BOOLÉENS

Introduction, notion de booléen

Historique :

Le calcul booléen, initié en 1854 par le mathématicien Georges Boole, consiste à transposer des informations logiques en informations calculables (plus rigoureusement on dit que c'est une approche *algébrique* de la logique). Le calcul booléen est étudié en tant que tel en mathématiques et est fortement utilisé en informatique et en électronique. On peut noter au passage que les notations et symboles utilisés sont parfois différents selon que l'on travaille en mathématiques, en informatique ou en électronique.

La notion d'objet booléen, quoique simple une fois comprise, peut poser difficulté en raison justement de sa simplicité qui peut rendre cette notion abstraite aux yeux de certains.

Exemples concrets

Vous avez tous déjà rencontré des affirmations qui peuvent être soit vraies soit fausses, par exemple dans des QCM ou dans des tests. Voici quelques affirmations qui sont soit vraies soit fausses :

- 7 > 3
- Le caractère "a" est présent dans la chaîne de caractères "Anaconda"
- Le caractère "a" est présent dans la chaîne de caractères "Python"
- 3 + 6 – 5 == 4
- 5 < 2

Bon, c'est simple, on est bien d'accord … (Dans l'ordre : Vrai, Vrai, Faux, Vrai, Faux).
Pour des élèves de Première on peut aller un peu plus loin (Rappel : ET signifie les deux à la fois et OU signifie que l'un, que l'autre ou les deux à la fois, c'est-à-dire au moins un) :

- (7 > 3) ET (9 < 3)
- (4 > -5) ET ("a" est présent dans la chaîne de caractères "Chihuahua")
- ("abcdef" == "xyz") OU (7 == 9 – 2)
- NON (6 == 5)
- (2 + 5 == 92) OU (6 == 7)

Cette fois-ci vous avez dû trouver 3 affirmations vraies et deux affirmations fausses.

Remarque : *Nous n'avons ici pas donné d'exemple tel que "Un chien possède quatre pattes" car les affirmations issues du monde réel sont toujours sujettes à interprétation. Ici, beaucoup diraient que cette affirmation est vraie, mais un pinailleur pourrait dire que certains chiens amputés n'ont que trois pattes donc que l'affirmation est fausse. C'est pourquoi nous nous restreindrons à des exemples inattaquables d'un point de vue logique.*

Notion de booléen

Amené à étudier la logique, Georges Boole a eu deux idées très simples qui ont permis de mathématiser la logique :
Idée 1 : On ne manipule finalement que deux objets qui sont soit VRAI soit FAUX.
Idée 2 : ET, OU et NON agissent sur VRAI et FAUX *un peu comme* $+$, $-$, $\times$ et $\div$ agissent sur les nombres.

Les objets VRAI et FAUX sont appelés des *booléens*.
Les ET, OU et NON sont appelés des *opérateurs booléens* (ou opérations booléennes).
Enfin, ce que nous avons appelé plus haut des "affirmations" pouvant être vraies ou fausses, seront appelées des *expressions booléennes*.

Pour ceux qui trouvent qu'un booléen c'est abstrait, il suffit de se dire :
Un booléen c'est comme le résultat de (3 < 5) ou de ("h" est présent dans "Ahhh") … c'est soit VRAI soit FAUX …

I Valeurs booléennes et opérateurs booléens fondamentaux

1) Booléens

On rencontre essentiellement deux notations pour les booléens :

- Vrai , Faux
- 1 , 0

Le programme de NSI mentionne explicitement les notations 1 , 0. Ce sont donc ces notations que nous utiliserons dans la suite du cours.

Les booléens sont donc des objets abstraits comme n'importe quel objet mathématique. Néanmoins ils correspondent à deux notions très concrètes :

- La première notion est celle de la valeur Vrai , Faux d'une affirmation (ou plutôt la valeur d'une *expression booléenne*) comme nous l'avons vu en introduction. C'est la notion qui correspond à la définition même d'un booléen.

- La seconde notion est celle d'assimiler les deux valeurs possibles des booléens aux deux valeurs possibles d'un bit. Cette *analogie* entre *booléen* et *bit* est évidente avec la notation 1 / 0. D'ailleurs en électronique on a le droit de parler, à propos de deux *bits* a et b, de (a ET b) ainsi que de (a OU b).

Remarque : *Pour ceux qui ne comprendraient pas la différence entre un booléen et un bit, dites-vous que le booléen est un objet abstrait qui n'existe qu'en pensée, comme n'importe quel objet mathématique. En revanche un bit est bien réel : il correspond à une marque de gravure sur un CD ou un DVD, ou bien il correspond à un signal électronique ou optique ou encore il correspond à une charge électronique dans une mémoire RAM.*

Exercices 1 et 2 en Annexe

2) Opérateurs booléens fondamentaux : and, or, not.

