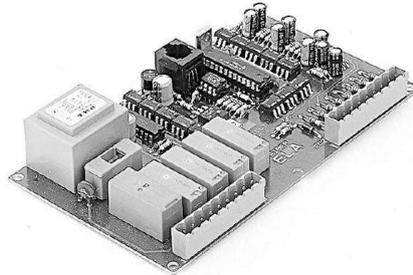


## Systemes à événements discrets

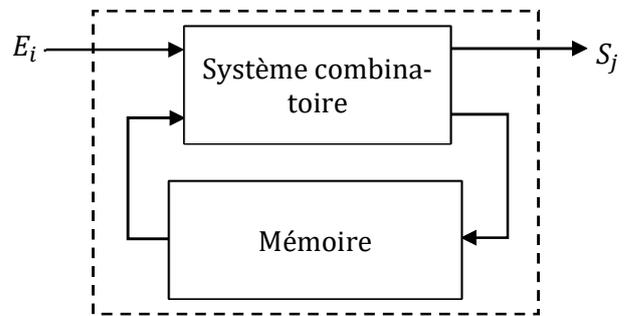
### Chronogrammes, diagrammes d'états, algorithme



Un système logique est dit **séquentiel** si les sorties  $S_j$  ne dépendent pas uniquement des  $E_i$  mais également de l'évolution antérieure du système.

- La même cause peut produire des effets différents : une même combinaison des variables d'entrée ne donne pas toujours la même sortie.
- Un effet peut rester maintenu alors même que sa cause a disparu : il y a mémorisation.

On différencie les systèmes à évènement discret, ou systèmes à logique séquentielle, des systèmes à logique combinatoire. Ces derniers sont caractérisés par le fait qu'une entrée, ou une combinaison d'entrée, engendre invariablement la même sortie. Il n'y a pas de notion de mémoire.



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

- Le **diagramme d'états** de SysML (State Machine) ;
- L'outil **algorithme** (ou algorithme).

Il s'agit essentiellement d'outils graphiques permettant de modéliser le comportement séquentiel en termes de déroulement d'actions temporelles. Ces outils sont, à la base, des outils de modélisation du comportement séquentiel, mais peuvent aussi servir à la programmation des composants réalisant la fonction **TRAITER** de la chaîne d'information (microcontrôleur, microprocesseur, automate programmable, ...).

Quel que soit l'outil adopté pour modéliser le comportement séquentiel d'un système, il existe souvent plusieurs solutions. La solution la plus simple qui respecte l'ensemble des contraintes est donc à privilégier. C'est le rôle de l'ingénieur de choisir le "bon" outil de description, et la "meilleure solution".

# 1. Diagrammes d'états

## 1.1. Constitution

### 1.1.1. Etat

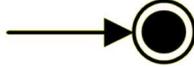
Dans un diagramme d'états, la loi d'évolution des états n'est évidemment pas aléatoire. Cette loi est choisie par le créateur du graphe afin que celui-ci remplisse la fonction précise.

Chaque état est dessiné sous la forme d'un rectangle aux coins arrondis, contenant son nom.



Un état représente une période de la vie du système. Pendant cette période, le système accomplit une ou plusieurs actions, ou attend un évènement. Il peut être actif ou non ; plusieurs états peuvent être actifs en même temps dans le même système.

On distingue également :

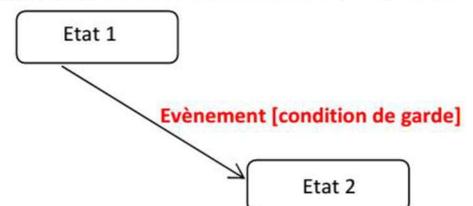
<p><b>Etat initial</b> : pseudo-état qui indique un point d'entrée dans un graphe.</p> 	<p><b>Etat final</b> : non obligatoire, il indique que le système décrit n'a plus d'état actif.</p> 
---	--

### 1.1.2. Transition, évènement et condition de garde

Une transition représente le passage instantané d'un état vers un autre. Une transition ne peut donc pas avoir de durée. On appelle état source l'état de départ d'une transition et état **destination** l'état d'arrivée.

