Systèmes à évènements discrets

Cours

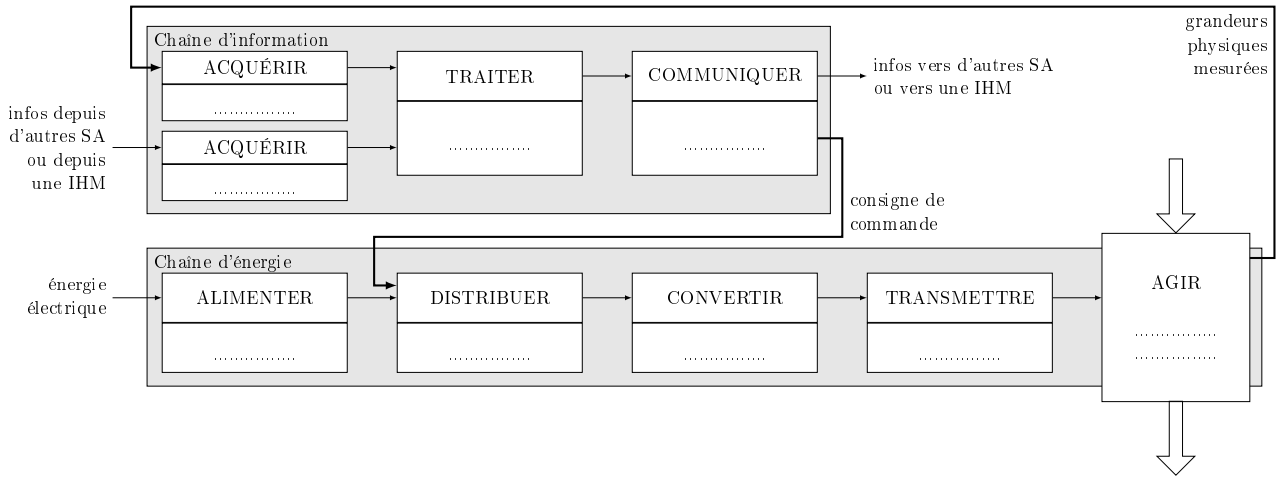
Professeur: YASSINE FARTOUH v0.3

CPGE - MARRAKECH

1 Préambule

Outre le dimensionnement du système en termes de performances (conception de la chaîne d'énergie), l'ingénieur doit de plus en plus concevoir la chaîne d'information qui aujourd'hui prend une part très importante dans les systèmes rencontrés.

Les capteurs acquièrent des grandeurs physiques qui sont ensuite traitées par la partie commande qui va ensuite communiquer des ordres à la chaîne d'énergie ou envoyer des informations à des éléments extérieurs.



Objectif

Analyser voire de décrire le fonctionnement d'un système.

2 Définitions

2.1 Variables binaires

De nombreux composants utilisés en automatisme ne peuvent normalement prendre que deux états différents : lampe allumée ou éteinte, bouton-poussoir actionné ou relâché, moteur tournant ou à l'arrêt, vérin pneumatique sorti ou rentré...

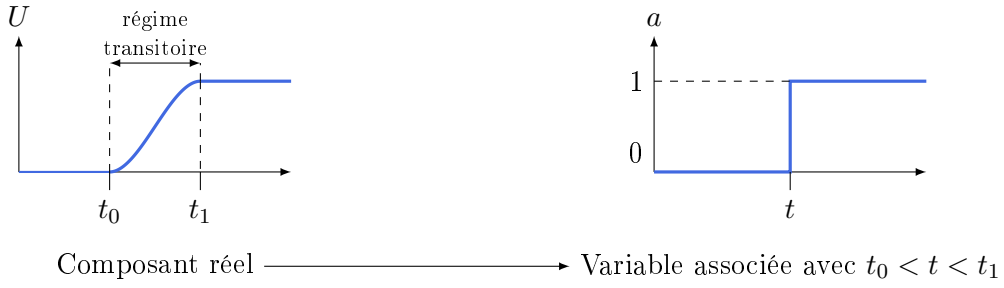
A chacun de ces composants, on peut associer une **variable binaire** (ou logique, ou Tout Ou Rien) qui ne peut prendre que deux valeurs notées 0 ou 1 (vrai ou faux, oui ou non).

