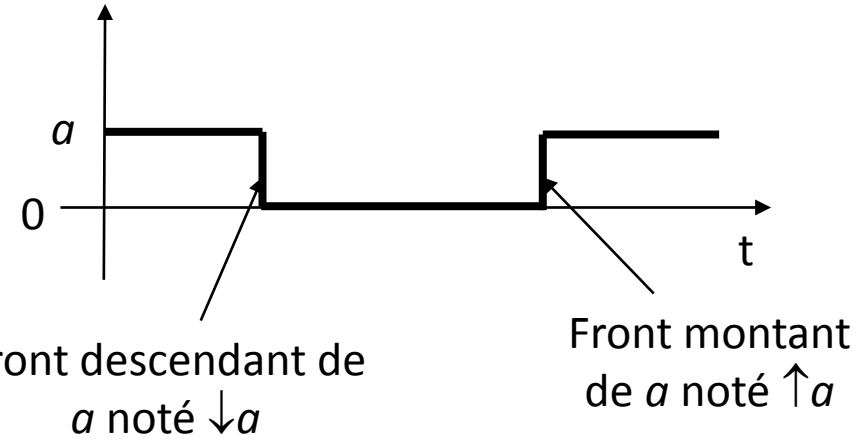
SII / Séquence n^{o} 10 – Modélisation et conception de la commande des systèmes

Modélisation des systèmes à évènements discrets (SED)

Évènements

- **Événement** : phénomène considéré comme localisé et instantané (discret), c'est-à-dire survenant en un point de l'espace et à un instant bien déterminé
(ex : appui sur un bouton).



L'apparition de *a* est appelé évènement **front montant**. (si booléen passage de 0 à 1).
Le passage de *a* de 1 à 0 est appelé évènement **front descendant**. (si booléen passage de 1 à 0).

Systèmes à évènements discrets

- **Systèmes continus** : systèmes à évolution temporelle décrite à l'aide de variables continues du temps (voir équations différentielles).
- **Systèmes à évènements discrets (SED)** : systèmes à évolution temporelle liée à l'apparition d'évènements discrets.

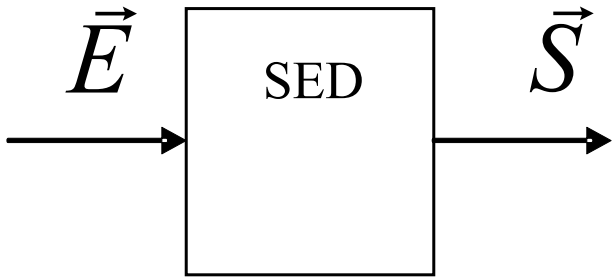
Systèmes à évènements discrets

- Les SED peuvent être décrits :
  - par des tables de vérité décrivant le cahier des charges ;
  - par des équations logiques (algèbre de Boole) et/ou des logigrammes ;
  - par des chronogrammes ;
  - par des diagrammes SysML comportementaux (diagramme de séquences, **diagramme d'états**, diagramme d'activités).

Systèmes à évènements discrets

Pour chaque système étudié, on peut définir, une frontière d'étude.

- Les évènements entrant sont notés $\vec{E}$
- Les sorties fournies par le SED sont notées $\vec{S}$

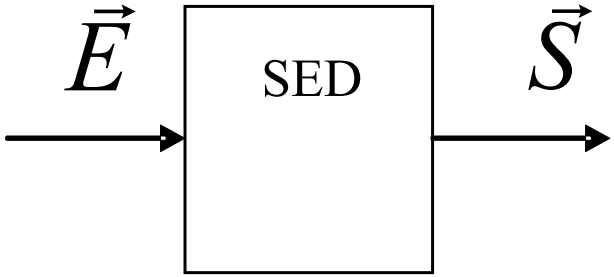


On distingue 2 catégories fondamentales de SED :

- les SED combinatoires ;

- les SED séquentiels.

SED combinatoires



2 propositions définissent un SED combinatoire :

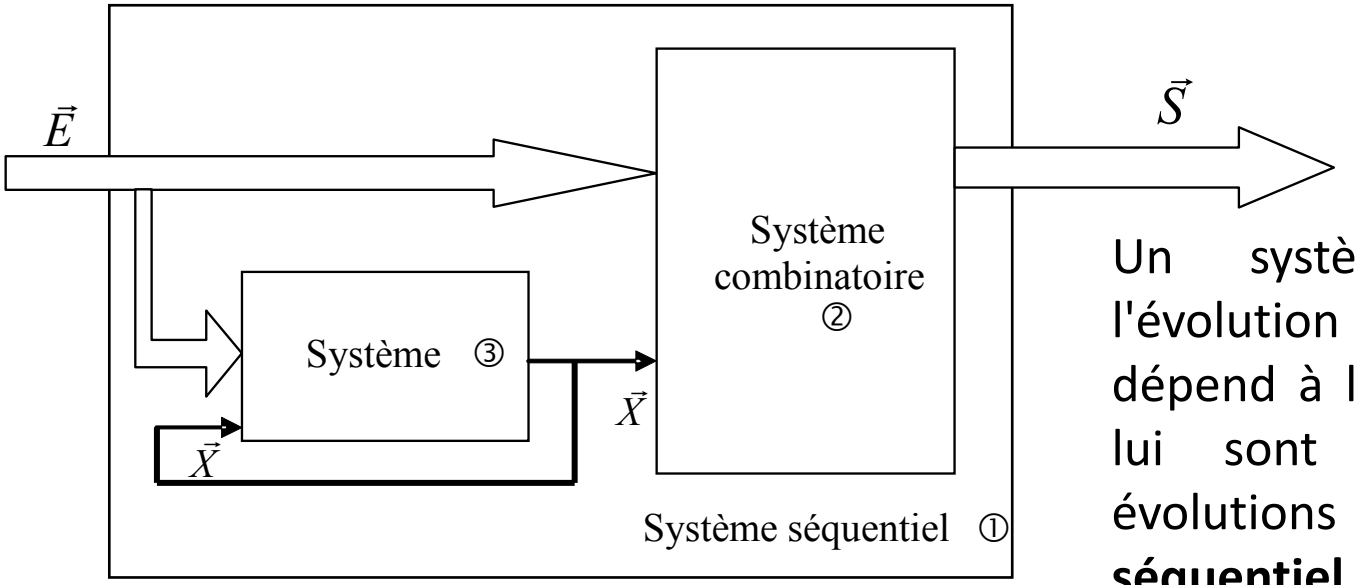
- À une valeur de vecteur d'entrée $\vec{E}$ correspond une seule valeur du vecteur de sortie $\vec{S}$.
- À une valeur du vecteur de sortie $\vec{S}$ peut correspondre plusieurs valeurs du vecteur d'entrée $\vec{E}$.

Un SED combinatoire peut être modélisé à l'aide d'une table de vérité dont les valeurs de sont sans équivoque.

SED Combinatoires ?

- *La gestion de la lumière de la salle*
- *Le bouton principal de votre smartphone*
- *Un va & vient à deux interrupteurs bistables*
- *Un va & vient à deux interrupteurs monostables*

SED séquentiels



Un système logique, dont l'évolution future de la sortie dépend à la fois des entrées qui lui sont appliquées et des évolutions passées, est un **SED séquentiel**.

