Les systèmes à évènements discrets (SED)

Largement inspiré du cours d'Olivier Gallo

http://olivier.legallo.pagesperso-orange.fr/documents/SED.pdf

Diagramme d’état : State machine diagramm (stm)

Les systèmes à événements discrets (SED) se définissent, par opposition aux systèmes continus dont l'évolution est continue dans le temps, comme des **systèmes dont les évolutions ont lieu à des instants précis.**

Dans ces systèmes, le passage d’un état à un autre est **déclenché par des événements ponctuels**.

Diagramme d’état : State machine diagramm (stm)

Les SED sont le plus souvent séquentiels, c’est-à-dire que la ou les sorties dépendent de la combinaison des entrées mais aussi de l’état du système lui-même.

Diagramme d’état : State machine diagramm (stm)

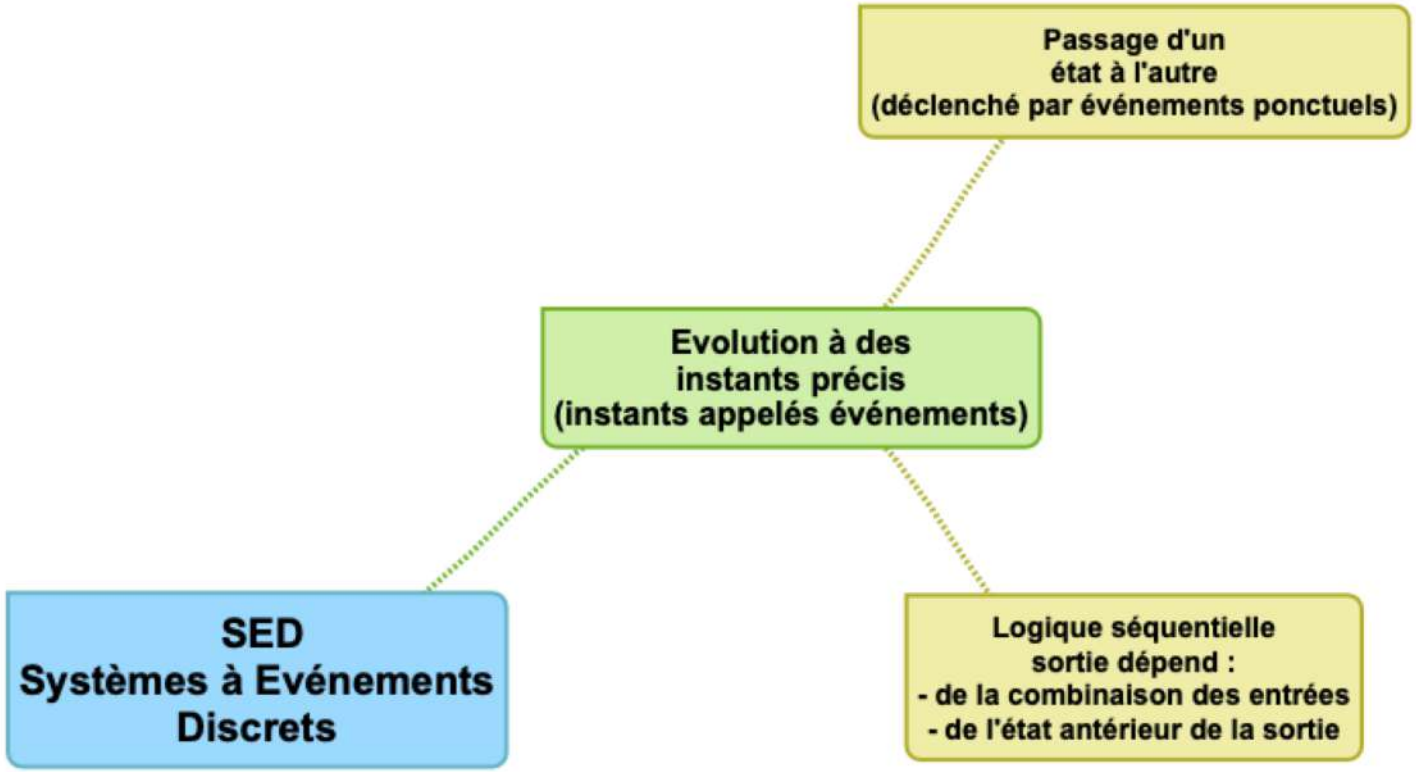


Diagramme d’état : State machine diagramm (stm)

Exemple : télécommande de téléviseur

En fonction de l’état du téléviseur au moment de l’appui sur le bouton cela entraine soit l’allumage ou soit l’extinction du téléviseur.

Diagramme d’état : State machine diagramm (stm)

Exemple : télécommande de téléviseur

En fonction de l’état du téléviseur au moment de l’appui sur le bouton cela entraine soit l’allumage ou soit l’extinction du téléviseur.

Ainsi, dans la cas d’un SED séquentiels :

- une même cause (même combinaison des entrées) peut produire des effets différents ;
- le temps peut être une cause déclenchante ;
- l'effet peut persister si la cause disparaît.

Diagramme d’état : State machine diagramm (stm)

En langage SysML, un **diagramme d’état (stm) est nécessairement associé à un bloc** du diagramme de définitions de blocs (bdd) ou du diagramme de blocs internes (ibd). Ce bloc peut être le système, un sous-système ou un composant.

Diagramme d’état : State machine diagramm (stm)

En langage SysML, **un diagramme d’état (stm) est nécessairement associé à un bloc** du diagramme de définitions de blocs (bdd) ou du diagramme de blocs internes (ibd). Ce bloc peut être le système, un sous-système ou un composant.

Le diagramme d’état décrit les différents états pris par le bloc ainsi que les transitions possibles entre ces différents états.

Diagramme d’état : State machine diagramm (stm)

Exemple : télécommande de téléviseur

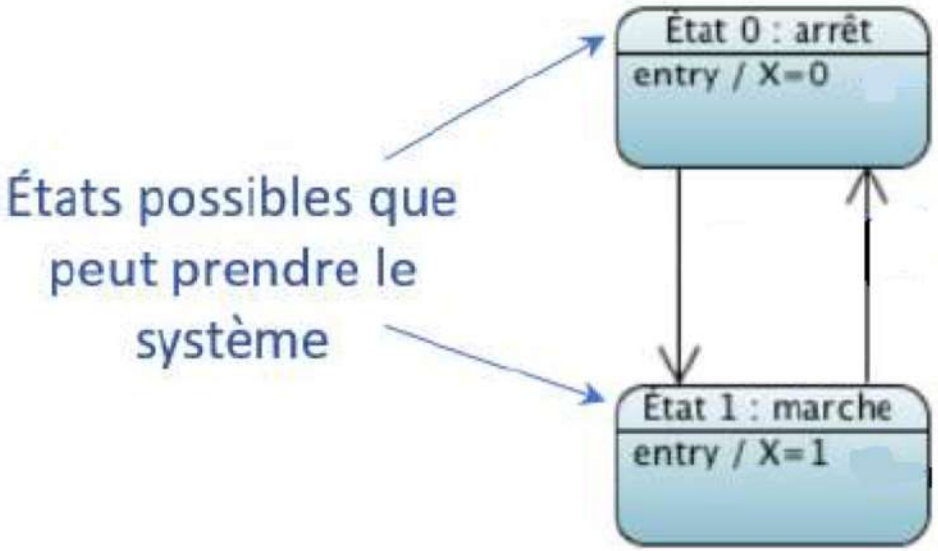


Diagramme d’état : State machine diagramm (stm)

Exemple : télécommande de téléviseur

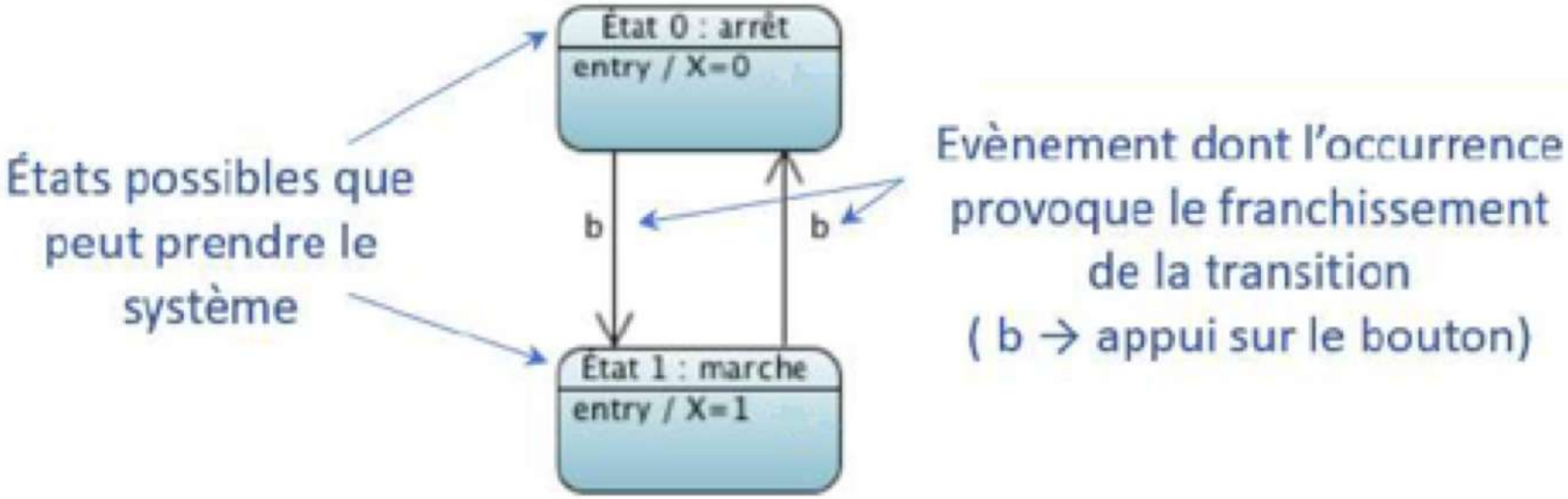


Diagramme d’état : State machine diagramm (stm)

