Chapitre

13 Systèmes à événements discrets

On définit les systèmes à temps discret souvent en opposition par rapport aux systèmes à temps continus.

Dans les systèmes à temps continus (les systèmes linéaires et asservis), les variables évoluent continûment et peuvent, en général, être décrites par un ensemble d'équations différentielles. Nous avons vu dans le chapitre sur les systèmes linéaires quelques outils mathématiques permettant d'étudier ces systèmes.

Les systèmes à événements discrets permettent eux de décrire le comportement de systèmes qui évoluent lorsqu'un événement est présent.

Considérons un ascenseur, le déplacement de l'ascenseur entre deux étages peut être décrit par un ensemble d'équations différentielles (P.F.D. pour obtenir l'équation différentielle du mouvement, les équations électriques du moteur…) ; à l'aide de ces équations on peut par exemple réaliser un asservissement de vitesse de la cabine. Par contre la gestion des appels à chaque étage, la demande de déplacement d'un passager (les multiples demandes), l'arrivée à un étage, le départ d'un étage,…correspond elle, à un traitement d'événements discrets (appui sur un bouton d'étage, sélection d'un étage, ouverture/fermeture de la porte, …).

13.1 Typologie des systèmes

13.1.1 Les systèmes continus

Le modèle d'un système continu ne comprend que des variables continues, de plus, le temps est également une variable continue (figure 13.2). Le temps et les variables prennent leurs valeurs dans $\mathbb{R}$. C'est le grand domaine des systèmes représentés par des équations différentielles. Les systèmes linéaires continus invariants que nous avons étudié au premier semestre sont une partie des systèmes continus.

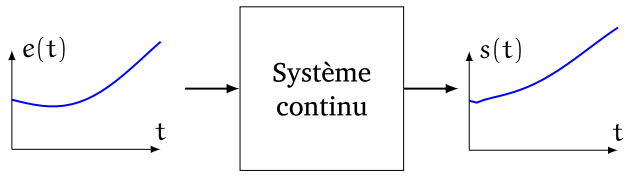


FIGURE 13.1 – Système continu - Information continue

— Les variables continues sont des variables qui prennent leurs valeurs sur le domaine des réels $\mathbb{R}$.

13.1.2 Les systèmes numériques échantillonnés

Dans les systèmes numériques échantillonnés, le temps n'est plus une variable continue mais une variable discrète. Le temps est en effet représenté comme une suite infinie d'instants repérés par des entiers naturels $\mathbb{N}$.

Les systèmes informatiques ne peuvent traiter que des systèmes échantillonnés en effet :

— le temps de calcul ne pouvant être nul, le système informatique ne peut prélever l'information qu'à des instants particuliers, cela nécessite un échantillonnage du temps ;
— une conversion analogique/numérique pour ramener les variables dans un domaine (codage sur 8, 16, 32,…bits) que peut traiter le système numérique ;
— la fréquence d'échantillonnage doit respecter le critère de Shannon sans être trop importante pour ne pas saturer les capacités du système de traitement.

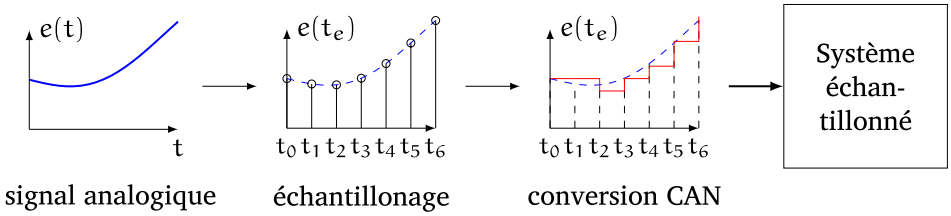


FIGURE 13.2 – Système échantillonné

Les systèmes échantillonnés sont la représentation des systèmes analogiques dans l'espace numérique.

13.1.3 Les systèmes discrets

Un système discret est un système dans lequel, les variables sont toutes discrétisées, soit par nature (lampe éclairée / éteinte), soit parce que l'on ne prend en compte que l'état de la variable (four en marche ou température atteinte, sans se préoccuper de l'évolution de la température).

Dans les systèmes discrets, le temps est en général continu ou discrétisé.

Les systèmes discrets ne manipulent que des variables logiques.

13.1.4 Variable logique

Une variable logique est une variable, à laquelle on associe deux états. On peut ainsi associer à un grand nombre de phénomènes physiques un état logique (porte ouverte/fermée, vrai/faux, voyant éclairé/éteint, 0/1, 0 V/5 V,…).

Un signal réel est une grandeur physique en général continue qui peut donc prendre une infinité de valeurs. Pour associer à ce signal, un signal binaire (0,1), il est nécessaire de fixer des seuils. Le passage d'un seuil caractérise le passage de l'état 1 à 0 et réciproquement - figure 13.3).

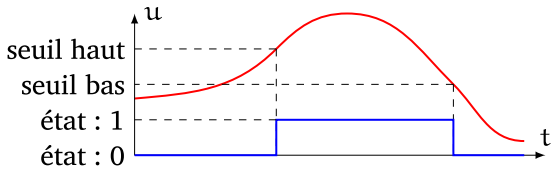


FIGURE 13.3 – Variable et état logique

Le seuil pour passer de l'état bas (0) à l'état haut (1) peut être identique ou différent de celui pour passer de l'état haut à l'état bas.

a) Convention

Il est nécessaire de distinguer : la grandeur physique représentée, le support de l'information logique, le nom attribué, l'état logique qu'elle peut prendre.

Par convention on établit une correspondance entre l'état physique de la grandeur et sa valeur logique correspondante (logique positive ou logique négative).

L'état logique peut être noté « 0 » ou « 1 » ou encore « L » (Low) ou « H » (High). On parle aussi de niveau logique haut ou niveau logique bas, état haut ou état bas.

On parle de logique positive lorsqu'on associe à l'état logique 1 l'état actif de la variable et de logique négative dans le cas contraire (figure 13.4).

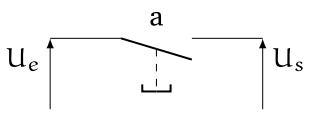
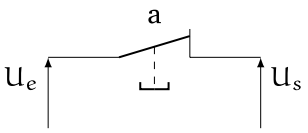
	Contact établissement de circuit		Contact coupure de circuit	
état physique	Actionné	Non actionné	Actionné	Non actionné
état électrique	Passant	Non passant	Non passant	Passant
état logique	1	0	1	0
	interrupteur  ne conduit pas au repos, le courant passe s'il est actionné		poussoir de réfrigérateur  conduit au repos, le courant passe que s'il n'est pas actionné	

FIGURE 13.4 – Modélisation d'un contact électrique

13.1.5 Les systèmes à événements discrets - S.E.D.

Pour les systèmes à événements discrets, les variables sont discrètes et le temps n'est connu que lors des changements d'états de ces variables.

L'évolution temporelle n'est prise en compte que lors des changement d'état, on ne se préoccupe pas de l'écoulement du temps.

Ainsi, si on s'intéresse à une barrière de péage, les événements qui font évoluer le système sont : la présence d'un véhicule, la validation du paiement, le cycle d'ouverture, le départ du véhicule, le cycle de fermeture, et cela quel que soit le temps qui s'écoule entre deux véhicules ou le temps mis par l'automobiliste à payer.

13.2 Typologie des S.E.D.

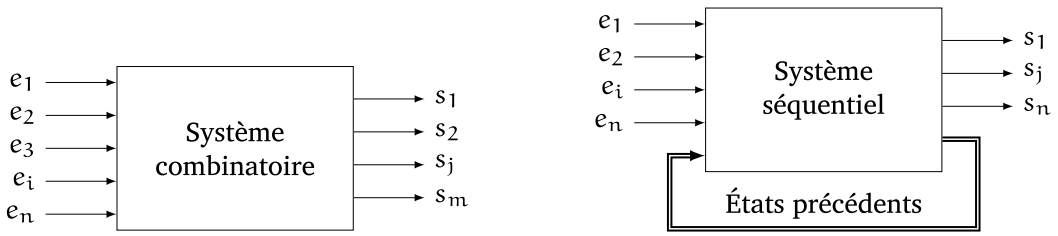
— Systèmes combinatoires

Un système est dit combinatoire (figure 13.5a), lorsque la ou les sorties ne dépendent que de la combinaison des entrées. La même cause (même combinaison des entrées) produit toujours le même effet (même état des sorties).

L'effet disparaît lorsque la cause disparaît.

Chaque sortie est une fonction des entrées :

$$s_j = f(e_1, e_2, \ldots, e_i, \ldots, e_n)$$



(a) Système combinatoire

(b) Système séquentiel

FIGURE 13.5 – Typologie des S.E.D.

13.2.1 S.E.D. séquentiels

Un système est dit séquentiel (figure 13.5b), lorsque la ou les sorties dépendent de la combinaison des entrées et de l'état précédent des sorties. Une même cause (même combinaison des entrées) peut produire des effets différents.

Le temps peut aussi être une des causes d'évolution des sorties. L'effet peut persister après la disparition de la combinaison l'ayant provoqué.

Chaque sortie, à chaque instant, est le résultat d'une combinaison des entrées et de l'état précédent des sorties ou des variables internes.

$$s_j = f(e_1, e_2, \ldots, e_i, \ldots, e_n, E_1, E_i)$$

13.3 Systèmes combinatoires

13.3.1 Algèbre de Boole

L'algèbre de Boole ou algèbre logique est l'algèbre définie pour des variables ne pouvant prendre que deux états. On appelle variable booléenne, une variable ne prenant que deux états, pour une variable a, les deux états sont a et NON a (noté $\overline{a}$).

a) Opérateurs logiques fondamentaux

On distingue 4 opérateurs fondamentaux.

Opérateur OUI : opérateur identité, le comportement est décrit par la table de vérité suivante : si S est la sortie et a la variable d'entrée :

Table de vérité

a	S
0	0
1	1

Équation logique : $S = a$

Opérateur NON : opérateur négation

a	S
0	1
1	0

Équation logique : $S = \overline{a}$
lire **NON a** (ou a barre)
Codage en Python

```
S=not a
```

Propriétés

$$\overline{1} = 0 \quad \text{et} \quad \overline{0} = 1$$
$$\text{involution} \quad \overline{\overline{a}} = a$$

Opérateur ET : produit logique

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

Équation logique : $S = a \cdot b$
lire **a ET b**
Codage en Python

```
S=a and b
```

Propriétés

$$\begin{cases} 0 \cdot 0 = 0 \\ 0 \cdot 1 = 0 \\ 1 \cdot 0 = 0 \\ 1 \cdot 1 = 1 \end{cases} \quad \text{et} \quad \begin{cases} a \cdot 0 = 0 \\ a \cdot 1 = a \\ a \cdot a = a \\ a \cdot \overline{a} = 0 \end{cases}$$

— Le produit logique est commutatif, comme dans l'algèbre classique : $a \cdot b = b \cdot a$.

— 1 est l'élément neutre pour le produit logique : $a \cdot 1 = a$.

— 0 est l'élément absorbant pour le produit logique : $a \cdot 0 = 0$.

Opérateur OU : somme logique

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

Équation logique : $S = a + b$
lire **a OU b**
Codage en Python

```
S=a or b
```

Propriétés

$$\begin{cases} 0+0=0 \\ 0+1=1 \\ 1+0=1 \\ 1+1=1 \end{cases} \quad \text{et} \quad \begin{cases} a+0=a \\ a+1=1 \\ a+a=a \\ a+\overline{a}=1 \end{cases}$$

— La somme logique est commutative : $a + b = b + a$.
— 0 est l'élément neutre pour le produit logique : $a + 0 = a$.
— 1 est l'élément absorbant pour le produit logique : $a + 1 = 1$.

