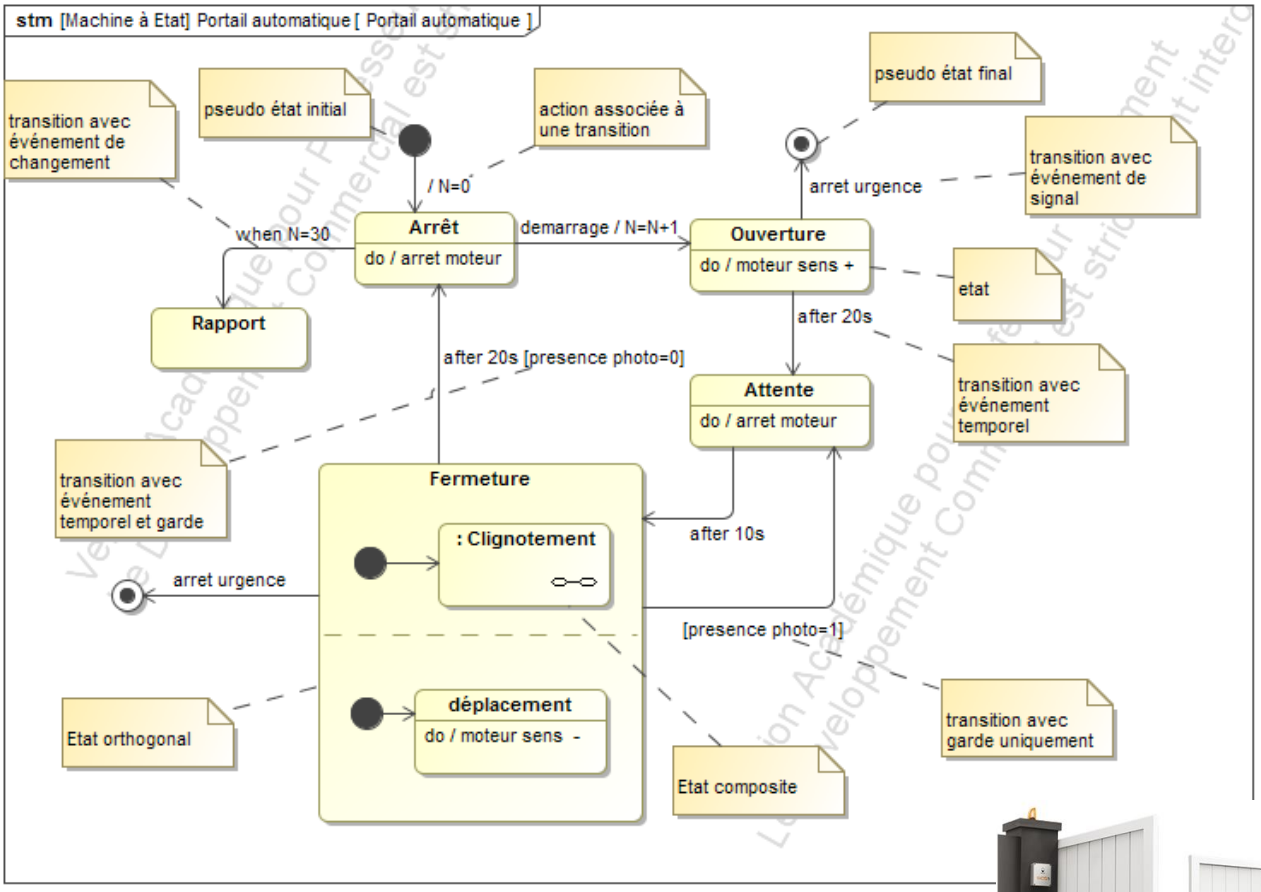
Systèmes à évènements discrets

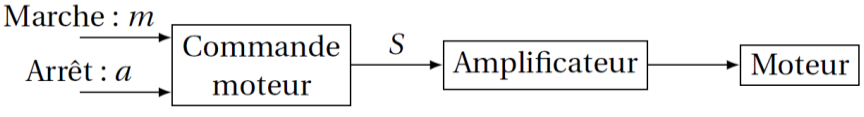


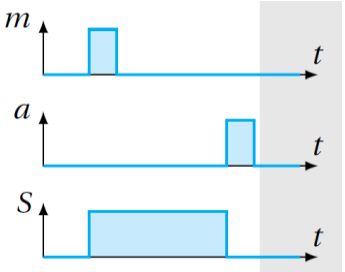
Portail automatique

On appelle système à événements discrets un système réel qui évolue par étapes.
La plupart des organes de commande de ces systèmes se programment en langage C. Les outils décrits à travers la norme SysML permettent de modéliser la structure du programme indépendamment de tout langage de programmation.