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 franchissement de la transition.



Une transition peut aussi être automatique, lorsqu'on ne spécifie pas l'évènement qui la déclenche.

En plus de spécifier un évènement précis, il est aussi possible de conditionner une transition, à l'aide d'une "**condition de garde**" : il s'agit d'une **expression booléenne** encadrée de crochets, évaluée lorsque l'état précédant la transition est vrai et que l'évènement déclencheur se produit. Si la condition de garde est vraie, la transition est alors franchie, sinon elle ne l'est pas et l'évènement est perdu.

Remarque : différence entre évènement et condition de garde :

- un **évènement** 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) ;

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

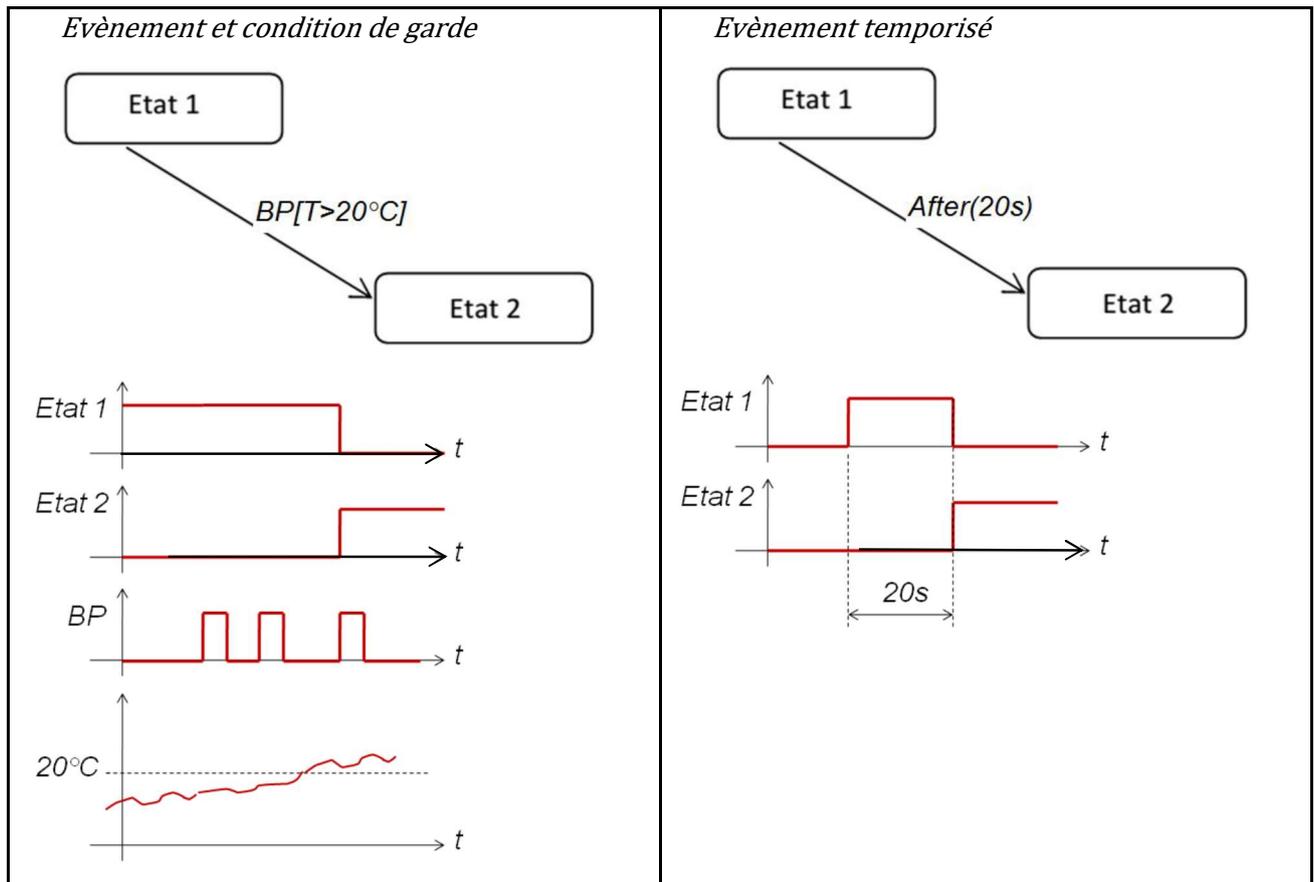
- une **condition de garde** n'est pas datée, elle doit être vraie (niveau logique 1) à l'instant où l'évènement survient pour que la transition soit franchie.