Dans l'algèbre usuelle, l'addition n'a pas d'élément absorbant.

b) Propriétés des opérateurs logiques

— Commutativité, associativité :
Le produit et la somme logique sont commutatifs et associatifs.
— Propriétés combinées de la somme et du produit
  — Distributivité du produit logique par rapport à la somme logique

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

  — Distributivité de la somme logique par rapport au produit logique (cette propriété n'existe pas dans l'algèbre usuelle).

$$a + b \cdot c = (a + b) \cdot (a + c)$$

— Absorption

$$a + a \cdot b = a$$
$$a + \overline{a} \cdot b = a + b$$

c) Théorèmes de De Morgan

— Le complément d'un produit est égal à la somme des compléments des termes du produit.

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

— Le complément d'une somme est égal au produit des compléments des termes de la somme.

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$

d) Règle du consensus

La règle du consensus est une règle de simplification des fonctions logiques, elle prend deux formes, celle d'une somme de produit, celle d'un produit de somme.

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

Remarque : on pourra utiliser la description par une table de vérité pour vérifier ces assertions, puis retrouver le résultat par des simplifications à partir de l'algèbre de Boole.

13.3.2 Table de vérité

Une table de vérité est un tableau qui permet de décrire le fonctionnement d'un système combinatoire. L'état de chaque entrée et de chaque sortie est représenté par sa valeur logique. Toutes les combinaisons possibles des entrées sont étudiées pour déterminer l'état des sorties. Pour décrire complètement un système combinatoire comportant n entrées, il faut étudier les combinaisons possibles des entrées.

Pour un système combinatoire comportant n entrées, la table de vérité comporte 2^n combinaisons des variables d'entrée, donc 2^n lignes.

Il est possible de déterminer l'équation de fonctionnement en recherchant toutes les valeurs pour lesquelles la sortie est vraie ($=1$). L'équation de fonctionnement est égale à la somme logique de toutes les combinaisons pour lesquelles la sortie est vraie.

Exemple guidé : Va et vient

L'éclairage d'un couloir est commandé par deux interrupteurs, un à chaque extrémité (d à droite, g à gauche), la lampe est notée L.

— Une personne traverse le couloir de droite à gauche, en entrant, elle bascule l'interrupteur d pour éclairer la lampe, à l'autre extrémité, elle appuie sur g pour éteindre.

— Une seconde traverse à sa suite dans l'autre sens, elle bascule g pour éclairer, elle éteint à l'autre extrémité (d).

— Une troisième arrive à sa suite, éclaire le couloir en appuyant sur g, elle éteint en sortant.

On supposera qu'au début, les deux interrupteurs sont dans la même position. À cet état repos est associée la valeur logique 0. La figure 13.6 décrit chronologiquement les différents états des deux interrupteurs et de la lampe.

	Action	d	g	L	Commentaires
	État repos	0	0	0	La lampe est éteinte.
Première personne	Entrée en d	1	0	1	Appuie sur d (d = 1), la lampe s'éclaire (L = 1).
	traversée	1	0	1	La lampe reste éclairée.
	Sortie en g	1	1	0	Appuie sur g (g = 1), la lampe s'éteint.
Deuxième personne	Entrée en g	1	0	1	(g = 0) la lampe s'éteint.
	traversée	1	0	1	
	Sortie en d	0	0	0	(d = 0) la lampe s'éteint.
Troisième personne	Entrée en g	0	1	1	(g = 1), la lampe s'éclaire.
	Traversée	0	1	1	
	Sortie en d	1	1	0	(d = 1) la lampe s'éteint.

FIGURE 13.6 – Fonctionnement d'un va et vient

On peut traduire de fonctionnement à l'aide d'une table de vérité.

	d	g	L	
les deux boutons au repos	0	0	0	
un bouton appuyé	1	0	1	$L = d \cdot \overline{g} = 1$
les deux boutons appuyés	1	1	0	$L = \overline{d} \cdot g = 1$
l'autre bouton relâché	0	1	1	

En recherchant les combinaisons, pour lesquelles la sortie est vraie, on peut déterminer l'équation logique décrivant le fonctionnement.

La lampe s'éclaire donc pour la combinaison suivante des variables d'entrée :

$$L = d \cdot \overline{g} + \overline{d} \cdot g$$

13.4 Fonctions logiques de base à 2 variables

13.4.1 Fonctions de base

À chaque opérateur de base, on associe une fonction logique représentée par un symbole graphique que l'on retrouve dans le tableau 13.1. À l'aide de ces symboles, on peut construire des schémas logiques (logigramme) traduisant l'équation logique.

Les fonctions logiques élémentaires permettent de décrire tous les fonctionnements logiques (voir le schéma logique du va et vient figure 13.7a).

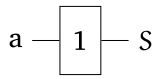
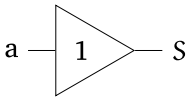
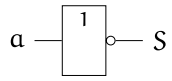
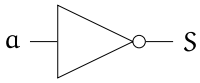
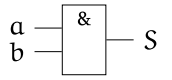
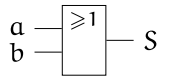
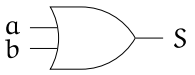
Fonction	table de vérité	symbole EU	symbole US	équation
OUI	a S 0 0 1 1			$S = a$
NON	a S 0 1 1 0			$S = \overline{a}$
ET	a b S 0 0 0 1 0 0 0 1 0 1 1 1			$S = a \cdot b$
OU	a b S 0 0 0 1 0 1 0 1 1 1 1 1			$S = a + b$

TABLEAU 13.1 – Fonctions logiques de base

13.4.2 Fonctions complexes

Il existe aussi d'autres fonctions[1], souvent utilisées en électronique pour construire des schémas plus complexes, ou plus compacts (tableau 13.2).

Les fonctions NON-OU (NOR) et NON-ET (NAND) sont des fonctions universelles, qui permettent de réaliser toutes les autres fonctions logiques.

Les fonctions NON-OU et NON-ET sont principalement utilisées en technologie électronique cablée (circuits intégrés) pour optimiser les circuits (1 seul type de composant pour toutes les fonctions).

1. Ces fonctions ne sont pas explicitement au programme.

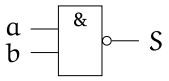
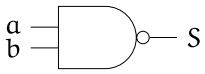
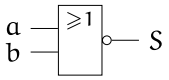
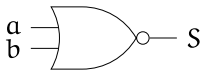
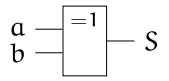
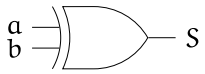
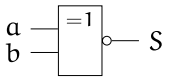
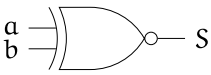
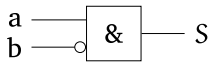
Fonction	table de vérité	symbole EU	symbole US	équation
NAND (NON-ET)	a b S 0 0 1 1 0 1 0 1 1 1 1 0			$S = \overline{a \cdot b} = \overline{a} + \overline{b}$
NOR (NON-OU)	a b S 0 0 1 1 0 0 0 1 0 1 1 0			$S = \overline{a + b} = \overline{a} \cdot \overline{b}$
XOR OUX ou exclusif	a b S 0 0 0 1 0 1 0 1 1 1 1 0			$S = \overline{a} \cdot b + a \cdot \overline{b}$ $S = a \oplus b$
XNOR NON- OUX non ou exclusif	a b S 0 0 1 1 0 0 0 1 0 1 1 1			$S = \overline{a} \cdot \overline{b} + a \cdot b$ $S = \overline{a \oplus b}$
INH inhibition	a b S 0 0 0 1 0 1 0 1 0 1 1 0			$S = a \cdot \overline{b}$

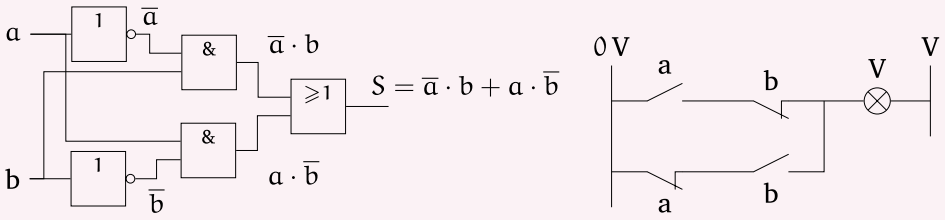
TABLEAU 13.2 – Tableau des fonctions logiques - suite

13.4.3 Logigramme

Un logigramme ou schéma logique est la représentation schématique d'une fonction logique obtenue en utilisant les symboles graphiques associés à chaque fonction. La figure 13.7a représente le schéma logique ou logigramme du va et vient.

Exemple guidé : Va et vient (suite)

On complète l'exemple du va et vient avec le circuit logique et le schéma électrique.



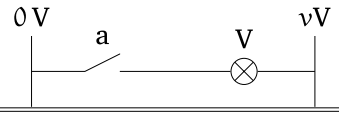
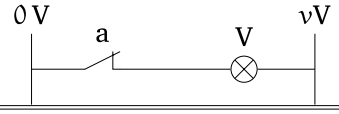
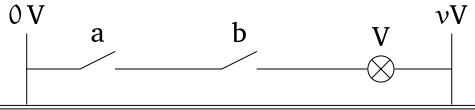
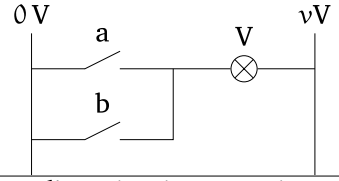
(a) Schéma logique (b) Technologie électrique

FIGURE 13.7 – Va et vient

13.4.4 Réalisation des fonctions logiques en technologie électrique câblée

a) Fonctions de base

Les fonctions logiques de base peuvent être réalisées à partir du câblage des contacts électriques en série et/ou en parallèle.

Fonction	Schéma
OUI	
NON	
ET	
OU	

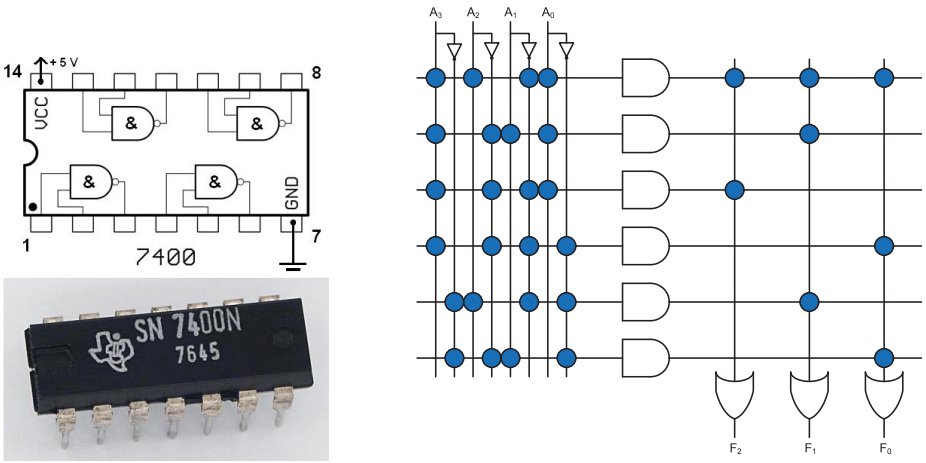
La figure 13.7b correspond à la réalisation d'un circuit va et vient (ou exclusif) en technologie électrique. On remarque les fonctions $\overline{a}$ et $\overline{b}$ sont réalisées par des contacts à ouverture.

