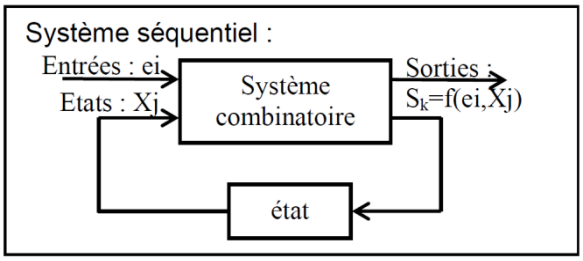
9 les systèmes à événements discrets

Sommaire

1) Rappels sur les systèmes à logique séquentielle :

Un système est dit à logique séquentielle, lorsque la ou les sorties **dépendent de la combinaison des entrées mais aussi de l'état précédent des sorties, et de la variable temps**.



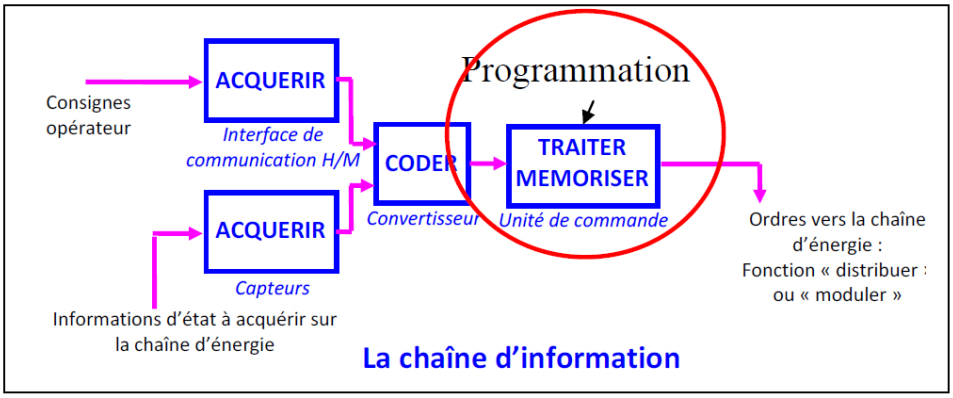
- Il peut exister des états différents des sorties pour un même état des entrées !
- L'effet peut persister si la cause disparaît.
  $S_i = f(e_1, e_2, \ldots, e_n; S_1, S_2, \ldots, S_p; \boldsymbol{t})$

Problématique : Comment décrire (et programmer) le comportement séquentiel d'un système ?

La description du comportement d'un système séquentiel peut être réalisée notamment par :

- L'outil « diagramme d'activité » SysML ; (hors programme, ne décrit pas les causes d'évolution)
- L'outil « diagramme de séquence » SysML ;
- L'outil algorithme (ou algorigramme) ;
- L'outil « diagramme d'états » SysML(ou graphe d'états) ;

Ces outils sont, à la base, des outils de modélisation du comportement séquentiel, mais peuvent aussi servir à la programmation des composants qui réalisent la fonction « Traiter » de la chaîne d'information (microcontrôleur, microprocesseur, automate programmable, puce à circuits logiques programmables…).



2) Description comportementale par le « Diagramme de séquences[1] » :

2.1) Objectif :

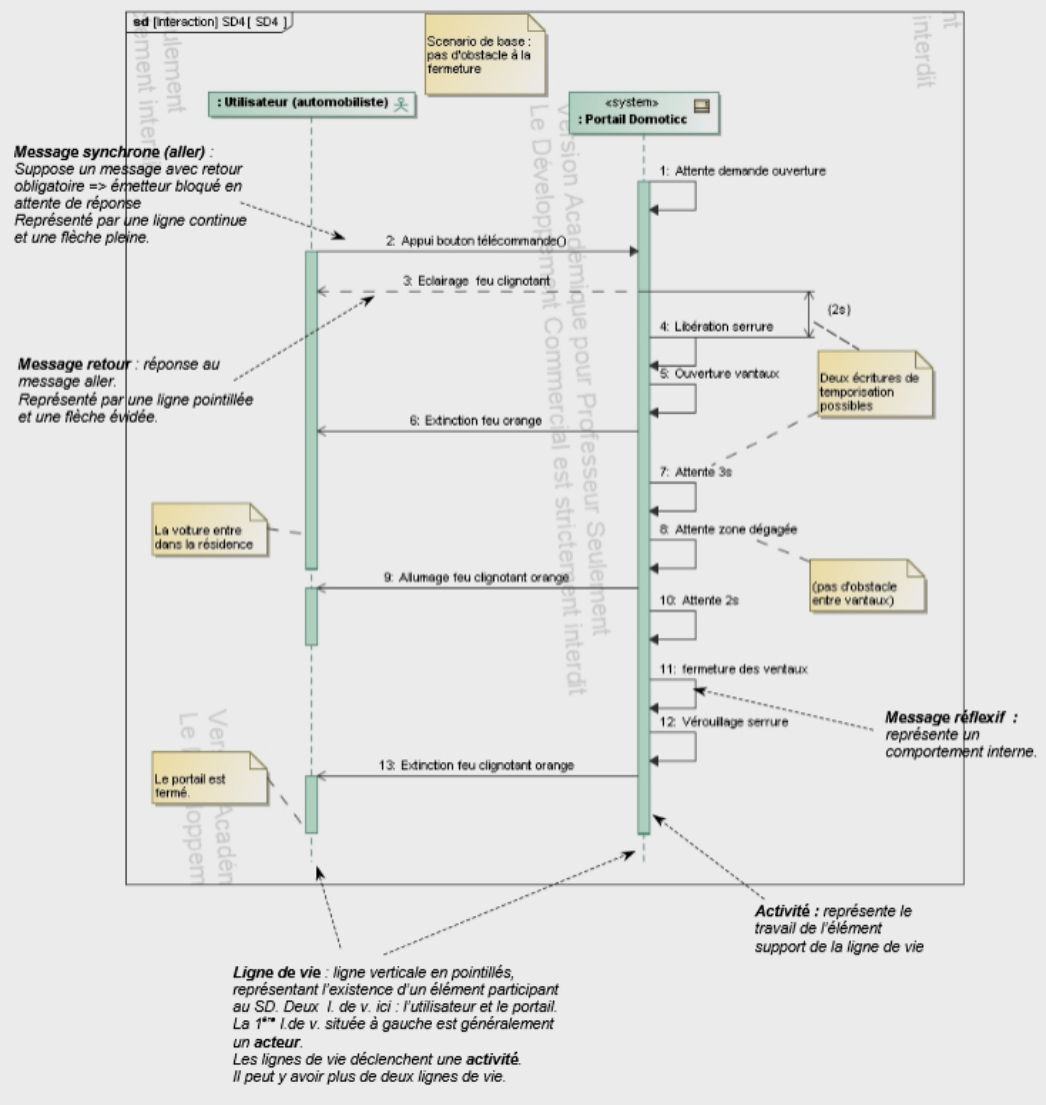
Décrire **dans l'ordre chronologique** les interactions entre les acteurs et composants d'un système, ou entre les composants eux-mêmes. Il décrit le(s) scénario(s) liés au(x) cas d'utilisation. C'est un diagramme temporel qui représente l'enchainement séquentiel des interactions nécessaires pour assurer un cas d'utilisation.

