62 Systèmes à événements discrets

Sommaire

1) Les systèmes séquentiels :

1.1) Définitions :

1.1.1) Systèmes à évènements discrets:

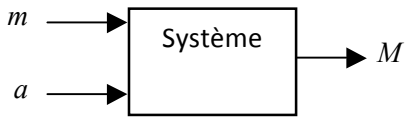
Les systèmes à évènements discrets sont des systèmes logiques dont les évolutions ont lieu à des instants précis en réponse à des évènements ponctuels. Contrairement aux systèmes combinatoires, le temps intervient dans leur évolution. C'est pourquoi ils sont également appelés systèmes séquentiels. Ces systèmes sont caractérisés par des états qui changent selon les évènements, conditionnant les activités et actions réalisées.

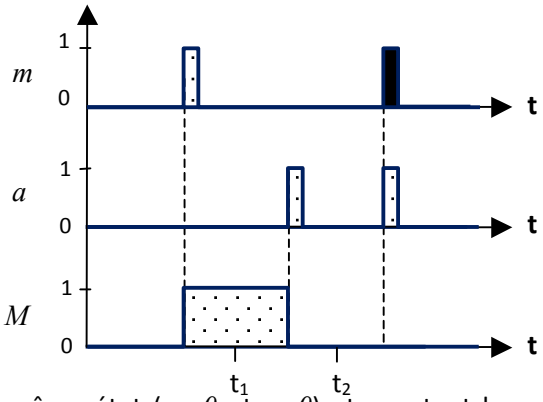
1.1.2) Combinatoire/Séquentiel :

le cours précédent indique qu'un **système combinatoire** est indépendant du temps
"la même cause produit toujours le même effet et l'effet disparaît dès que la cause disparaît."

Un système séquentiel (ou à évènements discrets) est un système logique qui n'est pas combinatoire. L'état des sorties dépend de celui des entrées mais aussi de l'état du système lui-même.
Prenons l'exemple d'un moteur électrique M piloté par deux boutons poussoir m (marche) et a (arrêt). Le cahier des charges impose le fonctionnement suivant :

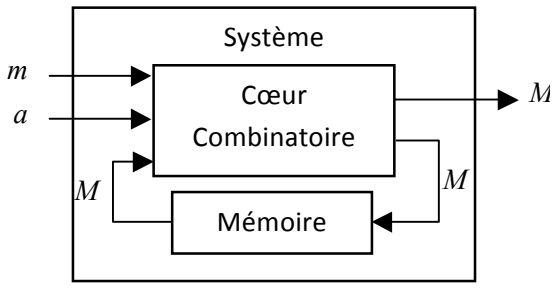
- Le moteur doit démarrer si le bouton marche m est actionné.
- Le moteur doit continuer à tourner après avoir relâché le bouton marche m et ce jusqu'à appui sur le bouton arrêt a.
- En cas d'appui simultané sur les deux boutons, la priorité est donnée par sécurité à l'arrêt du moteur.





Aux instants t_1 et t_2, les entrées sont dans le même état ($m=0$ et $a=0$) et pourtant la sortie n'a pas le même état.
"Pour un système séquentiel : la même cause peut produire des effets différents et un effet peut rester maintenu même si sa cause a disparu."

Il n'est ainsi pas possible de dresser la table de vérité du système de commande du moteur car au mot d'entrée *00* peut correspondre deux états pour la sortie. Cela peut quand même être fait en ajoutant une nouvelle variable binaire qui permet de distinguer ces deux états : cette nouvelle

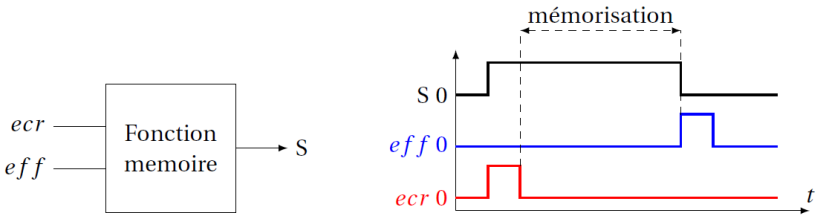


variable (appelée *variable d'état* ou *variable interne,* parfois notée x) joue le rôle de mémoire. La structure générale d'un système séquentiel peut ainsi être décomposée en un cœur combinatoire plus une mémoire qui décrit l'état du système.