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

Il existe 3 sortes d'évènements :

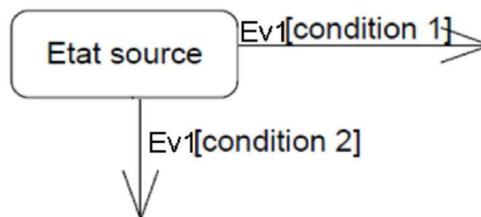
- **évènement signal** : un signal est émis à destination d'un objet ; cette émission est asynchrone, c'est-à-dire que le destinataire ne l'attend pas, et qu'elle peut survenir n'importe quand.  
*Exemple : l'appui sur un bouton-poussoir.*
- **évènement temporisé** : un évènement de ce type fait intervenir le temps. Il nécessite l'utilisation des mots réservés **at(date)** pour spécifier un temps absolu, ou **after(durée)** pour spécifier une durée à partir de l'instant d'activation de l'état précédent.
- **évènement de changement** : une valeur a changé de telle sorte que la transition est franchie : **when[valeur = 5]**.

*Exemples :*



### 1.1.3. Transition conditionnelle

Plusieurs transitions peuvent quitter un même état. **Une seule d'entre elles doit être déclenchée ; les évènements et/ou les conditions de garde doivent donc être exclusives.**



### 1.1.4. Actions

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

- **entry/** est suivi des actions exécutées lorsque l'état devient actif ;
- **do/** est suivi d'une ou plusieurs actions exécutées dans l'ordre de leur écriture, à partir de l'instant où l'activité /entry est terminée ;
- **exit/** est suivi des actions qui se déroulent lorsque l'état se désactive ;

Pendant que l'état est actif, un évènement peut lancer une action avec la syntaxe : évènement/suivi de l'action. Cette action est lancée chaque fois que l'évènement survient, tant que l'état est actif.

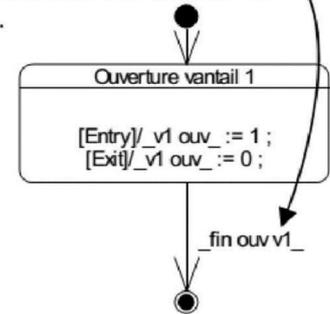
#### Remarques :

- On peut aussi ne pas utiliser de mot réservé, auquel cas cela correspond à un do/.
- Un état peut ne pas contenir d'action. Il sert alors à attendre le déclenchement de la transition suivante.

Une action peut être :

- un ordre de mise à 1 d'une sortie ; la syntaxe est alors **SORTIE := 1 ;**
- un ordre de mise à 0 d'une sortie ; la syntaxe est alors **SORTIE := 0 ;**
- une modification d'une variable numérique interne, par exemple un compteur C ; la syntaxe est alors **C := C + 1 ;** pour incrémenter la valeur du compteur C de 1.

Cet évènement met fin à l'état "Ouverture vantail 1", ce qui met à 0 la commande du moteur du vantail 1.



### 1.1.5. Entrées/Sorties

Les **informations en entrée** vont intervenir dans les **transitions** du diagramme d'états.

Les **ordres de commande en sortie** vont intervenir dans les **actions lancées dans les états actifs**.

### 1.1.6. Exemple 1 : Lave-linge

On peut ici commencer par déterminer les états : **Prélavage, Lavage, Rinçage, Essorage** et bien sûr, **Arrêt**.