Scénario :

Le SD présente un scénario possible lors du fonctionnement du système. On peut donc réaliser autant de SD qu'il existe de scenarios d'un système : scénarios de réussite, scénarios d'échec (ex : on place sur une balance électronique maxi 3kg, un objet pesant 4kg), scénarios de dysfonctionnement (plantage d'un appareil électronique).

2.2) Syntaxe de base :

Observons le cas d'utilisation principal du portail « Automatiser l'accès à la résidence». Nous pouvons proposer le diagramme de séquence suivant :



[1] le diagramme de séquence est un des outils de la norme « SysML »

- Le message : Elément de communication unidirectionnel entre deux lignes de vie qui déclenche une activité de l'émetteur. La réception du message provoque une activité du récepteur. Le message retour s'indique en pointillés. Cela signifie qu'il s'agit de la réponse directe au message précédent.

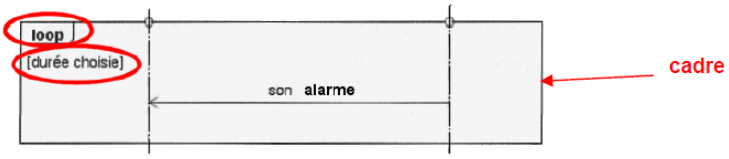
  Deux types de messages :

  - Le message synchrone attend un retour de la part du récepteur (l'émetteur est bloqué, en attente d'une réponse du récepteur). Il se représente par une flèche pleine.
  - Le message asynchrone n'attend aucun retour. Il se représente par une flèche évidée.

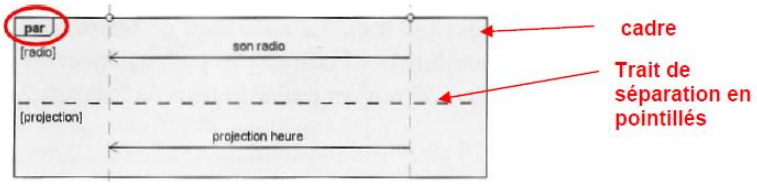
- L'activation : les bandes verticales larges superposées aux lignes de vie représentent une période d'activation (activité) de celle-ci.

2.3) Compléments de syntaxe du diagramme de séquences :

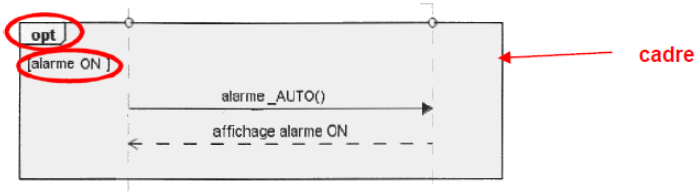
- L'action **maintenue** (boucle « *loop* ») : répète la séquence tant que la [*condition*]est respectée.



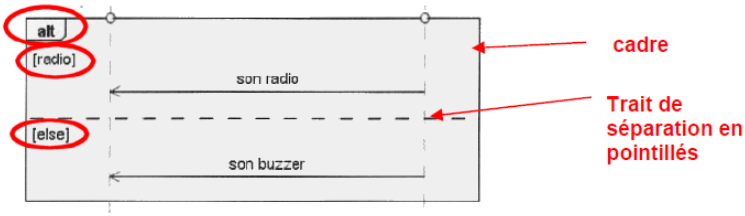
- Les interactions en « *parallèle* » : **les interactions sont exécutées simultanément** :



- Les interactions optionnelles : **l'interaction n'existe que si la condition fournie est vraie** :

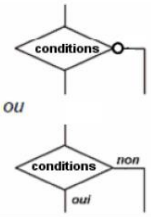
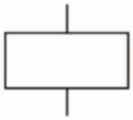
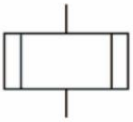
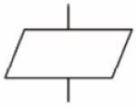


- Les interactions « *alternatives* » : **seule l'interaction possédant la condition vraie s'exécute** :