1.2) Fonction mémoire :

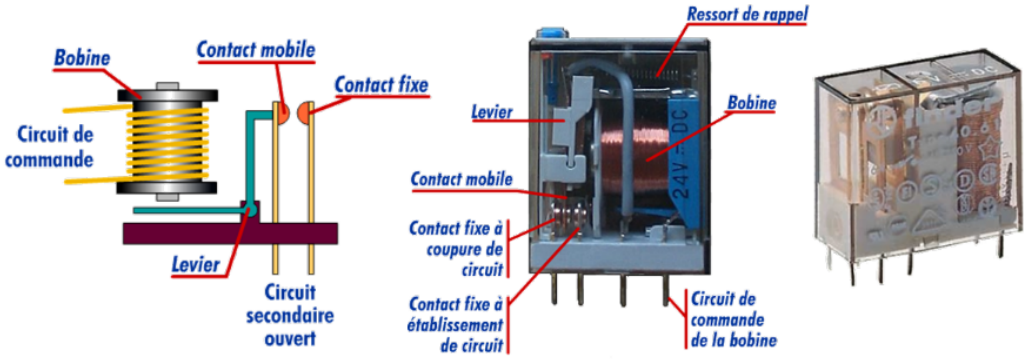
1.2.1) Définition :

À l'apparition du signal *ecr* (écriture), la sortie passe à l'état *1*, à la disparition du signal la sortie reste dans le même état . L'apparition du signal *eff* (effacement), la sortie repasse à l'état *0*.
Le maintien de la sortie correspond à l'effet mémoire.

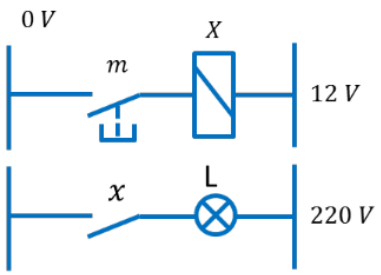


On peut noter l'existence des mémoires permanentes (qui conservent l'état des variables en cas de coupure d'énergie) et les mémoires volatiles (qui perdent l'état des variables en cas de coupure d'énergie).
Les mémoires peuvent être réalisées de façon optique (blu-ray), électropneumatiques (distributeurs électropneumatiques bistables), magnétiques (disques durs), électriques (grâce à des transistors, mémoire flash), électromagnétique (relais).
Dans les systèmes automatisés industriels, le relais auto-maintenu électromagnétique est un élément clef de la mémorisation des états.

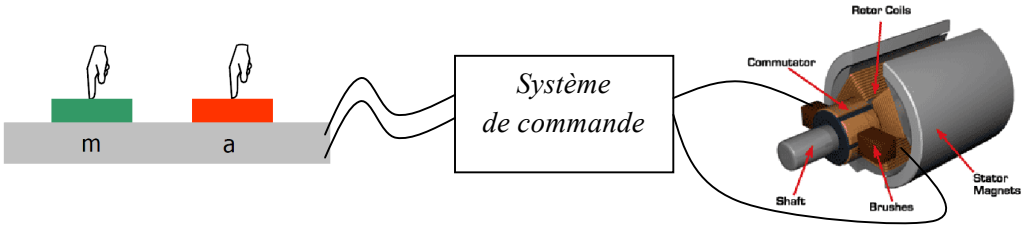
1.2.2) Réalisation d'une mémoire :



Lorsque l'utilisateur presse le bouton m, le relais X est activé. Ainsi, la variable x est mise à 1 ce qui provoque l'éclairage de la lampe L. À ce stade, si on relâche l'interrupteur m le relais X est désactivé, la variable x passe à l'état 0 et la lampe s'éteint.