13.4.5 Réalisation des fonctions logiques en technologie électronique

Les différentes fonctions logiques sont directement réalisées à partir de circuit intégré à base de semi-conducteur.

Ces fonctions existent aussi bien sous la forme de composant discret que de circuit programmable.

Les fonctions discrètes sont la transposition des fonctions logiques, il suffit de les connecter comme sur le schéma.



(a) Circuit intégré Nand 7400 (b) Schéma de principe d'un PAL

FIGURE 13.8 – Technologie électronique

Il existe aussi des composants de type P.A.L. (Programmable Array Logic) qui permettent de réaliser l'ensemble des fonctions possibles en combinant les opérateurs logiques OU, ET et NON. La figure 13.8b montre la structure interne d'un circuit P.A.L. à 4 entrées et à 3 sorties. L'équation souhaitée est obtenue en réalisant par programmation la connexion entre les lignes et les colonnes du réseau.

La sortie F_2 du schéma est ainsi :

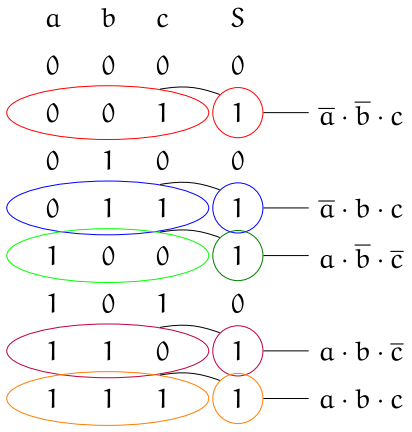
$$F_2 = A_0 \cdot \overline{A_1} \cdot A_2 \cdot A_3 + A_0 \cdot \overline{A_1} \cdot \overline{A_2}$$
$$F_2 = A_0 \cdot \overline{A_1} \cdot A_3$$

13.5 Détermination et simplification des fonctions logiques

13.5.1 Simplification à partir des relations de l'algèbre

Soit un comportement combinatoire décrit par une table de vérité, commençons par rechercher toutes les combinaisons pour lesquelles la sortie est vraie.

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



La sortie est vraie pour la somme logique de toutes les combinaisons vraies.

$$S = \overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot \overline{c} + a \cdot b \cdot c$$

Simplification à partir des règles de l'algèbre de Boole.

$$S = \overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot \overline{c} + a \cdot b \cdot c \qquad \text{(factorisation)}$$

$$S = \overline{a} \cdot c \cdot \left(\overline{b} + b\right) + a \cdot \overline{b} \cdot \overline{c} + a \cdot b \cdot (\overline{c} + c) \qquad (b + \overline{b} = 1)$$

$$S = \overline{a} \cdot c + a \cdot \overline{b} \cdot \overline{c} + a \cdot b$$

$$S = \overline{a} \cdot c + a \cdot \left(\overline{b} \cdot \overline{c} + b\right) \qquad \text{(absorbtion : } \overline{b} \cdot \overline{c} + b = b + \overline{c}\text{)}$$

$$S = \overline{a} \cdot c + a \cdot (\overline{c} + b)$$

$$S = \overline{a} \cdot c + a \cdot \overline{c} + a \cdot b$$

$$S = a \oplus c + a \cdot b \qquad \text{(ou exclusif : } \overline{a} \cdot c + a \cdot \overline{c} = a \oplus c\text{)}$$

Finalement

$$S = a \oplus c + a \cdot b$$

13.6 Systèmes séquentiels

Avant d'aborder les systèmes à événements discrets, nous allons définir les systèmes séquentiels.

La plupart des traitements ne sont pas uniquement combinatoires mais souvent séquentiels. Dans un traitement séquentiel le système doit pouvoir mémoriser certaines valeurs pour pouvoir les réutiliser.

Un système est dit séquentiel, lorsque la ou les sorties dépendent de la combinaison des entrées et de l'état précédent des sorties.

Une même cause (même combinaison des entrées) peut produire des effets différents et l'effet peut persister si la cause disparaît.

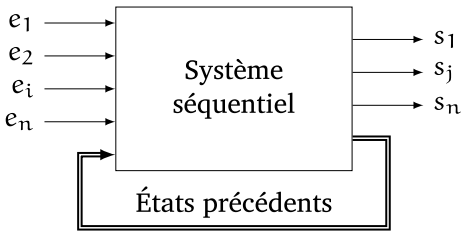
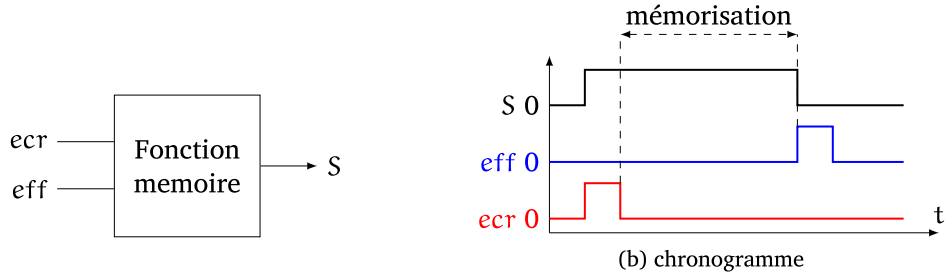


FIGURE 13.9 – Système séquentiel

13.6.1 Fonction mémoire

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 (figure 13.10). À l'apparition du signal eff (effacement), la sortie repasse à l'état 0.

Le maintien de la sortie correspond à l'effet mémoire.



(b) chronogramme

FIGURE 13.10 – Fonction mémoire

a) Fonction mémoire à effacement prioritaire

Soit un système constitué de deux boutons poussoirs, m (marche) et a (arrêt) et d'un voyant V.

Un appui sur m éclaire le voyant V, si on relâche le bouton, la sortie reste à l'état 1, si on appuie sur a, quel que soit l'état du voyant V et de m, le voyant s'éteint. Nous allons construire la table de vérité traduisant ce comportement.

	m	a	V
les deux boutons sont relâchés et le voyant est éteint	0	0	0
le bouton m est appuyé, le voyant s'éclaire	1	0	1
le bouton m est relâché, le voyant reste éclairé	0	0	1
le bouton a est appuyé, le voyant s'éteint	0	1	0
les boutons m et a sont appuyés, le voyant est éteint	1	1	0

On constate sur cette table que le fonctionnement n'est pas combinatoire, en effet, pour deux états identiques, ($m = 0$ et $a = 0$) la sortie V a deux états différents.

Pour mettre en évidence l'état interne mémorisé, nous allons introduire une nouvelle entrée dans le tableau, $V_{t-\delta t}$ qui représente l'état précédent de V.

On construit la table en remplissant dans l'ordre les différents états des entrées et sorties, en reportant l'état de la sortie V dans $V_{t-\delta t}$ de la ligne suivante.

m	a	$V_{t-\delta t}$	V
0	0	0	0
1	0	0	1
1	0	1	1
0	0	1	1
0	1	1	0
0	1	0	0
1	1	0	0
1	1	1	0

En ajoutant l'état transitoire qui recopie la sortie sur l'entrée, la table comporte maintenant 8 combinaisons différentes des 3 variables m, a, et $V_{t+\delta t}$. Le fonctionnement est maintenant combinatoire (8 combinaisons pour $2^3 = 8$ possibles).

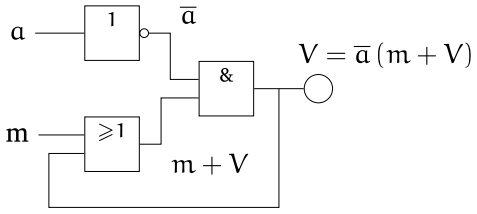
D'où l'équation logique du voyant :

$$
\begin{aligned}
V &= m \cdot \overline{a} \cdot \overline{V_{t-\delta t}} + m \cdot \overline{a} \cdot V_{t-\delta t} + \overline{m} \cdot \overline{a} \cdot V_{t-\delta t} \\
V &= \overline{a} \cdot \left(m \cdot \overline{V_{t-\delta t}} + m \cdot V_{t-\delta t} + \overline{m} \cdot V_{t-\delta t} \right) \\
V &= \overline{a} \cdot \left(m \cdot \left(\overline{V_{t-\delta t}} + V_{t-\delta t} \right) + \overline{m} \cdot V_{t-\delta t} \right) \\
V &= \overline{a} \cdot \left(m + \overline{m} \cdot V_{t-\delta t} \right) \\
V &= \overline{a} \cdot \left(m + V_{t-\delta t} \right)
\end{aligned}
$$

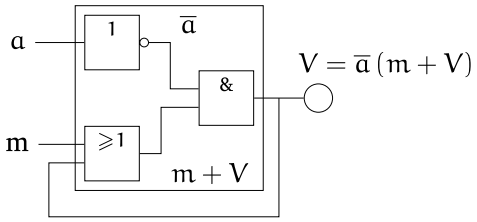
Le temps δt peut être aussi petit que possible, on peut donc écrire :

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

Et le schéma logique :



que l'on peut mettre sous la forme :



où on retrouve, la structure d'un système séquentiel.

On dit que le fonctionnement de cette mémoire est à « effacement prioritaire », en effet en cas de demande de mise à 1 et à mise à 0 simultanée, la mise à 0 l'emporte (dernière ligne de la table de vérité). On retrouve le fonctionnement décrit sur le chronogramme de la figure 13.11.

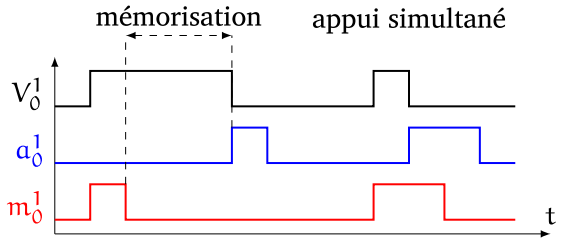


FIGURE 13.11 – Mémoire dite à « effacement prioritaire »

13.7 Machine à états - *State machine diagram*

Il n'est pas toujours possible de décrire le comportement d'un système séquentiel, à partir d'une simple table de vérité. Le nombre de combinaisons possibles 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 un diagramme 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 d'événements extérieurs.

Le diagramme d'état est un diagramme constitué d'états représentés par des blocs reliés par des arcs supportant les transitions.

13.7.1 États

On distingue 3 états de base.

— **L'état initial** représenté par un cercle plein noir.

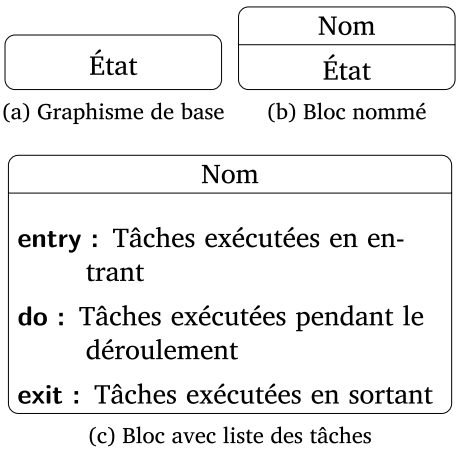
Cet état (pseudo état) est nécessaire, il indique le point d'entrée dans un graphe.

— **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 (figure 13.12).