Exemple : télécommande de téléviseur

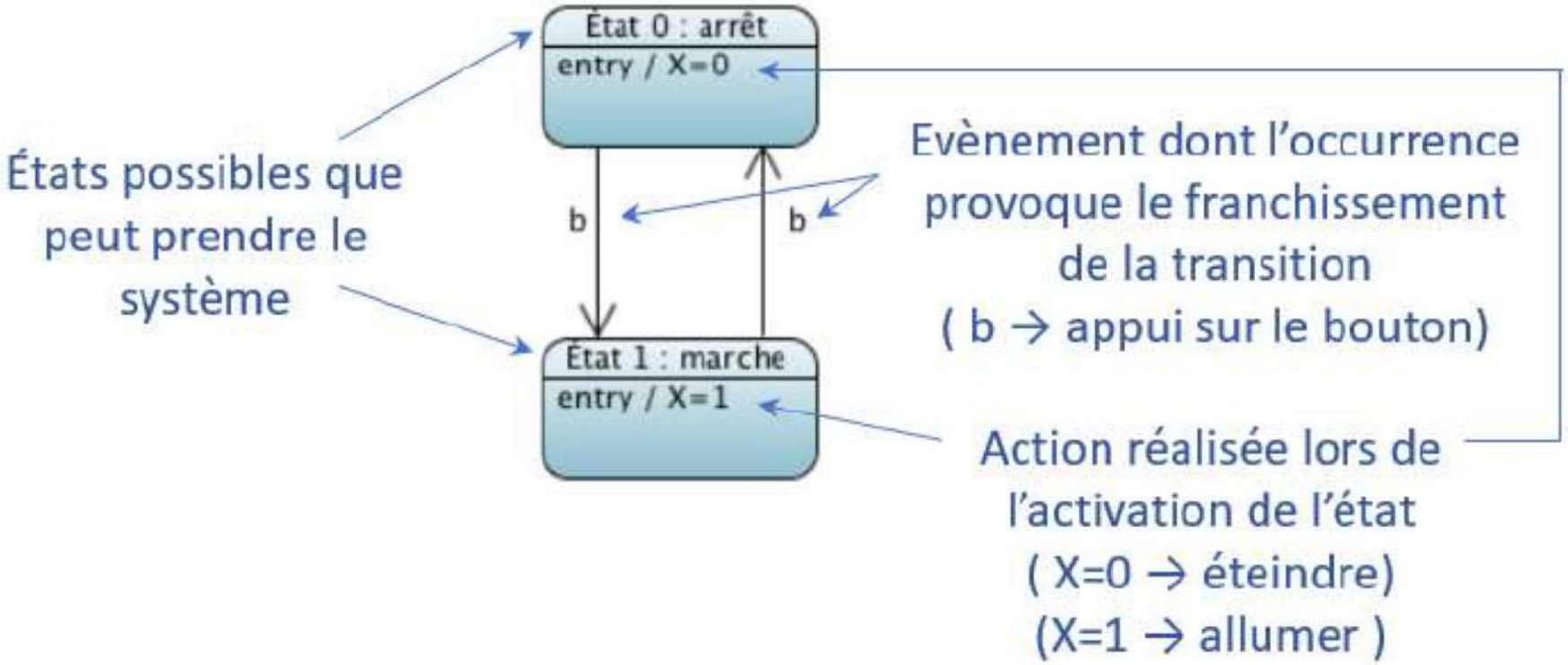


Diagramme d’état : State machine diagramm (stm)

Etats intermédiaires et activités associées

Un état intermédiaire modélise une phase du fonctionnement du système.

Pendant cette période, l'**état** est dit **actif** et le **système accomplit** :

- une simple activité ;
- OU une séquence d’activités ;
- OU est en attente.

En dehors de cette période, l'**état** est dit **inactif.**

Diagramme d’état : State machine diagramm (stm)

Etats intermédiaires et activités associées

Un état intermédiaire modélise une phase du fonctionnement du système.

Pendant cette période, l'état est dit actif et le système accomplit :

- une simple activité ;
- OU une séquence d’activités ;
- OU est en attente.

En dehors de cette période, l'état est dit inactif.

Par définition :

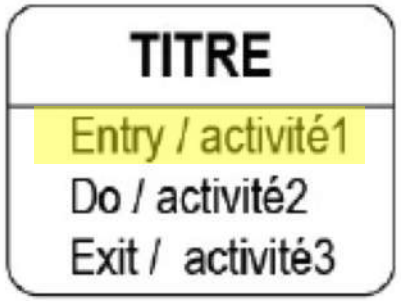
- il n’y a qu’un seul état actif à chaque instant ;
- un état possède un titre unique dans le diagramme.

Diagramme d’état : State machine diagramm (stm)

Etats intermédiaires et activités associées

Le lancement des activités à l'intérieur de l'état actif est organisé selon des **mots réservés :**

entry Activité ayant une fin* , elle ne peut pas être interrompue. Elle est exécutée lors de l’activation de l’état.



* En général une activité instantanée :

- allumer un voyant,
- incrémenter un compteur,
- éteindre un voyant…

Diagramme d’état : State machine diagramm (stm)

Etats intermédiaires et activités associées

Le lancement des activités à l'intérieur de l'état actif est organisé selon des **mots réservés :**

entry	Activité ayant une fin , elle ne peut pas être interrompue. Elle est exécutée lors de l’activation de l’état.
do	Activités interruptibles. Elles sont exécutées dans l'ordre de leur écriture, à partir de l'instant où l’activité associée à entry est terminée.

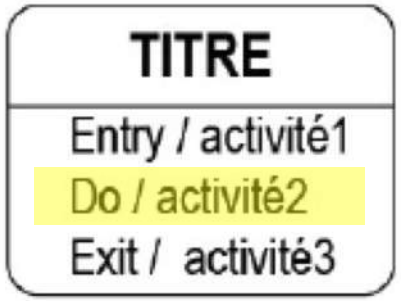
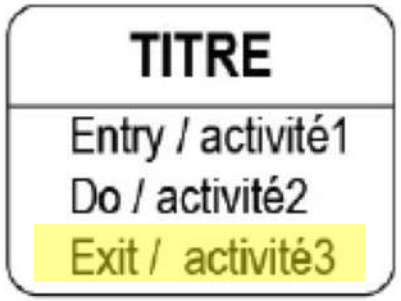


Diagramme d’état : State machine diagramm (stm)

Etats intermédiaires et activités associées

Le lancement des activités à l'intérieur de l'état actif est organisé selon des **mots réservés** :

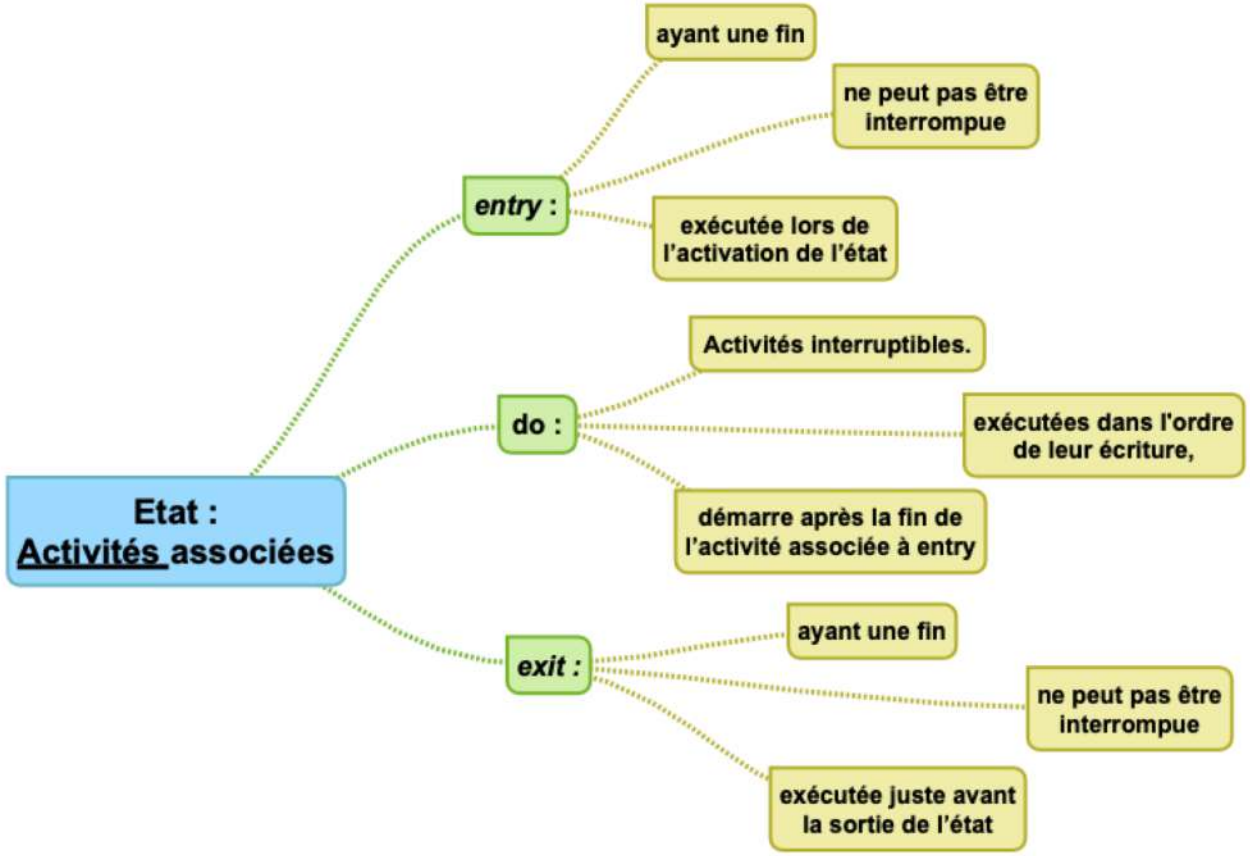
entry	Activité ayant une fin , elle ne peut pas être interrompue. Elle est exécutée lors de l’activation de l’état.
do	Activités interruptibles. Elles sont exécutées dans l'ordre de leur écriture, à partir de l'instant où l’activité associée à entry est terminée.
exit	Activité ayant une fin *, elle ne peut pas être interrompue. Elle est exécutée lors de la désactivation de l’état, juste avant la sortie de l’état



* En général une activité instantanée :

