Chapitre 5

Algèbre booléenne

5.1. L'algèbre de Boole


George **Boole**
(1815-1864)

L'algèbre de Boole, ou calcul booléen, est la partie des mathématiques qui s'intéresse aux opérations et aux fonctions sur les variables logiques. Elle fut inventée par le mathématicien britannique George **Boole**. Aujourd'hui, l'algèbre de Boole trouve de nombreuses applications en informatique et dans la conception des circuits électroniques.

On appelle B l'ensemble constitué de deux éléments appelés valeurs de vérité {FAUX, VRAI}. Cet ensemble est aussi noté $B = \{0, 1\}$, notation que l'on utilisera désormais.

Sur cet ensemble on peut définir les lois ET et OU et une transformation appelée « complémentaire » (parfois « inversion » ou « contraire »).

ET

Elle est définie de la manière suivante : *a* ET *b* est VRAI si et seulement si *a* est VRAI et *b* est VRAI. Cette loi est aussi notée :

- $a \cdot b$
- $a \wedge b$ (dans quelques notations algébriques, ou en **APL**)
- *a*&*b* ou *a*&&*b* (**Perl**, **C**, **PHP,** ...)
- *a* AND *b* (**Ada**, **Pascal**, **Python**, ...)

OU

Elle est définie de la manière suivante : *a* OU *b* est VRAI si et seulement si *a* est VRAI ou *b* est VRAI, ou si *a* et *b* sont vrais. Cette loi est aussi notée :

- $a+b$
- $a \vee b$ (dans quelques notations algébriques ou en **APL**)
- *a*|*b* ou *a*||*b* (**Perl**, **C**, **PHP,** ...)
- *a* OR *b* (**Ada**, **Pascal**, **Python**, ...)

NON

Le contraire de « *a* » est VRAI si et seulement si *a* est FAUX. Le contraire de *a* est noté :

- $\overline{a}$
- $\neg a$
- ~*a* (dans quelques notations algébriques ou en **APL**)
- !*a* (**C**, **C++**...)
- NOT *a* (**ASM**, **Pascal**, ...)

5.2. Fonctions logiques et tables de vérité

Une table de vérité est un tableau qui représente des entrées (en colonne) et des états binaires (0 et 1). Le résultat, exprimé lui aussi sous forme binaire, se lit dans la dernière colonne.

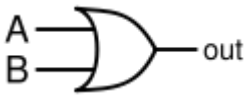
	Symbole de la porte logique (voir § 5.4)	Opération booléenne	Table de vérité

ET (AND)

$A \cdot B$

Entrées		Sortie
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

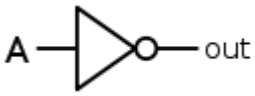
OU (OR)



$A + B$

Entrées		Sortie
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

NON (NOT)

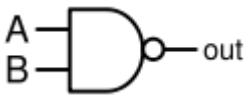


$\overline{A}$

Entrée	Sortie
A	NOT A
0	1
1	0

En électronique, une porte NON est plus communément appelée *inverseur*. Le cercle utilisé sur la représentation est appelé « bulle », et montre qu'une entrée ou une sortie est inversée.

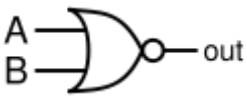
NON-ET (NAND)



$\overline{A \cdot B}$

Entrées		Sortie
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

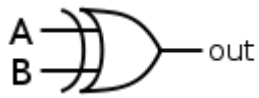
NON-OU (NOR)



$\overline{A + B}$

Entrées		Sortie
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

OU exclusif (XOR)



$A \oplus B$

$= A \cdot \overline{B} + \overline{A} \cdot B$

Entrées		Sortie
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Priorité

Pour faciliter leur compréhension, il a été décidé que ces opérations seraient soumises aux mêmes règles que les opérations mathématiques. La fonction ET (multiplication logique) est ainsi prioritaire par rapport à la fonction OU (somme logique).

On peut évidemment, pour s'aider, placer des parenthèses dans les opérations.

Exercice 5.1

Écrivez les tables de vérité des expressions suivantes :

a. $\overline{A \cdot \overline{B}}$ **b.** $A + B \cdot \overline{C}$ **c.** $A \cdot \overline{B} + (\overline{C} \oplus D)$

Quelques propriétés

Toutes ces propriétés peuvent être facilement démontrées à l'aide de tables de vérité.

