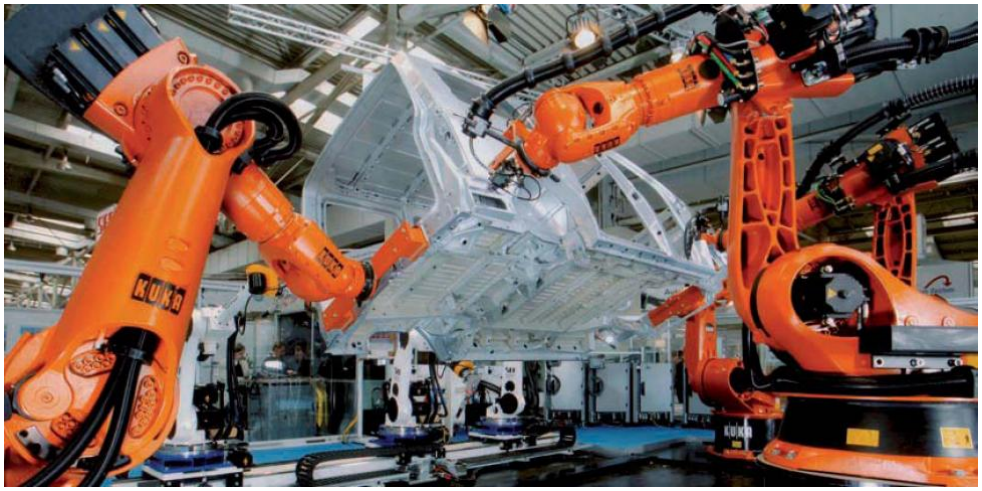
Systèmes à évènements discrets

Lycée Jean Zay – 21 rue Jean Zay – 63300 Thiers – Académie de Clermont-Ferrand



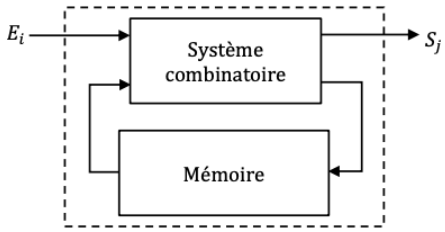
Compétences visées:

B2-20 Décrire le comportement d'un système séquentiel.

Table des matières

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.



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 algorigramme).

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".

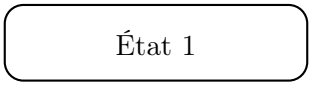
1 Diagrammes d'états

1.1 Constitution

1.1.1 État

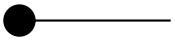
Dans un diagramme d'états, la loi d'évolution des états n'est évidemment pas aléatoire. Cette loi est choisie par le créateur du graphe afin que celui-ci remplisse la fonction précise.

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

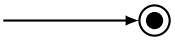


Un état représente une période de la vie du système. Pendant cette période, le système accomplit une ou plusieurs actions, ou attend un évènement. Il peut être actif ou non ; plusieurs états peuvent être actifs en même temps dans le même système. On distingue également :

État initial pseudo-état qui indique un point d'entrée dans un graphe.

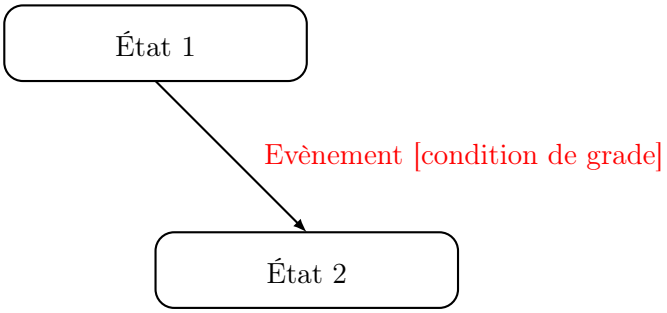


État final non obligatoire, il indique que le système décrit n'a plus d'état actif.



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'événement est perdu.

