

18.1 Numération, codage, transcodage 1

18.2 Systèmes combinatoires 3

18.3 Fonctions logiques . . . 7

18.4 Simplification des fonctions logiques 9

18.5 Représentation des fonctions logiques 10

18.1 Numération, codage, transcodage

Remarque

On considère acquis les notions de systèmes de numérations décimaux, binaires et hexadécimaux ainsi que le passage d’une base à l’autre.

On considère aussi qu’il existe des notions sur la norme ASCII.

18.1.1 Systèmes de codages utiles en technologie

Code binaire naturel

Ce code pondéré correspond à donner un nombre selon sa valeur en système de numérotation binaire.

C’est le seul code qui permet de réaliser des opérations.

Inconvénient : il introduit des erreurs lors d’un changement de code. Pour passer de $(01)_2$ à $(10)_2$, donc de 1 à 2, il faut modifier chaque digit et afficher (même brièvement) $(11)_2$ (soit 3) ou $(00)_2$ (soit 0). Cela peut provoquer des perturbations ou des erreurs.

Code décimal codé binaire (DCB)

C’est un code pondéré base sur le code binaire naturel mais qui est adapté à la représentation des nombres en base 10. En effet le code binaire pur n’associe pas des bits spécifiques aux unités, dizaines, centaines, ... La propriété du code DCB est d’associer 4 bits différents à chaque puissance de 10. Ainsi,

$$(1664)_{10} \implies (0001.0110.0110.0100)_{DCB}$$

Ce code est utilisé pour les afficheurs 7 segments. Chaque afficheur reçoit le chiffre codé en binaire sur 4 bits.

Inconvénient : Cette représentation adaptée à la représentation binaire des nombres décimaux utilise un nombre de bits supérieur à celui du binaire naturel, et donc une place plus important en mémoire de l’ordinateur.

Remarque

On peut réaliser la même typologie pour le code hexadécimal. Notons ce codage HCB. Ainsi,

$$(0C3F)_{16} \implies (0000.1100.0011.1111)_{HCB}$$

Entrée				Sortie
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

TABLE 18.1 – Table de vérité du code binaire naturel



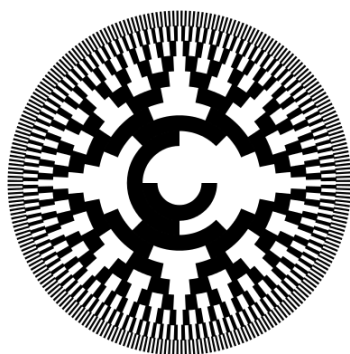


FIGURE 18.1 – Disque codage Gray

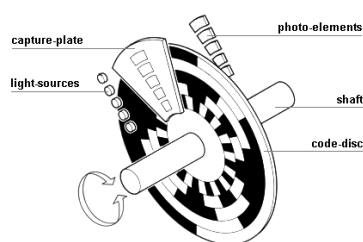


FIGURE 18.2 – Disque codage Gray

Code binaire réfléchi ou code Gray

Ce code non pondéré est un arrangement du système binaire. Le passage d'un nombre à l'autre se fait en changeant l'état d'un seul bit. Il est très utilisé pour décrire des automatismes (un changement d'état d'un composant correspond à un bit qui change), en particulier dans les codeurs de position absolue.

Avantage : il apporte une garantie d'interprétation avec une erreur maximale d'incrément.

Pour obtenir le code Gray, il faut faire toutes les $2^1, 2^2, 2^3 \dots$ lignes, une symétrie en commençant par le bit de droite et changer la valeur du bit de gauche.

Décimal	Binaire pur	Binaire réfléchi
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

18.1.2 Les codeurs optiques

Le codeur incrémental

Un codeur incrémental est constitué d'un disque généralement muni de deux pistes ainsi que de trois couples de DEL – récepteurs. Une des deux pistes est percée d'une seule fente. Le détecteur noté Z permet de détecter le passage de cette fente. La seconde piste est composée de n fentes. Les deux autres DEL (A et B) détectent le passage des fentes sur cette piste.

Dans certains cas, les impulsions peuvent être mesurées par des capteurs à effets Hall.

