronisation de l'ascenseur

- Le produit des 5 composants contient :
  - Des états à 5 éléments  $\langle \text{porte0}, \text{porte1}, \text{porte2}, \text{cabine}, \text{contrôleur} \rangle$
  - Des transitions à 5 éléments, un pour chaque système (porte0, porte1, porte2, cabine, contrôleur).
- La contrainte de synchronisation Sync garantit la simultanéité des émissions/réception de messages :
  - $\text{Sync} = \{ \langle ?\text{ouvrir}_0, -, -, -, !\text{ouvrir}_0 \rangle, \langle ?\text{fermer}_0, -, -, -, !\text{fermer}_0 \rangle, \langle -, ?\text{ouvrir}_1, -, -, !\text{ouvrir}_1 \rangle, \langle -, ?\text{fermer}_1, -, -, !\text{fermer}_1 \rangle, \langle -, -, ?\text{ouvrir}_2, -, !\text{ouvrir}_2 \rangle, \langle -, -, ?\text{fermer}_2, -, !\text{fermer}_2 \rangle, \langle -, -, -, ?\text{descend}, !\text{descend} \rangle, \langle -, -, -, ?\text{monte}, !\text{monte} \rangle \}$



# Spécification de propriétés pour l'ascenseur

- Avant de livrer ce produit, il est utile de s'assurer qu'il ne cause pas d'ennuis à ses utilisateurs.
- Il est important de vérifier :
  - (P1) « *la porte d'un étage ne peut s'ouvrir si la cabine n'est pas à cet étage* ».
  - (P2) « *la cabine ne peut se déplacer si une des portes est ouverte* ».



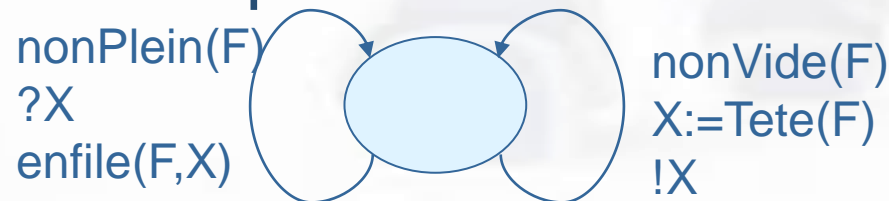
## Vérification de propriétés de l'ascenseur

- (P1) signifie pour la porte 0 : tout état ayant « 0 » comme premier composant (porte 0 ouverte), a nécessairement « 0 » comme quatrième composant (la cabine est dans l'étage 0).
- Il suffit donc de vérifier qu'il n'existe pas d'états atteignables ayant sous la forme :
  - $(0, -, -, 1, -)$  ou
  - $(0, -, -, 2, -)$
- Cette propriété peut être vérifiée aussi pour les autres portes.



# Messages asynchrones

- Lorsque les messages ne sont pas reçus instantanément : *communication asynchrone*.
- Les messages émis et non encore reçus sont stockés dans des canaux de communication (*buffers*) et gérés en FIFO.
- Le buffer est synchronisé avec l'émetteur et le récepteur, mais ces derniers sont asynchrones.
- Peut être représenté par un automate synchrone







## Conclusion sur la synchronisation par message

- ❑ Communication synchrones est utilisée pour modéliser les système de contrôle/commande.
- ❑ Communication asynchrones est souvent utilisée pour modéliser des protocoles de communication.



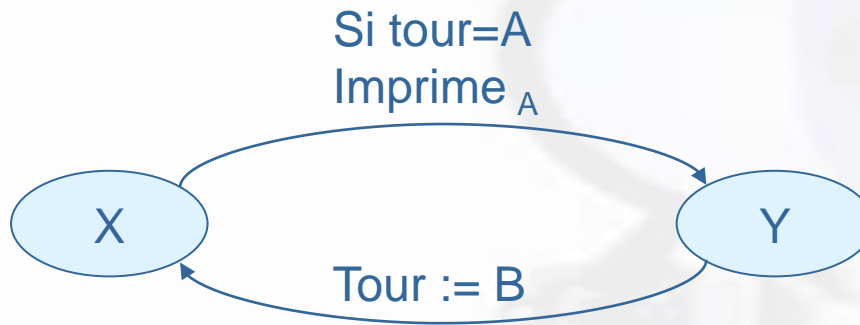
# Synchronisation par variable partagée

- ❑ Il s'agit de faire communiquer des automates en les faisant partager des variables.
- ❑ Utiliser les variables locales à chaque automate comme des variables globales partagées par l'automate synchrone.
- ❑ Exemple : imprimante partagée par deux personnes A et B on utilisera une variable partagée *tour* indiquant lequel des deux a le droit d'imprimer.