On complète le graphe en plaçant des transitions entre les états, pour indiquer la loi d'évolution des états en fonction des entrées. On se sert pour cela du comportement séquentiel souhaité ou observé.

Les variables d'entrée sont ici les informations suivantes :

- **m** : bouton marche/arrêt du lave-linge ; m = 1 indique marche.
- **p** : bouton qui indique si le programme du lavage sélectionné par l'utilisateur comporte ou non une phase de prélavage.

Les durées des différentes étapes du lavage sont fixées par le constructeur :

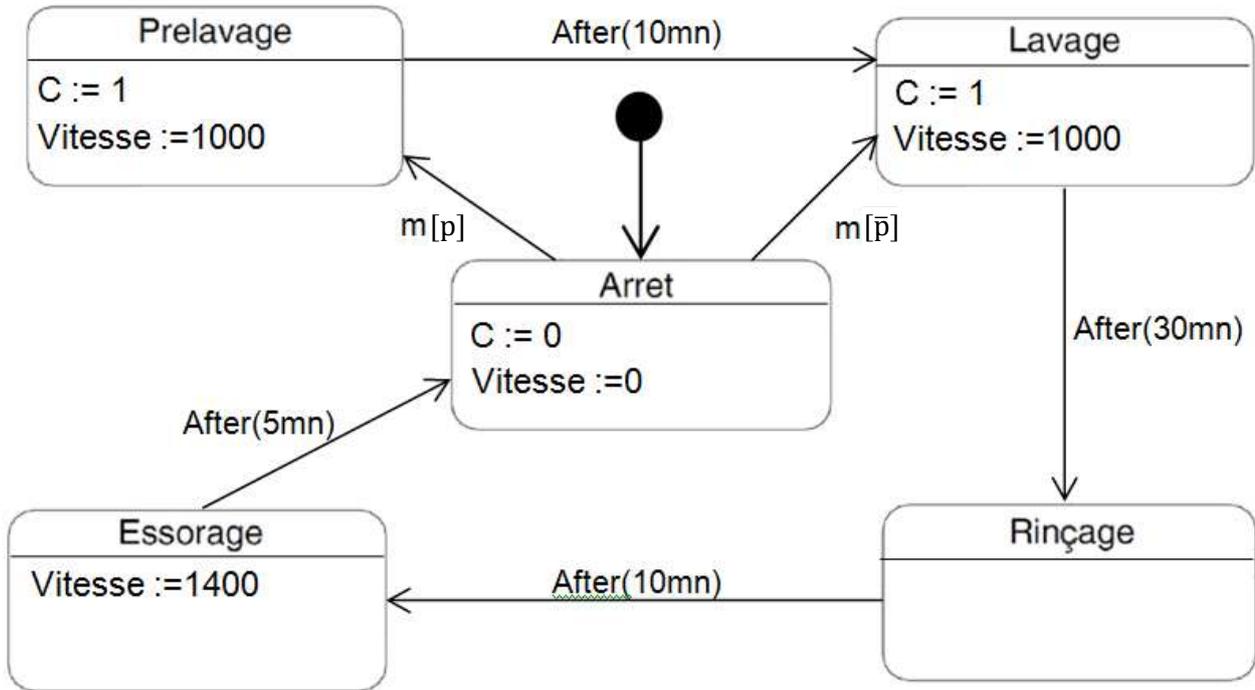
- prélavage : 10 minutes ;
- lavage : 30 minutes ;
- rinçage : 10 minutes ;
- essorage : 5 minutes.

Avec ces données, on peut compléter le diagramme des états avec les transitions et les évènements déclencheurs.

Il reste ensuite à définir les variables de sorties, pour lancer les actions :

- **C** : égale à 1 si le moteur doit tourner, sinon 0 ;
- **Vitesse** : égale à 0 à l'arrêt ; 1 000 tr/min en pré-lavage, en lavage et en rinçage ; 1 400 tr/min en essorage.

On obtient le graphe suivant :



### 1.1.7. Exemple 2 : porte de garage

Fonctionnement souhaité :

Une porte de garage basculante est mise en mouvement par un moteur à 2 sens de rotation : **Ouvrir** ou **Fermer**.