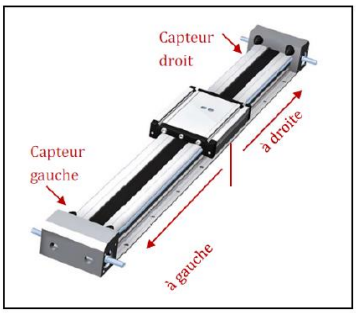


3) Description par « l'algorigramme » :

Syntaxe générale :

SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	**début** ou **fin** d'un algorithme		**Test ou Branchement conditionnel** décision d'un choix parmi d'autres en fonction des conditions
	symbole général de « **traitement** » opération sur des données, instructions, … ou opération pour laquelle il n'existe aucun symbole normalisé		**sous-programme** appel d'un sous-programme
	**entrée / sortie de données**		**Liaison** Les différents symboles sont reliés entre eux par des lignes de liaison. Le cheminement va de haut en bas et de gauche à droite. Un cheminement différent est indiqué à l'aide d'une flèche

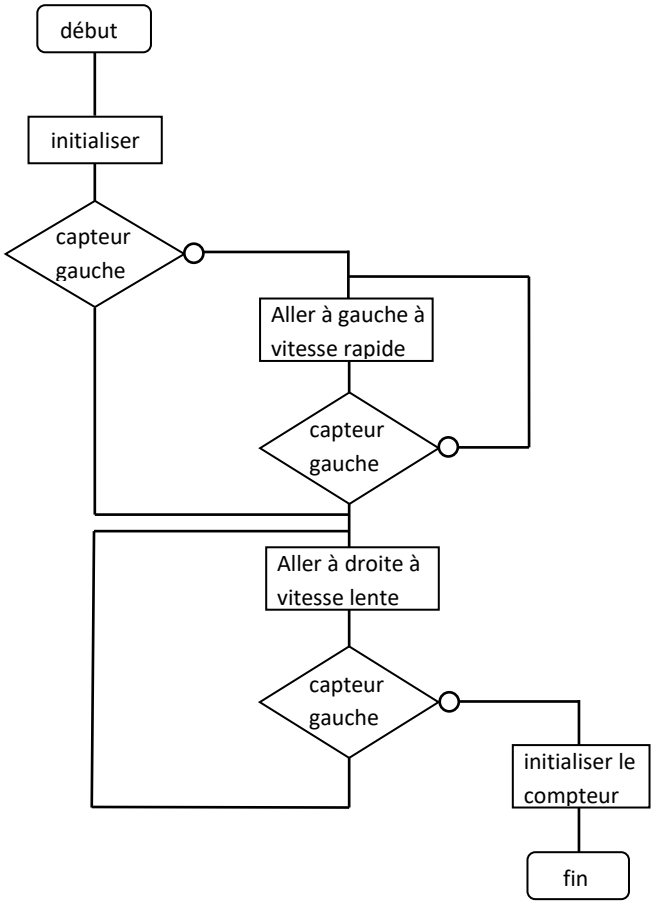
Exemple : Axe linéaire à codeur incrémental ; Description de la procédure de prise d'origine :



Après appui sur le bouton d'initialisation si le capteur gauche n'a pas détecté la présence du chariot celui-ci se translate à vitesse rapide en boucle ouverte sur la gauche. Dès que la présence du chariot est détectée à gauche il repart à droite à vitesse lente toujours en boucle ouverte jusqu'à ce que le capteur gauche commute à nouveau.

Si au moment de l'appui sur le bouton d'initialisation, le chariot est à gauche et donc que le capteur gauche a détecté sa présence, il se déplace vers la droite à vitesse lente en boucle ouverte jusqu'à ce que le capteur gauche commute.

Au moment de la commutation du capteur gauche lorsque le chariot se déplace vers la droite, la boucle en position est fermée et la consigne de position imposée nulle. Le compteur associé au codeur est simultanément initialisé. Le compteur est incrémenté pour un déplacement vers la droite et décrémenté pour un déplacement vers la gauche. Le cycle d'initialisation de l'axe peut être représenté par l'algorigramme ci-contre.



Conseil : « décomposer pour mieux résoudre »

Il est préférable de réaliser de petits algorigrammes (sous programmes) appelés parfois «macros» ou «fonctions» décrivant chacun une tâche simple, puis de faire un algorigramme plus général qui fait appel à ces sous-programmes.

4) Description et programmation par l'outil « Graphe d'état[2] » :

4.1) Définition :

Le **diagramme d'état**, encore appelé **machine d'état** (SMD), montre les différents **états** pris par un système ou un sous-système en fonction d'**interactions avec le milieu extérieur ou internes**.

Il répond à la question : « comment représenter les différents états du système ? ».

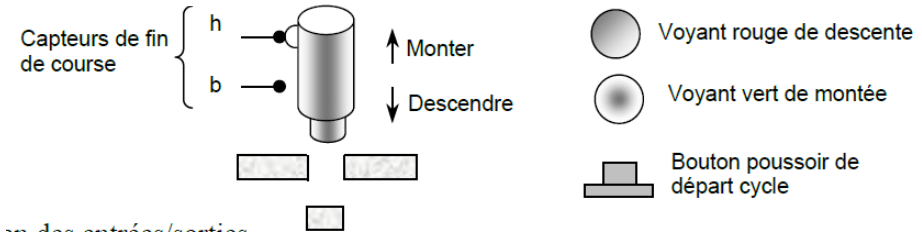
Remarque : le système étudié (ou le sous-système) est appelé **bloc**. Un **bloc** passe par une **succession d'états** au cours de sa vie.