Définition *Variable binaire*

Une variable a est binaire si et seulement si elle ne peut prendre, à chaque instant, qu'une seule valeur parmi un ensemble de 2 valeurs possibles.

Remarque

- le comportement « Tout ou Rien » (TOR) ne correspond qu'au comportement normalement prévu en régime stabilisé et en l'absence de tout dysfonctionnement.
- l'association d'une variable binaire à un composant ne peut pas rendre compte des états transitoires apparaissant entre deux états stables. C'est donc une simplification du comportement réel.



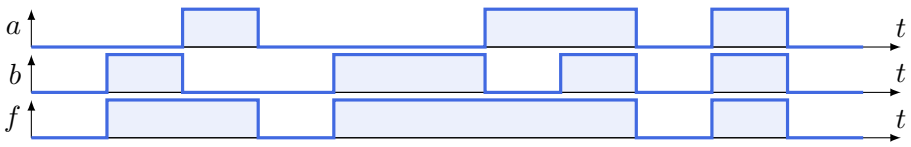
2.2 Chronogramme

Définition *Chronogramme*

Un chronogramme est une représentation de l'évolution temporelle d'une variable ou une fonction logique en fonction du temps (aussi appelés diagrammes de Gantt).

Exemple:

Soit la fonction logique $f(a,b) = a + b$. La figure suivante donne l'évolution de f en fonction du temps.



Remarque

On s'aide d'un chronogramme pour savoir si un système est combinatoire ou séquentiel. Il suffit de regarder si pour un même état des variables d'entrée, les sorties sont toujours les mêmes (cf. exemple de l'interrupteur).

Notion de fronts montant et descendant Les fronts permettent de représenter le changement d'état d'une variable. Cette notion est très importante dans le cadre de la synchronisation des processus mais aussi la prise en compte de l'évolution du système à partir de capteurs (exemple appui sur un bouton poussoir, codeur incrémental...).

Définition

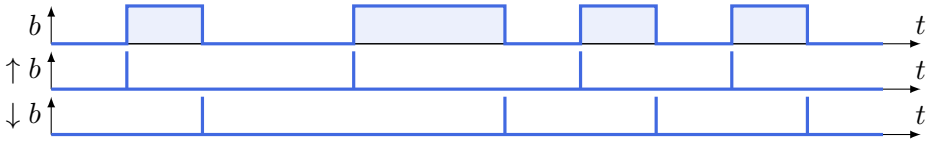
Le front montant d'une variable est vrai à l'instant pour lequel la variable passe de son état logique 0 à son état logique 1. Il est noté $\uparrow a$.

Le front descendant d'une variable est vrai à l'instant pour lequel la variable passe de son état logique 1 à son état logique 0. Il est noté $\downarrow a$.

Un front est d'une durée théorique nulle. Il sera représenté par une impulsion de Dirac sur un chronogramme.

Exemple:

Soit la variable logique b. La figure suivante donne l'évolution de b en fonction du temps et l'évolution des fronts montant et descendant de b.