Le programme de NSI mentionne les opérateurs booléens fondamentaux en notation anglaise. Bien entendu and, or, not correspondent à et, ou, non.
On remarque que les opérateurs and ainsi que or agissent sur deux booléens alors que non agit sur un seul booléen (bien que ce soit hors-programme, on parle d'opérateurs binaire ou d'opérateur unaire).
Si on souhaite définir ces opérateurs on obtient :

- (A and B) = 1 si, et seulement si, on a ***à la fois*** les deux booléens A et B qui sont égaux à 1.
- (A or B) = 1 si, et seulement si, on a ***au moins un*** des deux booléens A ou B qui est égal à 1.
- (not A) = 1 si, et seulement si, on a A = 0.

Pour avoir une vision complémentaire et non ambigue de ces opérateurs, on peut facilement dresser les *tables de vérité* de ces opérateurs qui indiquent tous les cas possibles :

A	B	A and B
0	0	
0	1	
1	0	
1	1	

A	B	A or B
0	0	
0	1	
1	0	
1	1	

A	not A
0	
1	

Exercice 3 en Annexe

3) Euh… pourquoi dit-on que ce qu'on vient de faire avec or, and, not c'est du calcul ?

Regardons les tables de multiplication de $A \times B$ et de $A + B$ où on suppose que les seuls nombres qui existent sont 0 et 1 (du binaire sur 1 seul bit)

A	B	$A \times B$
0	0	
0	1	
1	0	
1	1	

A	B	$A + B$
0	0	
0	1	
1	0	
1	1	

On remarque que la table de multiplication des *nombres* de 0 à 1 est la même que la table de vérité de l'opérateur *booléen* and.
Et que la table d'addition des *nombres* de 0 à 1 *ressemble beaucoup* à la table de vérité de l'opérateur *booléen* or (*).

Cette analogie entre (multiplication , addition sur les nombres) et (opérateur and , opérateur or sur les booléens) est hors-programme mais elle est utile pour deux raisons :

- Pour votre culture générale. Parfois, en électronique, on remplace la notation AND par la notation . et on remplace la notation OR par la notation $+$. Désormais vous saurez pourquoi.

- Pour retenir les règles de priorité lorsqu'on travaille avec des booléens. Georges Boole a bien fait son travail : par analogie avec $\times$ et $+$, il a été décidé que le AND serait prioritaire sur le OR. Soit on mémorise la règle telle quelle, soit on se rappelle juste de l'analogie, il est alors naturel que AND soit prioritaire sur le OR car la multiplication est prioritaire sur l'addition.

(*) : *ressemble beaucoup* car avec des nombres entiers on a l'addition qui nous donne : 1 + 1 = 2 alors qu'avec des booléens on a : 1 or 1 = 1… donc on a une légère différence….

II Expressions Booléennes

1) Propriétés des opérateurs

Rappelons quelques évidences en calcul sur les nombres :
Associativité : $3 + 4 + 5 = (3 + 4) + 5$ ou bien : $2 \times (4 \times 3) = (2 \times 4) \times 3$
Commutativité : $3 + 4 = 4 + 3$ ou bien $3 \times 4 = 4 \times 3$
Distributivité : $3 \times (4 + 2) = 3 \times 4 + 3 \times 2$
Priorité : $3 + 4 \times 5 = 3 + (4 \times 5)$

Et bien c'est la même chose pour les booléens en considérant que and correspond à $\times$ et que or correspond à $+$ (relire I.3 si besoin). Ainsi, si A, B et C sont trois booléens on a :

- Associativité : A and B and C = (A and B) and C = A and (B and C)
  A or B or C = (A or B) or C = A or (B or C)
- Commutativité : A and B = B and A
  A or B = B or A
- Distributivité : A and (B or C) = (A and B) or (A and C)
- Priorité : A and B or C = (A and B) or C
  A or B and C = A or (B and C)

On a presque fini pour les propriétés des opérateurs. En effet il ne vous aura pas échappé qu'on a oublié de parler de quelque chose : la priorité de l'opérateur not. Retenez que, par convention, il est prioritaire sur les deux autres.

- Priorité de not : not A and B = (not A) and B
  not A or B = (not A) or B

Exercice 4 en Annexe

2) Dresser la table de vérité d'une expression booléenne

Pour obtenir la table de vérité d'une expression booléenne un peu longue, on peut s'aider de colonnes intermédiaires, exactement comme dans l'exercice précédent.
Voici quelques exemples à effectuer.
Le premier a été commencé afin de vous montrer la voie.
Pour les suivants, c'est à vous de tout faire.

