Systèmes à événements discrets

Table des matières

Introduction

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.

Les systèmes à événements discrets évoluent **lorsqu'un événement est présent**.

On peut définir alors un **modèle de fonctionnement logique** qui est l'ensemble des relations causales qui traduisent le fonctionnement de tels systèmes.

Définitions

1. Variables logiques

Définition

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

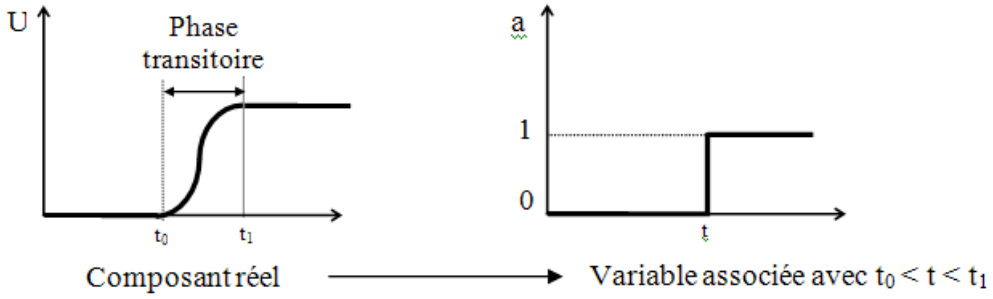
A chacun de ces composants, on peut associer une **variable** binaire ou **logique** qui ne peut prendre que deux valeurs notées **1** ou **0** (vrai ou faux, oui ou non).

La convention habituellement utilisée est d'associer l'état 1 de la variable à la situation "actionnée", "activée", "en travail" ou "en énergie" du composant.

L'état 0 est alors associé à la situation "relâchée", "désactivée", "au repos" ou "hors énergie" du composant.

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.



Simplification

Exemple

Les Automates Programmables Industriels (API) sont des appareils très utilisés dans la réalisation de commandes de systèmes automatisés. Par l'intermédiaire de modules d'entrées TOR, ils reçoivent des informations sous forme de signaux électriques. L'API travaille donc sur une variable binaire associée à chaque entrée. Il importe donc que le constructeur définisse parfaitement les paramètres reconnus par le module d'entrée comme étant caractéristiques des niveaux 0 et 1.

niveau 0 : U < 7 Volts et I < 3 mA

niveau 1 : 15 V < U < 30 V et 5 mA < I < 7 mA

Le constructeur de l'API ne garantit rien pour tout signal de valeur intermédiaire.

2. Système logique

Causalité

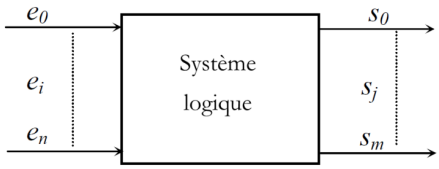
Parmi les variables logiques, on peut distinguer deux classes :

- les **entrées**, correspondant à des **états**, des **informations** (exprimés le plus souvent par des participes passés)
- les **sorties**, correspondant à des **actions** (exprimés le plus souvent par des verbes à l'infinitif)

Fonctions logiques

Soient les entrées e_i et les sorties s_j d'un système logique.

La fonction logique f telle que $s_j = f(e_i)$, indépendante du temps, exprime la **causalité**, c'est-à-dire les raisons pour lesquelles des actions sont exécutées en fonction des informations.



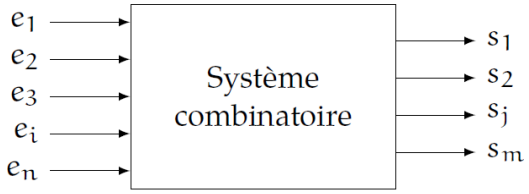
Entrées et Sorties

2.1. Système combinatoire

Définition

Un système logique est dit **combinatoire** si à **une** combinaison des variables d'entrée, ne correspond qu'**une seule** combinaison des variables de sortie.

La même combinaison des entrées produit toujours le même état des sorties.

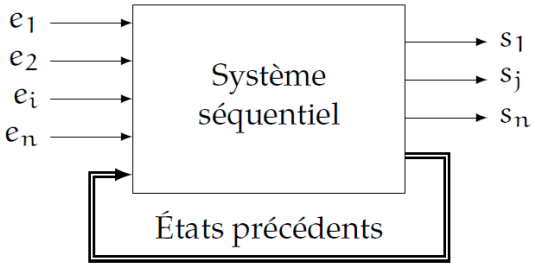


2.2. Système séquentiel

Définition

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

Une 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.



3. Équation logique

Une équation logique est une combinaison (réalisée à l'aide d'opérations logiques) de **plusieurs** variables logiques (d'entrée ou de sortie), donnant l'état d'**une variable de sortie** associée.

Par exemple : $S = a + b \cdot c$

4. Table de vérité

La table de vérité représente l'état d'**une variable de sortie** pour **chacune** des combinaisons des **variables d'entrée**.

Ainsi, s'il y a n variables d'entrées, la table comporte 2^n combinaisons différentes d'entrées.

Éclairage d'un couloir ? ***Exemple***

Un couloir est éclairé par une lampe L, actionnée par deux interrupteurs en "va-et-vient" d et g. On suppose qu'au départ les deux interrupteurs sont dans la même position, avec la lampe éteinte (état logique 0).

d	g	L
0	0	0
1	0	1
1	1	0
0	1	1

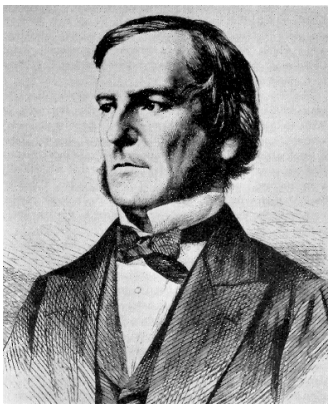
Algèbre de Boole

Pour traiter les systèmes logiques, on utilise l'algèbre de Boole, mise au point par le mathématicien anglais Georges Boole (1815-1864).

C'est l'ensemble $B = \{0, 1\}$ muni des lois "." (fonction ET), "+" (fonction OU) ainsi que la négation (notée par une barre horizontale tracée au-dessus du texte).

1. Georges Boole et Augustus De Morgan

Georges Boole (1815 - 1864) *Complément*



Georges Boole

Issu d'une famille pauvre, George Boole est autodidacte et devient enseignant à 16 ans.

Bénéficiant des moyens de l'Institut de mécanique de sa ville, il se confronte quelques années plus tard aux œuvres d'Isaac Newton, Pierre-Simon de Laplace et Joseph-Louis Lagrange.

En 1839, il publie ainsi sa première étude dans le Cambridge Mathematical Journal. Cette publication et l'appui qu'il obtient du cercle des algébristes de Cambridge lui permettent de s'imposer petit à petit comme une personnalité importante du monde des mathématiques. En 1844, après la publication d'un mémoire d'analyse dans les *Philosophical Transactions*, la *Royal Society* lui décerne une médaille.

C'est le début d'une série de travaux posant les bases de ce qu'on nommera plus tard l'algèbre de Boole. En 1847 est publié *Mathematical Analysis of Logic*, puis *An Investigation Into the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities* en 1854. Boole y développe une nouvelle forme de logique, à la fois symbolique et mathématique. Il crée une algèbre binaire n'acceptant que deux valeurs numériques : 0 et 1. Les travaux de Boole, s'ils sont théoriques, n'en trouveront pas moins des applications primordiales dans des domaines aussi divers que les systèmes informatiques, la théorie des probabilités, les circuits électriques et téléphoniques, etc.. grâce à des scientifiques comme Peirce, Frege, Bertrand Russell, Alan Turing et Claude Shannon.

En 1849, George Boole se voit proposer une chaire de professeur des mathématiques au *Queen's College* de Cork, en Irlande. Et en 1857, il est nommé membre de la *Royal Society*. Il s'intéresse ensuite aux équations différentielles à travers deux traités : *Treatise on Differential Equations* (1859) et *Treatise on the Calculus of Finite Differences* (1860).

https://youtu.be/Bl2RlVekfwM

Augustus De Morgan (1806 - 1871) *Complément*

Augustus De Morgan

Il entre au *Trinity College* de Cambridge en 1823, après une scolarité passée en Angleterre où ses talents pour les mathématiques sont décelés. Laïc convaincu, il se heurte régulièrement aux principes religieux alors associés à l'éducation. Il devient professeur de mathématiques à Londres, au sein de l'*University College* en création. En pédagogue doué, il est apprécié pour ses nouveautés apportées à l'enseignement (travaux dirigés succédant à des cours magistraux).

2. Propriétés et théorèmes

Fondamental

commutativité	$a+b=b+a$	$a\cdot b=b\cdot a$
distributivité	$a+(b\cdot c)=(a+b)\cdot(a+c)$	$a\cdot(b+c)=(a\cdot b)+(a\cdot c)$
élément neutre	$a+0=a$	$a\cdot 1=a$
complémentarité	$a+\bar{a}=1$	$a\cdot\bar{a}=0$
idempotence	$a+a=a$	$a\cdot a=a$
élément absorbant	$a+1=1$	$a\cdot 0=0$
absorption	$a+a\cdot b=a$	$a\cdot(a+b)=a$
associativité	$a+(b+c)=(a+b)+c=a+b+c$	$a\cdot(b\cdot c)=(a\cdot b)\cdot c=a\cdot b\cdot c$
Théorèmes de De Morgan	$\overline{a+b}=\bar{a}\cdot\bar{b}$	$\overline{a\cdot b}=\bar{a}+\bar{b}$

Quelques explications *Exemple*

Imaginons une lampe L allumée par deux boutons poussoirs a et b.

Complémentarité :

- 1ère colonne : pour que L soit allumée, il faut appuyer sur a OU ne pas appuyer sur a: la lampe est donc toujours allumée (état 1)
- 2ème colonne : pour que L soit allumée, il faut appuyer sur a ET ne pas appuyer sur a: la lampe est donc toujours éteinte (état 0)

Idempotence :

- 1ère colonne : pour que L soit allumée, il faut appuyer sur a OU appuyer sur a: il faut donc simplement appuyer sur a
- 2ème colonne : pour que L soit allumée, il faut appuyer sur a ET appuyer sur a: il faut donc simplement appuyer sur a

Absorption :

- 1ère colonne : pour que L soit allumée, il faut appuyer sur a OU appuyer sur a ET b en même temps : il suffit donc simplement d'appuyer sur a
- 2ème colonne : pour que L soit allumée, il faut appuyer sur a ET appuyer sur a OU b en même temps (les parenthèses sont importantes) : il suffit donc simplement d'appuyer sur a

Conséquence sur l'équation logique : b a été absorbé par a.

De Morgan :

- 1ère colonne : "il NE FAUT PAS appuyer sur a OU b" est équivalent à "il FAUT : ne pas appuyer sur a ET ne pas appuyer sur b en même temps"
- 2ème colonne : "il NE FAUT PAS appuyer sur a ET b en même temps" est équivalent à "il FAUT : ne pas appuyer sur a OU ne pas appuyer sur b

Remarque

Toute égalité logique possède son équivalent par dualité : **on peut remplacer les ET par des OU, et réciproquement**.

Par exemple : $a.(a+b)=a$ est équivalent $a+(a.b)=a$ (propriété de l'absorption)

3. Fonctions logiques fondamentales

Deux variables d'entrée donnent quatre combinaisons possibles d'état.

Il existe 2^4 tables de vérité différentes possibles pour une sortie dépendant de ces deux variables d'entrée : on définit donc 16 fonctions logiques possibles à deux entrées.

Deux d'entre elles sont triviales : la fonction constante égale à 0, et la fonction constante égale à 1.

Six autres correspondent aux fonctions suivantes :

3.1. Fonction OUI

$\forall a \in E, OUI(a) = a$

a	$OUI(a)$
0	0
1	1

Remarque

Parmi les 16 fonctions possibles, 2 correspondent à la fonction **OUI** : chacune pour l'une des deux entrées.

3.2. Fonction COMPLÉMENT, ou NON

$\forall a \in E, a = 1 \Leftrightarrow \bar{a} = 0$ c'est la fonction **NON** (inverse ou complémentation ou négation)

a	$\bar{a}$
0	1
1	0

Remarque

Parmi les 16 fonctions possibles, 2 correspondent à la fonction **NON**: chacune pour l'une des deux entrées.

3.3. Fonction OU

$\forall (a \times b) \in E \times E, (a + b) \Leftrightarrow$ l'un au moins des deux éléments a et b vaut 1, c'est la fonction **OU** (somme logique, loi de composition interne notée « + »)

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

3.4. Fonction ET

$\forall (a \times b) \in E \times E, (a \cdot b) \Leftrightarrow a$ et b valent simultanément 1, c'est la fonction **ET** (produit logique, loi de composition interne notée « . »)

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

4. Fonctions logiques composées

Les huit fonctions logiques restantes sont détaillées ici.