4.1.1) Premier diagramme d'état : vocabulaire et notions de base

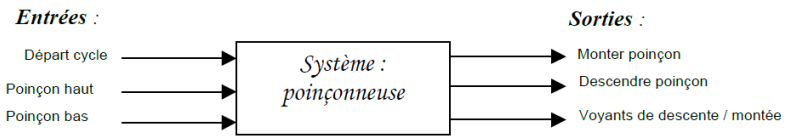
Exemple d'une poinçonneuse :

Cahier des charges de la séquence de poinçonnage :

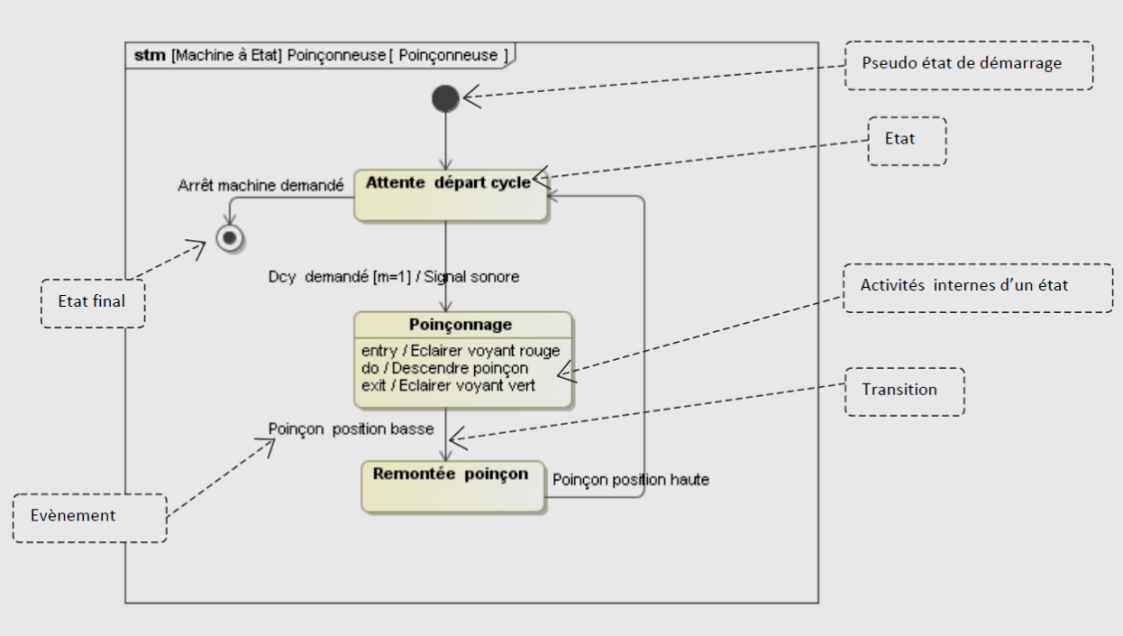
Quand le départ cycle est demandé par l'opérateur, un signal sonore de prévention retenti. Le poinçon descend et un voyant rouge s'allume. Quand le poinçon atteint sa position basse, un voyant vert s'allume, et il remonte. Arrivé en position haute il attend un nouvel ordre de l'opérateur. Si on arrête la machine, le poinçonnage ne peut plus avoir lieu.



Bilan des entrées/sorties :



[2] le graphe d'états est un des outils de la norme « SysML »



4.1.2) Etat : activité et action :

Un état représente une période de la vie du système. L'état est représenté par un rectangle aux coins arrondis dans le SMD. On lui attribue un substantif d'action appelé activité de l'état (Eclairage voyant, Rotation moteur, Attente, Comptage, Déplacement, etc…). Cet état peut être **actif** ou non.

état
entry /action d'entrée do / activité exit / action de sortie

À un état, on peut principalement rattacher une activité, une action d'entrée et une action de sortie :

- Action exécutée en entrant dans l'état (repérées par le terme « *entry* »); Une **action** ne prend pas de temps et ne peut pas être interrompue. Son exécution peut par exemple provoquer un changement d'état, l'émission d'un ordre pour un préactionneur ou un retour de valeur.
- Activité exécutée tant que l'on reste dans l'état (terme «*do* ») ; Une **activité** peut être considérée comme une unité de comportement. Elle prend du temps et peut être interrompue.
- Action exécutée en sortant de l'état (terme « *exit* »).

Important : dans un SMD donné, un seul état peut être actif à un instant donné… **SAUF** dans des états orthogonaux dont le découpage en plusieurs **régions** permet la simultanéité d'états différents.

Etat de démarrage : Il précède la première transition à franchir au lancement du diagramme d'état.

- Ne peut être suivi que d'une seule transition.
- Ne peut pas être la cible d'une transition.
- Pas d'activité associée.
- Il ne peut y avoir qu'un seul pseudo état de démarrage dans un SMD. (Sauf dans des états orthogonaux).

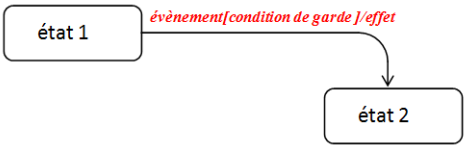
Etat de fin : Il termine une séquence décrite dans le SMD.

- Il peut y avoir plusieurs pseudo états finaux dans un SMD.
- Quand un état final est atteint toute la séquence décrite dans le SMD s'arrête.

Plusieurs **transitions** peuvent aboutir à un même pseudo état final.

4.1.3) Transition : évènement, condition de garde et effet :

Une transition lie deux états 1 et 2. Elle représente la possibilité de sortie d'un état 1 d'entrer dans un état 2. L'état précédent est appelé la **source** de la transition, l'état suivant est la **cible**.