(a) Graphisme de base (b) Bloc nommé

(c) Bloc avec liste des tâches

FIGURE 13.12 – Graphismes d'un bloc

13.7.2 Les transitions

Les transitions d'un état à l'autre (ou sur lui-même) sont représentées par un lien orienté, la condition de transition est décrite sur le lien. Une transition représente le passage instantané d'un état vers un autre. 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.

La syntaxe d'une transition peut être relativement complexe et comporter, jusqu'à trois éléments distincts, elle est notée **Événement [Condition de garde] /Action** :

— **Événement** : c'est la condition principale qui déclenche le passage d'un état à un autre. Cela peut être,

  — le changement d'état d'une variable (b : passage de 0 à 1 de la variable b et $!b$ passage de 1 à 0),

  — la validation d'une condition notée avec le mot clef **when** (température atteinte par exemple : **when**($T > T_0$)),

  — une condition temporelle : **after(2s)**.

— **Condition de garde** : c'est une condition booléenne qui complète l'événement et qui en contraint l'effet. Si une transition comporte un événement et une condition de garde : **événement [garde]** entre un état source et un état de destination, pour que le système évolue, il faut que l'événement soit vrai et que la condition de garde soit vraie.

— **Action** ou **activité** : décrit les actions qui sont exécutées pendant le franchissement.

— Chaque terme de la transition est optionnel.

La transition est franchie lorsque les conditions de franchissement associées sont vraies :

— Si l'événement apparaît (déclencheur) et si la condition de garde est vraie à ce moment alors la transition est franchissable : si l'état source était actif alors il se désactive et l'état de destination s'active. L'activité, si elle existe, est exécutée avant celles situées dans l'état de destination. La condition de garde n'est prise en compte qu'au moment de l'occurrence de l'événement.

— Si aucun événement n'est écrit, alors il est toujours vrai.

— Si aucune condition de garde n'est écrite, alors elle est toujours vraie.

— Une transition qui ne contient aucune condition de franchissement est dite automatique, elle est franchie dès que la tâche à l'intérieur de l'état source est terminée.

a) Transition réflexive

Arc orienté qui relie le même état (source = destination). Cela permet d'exécuter les activités associées aux instants « exit » et « entry » de nouveau.

b) Transition alternative

Lorsqu'un choix se présente entre deux ou plusieurs évolutions, il est possible d'utiliser le mot clef **else** pour préciser l'alternative. On peut aussi préciser l'alternative en utilisant le pseudo état de **point de décision** (◇)dans la transition (figure 13.13).

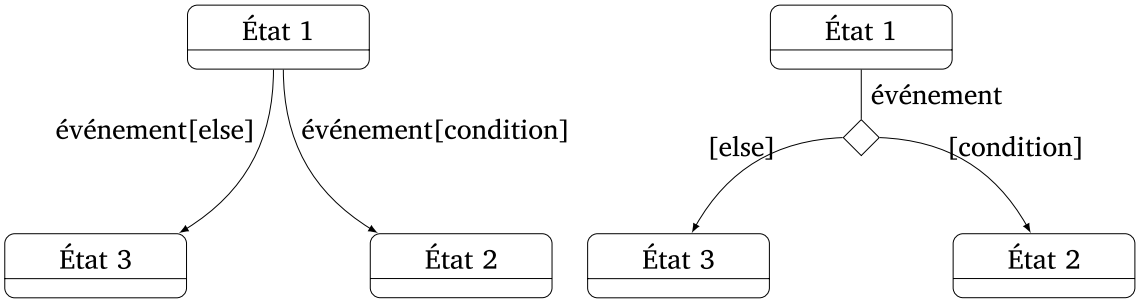


FIGURE 13.13 – Transition alternative

La clause **else** ne peut se trouver que dans une condition de garde.

13.7.3 Les blocs

Un bloc est donc une entité qui précise l'état du système lorsque le bloc est actif.

Le graphisme de base est un rectangle précisant l'état et/ou les actions à réaliser pendant cette phase de vie (figure 13.12a). Cette description peut être précisée en nommant le bloc (figure 13.12b).

Il est parfois nécessaire de préciser le comportement d'un état, on peut ainsi le détailler en prenant en compte 3 phases (figure 13.12c) :

— ce qui doit se passer en entrant dans l'état, cette phase est introduite par le mot clef :**entry**. Les taches ou actions décrites dans cette partie sont exécutées à chaque fois que l'on rentre dans le bloc, quelle que soit la durée de ce bloc ;

— ce qui doit se passer tant que l'état est actif, le mot clef **do** introduit cette phase. Les tâches du **do** sont réalisées tant que l'état est actif. Une action réalisée dans le **do** ne prend fin que si un événement est présent ;

— ce qui doit se passer en quittant le bloc, cette phase est décrite dans le mot clef **exit**. Les tâches ou actions décrites dans cette partie sont exécutées à chaque fois que l'on rentre dans le bloc, quelle que soit la durée de ce bloc

a) États composites

Un bloc peut aussi contenir une sous-machine à état. Cette encapsulation permet de décrire des fonctionnements plus complexes sans trop alourdir le graphisme. Un tel état est appelé état **composite** ou **super état**.

— L'activation de l'état composite entraîne l'activation du pseudo-état initial interne (rond noir).

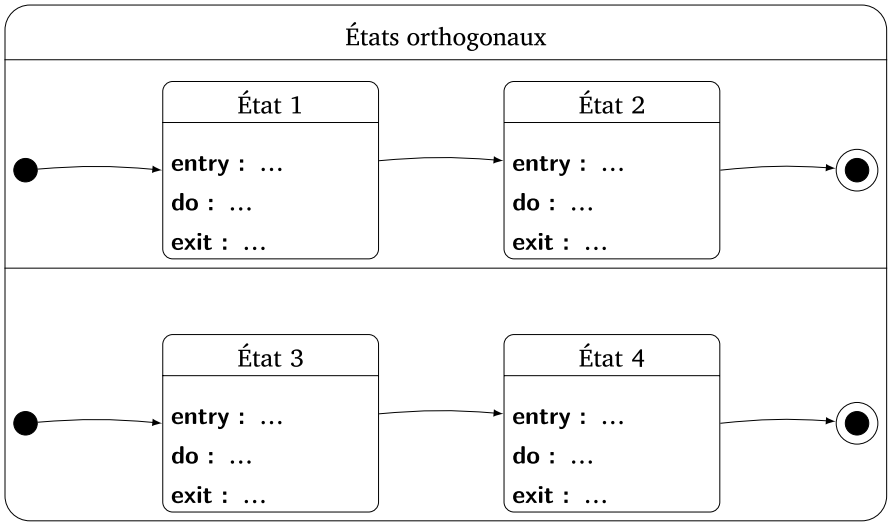
— La désactivation de l'état composite entraîne la désactivation de l'état actif.

— Chaque sous-état peut aussi être un état composite et ainsi de suite…

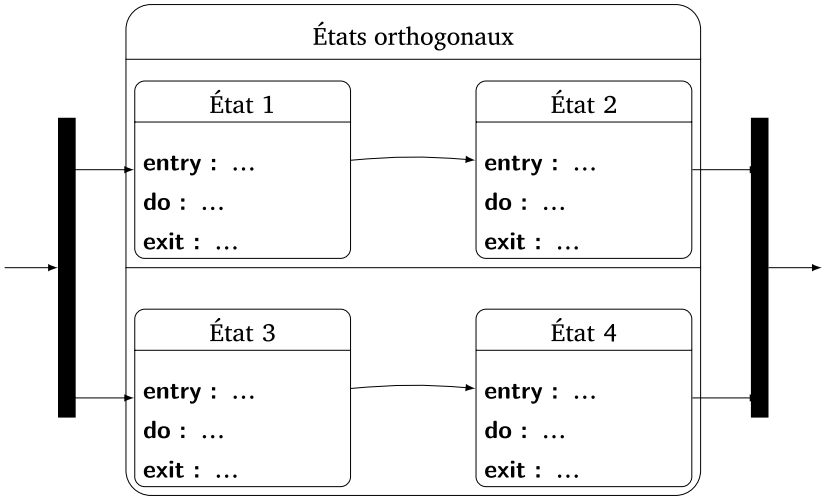
— Un état composite peut contenir des activités associées à entry, do et exit. entry et do sont traités avant l'activation du pseudo-état initial de la région, exit est traité au moment de la désactivation de l'état composite.

b) États orthogonaux

Un bloc peut aussi contenir plusieurs états dits **orthogonaux**, les états orthogonaux sont exécutés simultanément dès l'activation de l'état encapsulant (figure 13.14).



(a) Cycles exécutés simultanément



(b) Transitions fork et join

FIGURE 13.14 – États orthogonaux

Quand un état orthogonal est terminé, les autres états orthogonaux peuvent rester actifs.
Une transition sortant de l'état englobant termine simultanément tous les états orthogonaux.

Transition fork et join : Les deux pseudos états **fork** et **join** sont représentés par une barre épaisse :

— le fork permet de d'activer plusieurs états orthogonaux ;

— le join de regrouper plusieurs transitions issues de plusieurs états orthogonaux.

c) Prise en compte de l'historique

Il est parfois nécessaire de mémoriser l'état actuel avant de le quitter afin de le retrouver lors du retour dans l'état et de poursuivre le déroulement.

L'historique, est précisé par le pseudo-état (H) ou (H*) :

— *shallow history* (H) : 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* (H*) : idem que précédemment mais avec la propagation de l'historique à tous les sous-états composites de niveaux hiérarchiques inférieurs.

13.7.4 Exemples

a) Lampe - Bouton poussoir

Une lampe L est éclairée par une impulsion sur un bouton poussoir B_p , une impulsion sur ce même bouton, l'éteint.

La figure 13.15 montre deux possibilités analogues pour décrire ce fonctionnement

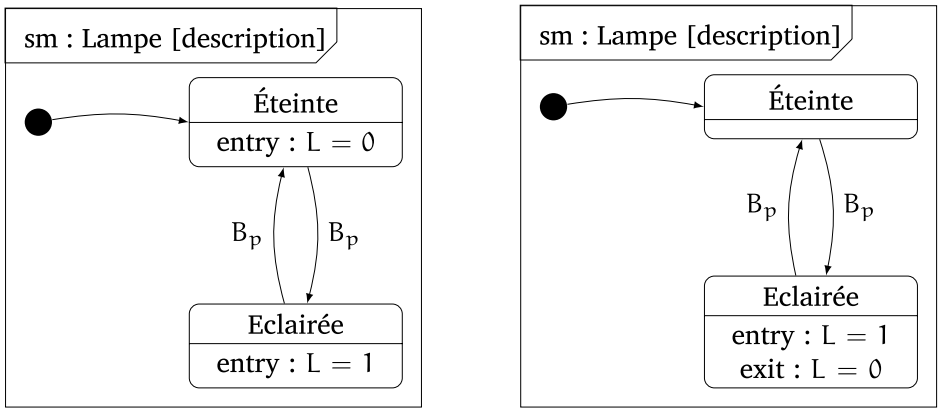


FIGURE 13.15 – Lampe et bouton poussoir : deux descriptions d'un même fonctionnement

On note dans cet exemple :

— l'utilisation de **entry** et **exit** ;

— les multiples représentations d'une même description.

Ces deux solutions ne sont pas les seules possibles.

b) Lampe temporisée

Un éclairage extérieur est mis en route par le passage d'une personne sous le détecteur (d), il fonctionne alors pendant 30 s. Si une nouvelle personne se présente alors que l'éclairage est actif, celui-ci redémarre pour 30 s.