a) mémoire à effacement prioritaire :



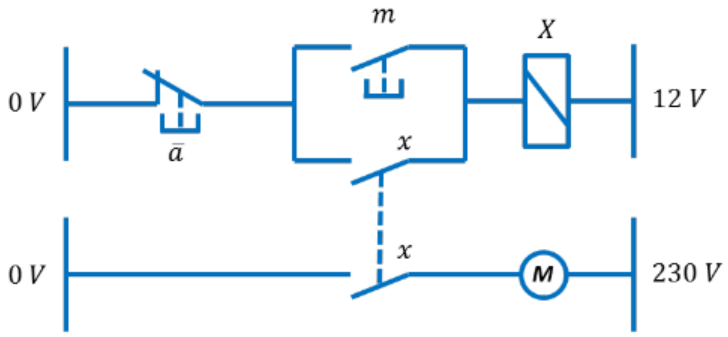
Pupitre *Moteur*

Dans ce cas, le moteur s'allume lorsque m est pressé. Il est éteint lorsque a est pressé ou lorsque m et a sont pressés simultanément. Il est possible de traduire le fonctionnement du moteur par l'équation booléenne suivante :

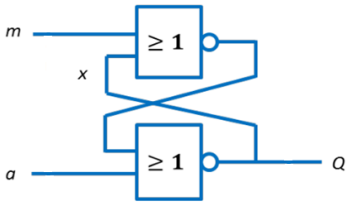
$$M = \overline{a} \cdot (m + x)$$

Ainsi dans un premier temps, en appuyant sur m le relais est actionné, x passe donc à 1 dans le circuit de puissance (allumant ainsi le moteur) **et** dans le circuit de commande.

On effectue ainsi un **auto maintien**. On comprend aisément qu'un appui simultané sur a et m provoque une désactivation du relais et donc un arrêt du moteur.



Lorsqu'on utilise une technologie électronique, il est courant d'utiliser de réaliser un câblage en utilisant des cellules NOR.



Il n'est pas directement possible d'écrire directement la table de vérité de la sortie Q en fonction de m et de a . Il faut avoir recours à une variable interne x . On retrouve bien l'équation précédente :

$$Q = \overline{m} \cdot \overline{a} \cdot x + \underbrace{m \cdot \overline{a} \cdot \overline{x} + m \cdot \overline{a} \cdot x}_{m \cdot \overline{a}} = \overline{a} \cdot (\overline{m} \cdot x + m) = \overline{a} \cdot (x + m)$$

m	a	x	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

2) Machine à états - *State machine diagram*- stm:

Il n'est pas facilement réalisable de décrire le comportement d'un système séquentiel, à partir d'une simple table de vérité. Le nombre de combinaison possible à étudier devenant très rapidement un obstacle à toute synthèse. (2^n combinaisons pour n variables).

Plusieurs outils, en général graphiques, ont été développés pour faciliter cette description. Nous allons utiliser deux diagrammes du langage *SysML* pour décrire les systèmes à événement discret. Le diagramme des machines à états du langage *SysML* permet de décrire l'évolution des états d'un système en fonction de événements extérieurs.

2.1) Présentation :

Le diagramme d'états fait partie du langage *SysML*. Il fait partie des diagrammes comportementaux comme le diagramme d'activité et le diagramme de séquence.

Il permet de décrire le comportement de l'ensemble du système (ou d'une de ses parties) qui peut se décomposer en :

- un **état initial**
- un ensemble d'**évènements**
- un ensemble d'**états**
- un ensemble d'**activités**
- un ensemble de **règles** permettant de déterminer le comportement du système.

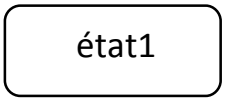
Le diagramme d'états est essentiellement un outil graphique permettant de modéliser le comportement séquentiel d'un système. Il peut aussi servir à programmer les composants réalisant la fonction *Traiter* de la chaîne d'information.