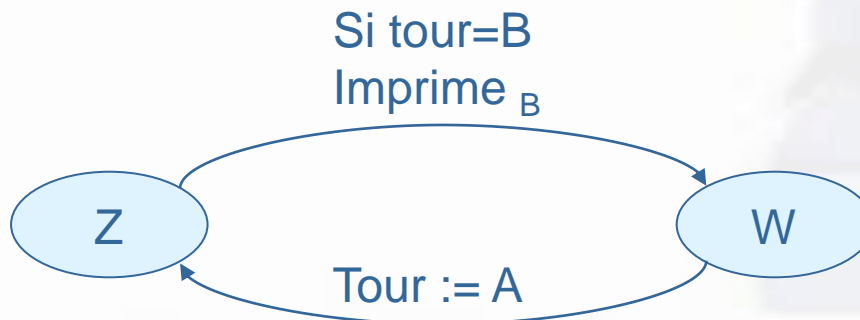


# Exemple : imprimante partagée

## □ Utilisateur A



## □ Utilisateur B

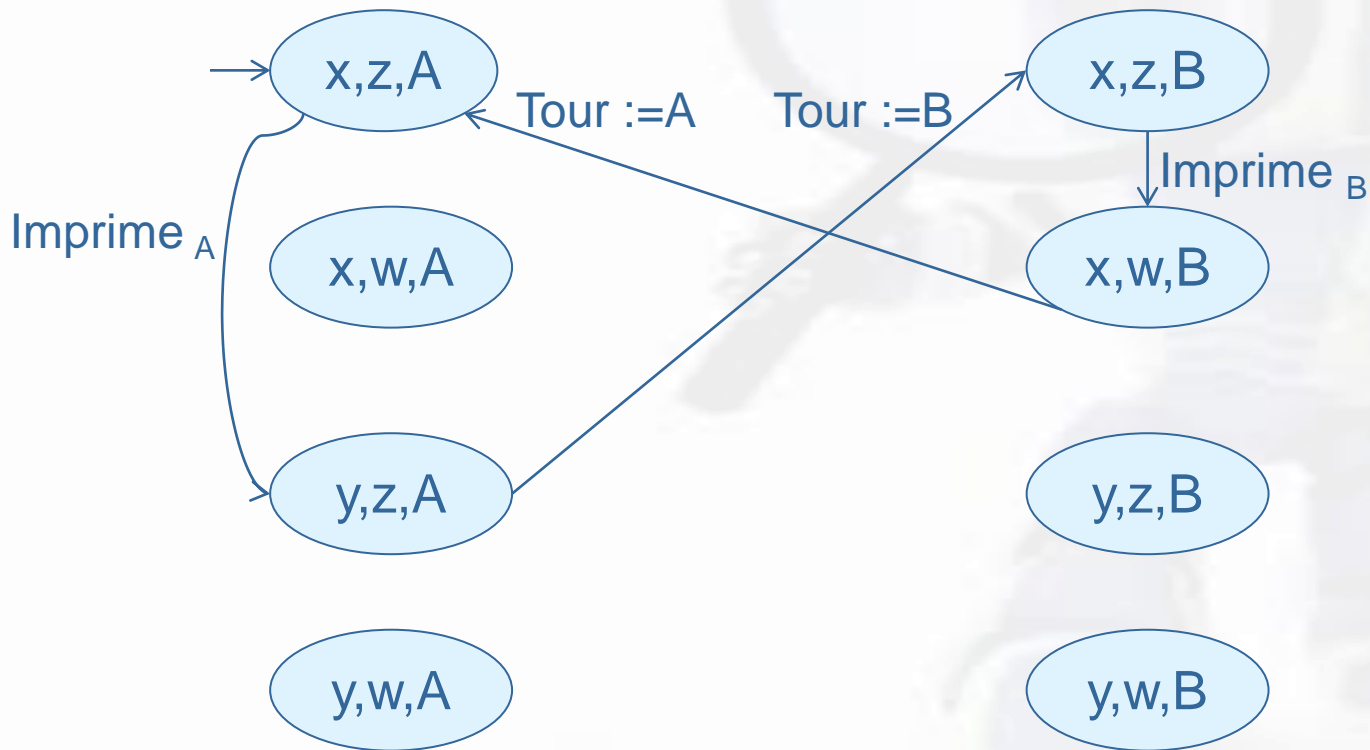


tour est une variable partagée  
par les deux automates



# Imprimante synchronisée

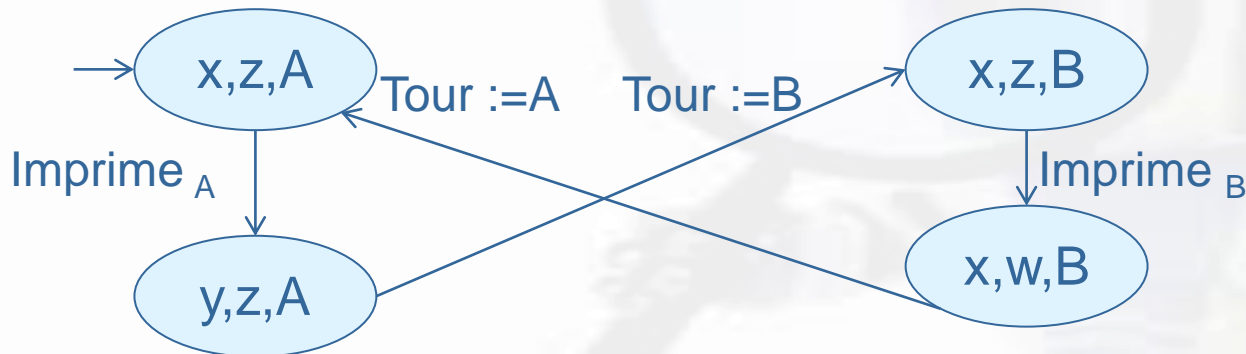
□ L'état initial dépend de l'initialisation de *tour* (ici A).





# Imprimante synchronisée

- L'automate synchronisé avec états atteignables.



- Il est évident que ce protocole d'impression simpliste fonctionne bien puisqu'aucun état de la forme  $(y,w,-)$  n'est atteignable.
- Sauf qu'il ne permet pas des impressions simultanées pour un seul utilisateur.



# Imprimante partagée par Peterson

- Deux processus  $P_A$  et  $P_B$ , trois variables partagées :
  - $d_A$  : l'utilisateur A met à vrai s'il veut imprimer. Initialement elle est fausse.
  - $d_B$  : l'utilisateur B met à vrai s'il veut imprimer. Initialement elle est fausse.
  - *tour* : départage le conflit.



## Code de Peterson pour $P_x$

□  $\bar{X} = A$  si  $X=B$  et  $A$  sinon.

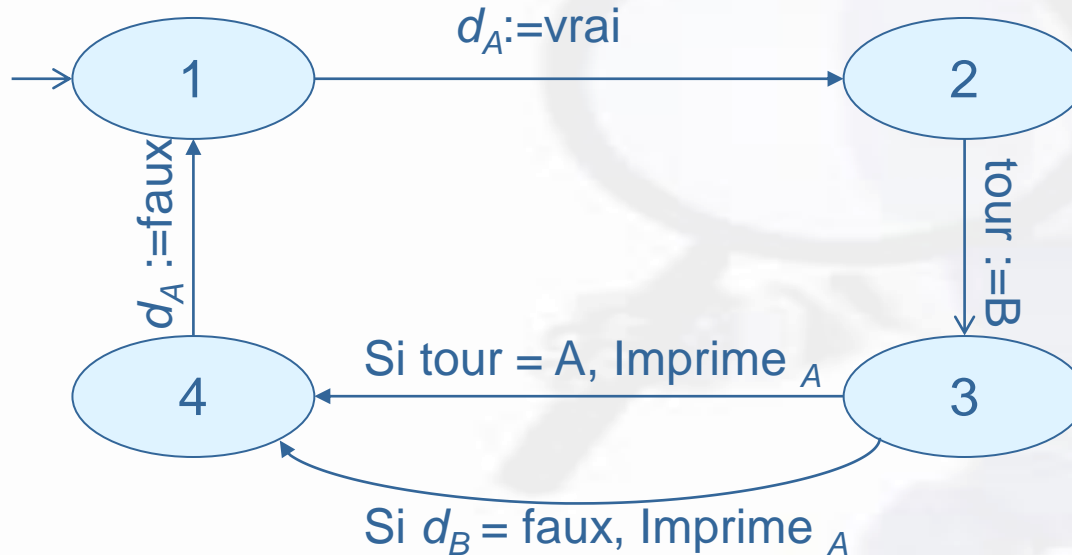
```
 $d_x := \text{vrai}$   
 $tour := \bar{X}$   
tant que ( $d_x \ \&\& \ tour = \bar{X}$ )  
    //Attendre  
     $section\_critique()$   
 $d_x := \text{faux}$ 
```

□ Garantit plusieurs propriétés, entre autres l'exclusion mutuelle.