Différents outils sont utilisés pour décrire ou spécifier un système à événements discrets.

- Le chronogramme permet de représenter l'évolution temporelle de variables en fonction du temps. Ces variables prennent en général 2 états (vrai / faux, 1 ou 0), voir exemple sur la figure.
- Le diagramme d'état représente les états d'un système et les évolutions entre ces états suivant les modes de fonctionnement.
- Le diagramme d'activités permet de décrire le comportement et les processus se déroulant dans un état donné.





On se limite dans le programme à la lecture des diagrammes d'état et d'activités

1. DIAGRAMME D'ETAT

1.1. Définition des états

Notion d'état :

Un *diagramme d'état* du langage SysML donne une représentation graphique des transitions possibles entre les différents états dans une *machine d'état*.

En langage SysML, les éléments graphiques présentés sur la figure 1 sont classiquement utilisés pour représenter les machines d'état dans un diagramme d'état.

(a) Pseudos-états initial et final. (b) État générique et comportements. (c) État composite (sous-machine). (d) État orthogonal.

La figure de la première page représente le fonctionnement d'un portail automatisé constitué d'un bouton arrêt d'urgence, d'un capteur photoélectrique qui détecte la présence d'un véhicule et d'un bouton démarrage (entrées de la machine d'état). Un moteur tournant dans un sens ou dans un autre actionne les vantaux. Le diagramme d'état proposé synthétise l'ensemble des éléments syntaxiques couramment rencontrés.

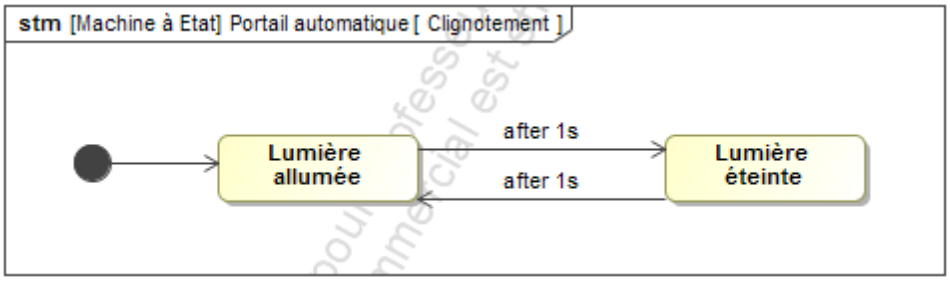
Pseudos-états de départ et de fin (figure a)
Les « pseudos-états » initial et final indiquent le début et la fin du fonctionnement de la machine d'état associée au diagramme d'état.
Il y a toujours **un pseudo-état initial** mais il peut ne pas y avoir de pseudo-état final.

Comportements associés à un état (figure b)
Des *comportements*, déclenchés grâce à des événements internes, peuvent éventuellement être précisés dans les états. La norme propose trois types de comportements qui ne sont utilisés qu'une fois par état, dans l'ordre suivant : *entry*, *do* et *exit*.
A chaque mot clé est associée une action ou une activité (ce peut être un diagramme d'activité).
Lors de la sortie de l'état, l'activité liée au comportement *do* est interrompue et celle associée au comportement *exit* est alors exécutée. Ceci permet d'effectuer des actions liées à la désactivation de l'état (éteindre une LED d'activité par exemple).

État composite (figure c)
L'*état composite* décrit les évolutions internes d'un état avec un autre diagramme d'état. Cette structure de « sous-machine d'état » facilite la lecture du diagramme en entrant dans le détail des évolutions internes.

Remarque :

Le signe indiqué sur l'état composite fait référence à un autre diagramme décrivant la sous-machine d'état : voir figure suivante. On peut également décrire la sous-machine d'état directement dans l'état composite.