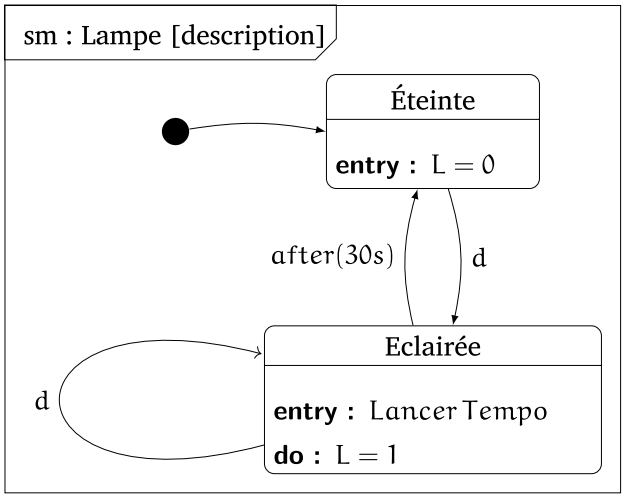


FIGURE 13.16 – Lampe temporisée

La figure 13.16 décrit le fonctionnement attendu, la boucle ré-entrante dans l'état « Éclairée » permet de relancer l'attente.

On note dans cet exemple :

— l'utilisation de **after** pour représenter une temporisation ;

— l'utilisation d'une transition réflexive ;

— l'action décrite dans le **do**, n'est pas arrêtée par la transition réflexive.

c) Fonctionnement d'une porte motorisée

La porte peut être dans l'un des trois états : « Ouverte », « Fermée » ou « Verrouillée ».

Le système peut répondre aux événements « Ouvrir », « Fermer », « Verrouiller » et « Déverrouiller ».

Le matin, la porte est déverrouillée et le soir, elle est verrouillée. La commande est réalisée par un bouton à clef placé à l'extérieur.

Les boutons « Ouvrir » et « Fermer » sont placés à l'intérieur, dans le local de surveillance.

Un détecteur permet de vérifier que le passage est libre.

Un voyant orange clignotant est alimenté lorsque la porte se déplace.

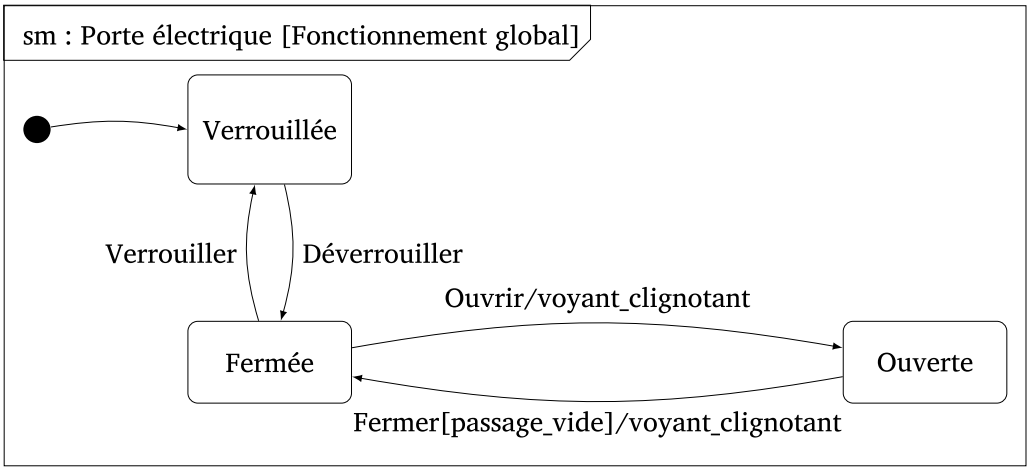


FIGURE 13.17 – Porte électrique

— À la mise sous tension (état initial) la porte est verrouillée.

— L'événement « Déverrouiller » met la porte dans l'état « Fermée ».

— L'événement « Verrouiller » met la porte dans l'état « Verrouillée ».

— La condition de passage « Ouvrir/voyant clignotant » met la porte dans l'état « Ouverte ». Pendant le déplacement, l'action « voyant clignotant » a lieu.

— L'évolution de l'état « Ouverte » à l'état « Fermée » ne peut se faire que si l'événement « Fermer » est vrai et que la condition de garde « [passage vide] » est vraie.

— Pendant le déplacement, l'action « voyant clignotant » a lieu.

d) Exemple guide

Exemple guidé : Lave-vaisselle

On considère un lave-vaisselle dont le fonctionnement est le suivant :

— Le cycle de lavage comporte un cycle de remplissage/lavage/vidange suivi d'un cycle remplissage/rinçage/vidange. La durée du cycle de lavage dépend du programme de lavage choisi.

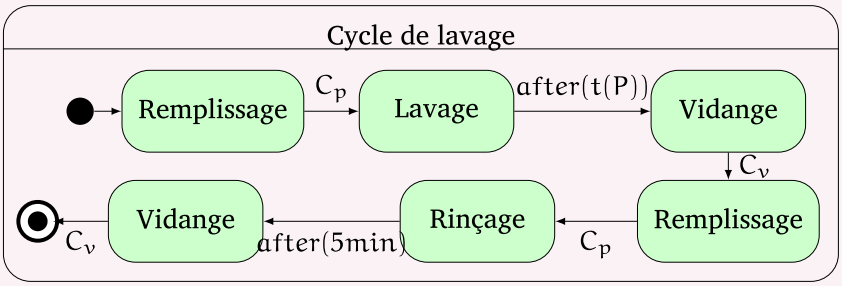
— Pendant les cycles de lavage et rinçage, l'eau est chauffée et maintenue à la température

prévue par le cycle.

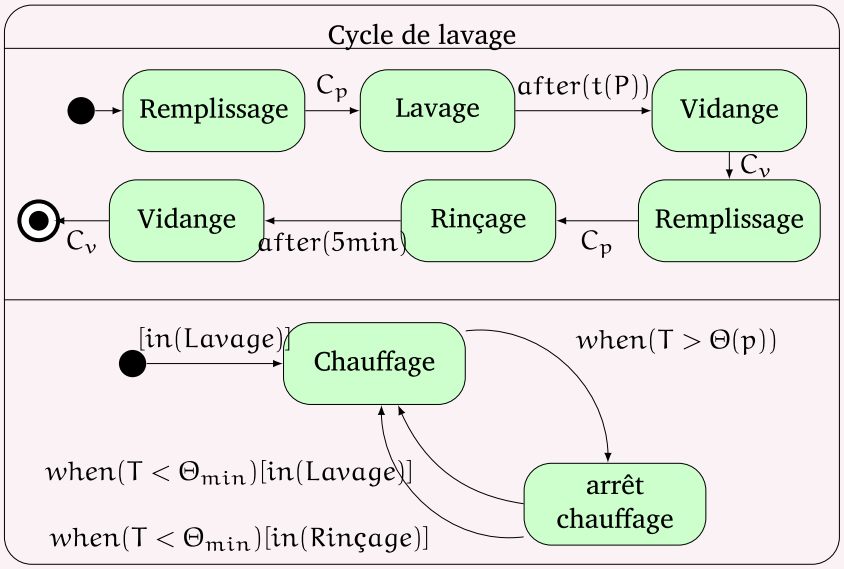
— Après la mise sous tension, l'utilisateur choisit parmi les cinq cycles de lavage en appuyant sur le bouton à impulsion `prog` qui incrémente le numéro du programme. Le cycle démarre après l'appui sur le bouton `start`.

— L'ouverture de la porte interrompt tout cycle en court, le cycle poursuit son déroulement lorsque la porte est refermée et après un appui sur `start`.

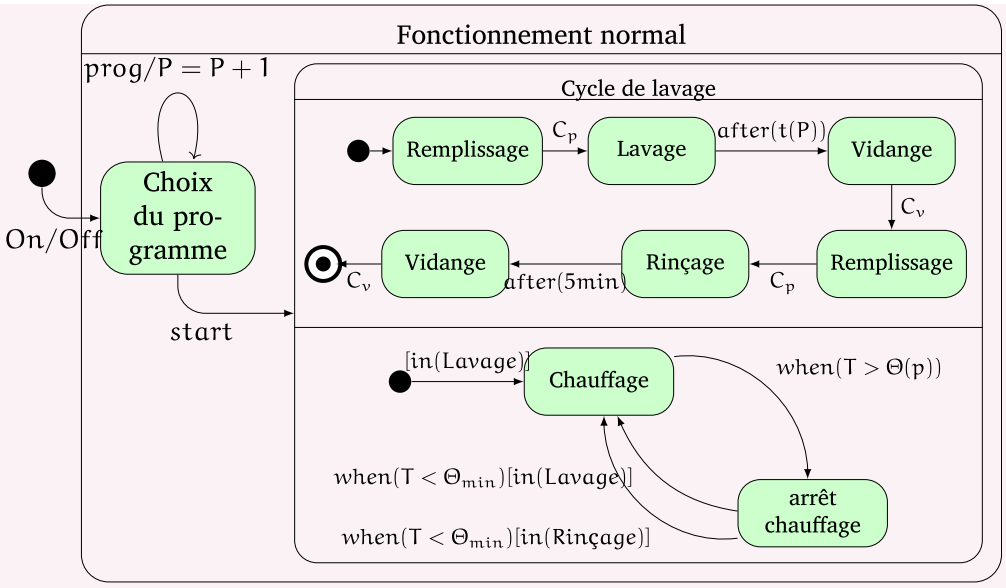
Cycle de lavage seul : Le graphe ci-dessous décrit le cycle de lavage seul, les événements cuve pleine et cuve vide sont notés C_p et C_v et $\mathtt{after(t(P))}$ la durée du lavage pour le programme numéro P.



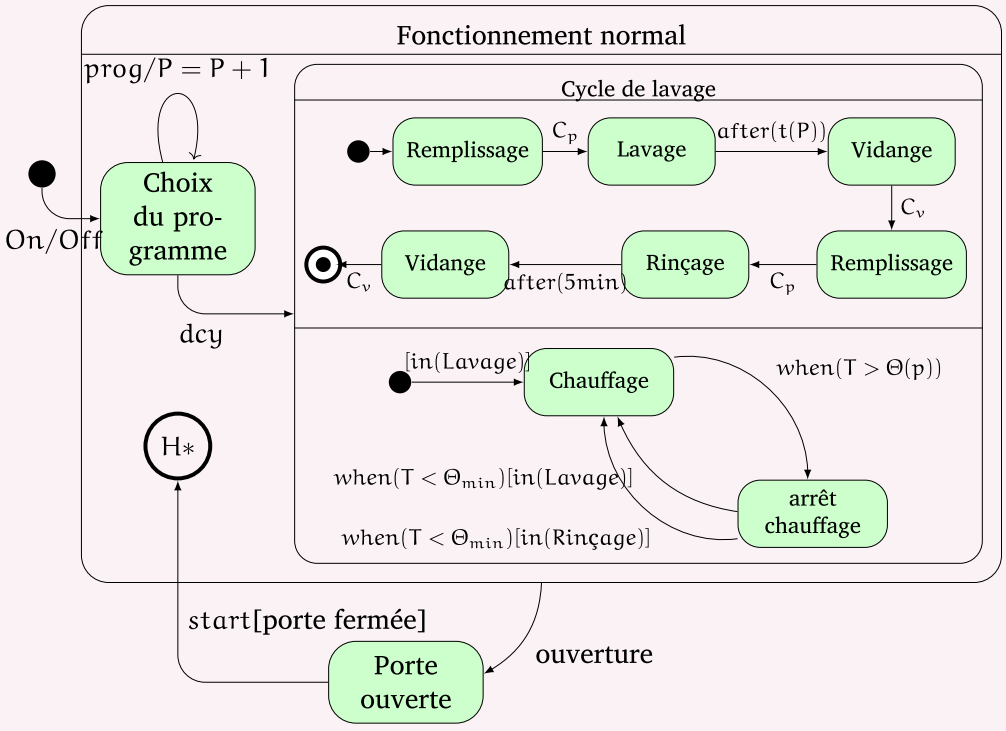
Lavage et chauffage : Les tâches de lavage et de chauffage se déroulent en même temps, il est donc nécessaire de préciser deux états orthogonaux. Le chauffage ne doit se réaliser que lorsque le système se retrouve dans l'état **Lavage** (mot clef **in**), cesser lorsque la température prévue par le programme n°P est atteinte (mot clef **when**) et reprendre si elle descend au-dessous d'une valeur minimale, à la condition que le système se trouve dans les états **Lavage** ou **Rinçage**.