Remarque : *Avant de faire les exemples 2, 3 et 4 il est nécessaire d'avoir remarqué sur l'exemple 1 comment sont remplies les colonnes A, B, C afin que toutes les combinaisons possibles de A, B, C aient bien été envisagées.*

Exemple 1 :
Soit A, B, C trois variables booléennes. Dresser la table de vérité de :
A or not B and C
Début de réponse :
Il y a 3 variables booléennes A, B et C à prendre en compte, soit 2^3 lignes à utiliser.
Pour plus de lisibilité, compte-tenu de la priorité des opérateurs, nous obtenons avec des parenthèses : **A or ((not B) and C)**
Nous utiliserons des colonnes intermédiaires pour (not B) puis pour ((not B) and C) [*on peut, si on est à l'aise, utiliser moins de colonnes intermédiaires*] .

A	B	C	not B	(not B) and C	A or ((not B) and C)
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0		
0	1	1	0		
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Exercice 5 : Voir annexe
Soit A, B, C trois variables booléennes. Dresser la table de vérité de :
not A and B and C

Exercice 6 : Voir Annexe
Soit A, B, C trois variables booléennes. Dresser la table de vérité de :
A and B or A and C

Exercice 7 : Voir Annexe
Soit A, B, C trois variables booléennes. Dresser la table de vérité de :
A and (B or C)
Comparer avec l'exemple précédent.

3) Le cas du XOR

L'opérateur booléen xor est le e**X**clusive **OR**, en français le ou exclusif. On peut le comprendre de deux façons différentes (qui sont équivalentes).
On peut ainsi voir que A xor B est vrai lorsqu'un seul - exclusivement un seul - des deux booléens A ou B est vrai (A xor B est faux sinon).
On peut aussi voir que A xor B est vrai lorsque A et B sont différents (A xor B est faux sinon).
Pour lever toute ambiguité, sa table de vérité est donnée ci-contre.

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	**0**

La question qui se pose est pourquoi le xor est-il dans ce cours ? Tout simplement parce que le xor est en pratique souvent utilisé. Voici quelques applications (pour votre culture générale en informatique) :

- En électronique, cela permet de faciliter la lecture de certains plans de montage. Sans (A xor B) on serait parfois obligé d'utiliser ((not A and B) or (A and not B)) ce qui est moins élégant (voir Exercice)
- Chiffrement par flot :
  Cette méthode de chiffrement symétrique par clef aléatoire jetable, a notamment été utilisée pour le célèbre téléphone rouge reliant le Kremlin à la Maison-Blanche durant la guerre froide (les clefs de chiffrement transitaient alors par valise diplomatique). C'est la seule méthode de chiffrement parfaitement sûre au sens défini par Shannon en 1949.
  Plus généralement, toutes les méthodes de chiffrement symétrique utilisent l'opérateur xor.
- Pour tester si deux bits sont différents.
- Montage "Va-et-Vient" en électricité (dans une pièce avec deux interrupteurs de lumière).
- En langage assembleur, sur certains processeurs, l'opérateur xor peut permettre de mettre des registres à zéro (en utilisant le fait que, quelle que soit la valeur d'un *bit* A, on a : A xor A = 0).

Exercice 8 en Annexe

4) Exercice : les lois de De Morgan

Exercice 9 en Annexe

Les lois de De Morgan : not (A and B) = (not A) or (not B) ainsi que not (A or B) = (not A) and (not B) *ne sont pas explicitement au programme. Néanmoins elles font partie de la culture générale concernant les opérateurs booléens. Elles se traduisent facilement en français car elles traduisent deux idées de bons sens :*
- Si je ne veux pas (deux choses A et B à la fois) alors il faut qu'il me manque au moins l'une d'elles.
- Si je ne veux pas avoir (une chose A ou une chose B) alors il faut que je n'aie ni l'une ni l'autre

III Applications : addition binaire et circuits combinatoires

1) Addition Binaire : Exemples

Ceux qui ont oublié les conversions de base 10 vers base 2 iront utilement relire leur cours.

Exercice de Cours 10 :
a) Déterminer l'écriture de 183_{10} en base 2

b) Soit $n = 01101011_2$ un entier représenté en base 2. Déterminer l'écriture de n en base 10.

Enfin ceux qui ont oublié comment on fait des additions en base 10 à la main iront utilement relire leur cours de l'école primaire.