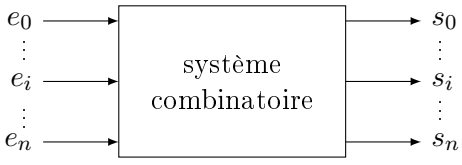
2.3 Systèmes combinatoire

Définition *Système combinatoire (FIGURE 1a)*

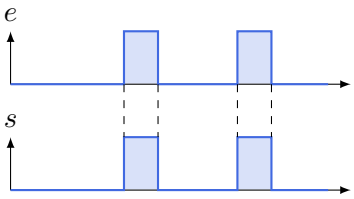
Un système logique combinatoire est un système binaire pour lequel à un état des variables d'entrée e_i correspond un unique état des variables de sortie s_i. (La réciproque n'est pas vraie)

La FIGURE 1b illustre l'appuie sur un interrupteur à deux positions :

- sur la position 0 (entrée à 0), la lumière est éteinte (sortie à 0),
- sur la position 1 (entrée à 1), la lumière est allumée (sortie à 1).



(a) Système combinatoire



(b) Chronogramme d'un système logique combinatoire

FIGURE 1 – Système logique combinatoire

3 Logique séquentielle

3.1 Introduction

Exemple : La FIGURE 2b illustre l'appuie sur un interrupteur de type bouton poussoir :

- sur la position 0 (entrée à 0), la lumière est éteinte (sortie à 0),

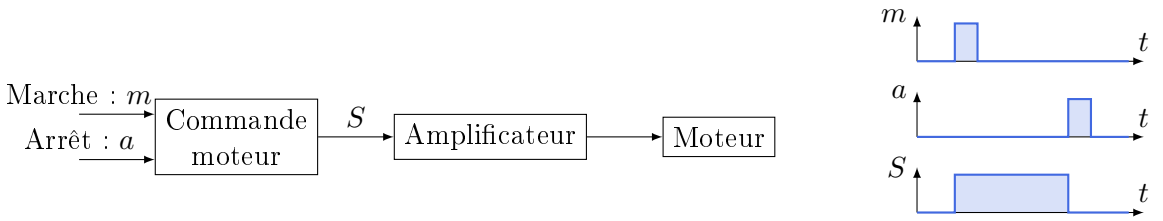
- sur la position 1 (entrée à 1, appui sur le bouton), la lumière s'allume (sortie à 1),
- sur la position 0 (entrée à 0, relâchement du bouton), la lumière reste allumée (sortie à 0),
- sur la position 1 (entrée à 1, appui sur le bouton), la lumière s'éteint (sortie à 0).

(a) Système logique combinatoire (b) Système logique séquentielle

FIGURE 2 – Différence entre système logique combinatoire et séquentiel

Exemple : Commande d'un moteur électrique par un système séquentiel

On remarque sur le chronogramme de droite que la sortie S peut présenter une valeur différente (0 ou 1) pour une configuration identique des entrées m et a.

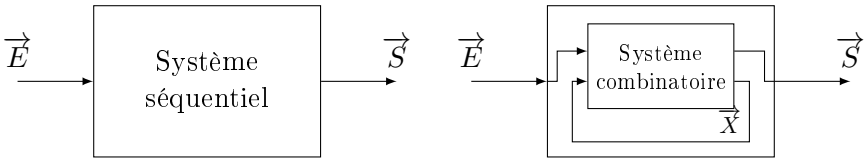


Définition *Système séquentiel*

Un système logique est dit séquentiel si les sorties S_j ne dépendent pas uniquement des entrées $E_i(t)$ mais aussi de l'évolution antérieure du système (historique).

En effet, à un état des $E_i(t)$ peuvent alors correspondre plusieurs états des S_j. Il y a alors nécessité de faire apparaître de nouvelles variables internes (appelées mémoires).

Le système est capable de mémoriser de l'information. Cette information mémorisée est l'état du système, qui peut être représenté par un vecteur d'état booléen : $\overrightarrow{X} = (x_1, x_2, \ldots, x_n)$.



Connaissant $\overrightarrow{X}$ et $\overrightarrow{E}$ (le vecteur d'état booléen des entrées), la sortie $\overrightarrow{S}$ peut être déterminée comme une fonction booléenne de $\overrightarrow{X}$ et de $\overrightarrow{E}$: $\overrightarrow{S} = f(\overrightarrow{X}, \overrightarrow{E})$ (donc via un système combinatoire).