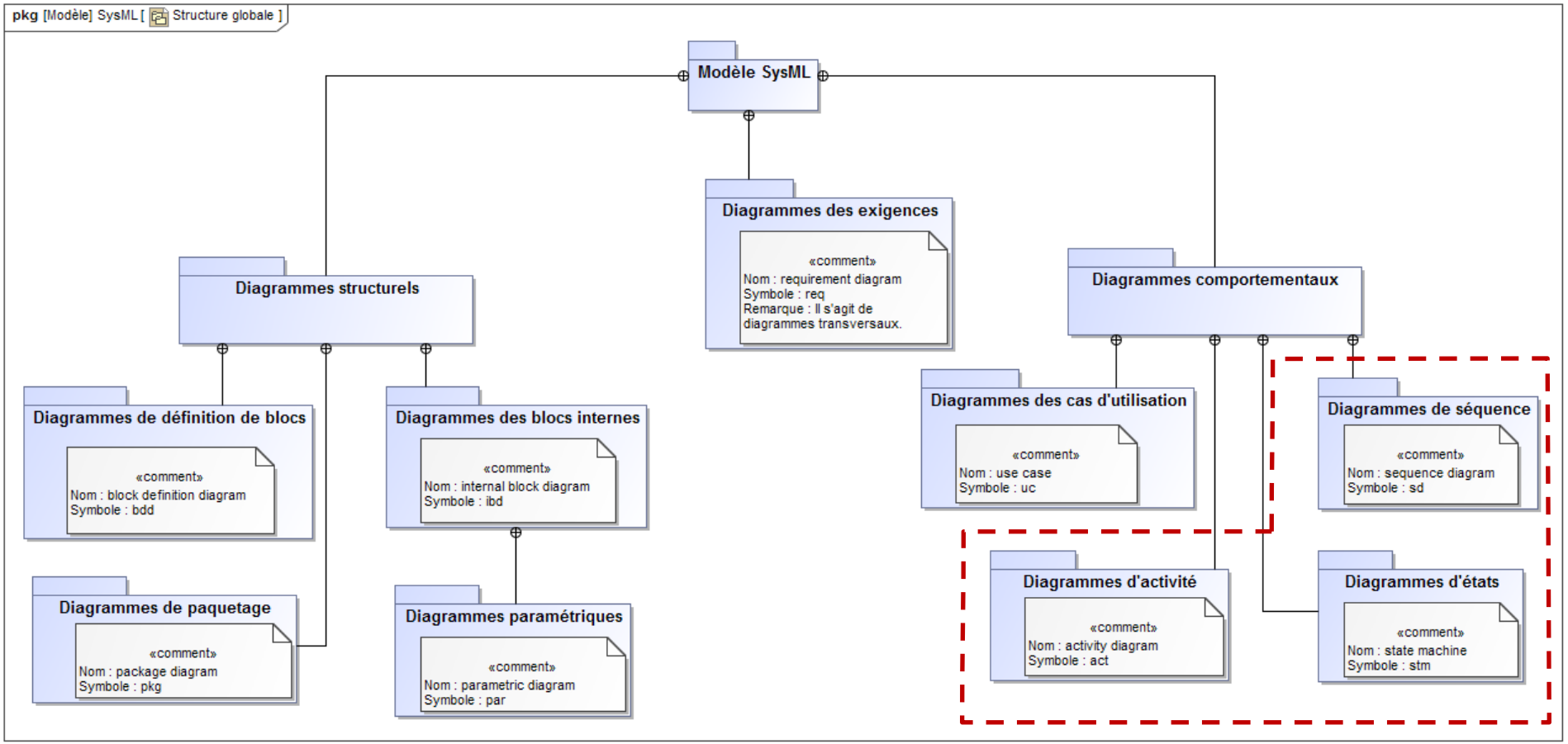
Deux propositions permettent de définir les relations entrée-sortie d'un **SED séquentiel** :

- À un état du vecteur d'entrée $\vec{E}$ peut correspond un ou plusieurs états du vecteur de sortie $\vec{S}$.

- À un état du vecteur de sortie $\vec{S}$ peut correspondre plusieurs états du vecteur d'entrée $\vec{E}$.

Un SED séquentiel ne peut pas être modélisé à l'aide d'une table de vérité sans ambiguïté $\Longrightarrow$ *nécessité de* ***mémoriser les états précédents****.*

SED en SysML


pkg [Modèle] SysML [Structure globale]
Modèle SysML
Diagrammes des exigences
«comment»
Nom : requirement diagram
Symbole : req
Remarque : Il s'agit de diagrammes transversaux.
Diagrammes structurels
Diagrammes comportementaux
Diagrammes de définition de blocs
«comment»
Nom : block definition diagram
Symbole : bdd
Diagrammes des blocs internes
«comment»
Nom : internal block diagram
Symbole : ibd
Diagrammes des cas d'utilisation
«comment»
Nom : use case
Symbole : uc
Diagrammes de séquence
«comment»
Nom : sequence diagram
Symbole : sd
Diagrammes de paquetage
«comment»
Nom : package diagram
Symbole : pkg
Diagrammes paramétriques
«comment»
Nom : parametric diagram
Symbole : par
Diagrammes d'activité
«comment»
Nom : activity diagram
Symbole : act
Diagrammes d'états
«comment»
Nom : state machine
Symbole : stm

Diagrammes comportementaux

- Si l'on souhaite modéliser l'interaction acteur/système, il est conseillé d'utiliser les diagrammes de séquences (sd).
- Si l'on souhaite décrire une succession d'actions élémentaires mais sans évènement extérieur, il faut utiliser le diagramme d'activité (act) qui correspond à un algorithme (interagissant avec les autres diagrammes SysML).
- Dès qu'un évènement doit être pris en compte pour modéliser le comportement dynamique interne d'un bloc, on doit utiliser les diagrammes d'états (stm).

Définitions fondamentales

- **État** : situation durant la vie d'un bloc pendant laquelle :
  - il satisfait une certaine condition ;
  - il exécute une certaine **activité** ;
  - il attend un certain **évènement.**

- **Activité** : séquence d'**actions** pouvant être interrompue.

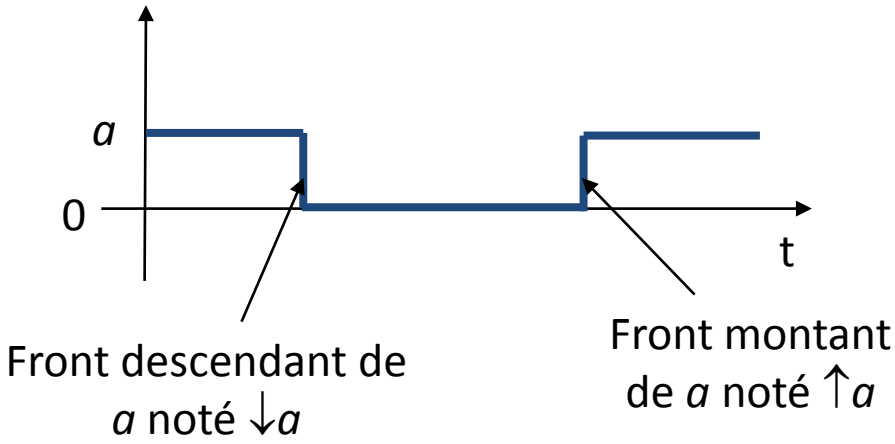
- **Action** : représente un comportement élémentaire du système. qui ne peut pas être interrompu. (ex : affectation de variable, incrémentation compteur, « envoi signal »…).

- **Evénement** : phénomène discret, localisé et instantané (ex : appui sur un bouton).

Qualités d'un système de décision

Qualités d'un système de décision :

- **Attentif** : réceptif à tout événement externe quel que soit l'instant où il se produit.
- **Efficace** : capable de traiter les conséquences de tout événement externe auquel il est réceptif.
- **Déterminé** : à circonstances identiques, comportement identique.



- L'apparition d'un évènement est appelé **déclencheur** (trigger).

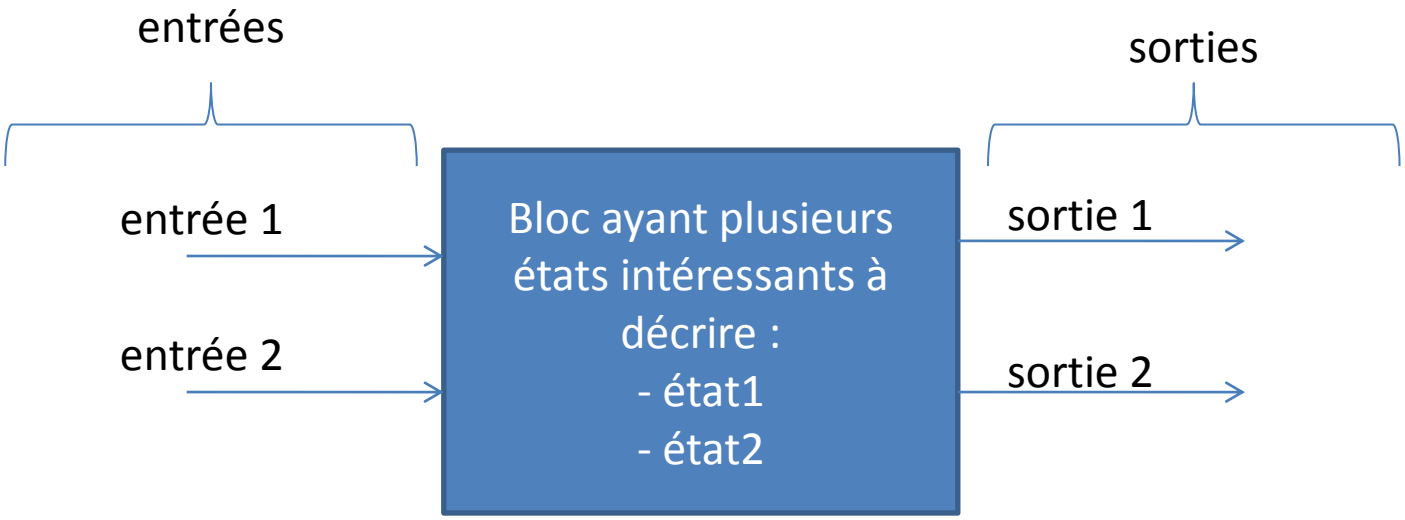
Tous les déclencheurs sont asynchrones (distincts dans le temps).
La conjonction ($\uparrow a \cdot \uparrow b$) est donc toujours fausse !

Diagramme d'états

stm *(state machine)*

- C'est un graphe, dont les sommets représentent les états, et les arcs orientés, les transitions.
- C'est un automate à nombre fini d'états (ou machine à états finis).
- Il permet de modéliser le comportement dynamique (« cycle de vie ») d'un bloc (ou instance de bloc) selon des scénarii définis au préalable.
- Il contient :
  - un état initial ;
  - un ensemble de déclencheurs (par extension des évènements) ;
  - un ensemble d'états possibles du bloc ;
  - un ensemble d'activités et/ou actions.