# Imprimante de Peterson synchronisée

## □ L'utilisateur A chez Peterson



## □ L'utilisateur B est construit de la même façon.





## Propriétés de l'automate de Peterson

- Un état de l'automate synchronisé est un quintuplet (état de A, état de B, valeur  $d_A$ , valeur  $d_B$ , valeur de tour). L'automate a donc  $4 \times 4 \times 2 \times 2 \times 2 = 128$  états.
- Le nombre d'états atteignables est 21.
- Un model checker peut vérifier qu'aucun état de la forme  $(4, 4, -, -, -)$  n'est atteignable garantissant l'impossibilité d'impression simultanée par A et B .
- Il peut aussi vérifier que toute demande d'impression finit par être satisfaite.



# Produit synchrone

- ❑ Cas particulier du produit synchronisé : systèmes totalement dépendants.
- ❑ Les composants doivent évoluer en même temps.
- ❑ A chaque instant tous les sous systèmes doivent exécuter une action.
- ❑ Exemple : feu de circulation.



# Produit asynchrone

- ❑ Cas particulier du produit synchronisé : systèmes indépendants.
- ❑ Exécution non simultanée des sous systèmes (interleaving).
- ❑ A chaque instant un seul composant est autorisé à exécuter une transition.
- ❑ Exemple : protocoles de communication.



# Automates temporisés

- ❑ Les automates classiques permettent de modéliser des systèmes à événements discrets sans considération du délais (temps) séparant 2 actions.
- ❑ Une propriété non temporelle (ordonnancement) :
  - ❑ si détection d'erreur alors déclenchement d'alarme.
- ❑ Une propriété temporelle (délai) :
  - ❑ si détection d'erreur alors déclenchement d'alarme dans moins de 5s.



# Pourquoi temporiser ?

- Cette notion temporelle est importantes pour la modélisation des systèmes réactifs (temps réel).



# Modélisation des automates temporisés

- Pour modéliser des automates temporisés, il est possible d'ajouter des horloges globales contrôlant les actions d'un **réseau d'automates**.
- Deux façons de représenter l'horloge :
  - Temps discret,
  - Temps continu.



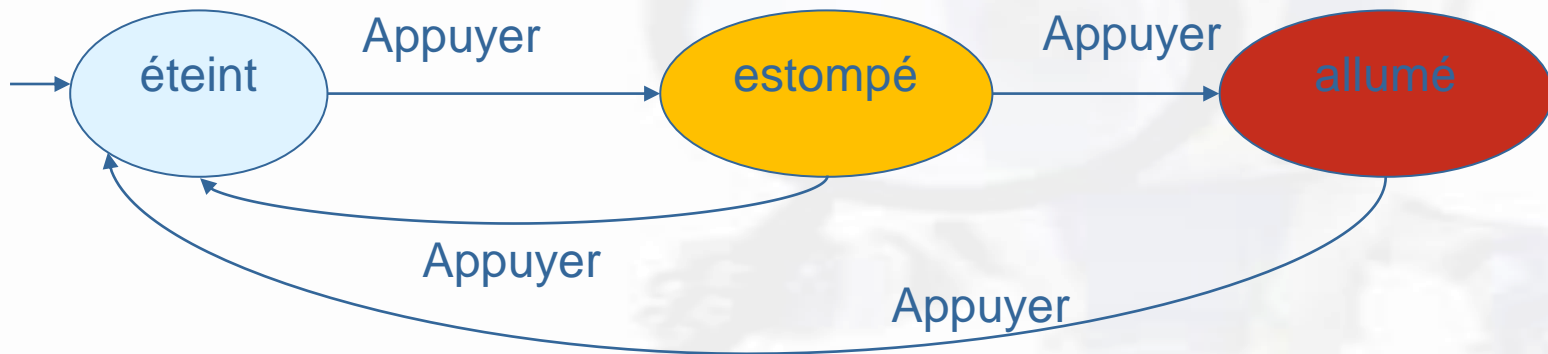
# Temps discret

- ❑ Les valeurs du temps sont des entiers positifs.
- ❑ Une transition spéciale ajoutée à l'automate classique : *ticks* d'horloge pour marquer l'écoulement du temps (unité choisie).
- ❑ Le temps est discret car entre 2 ticks successifs, le comportement du système est non observable.



# Exemple d'un Gradateur

## ❑ Interrupteur intelligent (gradateur) :



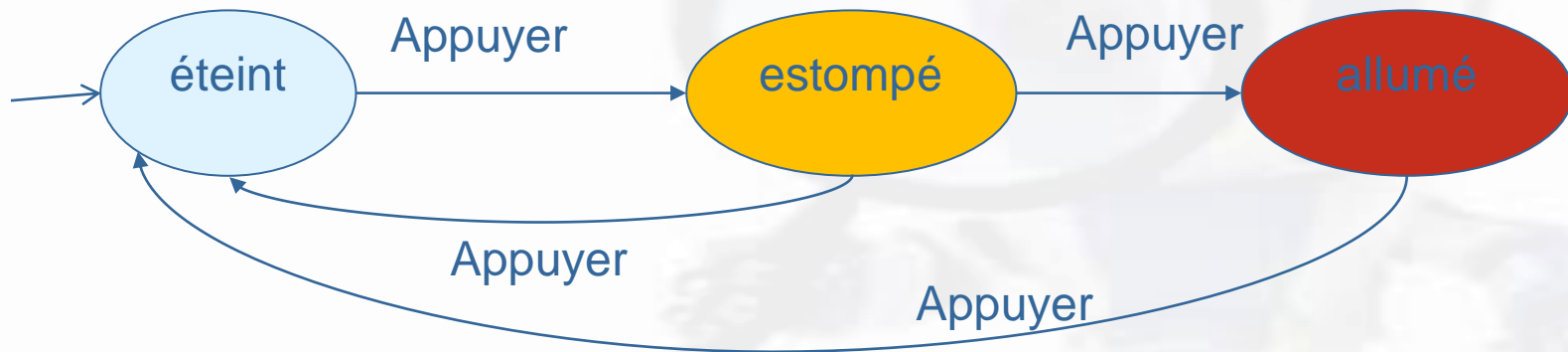
- ❑ Appuyer 2 fois rapidement pour allumer.
- ❑ Appuyer 1 fois pour une lumière estompée.
- ❑ Appuyer une fois, quand la lumière est allumée (estompée ou allumée), pour éteindre.