Choix du programme et lancement : À chaque impulsion sur le bouton `prog`, le numéro du programme est incrémenté, le bouton `start` lance le cycle de lavage.



Ouverture de la porte : L'ouverture de la porte provoque l'arrêt du cycle de lavage, quel que soit l'état dans lequel il se trouve. Lorsque la porte est refermée, un appui sur `start` permet le redémarrage du cycle à partir de l'état dans lequel il était (on note **H***) précisant la reprise sur un historique profond.



13.8 Feuille de travaux dirigés n°13

Exercice 1 - Code barre entrelacé

X MP 2001 *Corrigé page 31*

Présentation

Le calculateur est équipé d'un lecteur optique de codes à barres capable de lire le code 2/5 INTERLEA VED ("2 parmi 5 entrelacé") permettant d'identifier automatiquement les pièces à souder. Chaque pièce est identifiée par un nombre de quatre chiffres décimaux C3, C2, C1 et C0 ?

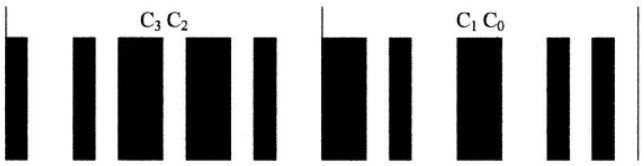


FIGURE 13.18 – Code entrelacé

Le code 2/5 INTERLEAVED utilise 5 bits (2 valent 1 et 3 valent 0) pour coder un chiffre décimal. Les chiffres de rang impair (C3 et C1) sont codés sur les barres noires, les chiffres de rang pair (C2 et C0) sont codés sur les espaces entre les barres noires. Les 1 sont codés par les barres ou espaces "larges" (utilisant deux largeurs de base), les 0 sont codés par les barres ou espaces "étroits" (utilisant une largeur de base).

Chaque chiffre de 0 à 9 est codé sur 4 bits a, b, c et d de poids respectifs 1, 2, 4 et 7. Le code est complété par un bit de contrôle de parité e.

	a	b	c	d	e
Poids	1	2	4	7	0
0	...				
9	...	...			

Q1. Déterminer les codes des chiffres de 1 à 9 en présentant le résultat sous forme de tableau. En déduire le code du chiffre 0 en justifiant son unicité. Déterminer le nombre correspondant au code de la figure 13.18.

Le calculateur traduit chaque chiffre de ce code à barres en un nombre binaire codé sur les quatre bits s_3, s_2, s_1 et s_0 (le poids du bit s_i vaut 2^i).

Q2. Établir la table de vérité des sorties s_i en fonction des entrées a, b, c, d et e. En déduire l'équation simplifiée de la sortie s_3.

Exercice 2 - Renault Twizy

Mines Ponts MP 2017 *Corrigé page 31*

Mise en situation

.1. Présentation

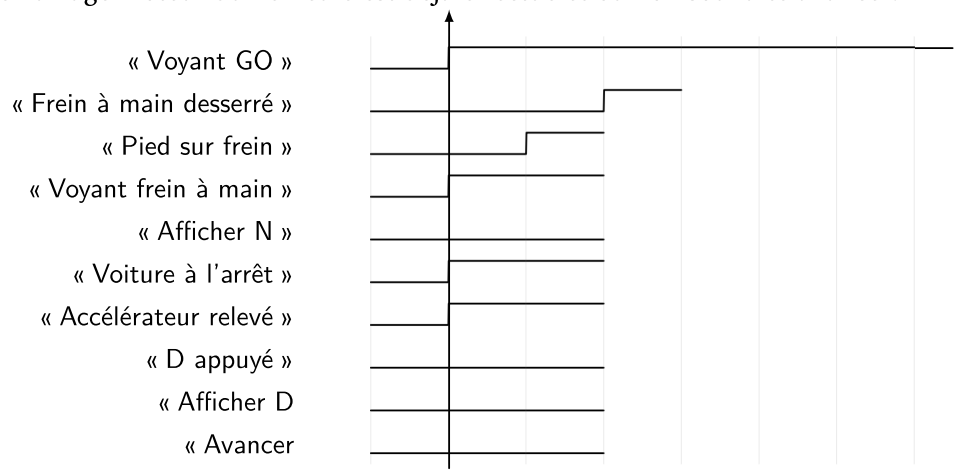
Dans le contexte actuel d'économie des énergies fossiles et de réduction des émissions de gaz nocifs, la Twizy est un quadricycle à propulsion électrique fabriqué par le constructeur automobile Renault. Elle constitue une alternative aux modes de déplacement urbains actuels. Se situant entre un scooter et une voiture, elle adopte un mode de propulsion entièrement électrique pour une autonomie d'environ 100 km. Son rayon de braquage très court et ses dimensions réduites lui permettent de stationner perpendiculairement au trottoir. Revers de la médaille, la Renault Twizy ne propose que deux places en tandem et un compartiment de 31 dm^3 sous le siège arrière. Logée sous le siège avant, la batterie, d'une capacité de 6,1 $kW \cdot h$ (105 $A \cdot h$), se charge complètement en 3 h 30 sur une simple prise secteur via un câble d'une longueur de trois mètres.

FIGURE 13.19 – Renault Twizy

Le diagramme fourni sur la figure 13.20 reprend la description du cycle de démarrage et d'arrêt du fonctionnement du véhicule Twizy en respectant la norme des diagrammes d'état définie dans le langage SysML.

Remarque : quand toutes les conditions sont réunies, dans l'état « Marche Avant », il faut appuyer sur l'accélérateur pour faire avancer le véhicule.

Q1. Après lecture du diagramme d'état, compléter le chronogramme dans la configuration où le démarrage moteur du véhicule est déjà effectué et où l'on souhaite avancer.



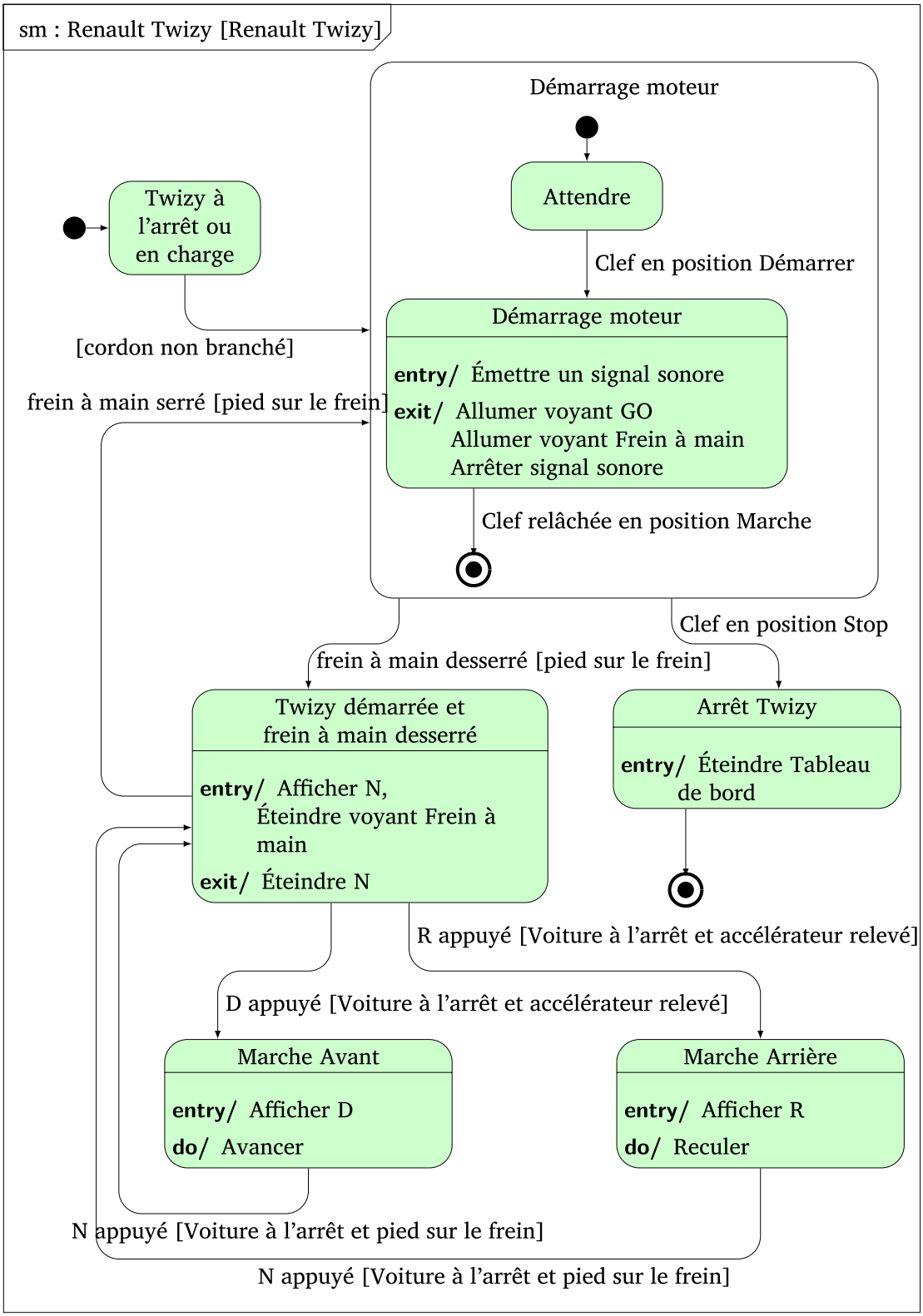


FIGURE 13.20 – Diagramme d'état partiel

Exercice 3 - Lève-vitre

CCINP PSI 2017 *Corrigé page 32*

A. Mise en situation

A.1. Fonctionnement

L'utilisateur demande simplement à ce que l'ouvrant se déplace jusqu'à une position prédéfinie. Une brève action sur l'interrupteur de sa part entraîne le déplacement complet de l'ouvrant. Dès lors, le système de contrôle/commande gère le déplacement de l'ouvrant dans le cas normal, mais aussi en cas de dysfonctionnement (perte de fonctionnalité ou présence d'un obstacle sur le trajet de la vitre). Il faut donc assurer un fonctionnement sûr et robuste du système d'ouvrant piloté automatisé pour éviter que le système blesse un occupant.