- allumer un voyant,
- incrémenter un compteur,
- éteindre un voyant…

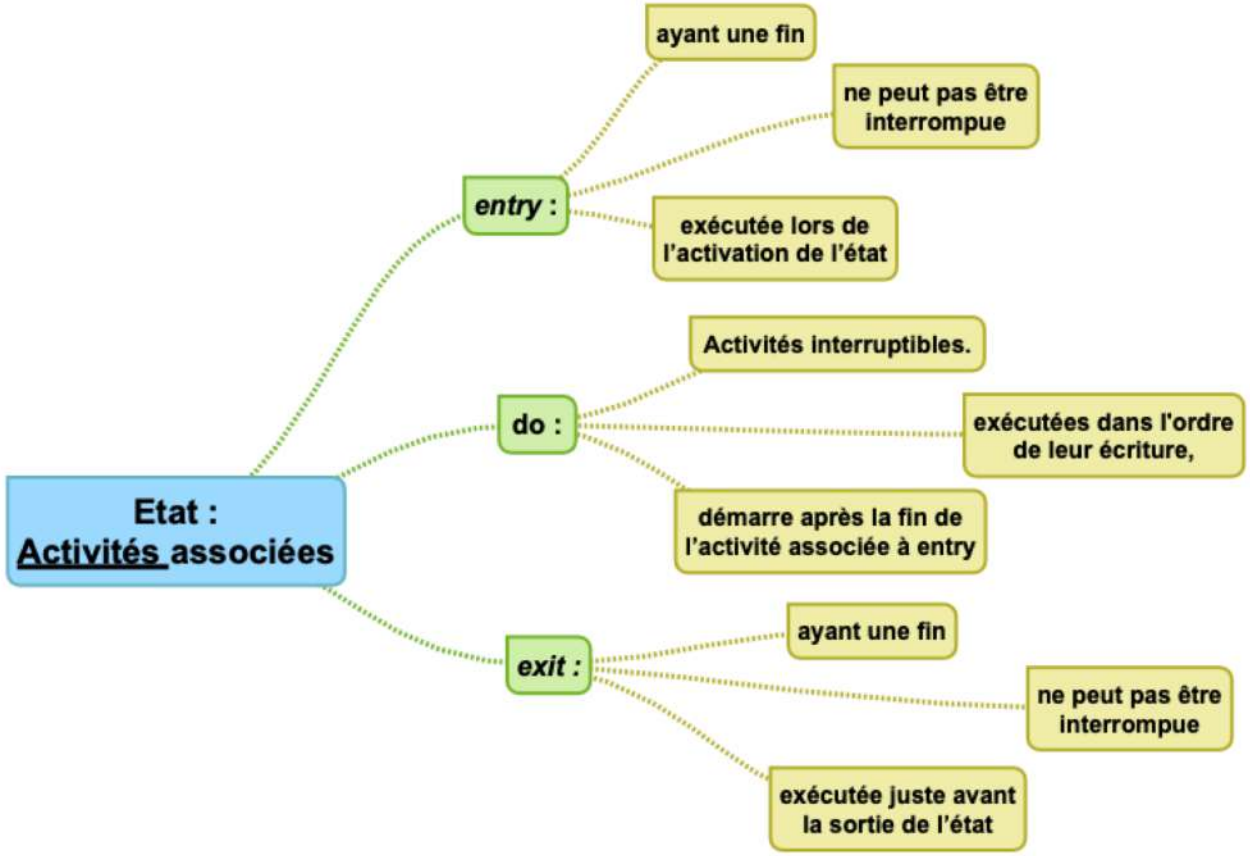
Diagramme d’état : State machine diagramm (stm)



Remarques :

- les trois comportements ***entry***, ***do*** et ***exit*** ne peuvent être **utilisés qu’une seule fois par état**, mais il est également possible de n’en **utiliser qu’une partie** (seulement *entry* par exemple) ;

Diagramme d’état : State machine diagramm (stm)



Remarques :

- si aucun mot réservé n'est utilisé, cela correspond à un ***do*** ;
- un **état vide** (sans activité) indique un **état d’attente** ;

Diagramme d’état : State machine diagramm (stm)

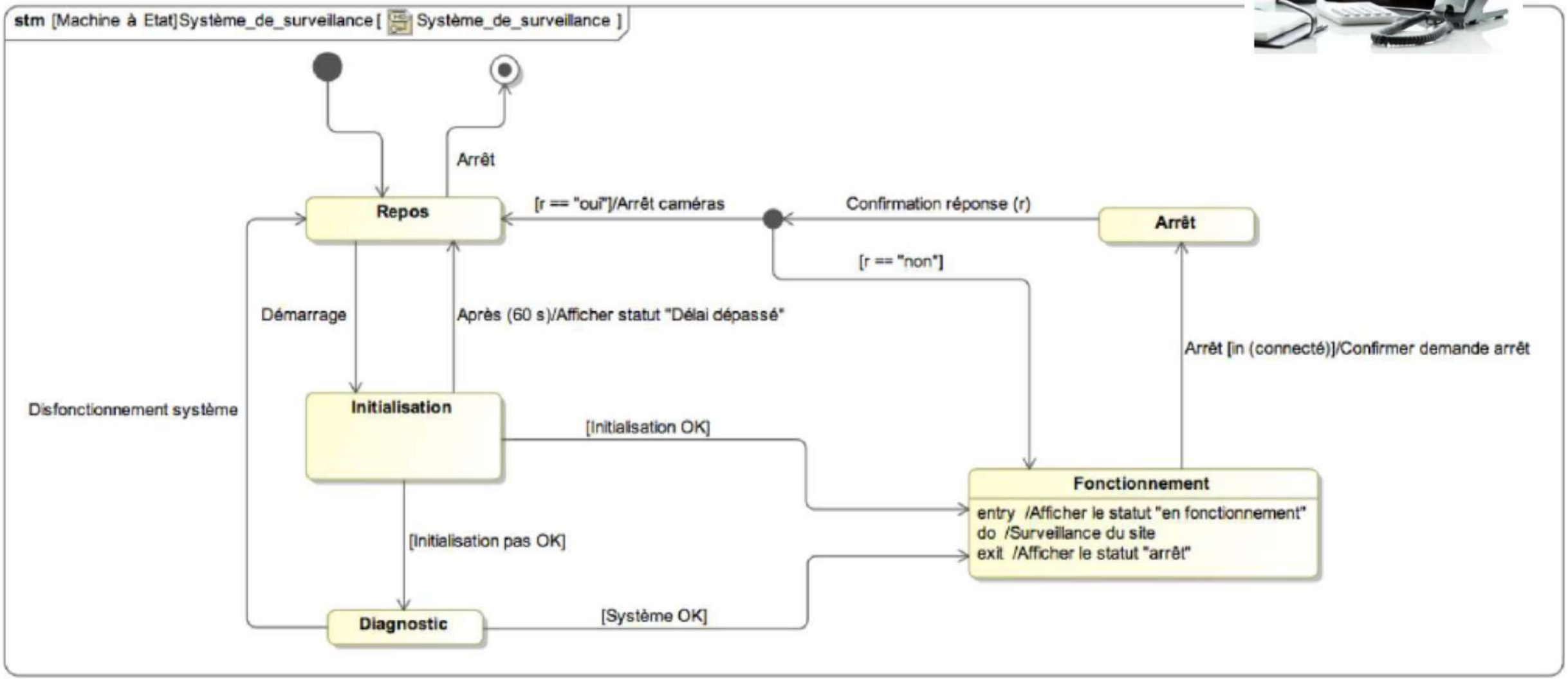
Etats intermédiaires et activités associées

Exemple : dispositif de vidéo surveillance

Le diagramme ci-après décrit le fonctionnement d’un système de vidéo-surveillance.

Exemple : dispositif de vidéo surveillance

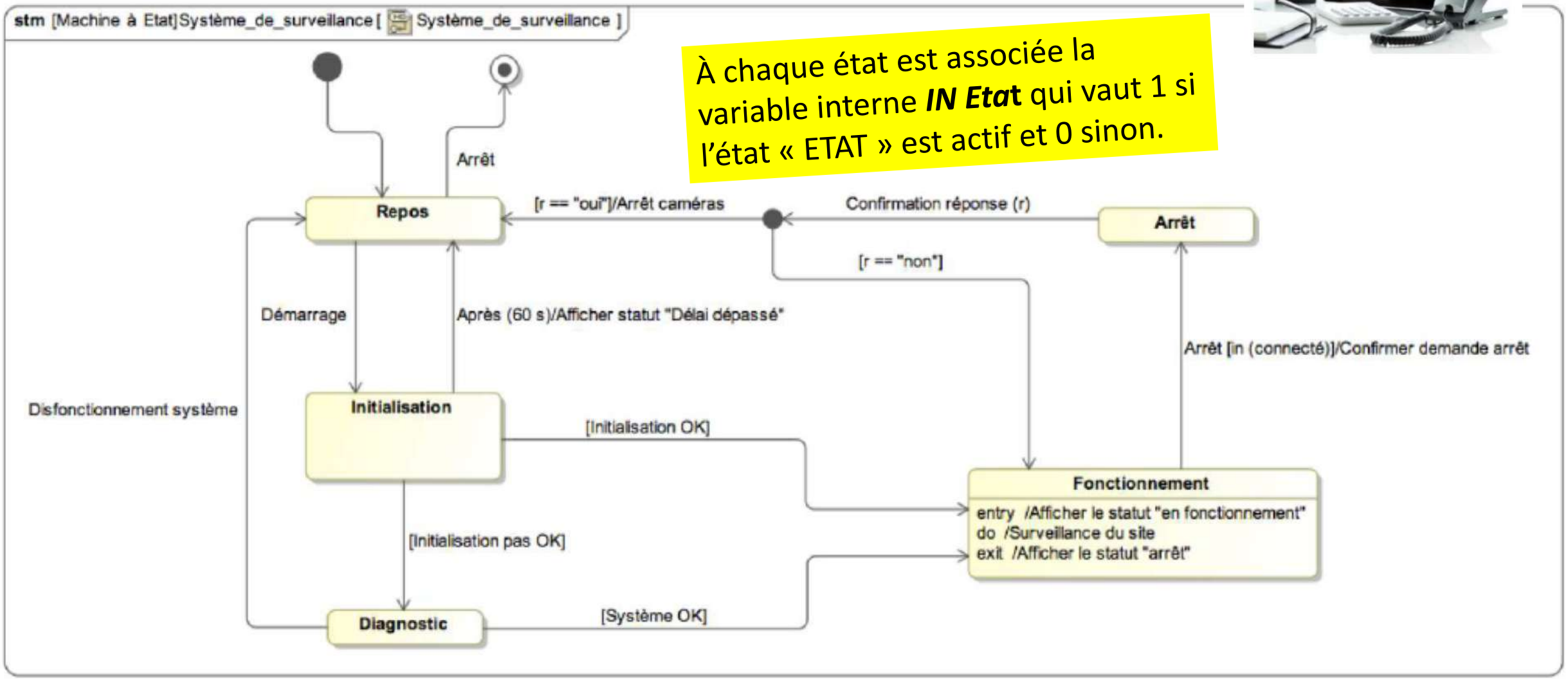
On trouve 5 états : Repos , Initialisation, Diagnostic, Arrêt et Fonctionnement.