# Exemple d'un Gradateur

❑ Interrupteur intelligent (gradateur) :

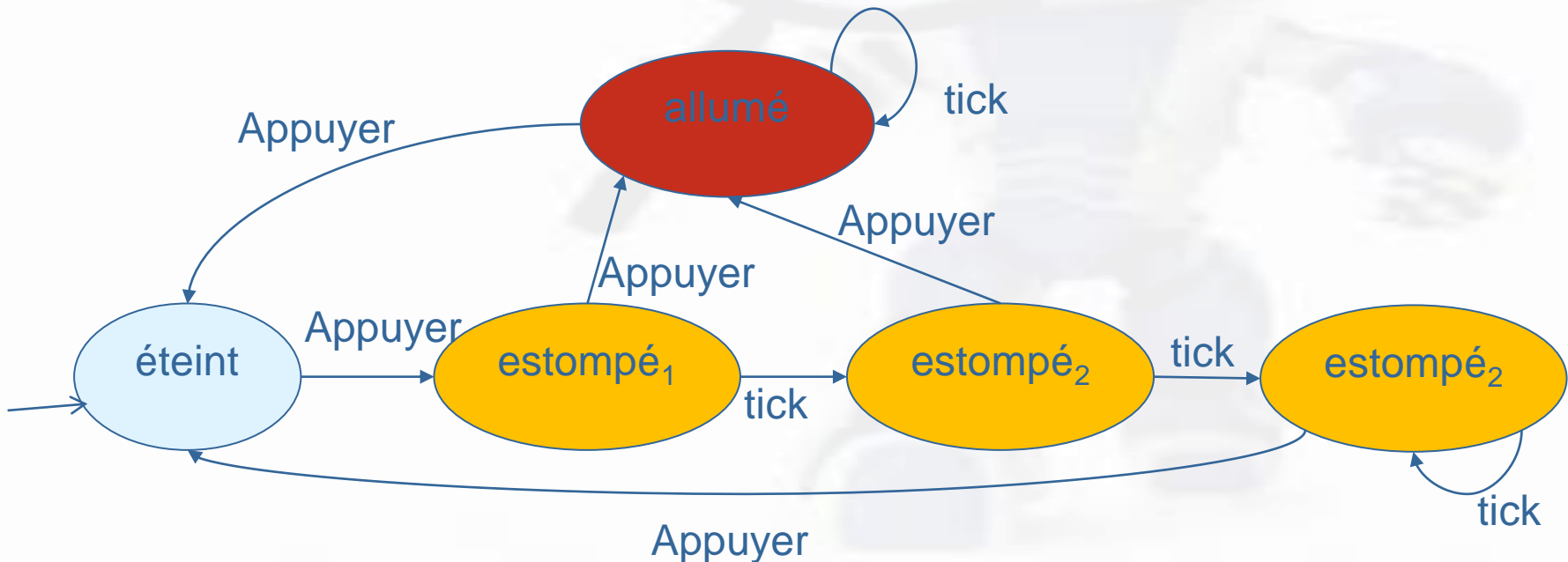


- ❑ Appuyer 2 fois rapidement pour allumer.
- ❑ Appuyer 1 fois pour une lumière estompée.
- ❑ Appuyer une fois, quand la lumière est allumée (estompée ou allumée), pour éteindre.
- ❑ **Mais le temps n'a pas été spécifié !**



# Gradateur avec ticks

- Un tick représente  $\frac{1}{2}$  seconde.
- Un appui rapide représente 2 appuis en moins d'une seconde.





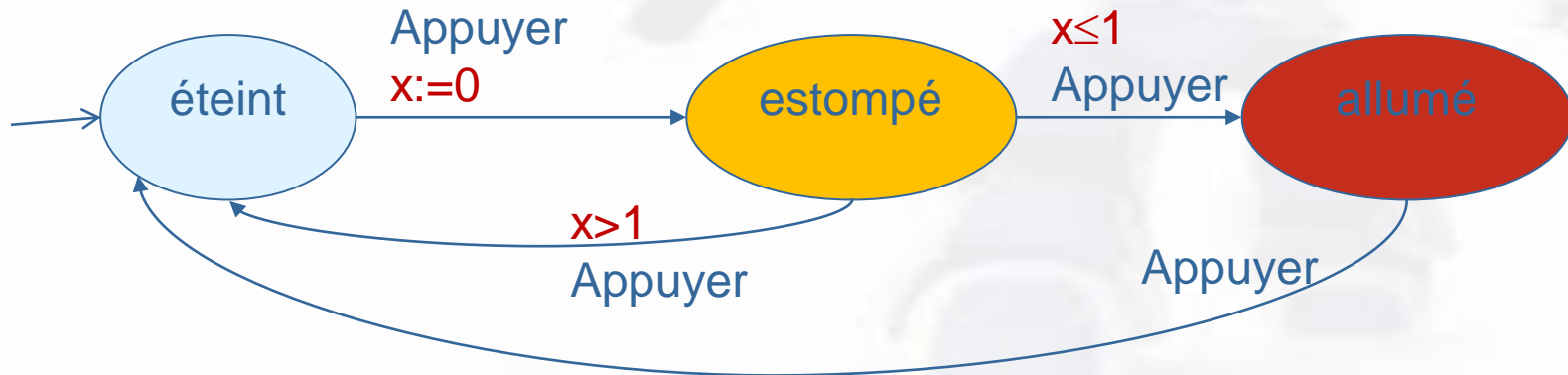
# Temps discret

- ❑ La taille du modèle croît avec la finesse du temps, ce qui entraîne une perte de lisibilité du modèle.
- ❑ Mélange entre transitions temporelles et celles qui modifient l'état de contrôle : modifier la procédure du gradateur en remplaçant  $\frac{1}{2}$  s séparant 2 appuis successifs par 10 s oblige le changement de la structure de l'automate.
- ❑ Manque de précision.



# Temps continu

- Le temps est précis puisqu'il est modélisé par  $\mathbb{R}_{\geq 0}$ .
- Des horloges mesurent le temps au sein du système.





# Automates temporisés

- ❑ Proposés par Alur et Dill en 1994.
- ❑ Il se compose de deux éléments :
  - ❑ Un automate fini décrivant les états de contrôle du système et les transitions, supposément instantanées, entre les états.
  - ❑ Des horloges, associées aux transitions, spécifient des contraintes quantitatives de temps.