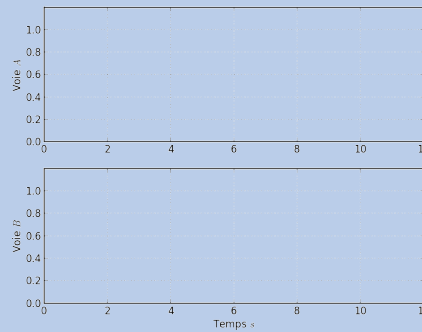
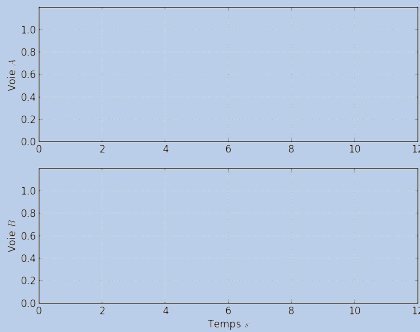
Ces codeurs permettent de réaliser une mesure incrémentale. Il faut donc fixer une référence (POM – Prise d'Origine Machine) pour connaître la position absolue.

Ces codeurs sont utilisés pour mesurer les déplacements des axes sur les machines outils. On en retrouve aussi sur l'axe asservi Emericc ou encore sur le Tribar.

Exemple –

La résolution d'un codeur incrémental peut varier de 1 000 à 25 000 impulsions par tour pour des fréquences de rotation allant jusqu'à 10 000 tr/min.

1. Donner la résolution en degrés du codeur.
2. Quelle doit être la fréquence minimale du système d'acquisition afin de pouvoir gérer les informations provenant du codeur ?
3. Réaliser le chronogramme des sorties des voies A et B en faisant l'hypothèse que les DEL A et B sont «décalées» d'un quart de période.
4. Expliquer en quoi l'existence des deux LED permet de détecter le sens de rotation du disque.

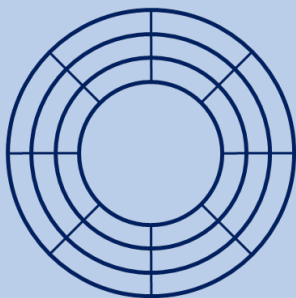


Le codeur absolu

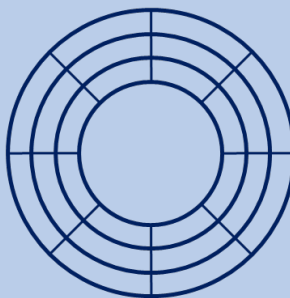
Un codeur absolu permet de déterminer la position angulaire réel d'un système. Il est constitué de N pistes avec des fentes qui sont agencées selon le codage Gray. Une «rampe» de N couples de DEL – récepteurs permettent de détecter, à un instant donné une combinaison de bits correspondant à la position du disque.

Exemple –

1. Combien de pistes faut-il pour coder 360 avec une précision de 0,1.
2. Combien faudrait-il de fentes avec codeur incrémental ?
3. Pour un codeur avec 3 pistes, teinter les différentes zones dans le cas d'un codage Gray puis dans le cas d'un codeur naturel.
4. Conclure sur l'intérêt du code Gray lorsque les DEL sont décalées.



Codage binaire naturel



Codage binaire réfléchi

18.2 Systèmes combinatoires

18.2.1 Définitions

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.

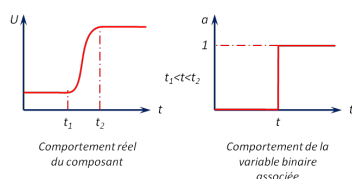
À chacun de ces composants, on peut associer une variable binaire (ou logique, Tout Ou Rien) qui ne peut prendre que deux valeurs notées 0 ou 1 (vrai ou faux, oui ou non).

Définition – Variable binaire

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

Exemple –

Lister les variables binaires pour un codeur incrémental avec une résolution de 0,1 degrés et pour un codeur absolu de même résolution.



Remarque

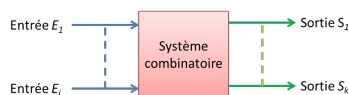
Le comportement tout ou rien (TOR) ne correspond qu'au comportement normalement prévu en régime stabilisé et en l'absence de tout dysfonctionnement. L'association d'une variable binaire à un composant ne peut pas rendre compte des états transitoires apparaissant entre deux états stables. C'est donc une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

Système binaire

Définition –

Un système est dit binaire ou logique si les variables d'entrée et de sortie sont binaires.

Systèmes combinatoire et séquentiel



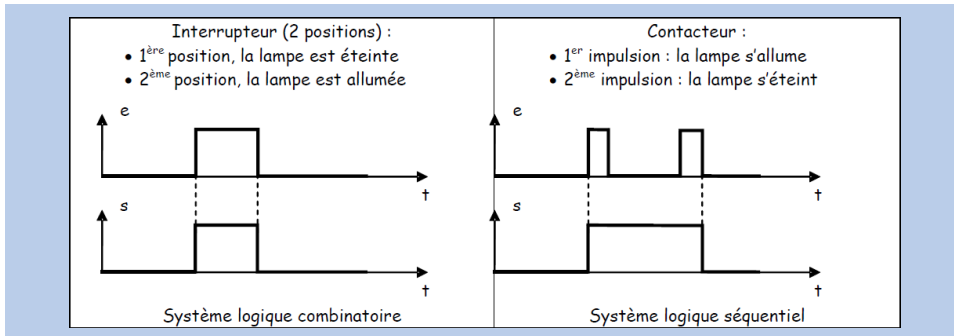
Définition –

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_j . (La réciproque n'est pas vraie.)