Boîtier de commande mural



Bouton-poussoir tel



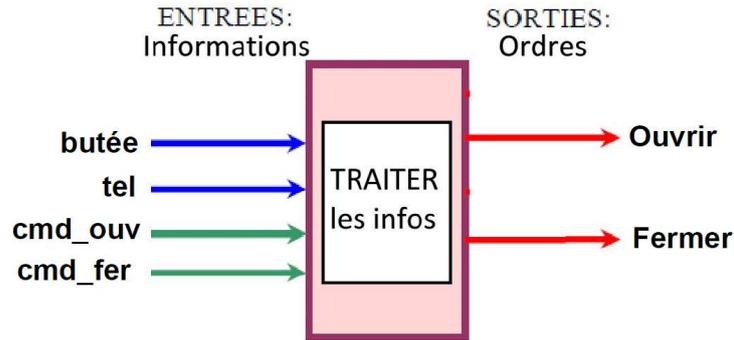
Un capteur **buté** détecte une surintensité moteur, qui correspond à l'atteinte d'une position en butée de la porte : ouverte ou fermée. Une télécommande possède 1 bouton **tel** dont l'appui produit le fonctionnement suivant :

- **Ouvrir** si la porte est fermée ;
- **Fermer** si elle est ouverte ;
- inverser le sens si la porte est en mouvement.

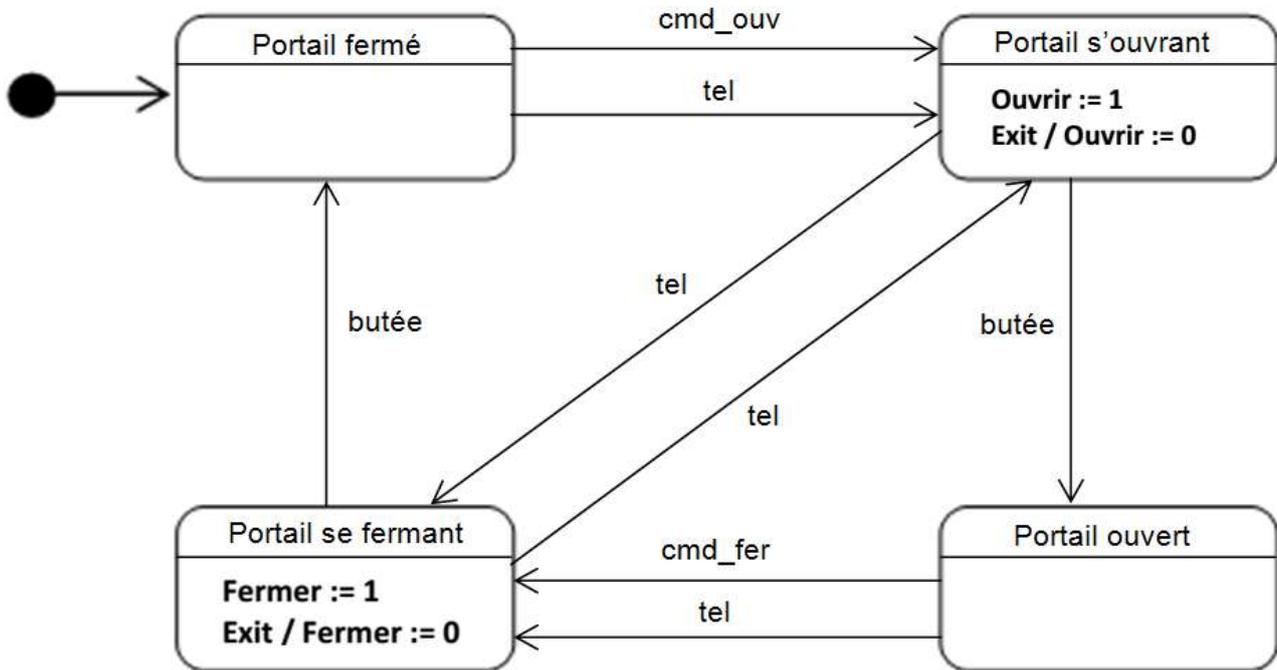
Enfin, un boîtier de commande mural dans le garage possède 2 boutons poussoirs **cmd\_ouv** et **cmd\_fer**.