# Description d'un automate temporisé

- Un automate temporisé est composé de :
  - Un graphe d'états (structure de kripke) décrivant les états du système avec transitions instantanées entre les états.
  - Un ensemble fini d'horloges  $H=\{h_1, h_2, \dots, h_n\}$ , associées aux transitions, utilisées pour spécifier des contraintes quantitatives de temps. Elles sont des variables à valeurs réelles positives et à l'état initial du système elles ont toutes la valeur 0.



# Transitions d'un AT

- Une transition d'un automate temporisée est un triplet d'actions :
  - Contrainte d'horloge (appelée aussi garde ou condition de franchissement) : portée sur les valeurs des horloges et spécifie une contrainte pour le franchissement de la transition.
  - Action : une action de l'automate fini.
  - Des actions de remise à zéros de certaines horloges.



- Un automate temporisé  $\mathcal{AT} = \langle Q, E, q_0, H, T \rangle$  est défini par ;
  - $Q$  un ensemble fini d'états de contrôle,
  - $E$  ensemble d'étiquettes d'actions,
  - $q_0$  l'état initial de l'automate,
  - $H$  un ensemble d'horloges
  - $T \subseteq Q \times C(H) \times E \times 2^H \times Q$
- Avec  $C(H)$  est une contrainte d'horloge présentée sous formule d'une formule  $\varphi$  ;
  - $\varphi ::= c \leq h \mid c < h \mid c \geq h \mid c > h \mid \varphi_1 \wedge \varphi_2 : h \in H, c \in Q$  (nombres rationnels).
- Et  $2^H$  est un ensemble d'horloges remises à zéro.





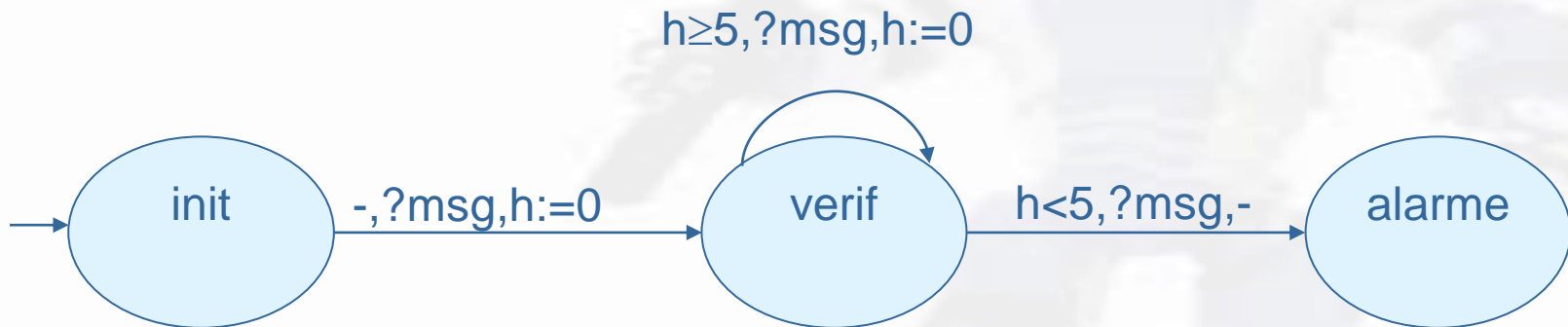
# Exemple d'un AT

- système déclenchant une alarme lorsque le délai séparant 2 réceptions de messages (?msg) est inférieur à 5 secondes.



# Exemple d'un AT

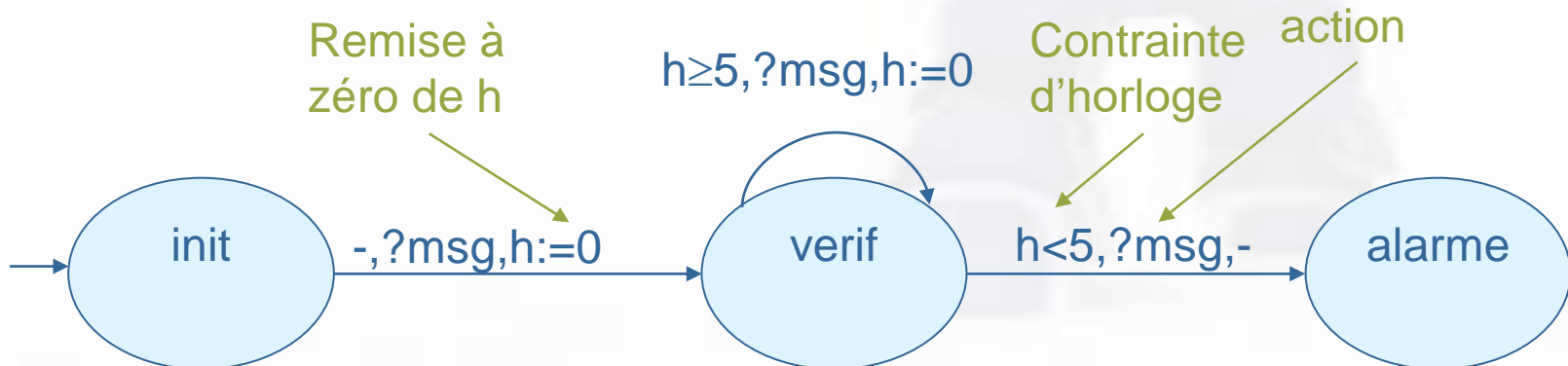
- système déclenchant une alarme lorsque le délai séparant 2 réceptions de messages (?msg) est inférieur à 5 secondes.





# Exemple d'un AT

- Une seule horloge  $h$ , elle est mise à zéro à l'état initial.
- La valeur de  $h$  augmente avec le temps.
- Dès qu'un message est reçu, on passe à l'état *verif* et  $h$  est remise à zéro.
- A la prochaine réception de message :
  - Si  $h < 5$ , l'alarme est déclenchée (état *alarme*)
  - Si  $h \geq 5$ , on peut recevoir un message et on doit initialiser  $h$ .





## Remarque

- ❑ Il est possible d'utiliser plusieurs horloges, notamment pour modéliser des systèmes communicants.
- ❑ Toutes les horloges doivent progresser à la même vitesse de manière synchrone.



# Configuration

- La configuration d'un système définit son état global, elle est donnée par :
  - L'état de contrôle courant, et
  - La valeur de chaque horloge.
- Elle est notée  $(q, v)$  :
  - $q$  : un état de contrôle,
  - $v \in H \rightarrow \mathbb{R}^+$  : une application de *valuation* associant à chaque horloge sa valeur courante.