4.1. Fonction OU EXCLUSIF

$\forall (a \times b) \in E \times E, (a \oplus b) \Leftrightarrow$ un seul des deux éléments a ou b vaut 1, c'est la fonction **OU** exclusif.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Remarque

Sans utiliser le signe $\oplus$, elle peut s'écrire $a.\bar{b} + \bar{a}.b$.

4.2. Identité (ou NON OU EXCLUSIF)

C'est la **négation du OU EXCLUSIF** : la sortie est vraie si les deux entrées ont le même état, et fausse sinon.

a	b	$\overline{a \oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

4.3. Fonction NAND (NON ET)

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

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

C'est une **fonction universelle** : n'importe quelle fonction logique peut être écrite en utilisant uniquement des fonctions NAND.

Exemple

Écrire $\bar{a}, a + b, a.b$ à l'aide la fonction NAND :

- $a = a.a$, donc $\bar{a} = \overline{a.a}$
- $a + b = \bar{\bar{a}} + \bar{\bar{b}} = \overline{\bar{a} \cdot \bar{b}}$ (de Morgan) donc $= \overline{\overline{a \cdot a} \cdot \overline{b \cdot b}}$

- $a \cdot b = (a \cdot b) + (a \cdot b) = \overline{\overline{a \cdot b}} + \overline{\overline{a \cdot b}}$ donc $= \overline{\overline{a \cdot b} \cdot \overline{a \cdot b}}$ (de Morgan)

4.4. Fonction NOR (NON OU)

$a \; NOR \; b = \overline{a + b} = \bar{a} \cdot \bar{b}$

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

C'est une **fonction universelle** : n'importe quelle fonction logique peut être écrite en utilisant uniquement des fonctions NOR.

Exemple

Écrire $\bar{a}, a + b, a.b$ à l'aide la fonction NOR :

- $a = a + a$, donc $\bar{a} = \overline{a + a}$
- $a + b = (a + b) \cdot (a + b) = \overline{\overline{a + b}} \cdot \overline{\overline{a + b}} = \overline{\overline{a + b} + \overline{a + b}}$ (de Morgan)
- $a \cdot b = \bar{\bar{a}} \cdot \bar{\bar{b}} = \overline{\bar{a} + \bar{b}} = \overline{\overline{a + a} + \overline{b + b}}$

4.5. Inhibition

Si b est vrai, alors la sortie est vraie, sauf si a est vrai : b est inhibé par a.

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

Remarque

L'inhibition de a par b existe aussi, ce qui totalise donc 2 combinaisons parmi les 16.

Démarrage de moteur à arrêt prioritaire ***Exemple***

Soit un moteur dont l'alimentation est gérée par deux interrupteurs :

- la mise en marche b
- l'arrêt a

La table de vérité montre que le moteur va uniquement démarrer si l'on appuie sur b seul. Si, par mégarde, on appuie simultanément sur a et b, le moteur ne démarre pas.

4.6. Implication

La table de vérité de l'implication est complémentaire de celle de l'inhibition :

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

Remarque

L'implication par a existe aussi, ce qui totalise donc 2 combinaisons parmi les 16.

Interrupteur et ampoule électrique ***Exemple***

Soit a un interrupteur électrique (ouvert à l'état 0) et b une ampoule électrique qui lui est reliée (allumée à l'état 1).

La table de vérité liste les **états possibles** pour le circuit :

- interrupteur **ouvert**, lampe **éteinte** : état **possible**
- interrupteur **ouvert**, lampe **allumée** : état **absolument impossible**
- interrupteur **fermé**, lampe **éteinte** : état **possible** en cas d'ampoule grillée par exemple
- interrupteur **fermé**, lampe **allumée** : état **possible**

5. Tableaux de Karnaugh

Les tableaux de Karnaugh sont une forme particulière de table de vérité. En respectant certaines règles de présentation, ils permettent d'obtenir la forme la plus simple possible d'une fonction logique.

Chaque case du tableau correspond à une ligne de la table de vérité d'une fonction logique: une fonction à **n variables** est donc représentée par un tableau à **2^n cases** agencé de telle façon qu'**une seule variable change de valeur quand on passe d'une case à une case adjacente**.

Attention

La dernière ligne et la première ligne sont aussi adjacentes. La dernière et la première colonne sont aussi adjacentes.

Dans l'idéal il faudrait donc représenter le tableau de Karnaugh sur un **tore** !

Remarque

Si un tableau contient peu de 0, on peut regrouper les 0 plutôt que les 1 pour obtenir le complémentaire de la fonction logique.

Si certaines combinaisons d'entrées sont **absurdes** et ne peuvent pas absolument se réaliser ("pièce trop grande" ET "pièce trop petite"), on écrit une **croix** dans la case correspondante. Cette croix pourra être considérée comme valant 1 ou 0 suivant ce qui nous arrange dans les regroupements.

Regroupement dans les tableaux de Karnaugh *Méthode*

1. Reporter d'abord dans le tableau les valeurs de la fonction pour chacune des combinaisons des entrées
2. Faire ensuite des groupes de 2^i cases adjacentes (donc pas en diagonale !) valant 1 (cf. remarque précédente) :
   - on essaie de faire des groupes les plus "grands" possibles
   - on peut utiliser plusieurs fois si nécessaire une même case pour plusieurs groupes différents
   - cependant, si toutes les cases à regroupées font partie d'un groupe au moins, on "arrête"
3. Pour chaque groupe, on ne conserve pour l'équation logique que les variables qui ne changent pas d'état
4. On déduit l'équation de la sortie en **sommant** les différents "morceaux" d'équation logique obtenus précédemment.

5.1. n=2

$f_4 = a\bar{b}$

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

5.2. n=3

$f_5 = \bar{a}b\bar{c} + ab\bar{c} = b\bar{c}$

$c\ ab$	00	01	11	10
0	0	1	1	0
1	0	0	0	0

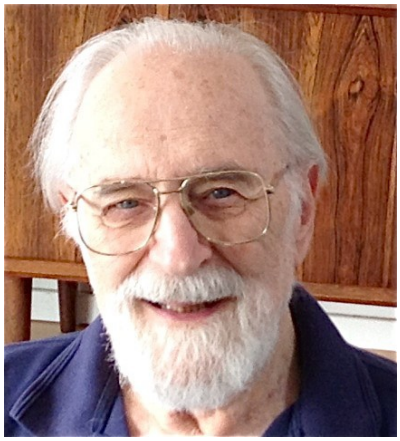
5.3. n=4

$f_6 = a\bar{b}cd + abcd + ab\bar{c}d + a\bar{b}\bar{c}d = ad$

$cd\ ab$	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

5.4. Maurice Karnaugh (1924 -)

Complément



Maurice Karnaugh

Ingénieur américain, docteur en physique (1952). Chercheur aux laboratoires Bell de 1952 à 1966, il y développe sa méthode de simplification d'équations logiques.

Il travaille ensuite dans le monde informatique au sein d'IBM (entre 1966 et 1993) et participe activement aux évolutions des télécommunications (membre de l'IEEE, association fixant notamment de nombreux standards en informatique).

Exercice : Simplifier les fonctions logiques par application des propriétés de l'algèbre de Boole

[solution n°1 p. 46]

Il faut simplifier les fonctions suivantes en n'utilisant ni la table de vérité ni les tableaux de Karnaugh :

$f_3 = c\,\bar{d}\,b + \bar{c}\,d\,\bar{a}\,b + b\,\bar{c} + a\,b\,c + \bar{a}\,b\,c\,d$

- ☐ $f_3 = b(\bar{c} + c\,a + \bar{d})$
- ☐ $f_3 = b(\bar{c} + c\,a + \bar{d})$
- ☐ $f_3 = b$

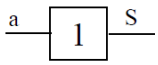
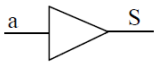
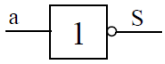
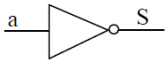
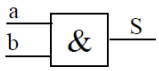
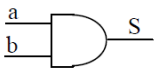
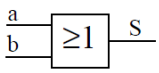
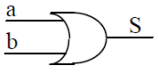
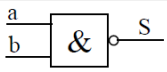
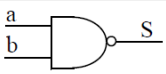
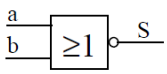
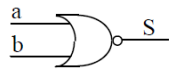
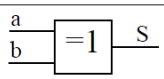
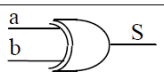
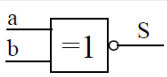
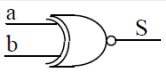
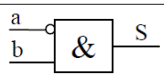
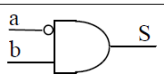
Représentations graphiques

Les modèles logiques de fonctionnement des systèmes peuvent être représentés de façon graphique.

1. Logigrammes

Chaque fonction logique peut être représentée de façon symbolique par une **cellule**.

Le câblage des cellules permet de réaliser des fonctions logiques complexes.

Opérateur	Fonction logique	Norme IEC (internationale)	Norme américaine
OUI	S = a		
NON	S = $\overline{a}$		
ET	S = a b		
OU	S = a + b		
NAND	S = $\overline{a\,b}$ = $\overline{a}$+$\overline{b}$		
NOR	S = $\overline{a+b}$ = $\overline{a}$ $\overline{b}$		
XOR	S = a ⊕ b		
IDENTITE	S = $\overline{a \oplus b}$		
INHIBITION	S = $\overline{a}$ b		

Représentation symbolique des fonctions logiques

Principe de lecture

Attention

- **entrée(s)** : hormis les fonction OUI et NON à une entrée, les cellules peuvent avoir deux ou davantage d'entrées.
- **sortie** : il n'y a toujours qu'une sortie par cellule.

Remarque

Les cellules peuvent être obtenues suivant **plusieurs technologies** : électronique (circuits intégrés) mais aussi pneumatique ou hydraulique.

La représentation sous forme de cellules peut être également utilisée en informatique.

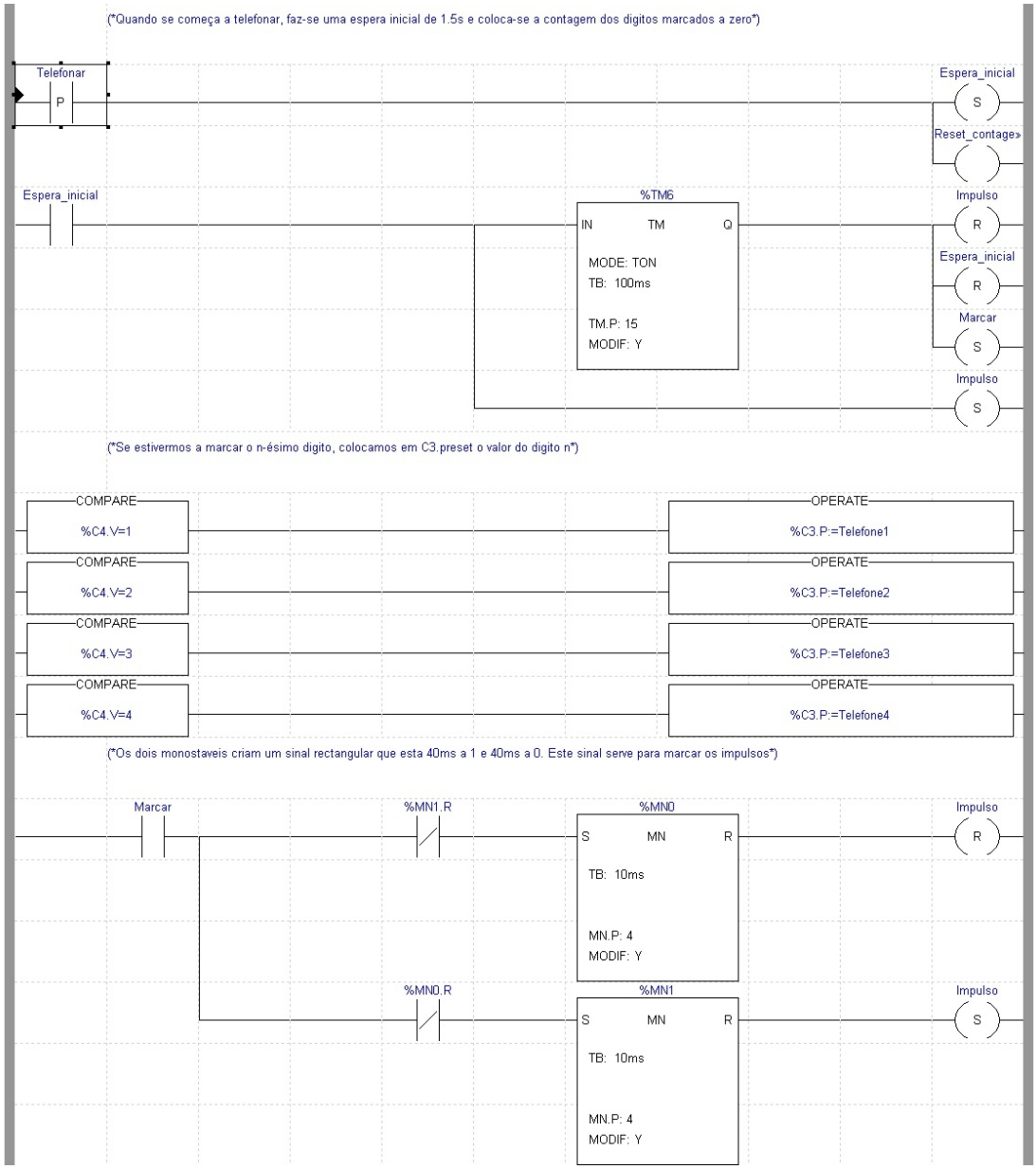
Le **temps de propagation** des informations, et les **durées d'élaboration** des sorties par les cellules sont considérés comme **nuls**.

