SYSTEMES A EVENEMENTS DISCRETS (S.E.D.)

L'électronique microprogrammée ou la programmation de micro-ordinateurs dédiés permettant de piloter des systèmes automatisés réalisant des tâches ou successions de tâches plus ou moins complexes fait appelle à de nombreux langages informatiques distincts plus ou moins élaborés dépendant des capacités du CPU, de la mémoire et de sa gestion, des interfaces d'entrée-sortie et des évènements ainsi gérés en temps réel.
On utilise un langage graphique simple pour ne pas avoir à apprendre à utiliser des langages informatiques dédiés à chaque partis commande très différentes selon les systèmes. Dans le laboratoire de SII, les langages de programmation utilisés sont tous différents sur les supports (NAO, Capsuleuse, Bras Beta, Trieuse, Cordeuse)
On définit donc quelques concepts fondamentaux et on décrit graphiquement le programme à implanter dans la cible (partis commande du système) à l'aide du diagramme d'état.

I) DEFINITION

Rappels :
Un circuit logique est **combinatoire** si à chaque combinaison des variables d'entrée (ou informations d'entrée) correspond **instantanément** une et une seule combinaison des variables de sortie, autrement dit : la même cause produit toujours le même effet (une même combinaison des entrées donne toujours la même sortie S) et l'effet disparaît dès que la cause disparaît. $S = f(e_1, e_2, \ldots, e_n)$

Un circuit logique est **séquentiel** si à une combinaison des variables d'entrée ne correspond pas toujours la même combinaison des variables de sortie. Il est nécessaire de prendre en compte la succession dans le temps (ou la séquence) des combinaisons des variables d'entrée. Un circuit séquentiel possède donc une fonction mémoire.
Conséquences : une même cause peut produire des effets différents (une même combinaison des entrées ne donne pas toujours la même sortie S) ; un effet peut rester maintenu alors même que sa cause a disparu. $S = f(e_1, e_2, \ldots, e_n,$ **instant t**$)$

II) OUTILS DE BASE

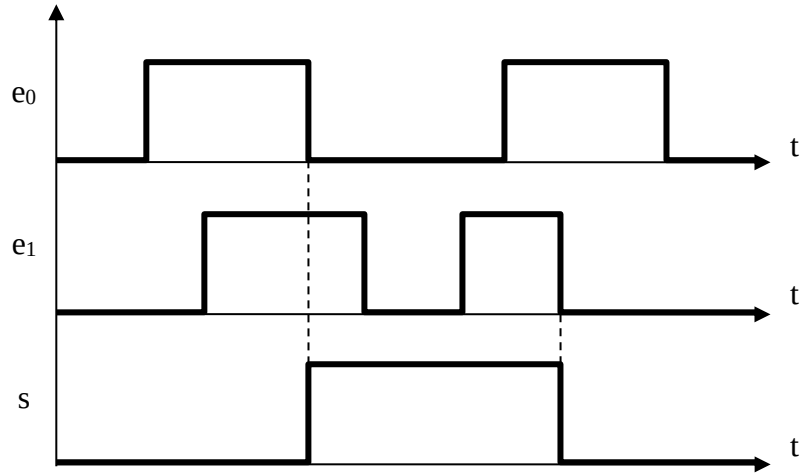
1. Chronogramme

2. Front montant et front descendant

3. Variante du chronogramme : le diagramme de Gantt

4. Exemple

N.B. : e_0 et e_1 sont des entrées et S la sortie



Il s'agit s'un système séquentiel, car (compléter) : ……..