# Configuration

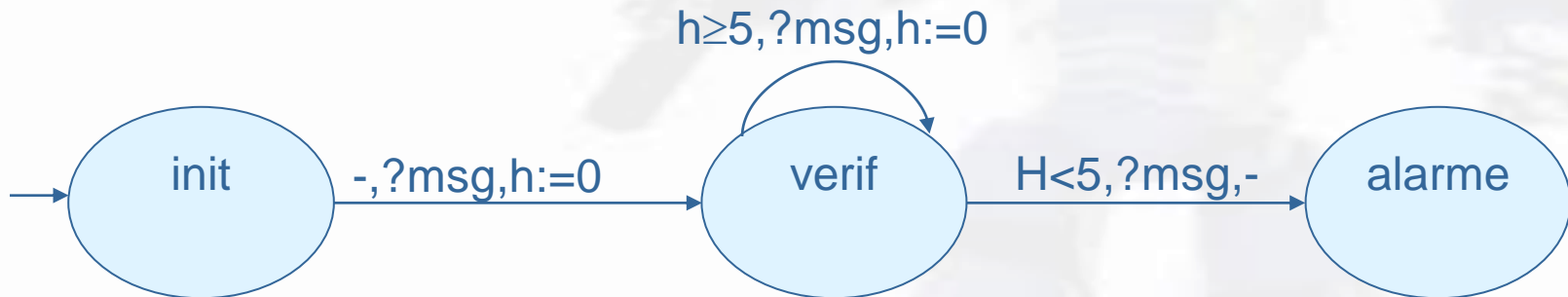
- Le système change de configuration par :
  - Transition de délais : laisse s'écouler un certain délai  $d$  (toutes les horloges s'incrémentent de  $d$  :  $v+d$ ). On passe donc de la configuration  $(q, v)$  à  $(q, v+d)$ .
  - Transition de d'action (discrète) : l'automate s'active par une transition franchissable. L'initialisation de l'un ou de plusieurs horloges est possibles, les valeurs des autres horloges restent alors inchangées.



# Exemple de configurations

- Evolution de l'automate de l'exemple précédent en partant de la configuration (init,0) :

$(init,0) \rightarrow (init,10.2) \xrightarrow{?msg} (verif,0) \rightarrow (verif,5.8) \xrightarrow{?msg} (verif,0) \rightarrow (verif,3.1) \xrightarrow{?msg} (alarme,3.1), \dots$



- Cette séquence de configurations est appelée **exécution**.



# Exécution

- Une **exécution** de l'AT est une séquence (infinie) de configurations, chaque configuration est soit :
  - Une action de l'automate, soit
  - L'écoulement d'un délai.





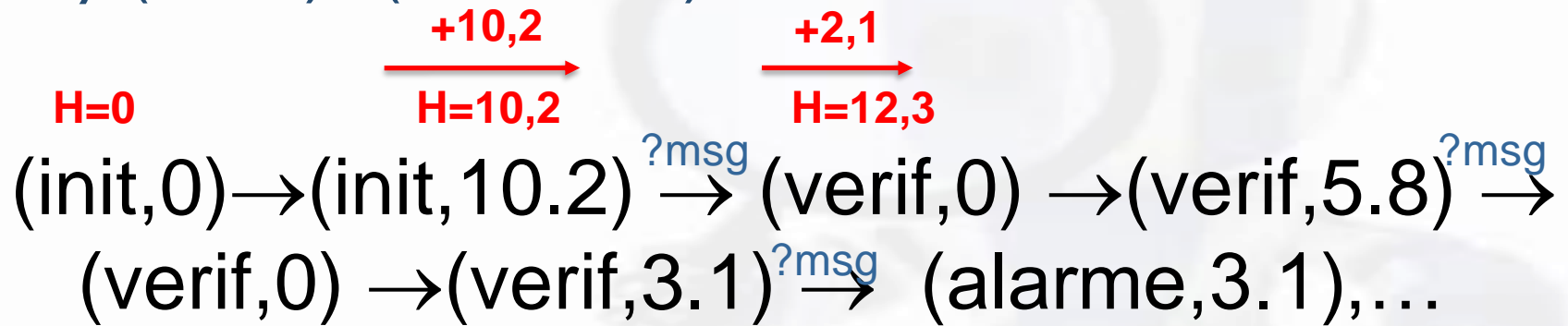
# Trajectoire

- ❑ Une trajectoire est une exécution vue globalement.
- ❑ C'est une application  $\rho$  de  $\mathbb{R}^+$  vers l'ensemble des configurations  $(q,v)$ .
- ❑ L'état initial est noté  $\rho(0)$  et  $\rho(t)$  est la configuration à l'instant  $t$  le long de cette configuration.
- ❑ Exemple :  $\rho(12.3)=(\text{verif},2.1)$



# Trajectoire : exemple

□  $\rho(12.3)=(\text{verif}, 2.1)$





# Réseaux d'AT et synchronisation

- ❑ Composer des automates temporisés et les synchroniser : réseau temporisé.
- ❑ Une exécution du réseau est une séquence de configurations.
- ❑ Une configuration comporte l'état de contrôle de chaque automate du réseau et la valeur de chacune des horloges.
- ❑ Un changement de configuration :
  - ❑ Écoulement d'un délai  $d$ , toutes les horloges s'incrémentent de  $d$ .
  - ❑ Exécution d'une action franchissable



# Synchronisation

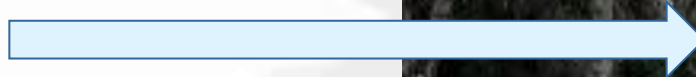
- Si 3 automates communiquent, le réseau passe de la configuration  $(q_0, q_1, q_2, v)$  à  $(q'_0, q'_1, q'_2, v')$  en exécutant les actions synchronisées  $a_0, a_1, a_2$ , alors  $v'$  ne diffère de  $v$  que par les horloges remise à zéro par ces transitions.



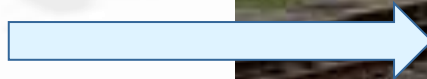
Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