Exemple : dispositif de vidéo surveillance

On trouve 5 états : Repos , Initialisation, Diagnostic, Arrêt et Fonctionnement.

À chaque état est associée la variable interne ***IN Etat*** qui vaut 1 si l'état « ETAT » est actif et 0 sinon.



Evolution d’un diagramme d’état par franchissement des transitions

Une **transition** modélise la possibilité d'un passage instantané d'un **état** vers un **autre**.
On appelle **état source** l'état de départ d'une transition, et **état cible** l'état d'arrivée d’une transition.

La **transition** :

- n'a **pas de durée** ;
- n'est **évaluée** que **si l'état source est actif**.

Son **franchissement** est **conditionné** par des **événements déclencheurs** et des **conditions de garde.**

Evolution d’un diagramme d’état par franchissement des transitions

Une **transition** modélise la possibilité d'un passage instantané d'un **état** vers un **autre**.
On appelle ét**at source** l'état de départ d'une transition, et **état cib**le l'état d'arrivée d’une transition.

La **transition** :

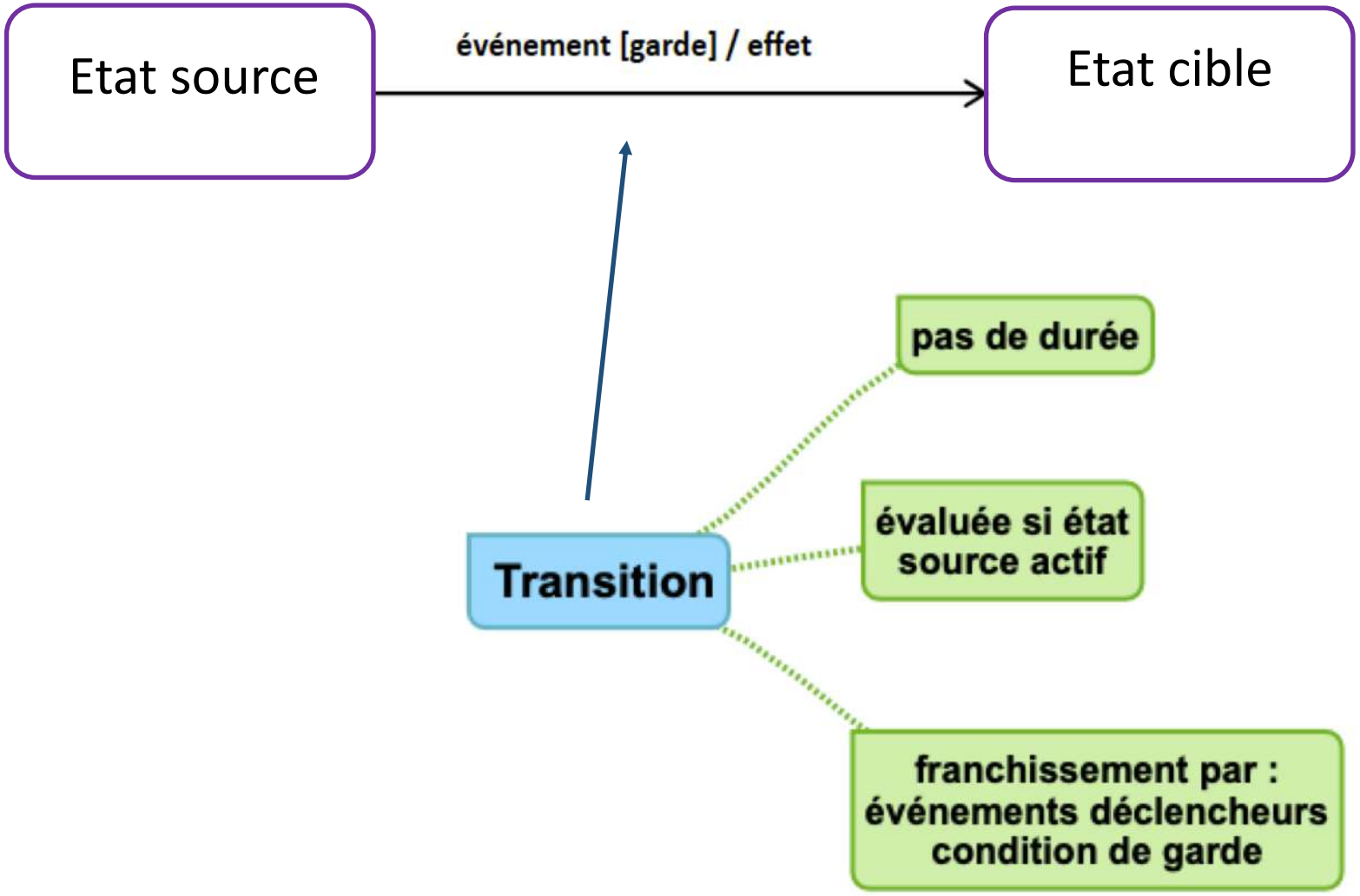
- n'a **pas de durée** ;
- n'est **évaluée** que **si l'état source est actif**.

Son **franchissement** est **conditionné** par des **événements déclencheurs** et des **conditions de garde.**

Ces événements et conditions de franchissement ainsi que l’éventuel effet associé à la transition sont indiqués le long de la flèche qui la symbolise suivant la notation :

événement [garde] / effet

Evolution d’un diagramme d’état par franchissement des transitions



Evolution d’un diagramme d’état par franchissement des transitions

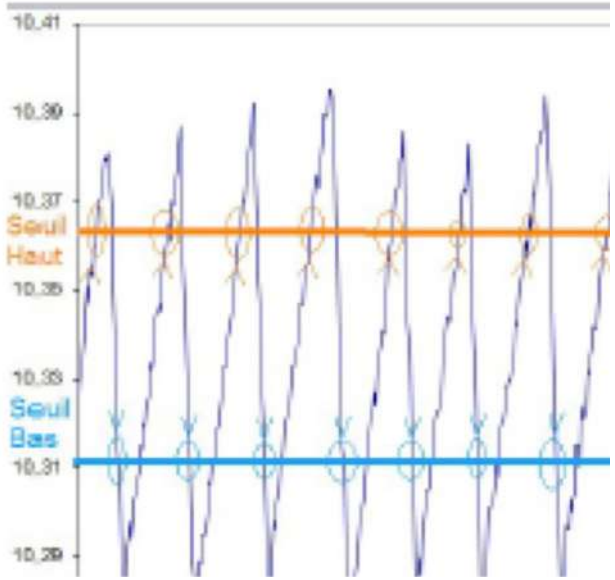
événement [garde] / effet

<u>Événement :</u>

Un **événement** correspond au **changement d’état** (supposé instantané) **d’une variable observée**. Il est daté dans le temps et il est **traité instantanément** lors de son occurrence (apparition).

Exemples :

- appui sur un bouton,
- arrivée en fin de course d’un mécanisme,
- Dépassement d’une valeur seuil…



Evolution d’un diagramme d’état par franchissement des transitions

<u>Événement :</u>

Un **événement** n’est **jamais mémorisé** et est donc **perdu** s’il ne mène à **aucune évolution** du diagramme d’état.

Il est possible d’utiliser des variables internes (compteurs ou horloge) pour spécifier un événement déclencheur :

When()	Se déclenche lors du changement d’état d’une valeur interne au diagramme d’état. Il permet par exemple d’utiliser un compteur : *when(N=3)* ou *when(N<10)*.

Evolution d’un diagramme d’état par franchissement des transitions

<u>Événement :</u>

Un **événement** n’est **jamais mémorisé** et est donc **perdu** s’il ne mène à **aucune évolution** du diagramme d’état.

Il est possible d’utiliser des variables internes (compteurs ou horloge) pour spécifier un événement déclencheur :

When()	Se déclenche lors du changement d’état d’une valeur interne au diagramme d’état. Il permet par exemple d’utiliser un compteur : *when(N=3)* ou *when(N<10)*.
after (T)	Se déclenche après une durée T passé dans l’état d’amont. Il permet de réaliser une temporisation.

Evolution d'un diagramme d'état par franchissement des transitions

événement [garde] / effet

<u>Événement :</u>

Un **événement** n'est **jamais mémorisé** et est donc **perdu** s'il ne mène à **aucune évolution** du diagramme d'état.

Il est possible d'utiliser des variables internes (compteurs ou horloge) pour spécifier un événement déclencheur :

When()	Se déclenche lors du changement d'état d'une valeur interne au diagramme d'état. Il permet par exemple d'utiliser un compteur : *when(N==3)* ou *when(N<10)*.
after (T)	Se déclenche après une durée T passé dans l'état d'amont. Il permet de réaliser une temporisation.
at(D)	Se déclenche à la date D dans un référentiel de temps dont l'origine correspond généralement au démarrage du fonctionnement du système.

Evolution d’un diagramme d’état par franchissement des transitions

événement [garde] / effet

<u>Événement :</u>

Un **événement** n’est **jamais mémorisé** et est donc **perdu** s’il ne mène à **aucune évolution** du diagramme d’état.