……

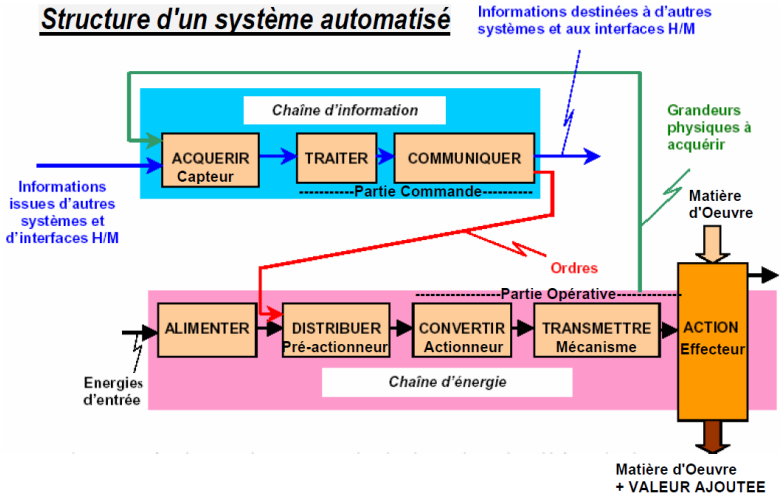
III) DIAGRAMME D'ETATS

1. But

Le diagramme d'états est un diagramme normalisé **SysML**. Il est noté **stm** (pout State Machine = machine d'états)

Le graphe d'états, comme l'algorigramme (cf. cours d'informatique), est essentiellement un outil graphique permettant de **modéliser** le comportement séquentiel, en termes de déroulement d'actions temporelles.

Mais il peut aussi servir à **programmer** les composants réalisant la fonction "Traiter" de la chaîne d'information (microcontrôleur, microprocesseur, automate programmable, …). Les variables d'entrée de la fonction "Traiter" sont alors les informations fournies par la fonction "Acquérir" (capteurs, IHM,…) et les variables de sortie sont les ordres pour la fonction "Distribuer" de la chaîne d'énergie, éventuellement via la fonction "Communiquer".



2. Constitution de base

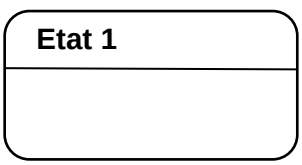
a. Etat

Dans un graphe 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.

• Il existe différentes sortes d'états :

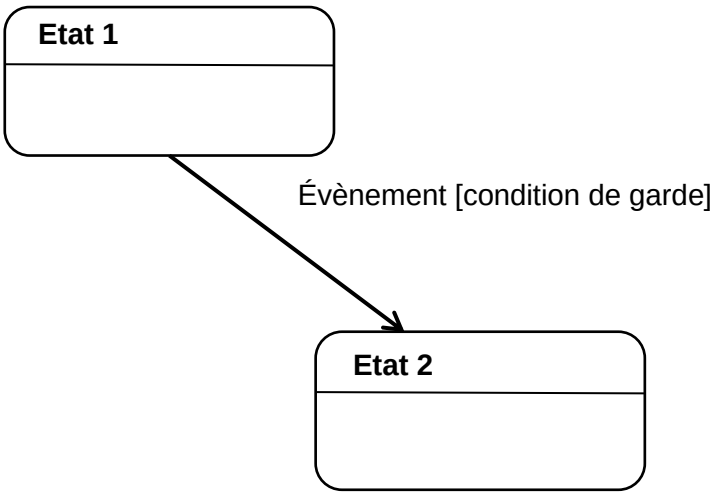
- l'état initial : pseudo-état qui indique un point d'entrée dans un graphe ;
- l'état final : non obligatoire, il indique que le système décrit n'a plus d'état actif ;
- état : un état représente une période de la vie du système. Pendant cette période, le système accomplit une ou plusieurs actions, ou attend un évènement. Il peut être actif ou non.



b. Transition, évènement et condition de garde

Une transition représente le passage instantané d'un état vers un autre.