Diagramme d'états (**stm**)

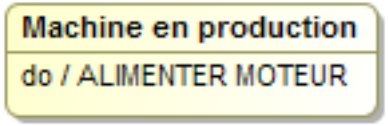


En fonction des apparitions des évènements (occurrence) provenant de certaines entrées et de **l'unique état actif du bloc**, le diagramme d'états évolue : l'état courant devient inactif et un autre devient actif pouvant ainsi exécuter une activité (activation d'une sortie).

États

- Un état modélise un comportement particulier du système modélisé : en général il est associé à une activité.
- Un état est représenté par un rectangle aux coins arrondis.

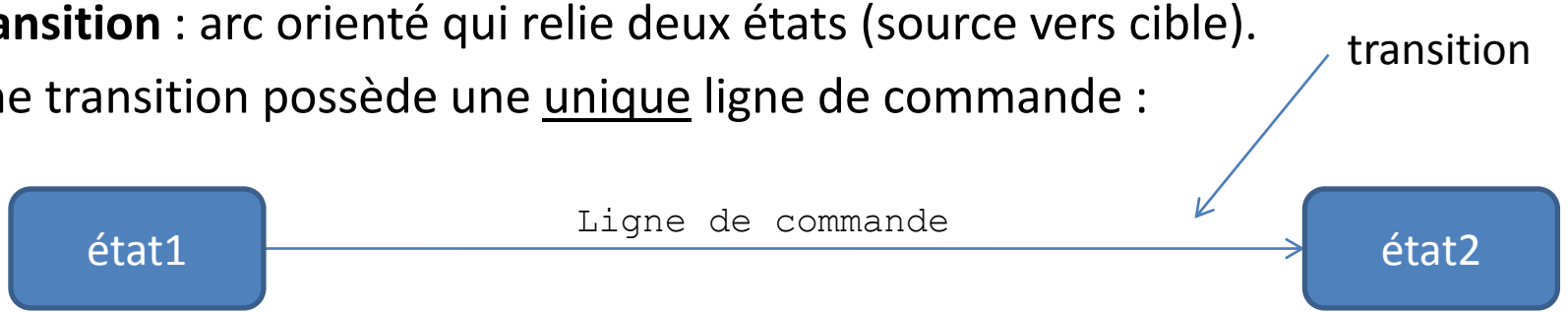
Exemple :



- Un état possède en général un titre informant du nom de l'état du bloc (ex : Machine en production). Ce titre est unique dans le diagramme.
- Il n'est pas possible d'incorporer des états prélevés de d'autres diagrammes sauf états sous-machine (voir + loin).
- Un état sans titre est un **état anonyme**.

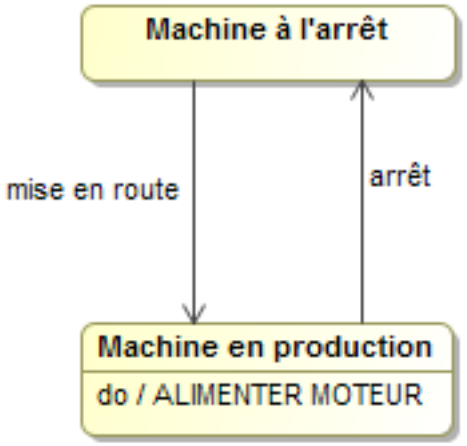
Transitions

- **Transition** : arc orienté qui relie deux états (source vers cible).
- Une transition possède une unique ligne de commande :



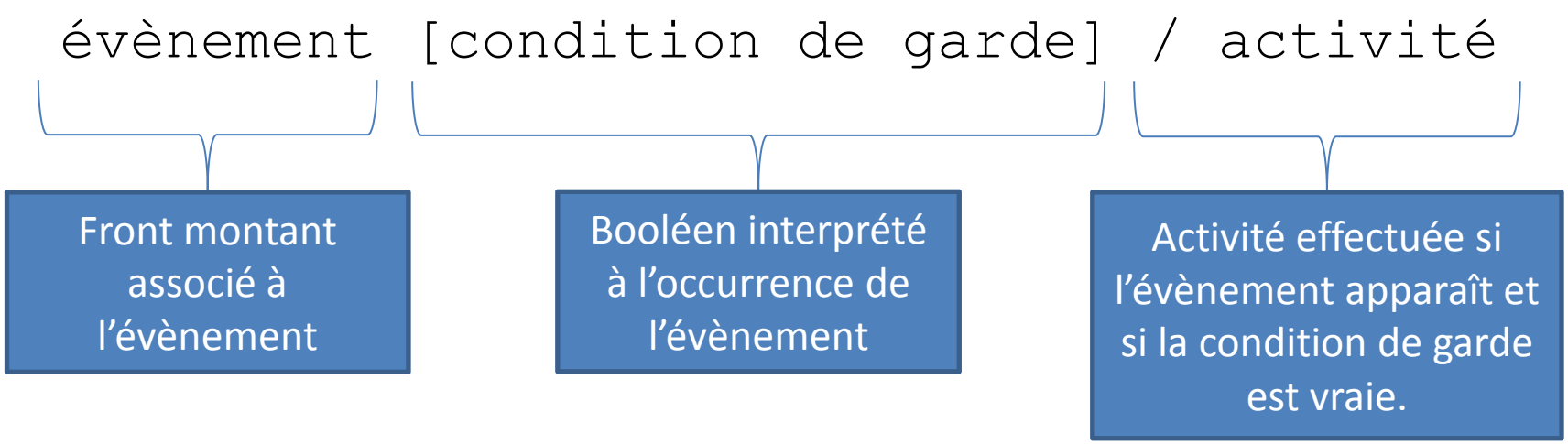
- Pour activer l'état2, il faut :
  - Que l'état1 soit actif
  - Et que la ligne de commande soit **validée**. La transition est franchissable.

- Si ces conditions sont vraies, alors l'état1 se désactive et l'état2 s'active.



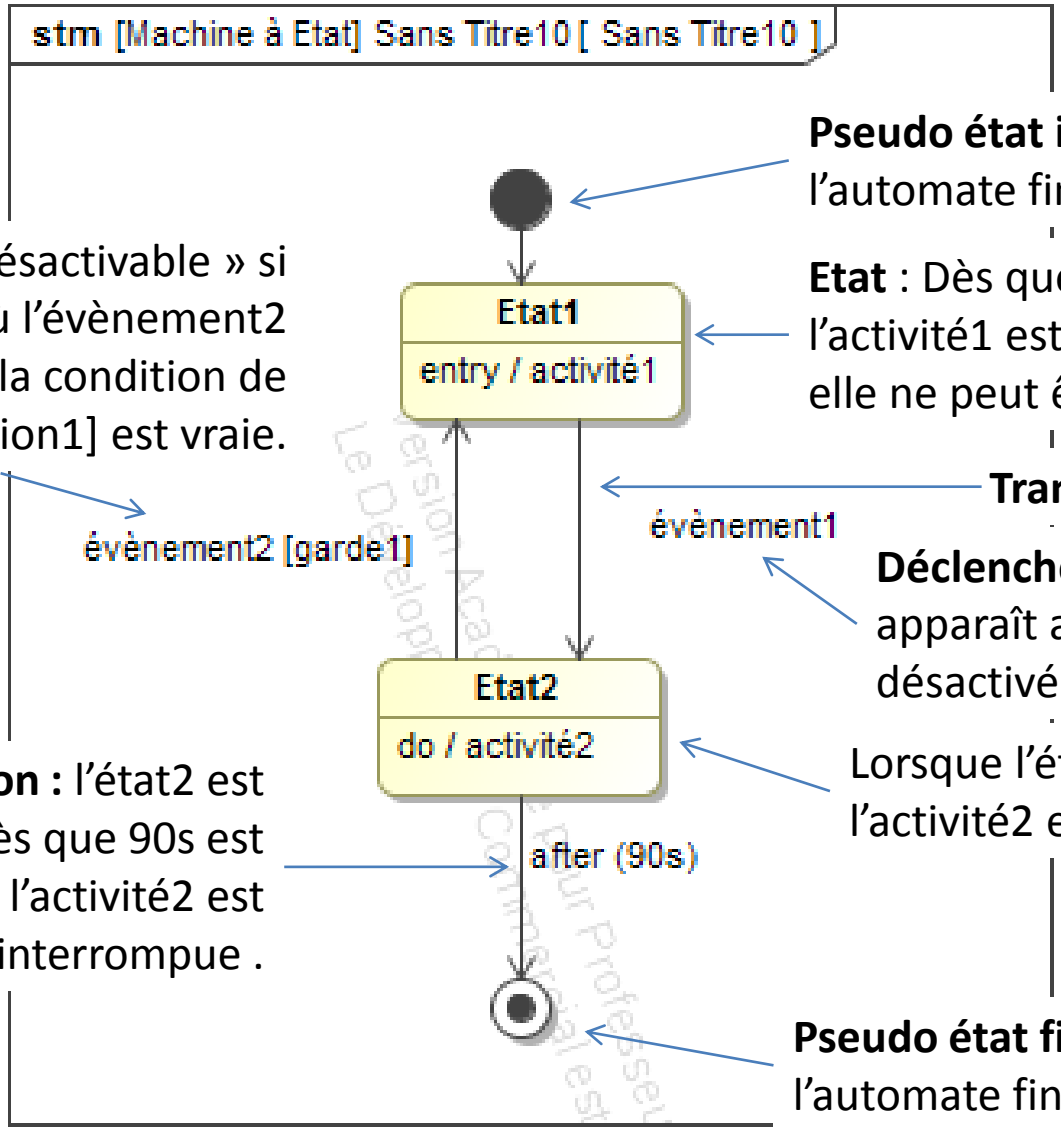
Ligne de commande stm

- L'écriture d'une ligne de commande :



Si le front montant de évènement apparait et que la condition de garde est vraie alors la ligne est validée et l'activité est effectuée.
Chaque terme est optionnel.
Si évènement n'est pas écrit, c'est le front montant de la fin de l'activité de l'état courant (associée à `do`) qui valide la ligne (déconseillé).
Si la ligne est associée à un arc du diagramme d'état (transition), il est courant de ne pas mettre d'activité.
Les lignes de commande peuvent être écrites dans les transitions **et** dans les états.