- Une transition **n'est évaluée** que **si l'état source est actif**.
- Une transition est **déclenchée** par un évènement. En d'autres termes : c'est l'arrivée d'un évènement qui conditionne le déclenchement de la transition. Le **déclenchement** d'une transition a pour conséquence : la sortie de l'état source et l'entrée dans l'état cible. Cela implique un changement d'état du bloc : il était dans l'état 1, il se trouve maintenant dans l'état 2.
- Une transition ne possède pas toujours un évènement associé, on l'appelle alors transition automatique.

Spécification d'un événement associé à une transition (3 parties optionnelles) :

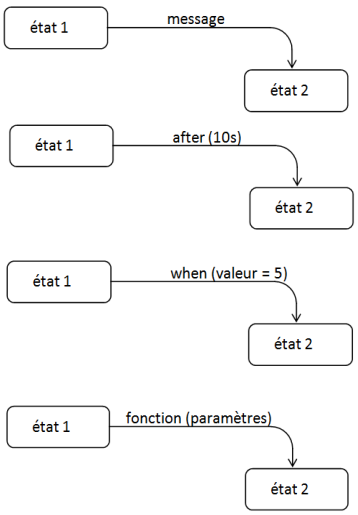
évènement [condition de garde]/effet

Exemple : Voir page précédente : *Dcy demandé [m=1] /Signalsonore*

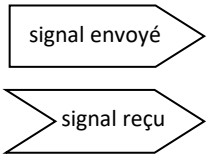
évènement*[condition de garde]/effet*
Dcy demandé *[m=1] /Signalsonore*

Il existe quatre types d'*évènements* associés à une transition :

- Le message (*signal event*) : un message asynchrone est arrivé,
- L'évènement temporel (*time event*) : un intervalle de temps s'est écoulé depuis l'entrée dans un état (mot clé « *after* ») ou un temps absolu a été atteint (mot clé « *at* »),
- L'évènement de changement (*change event*) : une valeur a changé de telle sorte que la transition est franchie (mot clé « *when* »),
- L'évènement d'appel (*call event*) : une requête de fonction (opération) du bloc a été effectuée. Un retour est attendu. Des arguments (paramètres) de fonction peuvent être nécessaires.



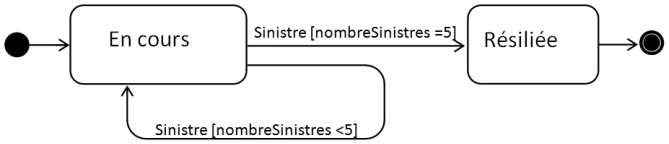
Les évènements peuvent être utilisés pour décrire les interactions entre les différents blocs d'un système. En effet, un évènement peut être émis par un autre bloc (*send signal action*) que celui pour lequel le diagramme d'états est spécifié. Il est alors par défaut destiné à tous les blocs du système (diffusion « *broadcast*) ». De même, un évènement peut être reçu (*receive signal action*).



*évènement[**condition de garde**]/effet*
*Dcy demandé [**m=1**] /Signalsonore*

La [*condition de garde*] est une expression booléenne faisant intervenir des entrées et / ou des variables internes.

- Elle est évaluée si l'évènement se produit.
- Quand la condition de garde est vraie on dit alors que la transition est **validée**.
- Elle autorise le passage d'un état à un autre (déclenchement de la transition).
- Les gardes doivent être mutuellement exclusives



Nota : Il est possible d'utiliser les notations non booléennes de front montant (↑) et front descendant (↓).

Différence entre évènement et condition de garde :

- **Un événement est parfaitement daté dans le temps**, il correspond par exemple à un passage d'une variable de 0 à 1 à un instant précis (front montant) ;
- **Une condition de garde n'est pas datée**, elle doit être vraie (niveau logique 1) à l'instant où l'événement survient pour que la transition soit déclenchée.

Exemple d'évènement : appui sur un bouton-poussoir, capteur fin de course atteint, etc…

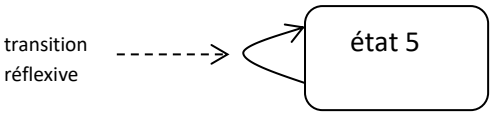
Exemple de condition de garde : vitesse du véhicule non nulle, température > 20°C, etc…

*évènement[condition de garde]/**effet***
*Dcy demandé [m=1] /**Signalsonore***

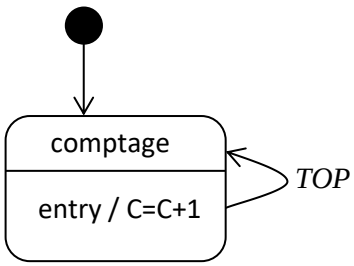
L'*effet* associé à une transition est effectué lorsque la transition est déclenchée (ici le *signal sonore* est **l'effet** qui se produit lorsque sa transition est déclenchée).

Nota : la présence d'actions dans les transitions nuit à la lisibilité des graphes d'états ; Donc sauf cas très particuliers, **on évitera l'utilisation de l'effet au franchissement**.

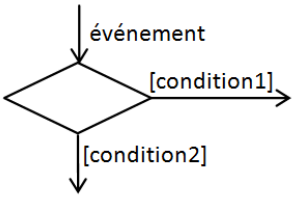
Transition réflexive : Une transition réflexive entraîne une sortie d'état puis un retour dans ce même état.