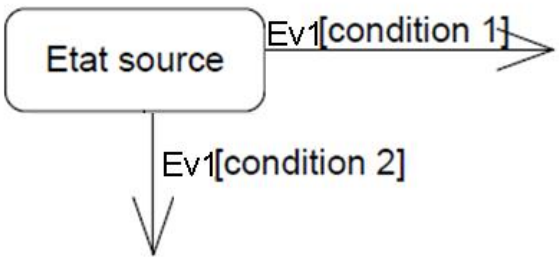
Une transition ne peut donc pas avoir de durée. On appelle état source l'état de départ d'une transition et état destination l'état d'arrivée.



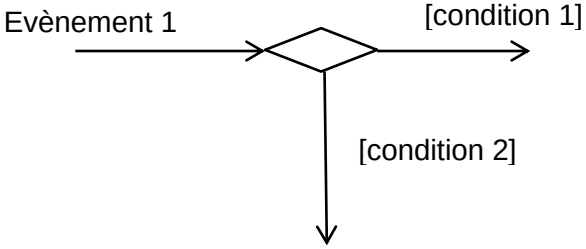
• Une transition n'est évaluée que si l'état source est actif.
• Une transition est déclenchée par un évènement. En d'autres termes : c'est l'arrivée d'un évènement qui conditionne le franchissement de la transition.
• Une transition peut aussi être automatique, lorsqu'on ne spécifie pas l'évènement 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.
Rq : 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 franchie.
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.

• Il existe 2 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. Par 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 when(date) pour spécifier un temps absolu, ou after(durée) pour spécifier une durée à partir de l'instant d'activation de l'état précédent.

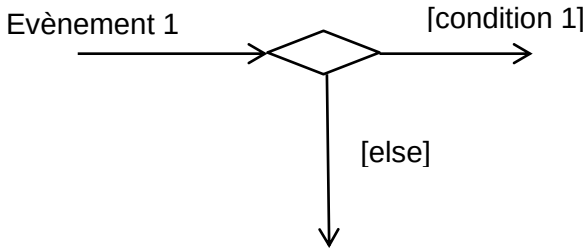
• Transition conditionnelle :
Plusieurs transitions peuvent quitter un même état. Une seule d'entre elles doit être déclenchée ; les événements et / ou les conditions de garde doivent donc être exclusives (ne jamais être vraies en même temps).



Une autre représentation est possible, qui utilise le point de décision :



Attention : les [condition 1] et [condition 2] doivent être exclusives (ne jamais être vraies en même temps).

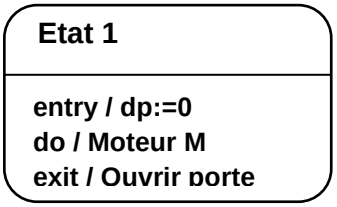


Il est possible d'utiliser une condition particulière, notée [else], sur un des segments en aval d'un point de décision. Ce segment n'est franchissable que si les conditions des autres segments sont toutes fausses.

c. Action

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'1 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 ;



exemple

• 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.

Rq1 : On peut aussi ne pas utiliser de mot réservé, auquel cas cela correspond à un do/.

Rq2 : Un état peut ne pas contenir d'action. Il sert alors à attendre le déclenchement de la transition suivante.

exemple

• 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,

De même *(compléter)* :

…… pour décrémenter la valeur du compteur C de 1

….. pour initialiser la valeur du compteur C.

d. Lien avec les entrées/sorties de la fonction "Traiter l'information" de la chaîne d'information

On aura compris que :

- Les informations en entrée vont intervenir dans les transitions du graphe d'états.
- Les ordres de commande en sortie vont intervenir dans les actions lancées dans les états actifs.