Remarque

différence entre évènement et condition de garde :

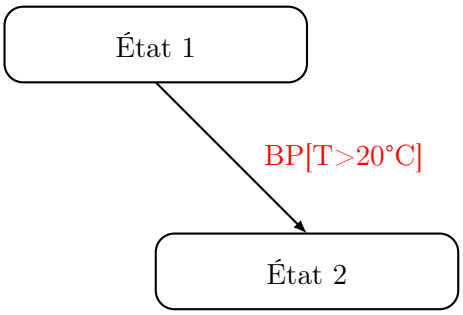
- Un **événement** est parfaitement daté dans le temps, il correspond par exemple à un passage d'une variable de 0 à 1 à un instant précis (front montant) ;
  Exemple d'évenements : appui sur un bouton-poussoir, capteur fin de course atteint, etc.
- Une **condition de garde** n'est pas datée, elle doit être vraie (niveau logique 1) à l'instant où l'événement survient pour que la transition soit franchie.
  Exemple de condition de grade : vitesse du véhicule non nulle, température>20°C, etc.

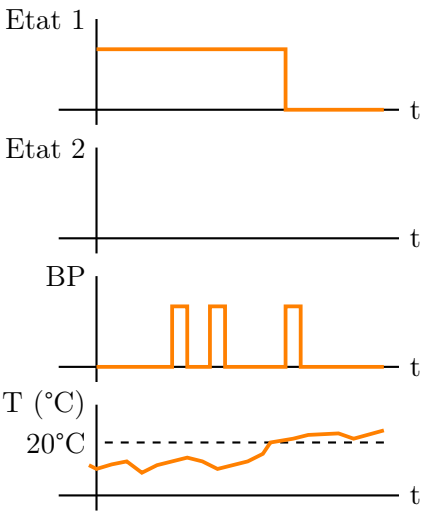
Il existe 3 sortes d'évènements :

- **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]**.

Exemple :

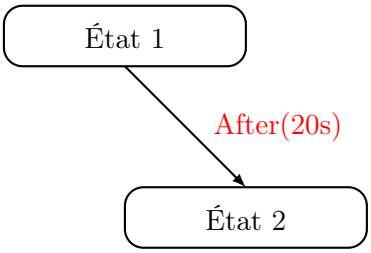
Événement et condition de grade

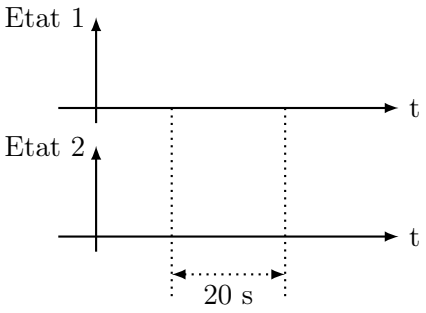




Exemple :

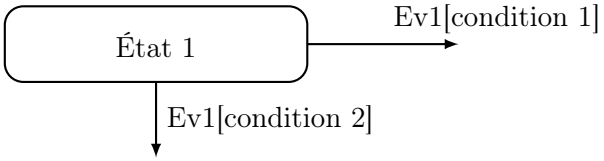
Événement temporisé





1.1.3 Transition conditionnelle

Plusieurs transitions peuvent quitter un même état. Une seule d'entre elles doit être déclenchée ; les événements 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 de 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.

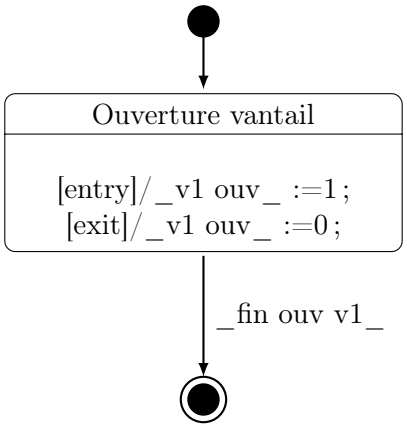
Remarque

- 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.

Exemple :



Cet événement met fin à l'état "ouverture vantail", ce qui met à 0 la commande du moteur.

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**.

Exemple : *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.