Diagramme d'états (**stm**)



Pseudo état initial : l'automate fini démarre

L'état2 est « désactivable » si à l'instant où l'évènement2 apparaît, la condition de garde [condition1] est vraie.

Etat : Dès que l'état1 est actif, l'activité1 est exécutée. Cas particulier elle ne peut être interrompue.

Transition

Déclencheur : lorsque celui-ci apparaît alors l'état 1 est désactivé et l'état 2 est activé.

Lorsque l'état2 est activé, l'activité2 est exécutée une fois.

Transition : l'état2 est désactivé dès que 90s est atteint ; l'activité2 est interrompue .

Pseudo état final : l'automate fini s'arrête.

Exemple générique de base

Évènements

- Un évènement est par nature instantané et est traité immédiatement.
- Il existe plusieurs types d'évènements :
  - l'évènement de signal (signal event) : un signal asynchrone est arrivé (ex : appui sur un bouton);
  - l'évènement de changement (change event) : une valeur interne au modèle a changé (« `when` (N=10) ») (ex : compteur atteint) ;
  - l'évènement temporel (time event) :
    - Relatif : `after`(90s) passe à vrai 90s après l'entrée dans l'état courant.
    - Absolu : `at`(11:00) passe à vrai (toutes les 11h) sur une base de temps absolue.

`when, after, at` sont des mots réservés d'évènements.

États

- Chaque état possède des lignes de commandes optionnelles :

`évènement [condition de garde] / activité`

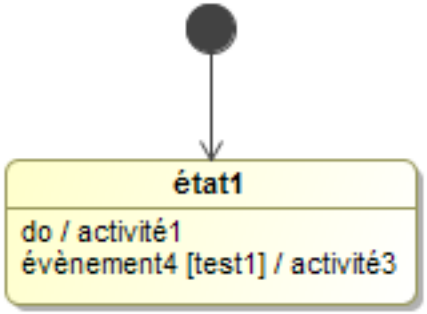
Mots réservés sur évènement (dans ligne de commande placée dans un état)

`entry`	Ex : `entry /` activité1	Le front `entry` apparaît à l'activation de l'état.
`do`	Ex : `do /` activité2	L'activité2 s'exécute tant que l'état est actif après l'activité1.
`exit`	Ex : `exit /` activité3	Le front `exit` apparaît à la désactivation de l'état. **L'activité 2 est interrompue**.

Les évènements `entry`, `do` ou `exit` ne peuvent être utilisés qu'une seule fois par état.
Exceptionnellement, les activités associées à `entry` et `exit` ne peuvent pas être interrompues comme des actions.
Il n'y a jamais de conditions de garde avec les mots réservés `entry`, `do` et `exit`.

Transitions internes dans un état

- Il est possible de placer des activités dans un état associées à un évènement et éventuellement une garde.
- Ces activités s'effectuent en parallèle de l'activité associée à un `do`.



Lorsque l'état1 est actif, l'activité1 est exécutée.
Dès que l'évènement4 apparaît avec la condition de garde test1 vraie, alors l'activité3 s'exécute aussi.

Dans la logique SED enseignée (machine de Moore), il est déconseillé d'utiliser ces transitions internes : pour nous à chaque apparition d'un évènement , on change d'état.

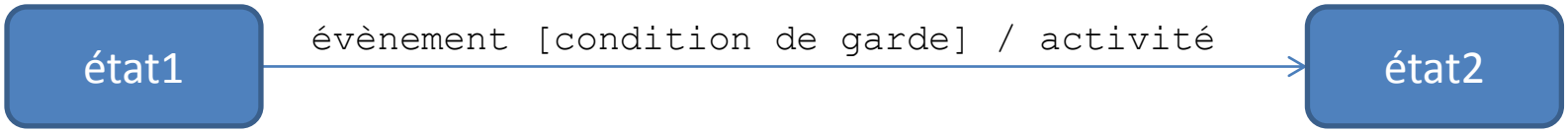
Pseudo-états

Un pseudo-état est un état anonyme qui ne contient aucune ligne de commande et **qui possède un comportement spécifique.**

	Pseudo-état initial	Il s'agit d'un état activé au lancement de la machine à états. Il est unique par machine d'états. Il n'a donc aucune transition entrante.
	Pseudo-état final	Lorsque cet état est activé, l'automate fini s'arrête. Il n'a donc aucune transition sortante. Il est optionnel.

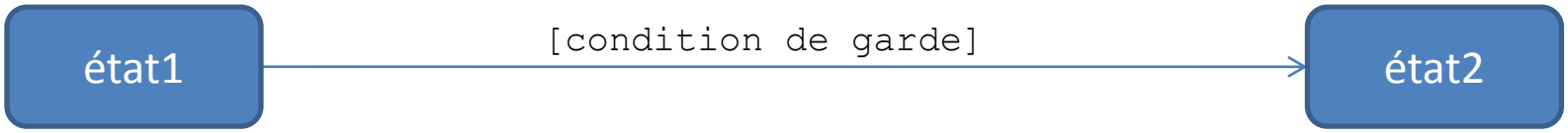
Transitions

- **Transition** : arc orienté qui relie deux états (source vers cible).
- Une transition possède une unique ligne de commande :



- Si l'événement apparaît (front montant du déclencheur) et si la condition de garde est vraie à ce moment alors la transition est franchissable : si l'état1 était actif alors il se désactive et l'état2 s'active. L'activité, si elle existe, est exécutée avant celles situées dans l'état2.
- La condition de garde n'est prise en compte qu'au moment de l'occurrence de l'évènement.
- Si aucune condition de garde est écrite alors elle est toujours vraie.
- Si plusieurs évènements au choix sont écrits, ils sont séparés par une virgule.

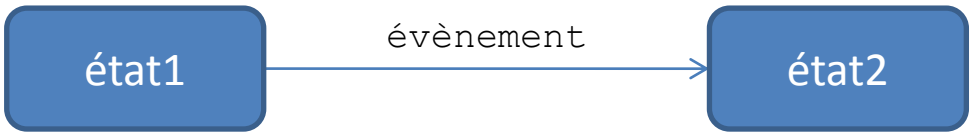
Transitions automatiques



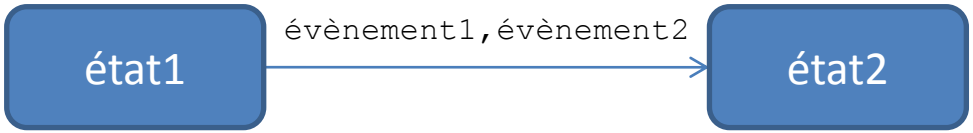
- Une transition automatique (completion transition) est une transition sans évènement explicite.
- L'évènement associé est :
  - La fin de l'activité `do` de l'état1, ou
  - La fin de l'activité `entry` de l'état1 si l'activité `do` n'existe pas, ou
  - L'activation de l'état 1 si celui-ci ne contient aucune activité. (solution privilégiée dans ce cas)

- Rappel : si la condition de garde n'existe pas, alors elle est considérée comme toujours vraie.
- ATTENTION : Si l'activité de l'état1 ne s'arrête jamais (allumage d'un voyant par ex.) alors l'automate est bloqué !

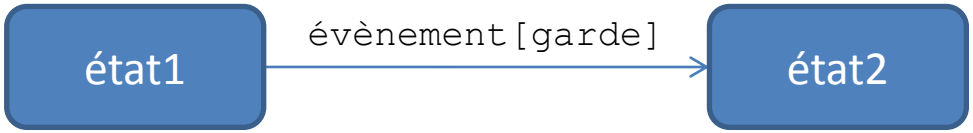
Transitions



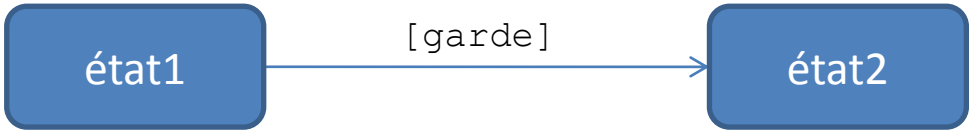
Dès que l'évènement apparaît, l'état1 se désactive, et l'état2 s'active.



