Expression booléenne

Sophie Chane-Lune

NSI - Première

Sommaire

Programme officiel

Contenus	Capacités attendues	Commentaires
Valeurs booléennes : 0, 1. Opérateurs booléens : and, or, not. Expressions booléennes	Dresser la table d'une expression booléenne.	Le ou exclusif (xor) est évoqué. Quelques applications directes comme l'addition binaire sont présentées. L'attention des élèves est attirée sur le caractère séquentiel de certains opérateurs booléens.

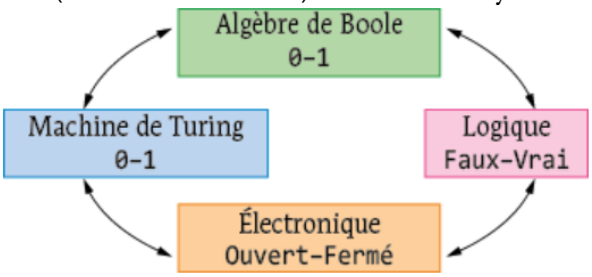
Les circuits d'un ordinateur (mémoire, microprocesseur, ...) manipulent uniquement des chiffres binaires 0 et 1 qui, en interne, sont simplement représentés par des tensions électriques. Ainsi, le chiffre 0 représente une tension basse et le chiffre 1 une tension haute.

Les opérateurs (logiques ou arithmétiques) sur ces nombres binaires sont construits de circuits électroniques dont les briques élémentaires sont appelés **transistors**. Les transistors que l'on trouvent dans les circuits des ordinateurs se comportent comme des interrupteurs qui laissent ou non passer du courant électrique, selon le mode du tout ou rien.

1 Rappels historiques

A partir de 1847, le britannique George Boole propose un mode de calcul permettant de traduire des raisonnements logiques par des opérations algébriques, que l'on appelle **l'algèbre de Boole**.

Il crée ainsi une branche des mathématiques qui définit des opérations dans un ensemble qui ne contient que **deux éléments** notés 0 et 1, ou bien Faux et Vrai (lien avec la logique), ou bien Fermé et Ouvert (lien avec l'électronique), ou bien Eteint et Allumé (lien avec l'électricité) ou encore en Python `False` et `True`.



En 1938, l'américain Claude Shannon prouve que des circuits électriques 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.

En informatique, un **booléen** est une variable à deux états (souvent notés `Vrai` ou `Faux`) qu'on représente numériquement par un bit de valeur 1 pour `Vrai` ou 0 pour `Faux`. Comme il n'existe que deux valeurs booléennes, les opérations ne sont pas les mêmes qu'avec les nombres. Nous allons commencer par définir les opérateurs de base qui sont **ET**, **OU** et **NON** puis identifier leurs propriétés dans l'algèbre de Boole.

Les valeurs booléennes sont souvent utilisées en programmation notamment pour l'exécution de tests conditionnels : la fameuse instruction `if ... else`.

2 Opérations fondamentales

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.
Il existe des opérations spécifiques pour les booléens : la négation, la conjonction et la disjonction que l'on appelle aussi le NON, le ET et le OU logiques.
Puisque l'algèbre de Boole ne contient que deux éléments, pour chacune de ces opérations, on peut étudier tous les cas possibles et les regrouper dans un tableau appelé **table de vérité**.

Définition

Une **table de vérité** récapitule dans un tableau toutes les valeurs des résultats possibles en sortie d'une opération booléenne en fonction des valeurs des entrées.

2.1 L'opérateur NON (NOT) : la négation

Le NON logique renvoie l'inverse de son entrée, i.e. `Faux` si l'entrée est à l'état `Vrai`, et vice-versa.
Le $NON(A)$ peut également s'écrire $\neg A$, $\bar{A}$ ou encore $!A$.

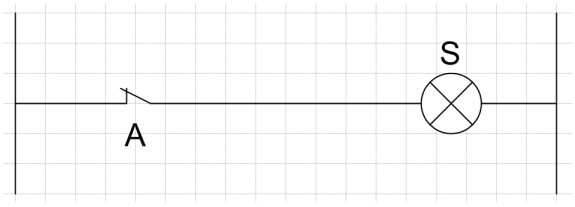
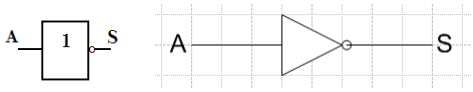


FIGURE 1 – Schéma électrique du NON

Table de vérité

Entrée	Sortie
A	$\neg A$
0	1
1	0

Symboles logiques



2.2 L'opérateur ET (AND) : la conjonction

Le ET logique renvoie l'état `Vrai` si toutes ses entrées sont à l'état `Vrai`.