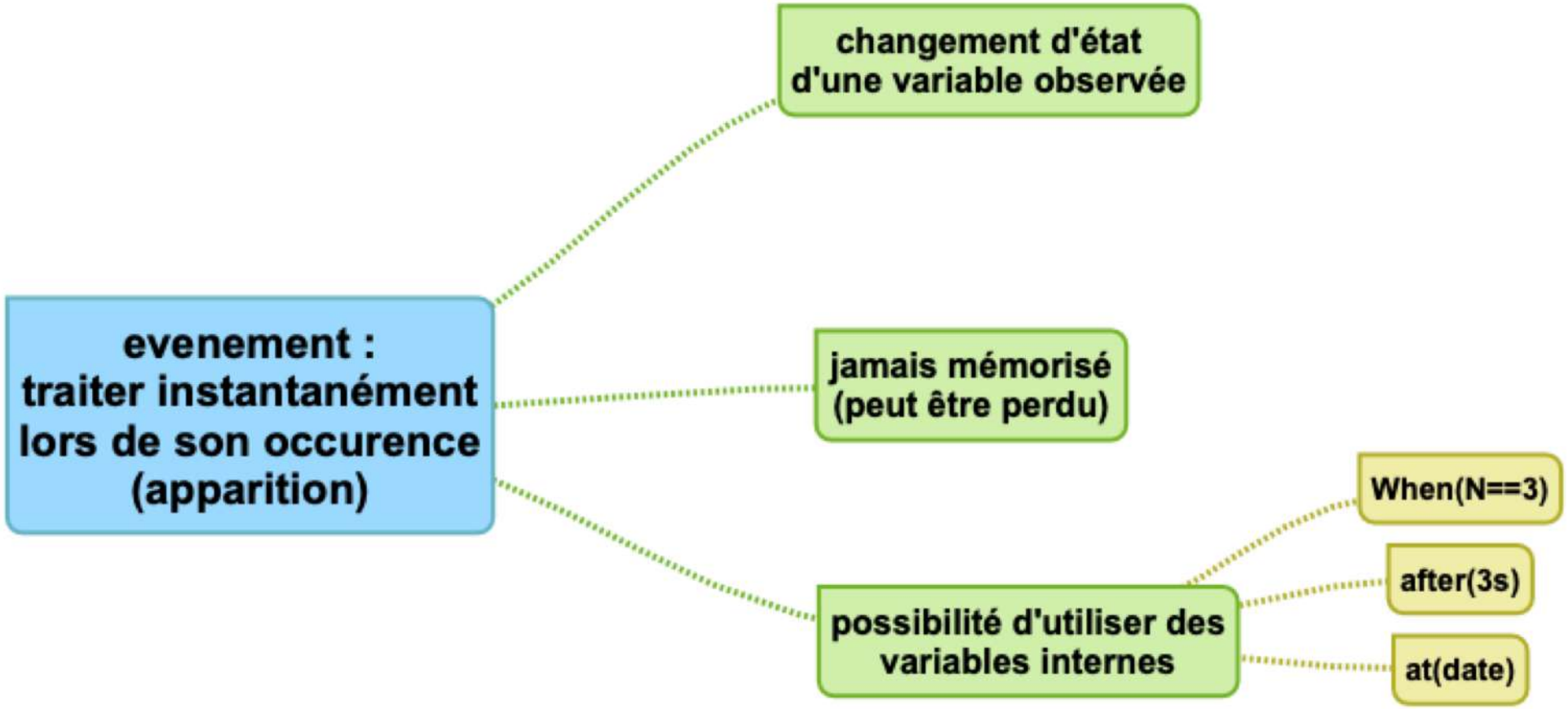
Si une **transition** n’a **pas d’événement** spécifié, l’**événement** déclencheur **implicite** est la **fin des activités*** liées au ***do*** de **l’état source**.

* Ces activités doivent donc avoir une fin.

Evolution d’un diagramme d’état par franchissement des transitions

événement [garde] / effet

Événement :



Evolution d’un diagramme d’état par franchissement des transitions

événement [garde] / effet

<u>Garde :</u>

La garde est une **condition de franchissement de la transition**. C’est une condition **logique** évaluée à **l’instant de l’évènement** déclencheur.

Exemple :
Pour un bouton :

- l’événement est associé à l’instant où le bouton est actionné ;
- la garde est associé à l’état du bouton : enfoncé ou non.

Evolution d’un diagramme d’état par franchissement des transitions

événement [garde] / effet

<u>Garde :</u>

La garde est une **condition de franchissement de la transition**. C’est une condition **logique** évaluée à **l’instant de l’évènement** déclencheur.

Exemple :
Pour un bouton :

- l’événement est associé à l’instant où le bouton est actionné ;
- la garde est associé à l’état du bouton : enfoncé ou non.

Si une **garde** n’est **pas présente** le long d’une transition, elle est **considérée** comme toujours **vraie**.

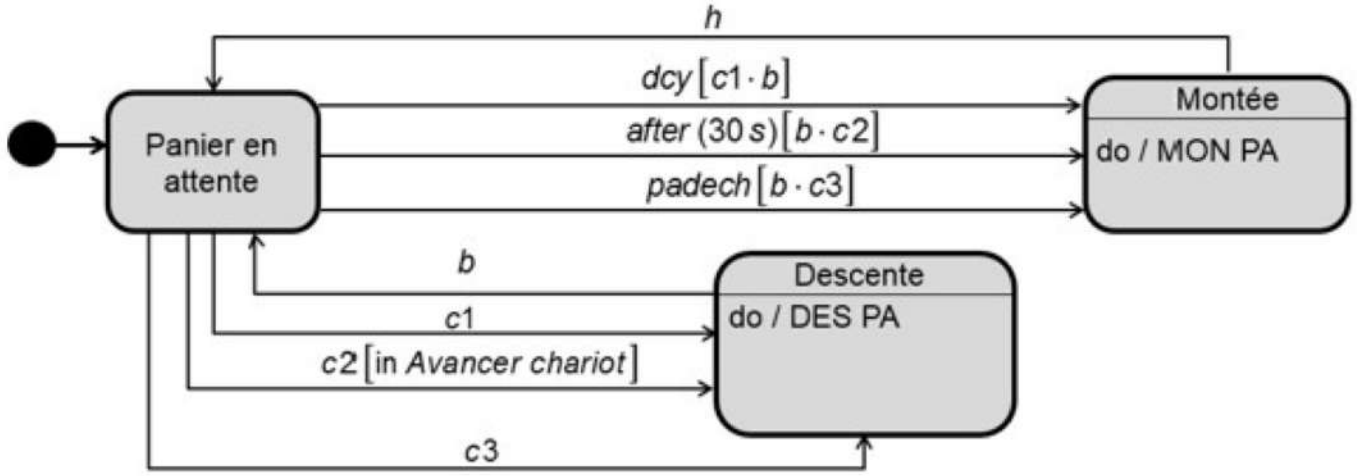
La syntaxe d’une condition de garde vérifiant si l’état ETAT est actif est : [in ETAT].

Evolution d'un diagramme d'état par franchissement des transitions

Garde :

Equation logique :

La condition logique évaluée pour la garde peut-être le résultat d'une combinaison de l'état de plusieurs grandeurs logiques, on parle alors du résultat d'une équation logique.



Dans ce type d'équation, on utilise des opérateurs logiques selon les règles de l'algèbre de BOOLE.

Ces 4 opérateurs logiques sont : **OUI, NON, OU, ET**.

Evolution d’un diagramme d’état par franchissement des transitions

événement [garde] / effet

<u>Effet :</u>

Un **effet** est une **activité** accomplie lorsque la **transition est franchie**.

Les activités associées aux effets sont considérées instantanées.

Une transition peut ne pas avoir d’effet associé*.

* Ce sera souvent le cas en pratique, les activités étant plutôt associées au états.

Chronologie du franchissement d'une transition :

On peut formuler le principe de fonctionnement de base d'un diagramme d'état de la façon suivante :

- Dés qu'un **état** est **actif**, la **transition** située **entre** cet **état source** et l'**état cible** est **franchissable** ;
- Lorsque l'**évènement** déclencheur **apparaît**, si la **condition** de garde est **vrai** :

l'éventuelle **activité** ***do*** de l'**état source** est **interrompue**

↓

l'éventuelle **activité** ***exit*** de l'**état source** est **exécutée**

↓

l'**état source** devient **inactif**

↓

l'éventuelle **activité** ***effet*** de la **transition** est **exécutée**

↓

l'**état cible** devient **actif**

↓

l'éventuelle **activité** ***entry*** de l'**état cible** est **exécutée**

Evolution d’un diagramme d’état par franchissement des transitions

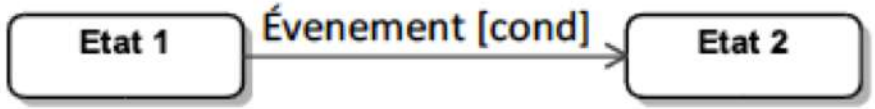
<u>Situations couramment rencontrées :</u>

À l'occurrence (apparition) de l'événement, la transition est franchie, sans condition.

Evolution d’un diagramme d’état par franchissement des transitions

<u>Situations couramment rencontrées</u> :

À l'occurrence (apparition) de l'événement, la transition est franchie, sans condition.

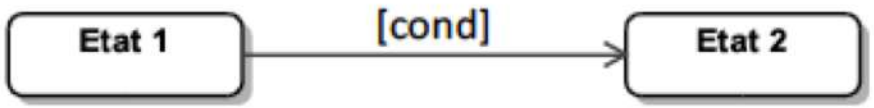


À l’occurrence de l'événement, si la garde « cond » est vraie, la transition est franchie .

Sinon, l'événement est « perdu » et il faut attendre une seconde occurrence pour éventuellement franchir la transition.