Exercice de Cours 11 :
Calculer les additions suivantes (en base 10) :

$\begin{array}{r} 45183 \\ +\ 24145 \\ \hline \end{array}$	$\begin{array}{r} 75874 \\ +\ 67598 \\ \hline \end{array}$	$\begin{array}{r} 22222 \\ +\ 87654 \\ \hline \end{array}$	$\begin{array}{r} 23456 \\ +\ 78910 \\ \hline \end{array}$

Faisons une synthèse des cas envisageables en base 10 lorsqu'on pose l'addition. Par exemple si on a déjà effectué l'addition des unités, des dizaines et des centaines on se trouvera dans un des quatre cas suivants pour l'addition des milliers :

$\begin{array}{cccccc} & X & 5 & X & X & X \\ + & X & 4 & X & X & X \\ \hline & & & X & X & X \end{array}$	$\begin{array}{cccccc} & X & 5 & X & X & X \\ + & X & 7 & X & X & X \\ \hline & & & X & X & X \end{array}$	$\begin{array}{cccccc} & & 1 & & & \\ & X & 2 & X & X & X \\ + & X & 4 & X & X & X \\ \hline & & & X & X & X \end{array}$	$\begin{array}{cccccc} & & 1 & & & \\ & X & 5 & X & X & X \\ + & X & 8 & X & X & X \\ \hline & & & X & X & X \end{array}$

De gauche à droite :

-5 milliers plus 4 milliers soit 9 milliers
-5 milliers plus 7 milliers soit 12 milliers soit 1 fois dix mille (en retenue) plus 2 milliers.
-1 millier de retenue plus $(2+4)$ milliers soit 7 milliers.
-1 millier de retenue plus $(5\ +\ 8)$ milliers soit 14 milliers soit 1 fois dix mille (en retenue) plus 4 milliers.

Adaptons cela à la base 2, les colonnes seront celles des $2^0 = unités$, des $2^1 = 'deuzaines'$, des $2^2 = 'quatraines'$, des $2^3 = 'huitaines'$ et des $2^4 = 'seizaines'$ (de droite à gauche).

La différence sera que si l'on a 1 paquet de 8 plus 1 paquet de 8 … on arrivera directement à 1 paquet de 16 (en retenue). Et que si l'on a 1 paquet de 8 de retenue plus $(1+1)$ paquet de 8 on arrivera à 3 paquets de 8 soit 1 paquet de 16 (en retenue) plus 1 paquet de 8.

Si on veut faire plus sérieux : $2^p + 2^p = 2 \times 2^p = 2^{p+1}$ et $2^p + 2^p + 2^p = 2^{p+1} + 2^p$. Cela donne :

$\begin{array}{cccccc} & X & 1 & X & X & X \\ + & X & 0 & X & X & X \\ \hline & & & X & X & X \end{array}$	$\begin{array}{cccccc} & X & 1 & X & X & X \\ + & X & 1 & X & X & X \\ \hline & & & X & X & X \end{array}$	$\begin{array}{cccccc} & X & 0 & X & X & X \\ + & X & 0 & X & X & X \\ \hline & & & 0 & X & X & X \end{array}$	$\begin{array}{cccccc} & & 1 & & & \\ & X & 1 & X & X & X \\ + & X & 1 & X & X & X \\ \hline & & & X & X & X \end{array}$

Exercice 12 en Annexe

2) Addition Binaire : formalisation grâce à une table de vérité

Afin de fixer les notations, considérons l'addition binaire des deux nombres $A = 10111_2$ et $B = 01010_2$ posée ci-contre.
Nous numéroterons les colonnes en partant de la droite. Ainsi la colonne 0 correspond aux $2^0 = unités$ et la colonne 1 correspond aux $2^1 = 'deuzaines'$ …

1	1	1	1		
	1	0	1	1	1
+	0	1	0	1	0
1	0	0	0	0	1

Comme mentionné en I.1) nous assimilerons les bits à des booléens.

Nous appelerons :
- r_n la retenue de la colonne n,
- a_n le bit du nombre A de la colonne n,
- b_n le bit du nombre B de la colonne n,
- s_n le bit de la somme de A et B de la colonne n.

Dit plus simplement, (r_n) correpond à la ligne des retenues, (a_n) à la ligne du nombre A, (b_n) à la ligne du nombre B et (s_n) à la ligne de la somme de A et B.

…	r_{n+1}	r_n	…	r_1	r_0
…	a_{n+1}	a_n	…	a_1	a_0
…	b_{n+1}	b_n	…	b_1	b_0
…	s_{n+1}	s_n	…	s_1	s_0

On remarque qu'avec ces notations $r_0 = 0$ (il n'y a jamais de retenue *avant* de commencer l'addition).