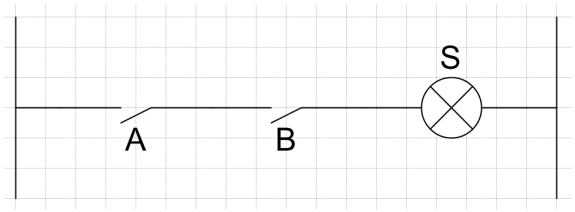


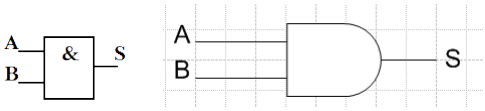
FIGURE 2 – Schéma électrique du ET

Le ET peut également s'écrire $A \wedge B$, $A \cap B$, $A \times B$ ou encore $A\&B$.

Table de vérité

Entrées		Sortie
A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Symboles logiques



2.3 L'opérateur OU (OR) : la disjonction

Le OU logique renvoie l'état `Vrai` si au moins une de ses entrées est à l'état `Vrai`.

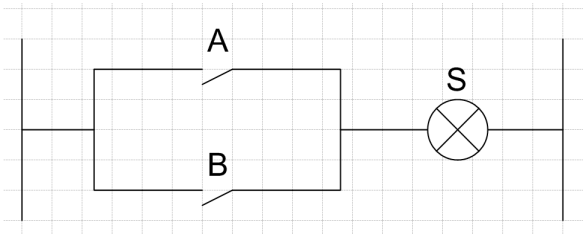


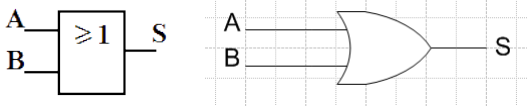
FIGURE 3 – Schéma électrique du OU

Le OU peut également s'écrire $A \vee B$, $A \cup B$, $A + B$ ou encore $A||B$.

Table de vérité

Entrées		Sortie
A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Symboles logiques



Remarques

— A partir de ces trois opérateurs fondamentaux, on peut également définir d'autres opérateurs tels que XOR, $NAND$ (i.e. $\neg ET$) ou encore le NOR (i.e. $\neg OU$).

— L'opérateur ET est prioritaire sur l'opérateur OU.

— Certains opérateurs ont plusieurs notations. Ces différentes notations dépendent du domaine d'étude (logique, électronique, algèbre...)

Définition

Une **expression booléenne** est une combinaison d'opérations élémentaires (OU, ET, NON) portant sur une ou plusieurs variables booléennes.

Exemple

Complétons la table de vérité de l'expression booléenne suivante $(A \vee B) \wedge C$.

Pour cela, il suffit de calculer les résultats de toutes les sous-expressions, en commançant par les calculs en profondeur puis en remontant. Dans cet exemple, cela revient tout d'abord à calculer les résultats de l'expression $(A \vee B)$ et enfin le résultat final.

Entrées				Sortie
A	B	C	$(A \vee B)$	$(A \vee B) \wedge C$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

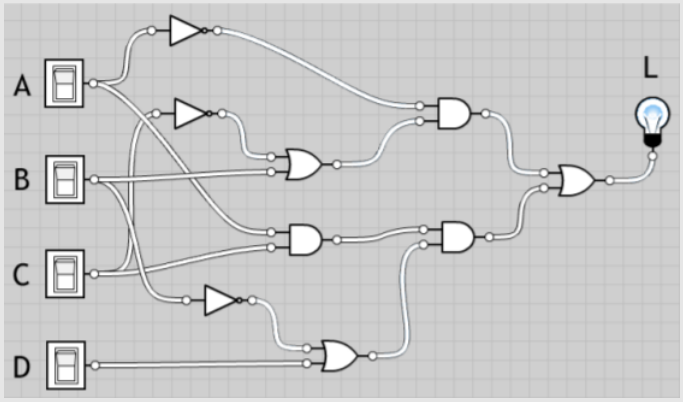
Exercice 1

Dresser les tables de vérité des expressions suivantes

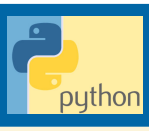
1. $(A \vee B) \wedge (A \wedge B)$
2. $(A \vee B) \vee (A \wedge B)$
3. $(\neg(A) \wedge B) \vee (A \wedge \neg(B))$
4. $(R \wedge T) \vee (S \wedge T)$
5. $(R \vee S) \wedge T$

Exercice 2

Soit le circuit ci-dessous :



1. Écrire l'expression booléenne représentant ce circuit électrique.
2. Etablir la table de vérité de ce circuit.



— Les booléens V et F se notent `True` et `False` mais aussi `1` et `0`.
— Les opérateurs sont `not` (négation), `and` (conjonction) et `or` (disjonction).
— L'opérateur `==` permet de tester si deux références contiennent les mêmes valeurs en renvoyant un booléen.