L'état interne $\overrightarrow{X}$ à l'instant t dépend de $\overrightarrow{E}(t)$ et de l'état interne immédiatement précédent : $\overrightarrow{X}(t) = g(\overrightarrow{E}(t), \overrightarrow{X}(t - \Delta t))$.

Les tables de vérité ou logigrammes ne permettent plus de décrire l'évolution de S en fonction de E. On peut utiliser alors les chronogrammes (ou diagrammes de Gantt).

Pour décrire le fonctionnement global du système, de la partie logicielle, on utilise plutôt des outils de description du SysML : les diagrammes de séquence, d'états (ou d'activité : hors programme).

3.2 Diagramme de séquence – SysML

Ce diagramme permet de définir la chronologie des interactions entre les acteurs (utilisateurs) et le système. Il permet de décrire un scénario correspondant à un cas d'utilisation.

Le principe est de décrire les messages envoyés entre les acteurs dans l'ordre chronologique (le temps s'écoule vers le bas). Le comportement interne des constituants n'est pas détaillé dans ce diagramme.

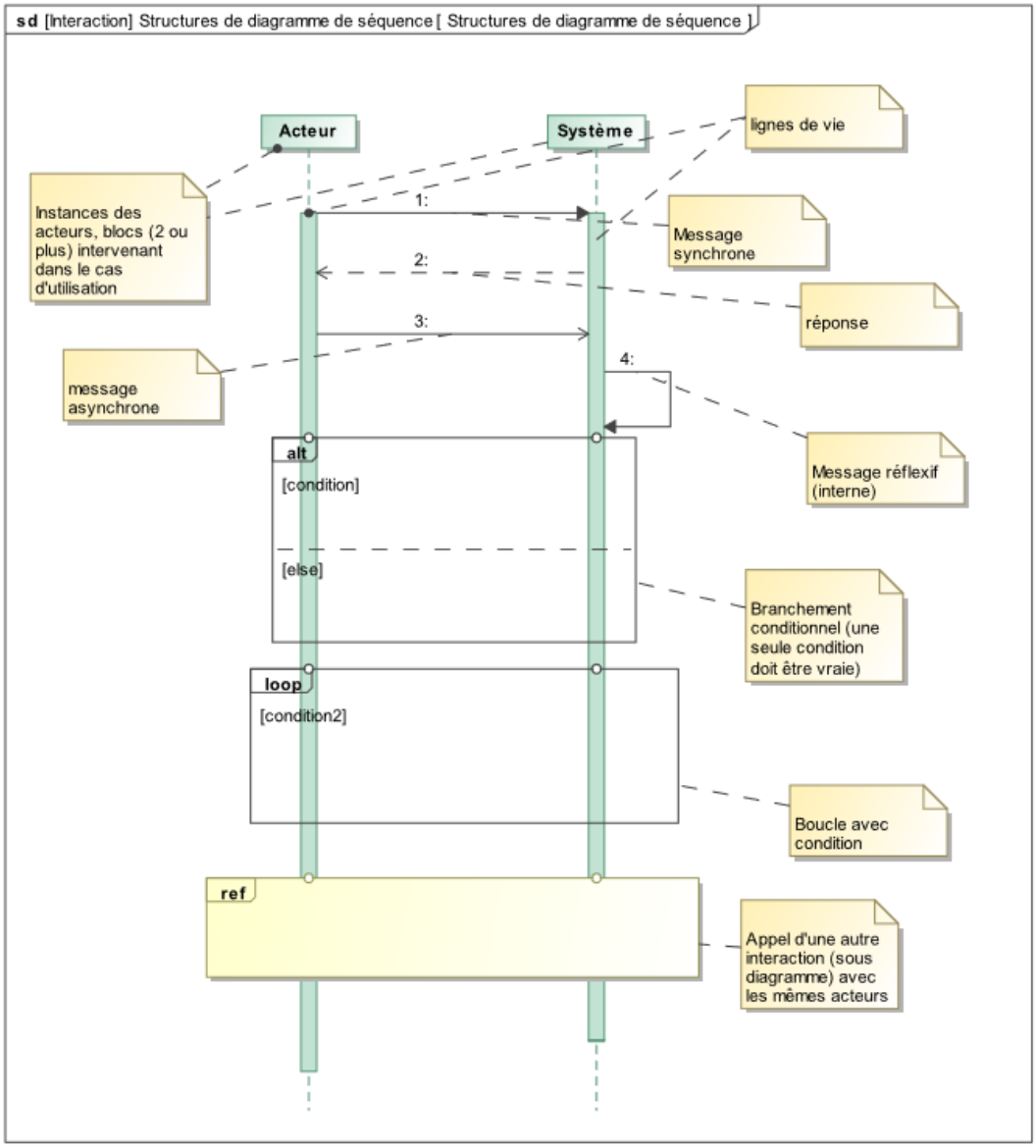


FIGURE 3 – Structure d'un diagramme de séquence.

Les éléments constitutifs du diagramme sont (FIGURE 3) :