2. Schématisation électrique

Le **schéma à contacts** (ou "schéma en échelle", *ladder*, etc) est une norme internationale de représentation symbolique de fonctions logiques.

Il est basé sur le principe suivant :

- une alimentation électrique en tension continue, représentée par deux barres verticales,
- un certain nombre de lignes horizontales reliant les barres verticales au moyen de **contacts** et de récepteurs électriques.

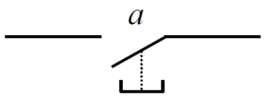


Exemple de programme en "ladder"

2.1. Contacts

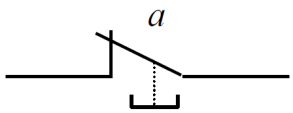
Les variables logiques peuvent être représentées par des contacts électriques :

contact normalement ouvert pour une variable "nue"



Contact normalement ouvert

contact normalement fermé pour une variable complémentée

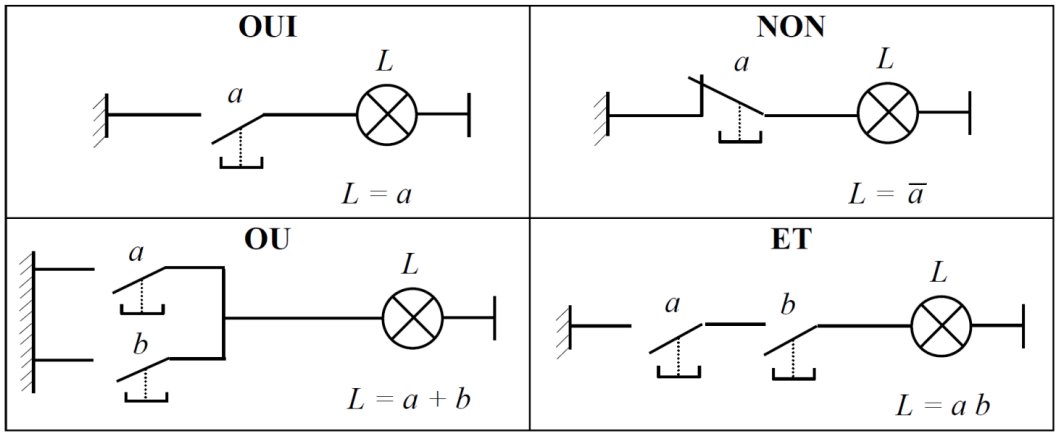


Contact normalement fermé

Remarque

Le nom d'un contact ne tient pas compte de la manière dont il est câblé : dans les deux exemples précédents, le contact s'appelle **a**.

2.2. Représentation électrique des fonctions élémentaires



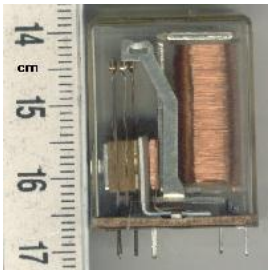
Représentation électrique des fonctions élémentaires

2.3. Relais

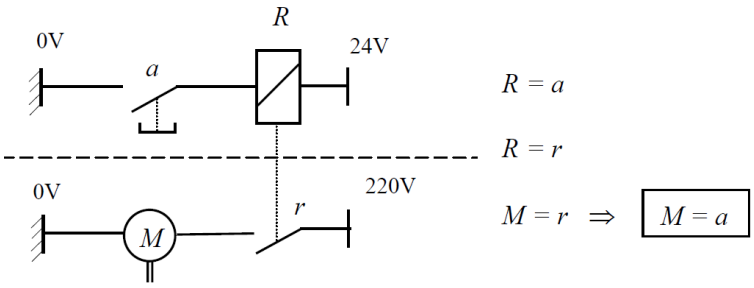
Un relais (ou contacteur à relais) est constitué :

- d'un circuit de **commande** comportant une **bobine** d'excitation électromagnétique R
- d'un circuit d'**utilisation** comportant un **contact** r, actionné par la bobine

Les deux circuits sont ainsi dissociés (pour des raisons de sécurité, par exemple).



Exemple



Exemple d'utilisation d'un relais

Conseil

Le principe est simple : une bobine "R" agit par "effet électromagnétique" sur un ou plusieurs contacts "r". Vous placez n'importe où sur votre schéma un ou plusieurs contacts "r" ; ceux-ci seront actionnés (qu'ils soient normalement ouverts ou normalement fermés) lorsque la bobine "R" sera active (c'est-à-dire parcourue par un courant électrique). Dans la réalité ces contacts sont proches de la bobine ; sur votre schéma ils peuvent être placés n'importe où. Simplement leur désignation doit correspondre à celle de la bobine pour qu'on puisse comprendre qu'ils lui sont reliés.

3. Schématisation pneumatique

De nombreuses applications industrielles utilisent l'énergie pneumatique pour transmettre une information ; dans ce cas

- une pression voisine de celle de la source (généralement de 2 à 10 bars) correspond à une information logique de 1
- une pression voisine de celle de la pression atmosphérique correspond à 0.

Une source pneumatique est représentée comme un triangle vide "entrant" (triangle plein en cas d'énergie hydraulique), et un échappement à l'air libre comme un triangle "sortant".

3.1. Vérins

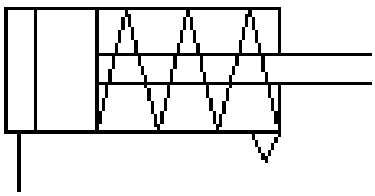
Les vérins sont des **actionneurs** : ils transforment l'énergie pneumatique en énergie mécanique.

Le mouvement est généralement celui d'une translation, mais il existe des vérins rotatifs (cf. tête de vissage de la capsuleuse de bocaux).

a) Vérins simple effet

Ils sont conçus pour ne travailler que dans un seul sens : à la sortie **ou** à la rentrée de tige.

Le mouvement qui s'effectue "à vide" est réalisé par un ressort interne.

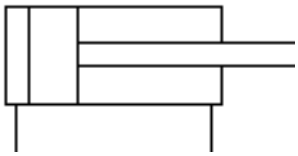


Commande des vérins simple effet — **Fondamental**

Les vérins simple effet **n'ont besoin que d'un seul ordre** pour effectuer leur travail ; l'**absence d'ordre** permet le retour en position de repos.

b) Vérins double effet

Les deux sens peuvent être utilisés pour réaliser un travail.



Commande des vérins double effet *Fondamental*

Deux ordres doivent donc être distingués : l'un à la rentrée de tige, l'autre à la sortie de tige.

En l'absence d'ordre donné, le vérin reste donc dans sa dernière position atteinte.

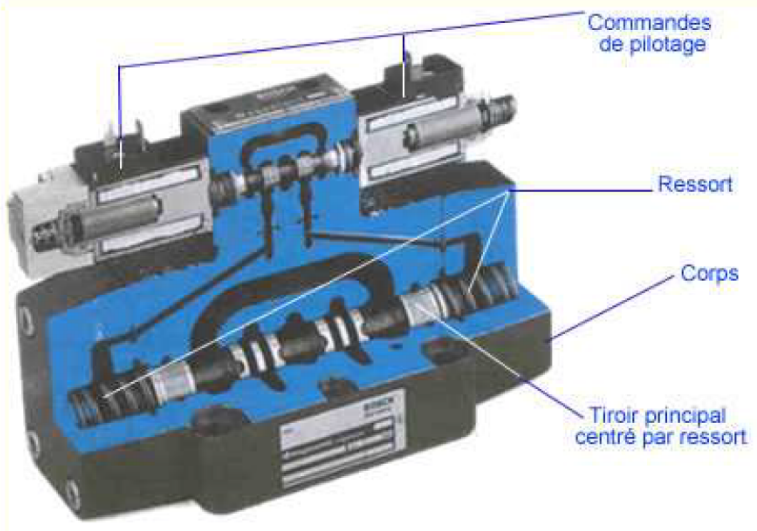
3.2. Distributeurs pneumatiques

Les distributeurs sont des **préactionneurs** : ils **modulent** ou **distribuent** l'énergie apportée aux actionneurs.

Solution technologique concrète

Les distributeurs pneumatiques (et hydrauliques) comportent plusieurs parties principales :

- le **corps**, qui est creux et percé de multiples orifices
- le **tiroir**, qui coulisse dans un sens ou dans un autre à l'intérieur du corps
- les organes de **commande** (poignées, bobines électriques, ressorts, etc...)



Nombre de positions

Le distributeur (via les positions de son tiroir) peut être dans **deux** ou **trois** situations différentes : pour chaque situation, les conduites sont associées selon une combinaison particulière.

Nombre de positions stables

Suivant le nombre de positions stables de son tiroir (c'est-à-dire des positions fixes **lorsqu'aucun ordre spécifique n'est donné** au distributeur), un distributeur peut être :

- **monostable** : un seul ordre est donc nécessaire pour le faire sortir de sa position de repos
- **bistable** : un ordre est nécessaire pour chaque position désirée (donc deux ordres au total)

Nombre de raccords

Pour chaque position, les conduites sont associées via des orifices réalisés dans le corps du distributeur.

Le nombre de connexions nécessaires pour chaque position est aussi appelé nombre d'orifices (il y en a au moins deux).

Attention

L'organe de commande, lorsqu'il est actif, **POUSSE** le tiroir.

Désignation normalisée

Fondamental

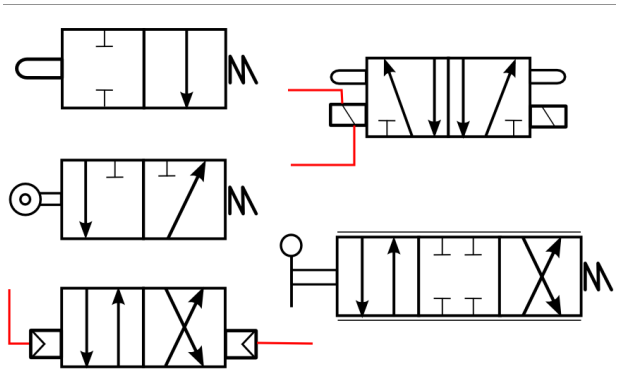
Un distributeur est désigné de la manière suivante :

- premier chiffre : nombre d'orifices
- barre oblique
- second chiffre : nombre de positions
- compléments (monostable ou bistable, à commande électrique, pneumatique, etc...)

Représentation normalisée

Fondamental

Un distributeur est toujours symbolisé **au repos**.



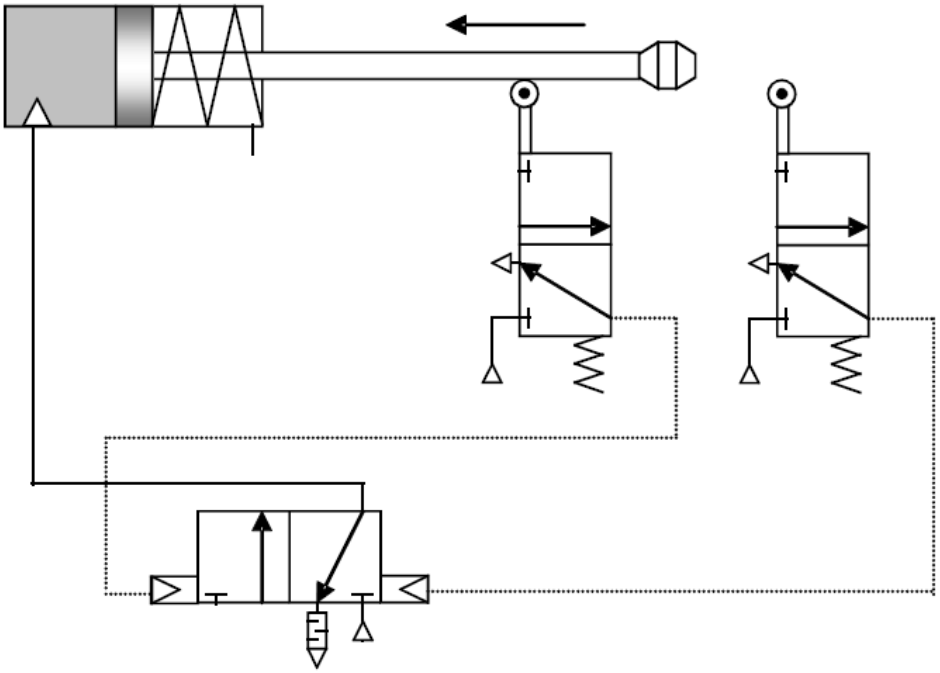
De haut en bas, colonne de gauche puis colonne de droite :

- 2 / 2 monostable à commande manuelle (bouton poussoir)
- 3 / 2 monostable à commande mécanique (capteur à galet)
- 4 / 2 bistable à commandes pneumatiques
- 5 / 2 bistable à commandes électriques et manuelles
- 4 / 3 monostable à commande manuelle par joystick, avec déplacement progressif (typique de la commande d'engins hydrauliques de chantier)

3.3. Schéma pneumatique

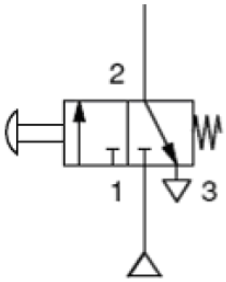
Exemple

Dans le système représenté ci-après, la tige du vérin effectue un aller-retour de manière continue.