Associativité

Comme avec les opérations habituelles, certaines parenthèses sont inutiles :

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

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

Commutativité

L'ordre est sans importance :

$$a + b = b + a$$

$$a \cdot b = b \cdot a$$

Distributivité

Comme avec les opérations mathématiques habituelles, il est possible de distribuer :

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

Attention : comportement différent par rapport aux opérateurs + et · habituels :

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

Élément neutre

$$a + 0 = a$$

$$a \cdot 1 = a$$

Élément nul

$$0 \cdot a = 0$$

$$1 + a = 1$$

Idempotence

$$a + a + a + [...] + a = a$$

$$a \cdot a \cdot a \cdot [...] \cdot a = a$$

Complémentarité

$$a = \neg(\neg a)$$

$$a + \overline{a} = 1$$

$$a \cdot \overline{a} = 0$$

Augustus **de Morgan** (1806-1871)

Lois de De Morgan

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

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

Maurice **Karnaugh**
(né en 1924)

5.3. Tables de Karnaugh

Une table de Karnaugh est une méthode inventée par Maurice **Karnaugh** en 1954 et qui sert à simplifier des équations logiques ou à trouver l'équation logique correspondant à une table de vérité.

La méthode utilisée est graphique. Elle fonctionne très bien avec 3 ou 4 variables, beaucoup moins bien avec 5 ou 6 variables, et plus du tout au-delà !

5.3.1. Construction de la table

Soit la table de vérité de **S** suivante avec les variables **A**, **B**, **C** et **D** :

A	B	C	D	S
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

La table de Karnaugh correspondante se présente ainsi :

S CD AB	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	1	1	1
10	1	0	1	1

Par exemple, la case sur la ligne 10 et sur la colonne 01 correspond à la valeur de S pour laquelle A=1, B=0, C=0 et D=1.

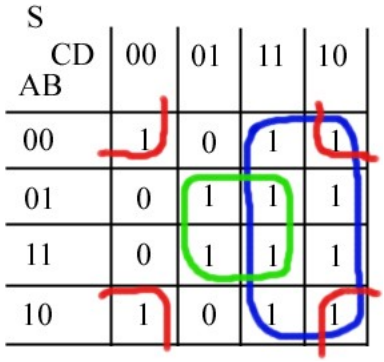
5.3.2. Méthode de recherche de l'équation de la table de vérité

- Pour trouver une équation logique, il faut regrouper les valeurs de S égales à 1 dans des rectangles ayant comme nombre de cases une puissance de 2 (16, 8, 4, 2 ou 1 cases).

- Les groupes formés doivent être les moins nombreux possibles, mais ils doivent englober tous les 1. On peut faire des chevauchements.
- On a intérêt à dessiner des rectangles les plus grands possibles.
- On peut considérer cette table comme un tore (comme dans Pac-Man) : la dernière ligne est adjacente à la première et la première colonne est adjacente à la dernière. On peut ainsi regrouper des 1 se trouvant à ces emplacements.

Pour les tables à 4 variables, il faut de préférence procéder dans l'ordre suivant :

1. les rectangles 8 cases, puis
2. les rectangles 4 cases, puis
3. les rectangles 2 cases, et enfin
4. les cases uniques.



S CD / AB	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	1	1	1
10	1	0	1	1

Dans l'exemple ci-contre : on peut former un rectangle de 8 cases (en bleu), puis un carré de 4 (en vert) et enfin on peut rassembler les deux 1 restants dans un groupe de 4 (en rouge).

Mise en équation

Le rectangle bleu correspond à l'équation « C », car dans ces deux colonnes C est toujours égal à 1. A, B et D prennent les valeurs 0 ou 1.

Le carré vert correspond à l'équation « B et D », car dans ces cases B = 1 et D = 1, tandis que A et C valent soit 0, soit 1.

L'équation du rectangle rouge est « $\overline{B}$ et $\overline{D}$ », car dans ces quatre cases, B = 0 et D = 0, alors que A et C valent soit 0, soit 1.

On fait ensuite la somme des trois équations : « S = C ou (B et D) ou ($\overline{B}$ et $\overline{D}$) », que l'on peut aussi écrire sous forme de l'équation « $S = C + B \cdot D + \overline{B} \cdot \overline{D}$ ».

Exercice 5.2

Trouvez les équations des tables de vérité de **S**, **T** et **U** avec les variables **A**, **B**, **C** et **D** :

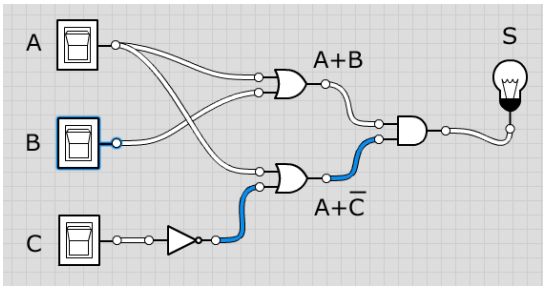
Facultatif :

Vérifiez vos formules à l'aide d'un programme Python.