Définition –

Un système logique est dit séquentiel si les sorties S_j ne dépendent pas uniquement des E_i .

Exemple –



18.2.2 Algèbre de Boole

La fonction logique qui caractérise le système est indépendante du temps. Pour traiter de tels systèmes, on utilise l'algèbre de Boole.

Définition

C'est un algèbre de propositions logiques mise au point par un mathématicien anglais, Georges Boole (1815 – 1864).

Définition –

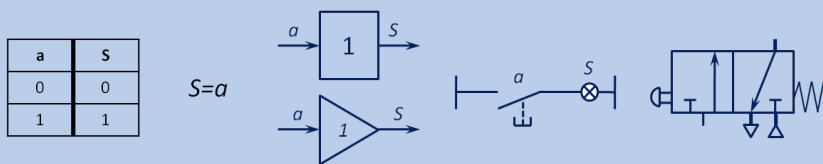
Un ensemble E a une structure d'algèbre Boole si on a défini dans cet ensemble :

- ▶ une relation d'équivalence notée $=$;
- ▶ deux lois de composition interne $+$ (addition booléenne) et \cdot (multiplication booléenne);
- ▶ une loi appelée complémentation (\bar{a} complément de a).

Une algèbre binaire est une algèbre de Boole dont les éléments B ne peuvent prendre que deux valeurs notées 0 ou 1 : $B = \{0, 1\}$.

Exemple –

Fonction OUI.



Fonction NON – appelée complément

Définition –

Fonction NON – appelée complément

$$B \mapsto \bar{B}$$

$$a \mapsto \bar{a}$$

a	\bar{a}
0	1
1	0

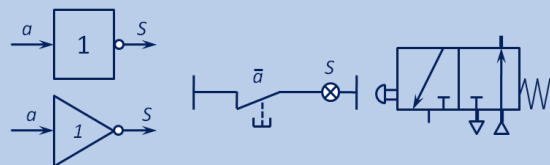
Il faut lire \bar{a} : NON a .

Exemple –

Fonction NON.

a	S
0	1
1	0

$$S = \bar{a}$$

**Fonction ET – Produit booléen****Définition –**

Fonction ET – Produit booléen

$$B \times B \mapsto B$$

$$(a, b) \mapsto a \cdot b$$

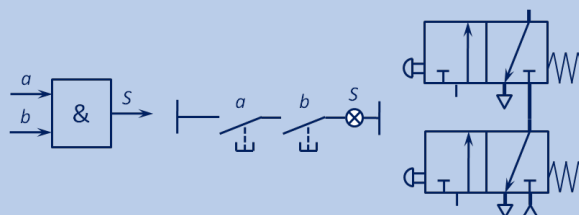
Il faut lire *a* ET *b*.

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Exemple –

a	b	S
0	0	0
0	1	0
1	1	1
1	0	0

$$S = a \cdot b$$

**Fonction OU – Somme booléenne****Définition –**

Fonction OU – Somme booléenne

$$B \times B \mapsto B$$

$$(a, b) \mapsto a + b$$

Il faut lire *a* OU *b*.

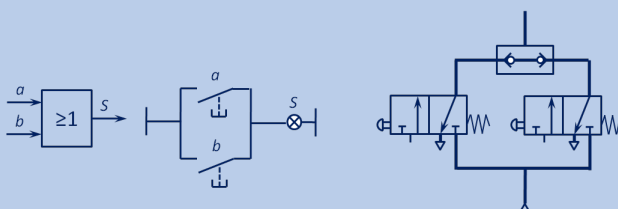
a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

Exemple –

Fonction OU.

a	b	S
0	0	0
0	1	1
1	1	1
1	0	1

$$S = a + b$$



Propriétés des opérateurs de base de l'algèbre de Boole

Propriété –

- ▶ **Commutativité :** $a + b = b + a$ $a \cdot b = b \cdot a$.
- ▶ **Distributivité :** $a \cdot (b + c) = a \cdot b + a \cdot c$ $a + (b \cdot c) = (a + b) \cdot (a + c)$.
- ▶ **Associativité :** $a + (b + c) = (a + b) + c = a + b + c$ $a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c$.
- ▶ **Élément neutre :** $a + 0 = a$ $a \cdot 1 = a$.
- ▶ **Élément absorbant :** $a + 1 = 1$ $a \cdot 0 = 0$.
- ▶ **Complémentarité :** $a + \bar{a} = 1$ $a \cdot \bar{a} = 0$. Par ailleurs $\bar{\bar{a}} = a$.
- ▶ **Idem potence :** $a + a = a$ $a \cdot a = a$.
- ▶ **Identités remarquables :**
 - absorption : $a + a \cdot b = a$;
 - inclusion : $a + \bar{a} \cdot b = a + b$.