Fonction OUI

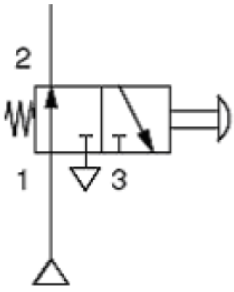
Exemple



Il faut activer la commande pour que la sortie soit vraie.

Fonction NON

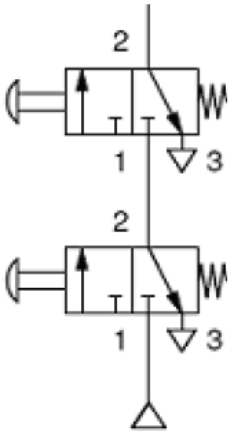
Exemple



Il faut activer la commande pour que la sortie soit fausse.

Fonction ET

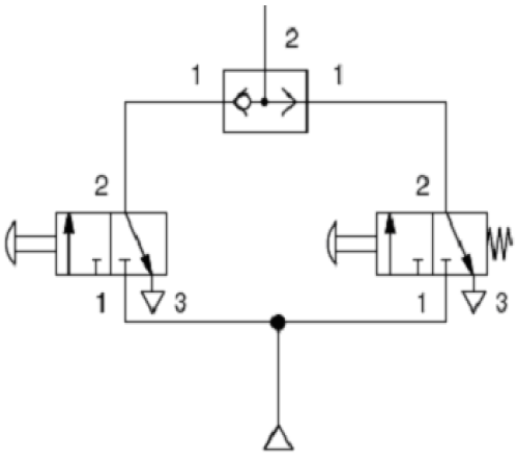
Exemple



Il faut activer les deux commandes pour que la sortie soit vraie.

Fonction OU

Exemple



Il faut activer l'une ou l'autre commande pour que la sortie soit vraie.

Le composant symbolisé en partie haute est une soupape à alternance : l'air sous pression peut être appliqué au niveau de chaque entrée 1, mais c'est la pression la plus haute qui arrive à la sortie 2.

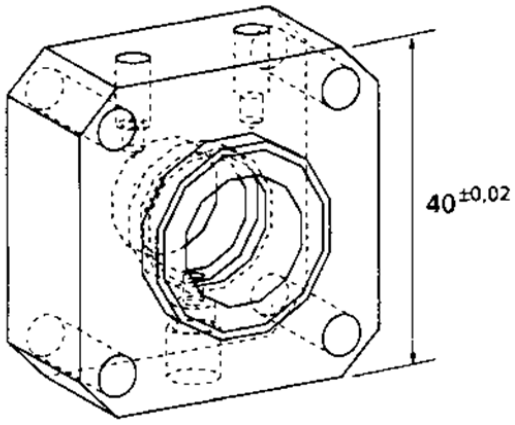
Exercices systèmes combinatoires

1. Exercice : Contrôle de pièces

Contexte industriel

Lors de la fabrication d'une pièce, on est amené à vérifier si la dimension fabriquée est bien égale à la dimension demandée. Mais il est **impossible** d'obtenir à la fabrication, une dimension **absolument égale** à celle demandée à cause des jeux fonctionnels des machines, des usures des outils, etc... On définit donc un **intervalle de tolérance** afin que la pièce fabriquée convienne au mécanisme dans lequel elle doit se monter.

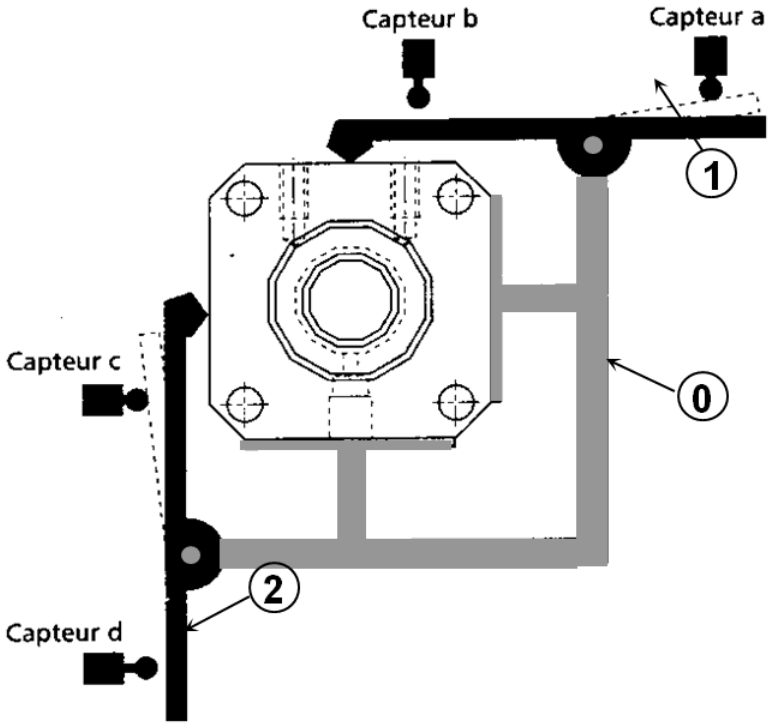
Par exemple, sur l'embout de vérin ci-contre, la cote $40^{\pm 0,02}$ signifie que si la dimension obtenue est comprise entre $39,98$ et $40,02$ alors la pièce est bonne. Sinon, elle est mauvaise.



Montage de contrôle

On désire vérifier les cotes extérieures horizontale et verticale de l'embout de vérin sur le montage de contrôle représenté ci-après. Ce mécanisme est composé de deux leviers **1** et **2** en liaison pivot par rapport au bâti **0**. Ces leviers sont munis chacun de deux capteurs à contact normalement ouverts.

Exemple: si le capteur **a** est activé, cela signifie que la cote verticale est trop faible; si le capteur **c** est activé, la cote horizontale est trop forte.



Le système est également muni de trois voyants bleu, vert et rouge non représentés :

- lorsque les deux cotes sont à l'intérieur des intervalles de tolérance, aucun des capteurs n'est actionné: la pièce est bonne et le voyant vert V s'allume,
- lorsque l'une des deux cotes est trop faible, la pièce est mise au rebut et le voyant rouge R s'allume,
- dans les autres cas, la pièce doit être réusinée et le voyant bleu B s'allume.

Question 1

[solution n°2 p. 46]

Représenter les tableaux de Karnaugh des voyants V, B et R et déterminer leur équation logique simplifiée en fonction de l'état des capteurs a, b, c et d.

Câblage électrique

Chaque contact normalement ouvert a, b, c et d commande un relais Ra, Rb, Rc et Rd alimentés en 5 Volts. Les voyants V, B et R sont alimentés en 12 Volts.

Question 2

[solution n°3 p. 47]

Représenter le schéma à contacts de l'alimentation des trois voyants.

2. Exercice : Perceuse automatisée

La mise en fonctionnement d'une perceuse automatisée est possible, **en mode normal,** si la pièce est bien positionnée, l'écran de protection fermé et le capteur de présence pièce activé. **En mode de réglage**, cette machine fonctionne avec ou sans écran de protection mais la pièce doit toujours être bien positionnée et le capteur de présence pièce activé. Le moteur M est donc commandé par les quatre entrées suivantes :

- p tel que p = 1 si la pièce est bien positionnée,
- e tel que e = 1 si l'écran de protection est fermé,
- c tel que c = 1 si le capteur de présence pièce est activé,
- r tel que r = 1 en mode de réglage.

Question 1

[solution n°4 p. 47]

Donner l'équation logique simplifiée du moteur M sous forme canonique (somme logique de produits logiques).

Question 2

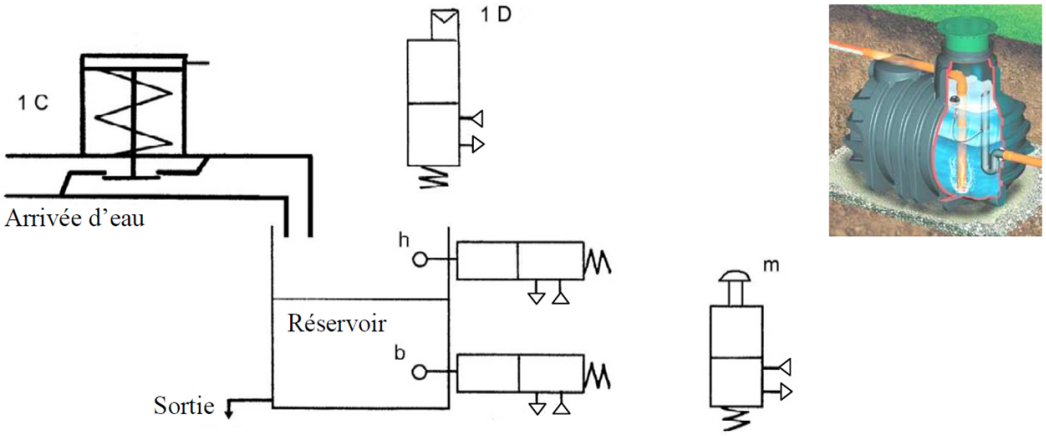
[solution n°5 p. 48]

Établir le logigramme de M avec le minimum de fonctions logiques de base ET, OU et NON à 2 entrées maximum.

3. Exercice : Partie commande d'un système de remplissage automatique de réservoir

Principe

On s'intéresse à un système de remplissage pneumatique automatique de réservoir.



Extrait du cahier des charges pour la vanne de remplissage 1C

La vanne d'alimentation en eau **1C** doit s'ouvrir lorsque le niveau bas est détecté, ou par action sur le bouton **m**. La vanne **1C** doit se fermer lorsque le niveau haut est atteint. La vanne 1C ne peut en aucun cas s'ouvrir si le niveau haut est atteint.

Le système doit être composé d'un circuit de commande à 3 bars et d'un circuit de puissance de 6 bars, ainsi que des composants suivants :

- Un vérin simple effet (**1C**, matérialisant la vanne)
- Un distributeur 3/2 monostable **1D** à commande pneumatique
- Un capteur 3/2 monostable **b** pour déceler le niveau bas (capteur actionné = interface eau/air détectée en face du capteur)
- Un capteur 3/2 monostable **h** pour déceler le niveau haut (capteur actionné = interface eau/air détectée en face du capteur)
- Un bouton poussoir 3/2 monostable **m** normalement ouvert pour le remplissage manuel.

Question 1

[solution n°6 p. 48]

Donner le tableau de Karnaugh permettant de décrire les conditions d'activation du distributeur de commande **1D** (ce qui ensuite activera le vérin **1C**).

Question 2

[solution n°7 p. 48]

En déduire l'équation logique simplifiée.

Question 3

[solution n°8 p. 49]

Tracer le logigramme permettant de décrire le fonctionnement du système et compléter le schéma pneumatique, avec :

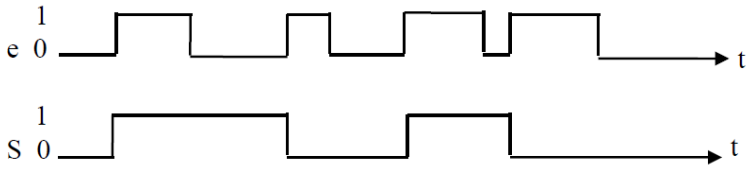
- en bleu pour le circuit de commande (3 bars)
- en rouge pour le circuit de puissance (6 bars)

Systèmes séquentiels

Considérons un système qui comporte une entrée e et une sortie S. Son fonctionnement peut être décrit de la façon suivante :

- S change de valeur chaque fois que l'entrée e passe de 0 à 1
- S garde sa valeur dans tous les autres cas.

