

« L'enseignement devrait être ainsi : celui qui le reçoit le recueille comme un don inestimable mais jamais comme une contrainte pénible. »

Albert Einstein

Systèmes logiques combinatoires

1 - Définitions

1.1 – Les variables binaires

Une *variable a est binaire* si elle peut prendre, à tout instant, une valeur unique, parmi un ensemble de deux valeurs possibles. Ces valeurs sont notées par convention 0 et 1.

Exemples :

- un contact électrique est ouvert ou fermé,
- une vanne est ouverte ou fermée,
- un distributeur pneumatique est positionné « à droite » ou « à gauche »...

Un ensemble ordonné de n variables binaires est *un mot binaire* de n digits ou bits.

Exemple : on note (a,b) ou ab le mot binaire à deux variables binaires.

Un ensemble binaire de n variables peut prendre au plus 2^n valeurs différentes.

Exemple : le mot binaire ab peut prendre 4 valeurs notées 00, 01, 10, 11.

La valeur d'un mot, à un instant donné, est appelée *état binaire* de ce mot.

1.2 – La notion de codage binaire

Le choix de l'affectation d'un état du mot binaire à une valeur de la variable constitue l'opération de codage. Elle n'est jamais unique.

Par exemple, le codage binaire d'un chiffre décimal n'est pas unique. Il faut au minimum quatre variables binaires pour coder un chiffre entre 0 et 9. Le tableau ci-dessous présente 2 codages binaires usuels en logique combinatoire : le codage binaire naturel et le codage binaire réfléchi (code GRAY).

d	n_4	n_3	n_2	n_1	r_4	r_3	r_2	r_1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1

¹ La « Pascaline », 1^{ère} machine à calculer, inventée en 1643 par Blaise PASCAL (1623-1662). Cette machine fut conçue par Pascal pour aider son père dans son travail de réorganisation des finances en Bretagne.

7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1

La première colonne est le chiffre décimal. Les 4 colonnes suivantes correspondent au codage binaire naturel. On code dans ce codage le chiffre 5 par $n_4n_3n_2n_1 = 0101$; la variable n_4 est le bit de poids fort et la variable n_5 est le bit de poids faible.

Remarquons que :

$$d = \sum_{i=1}^4 n_i 2^{i-1}$$

Les 4 dernières colonnes correspondent au code Gray. Remarquons qu'avec ce codage, une seule variable binaire change d'état quand on passe d'une ligne à la suivante.

1.3 – Les fonctions binaires

Une *fonction binaire* f est une fonction qui, à un mot binaire $y = x_n \dots x_1$, associe une variable binaire z .

Une *fonction binaire est combinatoire* si et seulement si l'état de la sortie à chaque instant ne dépend que de l'état du mot d'entrée à cet instant. Dans ce cas, la fonction f est indépendante du temps et l'on a :

$$s(t) = f(e(t)), \forall t$$

Un système combinatoire est un système comprenant plusieurs fonctions combinatoires. A partir d'un mot binaire d'entrée, le système élabore plusieurs variables de sortie.

Exemple : le transcodage qui associe le code binaire réfléchi $r_4r_3r_2r_1$ d'un chiffre décimal au code binaire naturel $n_4n_3n_2n_1$ de ce même chiffre est un système combinatoire constitué de 4 fonctions et de 4 variables.

Remarque : une fonction logique de n variables binaires est une combinaison de ces n variables binaires et des éléments 0 et 1. Il existe $\sum_{k=0}^{2^n} C_{2^n}^k = 2^{2^n}$ combinaisons de ces n variables.

En effet, considérons une fonction de n variables. Le nombre de combinaisons de ces n variables s'élève à 2^n ; à partir de la table de vérité de ces fonctions, on peut dénombrer :

- les fonctions ayant zéro « un » dans la colonne de droite : $C_{2^n}^0 = 1$;
- les fonctions ayant un « un » dans la colonne de droite : $C_{2^n}^1 = 2^n$;
- les fonctions ayant p « un » dans la colonne de droite : $C_{2^n}^p$;
- les fonctions ayant 2^n « un » dans la colonne de droite : $C_{2^n}^{2^n} = 1$.

Le nombre total de fonction correspond à la somme de toutes les fonctions précédentes et conduit au résultat énoncé.

Pour le cas où $n = 1$, il y a 4 fonctions possibles – « toujours faux », « identité », « négation » et « toujours vrai ».

Pour $n = 2$, il y a 16 fonctions possibles, mais seul un petit nombre intervient dans la réalisation technologique associée à la structure d'algèbre de Boole. Nous donnons ci-après les fonctions les plus utilisées appelés opérateurs de base.

2 - Algèbre de Boole²

Il s'agit de l'algèbre des variables et des fonctions binaires.

2.1 – Les opérateurs de base

Soit la variable binaire a . On définit les 2 opérateurs binaires sur cette variable :

- l'opérateur OUI ou identité,
- l'opérateur NON ou négation ou complémentaire

avec

a	OUI	NON
0	0	1
1	1	0

Le tableau ci-dessus est appelé une table de vérité.

Soit les variables a et b . On définit les 2 opérateurs binaires sur ces 2 variables :

- l'opérateur OU noté « + », appelé somme logique tel que $a + b = 1$ si et seulement si l'une au moins des variables a ou b est égale à 1.
- l'opérateur ET noté « . », appelé produit logique tel que $a.b = 1$ si et seulement si les deux variable a et b sont égales à 1.

La table de vérité de chaque fonction est donné par :

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

a	b	$a.b$
0	0	0
0	1	0
1	0	0
1	1	1

- l'opérateur dilemme ou OU exclusif noté « \oplus » défini par la table de vérité

² George BOOLE, mathématicien et logicien anglais (1815-1864).