Théorème de Morgan

Theorem 18.2.1

$$\overline{\sum_{i=1}^n a_i} = \prod_{i=1}^n \bar{a}_i \quad \overline{\prod_{i=1}^n a_i} = \sum_{i=1}^n \bar{a}_i$$

Exemple –

Déterminer les expressions complémentaires \bar{P} et \bar{Q} des expressions suivantes :
 $P = x \cdot y \cdot (z + t)$ et $Q = \bar{x} \cdot \bar{y} + y + x \cdot t$.

18.3 Fonctions logiques

Un système combinatoire est décrit par une fonction logique qui permet de définir de manière unique la sortie pour une combinaison des entrées.

18.3.1 Table de vérité

La table de vérité d'une fonction logique est une écriture systématique qui consiste à décrire toutes les combinaisons des variables et à y associer les valeurs correspondantes de la fonction.

Exemple –

Compléter la table de vérité suivante et vérifier les théorèmes de De Morgan pour deux variables.

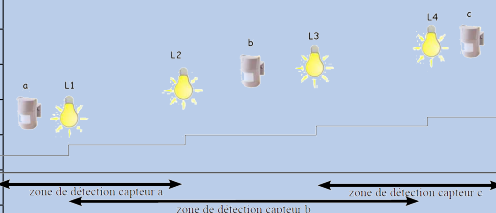
a	b	$\overline{a \cdot b}$	$\overline{a + b}$	$\overline{a} + \bar{b}$	$\bar{a} \cdot \bar{b}$
0	0				
0	1				
1	1				
1	1				

Exemple –

Soit 4 lampes L_1, L_2, L_3, L_4 permettant d'éclairer un escalier. L'allumage de ces 3 lampes est régit par 3 détecteurs de présences a, b et c . La table de vérité est la suivante :

a	b	c	L_1	L_2	L_3	L_4
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	1	0	0	1	1
0	1	0	0	1	1	0
1	1	0	1	1	0	0
1	1	1	1	1	1	1
1	0	1	1	0	0	1
1	0	0	1	0	0	0

Donner l'expression de L_1, L_2, L_3, L_4 .



Il est possible à l'aide de la table de vérité de définir une équation donnant les variables de sortie en fonction des variables d'entrée. Il existe 2 modes d'écriture particuliers :

- somme canonique (somme de produit) : $S = a \cdot b + c \cdot d$;
- produit canonique (produit de sommes) : $S = (a + b) \cdot (c + d)$.

Méthode – Détermination de la somme canonique

Pour chaque ligne où la sortie vaut 1, déterminer la combinaison d'entrées correspondante à l'aide de l'opérateur ET puis sommer ces combinaisons.

Exemple –

Donner l'expression sous forme canonique de L_1 et L_2 à partir de la table de vérité précédente.

Remarque

On utilise la forme de sommes canoniques si le nombre de 1 est inférieur au nombre de 0. Il est souvent nécessaire de simplifier ces équations pour réaliser technologiquement ces fonctions (voir plus loin).

Méthode – Détermination du produit canonique

Déterminer l'expression de \bar{S} ce qui revient à : Déterminer la combinaison d'entrées pour chaque ligne où la sortie vaut 0, les sommer, puis passez au complémentaire en utilisant les théorèmes de De Morgan.

Exemple –

Donner l'expression sous forme de produit canonique de L_1 et L_2 .

Remarque

Cette méthode est utile lorsque la sortie comporte peu de 0 et beaucoup de 1 ou lorsque l'on recherche la fonction complémentaire.

18.4 Simplification des fonctions logiques

La simplification des expressions logiques est destinée à économiser le matériel nécessaire à la réalisation (utilisation d'un composant réalisant plusieurs fonctions identiques) ou diminuer l'importance des équations programmées.

18.4.1 Cellules universelles

Définition – Cellule universelle

Une cellule (ou fonction) est dite « universelle » si elle permet de réaliser les fonctions ET, OU, NON. Il est alors possible de réaliser toutes les fonctions logiques à l'aide de cette seule cellule.

Exemple –

Les cellules NAND (non et – $aNANDb = \overline{a \cdot b}$) et NOR (non ou – $aNORb = \overline{a + b}$) sont des cellules universelles.