Ce système peut être décrit par le **diagramme temporel** (ou « *chronogramme* ») suivant :



Ce système **n'est pas combinatoire**, car à un état de la variable d'entrée (1 par exemple) correspondent deux états possibles de la sortie (0 ou 1).

Le comportement du système ne dépend pas seulement des entrées à l'instant considéré mais de l'évolution antérieure du système.

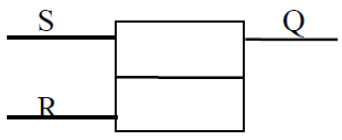
Il faut donc pouvoir caractériser le passé du système avec un certain nombre de variables qui devront être mémorisées à l'intérieur de ce système.

1. Fonction mémoire

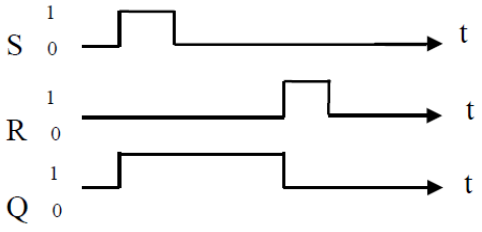
Fondamental

C'est le composant élémentaire de la logique séquentielle. C'est un circuit qui a pour propriété de conserver la valeur binaire précédemment inscrite en l'absence de toute sollicitation extérieure. Il comporte :

- deux entrées **R** (comme *Reset* ou "remise à 0" de la sortie) et **S** (comme *Set* ou "mise à 1" de la sortie)
- une sortie **Q**



Symbole de la fonction mémoire



Chronogramme de la fonction mémoire

Remarque

Le changement d'état des sorties est uniquement activé lorsque l'une des entrées R ou S passe de 0 à 1. Leur retour à 0 est sans effet.

Tentative de table de vérité

S	R	Q^+
0	0	Q^-
0	1	0
1	0	1
1	1	**

** : cet état dépend du fonctionnement de la fonction mémoire :

- **0** fonction à arrêt (ou "déclenchement") prioritaire
- **1** fonction à marche (ou "enclenchement") prioritaire

Q^- et Q^+ sont les états de la sortie Q respectivement **avant** et **après** modification des entrées.

2. Réalisations et schématisations de la fonction mémoire

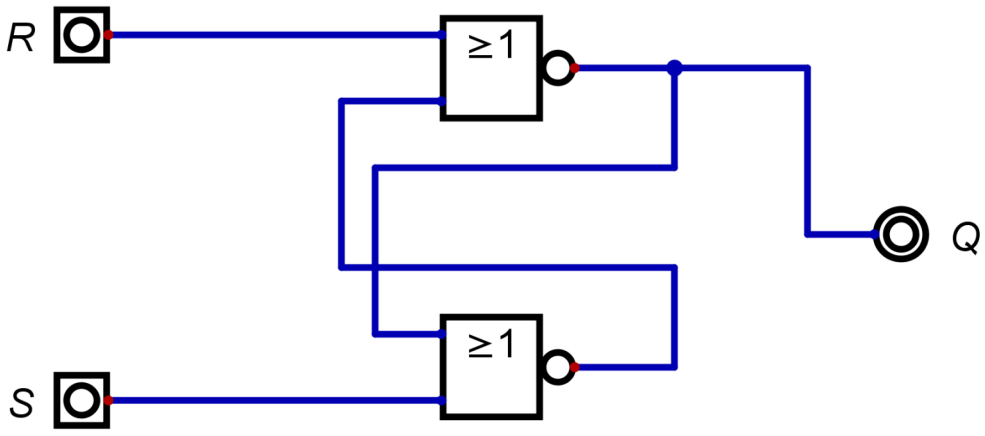
Équation logique

Selon la table de vérité précédente :

- arrêt prioritaire : $Q^+ = \bar{S}.\bar{R}.Q^- + S.\bar{R} = \bar{R}(S + Q^-)$
- marche prioritaire : $Q^+ = \bar{S}.\bar{R}.Q^- + S.\bar{R} + S.R = \bar{R}Q^- + S$

2.1. Logigramme

Mémoire à arrêt prioritaire

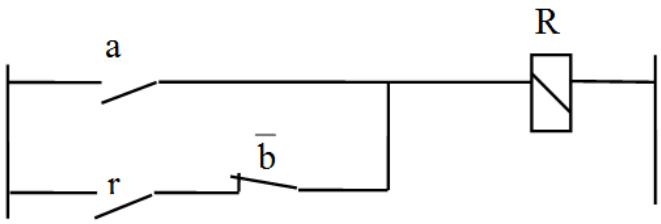


$\overline{R + \left(\overline{S + Q^-}\right)} = \overline{R}.(S + Q^-)$ (cf. De Morgan)

2.2. Schéma électrique

Avec un schéma utilisant le relais, on peut câbler une mémoire ; par exemple ici à marche prioritaire :

- a correspond à "marche" et b correspond à "arrêt"
- r est le contact fermé lorsque la bobine R est alimentée



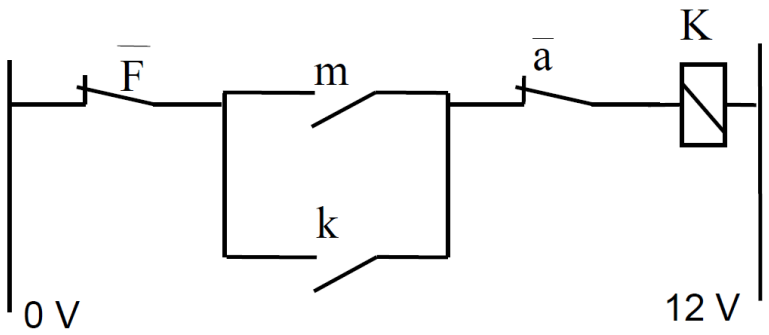
Commande manuelle de moteur triphasé *Exemple*

Un moteur triphasé est commandé à l'aide de deux boutons poussoirs :

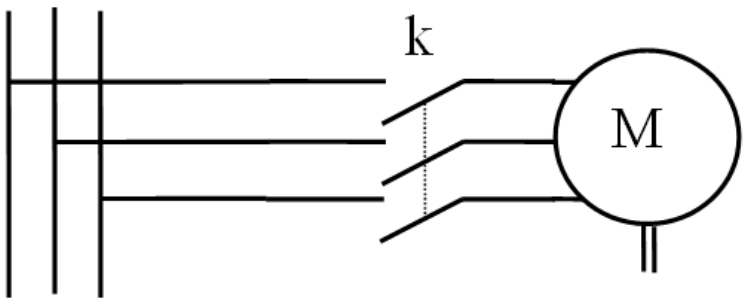
- m : mise en marche
- a : arrêt

En cas de commande simultanée, le moteur doit s'arrêter pour des raisons de sécurité.

De plus, une protection en cas de surintensité est assurée par le relais thermique F. Ce contact, normalement fermé, s'ouvre lorsque l'intensité traversant le circuit est trop forte. L'alimentation de la bobine K est coupée, et le moteur s'arrête.



Circuit de commande

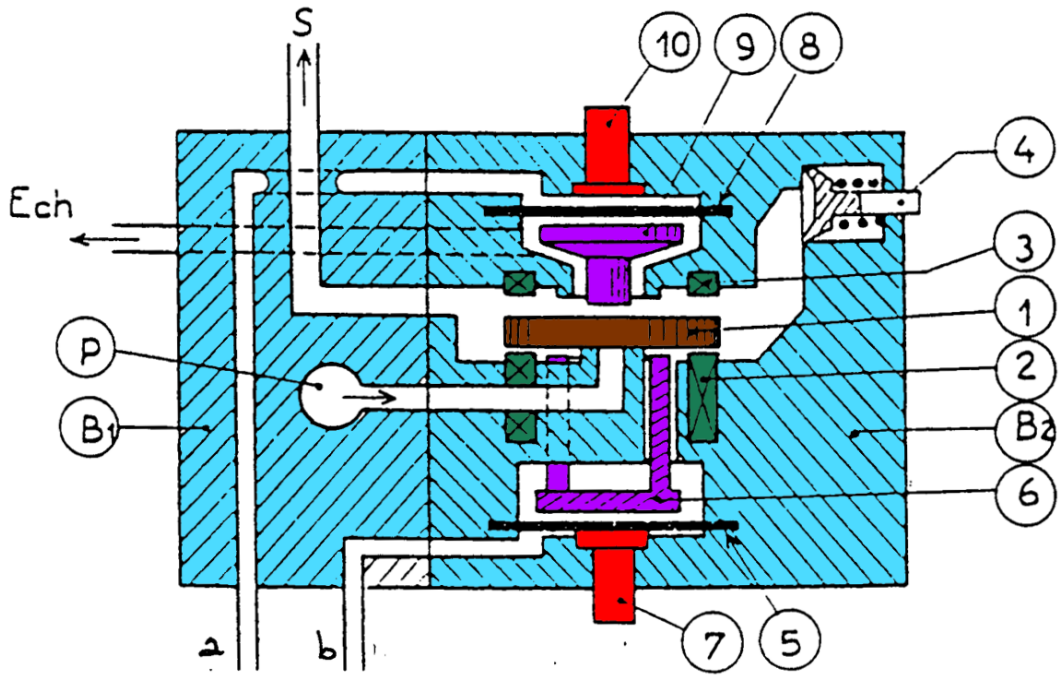


Circuit de puissance

La mise en marche du moteur se fait par action sur m, ce qui alimente la bobine K et ferme les contacts k correspondants. Une action sur a ou de F arrête le moteur : on est ici dans une configuration à arrêt prioritaire.

2.3. Réalisation pneumatique

Vue en coupe



- **1** : clapet
- **2** et **3** : aimants
- **4** : voyant
- **5** et **8** : membranes souples
- **6** et **9** : poussoirs
- **7** et **10** : boutons de commande manuelle
- **P** : alimentation en air comprimé

Fonctionnement

a	b	événements internes	S
0	0	**1** reste dans la position du schéma, maintenu par **2**	0
0	1	**5** se déforme, pousse **6** qui soulève **1** pour l'amener en position haute, maintenu par **3**	1
0	0	**1** reste maintenu en position haute par **3**	1
1	0	**8** se déforme, pousse **9** qui décroche **1** pour le ramener en position basse	0
1	1	l'aimant **2** est prioritaire	0

Remarque

Les fonctions de commande réalisées en technologie pneumatique **se raréfient** en raison du faible coût des technologies programmées.

De façon générale, les composants pneumatiques de commande ne sont justifiés que lorsqu'une raison particulière les impose:

- extension d'une commande pneumatique existante
- pas d'énergie électrique disponible
- **ambiance explosive**...

3. Utilisation du langage SysML

La description du comportement d'un système séquentiel, trop complexe avec une table de vérité, peut se faire par l'utilisation des diagrammes comportementaux SysML.

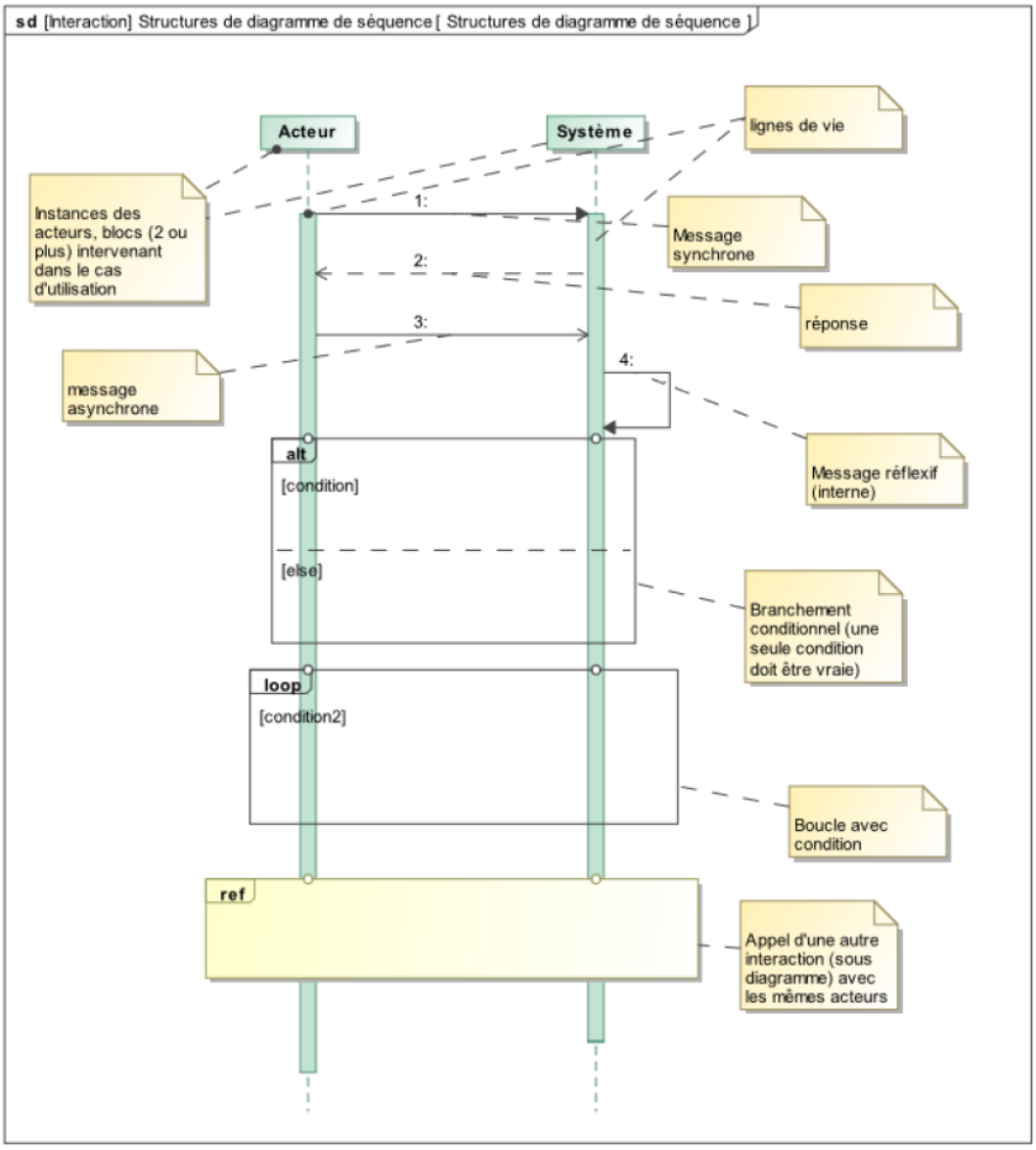
3.1. Séquences d'un système

Pour chaque cas d'utilisation d'un système, on peut décrire plus précisément quelles sont les **étapes successives** (la "séquence") qui permettent de rendre le service prévu.