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

On vérifie que : $a \oplus b = \bar{a}b + a\bar{b} = (a + b)(\bar{a} + \bar{b})$

2.2 – Les propriétés issues de la structure d'algèbre

Propriétés : L'ensemble des variables binaires muni des opérations NON, OU et ET possède une structure d'algèbre. Les propriétés de cette algèbre se traduisent par le tableau ci-après :

	OU	ET
commutativité	$a+b = b+a$	$a.b = b.a$
associativité	$a+(b+c) = (a+b)+c$	$a.(b.c) = (a.b).c$
distributivité	$a+(b.c) = (a+b).(a+c)$	$a.(b+c) = (a.b)+(a.c)$
élément neutre	$a+0 = a$	$a.1 = a$
élément absorbant	$a+1 = 1$	$a.0 = 0$
complémentaire	$a + \bar{a} = 1$	$a.\bar{a} = 0$

Ces propriétés se démontrent aisément, en utilisant par exemple, les tables de vérité associées.

D'autres propriétés intéressantes sont utilisées dans cette algèbre. Le tableau suivant en font un descriptif.

	OU	ET
involution	$\bar{\bar{a}} = a$	
idempotence	$a+a = a$	$a.a = a$
absorption 1	$a+ab = a$	$a.(a+b)=a$
absorption 2	$a + \bar{a}b = a + b$	$a.(\bar{a} + b) = ab$
consensus	$ab + \bar{a}c + bc = ab + \bar{a}c$	$(a + b).(\bar{a} + c).(b + c) = (a + b).(\bar{a} + c)$
De Morgan	$\overline{a + b} = \bar{a}.\bar{b}$	$\overline{a.b} = \bar{a} + \bar{b}$

A titre d'exercice, démontrer ces propriétés.

Le théorème de De Morgan se généralise à une expression de n variables binaires ; on a en effet :

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

2.3 – Les systèmes complets d'opérateurs logiques

C'est un ensemble à partir duquel il est possible de construire toutes les fonctions logiques : cet ensemble est appelé une base des opérateurs logiques. Il permet de construire la structure d'algèbre de Boole. On a vu que {NON,ET,OU} répond à cette définition. Il est possible de montrer que les ensembles suivants répondent également à la définition des bases d'opérateurs :

{NON,ET}, l'opérateur OU s'exprime en effet avec ces deux opérateurs : $a + b = \overline{\overline{a} \cdot \overline{b}}$.

{NON,OU}, l'opérateur ET s'exprime en effet avec ces deux opérateurs : $a \cdot b = \overline{\overline{a} + \overline{b}}$.

Il existe aussi deux opérateurs couramment utilisés dans la pratique industrielle qui possède chacun seul une structure de base d'opérateurs ; il s'agit des opérateurs NOR (NON OU) et NAND (NON ET) définis de la manière suivante :

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

opérateur NOR

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

opérateur NAND

D'après les lois de De Morgan, on a :

$$\overline{a + b} = \overline{a} \cdot \overline{b} \quad \text{et} \quad \overline{a \cdot b} = \overline{a} + \overline{b}$$

Il n'y a plus de symbolique utilisée pour ces opérateurs

2.4 – La dualité

Soit une fonction binaire f . L'expression duale de f est obtenue en interchangeant les opérateurs ET et OU d'une part et les valeurs 0 et 1 d'autre part dans son expression.

Cette dualité découle des propriétés de symétrie de l'algèbre binaire par rapport aux fonctions ET et OU d'une part, aux valeurs 0 et 1 d'autre part.

Exemples : $a \cdot (a + b) = a$ a pour expression duale : $a + (a \cdot b) = a$

$$a + 0 = a \quad \text{a pour expression duale : } a \cdot 1 = a$$

2.5 – La forme décimale d'une fonction logique

Une fonction logique peut être définie à l'aide de valeurs décimales par l'intermédiaire du codage binaire naturel. Par exemple, on a :

le mot binaire $x_3 x_2 x_1 = 110$ est représenté en décimal par $d = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6$.

Toute fonction logique peut alors s'écrire comme :

- ☞ somme des états pour lesquels elle vaut 1 que l'on notera : $f = \sum(d_1, \dots, d_p)$
- ☞ produit des états pour lesquels elle vaut 0 que l'on notera : $f = \prod(d_1, \dots, d_p)$

On affecte un « poids » constitué d'une puissance croissante ($p-1$) de 2 à la variable placée à la place p à partir de la droite dans la parenthèse représentant la fonction. Une variable complémentée vaut 0, une variable non complémentée vaut sont poids binaire (par exemple, une variable placée en troisième place vaut 4, une variable en 5^{ème} place vaut 16).

Exemples : $f_1(a, b, c) = \bar{a} \bar{b} \bar{c} + \bar{a} b \bar{c} + \bar{a} b c + a \bar{b} c + a b \bar{c}$ s'écrit $f_1(a, b, c) = \sum(0, 2, 3, 5, 6)$

$f_2(a, b, c) = (\bar{a} + \bar{b} + \bar{c})(a + b + \bar{c})(\bar{a} + b + c)$ s'écrit $f_2(a, b, c) = \prod(1, 4, 7)$.

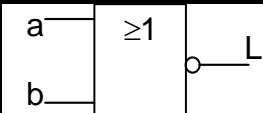
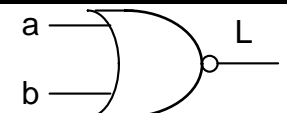
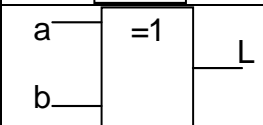
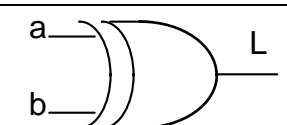
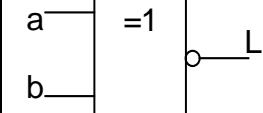
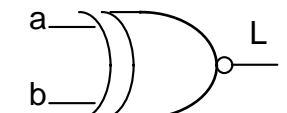
Cette écriture décimale est assez pratique, puisqu'elle diminue les risques d'erreur d'écriture car il est plus facile d'oublier un trait sur une variable que de transformer un 4 en 6 par exemple ; en outre, comme on le verra plus loin lors de la simplification des fonctions logiques, l'utilisation du tableau de Karnaugh à l'aide de cette représentation se révèle assez commode.

3 - La représentation des fonctions logiques usuelles

3.1 – Les logigrammes

Le tableau ci-après donne les représentations schématisés normalisés des opérateurs logiques usuels. La 3^{ème} colonne fournit la représentation de la norme internationale IEC (International Electrotechnical Commission) dont les éléments sont appelés des portes logiques et le 4^{ème} colonne celle de la norme américaine MIL STD 806 B.