- les lignes de vie qui représentent les acteurs (du diagramme de cas d'utilisation) participant à la communication. Elles peuvent être actives ou inactives (bandes verticales).
- les messages : éléments de communication unidirectionnels entre les lignes. Il existe plusieurs types de messages :
  - ◇ Message synchrone : l'émetteur est bloqué en attente de réponse
  - ◇ Message asynchrone : pas de réponse attendue
  - ◇ Message réflexif : comportement interne
- les opérateurs qui permettent de réaliser des tests, des opérations en parallèle, des boucles...
  - ◇ **loop** (boucle) : plusieurs exécutions successive d'un même fragment,
  - ◇ **opt** (optionnel) : exécution si la condition fournie est vraie
  - ◇ **alt** (fragments alternatifs) : éxecution du fragment si une condition parmi plusieurs est vraie
  - ◇ **ref** : appel d'un autre diagramme de séquence défini ailleurs
  - ◇ **par** : exécution en parallèle

3.3 Diagramme d'états – SysML

Ce diagramme permet de décrire les différents états que peut traverser le système en fonction des évènements perçus.

Il est utilisé essentiellement pour décrire les modes de marche, d'arrêt, d'attente d'un système ou sous-système.

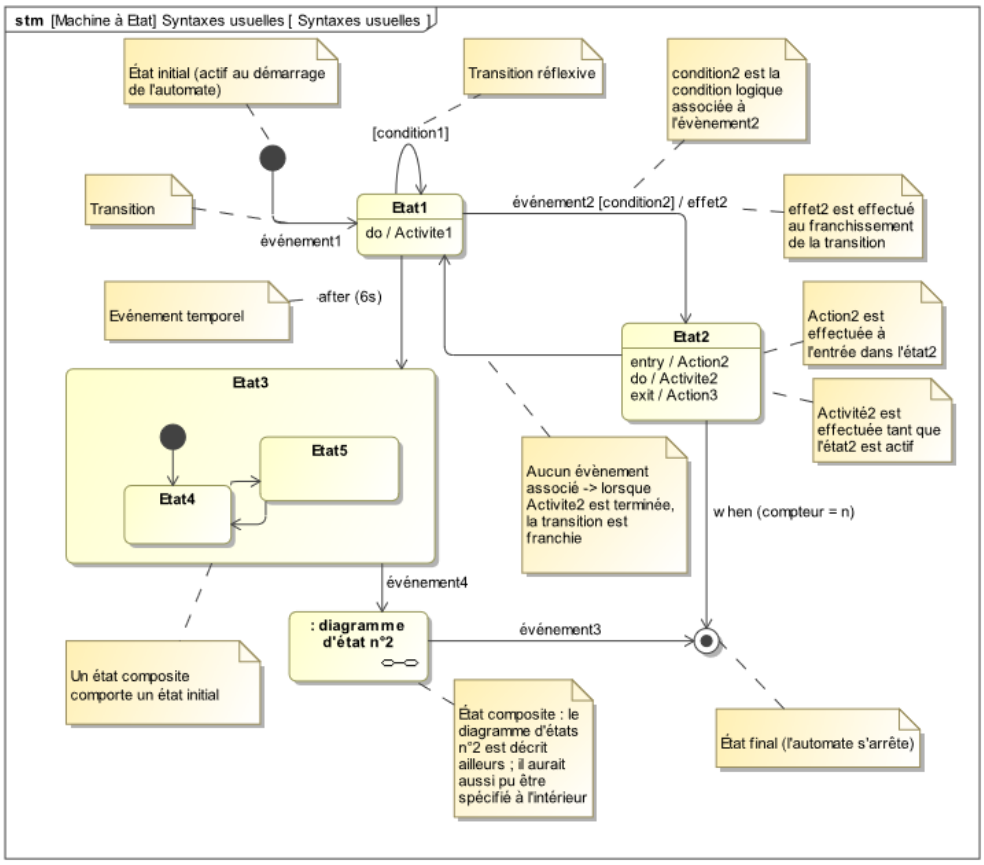
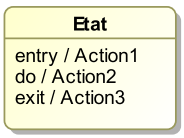