3. Structures

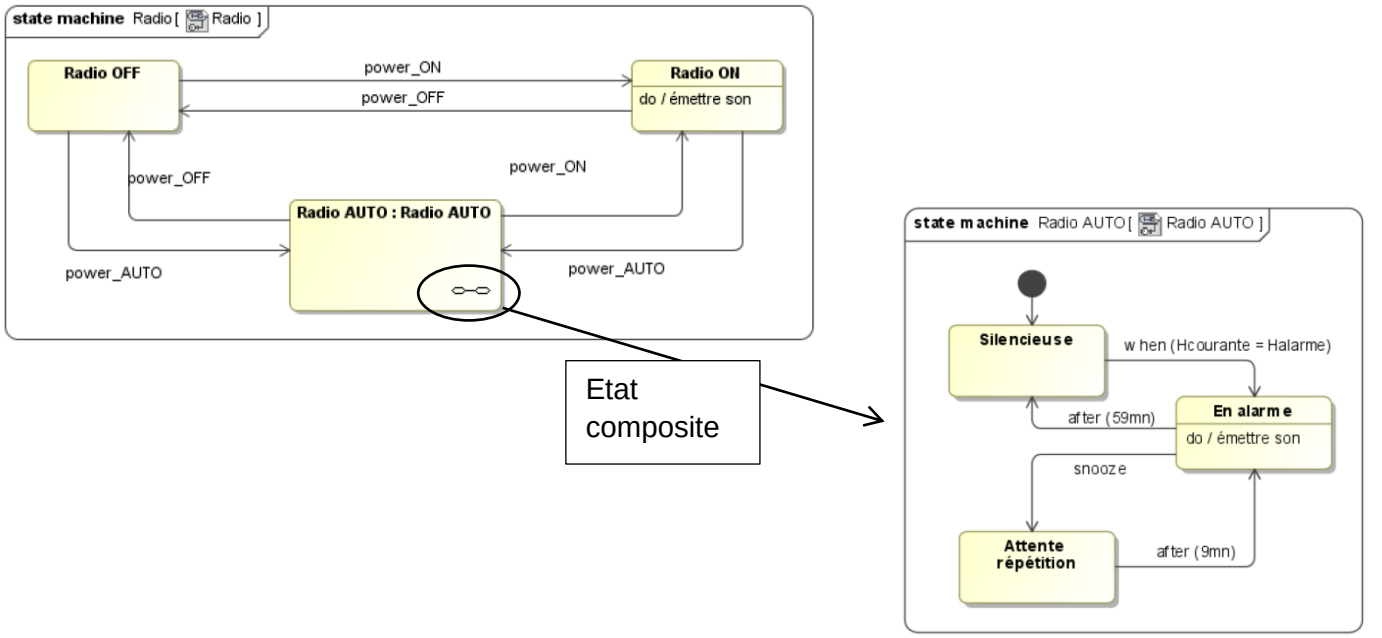
a. Etat composite

Pour simplifier Quand un graphe d'états est trop complexe, il est possible de le simplifier en regroupant certains états dans un état dit composite.

Dans le graphe d'états principal, l'état composite peut être détaillé (voir l'exemple ci-après) ou non. Dans ce cas il est simplement représenté ainsi :

Exemple : Radio-réveil :

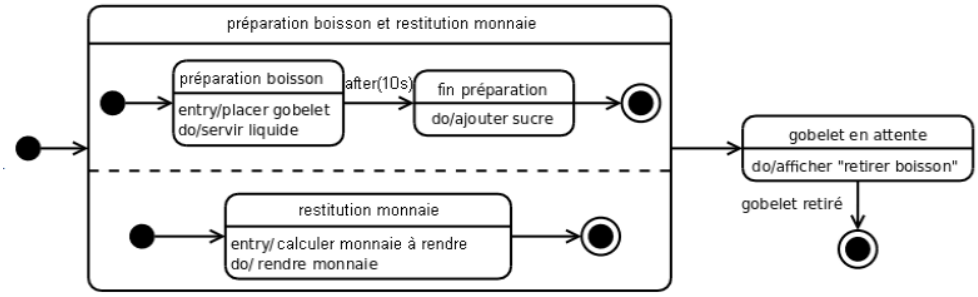
Diagramme d'états global :



b. <u>Régions concurrentes</u>

Dans les états composites, on autorise plusieurs états actifs dans des régions différentes, séparées par un trait pointillé. Chaque région évolue de manière indépendante. Attention : un seul état ne peut être actif dans chaque région. On dit que les régions sont orthogonales ou concourantes.