On suppose qu'à la mise sous tension, la porte est en position fermée.

Liste des entrées-sorties :



*Proposition de diagramme d'états :*



## 1.2. Synchronisation et hiérarchisation des états

### 1.2.1. Etat composite

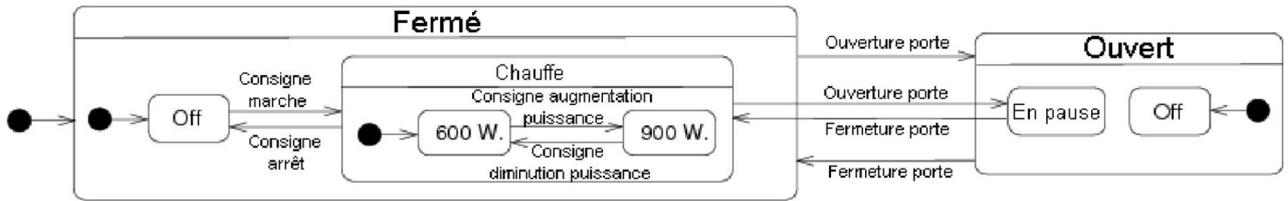
Lorsque le comportement à décrire risque d'être peu lisible car trop complexe ou fait apparaître des parties "séparables" de l'ensemble, alors on extrait ces parties pour en faire des sous-graphes séparés, hiérarchiquement inférieurs au graphe principal.

On englobe ces sous-graphes dans un état appelé "état composite" ou "super-état".

**Un état composite est composé de plusieurs sous-états internes. Un état composite possède un état initial.**

*Exemple :*

Comportement d'un four micro-ondes

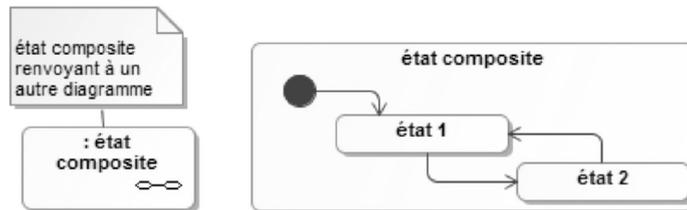


- Il y a 2 états composites "Fermé" et "Ouvert". L'état composite "Fermé" contient lui-même un sous-état composite "Chauffe". Il contient lui-même un sous-graphe formé de 2 sous-états "600 W" et "900 W".
- Il peut y avoir des transitions ayant pour source/cible la frontière d'un sous-état ou la frontière 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 et donc l'état "Off" qu'il contient.

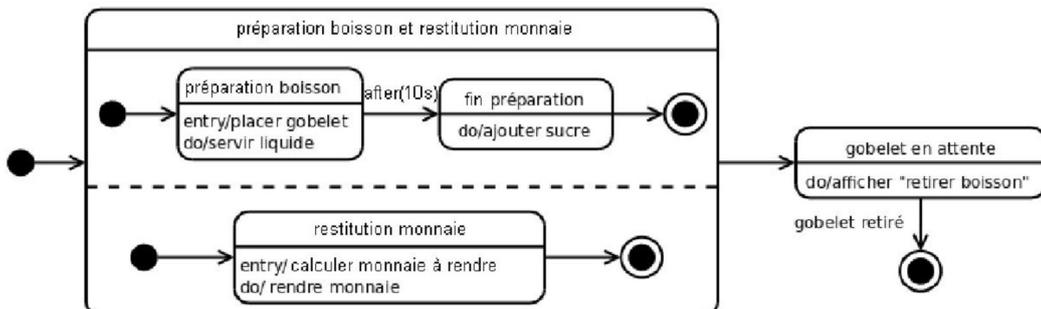
Il existe une notation abrégée pour un état composite :



1.2.2. Concurrency

Dans un état composite, plusieurs graphes d'états peuvent évoluer simultanément (en parallèle). On dit qu'il y a concurrence de plusieurs états.

