

# Opérations fondamentales de l'algèbre booléenne

La traduction des raisonnements logiques par des opérations algébriques proposée par le mathématicien George Boole est un des fondements de l'informatique.

## 1 – L'algèbre de Boole

### 1 – Rappels historiques

A partir de 1854, le britannique George Boole propose un mode de calcul permettant de traduire des raisonnements logiques par des opérations algébriques.

Il crée ainsi une branche des mathématiques qui définit des opérations dans un ensemble qui ne contient que 2 éléments, notés 0 et 1 / Faux et Vrai (lien avec la logique) / Ouvert et Fermé (lien avec l'électronique) / False et True (programmation).

L'algèbre de Boole ou calcul booléen, est donc une branche des mathématiques qui s'intéresse à une approche algébrique de la logique, à l'aide d'opérandes, d'opérateurs et d'expressions booléennes.

Opérande :

Un opérande est un élément sur lequel s'applique un opérateur booléen.

Opérateur :

Un opérateur booléen est un opérateur mathématique qui relie 2 variables binaires ou 2 expressions binaire.

Expression booléenne :

Une expression booléenne est une expression dont l'évaluation ne peut être que Vrai ou Faux.

En 1938, l'américain Claude Shannon prouve que des circuits électrique peuvent résoudre tous les problèmes que l'algèbre de Boole peut résoudre. Avec les travaux d'Alan Turing de 1936, cela constitue les fondements de ce qui deviendra l'informatique.

## 2 – Les opérations fondamentales

Les opérations fondamentales de l'algèbre de Boole ne sont plus l'addition et la multiplication des entiers mais :


- la CONJUNCTION, notée  $\&$ ,  $\wedge$ ,  $.$  est lue "et"
- la DISJUNCTION, notée  $|$ ,  $\vee$ ,  $+$  est lue "ou"
- la NEGATION, notée  $\sim$ ,  $-$ ,  $-$  est lue "non"

Ces opérations fondamentales sont des exemples de fonctions logiques.

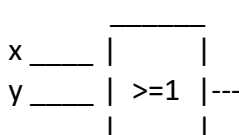
Puisque l'algèbre de Boole ne contient que 2 éléments, pour chacune de ces fonctions, on peut étudier tous les cas possibles et les regrouper dans un tableau appelé table de vérité.

On représente souvent les opérateurs booléens à l'aide de portes logiques qui représentent à la fois la fonction logique qui lui est associée et le circuit électronique de la machine qui laisse passer le courant (Vrai) ou non (Faux) selon que le courant y entre ou non. Le calcul booléen permet donc d'utiliser la puissance du calcul algébrique pour régler des problèmes de logique et se traduit sur machine par des composants électroniques.


La conjonction (et, and) est l'opération définie par :

					
x	y	x & y	x	y	x & y
F	F	F	0	0	0
F	V	F	0	1	0
V	F	F	1	0	0
V	V	V	1	1	1

La disjonction (ou, or) est l'opération définie par :

					
x	y	x   y	x	y	x   y
F	F	F	0	0	0
F	V	V	0	1	1
V	F	V	1	0	1
V	V	V	1	1	1

La négation (non, not) est l'opération définie par :

			x	~x	x	~x
x	1	~x	F	V	0	1
			V	F	1	0

### 3 – Comment évaluer une expression booléenne ?

Comme dans une expression mathématique classique, il faut tenir compte des parenthèses. Les opérateurs booléens prioritaires sont **not** puis **and** et enfin **or** ou **xor**.

#### **Expression comportant uniquement des valeurs booléennes**

Dans une expression comportant uniquement des valeurs booléennes, il faut simplifier l'expression en tenant compte des parenthèses et des règles de priorité entre les opérateurs.

Exemple :

A = True and True or (not False and False)

On commence par ce qui est entre parenthèses :

*not False vaut True*

A = True and True or (True and False)

*Comme True and False vaut False*

A = True and True or False

*And est prioritaire et comme True and True vaut True*

A = True or False

A = True

### ***Expression comportant des variables booléennes***

Pour évaluer une expression booléenne comportant des variables booléennes, on peut dresser une table de vérité de chaque partie de l'expression en tenant compte des parenthèses et des priorités entre les expressions.

Exemple :

*a or b and not a*

a	b	not a	b and not a	a or b and not a
False	False	True	False	False
False	True	True	True	True
True	False	False	False	True
True	True	False	False	True

## **2 – En Python**

Les booléens se notent True et False mais aussi 0 et 1

Les opérateurs sont or, and et not

```
not(True or False) == (not True) and (not False)
```

```
True
```

```
not(1 or 0) == (not 1) and (not 0)
```

```
True
```

L'opérateur == permet de tester si 2 références contiennent les mêmes valeurs en renvoyant un booléen.