Le réducteur du lève-vitre est constitué d'un dispositif roue et vis sans fin. La roue possède $Z = 53$ dents et la vis est constituée d'un filet (figure 13.22). Le câble s'enroule sur le tambour de diamètre $D = 41{,}5\,\text{mm}$, solidaire de la roue. Le câble est solidaire du coulisseau sur lequel est fixée la vitre.

On prendra dans la suite la valeur $r = 0{,}39\,\text{mm}\cdot\text{rad}^{-1}$.

Q1. Justifier la valeur de r.

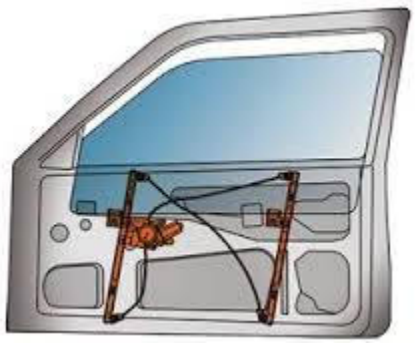


FIGURE 13.21 – Lève-vitre

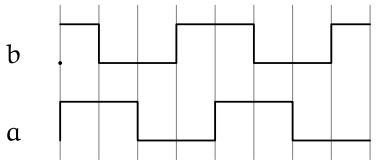
A.2. Modélisation du contact avec un obstacle

Dans le cas d'un lève-vitre, l'obstacle est souvent une main. Des études montrent que les phalanges sont très résistantes et peuvent supporter des efforts allant de 250 à 1 150 N en fonction des différentes phalanges. On modélise donc l'obstacle entre le châssis et la vitre par une raideur k (cette raideur peut varier de 10 à 50 N/mm).

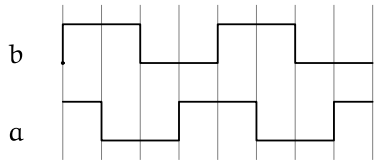
La position de la vitre est détectée à l'aide de capteurs à effet Hall situés près du moteur (figure 13.22). Une roue magnétique possédant 2 paires de pôles nord/sud est solidaire de l'axe du rotor du moteur. Deux capteurs à effet Hall sont placés en quadrature et repèrent les changements de champ magnétique (fronts montants et descendants) de la roue en fonction de la rotation du moteur.

Q2. Tracer l'évolution des signaux des deux capteurs à effet Hall (a et b) pour un tour du moteur dans le sens trigonométrique. On considère que le signal émis par le capteur a le niveau logique 1 lorsque le pôle nord de l'aimant est face au capteur.

sens trigonométrique



sens anti-trigonométrique



Q3. Quels sont les intérêts d'utiliser deux capteurs à effet Hall placés en quadrature ?

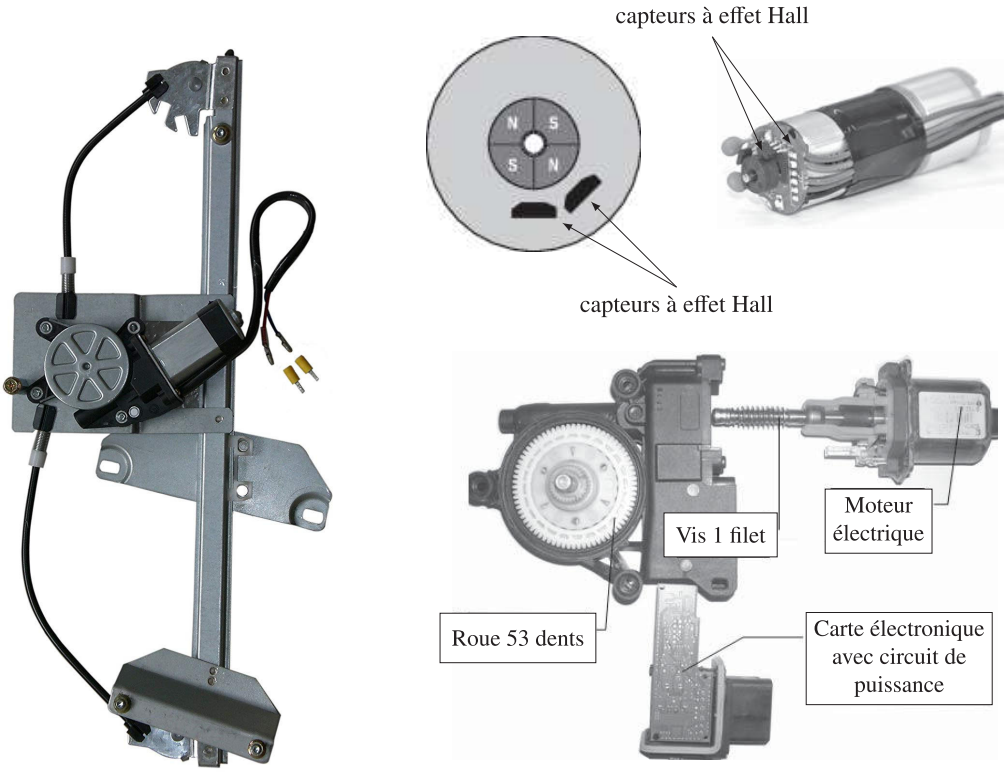
Q4. Montrez que $c = a \oplus b$ permet d'obtenir une précision de $1/8^{e}$ de tour du moteur.

Q5. Déterminer le plus petit déplacement de la vitre en mm qu'il est possible de mesurer avec ce capteur.

Q6. En prenant une raideur d'obstacle $k = 20\,\text{N}\cdot\text{mm}^{-1}$ correspondant à la dernière phalange de l'auriculaire, combien d'impulsions auront été comptées à partir du moment où la phalange commence à être écrasée jusqu'à ce que l'effort dans la phalange soit de $50\,\text{N}$? Commenter ce résultat.

A.3. Algorithme de commande

L'algorithme finalement mis en place se base sur la variation des temps mesurés entre deux impulsions successives. Après la détection d'une impulsion, un prédicteur temporel permet de déterminer le temps auquel la prochaine impulsion est attendue. Si la nouvelle impulsion intervient avant le



(a) structure du lève vitre

(b) Moteur et capteur à effet Hall

FIGURE 13.22 – Constituants du lève vitre

temps prédit, alors il n'y a pas de blocage, sinon un blocage est détecté et une alarme est déclenchée. En réalité, cette technique conduit à de fausses détections et une modification permettant d'améliorer la robustesse est de ne déclencher l'alarme qu'au bout de 3 dépassements du temps prédit. Cet algorithme est résumé sur la figure suivante pour lequel :

appui bouton haut est un événement qui survient quand le bouton « monter la vitre » est actionné,

M+ est la variable permettant de faire tourner le moteur dans le sens de la montée de la vitre, **M0** permet d'arrêter le moteur, cet événement permet soit la montée soit l'arrêt,

impulsion est un événement qui survient à chaque nouvelle impulsion envoyée par les capteurs,

fin course haut est un événement permettant de détecter l'arrivée en position haute de la vitre,

prediction() est une fonction qui renvoie le temps auquel la prochaine impulsion est attendue,

alarme permet d'activer l'alarme.

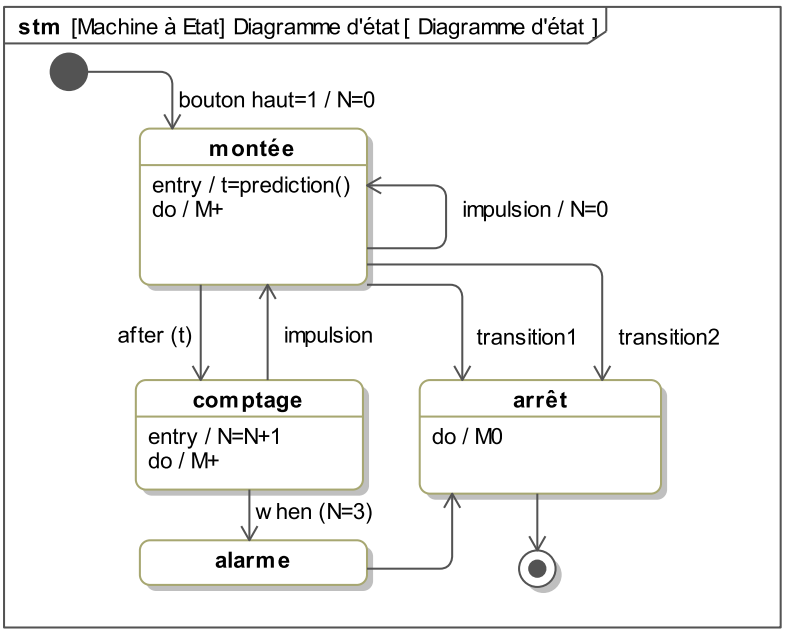
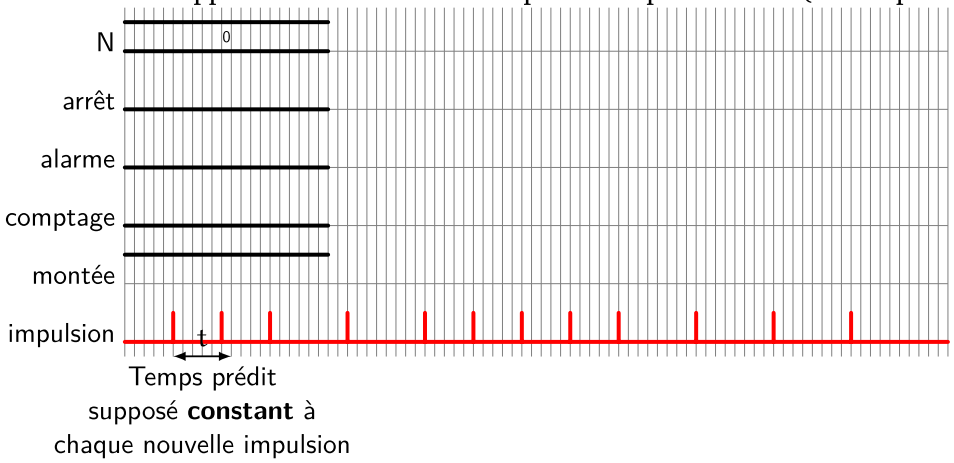


FIGURE 13.23 – Algorithme du fonctionnement

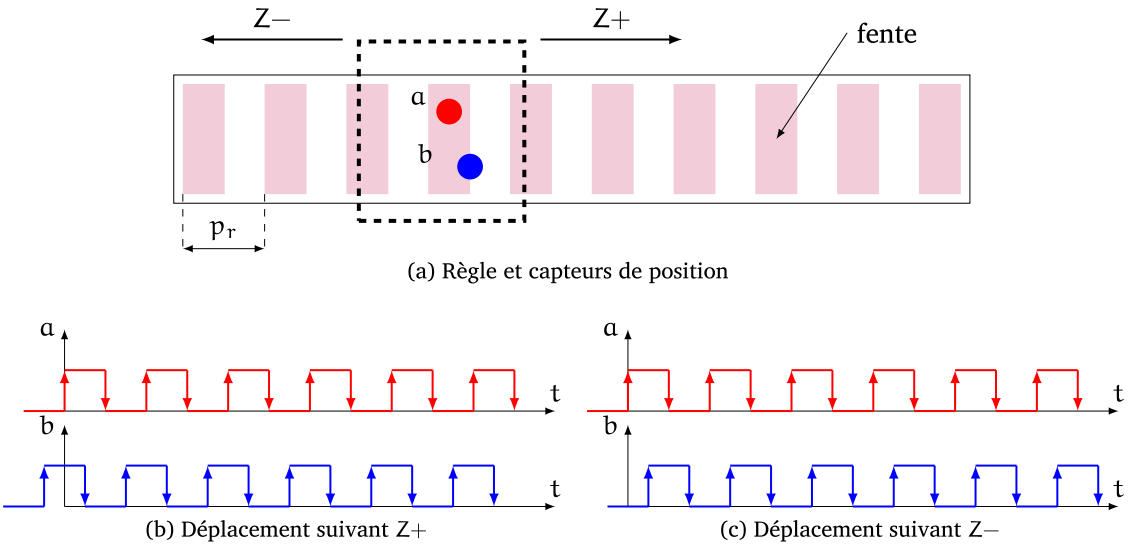
Q7. Donner l'expression des deux conditions notées « transition 1 » et « transition 2 » permettant de passer de l'état « montée » à l'état « arrêt » directement.