Exemple : Transition réflexive utilisée en comptage de la variable *TOP*



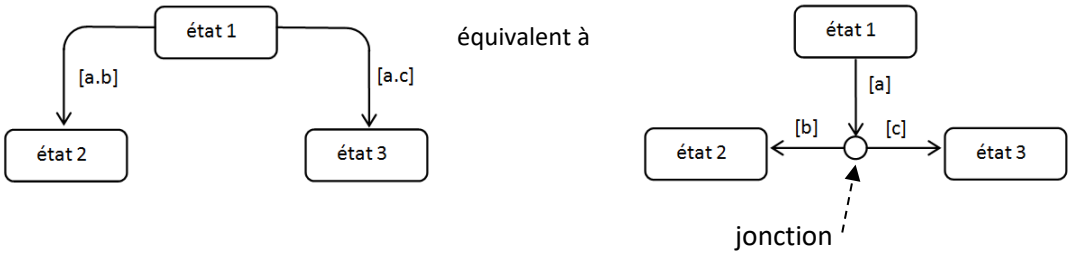
4.1.4) Les syntaxes particulières :pseudo états "*junction*" et "*choice*"



Pseudo état "*choice*" ou point de décision : Il permet la sélection et la convergence de séquences exclusives. Il est nécessaire qu'une condition située en aval soit vraie pour que l'évolution du système se poursuive. Les **conditions de gardes doivent être exclusives**. Le mot clé « *else* » peut-être utilisé pour englober tout ce qui n'est pas décrit dans les autres expressions booléennes. Les conditions de garde situées en aval sont toutes évaluées une fois le pseudo-état atteint.

Pseudo état "*junction*" : La jonction est un **pseudo-état** utilisé par exemple pour factoriser des expressions booléennes associées à des conditions.

Exemple :

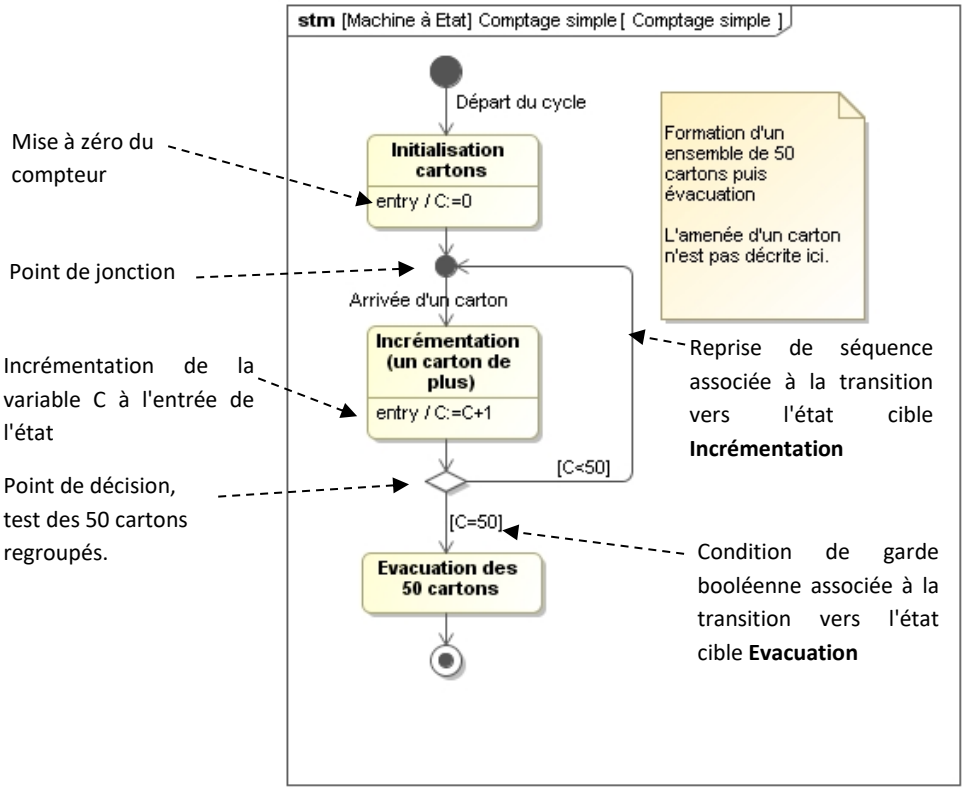


Lorsque la condition [a] est vraie, la jonction devient active, mais *état 1* est désactivé seulement si *[b]* ou *[c]* deviennent vrais.

Pour qu'un chemin soit emprunté, toutes les conditions de garde situées en aval et en amont, doivent être vraies. L'évaluation des conditions avales est réalisée avant que le pseudo-état soit atteint.

Exemple montrant un processus de comptage et de reprise de séquence :

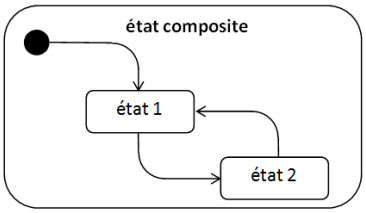
Cahier des charges : A la demande du départ cycle par l'opérateur, on doit compter 50 cartons, puis les évacuer. Le système d'approvisionnement de chaque carton n'est pas l'objet de l'étude ici. Seul le système de comptage doit être décrit. Seul l'évènement d'arrivée d'un carton déclenche donc le comptage.

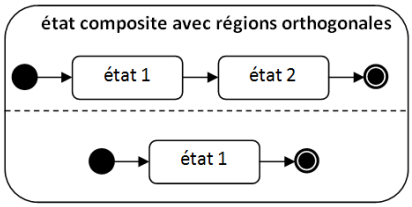


4.2) Hiérarchisation, état composite, états orthogonaux :

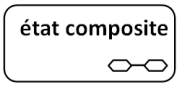
4.2.1) Définition d'un état composite :

Un **état composite**[3] est constitué de sous-états internes liés par des transitions. Cela permet d'introduire la notion d'état de niveau **hiérarchique** inférieur et supérieur.
Un état composite contient un état initial.
Différentes représentations :





notation abrégée :



Intérêt :
Les états composites permettent :

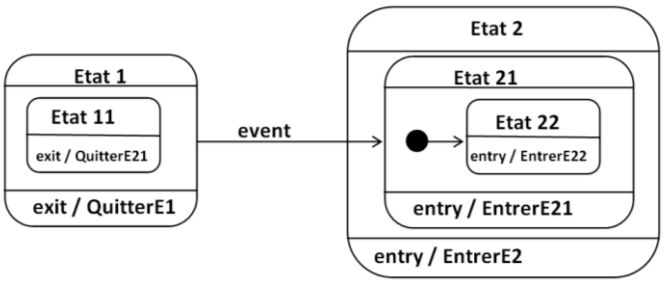
- De donner de la lisibilité en simplifiant la description des évolutions
- De gérer par zones séparées (régions), les différentes parties d'une machine ;
- De fournir la possibilité de hiérarchiser les évolutions. ….