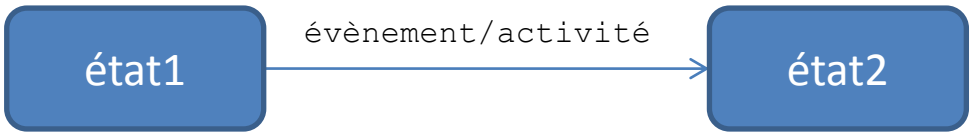
Dès que, soit l'évènement1, soit l'évènement2 apparaît, l'état1 se désactive, et l'état2 s'active.



Dès que l'évènement apparaît et que la garde est vraie (booléen), l'état1 se désactive, et l'état2 s'active.



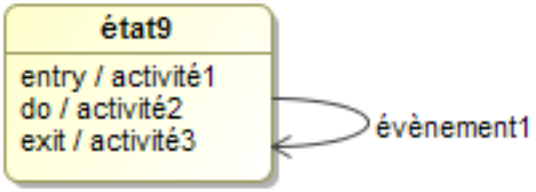
Transition automatique avec garde : l'évènement associé est soit l'activation de l'état1 s'il ne contient pas d'activité, soit la fin de entry ou do activité. (voir synchronisation d'état)



Dès que l'évènement apparaît, l'état1 se désactive, l'activité s'exécute puis l'état2 s'active. (déconseillé)

Transitions

- **Transition réflexive** : arc orienté qui relie un même état (source = cible).
- Cela permet d'exécuter à nouveau les activités associées à `exit` et à `entry`.



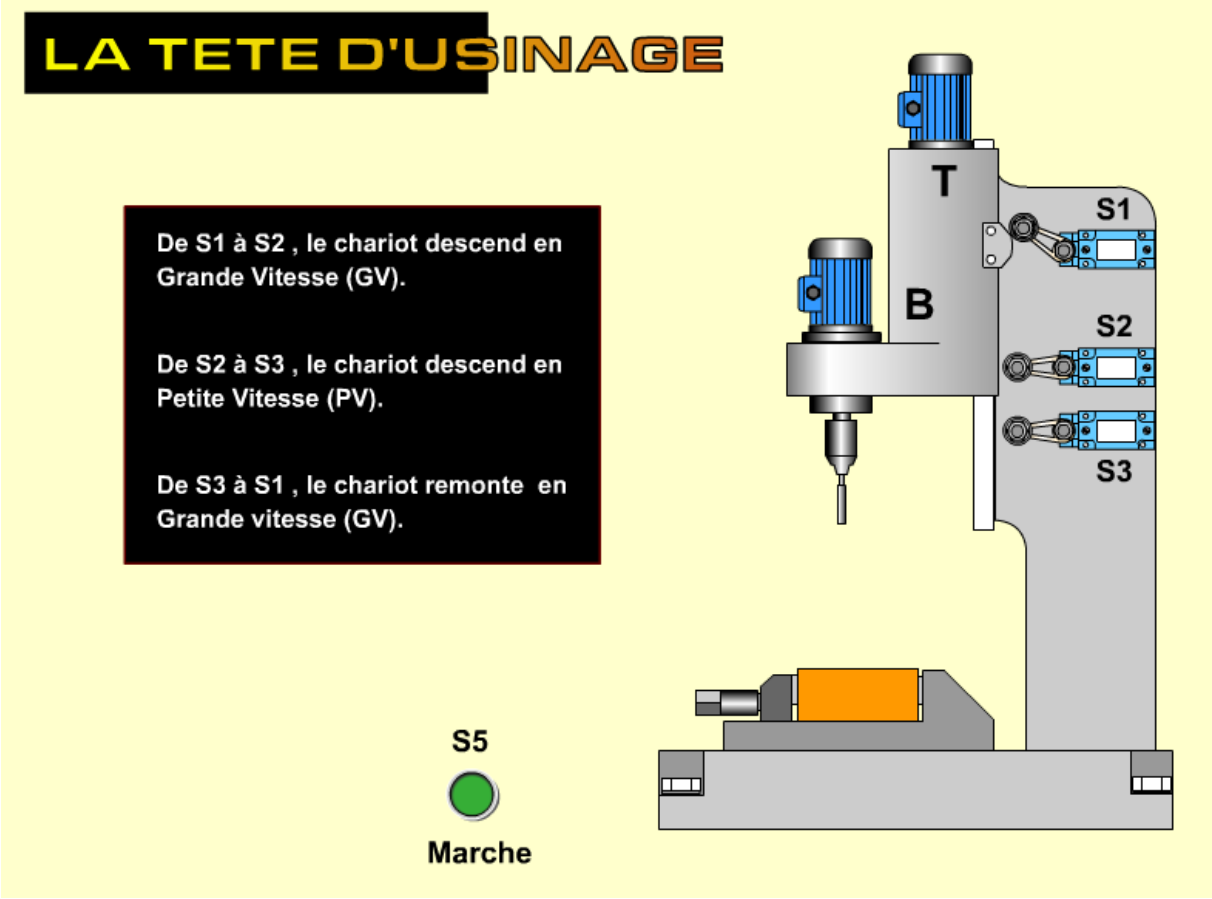
Si l'état9 est actif, il est sensé exécuter l'activité2 sauf si celle-ci s'est terminée d'elle-même.
Si évènement1 apparait, l'état9 se désactive et se réactive aussitôt donc l'activité2 est interrompue, l'activité3 est exécutée puis lorsqu'elle est terminée, l'activité1 se réalise et enfin de nouveau l'activité2.
Noter que l'évènement implicite d'entrée dans l'état courant apparaît. (voir transition automatique, ou évènement temporel, on relance les temporisations)

Rappel : les activités associées à `entry` et `exit` ne peuvent pas être interrompues à l'instar des actions.

Exemple

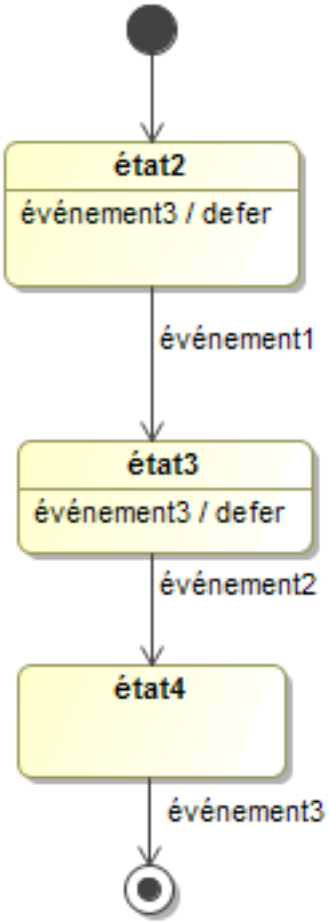
Décrire les évènements et les activités associés.

Décrire par un diagramme d'états (stm) le fonctionnement de cette machine de production :



Evénements référencés

- Il est possible de créer une liste de déclencheurs (triggers) de différents évènements pouvant apparaître lorsqu'un état est actif mais utiles dans une transition ou un autre état « plus tard ».
- Pour mettre en file d'attente ces déclencheurs il faut utiliser l'action mot réservé `/defer`.



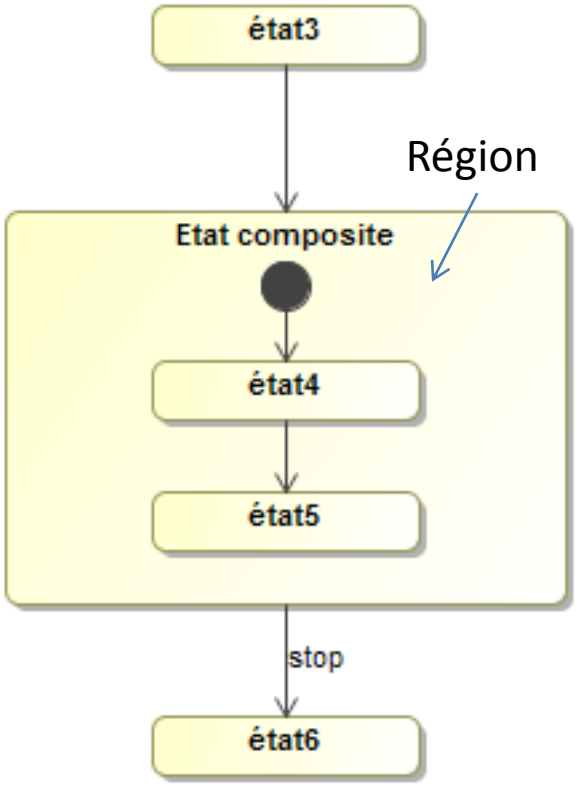
A l'activation de la machine d'états, l'état2 est activé (via le pseudo-état initial et la transition automatique) ;
Dès que l'occurrence de l'événement1 apparaît, on désactive l'état2 et on active l'état3.
Les états 2 et 3 contiennent une action `/defer` associée à l'événement3 si cet événement3 apparaît pendant que soit l'état2 soit l'état3 est actif, alors il est référencé (trigger pool) et en attente de consommation !