FIGURE 4 – Structures usuelles d'un diagramme d'état

Le diagramme d'états utilise plusieurs symboles particuliers :

- un état représenté par un bloc qui représente une situation d'une durée finie. Pendant cet état, il peut se produire des actions. L'état initial dans lequel est placé le diagramme au départ est représenté par un symbole particulier ●. On peut également utiliser un symbole pour représenter le ou les états finaux (le diagramme ne peut plus évoluer) ◉.
- les événements représentés par des flèches permettent de décrire comment passer d'un état à un autre. On peut associer une condition booléenne (appelée condition de garde) à l'événement ou utiliser des mots clés particuliers :
  - ◇ `after x s` : après x secondes on passe à l'état suivant,
  - ◇ `at h` : à l'heure indiquée, on passe à l'état suivant
  - ◇ `when x=xc` : on passe à l'état suivant quand une valeur atteint une valeur donnée
  - ◇ si aucun texte n'est noté sur la flèche (on parle de transition de complétion), ceci indique que l'on passe à l'état suivant dès que les actions associées à un état sont terminées.

Il peut être utile de spécifier à l'intérieur d'un état des actions qui sont effectuées. On utilise pour cela le mot clé `do` suivi d'une ou plusieurs actions. On peut spécifier plus précisément à quel moment se feront les actions à l'intérieur d'un état :



- `entry` suivi d'une action indique que l'action est faite en entrant dans l'état
- `exit` suivi d'une action précise que l'action est faite en sortie de l'état

Un état peut contenir lui même un diagramme d'état. On parle d'état composé ou composite. Comme tout diagramme d'état, l'état composite contiendra une étape initiale qui sera activée lorsqu'on entrera dans l'état composite. On le représente à l'intérieur d'un bloc ou bien on utilise un symbole pour faire référence au diagramme d'état.

3.4 Analyse d'un algorithme – Symbole

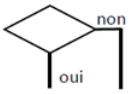
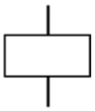
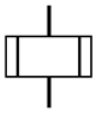
SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	**Début** ou **fin** d'un algorithme		**Test ou branchement conditionnel** Décision d'un choix parmi d'autres en fonction des conditions
	Symbole général de « **traitement** » Opération sur des données, instructions ou Opération pour laquelle il n'existe aucun symbole normalisé		**Sous-programme** Appel d'un sous-programme
	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.		**Commentaire**

FIGURE 5 – Symboles