A	B	C	D	S	T	U
0	0	0	0	0	1	0
0	0	0	1	0	1	1
0	0	1	0	0	0	1
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	1	1	1
0	1	1	0	1	1	0
0	1	1	1	1	1	0
1	0	0	0	0	0	0
1	0	0	1	1	1	1
1	0	1	0	0	1	1
1	0	1	1	1	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	1	1	0
1	1	1	1	1	1	0

5.4. Circuits logiques

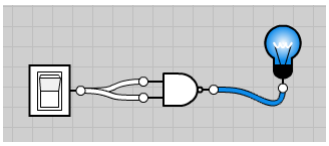
On appelle circuit logique (ou circuit combinatoire) un assemblage de portes logiques reliées entre elles pour schématiser une expression algébrique. Par exemple, l'expression algébrique $S=(A+B)\cdot(A+\overline{C})$ sera schématisée comme suit :



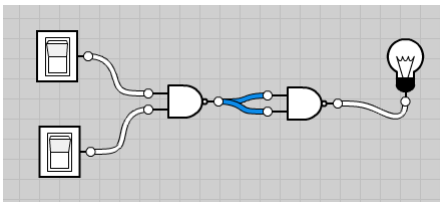
Les circuits logiques ont été dessinés grâce au programme Logicly. http://logic.ly/

La porte NAND

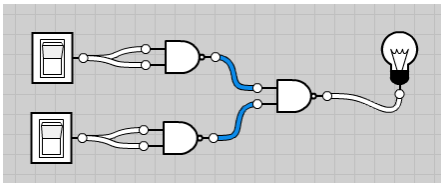
La porte NAND est la plus simple à réaliser du point de vue technologique. Il est possible de réaliser toutes les fonctions logiques en utilisant uniquement ce type de porte.



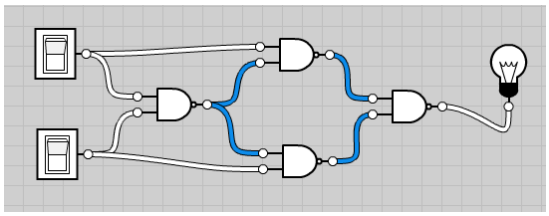
Porte NON



Porte ET



Porte OU



Porte OU exclusif

Exercice 5.3

Reprenez les équations trouvées à l'exercice 5.2, simplifiez-les grâce aux propriétés des fonctions logiques, puis construisez les circuits logiques correspondants.

Exercice 5.4

Réalisez une porte XOR avec des portes AND, OR et NOT.

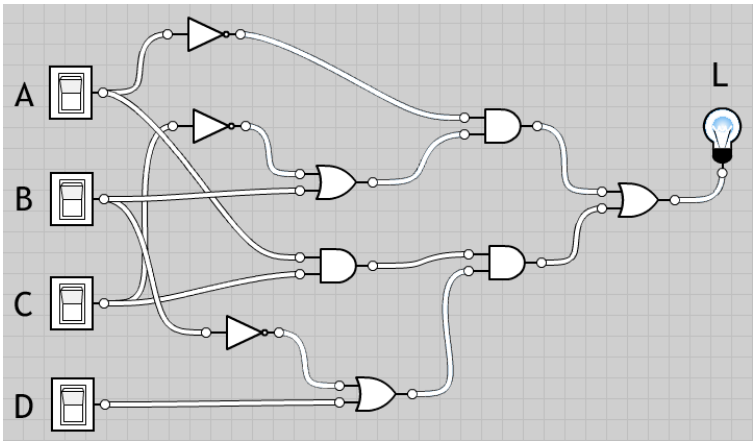
Exercice 5.5

On a trois interrupteurs pouvant être en position 0 ou 1 et trois ampoules pouvant être allumées ou éteintes. On veut créer un circuit logique où le nombre de lampes allumées correspond au nombre d'interrupteurs positionnés sur 1, mais on ne veut pas savoir quels interrupteurs le sont.

1. Établissez les tables de vérité de ce problème.
2. Trouvez l'équation la plus simple possible pour chaque table de vérité.
3. Dessinez le circuit logique correspondant.

Exercice 5.6

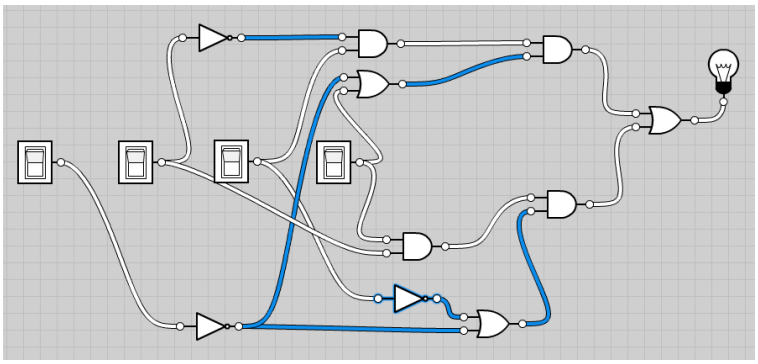
Soit le circuit ci-dessous :



1. Écrivez l'équation de ce circuit.
2. Établissez la table de vérité de ce circuit.