Opérateur	Fonction logique	Norme IEC	Norme américaine
OUI	$L = a$		
NON	$L = \bar{a}$		
ET	$L = a \cdot b$		
OU	$L = a + b$		
NAND	$L = \overline{a \cdot b} = \bar{a} + \bar{b}$		

NOR	$L = \overline{a + b} = \bar{a} \cdot \bar{b}$		
OU exclusif	$L = a \oplus b = \bar{a}b + a\bar{b}$		
IDENTITE	$L = \overline{a \oplus b} = \bar{a}\bar{b} + ab$		

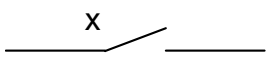

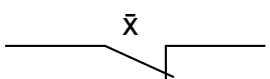

Remarque : l'association d'opérateurs logiques pour représenter une équation logique n'est pas unique.

Exemple à traiter : Représenter sous forme de logigramme (schéma de portes logiques) l'équation logique ci-dessous :

$$L = \bar{a}b + ac + d\bar{b}$$

3.2 – Les circuits à contacts et relais

La représentation à contacts se réalise avec les deux méthodes dont les éléments de base sont indiqués ci-après :

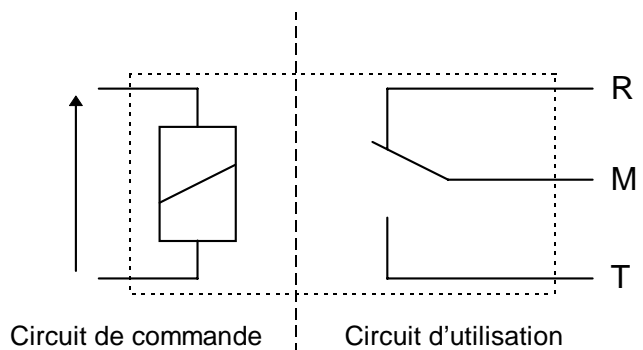
contact travail		
contact repos		
représentations	normalisée	simplifiée

De plus, un relais est un élément électromécanique comportant :

- un circuit de commande muni d'une bobine d'excitation, et de faible niveau d'énergie,
- un circuit d'utilisation indépendant du précédent comportant un ou plusieurs contacts repos ou travail ou repos/travail, et de niveau d'énergie plus élevé que le précédent.

Le schéma ci-contre précise la structure du relais. Ici, nous n'avons considéré qu'un seul contact repos/travail réalisé par trois sorties R, T, M comportant un point milieu M.

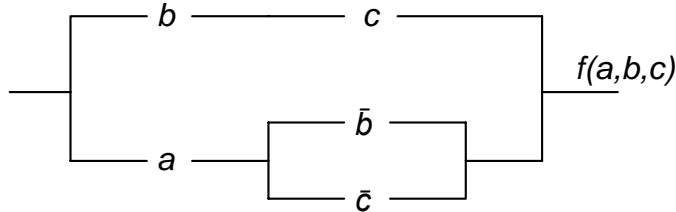
Le fonctionnement est donné par : Lorsque la bobine n'est pas excitée, les contacts sont dans la situation repos : R est fermé, T est ouvert. Quand la bobine est parcourue par un courant, la



lame métallique bascule sur le contact T et le contact R est ouvert. Si l'on associe une variable binaire x au contact repos et une variable binaire y au contact travail, on peut représenter le fonctionnement du relais par le tableau :

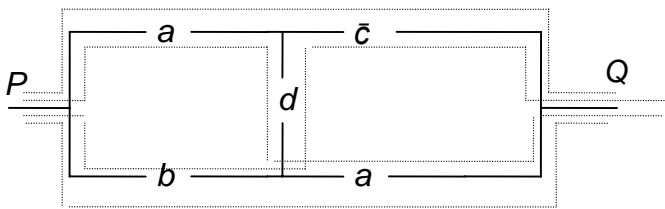
commande	contacts		variables binaires	
	repos	travail	x	y
$I = 0$	fermé	ouvert	1	0
$I \neq 0$	ouvert	fermé	0	1

exemple de schéma simplifié d'une fonction à l'aide de contacts :



Détermination des équations logiques à partir du schéma

1 – La méthode des chemins

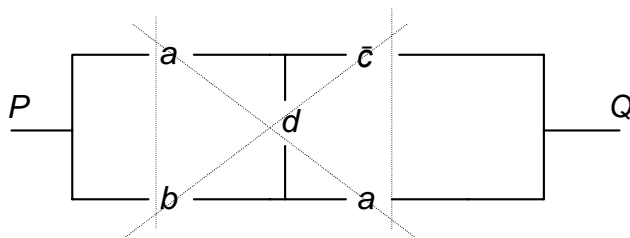


On prend en compte tous les chemins possibles pour aller de P à Q (analyse par les 1). On obtient la 1^{ère} forme canonique (voir § suivant). Ici, il y a 4 chemins possibles, ce qui donne

l'équation du circuit :

$$f(a,b,c,d) = a\bar{c} + ad + b\bar{c}d + ab$$

2 – La méthode des coupures



On prend en compte toutes les coupures possibles du circuit entre P et Q et garantissant les ouvertures des contacts (analyse par les 0). On obtient la seconde forme canonique. L'équation s'écrit maintenant :

$$f(a,b,c,d) = (a + b)(a + d)(b + \bar{c} + d)(a + \bar{c})$$

4 – Les formes canoniques

4.1 – Première forme canonique : somme de produits canonique d'une fonction

- cas d'une fonction d'une variable

On considère une fonction f de la variable binaire x . Soit $f(1)$ la valeur de la fonction quand $x = 1$ et $f(0)$ sa valeur quand $x = 0$. On vérifie aisément que l'on a :

$$f(x) = x \cdot f(1) + \bar{x} \cdot f(0)$$

- *cas d'une fonction de deux variables*

On considère une fonction de deux variables binaires x et y . A partir du cas précédent, on en déduit la relation :

$$f(x, y) = x \cdot y \cdot f(1,1) + x \cdot \bar{y} \cdot f(1,0) + \bar{x} \cdot y \cdot f(0,1) + \bar{x} \cdot \bar{y} \cdot f(0,0)$$

Les termes $f(0,0)$, $f(1,0)$, $f(0,1)$ et $f(1,1)$ représentent les 4 valeurs que peut prendre la fonction pour les états d'entrées correspondantes

Toute fonction de 2 variables binaires peut s'écrire comme l'expression précédente, c'est-à-dire comme la somme de produit de monôme. Cette expression constitue la première forme canonique de toute fonction logique de deux variables.