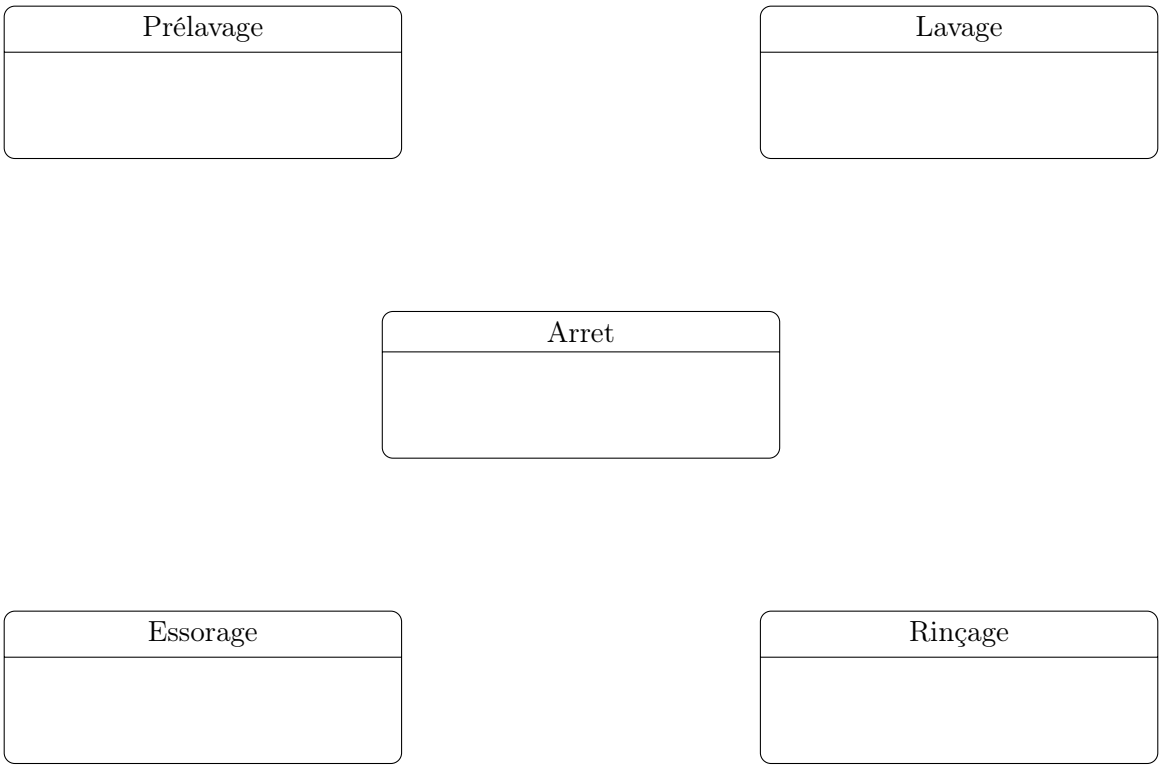
Les durées des différentes étapes du lavage sont fixées par le constructeur :

- **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 événements 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 prélavage, en lavage et en rinçage ; 1 400 tr/min en essorage.

On obtient le graphe suivant :



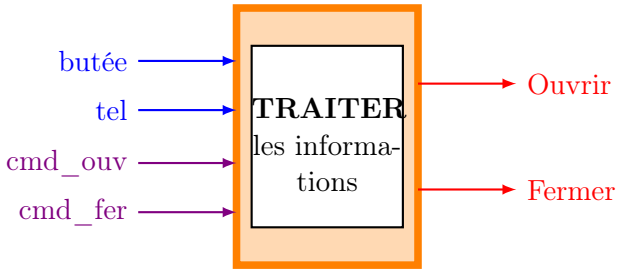
Exemple : *Porte de garage*

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



On obtient le graphe suivant :

Porte fermée	Porte s'ouvrant

Porte se fermant	Porte ouverte

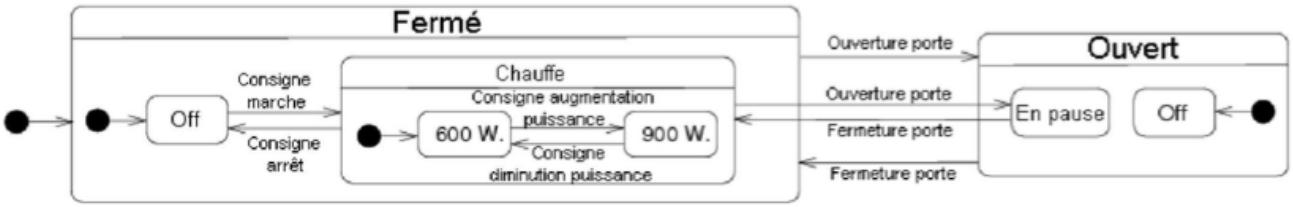
1.2 Synchronisation et hiérarchisation des états