## Exemple : passage à niveau

□ Un train



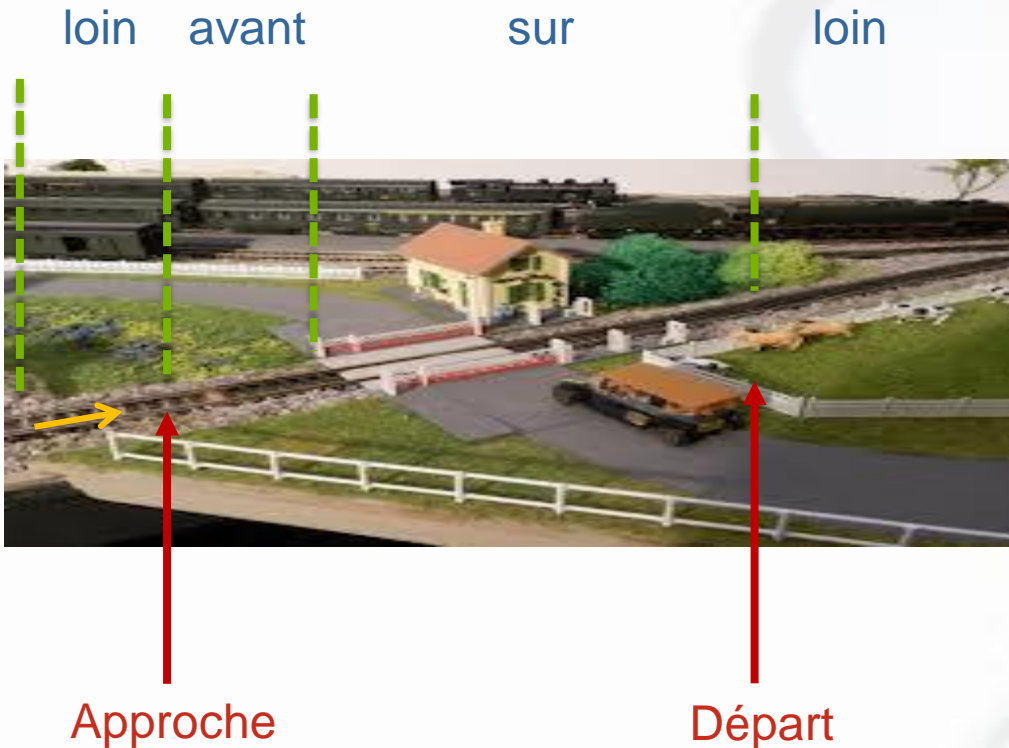
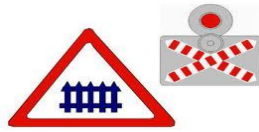
□ Une barrière





Modélisation  
Exemples de modèles  
Concepts généraux  
Automates avec variables  
Automates communicants  
Automates temporisés  
Abstraction

# Passage à niveau



- ☐ Deux capteurs : **Approche** et **Départ** signalant à la barrière l'approche du train et son éloignement.
- ☐ Le train peut être :
  - ☐ **loin** : hors de la considération de la barrière.
  - ☐ **avant** : entre le capteur d'approche et le début de la zone protégée par la barrière.
  - ☐ **sur** : sur la zone protégée.



# Passage à niveau

□ **Système faisant coopérer 2 composants :**

□ Le train

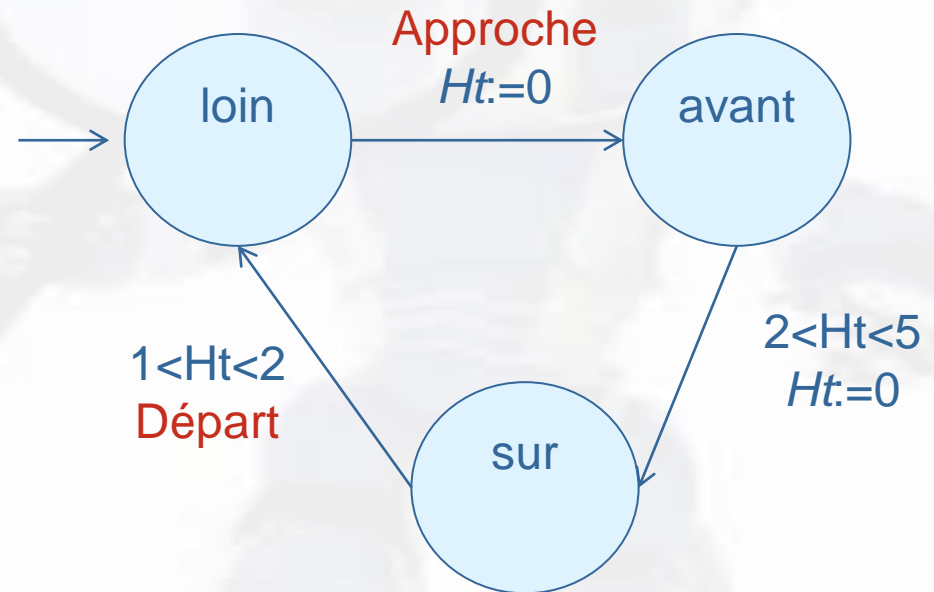
□ La barrière

□ **Système = train || barrière**



# Modèle du train

- ❑ Le train passe de l'état loin à l'état avant en émettant un signal Approche.
- ❑ Le train met entre 2 et 5 mn pour parcourir la portion avant (arrive au passage à niveau).
- ❑ Le train met entre 1 à 2 mn pour parcourir la portion sur (quitter la zone protégée).