On peut étendre à un nombre quelconque de variables ; si n est le nombre de variables, la forme canonique comportera 2^n produits de monômes.

exemples :

fonction ET : $f(0,0) = 0$, $f(1,0) = 0$, $f(0,1) = 0$ et $f(1,1) = 1$, on en déduit le résultat :

$$f(x,y) = x \cdot y$$

On a développé la fonction par les 1

fonction OU exclusif : $f(0,0) = 0$, $f(1,0) = 1$, $f(0,1) = 1$ et $f(1,1) = 0$, on en déduit le résultat :

$$f(x,y) = x \cdot \bar{y} + \bar{x} \cdot y$$

4.2 – Seconde forme canonique : produit de sommes canonique d'une fonction

- *cas d'une fonction d'une variable*

On considère une fonction f de la variable binaire x . Soit $f(1)$ la valeur de la fonction quand $x = 1$ et $f(0)$ sa valeur quand $x = 0$. On vérifie aisément que l'on a :

$$f(x) = (x + f(0)) \cdot (\bar{x} + f(1))$$

- *cas d'une fonction de deux variables*

On considère une fonction de deux variables binaires x et y . A partir du cas précédent, on en déduit la relation :

$$f(x, y) = (x + y + f(0,0)) \cdot (x + \bar{y} + f(0,1)) \cdot (\bar{x} + y + f(1,0)) \cdot (\bar{x} + \bar{y} + f(1,1))$$

Les termes $f(0,0)$, $f(1,0)$, $f(0,1)$ et $f(1,1)$ représentent les 4 valeurs que peut prendre la fonction pour les états d'entrées correspondantes

Toute fonction de 2 variables binaires peut s'écrire comme l'expression précédente, c'est-à-dire comme le produit de sommes de monôme. Cette expression constitue la seconde forme canonique de toute fonction logique de deux variables.

On peut étendre à un nombre quelconque de variables ; si n est le nombre de variables, la forme canonique comportera 2^n produits de monômes.

exemples :

fonction ET : $f(0,0) = 0$, $f(1,0) = 0$, $f(0,1) = 0$ et $f(1,1) = 1$, on en déduit le résultat :

$$f(x,y) = (x + y) \cdot (\bar{x} + y) \cdot (x + \bar{y})$$

On a développé la fonction par les zéros.

fonction OU exclusif : $f(0,0) = 0$, $f(1,0) = 1$, $f(0,1) = 1$ et $f(1,1) = 0$, on en déduit le résultat :

$$f(x,y) = (x + y) \cdot (\bar{x} + \bar{y})$$

5 - Méthodes de simplification des fonctions logiques

Une fonction peut être représentée par plusieurs expressions algébriques différentes mais équivalentes. Nous allons appliquer deux méthodes aboutissant à « minimiser » les expressions des fonctions logiques.

5.1 – La méthode algébrique

Elle repose sur une application astucieuse des propriétés algébriques des variables binaires.

On étudie la méthode sur un exemple d'une fonction de 4 variables a , b , c et d .

Soit la fonction : $f(a,b,c,d) = \sum(1,5,7,9,11,12,14,15)$

Écrite sous forme canonique, on obtient :

$$f(a,b,c,d) = \underset{1}{\bar{a}\bar{b}\bar{c}d} + \underset{2}{\bar{a}b\bar{c}d} + \underset{3}{\bar{a}bcd} + \underset{4}{a\bar{b}\bar{c}d} + \underset{5}{a\bar{b}cd} + \underset{6}{ab\bar{c}\bar{d}} + \underset{7}{abc\bar{d}} + \underset{8}{abcd}$$

Simplifions cette expression :

* (1+2) donne $\bar{a}\bar{c}d$

* (3+8) donne bcd

* (1+4) donne $\bar{b}\bar{c}d$ d'où le résultat : $f(a,b,c,d) = \bar{a}\bar{c}d + bcd + \bar{b}\bar{c}d + acd + ab\bar{d}$

* (5+8) donne acd

* (6+7) donne $ab\bar{d}$

On pourrait, en simplifiant d'une manière différente, montrer que :

$$f(a,b,c,d) = \bar{a}\bar{c}d + bcd + a\bar{b}d + ab\bar{d}$$

ou

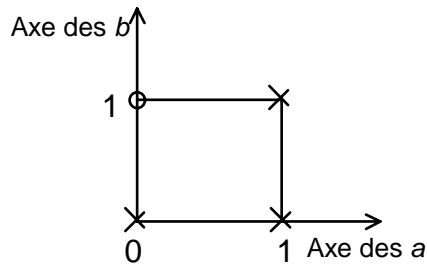
$$f(a,b,c,d) = \bar{b}\bar{c}d + \bar{a}bd + ab\bar{d} + acd$$

5.2 – La méthode des tableaux de Karnaugh

5.2.1 – La représentation graphique des fonctions booléennes

En algèbre classique, il est souvent très pratique de représenter une fonction par un graphe qui donne en lui-même un grand nombre de renseignements sur cette fonction. On a cherché à en faire autant avec les fonctions booléennes. Seulement, les fonctions booléennes ne sont pas continues, elles ne peuvent donc se représenter que par un nombre fini de points (2^n pour n variables). Il suffira de porter sur chacun de ces points, « images » d'une combinaison, des variables de la fonction, la valeur de la fonction, c'est-à-dire 0 ou 1.