État orthogonal (figure d)
Un *état orthogonal* possède plusieurs *régions*, séparées par des pointillés, chacune ayant sa propre description d'état. Les différentes régions d'un état orthogonal fonctionnent toutes en parallèle sans aucune influence les unes sur les autres.

1.2. Transitions : évolution et comportement associé

Une fois recensés tous les états d'un bloc, il faut ensuite en représenter les évolutions : ***on utilise des transitions****.*

Notion de transition :

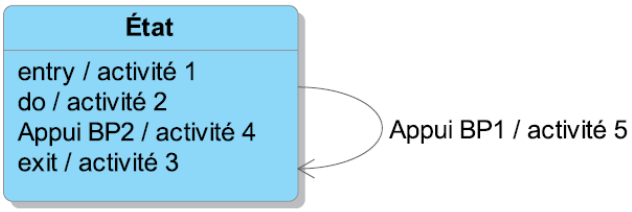
La transition est un lien orienté entre deux états. Elle est représentée par une flèche reliant l'état de départ, appelé *état source*, et l'état d'arrivée, appelé *état cible*.

La condition de franchissement et l'éventuel effet qui y est associé sont indiqués à côté de la flèche en suivant la notation événement [garde] / effet dont les termes sont définis dans la suite.

Remarque :

Une transition sans condition est dite « automatique ». Dans ce cas, la condition prise en compte pour la transition est la fin de l'activité associée au comportement *do* de l'état amont (ou *entry*).

Le cas d'une transition bouclant sur un état, appelée *transition réflexive*, est représenté sur la figure ci-dessous. Cette structure permet d'exécuter à nouveau les activités associées aux comportements d'entrée (*entry*) et de sortie (*exit*) de l'état.



Événement
L'événement est la première condition de franchissement de la transition.

Notion d'évènement :

Un événement correspond à l'occurrence du changement d'état d'un élément du modèle ou de la survenue d'un changement extérieur. Par définition, un événement :

-
-

À chaque transition est associé un événement unique.

La norme définit plusieurs types d'événements, parmi lesquels :

- L'*événement de signal* prend en compte la réception d'un signal asynchrone qui peut arriver à n'importe quel moment : c'est le cas par exemple pour l'appui sur un bouton de l'interface homme-machine ou l'arrivée en fin de course d'un mécanisme.
- L'*événement de changement* prend en compte le changement d'une valeur interne du modèle : c'est le cas par exemple pour un compteur qui délivre l'information *when(N=30)* quand 30 appuis ont été effectués.
- L'*événement temporel* peut être *relatif* ou *absolu* :
  - Relatif : *after(T)* se déclenche après le temps *T* passé dans l'état amont.
  - Absolu : *at(H)* se déclenche à la date *H* dans un référentiel de temps dont l'origine correspond généralement au démarrage de la machine d'état.

Garde

La garde est la deuxième condition de franchissement de la transition.

Notion de garde :

À l'occurrence de l'événement, la valeur vraie de la ***garde est une condition supplémentaire*** à l'événement pour que la transition soit franchie.

Un événement est obligatoirement associé à une transition, alors que la garde est optionnelle. Si la condition de garde n'existe pas, elle est considérée comme toujours vraie.

Contrairement à l'événement qui est de nature ponctuelle et non mémorisée (changement), la garde traduit une condition qui dure dans le temps (persistance).

Exemple :

Lors de l'appui sur un bouton :

-
-

Effet

Une transition peut ne pas avoir d'effet alors qu'il y aura toujours un événement et une garde (même implicite).

Notion d'effet :

Un effet est **un comportement** (action, envoi d'un message, mémorisation, etc.) accompli lorsque la **transition est franchie**.

1.3. Règles d'évolution

L'évolution d'une machine d'état décrite par un diagramme d'état suit les étapes suivantes :

- Le point de départ de la machine d'état est indiqué par le pseudo-état initial.
- Les transitions sont franchies en fonction des événements et des gardes : la suite des états actifs est donc fonction des événements qui se produisent.
- Un seul état peut être actif à la fois dans chaque machine ou sous-machine d'état. Ainsi, dans les états orthogonaux décrivant chacun une machine d'état indépendante, chaque région a un état actif : il existe donc plusieurs états actifs situés dans des machines d'état distinctes.