Réalisation de la fonction NON : $NON(a) = \bar{a} = \overline{a \cdot a}$. Besoin d'une seule cellule NAND.

Réalisation de la fonction ET : $aETb = a \cdot b = \overline{\overline{a \cdot b}}$. Besoin de deux cellules NAND en cascade.

Réalisation de la fonction OU : $aOUb = a + b = \overline{\overline{a + b}} = \overline{\bar{a} \cdot \bar{b}}$. Besoin de trois cellules NAND en cascade.

De même, la cellule NOR est universelle.

Exemple –

Écrire l'expression logique suivante uniquement avec des opérateurs NAND puis indiquer le nombre minimal d'opérateurs utilisés.

$$F = c \cdot (\bar{a} + \bar{b})$$

18.4.2 Simplification algébrique d'une fonction

Il s'agit d'utiliser les propriétés, théorèmes et identités remarquables de l'algèbre de Boole afin de simplifier une équation. Cette méthode n'est pas forcément simple à utiliser et demande beaucoup d'intuition.

Exemple –

$$\begin{aligned} F(a, b, c) &= \bar{a} \cdot \bar{b} \cdot c + a \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot c = \bar{a} \cdot c \cdot (\bar{b} + b) + a \cdot \bar{b} \cdot c = \bar{a} \cdot c + a \cdot \bar{b} \cdot c = c(\bar{a} + a \cdot \bar{b}) \\ &= c(\bar{a}(1 + \bar{b}) + a \cdot \bar{b}) = c \cdot (\bar{a} + \bar{b}) \end{aligned}$$

Exemple –

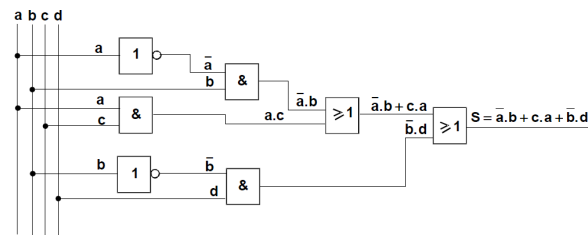
Simplifier par cette méthode l'expression des lampes L_1 et L_2 .

18.5 Représentation des fonctions logiques

Dans un système combinatoire :

- les variables d'entrée sont des informations qui décrivent à un moment donné l'état de certaines parties (position d'un chariot, d'un outil, ...)
- les variables de sortie permettent de décrire l'état dans lequel doivent se retrouver les organes récepteurs.

Une fois l'expression logique obtenue et simplifiée, il est possible d'exprimer graphiquement le comportement du système combinatoire. On utilise pour cela en général le logigramme ou le schéma électrique. On peut également utiliser un schéma pneumatique ou électronique.



18.5.1 Logigrammes pour un codeur incrémental

Exemple –

1. Réaliser le logigramme illustrant le comptage et le décomptage des impulsions provenant d'un codeur incrémental.
2. Réaliser le logigramme illustrant le comptage et le décomptage des impulsions provenant d'un codeur incrémental en multipliant par 4 sa résolution.

Remarque

On appellera front montant de a et notera $\uparrow a$ le passage de la variable a de l'état 0 à l'état 1.

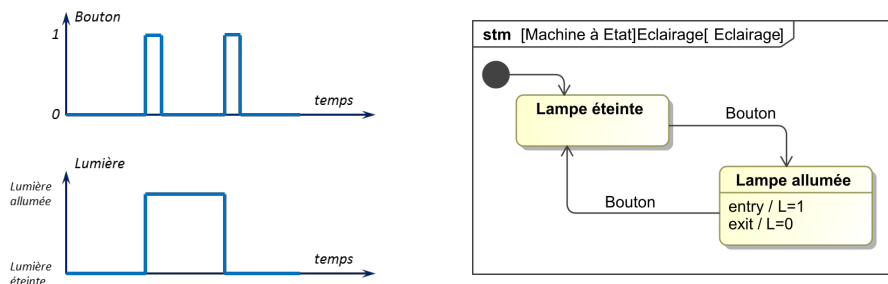
Bibliographie

- [1] Supports de cours de David Violeau, Lycée Saint-Louis, Paris.
- [2] Supports de cours de Florestan Mathurin, Lycée Bellevue, Toulouse.