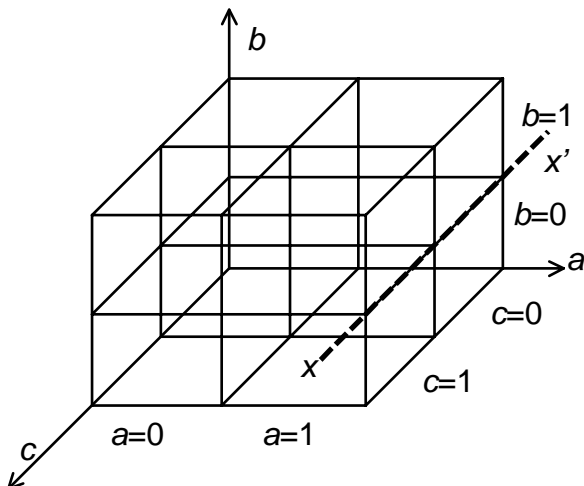
Soit la fonction de 2 variables : $f = a + \bar{b}$, on peut faire un graphe dans un plan, et définir ainsi les 4 points images des 4 combinaisons possibles sur a et b ; on conviendra, par exemple de mettre une croix dans les cas où $f = 1$ et un cercle dans le cas où $f = 0$.



$$\begin{cases} f = 0 \text{ pour } \bar{a} \cdot b \\ f = 1 \text{ pour } \bar{a} \cdot \bar{b} + a \cdot \bar{b} + a \cdot b \end{cases}$$

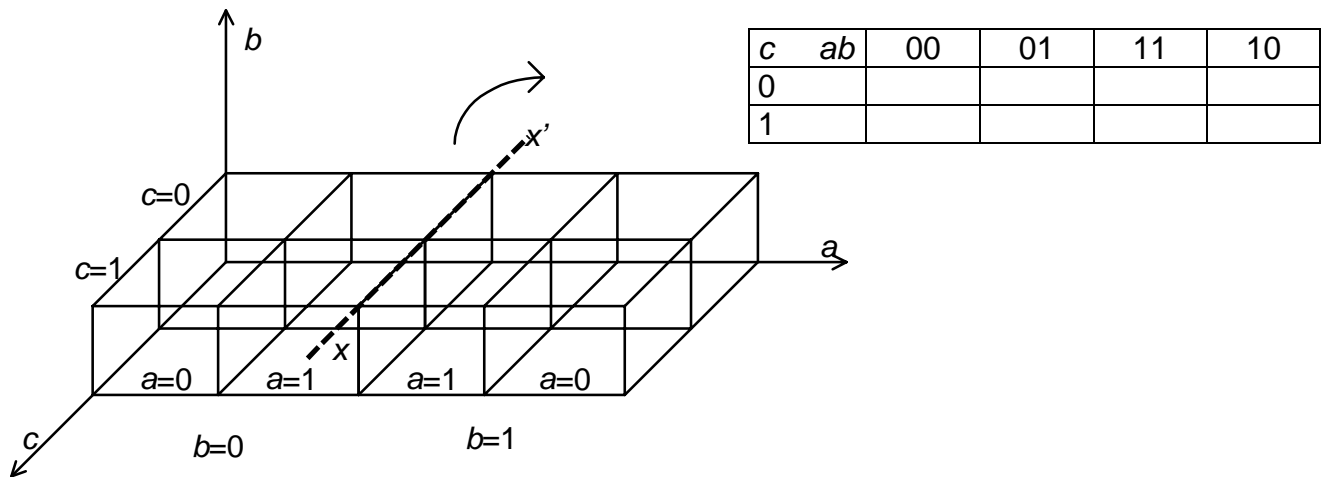
On peut « agrandir » chacun de ces points de façon à faire une case carrée dans laquelle on écrira 0 ou 1 et l'on obtient un tableau de Karnaugh pour 2 variables.

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



Avec 3 variables, il vient naturellement à l'esprit de prendre un troisième axe et de travailler dans l'espace à 3 dimensions. Chaque case devenant un cube, chaque « plaque » de 4 petits cubes ayant une des 3 variables dans le même état :

Comme il n'est pas très commode d'exploiter une figure spatiale, nous allons repasser dans le plan, en « dépliant » ce cube suivant, par exemple, l'axe $x'x$. Nous obtenons le tableau de Karnaugh pour 3 variables.



Nous voyons que la première et la dernière colonne, bien que séparées dans le tableau appartiennent en fait à la même plaque de 4 petits cubes ($a = 0$) et ont donc une variable commune. On dit qu'elles sont adjacentes. Elles le sont réellement dans l'espace. Cela n'apparaît pas directement sur le tableau.

Au-delà de 3 variables, il faut théoriquement travailler dans des espaces à 4, 5, 6, ... dimensions pour avoir le graphe de la fonction. La remarque faite sur le dépliage pour abaisser le nombre de dimensions reste vraie quelque soit le nombre de variables. Nous obtenons des tableaux de Karnaugh à 4, 5, 6, ... variables.

	00	01	11	10	x'
00	0	+	+	0	
01	-	x	x	-	
11	-	x	x	-	
10	0	+	+	0	x

Si l'on trace le symétrique par rapport à $x'x$, on ajoute une dimension donc une variable e .

La partie gauche sera le « rez-de-chaussée » $e = 0$, la partie droite le « 1^{er} étage ».

Les conditions d'adjacence notées restent les mêmes. Il s'en ajoute d'autres.

Cette façon de considérer le tableau de Karnaugh comme le graphe de la fonction booléenne, permet de comprendre pourquoi d'une part les cases cerclées par exemple sont adjacentes, et pourquoi, d'autre part, le codage se fait à l'aide du code Gray.

Les tableaux de Karnaugh sont utilisés pour la simplification des équations.

5.2.2 – Passage réciproque d'une représentation algébrique à une représentation graphique

Chaque case d'un tableau de Karnaugh représente un point de la fonction booléenne, c'est-à-dire une combinaison et une seule des variables. Pour ce point, on inscrira dans la case le signe 0 ou 1 selon la valeur prise par la fonction. Puisque

chaque point représente une combinaison exclusive des variables on peut écrire cette combinaison par une opération ET portant sur toutes les variables, sous forme directe ou complémentée.