3 Les lois de l'algèbre de Boole

Propriété	**Signification**
Commutativité	$R \wedge S = S \wedge R$ $R \vee S = S \vee R$
Associativité	$R \wedge (S \wedge T) = (R \wedge S) \wedge T = R \wedge S \wedge T$ $R \vee (S \vee T) = (R \vee S) \vee T = R \vee S \vee T$
Distributivité	$R \wedge (S \vee T) = (R \wedge S) \vee (R \wedge T)$ $R \vee (S \wedge T) = (R \vee S) \wedge (R \vee T)$
Élément neutre	$R \wedge 1 = R$ $R \vee 0 = R$
Élément absorbant	$R \wedge 0 = 0$ $R \vee 1 = 1$
Involution	$\neg(\neg R) = R$
Idempotence	$R \wedge R = R$ $R \vee R = R$
Complément	$R \wedge \neg R = 0$ $R \vee \neg R = 1$
De Morgan	$\neg(R \wedge S) = \neg R \vee \neg S$ $\neg(R \vee S) = \neg R \wedge \neg S$

Ces lois peuvent facilement être démontrées à l'aide de tables de vérité.

Exemple

En utilisant ces lois, on peut montrer l'égalité suivante : $\neg(Y \wedge (X \vee \neg Y)) = (\neg X \vee \neg Y)$

Pour cela, on applique successivement les propriétés indiquées à droite

$$
\begin{array}{lcll}
\neg(Y \wedge (X \vee \neg Y)) & = & \neg Y \vee \neg(X \vee \neg Y)) & \text{De Morgan} \\
& = & \neg Y \vee (\neg X \wedge Y) & \text{De Morgan} \\
& = & (\neg Y \vee \neg X) \wedge (\neg Y \vee Y) & \text{Distributivité} \\
& = & (\neg Y \vee \neg X) \wedge 1 & \text{Complément} \\
& = & (\neg Y \vee \neg X) & \text{Élément neutre} \\
& = & (\neg X \vee \neg Y) & \text{Commutativité}
\end{array}
$$

4 Opérations composées

4.1 L'opérateur OU EXCLUSIF (XOR)

Le XOR logique renvoie l'état `Vrai` si uniquement une des deux entrées est à l'état `Vrai` (i.e. quand les deux entrées sont de valeurs opposées).

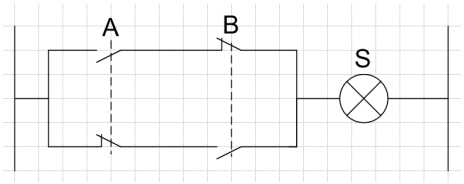


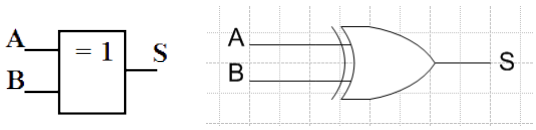
FIGURE 4 – Schéma électrique du XOR

Le XOR peut également s'écrire $A \oplus B$.

Table de vérité

Entrées		Sortie
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Symboles logiques



Différenciation des deux « OU »

Le premier OU (OR) correspond à celui utilisé dans la phrase « J'irai en bus ou un taxi ».
Le second $OU EXCLUSIF$(XOR) correspond à : « J'irai à la plage ou à la montagne ».

4.2 L'opérateur NON ET (NAND)

Le $NAND$ logique est la combinaison d'une porte ET suivi d'une porte NON. On a donc la sortie de la porte ET qui est branchée à l'entrée de la porte NON, ce qui inverse tous les résultats de la porte ET initial. Le $NAND$ renvoie l'état `Vrai` si au moins une de ses entrées est à l'état `Faux`.

Le $NAND$ peut également s'écrire $\neg(A \wedge B)$ ou encore $A \uparrow B$.

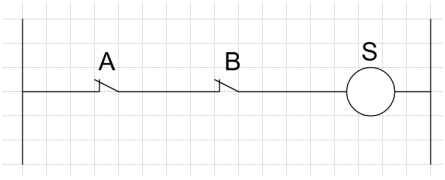
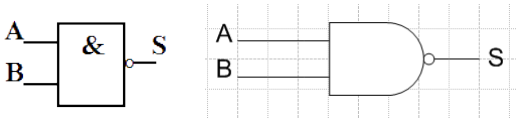


FIGURE 5 – Schéma électrique du $NAND$

Table de vérité

Entrées		Sortie
A	B	$A \uparrow B$
0	0	1
0	1	1
1	0	1
1	1	0

Symboles logiques



4.3 L'opérateur NON OU (NOR)

Le NOR logique est la combinaison d'une porte OU suivi d'une porte NON. On a donc la sortie de la porte OU qui est branchée à l'entrée de la porte NON, ce qui inverse tous les résultats de la porte OU initial. Le NOR renvoie l'état `Vrai` si toutes ses entrées sont à l'état `Faux`.