2.2) Etats d'un système :

L'état d'un système se caractérise par une phase de son fonctionnement stable pendant laquelle il est caractérisé par la valeur d'un certain nombre de variables. C'est une phase du cycle de vie du système pendant laquelle il peut réaliser des **activités** ou être en **attente**.

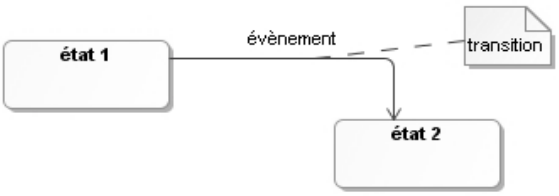
On distingue 3 états principaux :

- l'état initial représenté par un cercle plein noir . ●
  Cet état (pseudo état) est nécessaire, il indique le point d'entrée dans un graphe.
- l'état final représenté par un cercle plein noir cerclé . ◉
  Ce pseudo état n'est pas toujours nécessaire, il décrit l'état final d'un comportement.
- l'état proprement dit. Il représente un phase de vie du système, le système reste dans l'état décrit tant que les conditions d'évolution ne sont pas remplies. Il est représenté par un cadre précisant l'état et/ou les actions à réaliser pendant cette phase de vie. Un état est représenté par un cadre.



2.3) Syntaxe et règle de base :

Les éléments graphiques de base d'un diagramme d'états sont des cadres (**nœud**) qui représentent les états possibles du système et des flèches (**transition**).



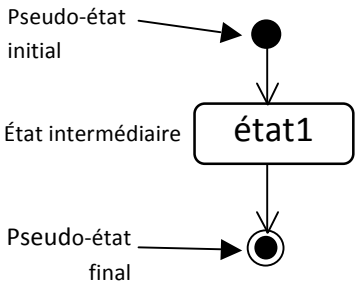
Le passage de l'état 1 (état source) à l'état 2 (état cible ou état pointé) ne peut se faire que si l'état 1 est actif et que l'évènement associé à la transition se produit.

La syntaxe des diagrammes d'état est complétée par d'autres symboles graphiques et par d'autres règles d'évolution dont certains sont présentés dans les paragraphes suivants. Cette liste non exhaustive permet de commencer à écrire et à lire un bon nombre de diagrammes d'états.

2.4) Types d'états :

On distingue :

- le pseudo-état initial qui est actif à l'initialisation du diagramme
- l'état intermédiaire qui décrit une phase du cycle de vie ;
- le pseudo-état final qui décrit la fin éventuelle du diagramme.



Les états intermédiaires peuvent correspondre à la réalisation d'activités et/ou d'actions. Ce n'est pas le cas des pseudo-états initial et final d'où leur qualification de pseudo-états. Un diagramme possède nécessairement un état initial et un seul. En revanche, il peut ne pas posséder d'état final, comme en posséder plusieurs.

À un instant donné : **un état et un seul est nécessairement actif**.

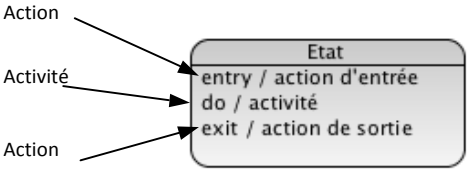
2.5) Activités et actions associées aux états :

2.5.1) Activité :

Une **activité** est un effet qui commence dès l'activation de l'état et possède une certaine durée. Elle s'indique par : *do/activité*.

Elle peut être :

- **finie** si elle s'arrête d'elle même au bout d'un certain temps ou lorsque l'état cesse d'être actif ;
- **continue** si elle ne cesse que lorsque l'état cesse d'être actif.



2.5.2) Action :

Contrairement à une activité, une **action** est un effet considéré comme instantané donc non interruptible. Il s'agit généralement de l'affectation d'une valeur à une variable. Les actions peuvent avoir lieu soit à l'activation de l'état soit à sa désactivation et s'indiquent par :

- *entry/action* : l'action est réalisée à l'activation de l'état,
- *exit/action* : l'action est réalisée à la désactivation de l'état.