1.2.1 État 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*

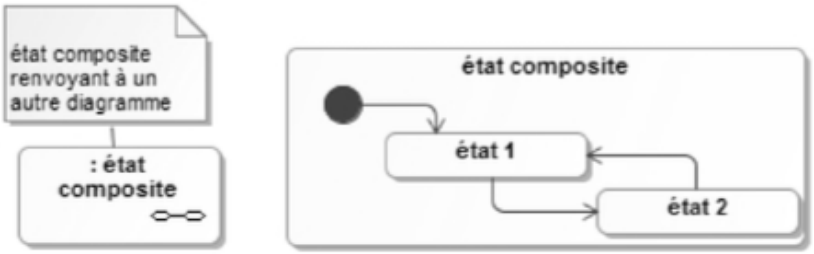


- Il y a 2 états composites "Fermé" et "Ouvert". L'état composite "Fermé" contient lui- même un sous-état composite "Chauffe". Il contient lui-même un sous-graphe formé de 2 sous-états "600 W" et "900 W".
- Il peut y avoir des transitions ayant pour source/cible la frontière d'un sous-état ou la frontière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

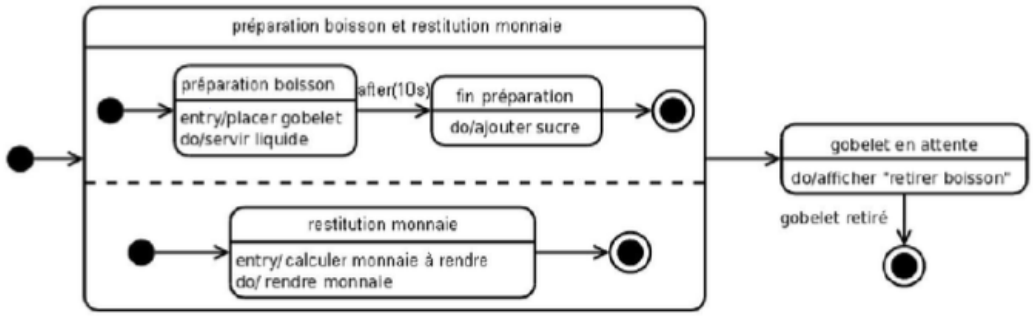
Il existe une notation abrégée pour un état composite :



1.2.2 Concurrence

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.

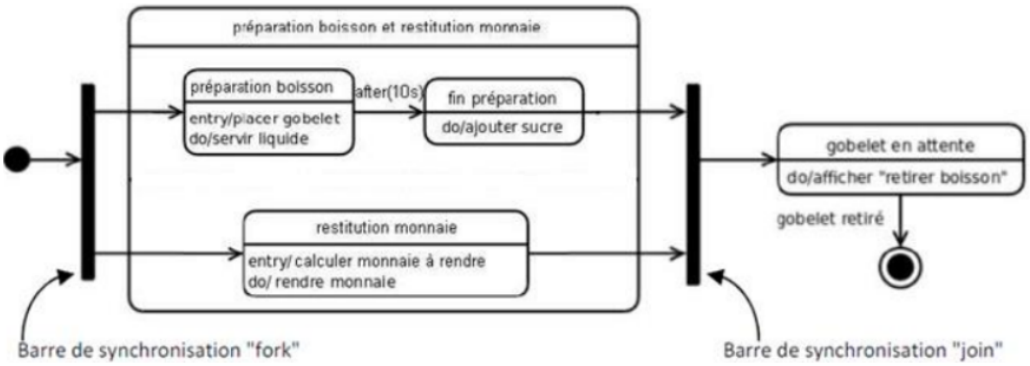
Chaque région peut posséder un état initial et final. Une transition qui atteint la bordure d'un état composite orthogonal est équivalente à une transition qui atteint les états initiaux de toutes ses régions concurrentes.

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 Barres de transitions « fork » et « join »

Les barres de transition « fork » et « join » permettent de réaliser des divergences et convergences de séquences parallèles.

Exemple : *Distributeur de boissons*



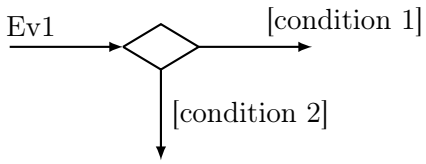
Le graphe ci-dessus est une représentation équivalente à la précédente avec des états composite à régions orthogonales :

- - 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.