Par exemple si événement2 apparaît, l'état3 se désactive et l'état4 s'active, mais comme événement3 était référencé, alors on désactive l'état4 et on atteint le pseudo-état final. Si l'état4 contient des actions/activités associées à `entry` et/ou `exit`, elles sont exécutées (mais pas l'activité associée à do).
Si événement3`/defer` n'était pas dans état3, alors l'événement3 serait perdu au cas ou il apparaît pendant l'activation de état2.

Etats composites

Etat composite : état contenant un automate fini détaillant son fonctionnement séquentiel.

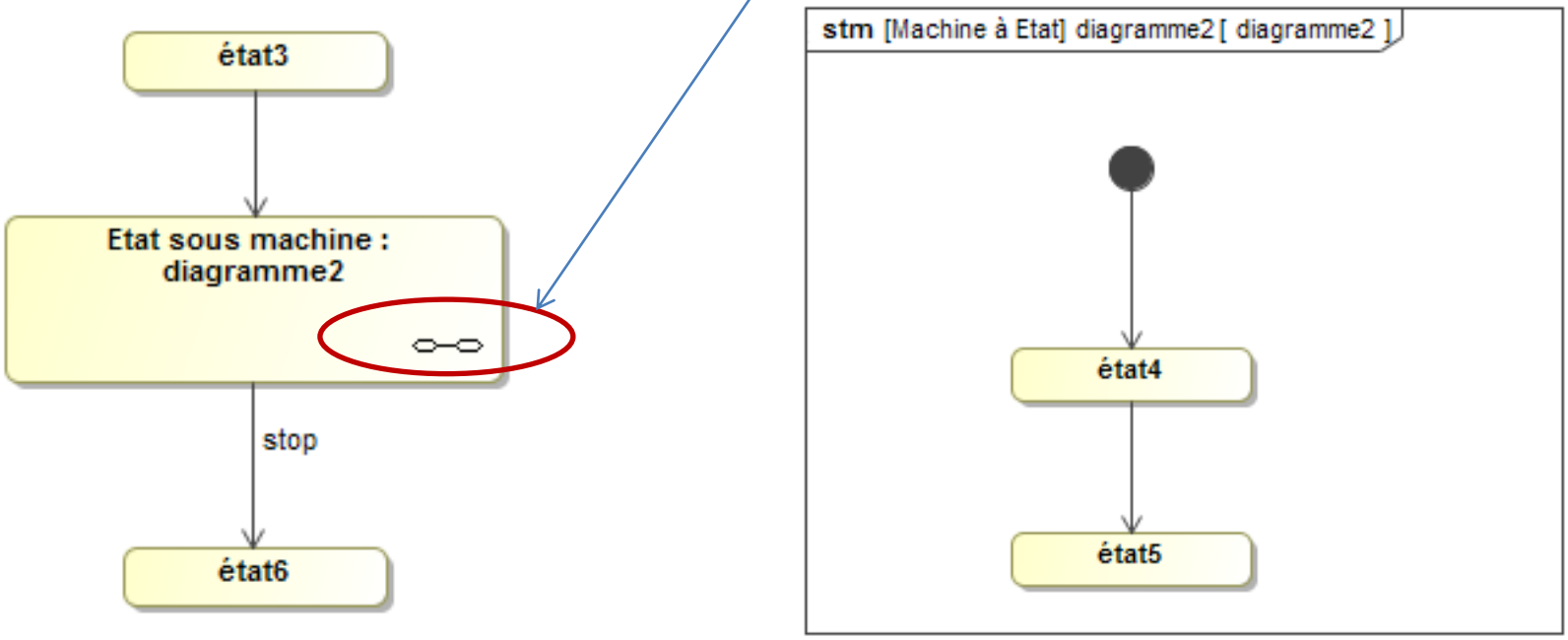
- L'activation de l'état composite entraîne l'activation du pseudo-état initial. (rond noir)
- La désactivation de l'état composite (ici évènement stop) entraîne la désactivation de l'état actif (ici état4 ou état5) : l'état composite est donc hiérarchiquement supérieur à l'automate fini qu'il contient.
- Chaque « sous-état » peut aussi être un état composite et ainsi de suite…
- L'automate fini contenu est appelée **région**.
- Un état composite peut contenir des activités associées à `entry`, `do` et `exit`. `entry` est traité avant l'activation du pseudo-état initial de la région, `do` est traité pendant, `exit` est traité au moment de la désactivation de l'état composite. (Je déconseille d'utiliser ces fonctionnalités.)



Sous machine à états

État sous machine : état instance d'un automate fini (diagramme d'états).

- Le diagramme d'états d'un état sous machine est dessiné dans un autre diagramme d'états : un pictogramme représentant 2 états + une transition permet de le signaler.



Ces diagrammes sont équivalents à l'exemple de la page précédente.

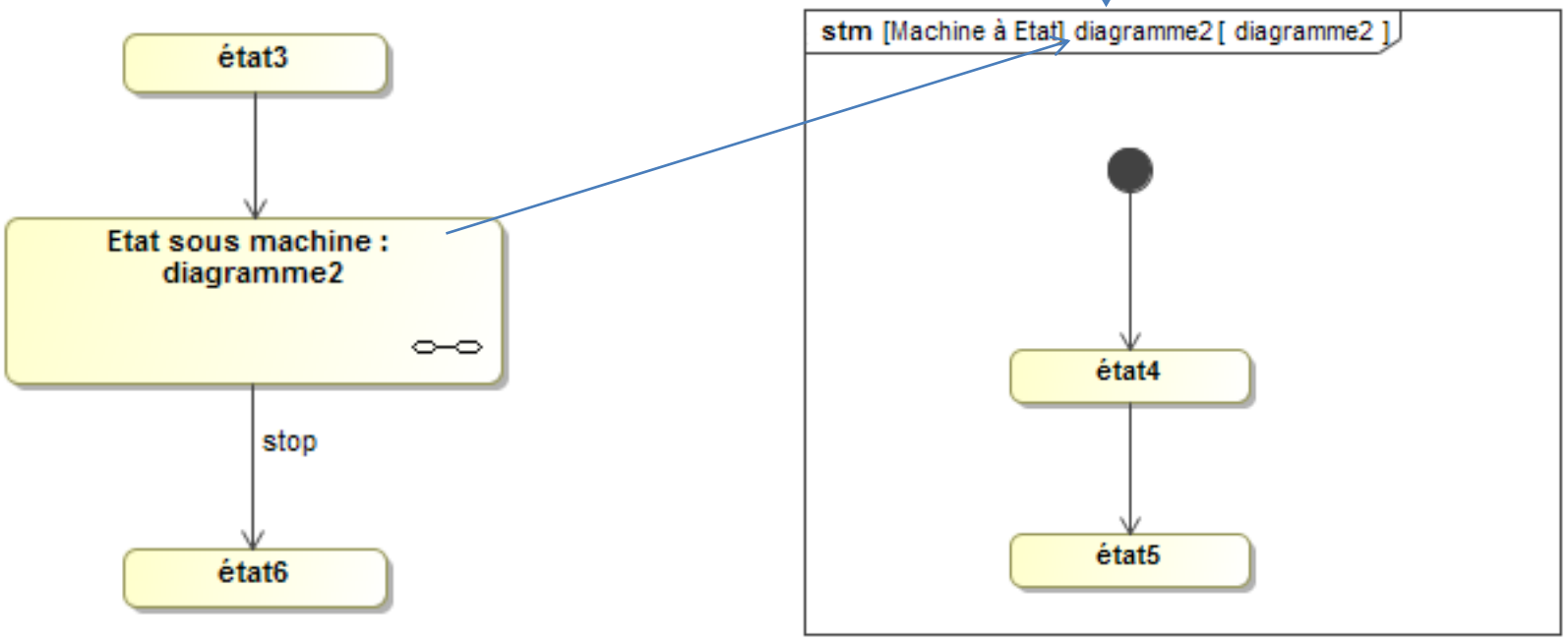
Sous machine à état

Etat sous machine 2

- Il est possible d'utiliser la même machine à états dans d'autres états sous-machine : dans ce cas le nom devient

  Nom de l'état sous machine : diagramme2

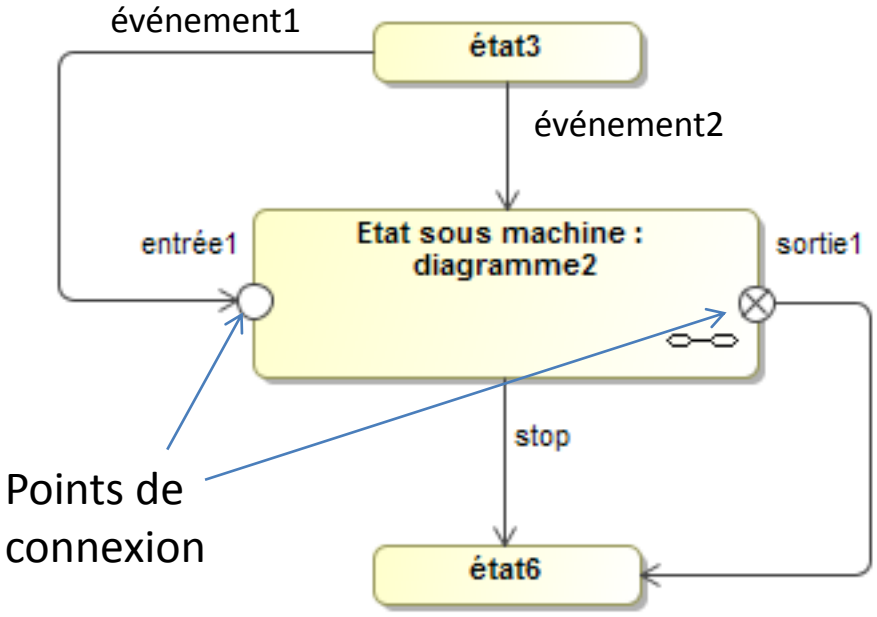
- On peut donc considérer le diagramme2 comme « un sous programme ».