C'est donc un enchaînement d'échanges, des **interactions** dont la nature est précisée, entre les acteurs (utilisateurs) et le système.

Diagramme de séquence en SysML *Fondamental*

(*Sequence Diagram*).



§ *Syntaxe*

Les "lignes de vie" représentent un élément du diagramme de cas d'utilisation.

Plusieurs types de messages peuvent être échangés :

- **synchrone** : l'émetteur attend une réponse (flèche retour en pointillé)
- **asynchrone** : pas de réponse est attendue
- **réflexif** : représente une activité au sein de l'émetteur

? *Exemple*

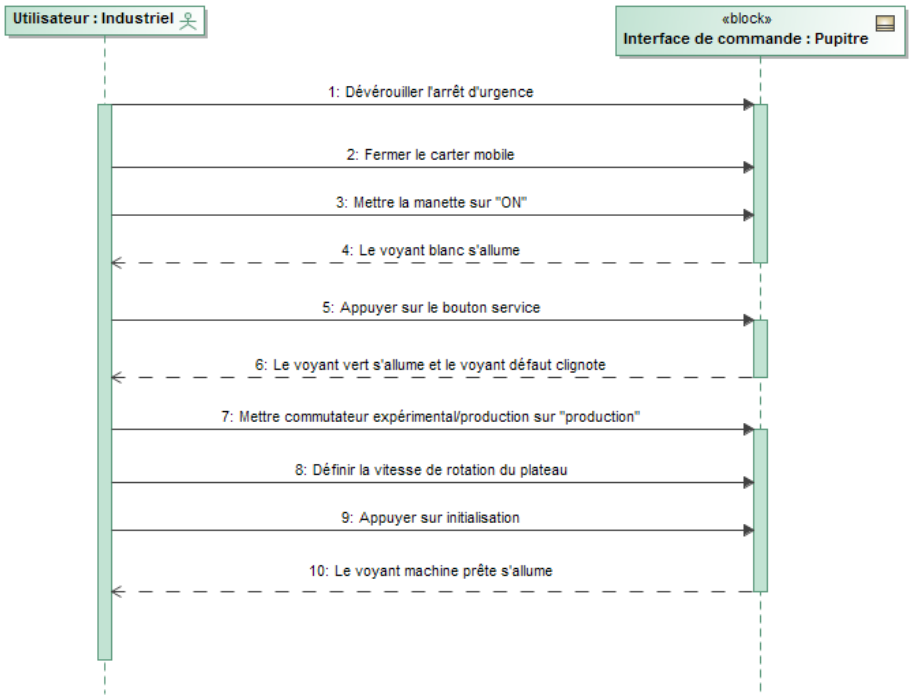


Diagramme de séquence de l'initialisation de la capsuleuse

3.2. États d'un système

a) Généralités

Les différents états possibles d'un système - dans son ensemble - peuvent être représentés sous forme d' "automate fini" (machine à états).

- Un état représente une **activité**, ou une **attente** d'un événement particulier.
- Des **transitions** permettent de passer d'un état à un autre.

Diagramme d'états en SysML *Fondamental*

(*State Machine Diagram*)

Ce diagramme est rattaché à un bloc des diagrammes de blocs.

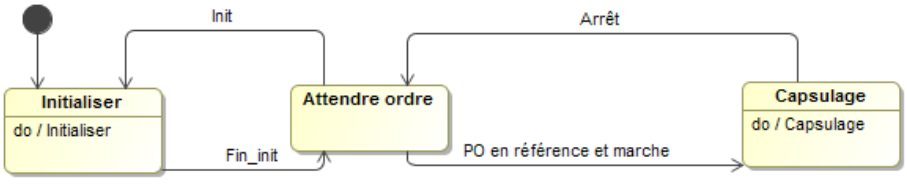


Diagramme d'états de la capsuleuse de bocaux

b) Détail

i) État

Les éléments graphiques utilisés dans ce diagramme sont principalement des rectangles aux coins arrondis pour représenter les états.

L'état initial est représenté par un rond plein, et les états finaux par des ronds "creux".

Remarque

Il peut y avoir plusieurs états finaux car plusieurs scénarios peuvent être possibles pour mettre fin à un comportement.

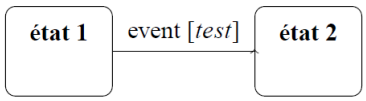
ii) Transition

Une **transition** peut être associée à un **événement**, une **condition de garde** et / ou à un **effet** (action).

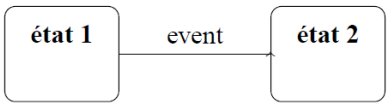
Elle s'écrit : "événement [condition de garde] / effet".

Quelques exemples :

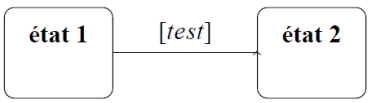
- A l'occurrence de **event**, **test** est évalué et la transition est franchie uniquement si **test** est vrai. L'éventuelle activité est interrompue. Si **test** n'est pas vrai, **event** est perdu et il faut attendre une seconde occurrence de **event** pour éventuellement franchir la transition si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



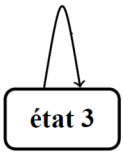
- A l'occurrence de **event**, la transition est franchie sans condition. L'éventuelle activité est interrompue.



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



- Transition de complétion : est immédiatement franchie dès la fin de l'éventuelle activité. Équivaut à [1].



Une transition réflexive entraîne une sortie d'état puis un retour dans ce même état.

Cela n'est donc pas sans conséquences selon les cas.

Événement

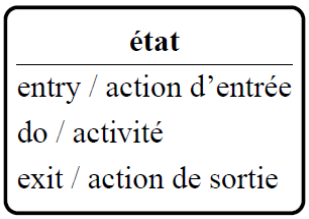
Il existe **quatre** types d'événements associés à une transition :

1. le **message** (*signal event*) : un message asynchrone est arrivé
2. l'**événement temporel** (*time event*) : un intervalle de temps s'est écoulé depuis l'entrée dans un état (mot clé *after*) ou un temps absolu a été atteint (mot clé *at*)
3. l'**événement de changement** (*change event*) : une valeur a changé de telle sorte que la transition est franchie (mot clé *when*)
4. l'**événement d'appel** (*call event*) : une requête de fonction du bloc a été effectuée et un retour est attendu ; des arguments (paramètres) de fonction peuvent être nécessaires.

Condition de garde

La **condition de garde** est une expression booléenne faisant intervenir des entrées et/ou des variables internes. Elle autorise le passage d'un état à un autre.

iii) Activité, action



A un état, on peut ainsi principalement rattacher une activité, une action d'entrée et une action de sortie.

Une **activité** peut être considérée comme une unité de comportement. Elle prend du temps et peut être interrompue. On la trouve à l'intérieur des nœuds du diagramme (mot clé *do*).

En revanche, une **action** ne prend pas de temps et ne peut pas être interrompue. Son exécution peut par exemple provoquer un changement d'état, l'émission d'un ordre pour un préactionneur ou un retour de valeur. On peut les trouver dans les transitions (**effet**) ou dans les états (mots clé *entry* ou *exit*). Les actions sont les éléments de base permettant de spécifier les activités dans le diagramme d'activités.

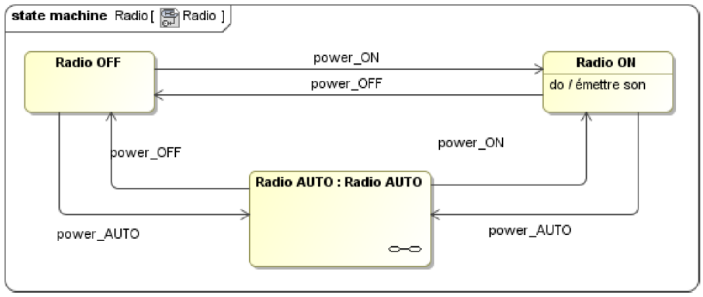
iv) État composite (super-état)

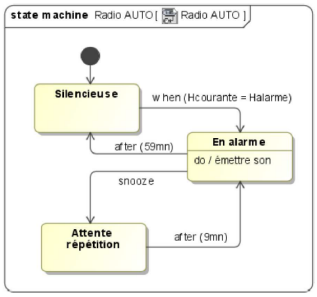
Un **état composite** est constitué de sous-états liés par des transitions. Cela permet d'introduire la notion d'état de niveau hiérarchique inférieur et supérieur.

Exemple

Dans l'exemple qui suit, l'état "Radio Auto" est composite :

- il apparaît de façon monobloc dans le diagramme de gauche
- le diagramme d'états de droite détaille son contenu.





3.3. Activités d'un système

a) Généralités

L'activité des **différents composants** d'un système peut être également être représentée sous forme d' "automate fini" (machine à états).

Des transitions permettent de passer d'un état à un autre.