1.2.4 Le point de décision « choice »

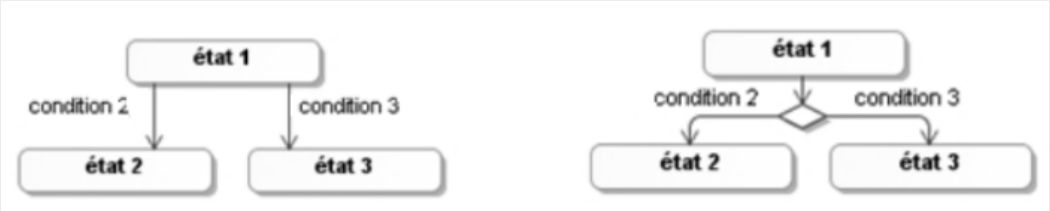
Le point de décision « choice » permet de réaliser la sélection et convergence de séquences exclusives. Il est nécessaire qu'une condition située en aval soit vraie pour que l'évolution du système se poursuive. Les

conditions de gardes doivent être exclusives. Les conditions de garde situées en aval sont toutes évaluées une fois le pseudo-état atteint.



Le mot clé « else » peut-être utilisé pour englober tout ce qui n'est pas décrit dans les autres expressions booléennes.

Remarque



Les 2 diagrammes ci-dessus ne sont pas équivalents.

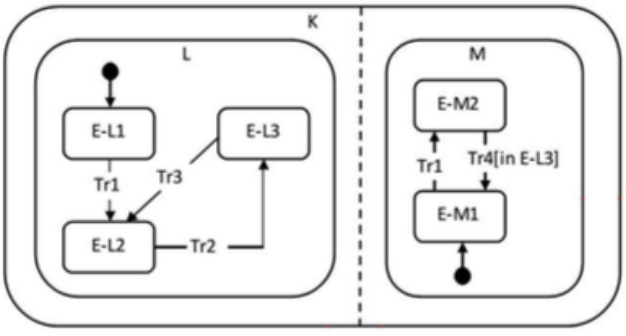
Dans le premier diagramme, dès que la l'évènement « condition 2 » ou « condition 3 » est vrai, on passe de l'état 1 à l'état 2 ou à l'état 3 sans terminer l'éventuelle activité de l'état 1.

Dans le second diagramme, on quitte l'état 1 après la fin de l'éventuelle activité associée (pas d'évènement associé à la transition) pour s'arrêter au pseudo-état « choice ». Ensuite, lorsque la condition 2 est satisfaite, le système passe dans l'état 2. Si la condition 3 est satisfaite, le système passe dans l'état 3.

1.2.5 Synchronisation de deux graphes

Il est aussi possible, et très utile, de conditionner une transition par l'activité de l'état d'un autre sous-graphe.

Exemple :



L'état composite orthogonal K est composé des sous-états L et M.

L et M s'activent simultanément, et évoluent indépendamment, en parallèle. Au départ, EL1 et E-M1 sont actifs. Pour que M passe d'E-M2 à E-M1, il faut que l'évènement Tr4 survienne, à condition que E-L3 soit actif. Cet évènement est conditionné par l'activité de l'état E-L3 de l'autre sous-graphe.

La syntaxe de la condition de garde vérifiant l'activité de ETAT est : **[in ETAT]**.

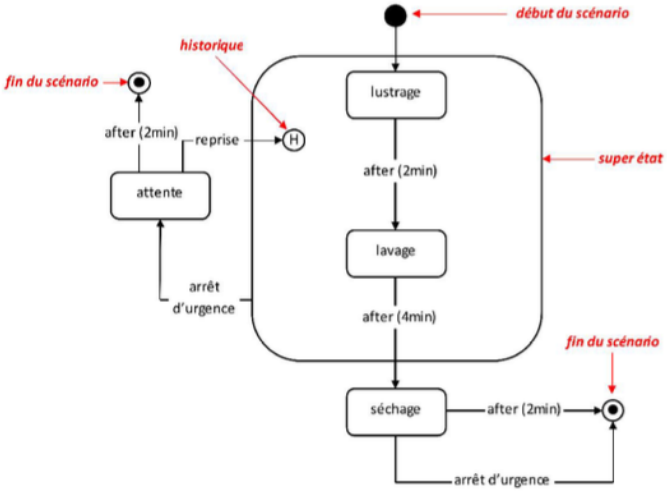
1.2.6 Historique

Il est possible de mémoriser le dernier sous-état actif d'un super-état, pour y revenir directement ultérieurement. On utilise pour cela le symbole de modélisation "historique".