Des conditions (indiquées entre crochets) peuvent être ajoutées aux actions. Par exemple :

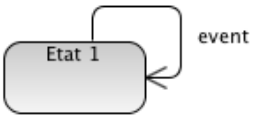
entry[test]/action réalisera l'action à l'activation de l'état uniquement si *[test]* est vrai. De même pour *exit[test]/action*. Par contre, une activité n'est jamais conditionnée. Un état peut ne contenir ni activité ni action. Il s'agit alors d'un état d'attente d'un événement activant un autre état. Mais on peut aussi voir l'attente comme une activité particulière...

2.6) Transitions :

Une transition permet d'indiquer la possibilité d'évolution d'un état source vers un état pointé.
Lorsque l'événement *event* survient, instantanément :

- Etat 1 est désactivé et :
  - si une activité était en cours, elle cesse ;
  - s'il y a une action à la désactivation, elle est réalisée.
- Etat 2 est activé et :
  - s'il y a une action à l'activation, elle est réalisée ;
  - s'il y a une activité prévue, elle débute.

L'état source et l'état pointé peuvent être le même auquel cas on parle de transition réflexive.
Lorsque l'événement *event* survient, instantanément *Etat1* est désactivé puis réactivé et donc :



- si une activité était en cours, elle cesse et redémarre ;
- s'il y a une action à la désactivation, elle est réalisée ;
- s'il y a une action à l'activation, elle est réalisée.

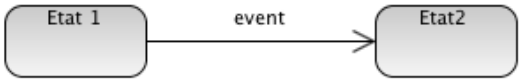
Tout ceci se faisant en un temps quasi nul.

La syntaxe générale d'une transition est *event[test]* où :

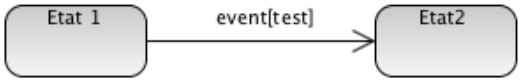
- *event* est **l'événement déclencheur**. C'est l'occurrence de cet événement (notion de front montant) qui peut produire l'évolution. On peut distinguer comme principaux évènements types :
  - des signaux divers provenant de capteurs, d'une IHM ou d'autres systèmes ;
  - des changements de valeurs de grandeurs internes comme un compteur par exemple, de syntaxe *when(valeur=...).*
  - des évènements temporels :
    - absolus, de syntaxe *at(date)* ou *when(date)* ;
    - relatifs, de syntaxe *after(durée),* la durée étant décomptée à partir de l'activation de l'état.
- *[test]* est appelée **condition de garde**. La variable booléenne « *test* » est évaluée à l'occurrence de l'événement « *event* » et si elle est vraie le franchissement a lieu.

Il peut y avoir des transitions avec seulement un événement déclencheur sans condition de garde, des transitions avec seulement une condition de garde sans événement déclencheur et aussi des transitions sans aucune condition, dites transitions automatiques ou de complétion :

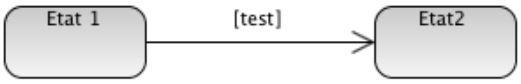
- À l'occurrence de *event*, la transition est franchie, sans condition. L'éventuelle activité de l'état 1 est interrompue.



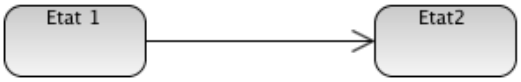
- À l'occurrence de *event*, *test* est évalué et la transition est franchie uniquement si *test* est vrai. L'éventuelle activité de l'état 1 est interrompue. Si *test* n'est pas vrai, *event* est « *perdu* » et il faut attendre une seconde occurrence de *event* pour éventuellement franchir la transition si cette fois *test* est vrai.



- Si *test* est vrai, la transition est immédiatement franchie dès la fin de l'éventuelle activité (qui doit donc être une activité finie). S'il n'y a pas d'activité associée à l'état 1, la transition est franchie immédiatement si *test* est vrai.



- Transition de complétion : est immédiatement franchie dès la fin de l'éventuelle activité de l'état 1.



Enfin, une action peut être réalisée lors du franchissement de la transition, elle s'indique par : *event[test]/action*.

2.7) Synthèse :