Evolution d’un diagramme d’état par franchissement des transitions

Situations couramment rencontrées :



Si la garde « cond » est vraie, la transition est :

- immédiatement franchie s’il n’y a pas d’activité associée à l’état 1 ;
- franchie dès la fin de l’éventuelle activité *do* de l’état 1. Cette fin d’activité constitue l’événement déclencheur implicite de la transition. **Il faut donc que cette activité do ait une fin**

Evolution d’un diagramme d’état par franchissement des transitions

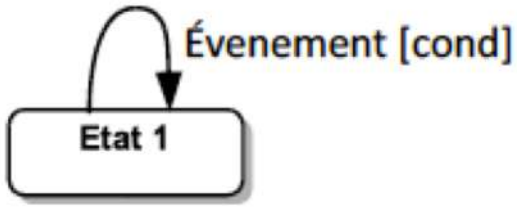
<u>Situations couramment rencontrées :</u>

La transition est dite automatique. Elle est :

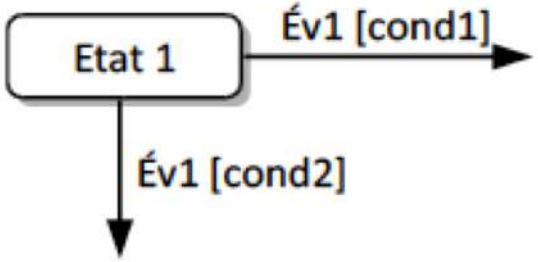
- immédiatement franchie s’il n’y a pas d’activité associée à l’état 1 ;
- franchie dès la fin de l’éventuelle activité *do* de l’état 1. Cette fin d’activité constitue l’événement déclencheur implicite de la transition. **Il faut donc que cette activité do ait une fin.**

Evolution d’un diagramme d’état par franchissement des transitions

<u>Situations couramment rencontrées :</u>



Une transition réflexive entraîne une sortie de l'état puis un retour dans ce même état, avec appel des éventuelles activités ***exit*** et ***entry***.



Plusieurs transitions peuvent quitter un même état.
Une seule d'entre elles doit être franchissable à un même instant.

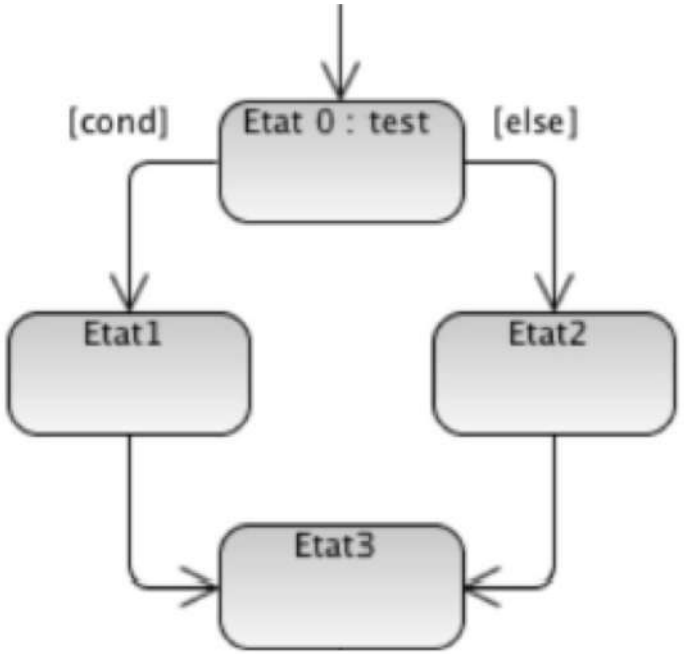
Les événements et/ou les conditions de garde doivent donc être exclusives.

Evolution d’un diagramme d’état par franchissement des transitions

<u>Structures algorithmiques de base :</u>

Les structures algorithmiques de base vues en informatiques peuvent être modélisées sous la forme des portions de diagrammes d’état suivantes :

Si *cond*, **alors** activer *Etat1,* **sinon** *Etat2* :

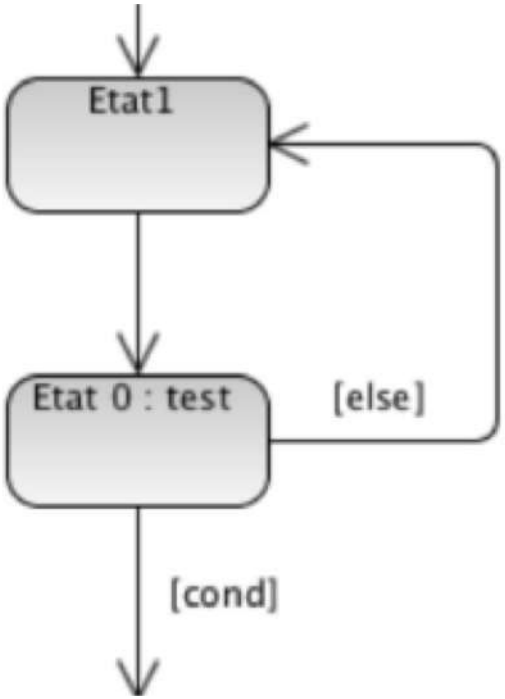


Evolution d’un diagramme d’état par franchissement des transitions

<u>Structures algorithmiques de base :</u>

Les structures algorithmiques de base vues en informatiques peuvent être modélisées sous la forme des portions de diagrammes d’état suivantes :

Répéter *Etat1*, jusqu’à *cond* :

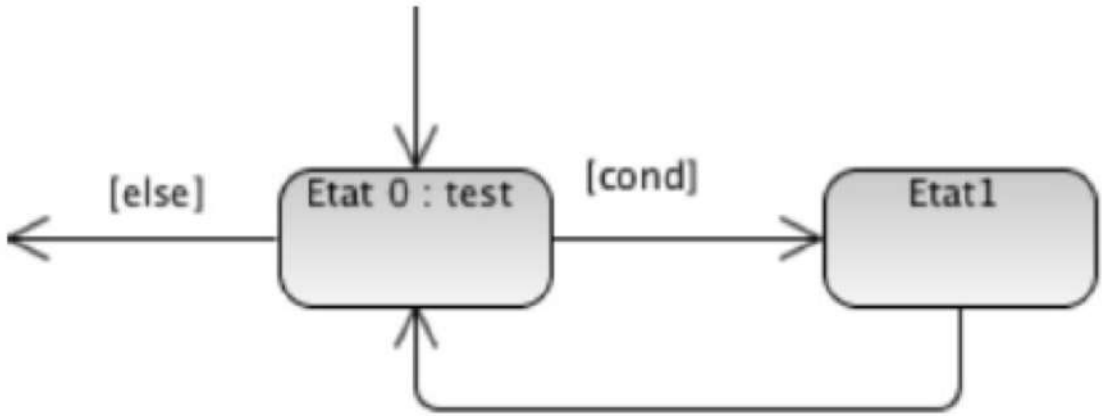


Evolution d’un diagramme d’état par franchissement des transitions

<u>Structures algorithmiques de base :</u>

Les structures algorithmiques de base vues en informatiques peuvent être modélisées sous la forme des portions de diagrammes d’état suivantes :

Tant que *cond*, activer *Etat1* :

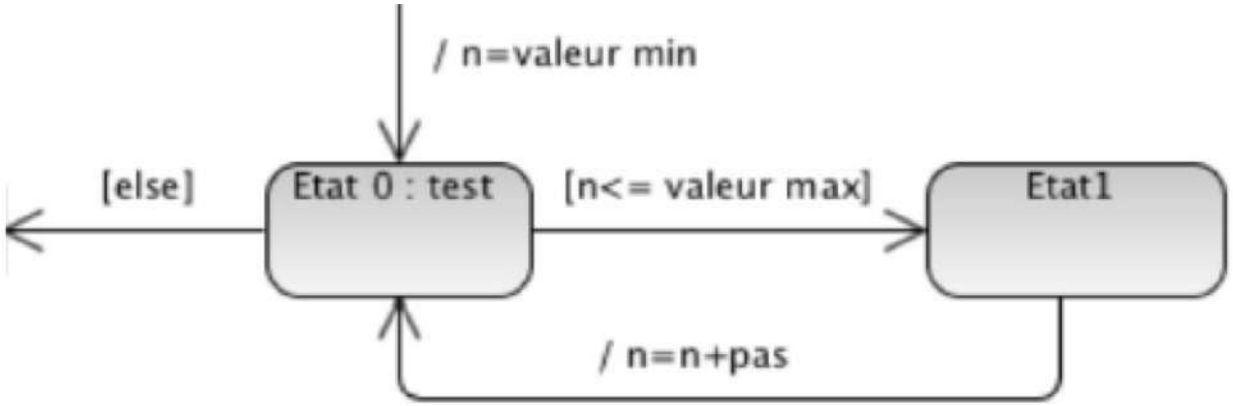


Evolution d’un diagramme d’état par franchissement des transitions

<u>Structures algorithmiques de base :</u>

Les structures algorithmiques de base vues en informatiques peuvent être modélisées sous la forme des portions de diagrammes d’état suivantes :

Pour *n* variant de *min* à *max*, activer *Etat1* :



Démarche de modélisation du comportement d’un SED

Une bonne modélisation du comportement séquentiel d’un système, en vue de programmer son unité de commande, nécessite de respecter la démarche suivante :

- Définir la **frontière** du système, **recenser les variables d’entrée** (consignes de l'utilisateur, grandeurs physiques acquises par les capteurs) **et de sortie** (messages à destinations de l'utilisateur ou d'autres systèmes et ordres vers les préactionnneurs).

Démarche de modélisation du comportement d’un SED

Une bonne modélisation du comportement séquentiel d’un système, en vue de programmer son unité de commande, nécessite de respecter la démarche suivante :

- Définir la **frontière** du système, **recenser les variables d’entrée** (consignes de l'utilisateur, grandeurs physiques acquises par les capteurs) **et de sortie** (messages à destinations de l'utilisateur ou d'autres systèmes et ordres vers les préactionnneurs).

- **Recenser**, **nommer** et tracer **les différents états** dans lesquels peut se retrouver le système lors de son utilisation. **Identifier les activités associées** à chaque état.

Démarche de modélisation du comportement d’un SED

Une bonne modélisation du comportement séquentiel d’un système, en vue de programmer son unité de commande, nécessite de respecter la démarche suivante :

- Définir la **frontière** du système, **recenser les variables d’entrée** (consignes de l'utilisateur, grandeurs physiques acquises par les capteurs) **et de sortie** (messages à destinations de l'utilisateur ou d'autres systèmes et ordres vers les préactionnneurs).
- **Recenser**, **nommer** et tracer **les différents états** dans lesquels peut se retrouver le système lors de son utilisation. **Identifier les activités associées** à chaque état.
- **Identifier** et tracer **les transitions** possibles entre les états en fonction du comportement séquentiel souhaité ou observé.