Exemple : *Machine à laver les voitures*

En phase de lustrage ou de lavage, le client peut appuyer sur le bouton d'arrêt d'urgence. S'il appuie sur ce bouton, la machine se met en attente. Il a alors deux minutes pour reprendre le lavage ou le lustrage (la machine continue en phase de lavage ou de lustrage, suivant l'état dans lequel elle a été interrompue), sans quoi la machine s'arrête.

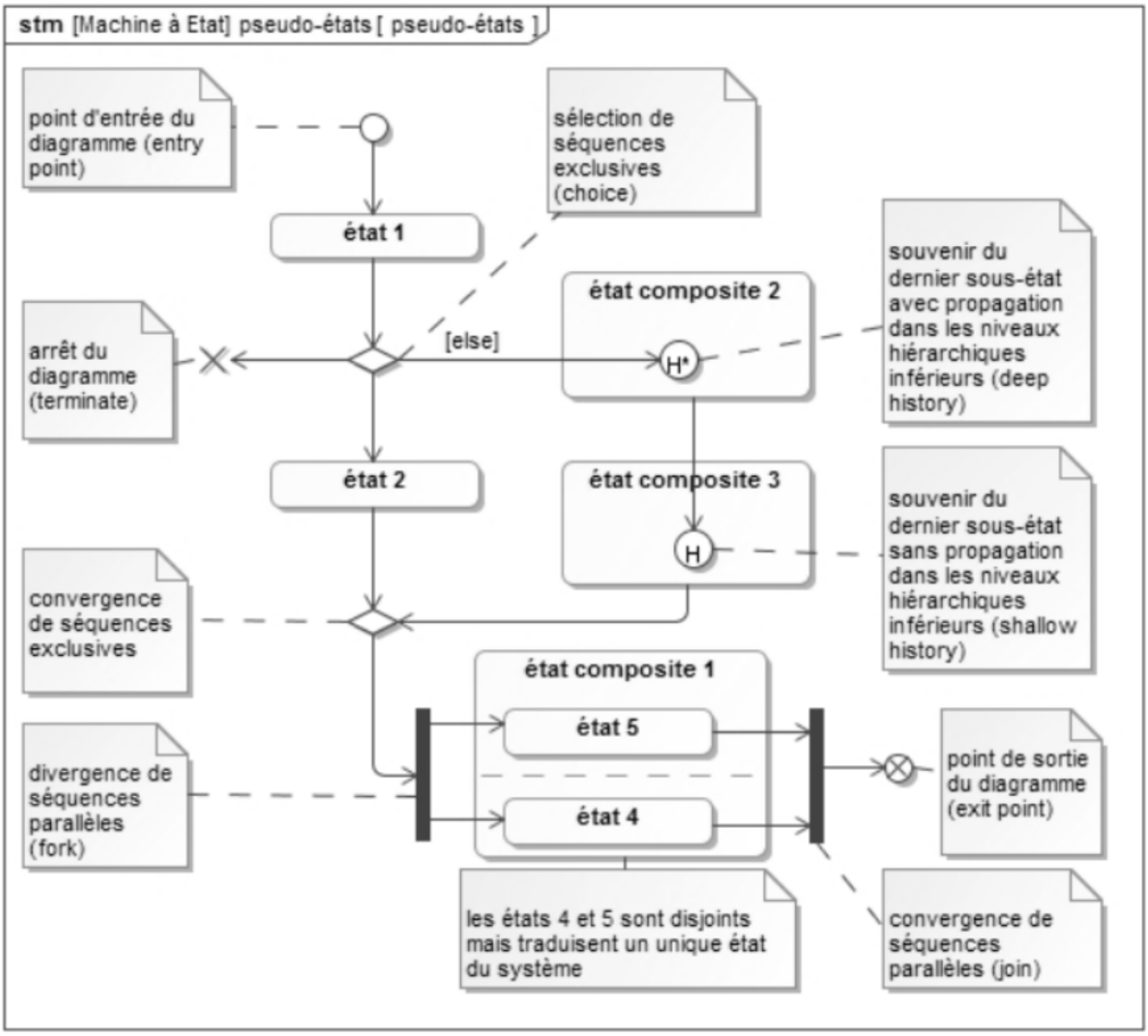
En phase de séchage, le client peut aussi interrompre la machine. Mais dans ce cas, la machine s'arrêtera définitivement (avant de reprendre un autre cycle entier).