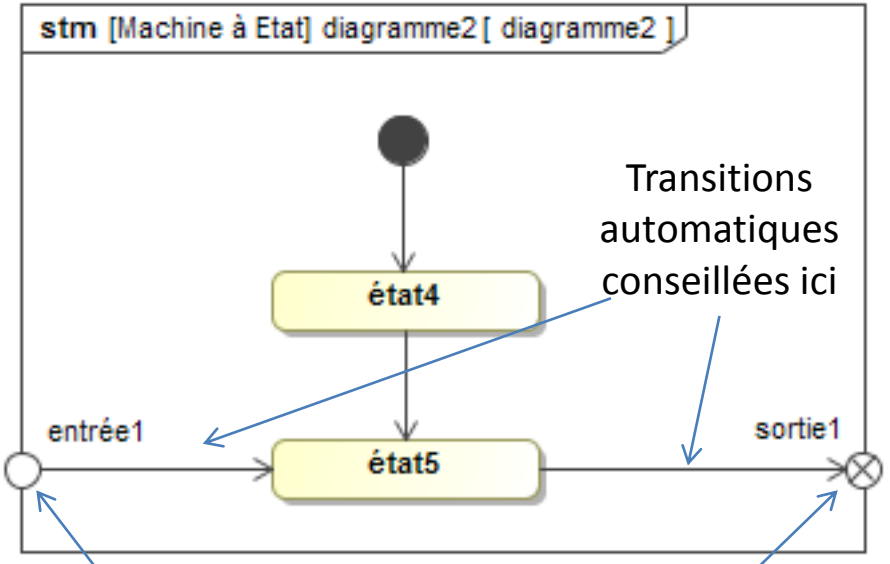


Sous machine à états

Pseudo-états d'entrée et de sortie

- Lorsque plusieurs points d'entrée et/ou sortie sont utiles dans une sous machine, il faut placer des pseudo-états correspondants :





C'est le nom (ex : entrée1) du point de référence qui fait le lien avec le pseudo-état point d'entrée de la sous machine.

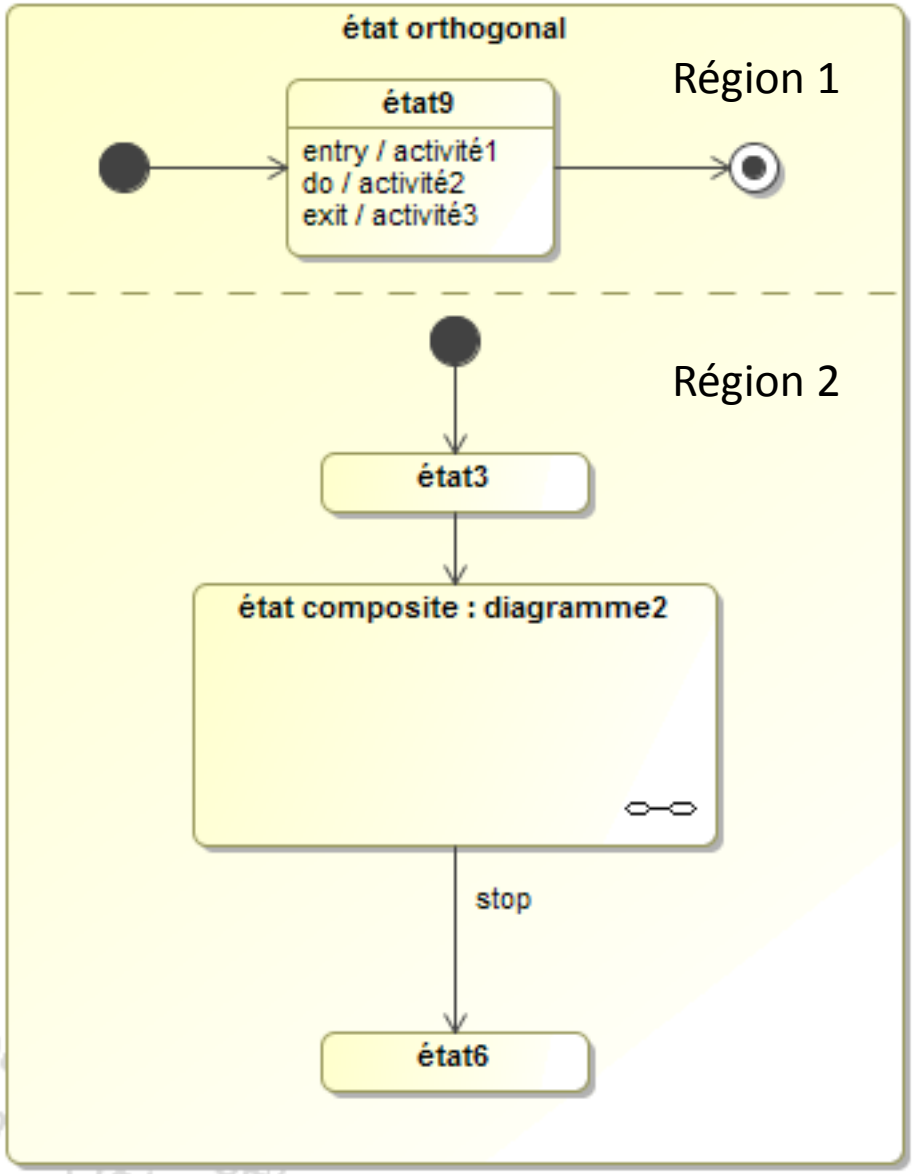
Pseudo-état d'entrée

Pseudo-état de sortie

États orthogonaux

État orthogonal : état contenant plusieurs automates finis (régions) fonctionnant en parallèle.

- Des droites en traits pointillés dans l'état créent des régions.
- Il est déconseillé de relier des états de deux régions distinctes (d'un même état composite)

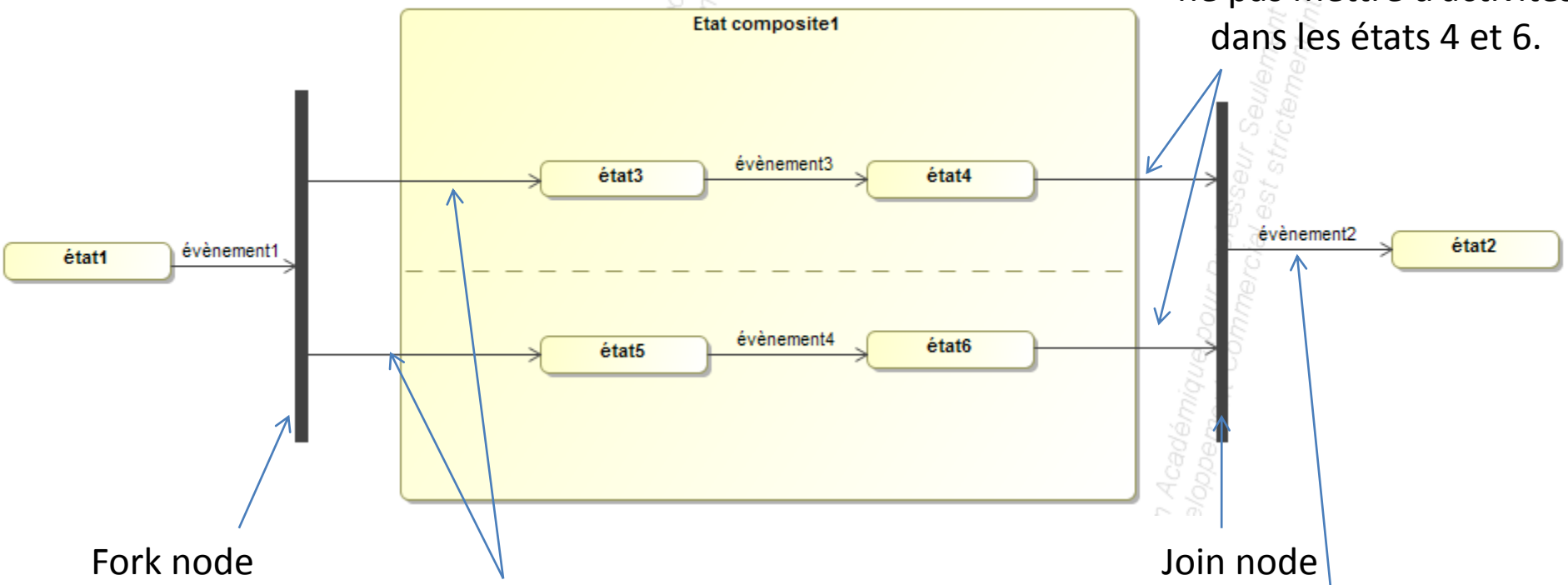


Magicdraw impose des régions horizontales, mais la norme autorise aussi une écriture verticale.