Q8. Compléter le chronogramme du document réponse DR4 en indiquant par des créneaux les durées pendant lesquelles un état est activé et l'évolution du contenu de la variable N. La durée de l'alarme et de l'arrêt est supposée très faible et sera représentée par un dirac (une impulsion).



Exercice 4 - Capteur de position incrémental
Extrait du sujet CCinp MP 2006 *Corrigé page 32*

La mesure de position est effectuée par une règle fixe collée au bâti munie de fentes espacées d'une distance p_r ($p_r = 4$ mm, pas de la règle) et par deux capteurs optiques a et b montés sur le bloc moteur et décalés d'un quart de pas p_r (figure 13.24a). Ces capteurs renvoient l'information 0 s'ils se situent face à une fente ou 1 dans le cas contraire.



(a) Règle et capteurs de position

(b) Déplacement suivant Z+

(c) Déplacement suivant Z−

FIGURE 13.24 – Principe de la mesure de position

Suivant le sens de déplacement du moteur, Z+ ou Z− , les capteurs A et B renvoient les informations a et b des figures 13.24b et 13.24c.

Un compteur C, est incrémenté ou décrémenté suivant le sens de déplacement du moteur à chaque changement d'état des variables a et b.

Le diagramme état-transition ci-dessous incomplet décrit le cycle de comptage.

Remarque : le point d'exclamation ! permet de noter le complément de la variable. Dans le cas d'une condition de garde alors $[!a]$ doit se comprendre comme la variable NONa. Dans le cas d'un événement, il s'agit de l'événement qui correspond au passage de l'état haut à l'état bas (front descendant).

Q1. Commenter le cycle décrit sur la figure 13.25.
Q2. Compléter le diagramme états-transitions en ajoutant le décomptage.
Q3. Justifier que le gain du capteur soit égal à $K_r = 10^6$ incréments/mètre.
Q4. Pourquoi dit-on que ce capteur de position est un capteur relatif ?

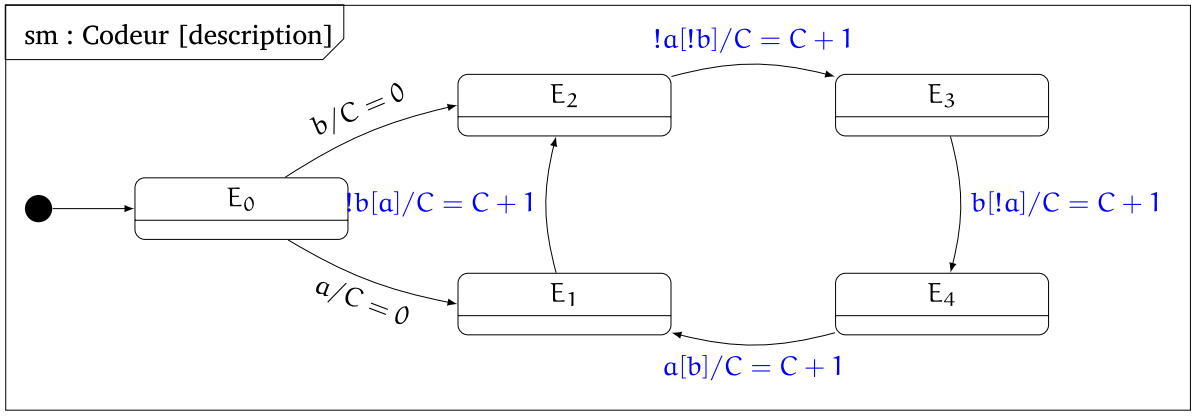


FIGURE 13.25 – Diagramme états-transitions

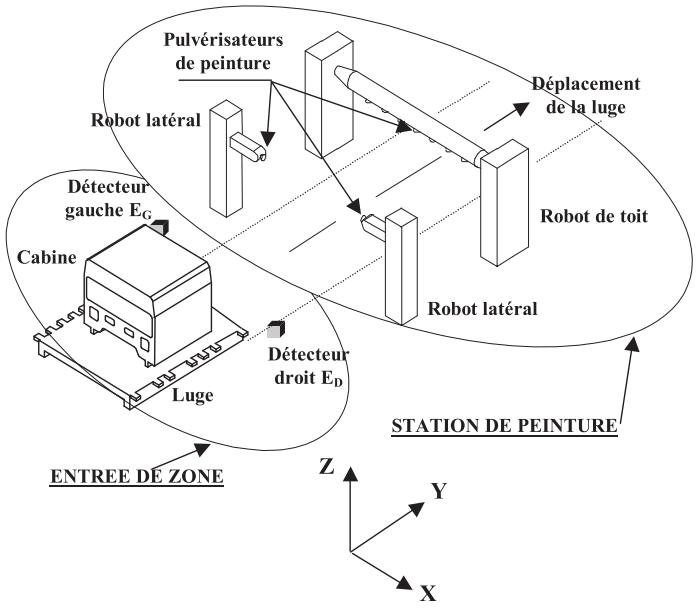
Exercice 5 - Robot de peinture pour cabine - Codage

Oral CCP *Corrigé page 33*

Présentation

La cabine de camion, après assemblage des différents éléments de tôlerie qui la composent, est envoyée dans l'atelier de recouvrement de surface où elle doit être protégée contre l'oxydation et peinte à la couleur choisie par le client.

La production « juste à temps » étant appliquée dans l'entreprise, chaque cabine fabriquée est déjà vendue à un client qui a choisi les options d'équipement et la couleur du camion. Pour informer chaque poste de travail des options choisies, la cabine est placée sur une luge identifiée par un code binaire. Sa lecture par deux détecteurs permet de renseigner chaque poste d'intervention sur les options choisies et donc de charger le programme adéquat.



La luge est entraînée par un système de chaîne selon l'axe Y (système non étudié). Lors de son passage sur les détecteurs situés en entrée de zone, le code binaire contenant les informations sur la cabine (type, teinte choisie), est lu et transmis au système de commande de la station de peinture.

Les automates assurant la gestion des robots de peinture prennent en compte ces informations et exécutent en fonction de l'avance de la cabine le programme permettant le bon positionnement des pulvérisateurs dans l'espace.

Les mouvements combinés des pulvérisateurs permettent le suivi du profil et la mise en peinture des côtés, de l'avant et de l'arrière de la cabine. Entre 2 cabines, les automates peuvent exécuter un changement de couleur en 15 secondes.

A. Codage des cabines

Afin de limiter les problèmes de lecture, plutot qu'un code barre ou une solution type carte à puce, le constructeur a choisi un codage physique à base de deux détecteurs inductifs.

Chaque luge métallique comporte 12 dents sur le côté gauche pour la synchronisation et peut comporter jusqu'à 12 dents sur le côté droit pour traduire l'information. Deux capteurs inductifs E_G et E_D détectent le passage des dents.

Les dents n°1 (droite et gauche) signalent le début de la luge et les dents n°12 la fin de la luge. Ces deux dents ne participent donc pas au codage du type de cabine.

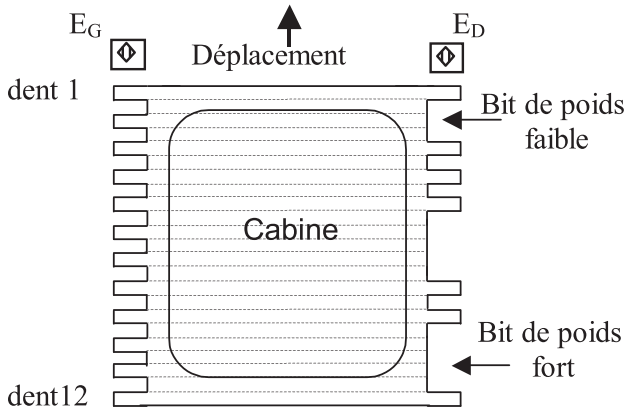
Le code déduit des autres dents permet d'identifier le type de cabine et de préciser la couleur de celle-ci. L'automate de gestion interprète l'information codée sur la luge au fur et à mesure de son déplacement.

— Le code lu sur la luge est mémorisée dans la variable M100. Ce code sera transmis aux différents postes de peinture.

— Le cycle de lecture du code débute dès que les deux dents n°1 sont à 1. La variable M100 est mise à zéro ($M100 = 0$).

— À chaque passage devant une dent du coté gauche, la variable est incrémentée de 2^{i-2} avec i le numéro de la dent si une dent est aussi présente du coté droit.

— Lorsque le nombre correct de dents a été lu, le code est contenu dans M100. L'évacuation de la luge est détectée sur la dernière dent.
— En cas d'erreur de lecture une alarme est activée afin qu'un opérateur intervienne et valide manuellement le code.



(a) Exemple de luge

E_G	E_D	
0	0	absence de luge ou intervalle entre deux dents
0	1	cas impossible, erreur de lecture
1	0	pas d'incrémentation de M100
1	1	incrémentation de M100

(b) Table de codage

FIGURE 13.26 – Luge et principe de lecture

B. Questions

Q1. Préciser le code de la luge représentée sur la figure 13.26.
Q2. Décrire par un diagramme états-transition (*state machine diagram*) le processus de lecture du code de la luge.

Exercice 6 - Sécurité « Homme-mort »

Corrigé page 33

L'Homme mort est une sécurité installée dans les systèmes où il est nécessaire de vérifier que l'opérateur est toujours vigilant. À l'origine, ce système a été installé dans les cabines de conduite de trains.

On se propose d'étudier l'installation de cette sécurité sur un camion oléo-serveur (figure 13.27) permettant d'alimenter en carburant les avions sur les aéroports.

Une poignée à distance permet de commander l'ouverture de la vanne de remplissage.

Compte tenu des dangers, l'opérateur doit maintenir et valider en permanence la commande, pour cela :

FIGURE 13.27 – Camion oléo-serveur

— L'opérateur doit actionner et maintenir la poignée (« Hom »).
— Au bout de 180 s, un voyant (V_{alerte}) clignote pour l'avertir de relâcher la poignée, il a alors 20 s pour la relâcher et la serrer de nouveau dans un délai de 2 s (cette action permet de vérifier que l'opérateur ne bloque pas la poignée !) sinon, la vanne se ferme.
— Le cycle de 180 s redémarre.
— À tout moment, dès qu'il desserre la poignée, il a un délai de 2 secondes pour la resserrer avant la fermeture de la vanne.
— Si la vanne se ferme, l'opérateur doit valider par un bouton (« acq ») dans un délai de 10 s avant de pouvoir reprendre le chargement, au-delà le cycle se termine.

Q1. Décrire par un diagramme d'état le fonctionnement de la sécurité « Homme mort ».