Diagramme d'activités en SysML *Fondamental*

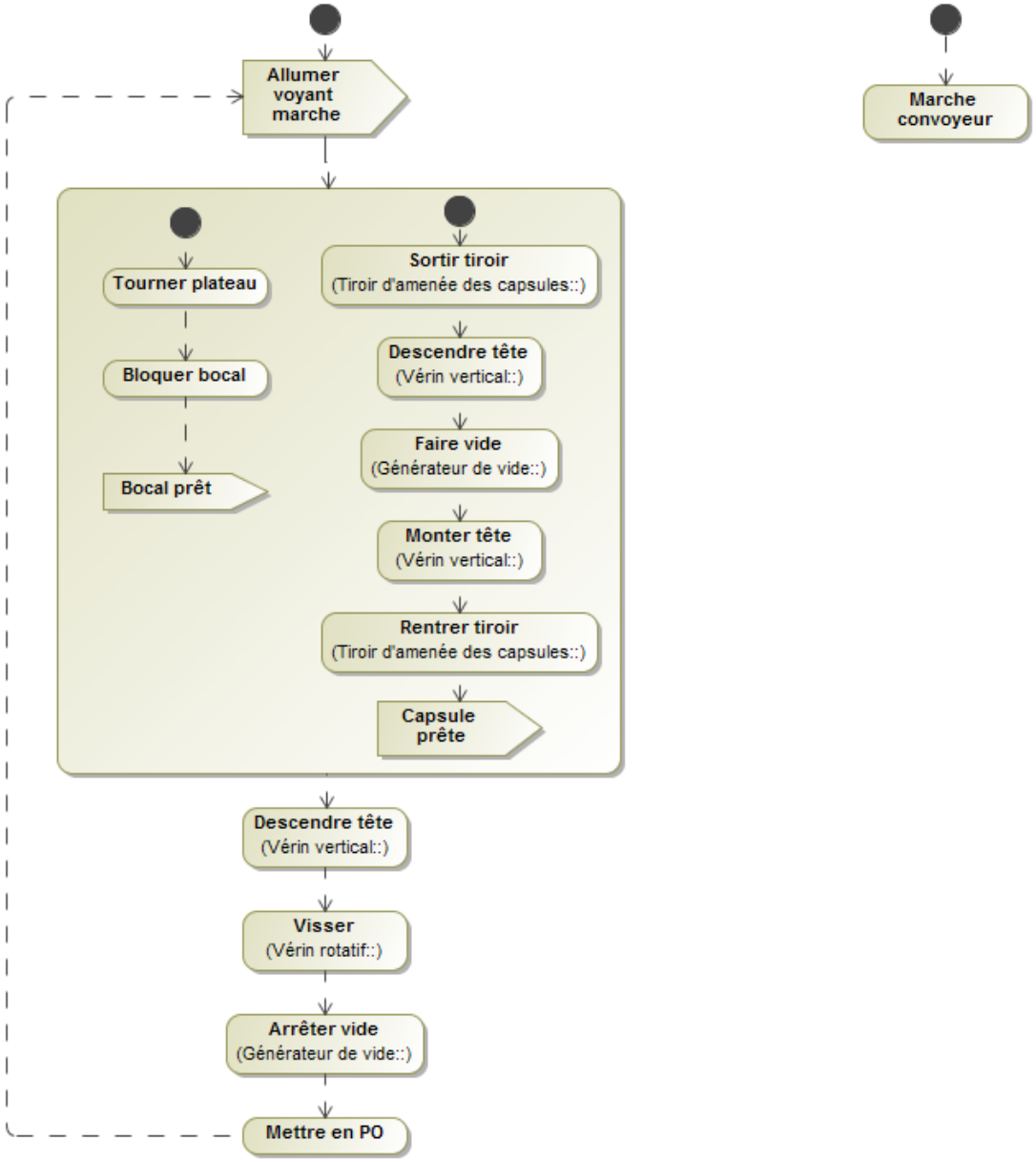


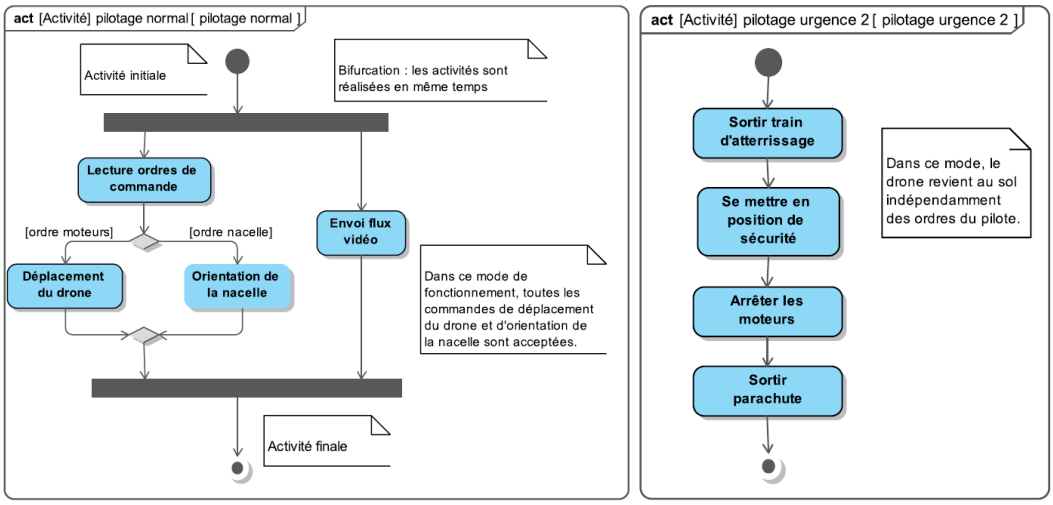
Diagramme d'activité du capsulage de la capsuleuse

b) Détail

Le diagramme d'activités permet de décrire la transformation des flux d'entrées en flux de sorties (matières, énergies, informations) par le biais de séquences d'actions ou d'activités déclenchées par des flux de contrôle.

Lorsqu'une tâche est terminée, la suivante commence : **il n'y a pas d'événement associé aux transitions** (au contraire du diagramme d'états).

Exemple



Signaux et événements

En plus de consommer et de produire des paramètres, une activité peut recevoir et émettre des signaux.

Les activités peuvent communiquer en incluant l'émission d'un signal et dans une autre la réception d'événements.

Il faut utiliser pour cela des types d'action particuliers, possédant chacun une représentation graphique spécifique :

- drapeau "entrant" : réception d'un événement
- drapeau "sortant" : envoi d'un signal
- "sablier" : événement temporel

3.4. Structures algorithmiques de base

Suite à la description des diagrammes d'états et d'activités, on peut dégager quelques structures de base dans un algorithme, indépendamment de tout langage spécifique.

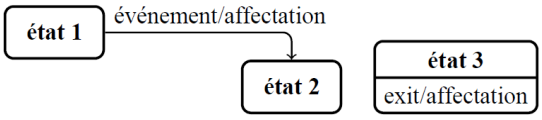
a) L'affectation

L'affectation consiste à donner une valeur particulière à une variable.

Cela peut se faire à l'aide d'une action et ne prend pas de temps significatif.

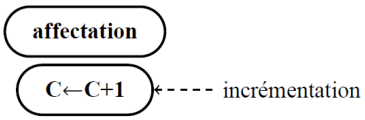
Avec le diagramme d'états

Exemple



Avec le diagramme d'activités

Exemple



b) Suite 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.

c) Fonctions et procédures

Scinder un algorithme en plusieurs fonctions et procédures permet :

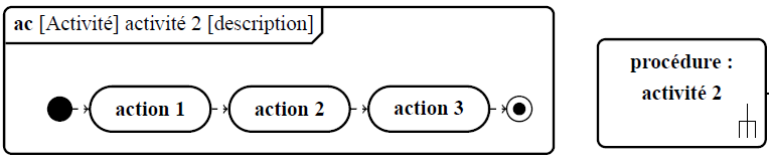
1. de décomposer un problème général en plusieurs problèmes élémentaires
2. de pouvoir réutiliser des programmes réalisant des tâches élémentaires.

Fonction ou procédure ? — *Attention*

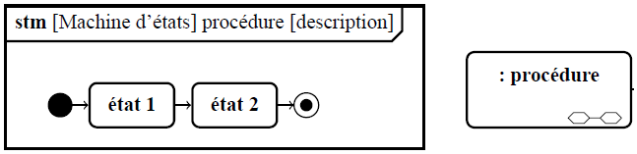
La **fonction** comporte une succession d'instructions et renvoie une valeur, une liste, un objet, etc...

La **procédure** comporte une succession d'instructions mais ne renvoie rien.

Avec le diagramme d'activités — *Exemple*



Avec le diagramme d'états — *Exemple*



d) Structures conditionnelles

Définition

Il y a une **alternative** :

"si ... alors faire ...., sinon faire ...."

Diagramme d'états — *Exemple*

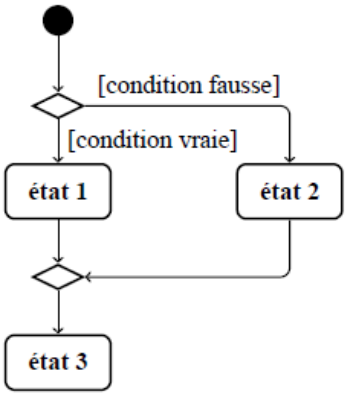
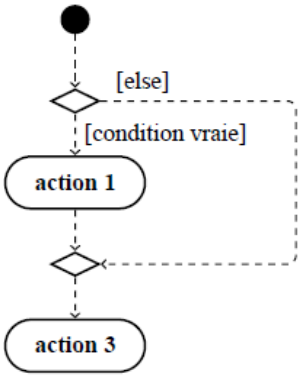


Diagramme d'activités *Exemple*



e) Structures répétitives

Le comportement est **itératif** :

- "tant que condition vraie, faire ..."
- "répéter ... jusqu'à condition vraie"
- "pour variable = valeur initiale, jusqu'à valeur maximale, faire..."

Diagramme d'états *Exemple*

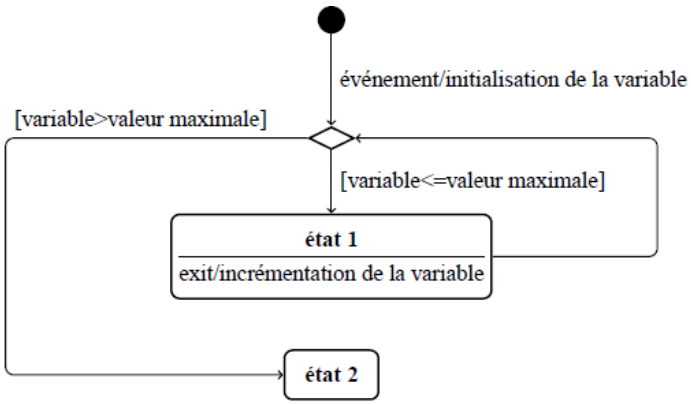
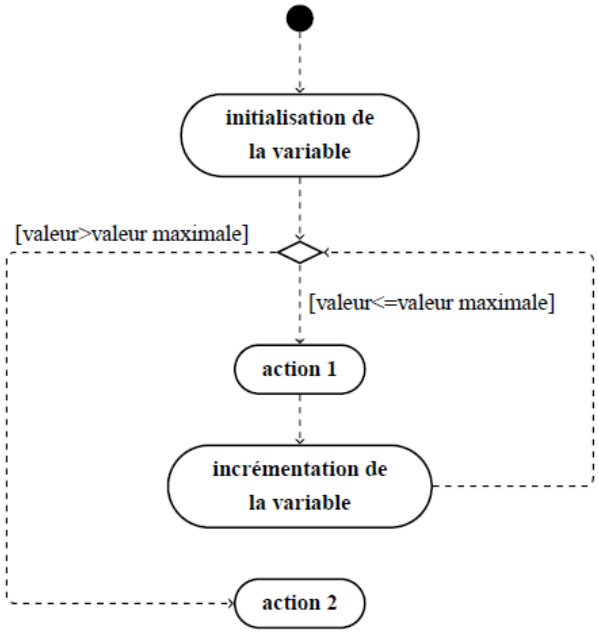


Diagramme d'activités

Exemple



Complément logiciels

On trouvera ici des liens vers des logiciels gratuits et sans installation pouvant aider dans la compréhension du comportement des systèmes à événements discrets.

1. Digital

Digital a été créé par Helmut Neemann dans le but de pouvoir tracer et simuler des diagrammes logiques ou électriques, allant d'un simple circuit basique jusqu'au microprocesseur.

On dispose d'une bibliothèque de portes logiques et composants que l'on vient déposer et relier dans l'espace de travail.

Liens de téléchargement

Fondamental

Page principale sur Github : https://github.com/hneemann/Digital

Lien de téléchargement : https://github.com/hneemann/Digital/releases/latest/download/Digital.zip

Attention

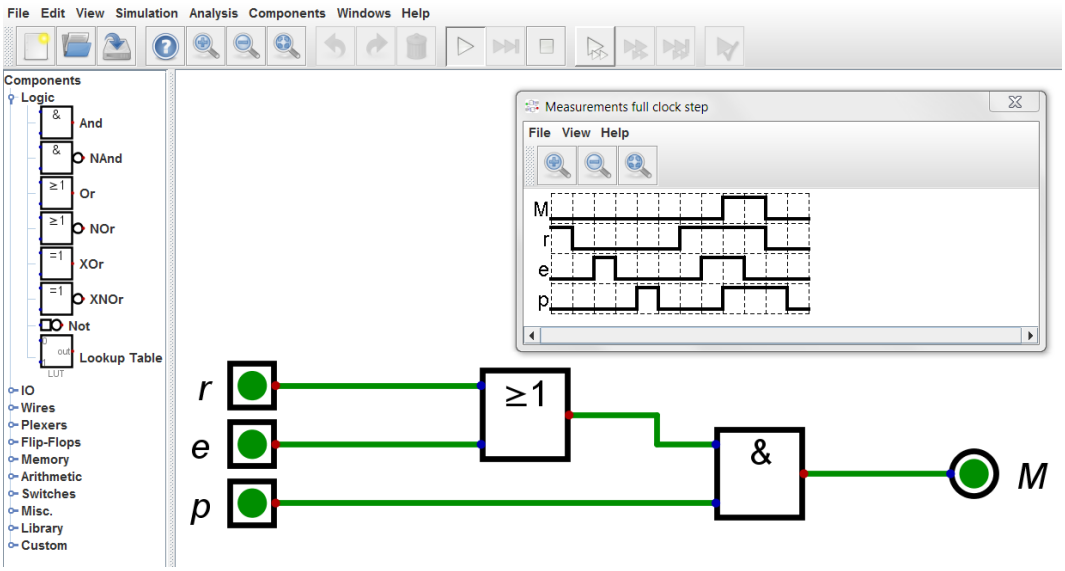
Le fichier téléchargé est un fichier compressé .zip. Après le téléchargement, il faut décompresser (et non simplement ouvrir) ce fichier dans le répertoire de son choix, et exécuter le fichier "Digital.exe" pour lancer le logiciel.

Un tutoriel de démarrage et une documentation sont disponibles dans le logiciel.