Exemple : la case dans laquelle $a = 0$, $b = 1$, $c = 0$, et $d = 1$ s'écrira $\bar{a}b\bar{c}d$ (c'est un « minterme »).

Pour obtenir la représentation algébrique de la fonction, il suffit de faire la somme logique (OU) de toutes les combinaisons donnant un 1 à la fonction.

Exemple : $f = \bar{a}b\bar{c}d + \bar{a}bcd + ab\bar{c}d$

La fonction est donc bien écrite sous forme algébrique. Mais cette forme a une particularité : c'est une somme de produits, c'est-à-dire de monômes ayant toutes les variables qui y figurent. On envisage quelquefois la deuxième forme du tableau de Karnaugh conduisant à la seconde forme canonique. Pour représenter l'exemple ci-dessus, la case $a = 0$, $b = 1$, $c = 0$, $d = 1$ s'écrira $\bar{a} + b + \bar{c} + d$. La fonction s'écrira algébriquement en faisant le produit de toutes les cases où $f = 1$.

Exemple : $f = (\bar{a} + b + \bar{c} + d)(\bar{a} + \bar{b} + c + d)(a + \bar{b} + c + d)$

Méthode de simplification :

Règle : On peut regrouper 2^k cases adjacentes pour lesquelles la fonction booléenne de n variables vaut 1 correspondant à un monôme qui peut s'écrire en fonction de $n - k$ variables. Il faut respecter les symétries du tableau.

Une expression algébrique irréductible d'une fonction f peut être obtenue à partir de son tableau de Karnaugh en effectuant la somme de tout ensemble de monômes, qui vérifie les propriétés suivantes :

– l'union des cases couvertes par chacun des monômes est égale à l'ensemble des cases de valeur 1 (dans le cas contraire l'expression obtenue n'est pas équivalente à f puisqu'il existe au moins une case pour laquelle la fonction prend une valeur différente de la somme),

- si un monôme est supprimé alors la propriété précédente n'est plus vérifiée (cette condition signifie qu'aucun monôme n'est redondant),
- pour aucun monôme considéré, il n'existe de monôme qui l'inclus (cette condition signifie que les monômes sont les plus grands possibles).

Remarques : il faut faire les regroupements les plus « gros » possibles, afin d'obtenir les monômes les plus petits possibles.

Il faut également faire le minimum de regroupement.

Exemples d'application :

simplification de la fonction $f = \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + \bar{a}bc$

<i>c</i>	<i>ab</i>	00	01	11	10
0		1			1
1		1			1

Le regroupement des 4 cases adjacentes permet de « supprimer » les variables changeant de valeurs : ici, on peut supprimer a et c . Puisque $b = 0$ pour ces 4 cases, il vient :

$$f = \bar{b}$$

simplification de la fonction $f = \bar{b}\bar{c}\bar{d} + \bar{a}bd + acd + ab\bar{d} + ab\bar{c}d$

<i>cd</i>	<i>ab</i>	00	01	11	10
00		0	0	1	0
01		1	1	1	1
11		0	1	1	1
10		0	0	1	0

4 regroupements donnant la simplification : $f = \bar{c}d + bd + ab + ad$.

Terminons ce paragraphe, en affirmant, par application de la dualité des opérations ET et OU, que tout ce qui a été fait pour les expressions de sommes de produits peut être transposé pour les expressions de produits de sommes en regroupant les zéros du tableau de Karnaugh (et en n'oubliant pas de transformer les 0 en 1 et les 1 en 0).

Si l'on regroupe les zéros de l'exemple précédent, on trouve :

$$f = (b + d)(a + d)(a + b + \bar{c})$$

5.2.3 – Les fonctions booléennes incomplètes

Certaines fonctions ne sont pas spécifiées pour toutes les combinaisons des variables ou certaines combinaisons de variables sont physiquement impossible. Pour ce dernier cas, prenons l'exemple d'un chariot se déplaçant entre deux fins de course notés a et b : il est évident que la combinaison de variable $ab = 11$ est physiquement interdite. On ne peut rien dire pour cette combinaison de variables quant à l'état du chariot. On n'affecte aucune valeur de sortie à cet état. Il y a indifférence de la sortie vis à vis de cette combinaison de variables. Dans le tableau de Karnaugh lié à la fonction booléenne traduisant l'état du chariot, on indique ϕ dans la case $ab = 11$.

Si l'on s'intéresse à la fonction f traduisant l'état de fin de course du chariot, son tableau de Karnaugh donne :

		<i>a</i>	
		0	1
<i>b</i>	0		1
	1	1	ϕ

En regroupant d'après les règles de simplification, et en prenant $\phi = 1$, on obtient :

$$f = a + b, \text{ et non pas } f = a\bar{b} + \bar{a}b$$

6 - Le codage de l'information

6.1 – Les systèmes de numération

Une base B de numération est un ensemble de B symboles (B entier exprimé en base 10).

Les 3 systèmes de base utilisés en sciences industrielles sont les bases 10, 2 et 16 appelées respectivement décimale, binaire et hexadécimale (on utilise parfois la base 8 appelée base octale).

Les relations de changement de base sont données par :

- pour un nombre entier : $N_{(B_1)} = \sum_{i=0}^n a_{i(B_1)} (B_2)^i_{(B_1)}$
 - pour un nombre non entier : $Q_{(B_1)} = \sum_{i=-\infty}^{+\infty} a_{i(B_1)} (B_2)^i_{(B_1)}$
- on note alors : $N_{(B)} = (a_n a_{n-1} \dots a_0)_{(B)}$

exemple : expression d'un nombre décimal en base 2

$$N_{(10)} = \sum_{i=0}^n a_{i(10)} 2^i$$

à titre d'exercice, montrer que : $(11011)_2 = (27)_{10}$ et $(73)_{10} = (1001001)_2$

Le système hexadécimal comporte 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

On a par exemple $(EDF)_{16} = (3807)_{10}$.

6.2 – Le codage de l'information

Le codage est un élément essentiel de la gestion des flux traversant les systèmes ; on doit coder les informations dans de multiples domaines, par exemple, pour le tri du courrier, pour la commande numérique des machines outils, pour l'utilisation de l'INTERNET par le protocole TCP/IP par exemple, ...