Par ailleurs, on remarque que lorsqu'on effectue "l'addition de $a_n + b_n$ ":
- il ne faut pas oublier de prendre en compte la retenue r_n, c'est-à-dire que l'on fait en réalité $a_n + b_n + r_n$
- le résultat va impacter à la fois s_n mais aussi r_{n+1}.

Exercice de cours 13 :

1) Compléter la table de vérité ci-contre :

r_n	a_n	b_n	s_n	r_{n+1}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

2) Soit $nb\_vrai(a, b, r)$ la fonction qui, à tout triplet de booléens (a, b, r), associe le nombre de 1 parmi a, b et r. Par exemple $nb\_vrai(1, 0, 1) = 2$ alors que $nb\_vrai(1, 1, 1) = 3$.

Donner les valeurs de s_n et $r_{(n+1)}$ lorsque :
. $nb\_vrai(a_n, b_n, r_n) = 0$ alors $s_n =$ et $r_{n+1} =$
. $nb\_vrai(a_n, b_n, r_n) = 1$ alors $s_n =$ et $r_{n+1} =$
. $nb\_vrai(a_n, b_n, r_n) = 2$ alors $s_n =$ et $r_{n+1} =$
. $nb\_vrai(a_n, b_n, r_n) = 3$ alors $s_n =$ et $r_{n+1} =$

3) Compléter l'expression booléenne suivante :

$$s_n = (nb\_vrai(a_n, b_n, r_n) == \quad) \; ou \; (nb\_vrai(a_n, b_n, r_n) == \quad)$$

4) Compléter l'expression booléenne suivante :

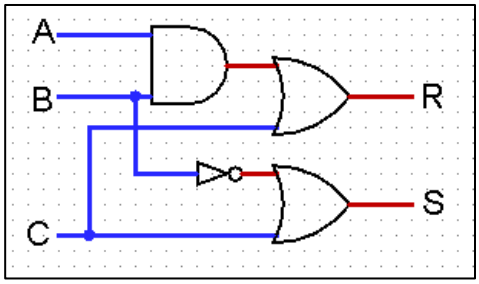
$$r_{n+1} = (nb\_vrai(a_n, b_n, r_n) == \quad) \; ou \; (nb\_vrai\,(a_n, b_n, r_n) == \quad)$$

5) Relier les expressions booléennes qui sont équivalentes entre elles par des traits :

$nb\_vrai(a, b, r) = 0$	■	■	$a \; and \; b \; and \; c$
$nb\_vrai(a, b, r) = 1$	■	■	$(a \; and \; not \; b \; and \; not \; c)$ $or \; (not \; a \; and \; b \; and \; not \; c)$ $or \; (not \; a \; and \; not \; b \; and \; c)$
$nb\_vrai(a, b, r) = 2$	■	■	$not \; a \; and \; not \; b \; and \; not \; c$
$nb\_vrai(a, b, r) = 3$	■	■	$(a \; and \; b \; and \; not \; c)$ $or \; (not \; a \; and \; b \; and \; c)$ $or \; (a \; and \; not \; b \; and \; c)$

3) Circuits combinatoires

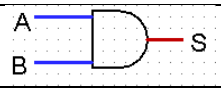
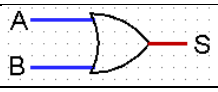
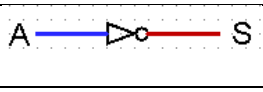
En électronique, on utilise des assemblages de *portes logiques* pour effectuer des *circuits combinatoires*.
Ci-contre, voici un exemple de circuit combinatoire :



Vous avez pu voir comment il fonctionne grâce à la présentation du logiciel Logisim qui vous a été faite au vidéo-projecteur.

Ce circuit :

- a trois entrées booléennes A, B et C,
- a deux sorties booléennes R et S.
- est composé de portes logiques dont voici les significations

$S = A\ and\ B$	$S = A\ or\ B$	$S = not\ A$
		

Exercice de cours 14 :
En utilisant le circuit combinatoire ci-dessus :
1) Déterminer l'expression booléenne de R en fonction de A, B et C.

2) Déterminer l'expression booléenne de S en fonction de A, B et C.

Exercice 15 en Annexe

Exercice 16 en Annexe

Remarque : *vous n'avez pas à apprendre par cœur les formes des portes logiques and, or et not. Si on vous interroge dessus, nous vous rappelerons à quoi telle ou telle porte correspond.*
Par ailleurs le logiciel Logisim est mis à votre disposition : vous pouvez construire quelques circuits combinatoires en plus des exercices pour appréhender davantage cette notion.