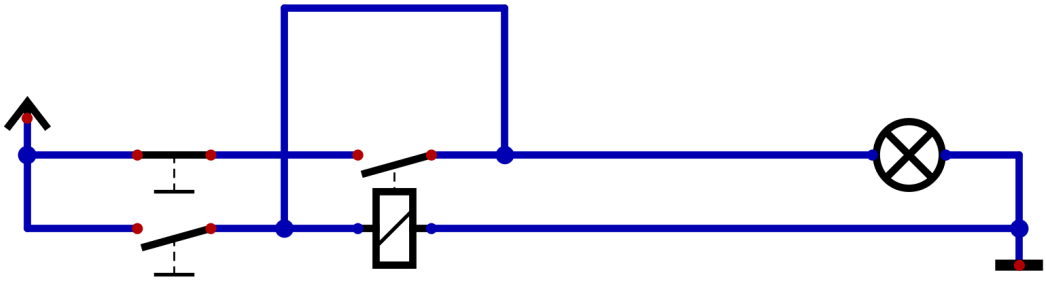
Tracé d'un logigramme

Exemple

Ici on simule le comportement d'un logigramme simple, avec affichage en "temps réel" d'un chronogramme.



Tracé d'un schéma électrique avec relais *Exemple*



2. Pfff

Pfff a été créé par A.D. Monteiro - Ribas dans le but de pouvoir tracer et simuler des schémas pneumatiques.

On dispose d'une bibliothèque de composants que l'on sélectionne dans une page dédiée ; on relie ensuite les composants par un bouton spécifique dans la barre de menus. L'ergonomie est inhabituelle, mais suffisante pour créer des schémas lisibles et surtout, que l'on peut simuler !

Fondamental

Paragraphe sur la page principale : http://admr.cad.free.fr/#la2

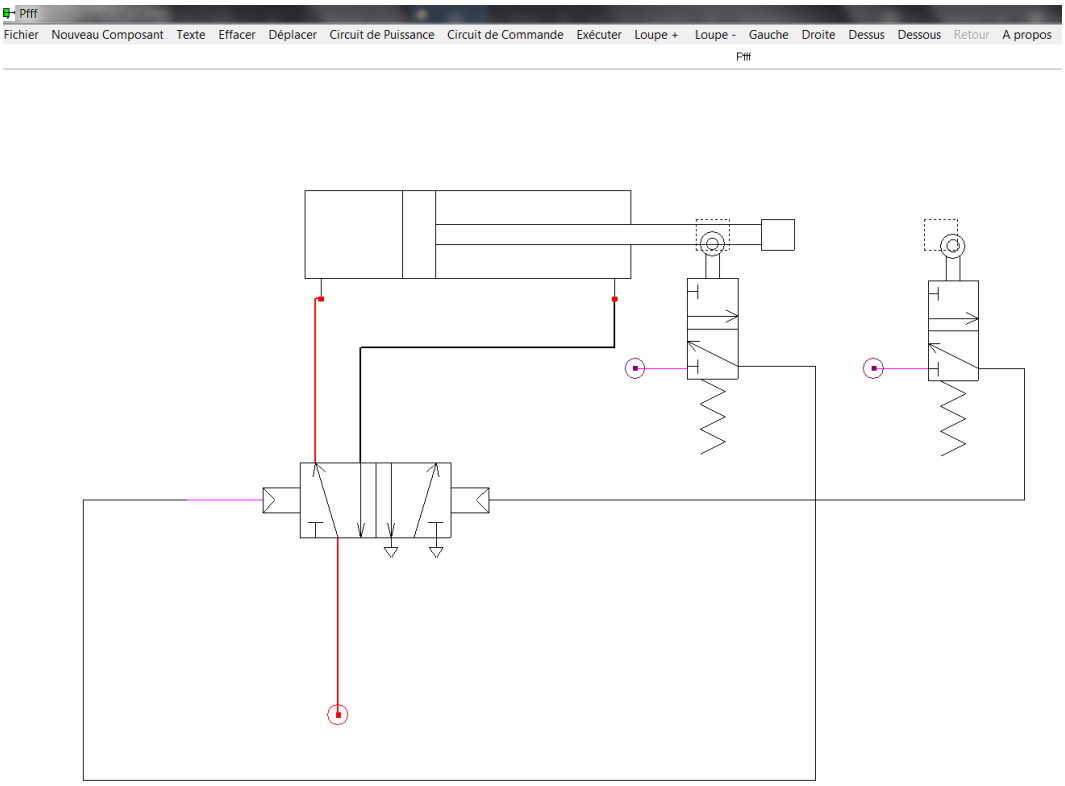
Lien de téléchargement : http://admr.cad.free.fr/pfff.zip

Attention

Le fichier téléchargé est un fichier compressé .zip. Après le téléchargement, il faut décompresser (et non simplement ouvrir) ce fichier dans le répertoire de son choix, et exécuter le fichier "pfff.exe" pour lancer le logiciel.

Circuit "aller - retour"

Exemple



Distinction puissance / commande

Remarque

Lors de la création du schéma, il faut bien distinguer la partie "commande" de la partie "puissance". Les composants ne sont utilisables que dans l'une ou l'autre catégorie.

Solutions des exercices

Solution n°1

[exercice p. 17]

Il faut simplifier les fonctions suivantes en n'utilisant ni la table de vérité ni les tableaux de Karnaugh :

$f_3 = c\,\bar{d}\,b + \bar{c}\,d\,\bar{a}\,b + b\,\bar{c} + a\,b\,c + \bar{a}\,b\,c\,d$

- ☐ $f_3 = b(\bar{c} + c\,a + \bar{d})$
- ☑ $f_3 = b(\bar{c} + c\,a + \bar{d})$
- ☑ $f_3 = b$

Il existe plusieurs manières d'arriver au même résultat ; la solution consiste à utiliser les propriétés d'absorption et les éléments absorbants.

Solution n°2

[exercice p. 27]

Voyant vert V

cd \ ab	00	01	11	10
00	1	0	X	0
01	0	0	X	0
11	X	X	X	X
10	0	0	X	0

V = $\bar{a}\,\bar{b}\,\bar{c}\,\bar{d}$

voyant bleu B

cd \ ab	00	01	11	10
00	0	1	X	0
01	0	0	X	0
11	X	X	X	X
10	1	1	X	0

B = $\bar{a}\,c + b\,\bar{d}$

voyant rouge R

cd \ ab	00	01	11	10
00	0	0	X	1
01	1	1	X	1
11	X	X	X	X
10	0	0	X	1

R = a + d

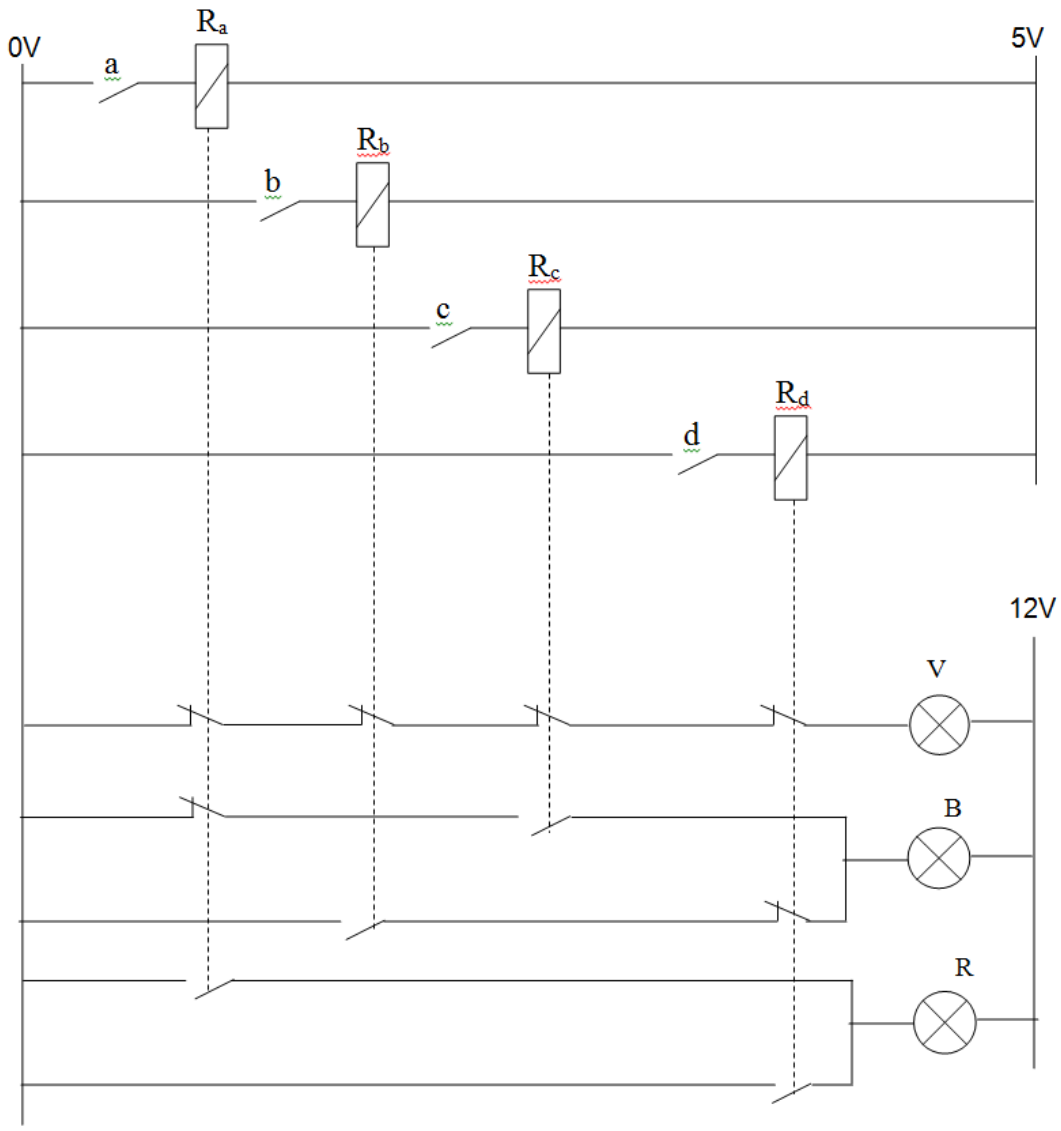
Le voyant vert s'allume (pièce bonne) si aucun des capteurs n'est activé.

Le voyant bleu s'allume (pièce à réusiner) si l'une des deux cotes est trop forte (c = 1 ou b = 1) mais aucune n'est trop faible (a = 0 et d = 0).

Le voyant rouge s'allume (pièce mauvaise) si l'une des deux cotes est trop faible (a = 0 ou d = 0).

Solution n°3

[exercice p. 27]



Solution n°4

[exercice p. 28]

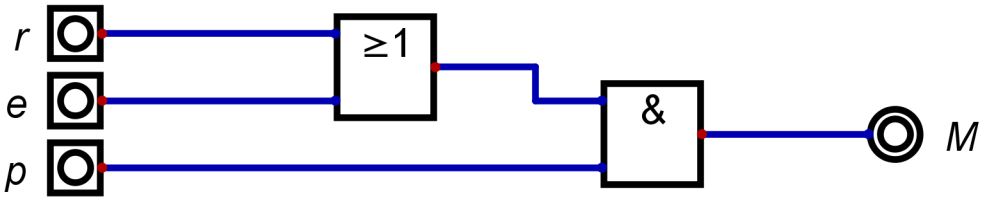
Sans tableau de Karnaugh, par compréhension du texte : $M = \bar{r}.p.e.c + r.p.c$

pe \ cr	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	1	1
10	X	X	1	0

Avec tableau de Karnaugh, l'équation est plus simple (on utilise aussi clairement le fait que le cas $p.c$ est impossible) : $M = pr + pe = p.(r + e)$

Solution n°5

[exercice p. 28]



Solution n°6

[exercice p. 28]

Table de vérité

b	h	m	1D	
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	Cas impossible
1	1	1	0	Cas impossible

Tableau de Karnaugh

Cas d'activation de la vanne 1D

m \ b.h	00	01	11	10
0	0	0	X	1
1	1	0	X	1

Solution n°7

[exercice p. 28]

Sans utilisation de Karnaugh

$$1D = \overline{b}.\overline{h}.m + b.\overline{h}.\overline{m} + b.\overline{h}.m$$

$$\rightarrow 1D = \overline{b}.\overline{h}.m + b.\overline{h}.\overline{m} + b.\overline{h}.m + b.\overline{h}.m$$

$$\rightarrow 1D = \overline{h}.m.(\overline{b} + b) + b.\overline{h}.(\overline{m} + m)$$

$$\rightarrow 1D = \overline{h}.m + b.\overline{h}$$

$$\rightarrow \boxed{1D = \overline{h}.(m + b)}$$

Avec Karnaugh et cas impossibles

$1D = b + m.\bar{h}$

Solution n°8

[exercice p. 29]