Cette codification utilise essentiellement la logique binaire, mais il existe d'autre codification basée sur des logiques de type floue, par exemple. Nous présentons dans la suite des codages très couramment utilisés dans l'industrie.

6.2.1 – Le code binaire naturel

Nous l'avons déjà présenté au § 1.2.

Il se prête parfaitement au traitement des opérations arithmétiques. Il faut cependant une grande quantité de bits pour exprimer un nombre dès que celui-ci est élevé : un mot binaire de 4 bits ne pourra représenter qu'un nombre compris entre 0 et 15 (c'est un quartet). Un mot binaire de 8 bits ne pourra représenter qu'un nombre compris entre 0 et 255 (c'est un octet). Il faudra un mot de 10 bits pour exprimer un nombre compris entre 0 et 1023 (c'est un kilo).

Un autre inconvénient du code binaire naturel est qu'il peut introduire des erreurs lors du codage de grandeurs variant de façon ordonnée. Entre deux codes successifs, plusieurs bits pourront alors être amenés à changer simultanément (par exemple entre 01 et 10, 2 bits changent simultanément, et physiquement, c'est impossible, il y a forcément une transition générant un code parasite 00 ou 11 pouvant entraîner une erreur).

6.2.2 – Le code binaire réfléchi (ou code GRAY)

Ce code a déjà été présenté au § 1.2.

Il pallie l'inconvénient que nous venons de signaler, car un seul bit change de valeur quand on passe d'un nombre décimal au suivant.

Le passage du code binaire naturel au code Gray, est donné par la relation :

$$G = \frac{B \oplus 2B}{2}$$

exemple : $(5)_{10} = (0101)_2 = B$, on en tire $0101 \oplus 1010 = 1111 \div 10 = 0111 = G$

Le code binaire réfléchi est utilisé dans les codeur absolu (roue codeuse).

6.2.3 – Le code DCB (Décimal Codé Binaire)

On code chaque chiffre selon son équivalent binaire :

$0 \rightarrow 0000$, $1 \rightarrow 0001$, $2 \rightarrow 0010$, ..., $9 \rightarrow 1001$

exemple : $(9708)_{10} = (1001\ 0111\ 0000\ 1000)_{DCB}$

Il est utilisé notamment dans le codage de l'affichage numérique.

6.2.4 – Le code p parmi n

Le code p parmi n est un code à n bits dont p bits sont à 1 et $n - p$ bits sont à 0. Le nombre de combinaison répondant à cette définition est égal à C_n^p .

Ce code est auto correcteur car la lecture du code peut être associée à la vérification du nombre de 1 et de 0 dans l'information et permet ainsi la détection d'une éventuelle erreur.

Ce code est personnel car il existe C_n^p ! arrangements de la codification ce qui permet la personnalisation du code.

Exemple : le code 3 parmi 5 permet $C_5^3 = 10$ combinaisons est utilisé dans le code postal.

6.2.5 – Le code ASCII (*American Standard Code for Information Interchange*)

C'est un code alphanumérique comportant 7 bits d'informations et 1 bit de parité. Il est utilisé en particulier pour les échanges d'informations entre une unité centrale (UC) et des périphériques en informatique. Il permet le codage de 128 informations différentes. Le bit de parité a pour rôle d'avoir la valeur 0 ou 1 afin que le codage sur 8 bits des informations contienne un nombre pair de 1.

Actuellement le code ASCII est sur 8 bits, le bit de parité est enlevé. Un autre code est envisagé, car le code ASCII est devenu insuffisant pour les nouveaux microprocesseurs extrêmement performants (technologies Pentium IV, Celeron, Athlon, Duron,...).

Il existe, bien entendu, de nombreux autres types de codage.

Le passage d'un code à un autre s'appelle le **transcodage**.

Le saviez-vous ? Le microprocesseur a été inventé en 1971 par Ted Hoff à Santa Clara (Californie), dans une région qui sera connue plus tard, sous le nom de *Silicon Valley*.



Georges BOOLE

LE CODE ASCII

American Standard Code for Information Interchange

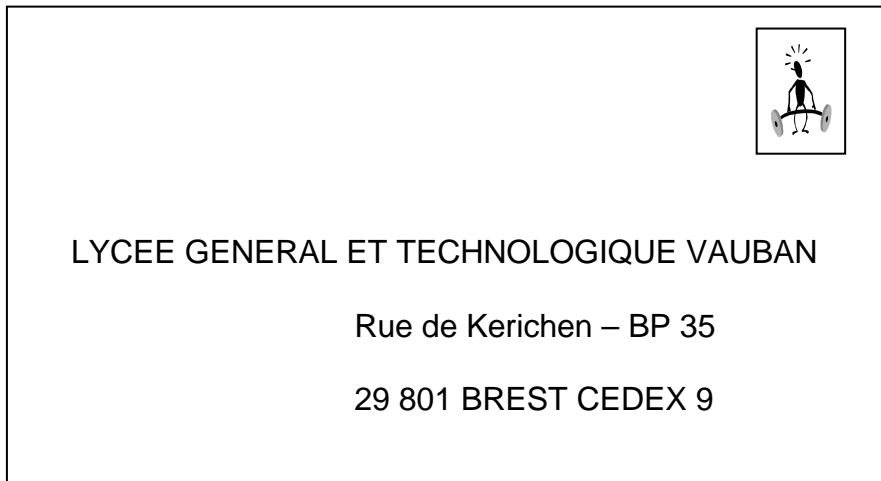
Table en codage hexadécimal

HEX	0	1	2	3	4	5	6	7
0		(DEL)	SP	0	@	P	`	p
1	(SOH)	DC1	!	1	A	Q	a	q
2	(STX)	DC2	«	2	B	R	b	r
3	(ETX)	DC3	#	3	C	S	c	s
4	(EOT)	DC4	\$	4	D	T	d	t
5	(ENQ)	(NAK)	%	5	E	U	e	u
6	(ACK)	(SYN)	&	6	F	V	f	v
7	BEL	(ETB)	'	7	G	W	g	w
8	(BS)	CAN	(8	H	X	h	x
9	(HT)	EM)	9	I	Y	i	y
A	(LF)	SUB	*	:	J	Z	j	z
B	(VT)	ESC	+	;	K	[k	{
C	(FF)	(→)	,	<	L	\	l	
D	(CR)	(←)	-	=	M]	m	}
E	SO	(↑)	.	>	N	^	n	~
F	SI	(↓)	/	?	O	_	o	