Démarche de modélisation du comportement d’un SED

Une bonne modélisation du comportement séquentiel d’un système, en vue de programmer son unité de commande, nécessite de respecter la démarche suivante :

- Définir la **frontière** du système, **recenser les variables d’entrée** (consignes de l'utilisateur, grandeurs physiques acquises par les capteurs) **et de sortie** (messages à destinations de l'utilisateur ou d'autres systèmes et ordres vers les préactionnneurs).
- **Recenser**, **nommer** et tracer **les différents états** dans lesquels peut se retrouver le système lors de son utilisation. **Identifier les activités associées** à chaque état.
- **Identifier** et tracer **les transitions** possibles entre les états en fonction du comportement séquentiel souhaité ou observé.
- **Définir les conditions** et **événements** déclencheurs associés à chaque transition et qui autorisent son franchissement.

Pseudos-états

Un pseudo-état est un état ne pouvant pas avoir d'activités.

Ils servent essentiellement comme éléments de liaisons et pour indiquer l'état initial ou l'arrêt du diagramme d'état.

Pseudo-état initial 

Unique et **obligatoire**, il est **activé au lancement de la machine** à états et marque le début de l'exécution du diagramme d'état. Il n'a **aucune transition entrante**.

Pseudo-état final 

Optionnel, il signe la fin de l'exécution du diagramme d'état. Il n'a aucune transition sortante.

Pseudos-états

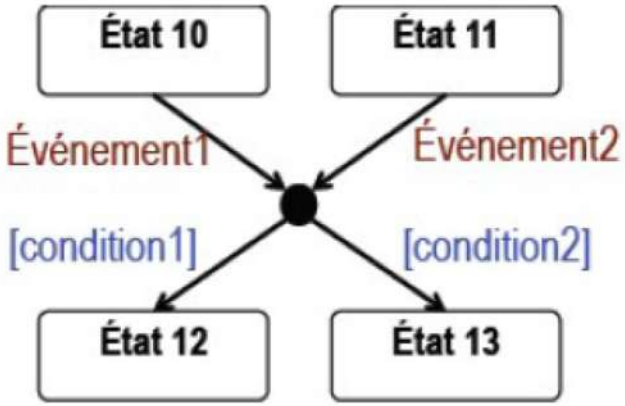
Pseudo-état de jonction ●

Utilisé pour regrouper (« factoriser ») des conditions de franchissement de transitions, en particulier des gardes communes à un événement.

Il permet de partager des segments de transition et d'aboutir à une notation plus lisible des chemins alternatifs.

L'évaluation des conditions de garde en aval du pseudo-état est réalisée avant qu'il ne soit atteint.

Exemple :
Ci-contre, on passe de l'état 10 à l'état 13 si l'événement 1 apparait et que la condition 2 est vrai



Pseudos-états

Pseudo-état de décision 

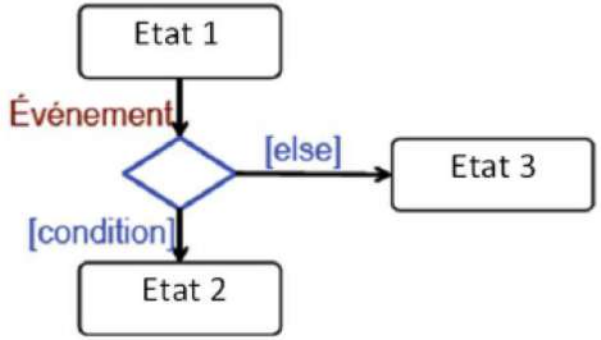
Utilisé pour une sélection ou une convergence de séquences exclusives.

L’évaluation des conditions de garde en aval du pseudo-état est réalisée au moment où il est atteint.

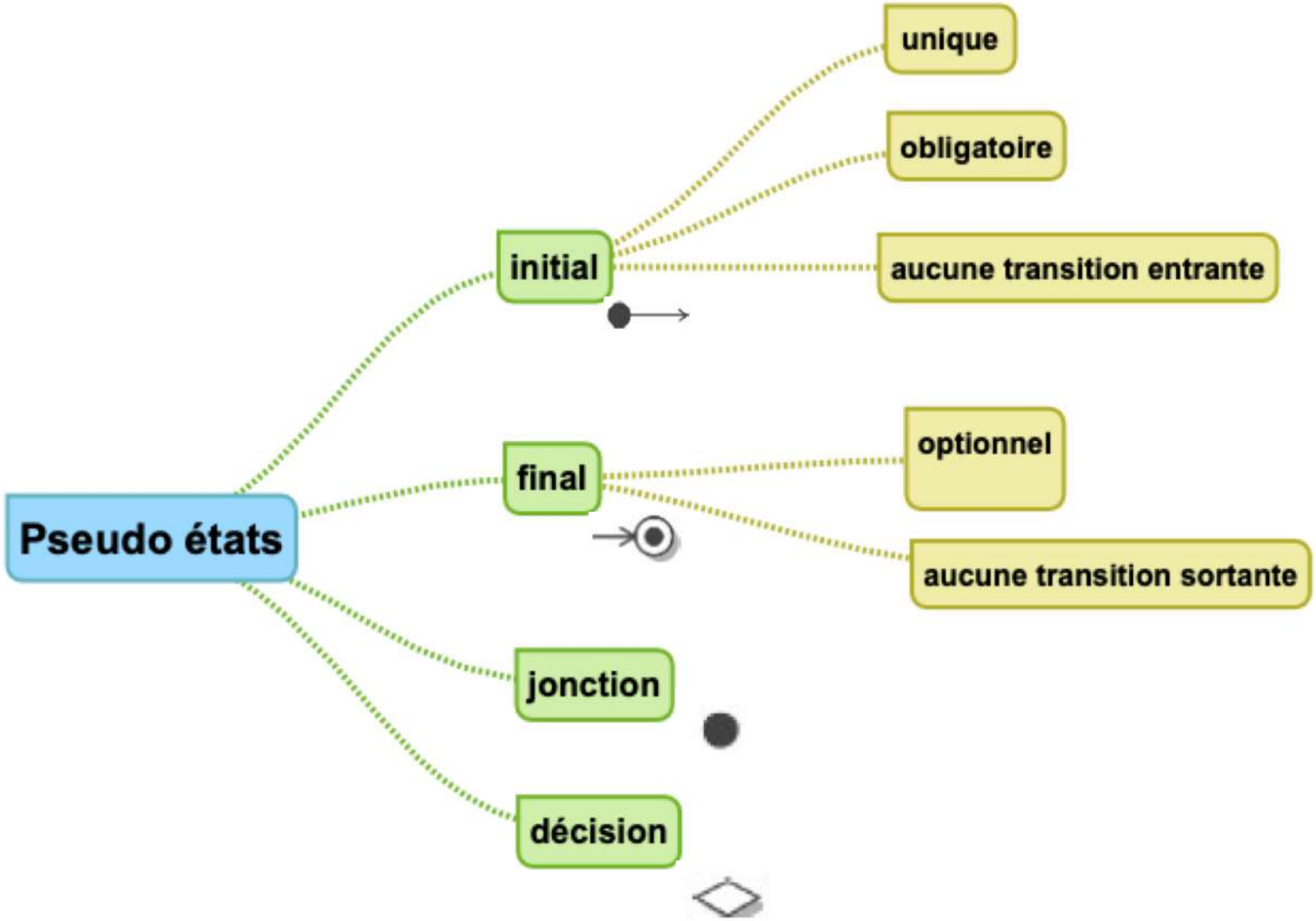
Les conditions de gardes doivent être exclusives.

L'utilisation d'une clause [*else*] est recommandée après un point de décision, car elle garantit un modèle correct en englobant tout ce qui n’est pas décrit dans les autres expressions logiques et en assurant ainsi qu’un moins un segment en aval est franchissable.

Exemple :
Ci-contre, dès que l’événement apparaît, le pseudo-état de choix est atteint. Si la condition est vraie, c’est l’état 2 qui devient actif, sinon, c’est l’état 3.



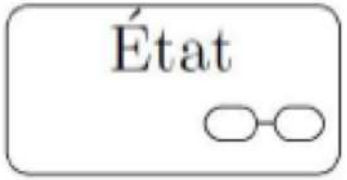
Pseudos-états



État composite

Notion de hiérarchisation

Un état composite décrit les évolutions internes d’un état à l’aide d’un autre diagramme d’état.