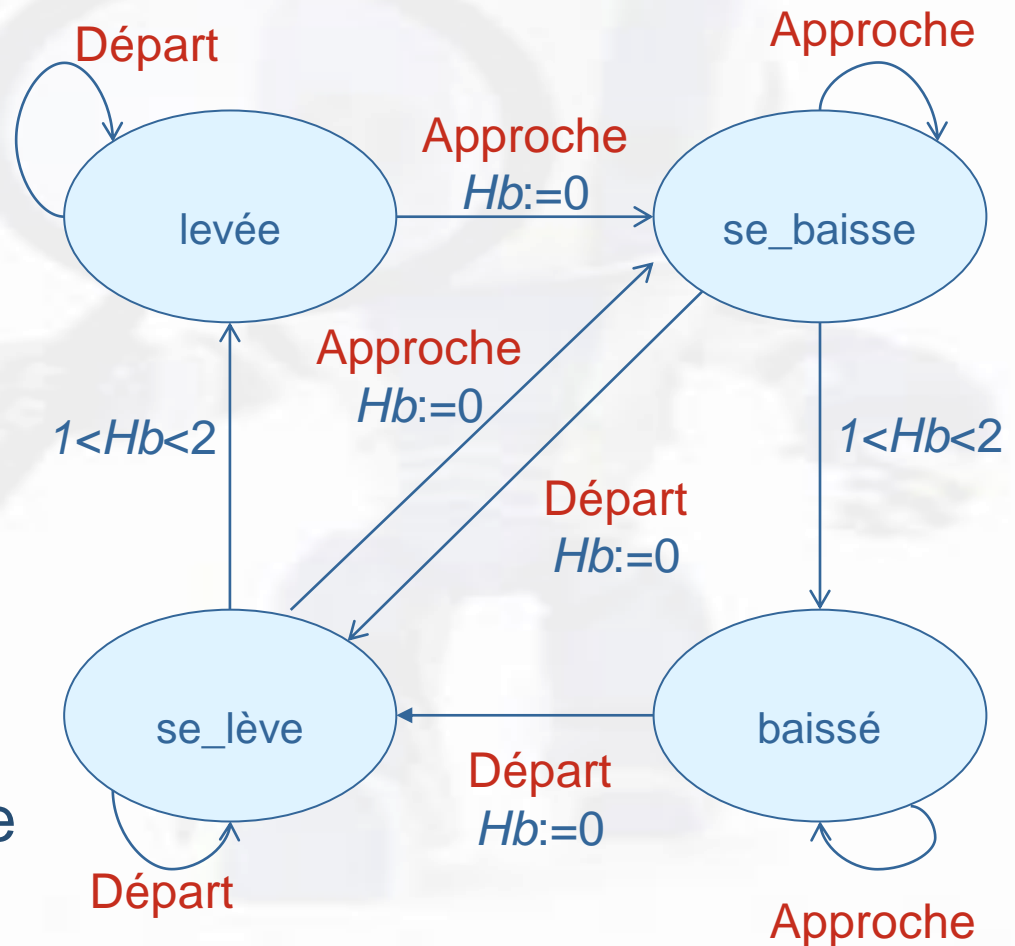






# Modèle de la barrière

- Lorsque la barrière reçoit le signal **Approche** (le train arrive), elle doit se mettre à baisser.
- Lorsqu'elle reçoit le signal **Départ** (train s'éloigne), elle doit se mettre à se lever.
- Elle met entre 1 à 2 mn pour se lever et se baisser.





# Synchronisation

- ❑ La synchronisation se fait sur les variables partagées : **Approche** et **Départ**.
- ❑ Par exemple, lorsque le train passe de l'état *loin* à l'état *avant* par la transition **Approche**, la barrière doit franchir une transition étiquetée par **Approche** pour assurer la synchronisation.

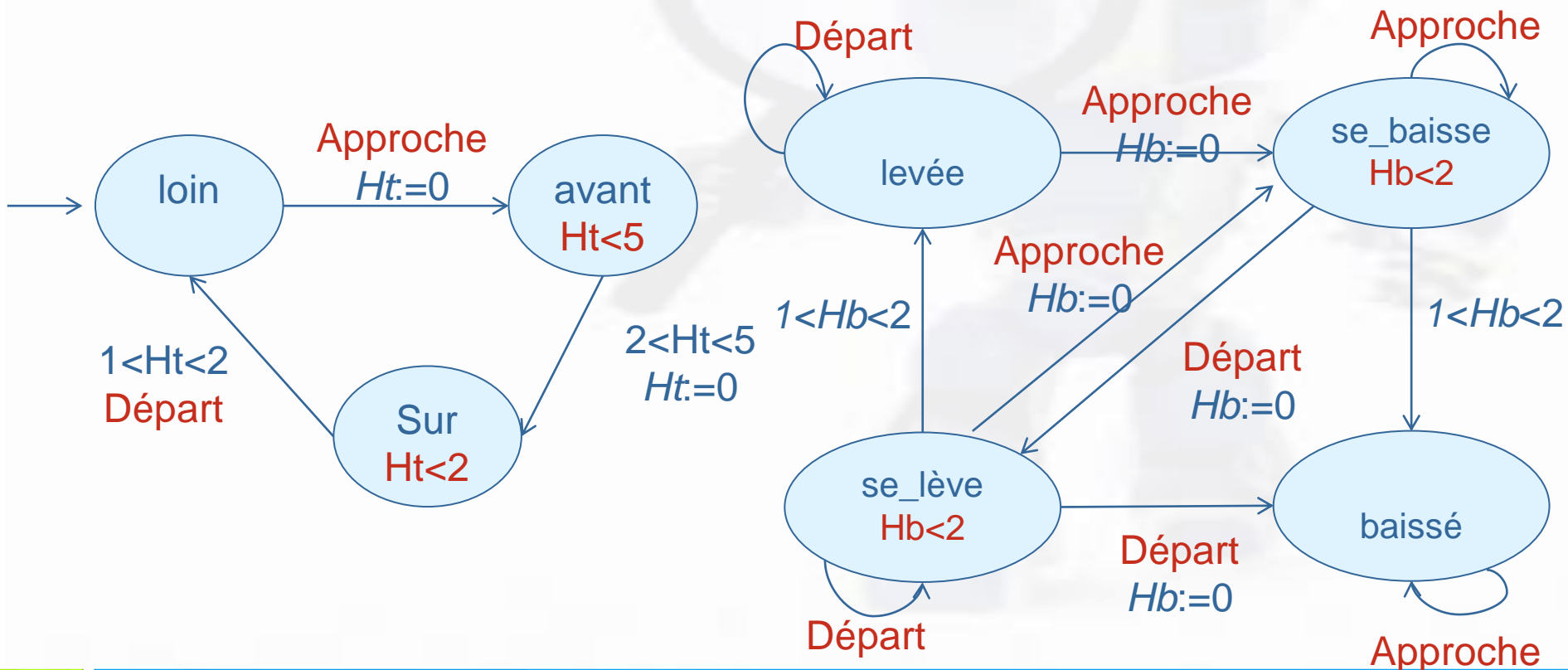


# Synthèse

- ❑ Cette modélisation peut aboutir à des blocages sur les actions franchissables.
- ❑ Exemple : si le train reste plus de 5 mn dans l'état avant, il y reste indéfiniment car l'action menant à l'état *sur* n'est plus déclanchable (propriété de vivacité non respectée).
- ❑ Une solution serait l'introduction d'invariants dans les états.



# Invariants pour le passage à niveau





# Invariants

- ❑ Les invariants ajoutés dans les états obligent les transitions d'écoulement du temps de les respecter.
- ❑ A la fin du délai, le système doit obligatoirement franchir une transition discrète.
- ❑ Si aucune configuration autorisée n'est atteignable, il y a blocage : *divergence* (*livelock*).



# Passage à niveau amélioré

- ❑ **Système faisant coopérer 3 composants :**
  - ❑ Le train
  - ❑ La barrière
  - ❑ Le contrôleur
  
- ❑ **Système = train || barrière || contrôleur**



# Méthodes d'abstraction

- ❑ Techniques utilisées pour simplifier les automates.
- ❑ Basées sur le fait d'ignorer certains aspects de l'automate.
- ❑ Un modèle pour un système à vérifier peut ne pas être adéquat pour une technique automatique de model checking (ex. taille de l'automate).
- ❑ Pour répondre à la question  $Q1 : A \models \varphi$ , il faudrait alors abstraire  $A$  en  $A'$  et répondre à la question  $Q2 : A' \models \varphi$ .
- ❑ Il faut bien sûr s'assurer qu'une bonne réponse pour  $Q1$  soit aussi bonne pour  $Q2$ .