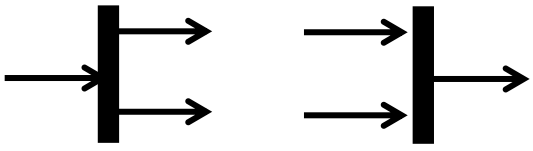
Exemple : Distributeur de boissons chaudes



Dans cet exemple, la préparation de la boisson se fait « parallèlement » à la restitution de la monnaie. Le diagramme d'états décrit donc un état composite à deux régions **concurrentes**.

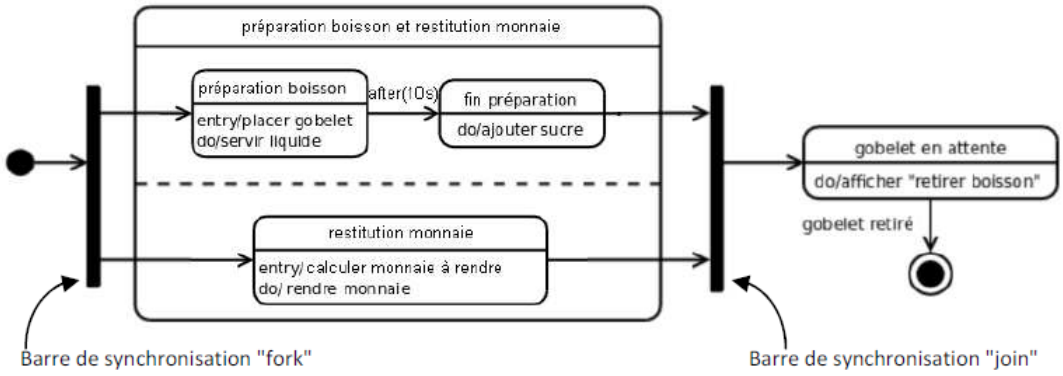
Pour qu'un état orthogonal soit terminé (dans son état final) il faut que chacune de ses régions soient dans leur état final.

Pour les régions concurrentes, il est possible d'utiliser des transitions de synchronisation de type « fork » (divergence) et « join » (convergence).



« fork » « join »

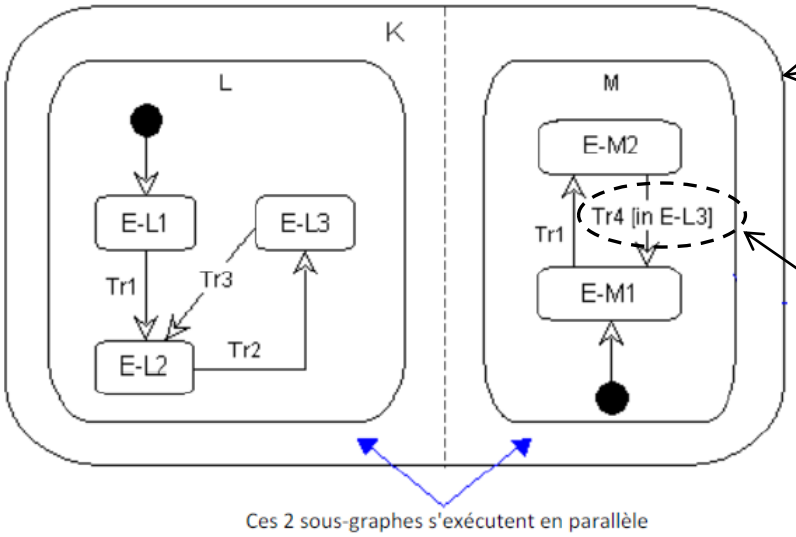
Ainsi le diagramme de l'exemple précédent est équivalent à celui-ci :