19.1 La fonction mémoire

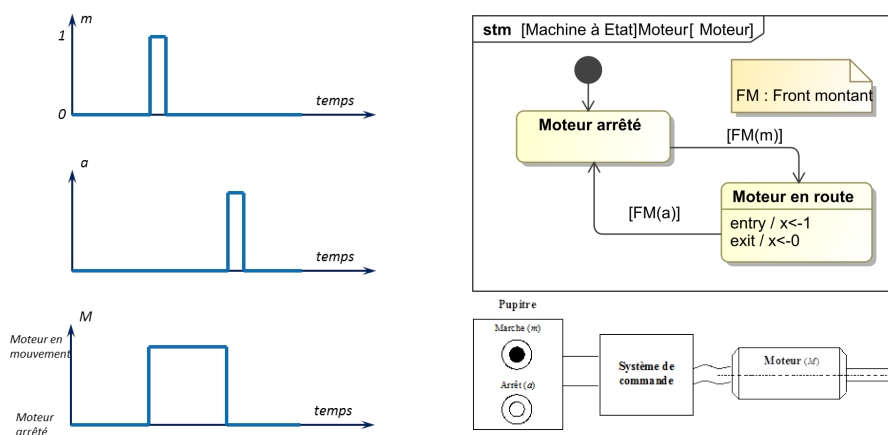
19.1.1 Le chronogramme

Le chronogramme est un outil permettant d'afficher l'état des entrées et des sorties en fonction du temps. Le chronogramme suivant indique le fonctionnement d'une lampe. L'interrupteur «va-et-vient» est relié à un relais.



Ainsi, pour une même combinaison des entrées, on observe des états différents de la sortie.

Le même constat peut être dressé à partir du fonctionnement d'un moteur (*Exemple : fonctionnement du système Doshydro*).



Ici aussi, on remarque qu'une même combinaison des entrées ($a = m = 0$) peut conduire à un état de marche ou d'arrêt du moteur. Une variable interne, ici, une mémoire, permet de mémoriser l'état du moteur.

19.1.2 Notion de mémoire

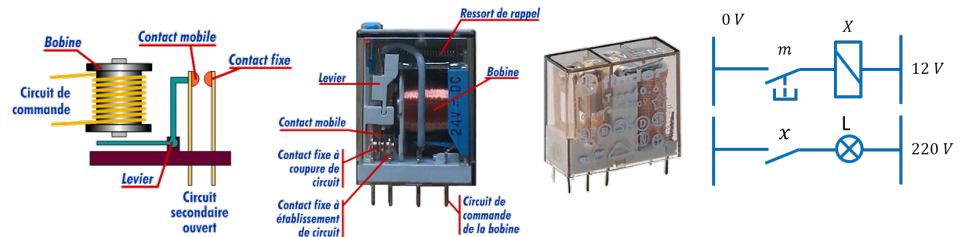
La mémorisation des états est à la base de l'existence des systèmes séquentiels. En notant x la variable interne d'un système, on lui associe une mémoire.

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).

19.1	La fonction mémoire . .	13
19.2	Modélisation du comportement des systèmes séquentiels – Diagrammes d'états [1]	15
19.3	Structures algorithmiques	17

Les mémoires peuvent être réalisées de façon optique (blue-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.



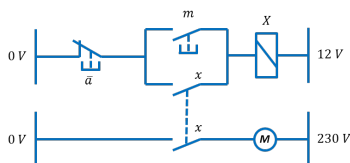
Lorsque l'utilisateur presse le bouton m , le relais X est activé. Ainsi, la variable x est mise à 1 ce qui provoque l'éclairage de la lampe L . À ce stade, si on relâche l'interrupteur m le relais X est désactivé, la variable x passe à l'état 0 et la lampe s'éteint.

Objectif

Comment réaliser une mémoire par l'utilisation d'un relais ? Que se passe-t-il lors de l'appui simultané sur marche et arrêt.

19.1.3 Réalisation d'une mémoire

Exemple : mémoire à effacement prioritaire



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.

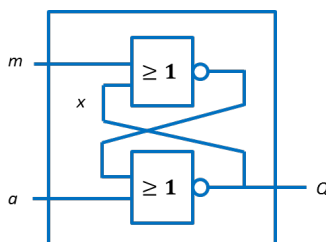
Il est possible de traduire le fonctionnement du moteur par l'équation booléenne suivante :

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

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.



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

Il n'est pas directement possible d'écrire directement la table de vérité de la sortie Q en fonction de a et de b . Il faut avoir recours à une variable interne x .