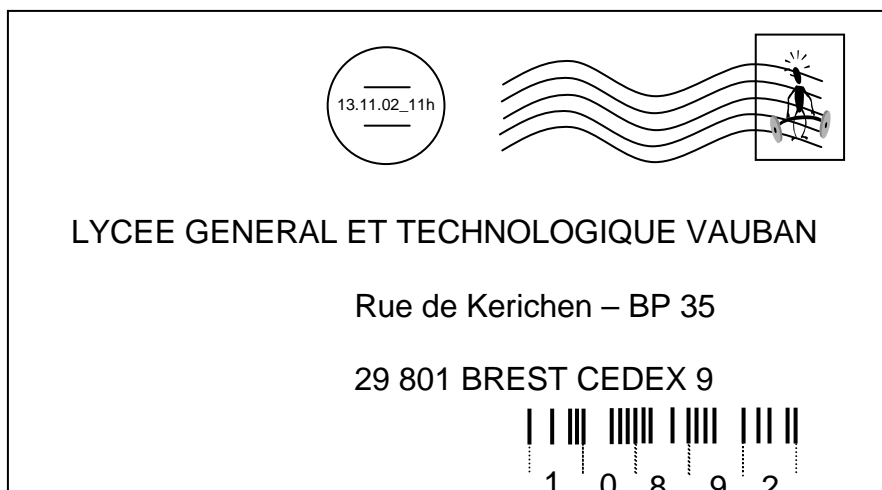
Exemple d'utilisation : lettre G ⇒ code ASCII : 47
 lettre k ⇒ code ASCII : 6B

Le code postal codé par code à barres

COURRIER A EXPEDIER



COURRIER TRAITE PAR LA POSTE



Code utilisé par la poste : code 3 parmi 5

Opération de reconnaissance optique de caractères par traitement OCR, puis traduction en code à barres sous la forme de bâtonnets fluorescents imprimés sur le courrier.

	b4	b3	b2	b1	b0
0	0	0	1	1	1
1	0	1	0	1	1
2	0	1	1	0	1
3	0	1	1	1	0
4	1	0	0	1	1
5	1	0	1	0	1
6	1	0	1	1	0
7	1	1	0	0	1
8	1	1	0	1	0
9	1	1	1	0	0

Trois codes à barres

Code numérique très dense, mais dont la moins bonne fiabilité intrinsèque oblige à l'utiliser soit en longueur fixe, soit avec une clé de contrôle (voir annexe). Le code 2 parmi 5 entrelacé utilise la même codification des caractères que le code 2 parmi 5, mais en entrelaçant les caractères deux par deux. Le premier caractère est codé avec les barres, tandis que le deuxième utilise les espaces de la même zone, et ainsi de suite. Les chiffres de rang impair sont donc codés avec les barres, tandis que les chiffres de rang pair sont codés avec les chiffres de rang pair. La conséquence est que le code 2 parmi 5 entrelacé encode toujours un nombre pair de caractères. Ce code utilise pour chaque caractère cinq éléments, dont 2 sont larges, d'où son nom. Comment calculer la longueur d'un symbole en code 2 parmi 5 entrelacé ? La formule est la suivante pour un symbole sans clé de contrôle :

2 parmi 5 entrelacé



$$\text{Longueur} = (N(2R+3)+6+R)X$$

Où la *longueur* représente la distance de la première barre à la dernière, marges non comprises.

N = le nombre de caractères utiles

R = le ratio barres larges/barres étroites

X = épaisseur des barres étroites.

Code numérique très dense spécifié par le GENCOD pour les applications de la grande distribution. Les symboles EAN codent 12 ou 8 chiffres, le cas le plus normal étant 12 caractères (toujours numériques). En plus de ces caractères, est toujours encodée une clé de contrôle. Pour certaines applications particulière de ce code, des caractères supplémentaires sont ajoutés à la droite du symbole de base, séparés de celui-ci par un espace (identification des journaux et magazines). Le code EAN utilise une technique de décodage particulièrement adaptée aux symboles imprimés sur les emballages par les moyens d'imprimerie traditionnels. Aux USA, ce code correspond au code UPC. Pour permettre une lecture omnidirectionnelle plus aisée, le symbole peut être décodé en deux moitiés puis reconstitué: ainsi, chaque moitié peut facilement être plus haute que large.

Code EAN 13



Code alphanumérique haute densité, permettant comme le code 93 de coder le jeu ASCII complet. Deux densités différentes sont obtenues suivant que les caractères encodés sont numériques ou alphanumériques. Une clé de contrôle est toujours utilisée. Comme les autres codes haute densité, le code 128 est un code continu. Chaque caractère est symbolisé au moyen de onze modules (sauf le caractère de début et de fin qui en comprends treize). Chaque caractère est composé de 3 barres et 3 espaces (4 et 3 pour le caractère de début/Fin). Les barres représentent toujours un nombre pair de modules et les espaces un nombre impair.

128 ALPHANUMERIQUE



Le code dit "EAN 128" est en fait un code 128, mais dans lequel un caractère de fonction (Fonction 1) est placé en première position du message. Ce caractère, lu par le lecteur, n'est pas transmis au système. Il permet au lecteur de s'assurer que le symbole lu est un EAN 128 et non un 128 standard. Comment calculer la longueur d'un symbole en code 128 ?

La formule est la suivante:

$$\text{Longueur} = X(11N+35)$$

Où la *longueur* représente la distance de la première barre à la dernière, marges non comprises.

N = le nombre de caractères utiles. N représente soit un caractère alphabétique (lettre ou signe spécial), soit deux caractères numériques (chiffres de 0 à 9). Les caractères de fonction éventuellement nécessaire doivent être ajoutés aux caractères utiles pour le calcul de la longueur.

X = épaisseur des barres étroites