Math93.com

TD - NSI
Variables booléennes

Objectif

Dresser la table d'une expression booléenne.

1 Variables booléennes

Définition 1

Un **booléen** est une variable informatique qui ne peut prendre que deux valeurs : `True` ou `False`.
Un booléen est représenté en machine par un bit, qui vaut :

— 1 pour la valeur `True` ;

— 0 pour la valeur `False`.

Définition 2

Il existe trois opérations élémentaires sur les booléens : **disjonction**, **conjonction** et **négation**.

— **OU** : La **disjonction de A et B** est notée : $\boxed{A \texttt{ or } B}$, $\boxed{A+B}$, $\boxed{A \bigvee B}$.
A `or` B vaut `True` si et seulement si A vaut `True` ou B vaut `True`.

A	B	$A \texttt{ or } B = A+B$
0	0	0
0	1	1
1	0	1
1	1	1

— **ET** : La **conjonction de A et B** est notée : $\boxed{A \texttt{ and } B}$, $\boxed{A \cdot B}$, $\boxed{A \bigwedge B}$.
A `and` B vaut `True` si et seulement si A vaut `True` et B vaut `True`.

A	B	$A \texttt{ and } B = A.B$
0	0	0
0	1	0
1	0	0
1	1	1

— **La négation de A** est notée $\boxed{\texttt{not } A}$ ou encore $\boxed{\overline{A}}$.
$\overline{A}$ vaut `True` si et seulement si A vaut `False`.

A	$\texttt{not } A = \overline{A}$
0	1
1	0

Définition 3 (Expression booléenne)

Une **expression booléenne** est une combinaison d'opérations élémentaires (`or`, `and`, `not`) portant sur une ou plusieurs variables booléennes.

Exercice 1.1

Compléter les égalités suivantes.

$A \texttt{ or } A = \ldots \qquad A \texttt{ and } A = \ldots \qquad \texttt{not}\,(\texttt{not } A) = \ldots \qquad A \texttt{ or }(\texttt{not } A) = \ldots \qquad A \texttt{ and }(\texttt{not } A) = \ldots$

Exercice 1.2

Formules de Morgan

1. Compléter les tables de vérité ci-dessous.

A	B	$A \texttt{ or } B$	$\overline{A \texttt{ or } B}$
0	0		
0	1		
1	0		
1	1		

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \texttt{ and } \overline{B}$
0	0			
0	1			
1	0			
1	1			

2. En déduire que $\overline{A \texttt{ or } B} = \overline{A} \texttt{ and } \overline{B}$.
3. Montrer de même que $\overline{A \texttt{ and } B} = \overline{A} \texttt{ or } \overline{B}$.

Exercice 1.3

1. Dresser la table de vérité de l'expression $S = (A \texttt{ or } B) \texttt{ and } (\overline{A} \texttt{ or } B)$.

A	B	$A \texttt{ or } B$	$\overline{A}$	$(\overline{A} \texttt{ or } B)$	S
0	0				
0	1				
1	0				
1	1				

2. Quelle égalité booléenne peut en déduire ?

Exercice 1.4

Dresser la table de vérité de l'expression $S = (A$ `and` $B)$ `or` $(A$ `and not` $C)$ `or` $($`not` B `and` $C)$.

A	B	C	A `and` B	A `and not` C	`not` B `and` C	S
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Exercice 1.5

On donne ci-dessous les tables de vérité de différentes expressions booléennes U, V et W.
Retrouver les expressions de U, V et W en fonction de A et B.

A	B	U
0	0	0
0	1	0
1	0	1
1	1	0

A	B	V
0	0	1
0	1	0
1	0	0
1	1	0

A	B	W
0	0	1
0	1	0
1	0	1
1	1	0

2 Opérations bit à bit

Bit à bit

On peut généraliser les opérations logiques `and` et `or` à des chaînes de bits ce que python peut faire facilement. Sous python :

— le `and` se note &

— le `or` se note |

— **Bit à bit : `and` se note &**

$$\begin{array}{cccccccc} & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ \& & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

```python
# Dans la console PYTHON
>>> bin(0b1011011&0b1010101)
'0b1010001'
```

— **Bit à bit : `or` se note |**

$$\begin{array}{cccccccc} & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ | & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

```python
# Dans la console PYTHON
>>> bin(0b1011011|0b1010101)
'0b1011111'
```

Exercice 2.6

Calculer à la main puis avec python :

— 1111001 & 1110100

— 1111001 | 1110100

3 Obtention des formes normales

Propriété 1

1. **Forme normale conjonctive (FNC)** : Une fonction booléenne peut toujours s'exprimer sous forme d'une conjonction (`and`) de formes disjonctives (`or`) . Par exemple :

$$f(a,b,c) = (a+b+c).(\overline{a}+b+c).(\overline{a}+b+\overline{c})$$

2. **Forme normale disjonctive (FND)** : Une fonction booléenne peut toujours s'exprimer sous forme d'une disjonction (`or`) de formes conjonctives (`and`) . Par exemple :

$$g(a,b,c) = \overline{a}bc + a\overline{b}c$$

Méthode

Il existe cependant un moyen d'obtenir les FND ou FNC facilement en travaillant sur la table de vérité. On s'intéresse aux différents cas où la fonction retourne 0 ou 1.