On retrouve bien l'équation précédente :

$$Q = \bar{m} \cdot \bar{a} \cdot x + \underbrace{m \cdot \bar{a} \cdot \bar{x} + m \cdot \bar{a} \cdot x}_{m \cdot \bar{a}}$$

$$Q = \bar{a} (\bar{m} \cdot x + m) = \bar{a} (x + m)$$

19.2 Modélisation du comportement des systèmes séquentiels – Diagrammes d'états [1]

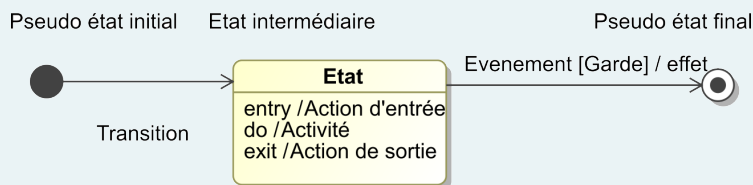
L'outil SysML privilégié pour décrire le fonctionnement d'un système séquentiel sera le diagramme d'état.

Le diagramme de séquence permet de présenter la succession d'étapes liés à un cas d'utilisation d'un point de vue utilisateur. Le diagramme d'état et le diagramme d'activité permettent de décrire l'évolution interne du système.

19.2.1 États – transition

Définition –

Un état caractérise la situation d'un système à un moment donné. Pour passer d'un état à l'autre, utilise des transitions. La succession étape – transition donne donc un diagramme d'états.



Dans un premier temps on peut définir 3 pseudo-états :

- ▶ le pseudo-état initial, unique et indispensable, marque le point de départ du diagramme d'état ;
- ▶ les pseudo-états intermédiaires décrivent le comportement du système pour un cas d'utilisation attendu ;
- ▶ le pseudo-état final (non obligatoire) marque la fin d'un cas d'utilisation.

Dans un diagramme d'état, UN SEUL ÉTAT est actif à la fois.

Pour passer d'un état à l'autre, on utilise des transitions. À ces transitions on peut associer les éléments (facultatifs) suivants :

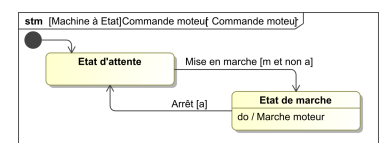
- ▶ un événement : celui-ci est l'élément déclencheur pour utiliser la transition. S'il est absent, il faudra attendre la fin de l'état pour utiliser la transition ;
- ▶ une condition de garde (entre crochet) : il s'agit d'une conditions booléennes à évaluer lorsque l'événement est déclenché. Si l'événement est déclenché **et si** la condition de garde est vraie **alors** la transition est franchie ;
- ▶ l'effet : il s'agit d'une action exécutée au passage de la transition.

Exemple – Commande tout ou rien d'un moteur

Deux états internes du système cohabitent. Dans un cas le moteur est en marche, dans l'autre il est à l'arrêt.

Les conditions de passage d'un état à un autre se font à l'aide des variables d'entrée arrêt « a » et marche « m ».

L'activité « Marche moteur » peut être détaillée dans un diagramme d'activité. Elle comporterait l'action « distribuer l'énergie ». Son exécution ne nécessite pas de temps significatif, elle est associée à un ordre de commande en direction du préactionneur du moteur (un contacteur électrique à arrêt prioritaire).



Définition – Activité et action

En SysML une activité a la propriété de durer un certain temps. Une action est quant à elle instantanée.

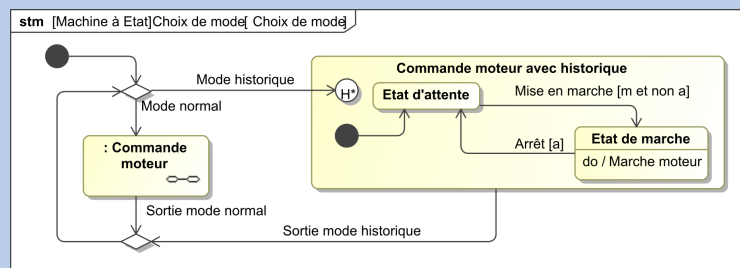
19.2.2 Utilisation des pseudo-états**Définition –**

Les pseudo-états permettent d'utiliser des fonctionnalités avancées sans trop compliquer le diagramme d'état avec le formalisme de base. Il est alors possible de montrer des sélections de séquences d'états (évolutions avec alternatives), des structures parallèles (évolutions simultanées), *etc.*

Exemple – Commande tout ou rien (TOR) d'un moteur

Un diagramme d'états « choix de mode » de niveau hiérarchique supérieur à celui décrit précédemment (« commande moteur »), permet de le démarrer selon deux modes :

- « mode normal » : moteur à l'arrêt dans l'« état d'attente » ;
- « mode historique » : dernier état avant que l'on sorte du diagramme « commande moteur » (« état d'attente » ou « état de marche »).

**19.2.3 États composites**

On ne rentrera pas dans le détail des états composites. Ceux-ci permettent de regrouper, à l'intérieur même d'un état, plusieurs états. Ainsi, on peut retrouver dans un état composite un diagramme d'état complet.