c. <u>Synchronisation de plusieurs sous-graphes</u>

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

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



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

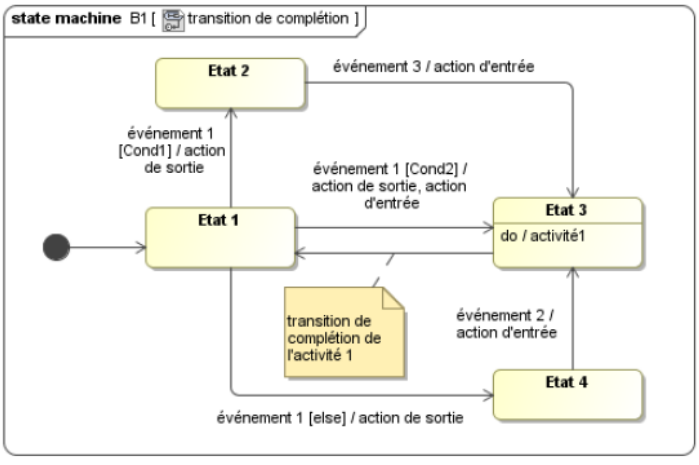
L et M s'activent simultanément, et évoluent indépendamment, en parallèle (régions concurrentes). Au départ, EL1et E-M1 sont actifs.
Pour que M passe d'E-M2 à E-M1, il faut que l'évènement Tr4 survienne, à condition que E-L3 soit actif.

Cet évènement est conditionné par l'activité de l'état E-L3 de l'autre sous-graphe.

d. <u>Transition de complétion</u>

La transition de complétion d'une activité finie, aussi appelée transition automatique, est représentée en SysML sans nom d'événement ni mot-clé.

La transition est franchie lorsque l'activité est terminée.



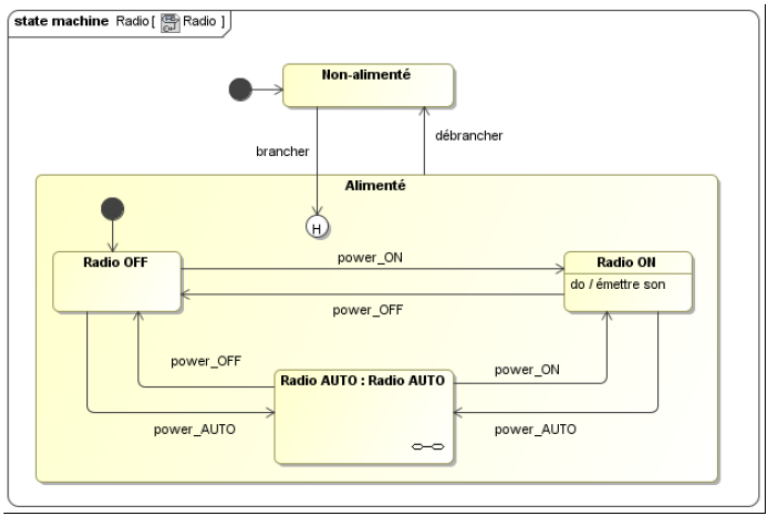
On n'utilise cette syntaxe que quand il n'y a pas d'ambiguïté possible

e. Pseudo-état « history »

L'activation du pseudo-état History permet à un super-état de se souvenir du dernier sous-état séquentiel qui était actif avant une transition sortante. Une transition vers l'état History rend à nouveau actif le dernier sous-état actif, au lieu de le ramener vers le sous-état initial

H

Exemple : Radio-réveil :
Dans cet exemple, en cas de débranchement intempestif, le système se souviendra dans quel état il se trouvait lorsque l'on rebranche le radio-réveil.



On peut aussi utiliser le pseudo-état « deep history » de sous-états de l'état composite sont mémorisés.

H'

l'activité de tous les niveaux hiérarchiques