États orthogonaux

Pseudo états divergence (fork) et convergence (join) de flux.
Permet de représenter graphiquement **le parallélisme** entre des régions d'un même état composite.

Transitions automatiques : conseil ne pas mettre d'activités dans les états 4 et 6.



Fork node

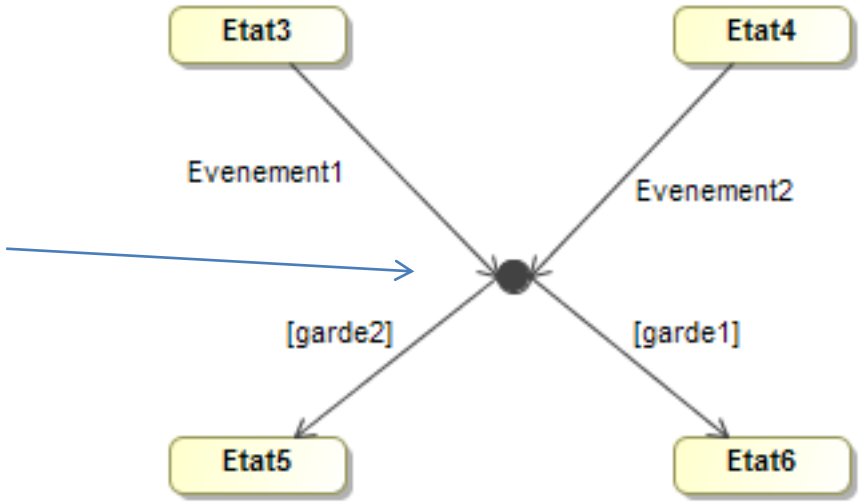
Join node

Les deux transitions automatiques sont franchies de manière synchrone.

Dès l'apparition de événement2, les état4 et état6 sont désactivés.

Pseudo-état de jonction

Pseudo-état
de jonction

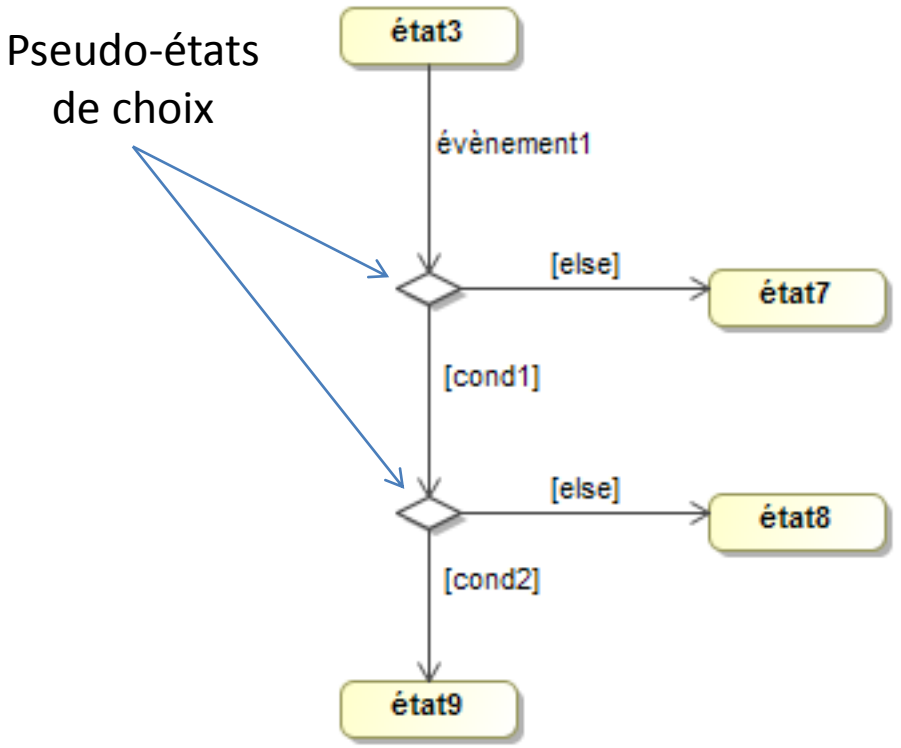


Le pseudo état de jonction (junction pseudo-state) permet de regrouper des conditions de franchissement de transitions en particulier des gardes communes à un évènement.
IMPORTANT : le test des gardes est réalisé au moment de l'apparition des évènement1 ou évènement2.

Exemple : on passe de l'état3 à l'état6 si l'évenement1 est apparu et que la garde1 est vraie.

Pseudo-états de choix (déconseillé)

Dès que l'événement1 apparaît, l'état3 est désactivé. Le 1er pseudo-état est atteint : les gardes sont évaluées à ce moment. Et ainsi de suite…
La différence entre le pseudo état de choix et celui de jonction apparait si des activités sont placées dans les transitions. (déconseillé)



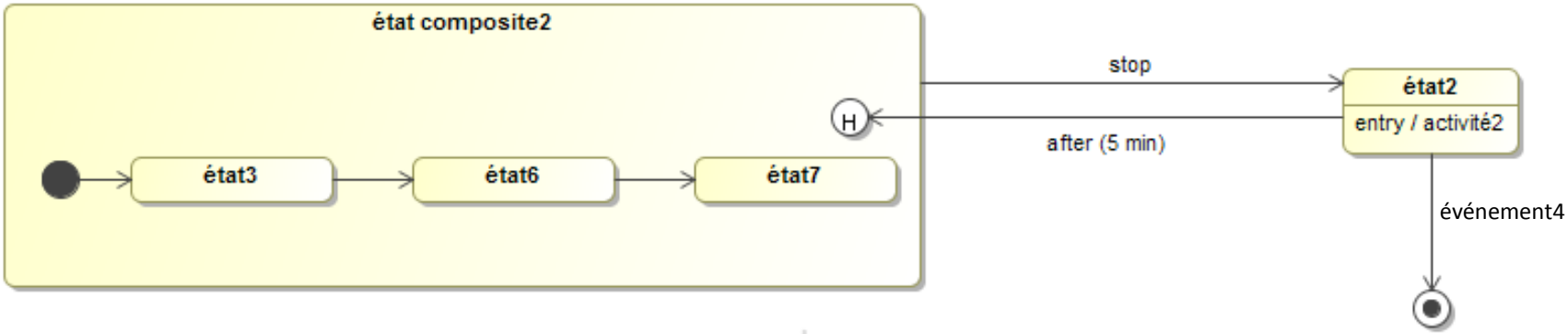
SysML/UML impose un choix exclusif entre les transitions pour obtenir un unique état actif par automate fini (ou par niveau hiérarchique). Il est donc conseillé de placer une condition de garde `[else]` pour ne pas bloquer l'évolution de l'automate fini. (on ne doit pas rester bloqué dans le pseudo-état de choix (losange)).

`else` est un mot réservé de garde.

Pseudo-états historiques

H	Pseudo-état historique (shallow history)	Cet état est associé à un état composite. Il permet de mémoriser l'état qui était actif au même niveau hiérarchique au moment où l'état composite a été désactivé. Lorsque ce pseudo-état devient actif, on reprend donc là où on en était dans l'état composite.
H*	Pseudo-état historique profond (deep history)	Similaire au précédent, mais réactive les états qui étaient actifs quelque soit leur niveau hiérarchique.

Une région ne peut avoir qu'un seul pseudo-état initial ou historique à la fois.



Lorsque stop devient vrai, l'état composite2 se désactive et l'automate fini mémorise l'état courant (par exemple l'état6).
Après 5min d'activation de l'état2, l'état composite2 est à nouveau activé, mais par le biais du pseudo-état historique : c'est donc l'état6 (mémorisé) qui s'active.

Conseils diagrammes d'états

- Bien vérifier qu'un unique état est actif à la fois par machine d'états.
- Il faut éviter de placer des activités dans la ligne exit.
- Il faut éviter de placer des activités dans les transitions (machine de Mealy).
- Il faut éviter les transitions automatiques sauf pour certains pseudo états (join, fork and initial pseudostates).
- Normalement il n'y a jamais un très grand nombre d'états par automate, ne pas hésiter à utiliser les états composites.
- Les états des machines de production au niveau hiérarchique supérieur se limite à des états du type : système en attente, système en production, système en phase de réglage, etc…