a	b	c	$f(a,b,c)$
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

— **Obtenir la FND**

Pour la FND, on s'intéresse aux cas pour lesquels la fonction vaut 1 et on écrit la conjonction des littéraux (les *midterms*) correspondante :

1. $a = b = 1$ et $c = 0$.
   Ce premier cas donne la conjonction $a.b.\overline{c}$, et la table de vérité nous dis que celle-ci donne 1 lorsque $a = 1$, $b = 1$ et $c = 0$, or $1.1.\overline{0} = 1$.
2. On a aussi $a = 1$ et $b = 0$, soit $c = 1$ et on a la conjonction $a.\overline{b}.c$;
3. On a aussi $a = 1$ et $b = 0$, soit $c = 0$ et on a la conjonction $a.\overline{b}.\overline{c}$

a	b	c	$f(a,b,c)$	midterms
1	1	1	0	
1	1	0	1	$a.b.\overline{c}$
1	0	1	1	$a.\overline{b}.c$
1	0	0	1	$a.\overline{b}.\overline{c}$
0	1	1	0	
0	1	0	0	
0	0	1	0	
0	0	0	0	

Pour avoir la (FND) on prend la disjonction des conjonctions :

$$\boxed{f(a,b,c) = a.b.\overline{c} + a.\overline{b}.c + a.\overline{b}.\overline{c}}$$

— **Obtenir la FNC**
On s'intéresse aux cas où la fonction vaut 0 (qu'on appelle des *maxterms*) , et de la même manière, on note les différentes conjonctions correspondantes.

a	b	c	$f(a,b,c)$	maxterms
1	1	1	0	$\left(\overline{a}+\overline{b}+\overline{c}\right)$
1	1	0	1	
1	0	1	1	
1	0	0	1	
0	1	1	0	$\left(a+\overline{b}+\overline{c}\right)$
0	1	0	0	$\left(a+\overline{b}+c\right)$
0	0	1	0	$\left(a+b+\overline{c}\right)$
0	0	0	0	$(a+b+c)$

On obtient alors

$$\boxed{f(a,b,c)=\left(\overline{a}+\overline{b}+\overline{c}\right).\left(a+\overline{b}+\overline{c}\right).\left(a+\overline{b}+c\right).\left(a+b+\overline{c}\right).(a+b+c)}$$

Remarque

$$\overline{f(a,b,c)}=a.b.c+\overline{a}.b.c+\overline{a}.b.\overline{c}+\overline{a}.\overline{b}.c+\overline{a}.\overline{b}.\overline{c}$$

Si on en prend la conjonction, la propriété assure que $\overline{\overline{f(a,b,c)}}=f(a,b,c)$:

$$\begin{aligned}\overline{\overline{f(a,b,c)}}=f(a,b,c)&=\overline{a.b.c+\overline{a}.b.c+\overline{a}.b.\overline{c}+\overline{a}.\overline{b}.c+\overline{a}.\overline{b}.\overline{c}}\\&=\left(\overline{a}+\overline{b}+\overline{c}\right).\left(a+\overline{b}+\overline{c}\right).\left(a+\overline{b}+c\right).\left(a+b+\overline{c}\right).(a+b+c)\end{aligned}$$

$$\boxed{f(a,b,c)=\left(\overline{a}+\overline{b}+\overline{c}\right).\left(a+\overline{b}+\overline{c}\right).\left(a+\overline{b}+c\right).\left(a+b+\overline{c}\right).(a+b+c)}$$

Exercice 3.7

L'opération « ou exclusif », noté $\texttt{xor}$, est défini par $A \texttt{ xor } B = (A \texttt{ or } B) \texttt{ and } \overline{A \texttt{ and } B}$.

1. Dresser la table de vérité de l'expression $A \texttt{ xor } B$.
2. En déduire la forme normale disjonctive de $A \texttt{ xor } B$.

A	B	$A \texttt{ or } B$	$A \texttt{ and } B$	$\overline{A \texttt{ and } B}$	$A \texttt{ xor } B = (A \texttt{ or } B) \texttt{ and } \overline{A \texttt{ and } B}$
0	0	0			
0	1	1			
1	0	1			
1	1	1			

Exercice 3.8

On donne ci-dessous les tables de vérité de différentes expressions booléennes U, V et W.
Retrouver les expressions de U, V et W en fonction de A et B et C.

A	B	C	U
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A	B	C	V
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

A	B	C	W
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Exercice 3.9

Question 1

Si A et B sont des variables booléennes, quelle est l'expression booléenne équivalente à $(\texttt{not } A) \texttt{ or } B$?

— $(A \texttt{ and } B) \texttt{ or } (\texttt{not } A \texttt{ and } B) \texttt{ or } (\texttt{not } A \texttt{ and not } B)$

— $(A \texttt{ and } B) \texttt{ or } (\texttt{not } A \texttt{ and } B)$

— $(\texttt{not } A \texttt{ and } B) \texttt{ or } (\texttt{not } A \texttt{ and not } B)$

— $(A \texttt{ and } B) \texttt{ or } (\texttt{not } A \texttt{ and not } B)$