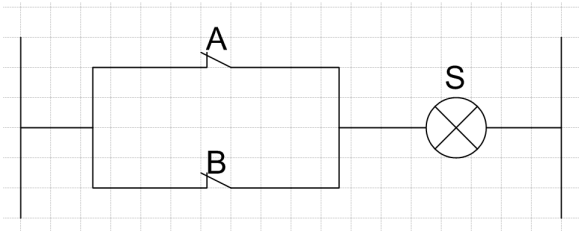


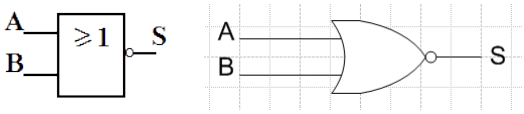
FIGURE 6 – Schéma électrique du NOR

Le NOR peut également s'écrire $\neg(A \vee B)$ ou encore $A \downarrow B$.

Table de vérité

Entrées		Sortie
A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

Symboles logiques



5 Équivalence d'expressions booléennes

Définition

Deux expressions booléennes sont dites **équivalentes** si la sortie de leurs tables de vérité est la même. L'équivalence d'expressions booléennes est notée avec le signe $=$.

Exemple

On souhaite montrer que l'expression $(R \wedge T) \vee (S \wedge T)$ est équivalente à l'expression $(R \vee S) \wedge T$.

Pour cela, commençons par dresser les tables de vérités de chacune des expressions (cf. Exercice). En comparant les deux tableaux, on remarque que les résultats (c'est-à-dire la dernière colonne) des deux tableaux sont identiques ! Donc ces deux expressions booléennes sont bien équivalentes, i.e. on a bien $(R \wedge T) \vee (S \wedge T) = (R \vee S) \wedge T$.

Exercice 3

Montrer que : $X \oplus Y = (X \wedge (\neg Y)) \vee ((\neg X) \wedge Y)$

Exercice 4

A l'aide d'une table de vérité, démontrer les lois de De Morgan :

$$\neg(A \vee B) = \neg A \wedge \neg B$$
$$\neg(A \wedge B) = \neg A \vee \neg B$$

Exercice 5

Montrer de deux manières différentes les égalités suivantes :

1. A ET B = NON(NON(A) OU NON(B))
2. A OU B = NON(NON(A) ET NON(B))
3. $(A \wedge B) \vee (\neg B \wedge C) = (A \vee \neg B) \wedge (B \vee C)$

6 Opérations bit à bit

Python dispose de nombreux opérateurs qui agissent directement sur les nombres au niveaux des bits. Ces opérateurs sont appelés **opérateurs bit à bit** (opérateur *bitwise* en anglais).

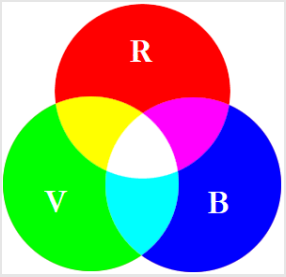
Opération posée	Avec Python
1011011 & 1010101 1010001	`>>> bin (0b1011011 & 0b1010101)` `'0b1010001'`
1011011 || 1010101 1011111	`>>> bin (0b1011011 || 0b1010101)` `'0b1011111'`
1011011 ⊕ 1010101 0001110	`>>> bin (0b1011011 ⊕ 0b1010101)` `'0b1110'`

Exercice 6

Compléter le tableau suivant en évaluant les opérations proposées :

Propriété	**Valeur**
X	01100101
Y	10101010
$X \& Y$	
$X \| \| Y$	
$X \oplus Y$	
$X \uparrow Y$	

Exercice 7 **: Additionner des couleurs**



Sur un écran, les couleurs sont créées en mélangeant du rouge, du vert et du bleu, c'est la synthèse additive des couleurs. On imagine un dispositif dans lequel trois lampes de chacune de ces couleurs sont dirigées vers le même endroit et peuvent être allumées ou éteintes.

Couleur	**R**	**V**	**B**
Noir	0	0	0
Bleu	0	0	1
Vert	0	1	0
Cyan	0	1	1
Rouge	1	0	0
Magenta	1	0	1
Jaune	1	1	0
Blanc	1	1	1

1. Justifier que l'on ne peut pas créer plus de 8 couleurs différentes dont les noms et les codes binaires sont donnés ci-contre.
2. Le complément d'une couleur est obtenu en allumant les lampes éteintes et en éteignant les lampes allumées. Déterminer les couleurs complémentaires des huit couleurs précédentes.
3. Quelle est la couleur obtenue en effectuant les opérations suivantes :
   `Bleu` $||$ `Rouge`
   `Magenta` $\&$ `Cyan`
   `Vert` $\oplus$ `Blanc`