2. DIAGRAMME D'ACTIVITE EN LANGAGE SYSML

Une fois recensés et organisés les états d'un bloc, il est possible de décrire les comportements pouvant se dérouler à l'intérieur de chacun d'eux. Les diagrammes d'activités du langage SysML permettent notamment de représenter des processus.

Malgré la proximité de syntaxe entre le diagramme d'état et le diagramme d'activités, la sémantique associée à leurs éléments graphiques est différente

Activité :

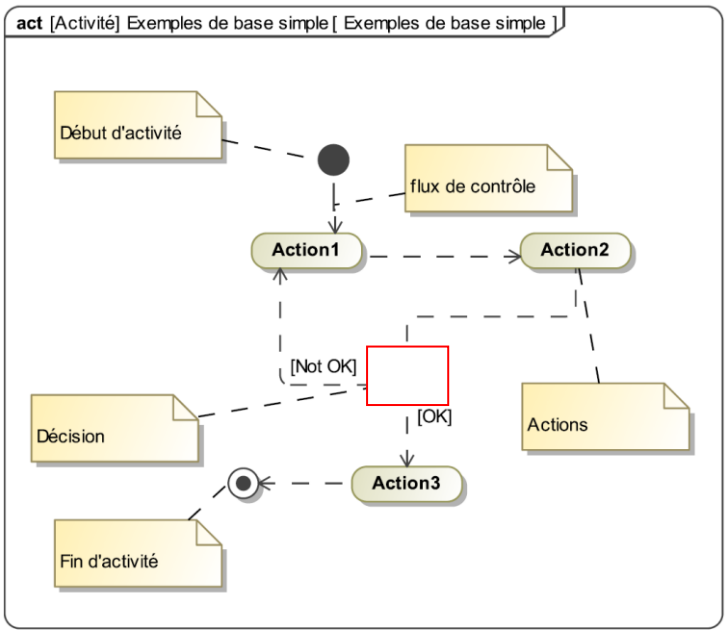
Pour simplifier, le diagramme d'activité peut être **assimilé à un algorigramme** : il décrit un déroulement d'actions qui, une fois engagé, va à son terme.

.

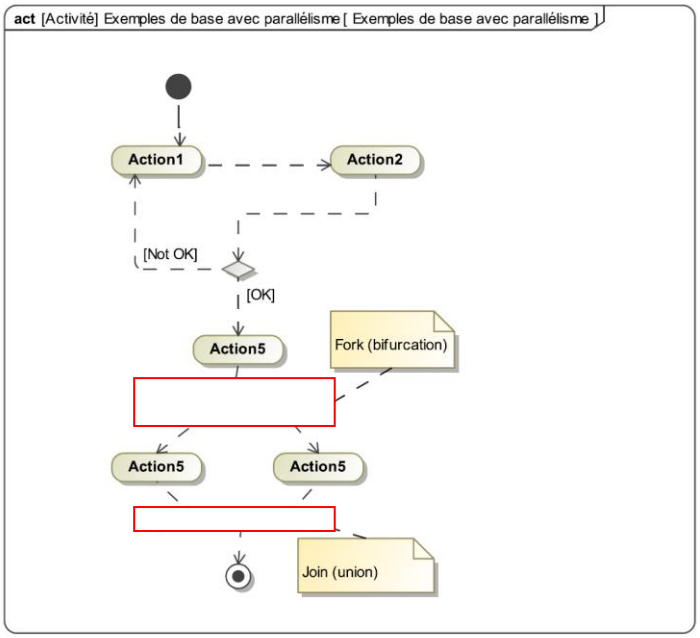
Les diagrammes utilisent les mêmes symboles de base que le diagramme d'état (blocs et flèches). Cependant, aucune condition n'est indiquée sur la flèche car celle-ci représente un signal émis dès que les actions associées à un bloc sont terminées.

Pour représenter **un choix** à faire entre plusieurs conditions exclusives, on utilise le **symbole de branchement conditionnel**. Les conditions sont notées entre crochets en sortie du branchement conditionnel (on peut utiliser le mot clé [else]). Ce symbole peut aussi être utilisé dans un diagramme d'état.

Actions en parallèle ou synchronisées : on utilise des symboles particuliers appelés **Fork et Join** comme indique sur la figure b (vrai également pour les diagrammes d'état lors de la sortie des états orthogonaux).



(a) Exemple avec 1 choix



(b) Exemple avec parallèlisme