Il existe aussi des états composites « orthogonaux » permettant de traduire la mise en parallèles d'états.

19.3 Structures algorithmiques

19.3.1 Affectation

L'affectation d'une valeur à une variable peut se faire à l'aide d'une action. Cela ne prend pas de temps significatif.



FIGURE 19.1 – Diagramme d'états – Affectation

19.3.2 Groupe d'instructions

Un groupe ou un bloc d'instructions peut être une séquence d'un diagramme d'activité. Cela correspond à une succession d'actions et / ou d'activités.

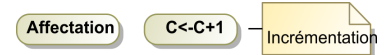


FIGURE 19.2 – Diagramme d'activité – Affectation

19.3.3 Fonctions et procédures

La décomposition d'un algorithme en fonctions et procédures, permet :

- ▶ d'une part, de scinder une problématique générale en plusieurs problématiques élémentaires ;
- ▶ d'autre part, de pouvoir réutiliser des sous-programmes réalisant des tâches élémentaires.

Une procédure comporte une succession d'instructions mais ne renvoie rien. À la fin de l'exécution d'une fonction, il y a le retour d'une valeur, d'une liste, d'un objet, etc.



FIGURE 19.3 – Groupe d'instructions

19.3.4 Structure alternative (conditionnelle)

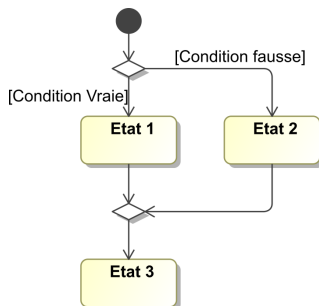


Diagramme d'états – Structure alternative complète

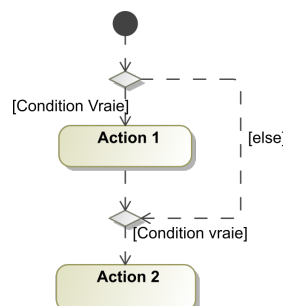


Diagramme d'activité – Structure alternative avec saut

19.3.5 Structure répétitive (itérative)

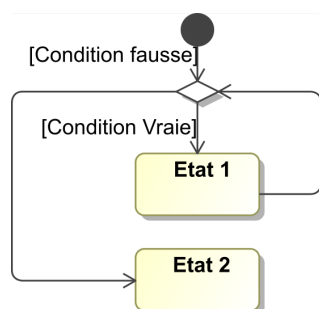


Diagramme d'états – Tant que condition vraie faire ...

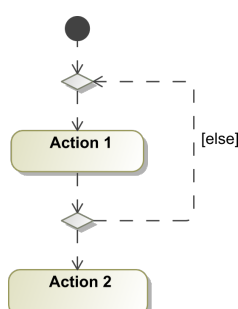


Diagramme d'activité – Répéter... jusqu'à condition vraie

Exemple –

Pour variable = valeur initiale, jusqu'à valeur maximale, faire...

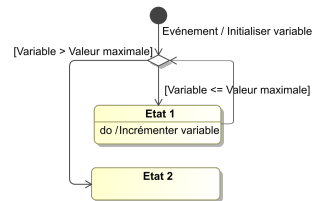


Diagramme d'états

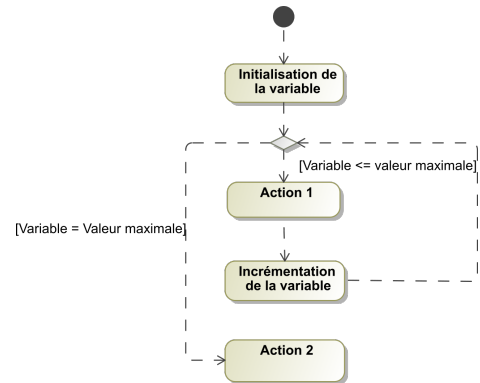


Diagramme d'activité

Bibliographie

- [1] Stéphane Genouël, *Systèmes Séquentiels – Fonction mémoire*, Cours de MPSI – PCSI, Lycée Chateaubriand, Rennes, <http://stephane.genouel.free.fr/>.
- [2] Patrick Beynet et Al., *Sciences Industrielles pour l'Ingénieur*, MPSI – PCSI, Éditions Ellipses.
- [3] Pierre Debout, *Automatique – Systèmes à événements discrets*, Cours de PCSI.
- [4] Thierry Schanen, *Guide des automatismes 9.1*, Pos Industry, 2009.
- [5] Olivier Le Gallo, *Systèmes à événements discrets*, Lycée Clémenceau, Nantes, <http://olivier.legallo.pagesperso-orange.fr/documents/SED.pdf>.