4.2.2) Règles :

- Un état composite contient un état initial qui est activé lorsque l'état composite est activé.
- Un sous-état ne peut être actif que si l'état composite dans lequel il se trouve l'est.
- Il ne peut y avoir qu'un seul état actif dans le sous-état et dans l'ensemble du graphe, sauf dans le cas des « structures à évolutions simultanées » où plusieurs graphes d'états peuvent être actifs simultanément (voir plus loin).
- Règle de sortie de l'état composite : la **transition de sortie** d'un état composite est **évaluée** quel que soit l'état actif de l'état composite. En conséquence, la transition peut être déclenchée quel que soit l'état actif à l'intérieur de l'état composite. On peut donc sortir de l'état composite alors que son dernier état n'est pas actif.

4.2.3) Exemples :

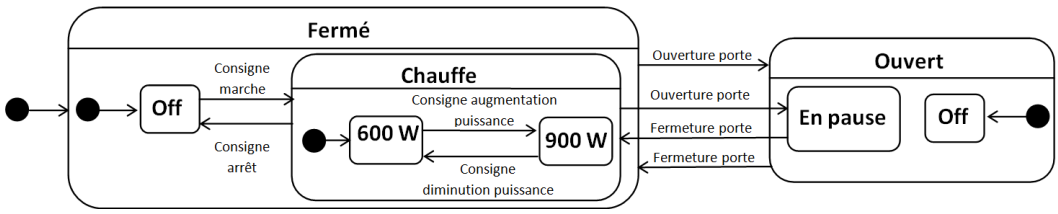
Règles d'évolution avec des états composites :
Supposons que l'état actif soit l'*Etat11*. Alors, lorsque l'évènement "*event*" arrive, l'*Etat1* est désactivé et il se produit successivement les actions suivantes :



- *QuitterE11*
- *QuitterE1*
- *EntrerE2*
- *EntrerE21*
- *EntrerE22*,

et l'état actif est l'*Etat22* (les états composites *Etat21* et *Etat2* sont aussi actifs).
Four à micro-ondes :

[3] l'état composite peut être appelé « **super-état** »



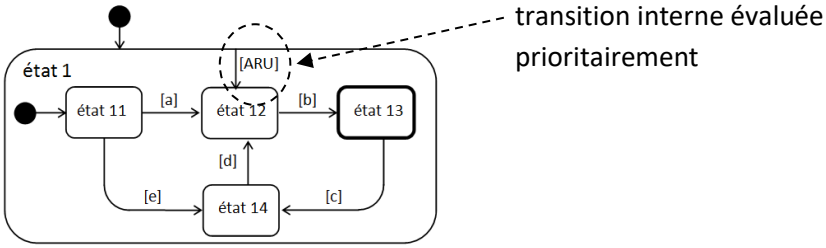
Dans cet exemple on met en évidence deux états composites "*Fermé*" et "*Ouvert*". L'état composite "*Fermé*" contient lui-même un sous-état composite "*Chauffe*". Il contient lui-même un sous-graphe formé de deux sous-états "*600 W*" et "*900 W*".

- Il peut y avoir des transitions ayant pour source / cible la frontière d'un sous-état ou la frontière d'un état composite.
- Quand plusieurs transitions sont possibles, on choisit de suivre celle qui part de l'état le plus en bas de la hiérarchie des états actifs, donc ici partant de "Chauffe" et non pas de "Fermé". Autrement dit :
  - Si l'état actif est "Chauffe" quand on ouvre la porte, l'état "En pause" s'active ;
  - Si l'état actif est "Off" contenu dans "Fermé" quand on ouvre la porte, l'état "Ouvert" s'active ainsi que l'état "Off" qu'il contient.

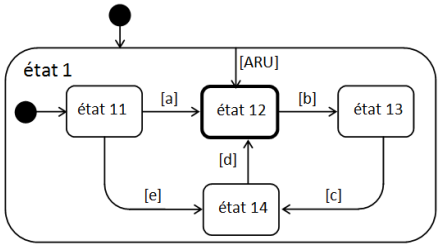
4.2.4) Les syntaxes particulières :

4.2.4.1) Transition interne : Elle relie le cadre d'un état donné au cadre d'un état de niveau hiérarchique inférieur. Cette transition de niveau hiérarchique supérieur est évaluée prioritairement.

Exemple : cas de forçage dans un état particulier.



Lorsque la condition *[ARU]* est vraie, l'*état 12* s'active et les autres états sont désactivés.



4.2.4.2) Etat « History » :

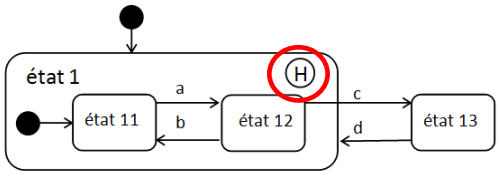
Il s'agit d'un pseudo-état qui mémorise l'activité d'un état composite.