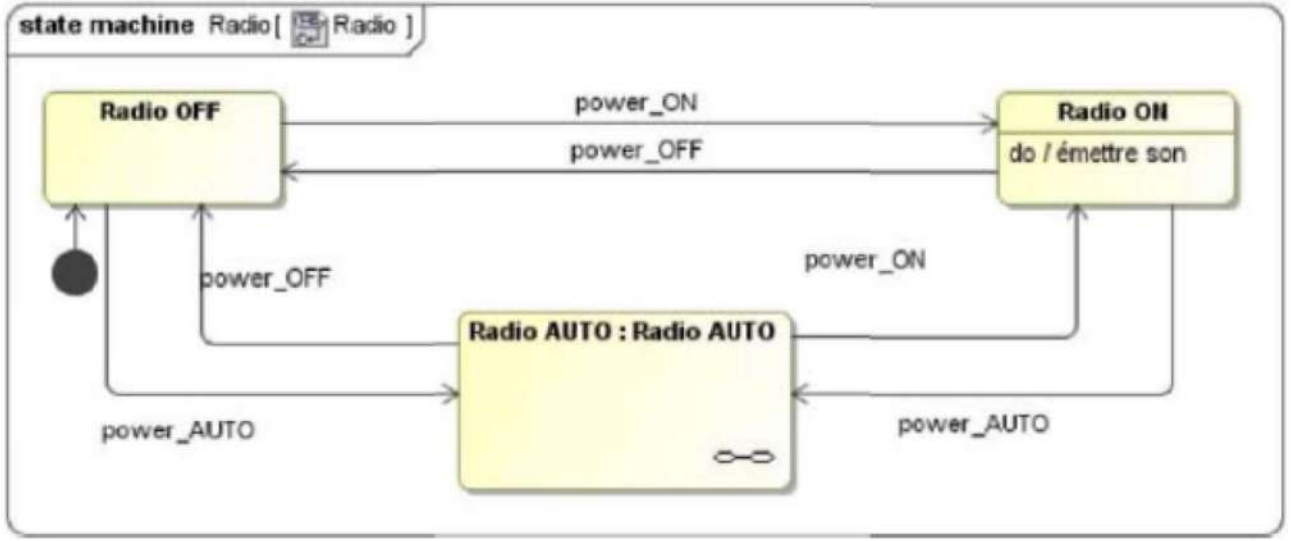


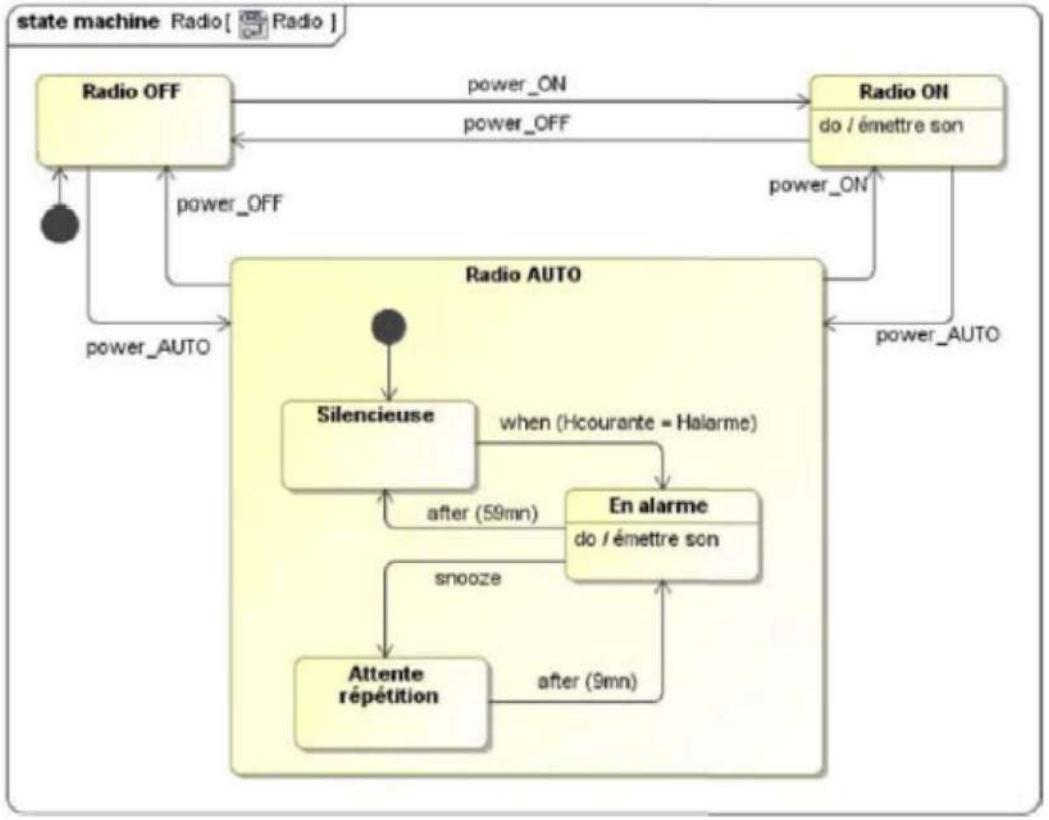
Pour repérer un état composite, un signe symbolisant des lunettes est apposé sur l’état.

Cette structure qui englobe plusieurs sous-états exclusifs considérés comme hiérarchiquement inférieur au diagramme principal, permet de rendre ce dernier plus lisible en entrant séparément dans le détail des évolutions internes du système.

État composite

Exemple : radio-réveil





État composite

Notion de hiérarchisation

Une **transition** qui atteint la **bordure** d'un **état composite** est **équivalente** à une **transition** qui atteint **l'état initial** de sa **région interne**.

Une **transition** qui sort de **la bordure** d'un **état composite** est équivalente à une **transition** qui **sort** de tous **les états de sa région interne**.

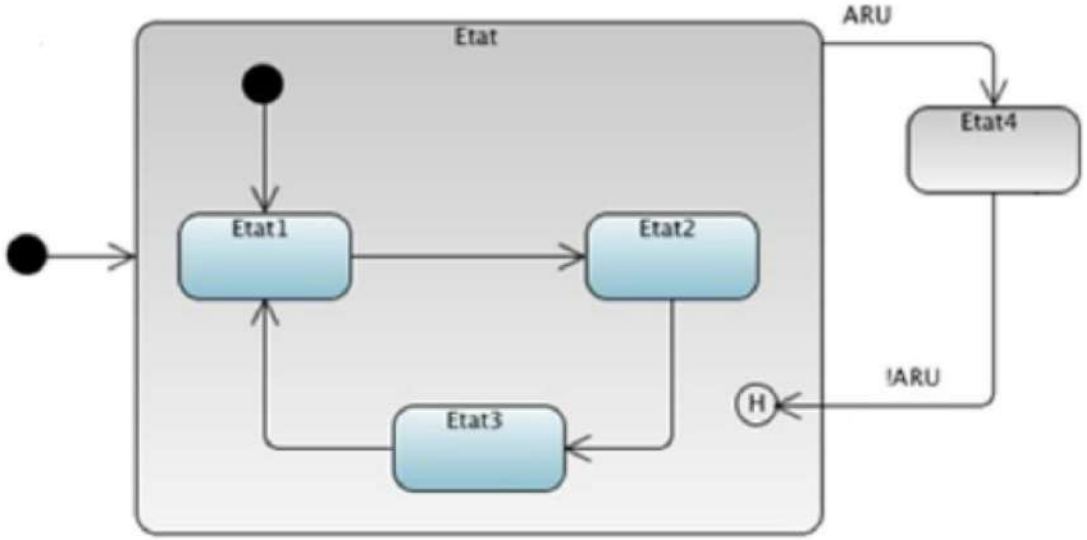
État composite

Historique d'un état composite

L'état actif au moment de la sortie d'un état composite peut être mémorisé par l'indication historique.

Lors de la réactivation de l'état composite, celui-ci se réactive à cet état.

Exemple :
L'historique est utilisé ici pour permettre à un système de recommencer en cours de cycle lors du redémarrage après un appui sur l'arrêt d'urgence (ARU).



État composite

État composite orthogonal

Dans un état composite orthogonal, plusieurs diagrammes d'états peuvent évoluer simultanément dans des régions séparées par des pointillés.
Les différentes régions de l'état orthogonal fonctionnent en parallèle sans aucune influence les unes sur les autres

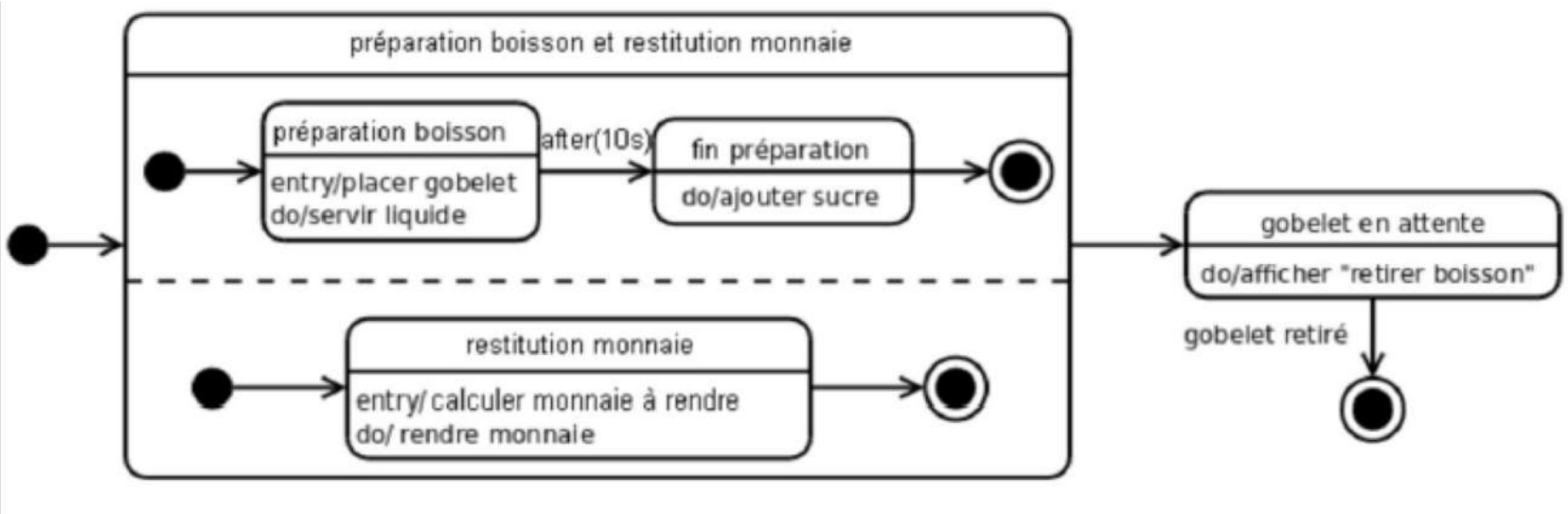
Une **transition** qui atteint **la bordure** d'un état **composite orthogonal** est **équivalente** à une **transition** qui atteint **les états initiaux** de **toutes** ses **régions**.

Toutes les **régions** d'un **état composite orthogonal** doivent **atteindre** leur **état final** pour **que l'état composite** soit considéré comme **terminé**. Ce n'est qu'à cette condition que la **transition** de **sortie** de **l'état composite** devient **franchissable.**

État composite

État composite orthogonal

Exemple : distributeur de boissons



État composite

État composite orthogonal

L**'activation** et la **sortie** d'un **état composite orthogonal** peuvent être également symbolisés par des **barres de synchronisation *fork* et *join*** qui fonctionnent par paire.

- Les **transitions**, nécessairement **automatiques**, qui **partent** d'une barre de synchronisation ***fork*** sont **franchies simultanément**.

- La **transition** qui **part** d'une barre de synchronisation ***join*** n'est franchissable qu'après le **franchissement** de **toutes les transitions**, nécessairement **automatiques**, qui **convergent** vers cette barre.

État composite

État composite orthogonal

Exemple : distributeur de boissons
Le diagramme d'état ci-dessous décrit le même comportement que celui présenté précédemment.