1.2.7 Résumé - Les 9 pseudo-états

Les pseudo-états sont des éléments de commande qui influencent le comportement d'une machine d'état. Ils peuvent être utilisés dans un diagramme d'états ou dans un diagramme d'activité. Le formalisme SysML admet neuf pseudo-états que l'on retrouve dans le diagramme d'après :

- « **fork** » et « **join** » : déjà évoqués plus haut,
- « **choice** » : déjà évoqué plus haut.
- « **junction** » : idem que le pseudo-état « choice », à la différence que 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,
- « **entry point** » et « **exit point** » : permet de créer un point d'entrée du diagramme et un point de sortie vers un autre diagramme,
- « **terminate** » : permet de terminer une séquence sans destruction de l'instance de bloc.
- « **shallow history** » : permet à un état de niveau hiérarchique supérieur (état composite) de se souvenir du dernier sous-état, avant qu'il n'évolue vers un autre état,
- « **deep history** » : idem que précédemment mais avec la propagation de l'historique à tous les sous-états composites de niveaux hiérarchiques inférieurs,



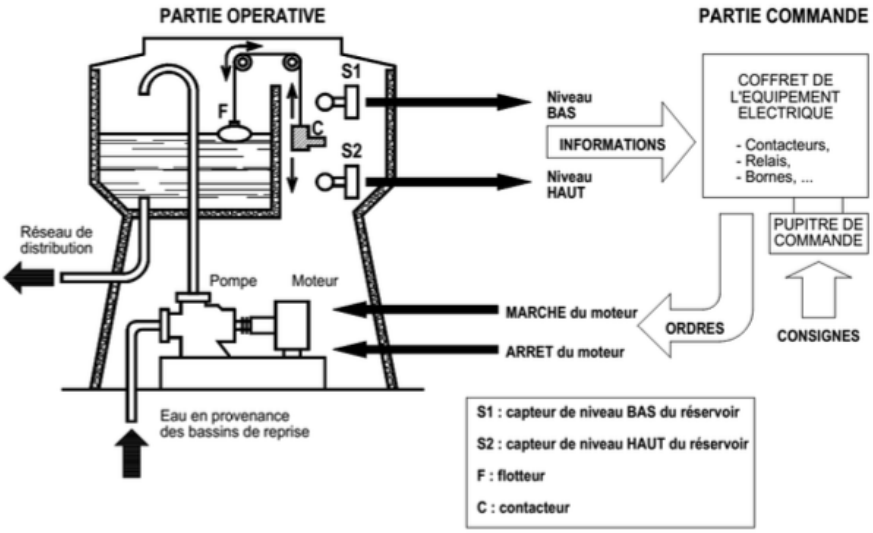
1.3 Application - Château d'eau

Après une mise en service de l'installation, le système fonctionne automatiquement de la façon suivante :

1. Le moteur actionne la pompe pour aspirer l'eau en provenance des bassins de reprise. Le réservoir du château d'eau se remplit. Le flotteur F se déplace vers le haut en suivant le niveau d'eau dans le réservoir. Le contacteur C relié au flotteur se déplace vers le bas.
2. Dès que le niveau haut du réservoir est atteint, le contacteur C bute contre le capteur de niveau haut S2 qui envoie une information à la partie commande.
3. La partie commande envoie un ordre à la partie opérative entraînant l'arrêt du moteur et de la pompe. Le pompage de l'eau s'arrête.

Remarque

les usagers utilisant l'eau pour leur besoin, le niveau d'eau diminue dans le réservoir.



4. La pompe n'étant plus en action et les usagers utilisant l'eau, le réservoir se vide. Le flotteur F se déplace vers le bas en suivant le niveau d'eau dans le réservoir. Le contacteur C entraîné par le flotteur se déplace vers le haut.
5. Dès que le niveau bas du réservoir est atteint, le contacteur C bute contre le capteur de niveau bas S1 qui envoie une information à la partie commande.
6. La partie commande envoie un ordre à la partie opérative entraînant la mise en marche du moteur et de la pompe. Le réservoir se remplit et le cycle recommence.

Q1 Établir le diagramme d'état correspondant à ce fonctionnement.