Lors d'une désactivation, puis d'une transition retour vers cet état composite, il retrouve son état antérieur.

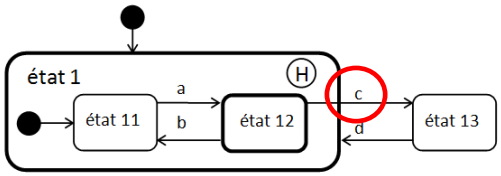
Notation : Ⓗ représente le pseudo-état « *History* » placé à l'intérieur de l'état composite ;

Exemple : voir ci-dessous.

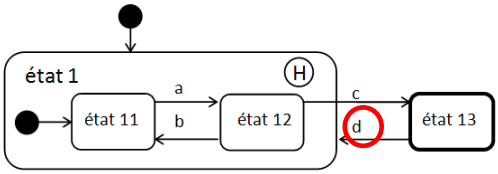
1.Le symbole «*H* » est présent dans l'*état 1* composite :



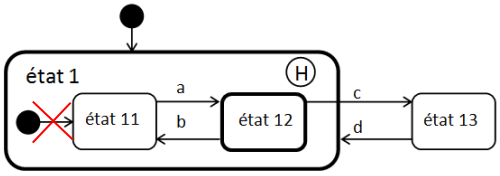
2. On suppose que l'évolution conduit à l'activation de l'*état 12* ; puis occurrence de l'événement «*c*»;



3. L'activation de l'*état 13* s'effectue normalement ; puis on suppose occurrence de l'événement « *d* »:

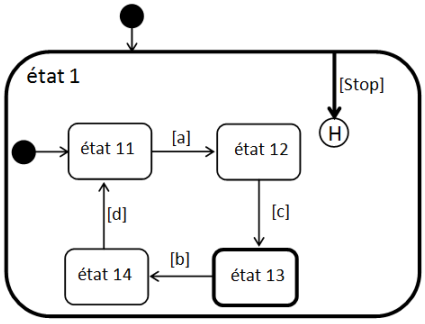


4. C'est alors l'*état 12* mémorisé par «*H* » qui s'active, au lieu de l'*état 11* qui aurait été activé si «*H* » n'avait pas été présent.



Application particulière :

Le « figeage » de l'état composite état *1* : La présence d'une transition interne vers le pseudo-état *history* conduit ici au figeage (ou blocage) du graphe de l'état composite *état 1* lors de l'activation de « *STOP* » ; dès que la condition « *STOP* » n'est plus vraie, l'état composite *état 1* reprend son mode d'évolution normal.

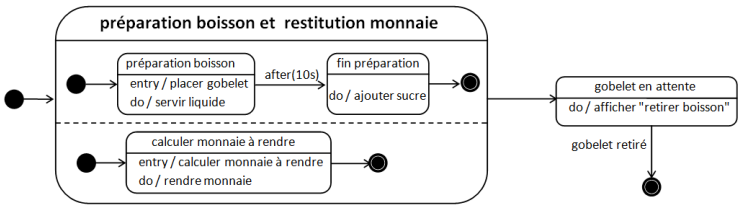


4.2.5) Etats orthogonaux, structures à évolutions simultanées :

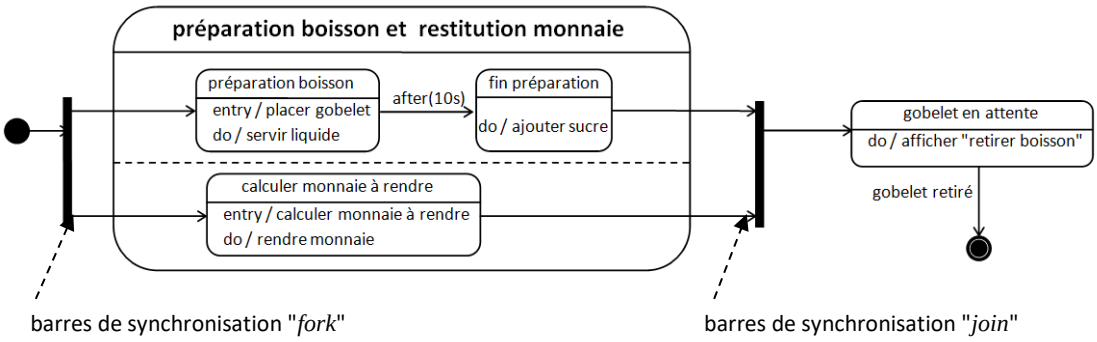
Un état **orthogonal** est un état composite incluant **plusieurs régions**, donc, comportant autant d'états actifs que de régions.

- Graphiquement, dans un état orthogonal, les différentes régions sont séparées par un trait horizontal ou vertical en pointillés allant d'un bord à l'autre de l'état composite.
- Chaque région peut posséder un état initial et final. 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 concurrentes.
- Toutes les régions concurrentes d'un état composite orthogonal doivent atteindre leur état final pour que l'état composite soit considéré comme terminé.
- La synchronisation est alors automatique et la transition de sortie de l'état composite est déclenchée.
- Des états parallèles sont aussi appelés « **concurrents** » (du mot anglais «concurrently» qui signifie « simultanément »)

Exemple :Distributeur de boissons :



Il est également possible de représenter ce type de comportement au moyen de transitions concurrentes constituées de barres de synchronisation "*fork*" et "*join*". Le graphe ci-dessous est une représentation équivalente à la précédente :



Remarque importante : Les transitions automatiques (sans évènement ou condition de garde) qui partent d'une barre de synchronisation "*fork*" se déclenchent en même temps. On ne franchit une barre de synchronisation "*join*" qu'après déclenchement de toutes les transitions qui s'y rattachent.