Exercice 5.7

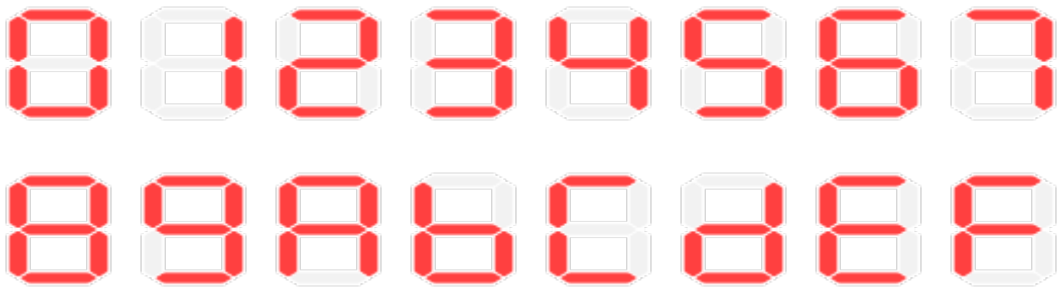
À quoi sert ce circuit ? **Indication :** les quatre interrupteurs représentent un nombre en base 2.



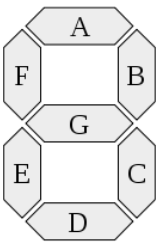
Exercice 5.8

Les afficheurs 7 segments sont un type d'afficheur numérique très présent sur les calculatrices et les montres à affichage numérique : les caractères (des chiffres, bien que quelques lettres soient utilisées pour l'affichage hexadécimal) s'écrivent en allumant ou en éteignant des segments, au nombre de sept. Quand les 7 segments sont allumés, on obtient le chiffre 8.

Voici les 16 symboles représentés avec l'affichage à 7 segments :



Dans un afficheur 7 segments, les segments sont généralement désignés par les lettres allant de A à G (voir ci-dessous).

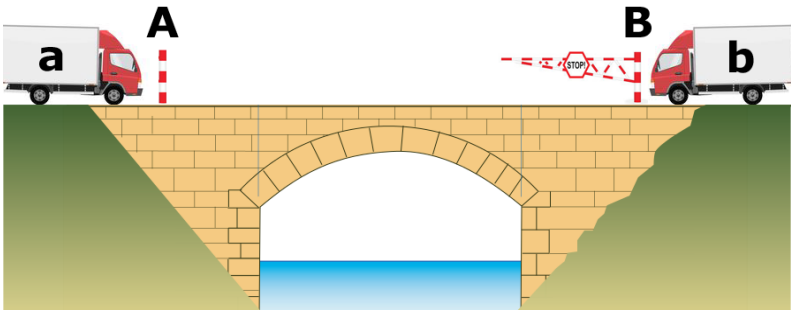


On dispose de 4 interrupteurs qui représenteront les 4 bits d'un nombre exprimé en base 2 et compris entre 0 et 15. On veut que ce nombre en base 2 s'affiche en base 16. Les sept segments seront symbolisés par des ampoules.

1. Établissez les tables de vérité de ce problème.
2. Trouvez les équations les plus simples possible des tables de vérité.
3. Dessinez le circuit logique correspondant.

Exercice 5.9

Un pont peut supporter 10 tonnes au maximum. La route menant au pont est strictement interdite aux véhicules de plus de 10 tonnes. À chaque extrémité du pont se trouve une barrière et une bascule pour mesurer le poids (a ou b) des véhicules.



Si un seul véhicule attend devant le pont, la barrière devant lui (A ou B) s'ouvre. Sinon :

- Si $a + b \leq 10$ tonnes, les barrières A et B s'ouvrent.
- Si $a + b > 10$ tonnes, seule la barrière correspondant au véhicule le plus léger s'ouvre. L'autre véhicule attend que le premier ait franchi le pont, puis le protocole d'ouverture des barrières recommence au début.
- Si $a = b$, la barrière A s'ouvre en priorité.

Indication : a et b n'étant pas des variables binaires, il convient de créer deux variables binaires x et y et de reformuler l'énoncé du problème.

a) Écrivez la table de vérité pour l'ouverture des barrières A et B.
b) Donnez les équations logiques pour l'ouverture des barrières A et B.
c) Dessinez le circuit logique déterminant l'ouverture des barrières.

Sources

[1] Mange Daniel, *Analyse et synthèse des systèmes logiques*, PPUR, 1995

[2] Wikipédia, « Portail de la logique », <http://fr.wikipedia.org/wiki/Portail:Logique>