*Exemple : distributeur de boissons*



L'état composite est dit **orthogonal** car il comporte **plus d'une région**, chaque région représentant un flot d'exécution. 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.

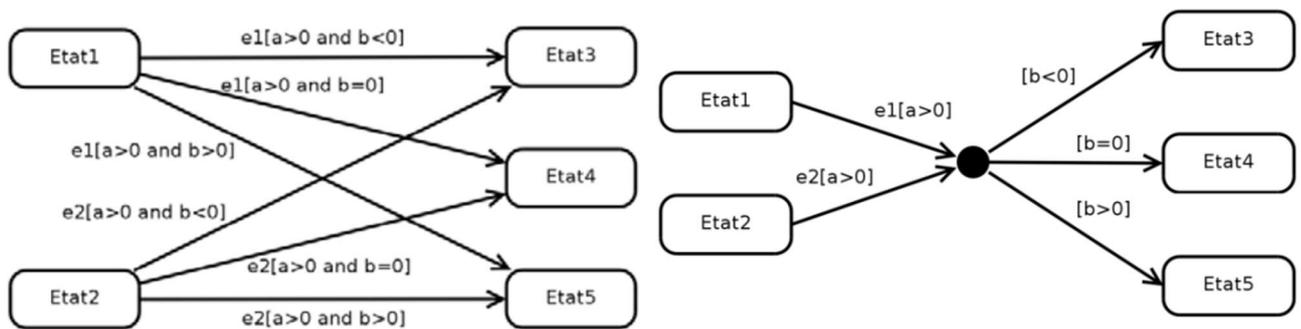
### 1.2.3. Les pseudo-états

Un pseudo état un élément de commande qui influence le comportement d'une machine d'état mais qui n'a pas d'activité.

- Jonction ●

Ce pseudo-état sert à regrouper (factoriser) des conditions.

Exemple :

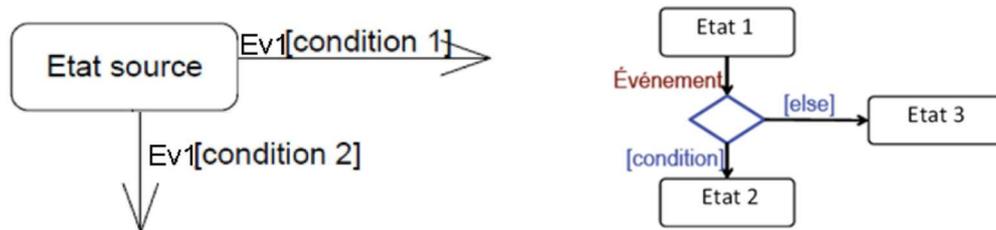


L'état actif passe de l'état 1 à l'état 3 si l'évènement e1 survient et que les conditions de gardes  $a > 0$  et  $b < 0$  sont vérifiées.

- Décision ◇

Il s'agit d'une décision sur une condition. Les conditions de gardes doivent être exclusives.

Ce pseudo-état est assez équivalent à la transition conditionnelle



Remarque : Par définition, les états initiaux et finaux sont des pseudo-états.

## 2. Algorithme

### 2.1. Définitions

**Algorithme** : c'est l'ensemble de règles opératoires ordonnant à un processeur d'exécuter dans un ordre déterminé un nombre d'opérations élémentaires.

**Algorithme** : c'est une représentation graphique de l'algorithme utilisant des symboles normalisés.

### 2.2. Constitution d'un algorithme

SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	début ou fin d'un algorithme		<p><b>Test ou Branchement conditionnel</b></p> <p>décision d'un choix parmi d'autres en fonction des conditions</p>
	symbole général de « <b>traitement</b> » opération sur des données, instructions, ... ou opération pour laquelle il n'existe aucun symbole normalisé		<p><b>sous-programme</b></p> <p>appel d'un sous-programme</p>
	entrée / sortie		<p><b>Liaison</b></p> <p>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</p>
	commentaire		