Numérique et sciences informatiques
Classe de première

B. Représentation de l'information
3. Représentation des expressions logiques

Pour qu'un ordinateur puisse exécuter un programme informatique, il faut qu'il soit capable d'effectuer des opérations logiques. Qu'est-ce qu'une opération logique ? Considérons une affirmation complexe constituée à partir de plusieurs affirmations plus élémentaires. Par exemple « Annie ira au cinéma s'il pleut et si elle s'ennuie. ». Pour savoir si Annie ira au cinéma , il me suffit de déterminer si chacune des deux affirmations élémentaires « Il pleut » et « Annie s'ennuie » est vraie ou fausse puis d'effectuer une sorte de « calcul logique » qui renverra la valeur Vrai ou Faux. Une des fonctions les plus élémentaires d'un ordinateur est d'être capable d'effectuer ce type de calcul logique. Avant d'évoquer comment ceci peut être réalisé concrètement, il faut d'abord introduire des notions logiques élémentaires.

I Notions fondamentales de logique

I.1 Les principaux opérateurs logiques

Considérons deux propositions P et Q dont la valeur de vérité est soit « vrai » (V) soit « faux » (F). Nous allons définir différents opérateurs logiques fondamentaux qui déterminent une valeur de sortie V ou F en fonction des valeurs attribuées à P et à Q.
* L'opérateur **not(...)** opère sur une seule proposition. On dit que c'est un **opérateur unaire**.
* Les opérateurs **and**, **or** et **xor** opèrent sur deux propositions. On dit que ce sont des **opérateurs binaires**.
Les opérateurs logiques sont aussi appelés opérateurs booléens en hommage au mathématicien et logicien George Boole qui a élaboré un calcul algébrique pour représenter les opérations logiques.

• L'opérateur **not(...)**

not(P) est vrai si et seulement si P est faux. Le fonctionnement de l'opérateur **not(...)** peut donc être représenté dans la table de vérité suivante :

P	**not(P)**
F	V
V	F

• L'opérateur **and**

P **and** Q est vrai si et seulement si P est vrai et Q est vrai. La table de vérité de l'opérateur **and** est donc :

P	**Q**	**P and Q**
F	F	F
F	V	F
V	F	F
V	V	V

• L'opérateur **or**

P **or** Q est vrai si et seulement si P est vrai ou Q est vrai. La table de vérité de l'opérateur **or** est donc :

P	**Q**	**P or Q**
F	F	F
F	V	V
V	F	V
V	V	V

• L'opérateur **xor** (ou exclusif)

P **xor** Q est vrai si et seulement si P est vrai et Q est faux ou si P est faux et Q est vrai. La table de vérité de l'opérateur **xor** est donc :

P	**Q**	**P xor Q**
F	F	F
F	V	V
V	F	V
V	V	F

I.2 Construire la table de vérité d'une expression logique

On peut construire n'importe quelle fonction logique à partir des opérateurs not, and et or. La table de vérité d'une telle fonction peut être retrouvée grâce aux tables de vérité des opérateurs qui la composent. Considérons l'exemple suivant :

$$f(P,Q,R) = (P\ and\ Q)\ or\ (P\ and\ not(R))$$

On a la table de vérité suivante :

P	Q	R	P **and** Q	**not**(R)	P **and not**(R)	$(P$ **and** $Q)$ **or** $(P$ **and not**$(R))$
F	F	F	F	V	F	F
F	F	V	F	F	F	F
F	V	F	F	V	F	F
F	V	V	F	F	F	F
V	F	F	F	V	V	V
V	F	V	F	F	F	F
V	V	F	V	V	V	V
V	V	V	V	F	V	V

I.3 Quelques propriétés des opérateurs logiques not, and et or

A l'aide des tables de vérité, on peut constater que les égalités suivantes sont vraies dans tous les cas : ce sont des identités.

• L'opérateur **not** est involutif :

$$\mathbf{not}\big(\mathbf{not}(P)\big) = P$$

• Les opérateurs **and** et **or** sont commutatifs :

$$P\ \mathbf{or}\ Q = Q\ \mathbf{or}\ P$$
$$P\ \mathbf{and}\ Q = Q\ \mathbf{and}\ P$$

• Les opérateurs **and** et **or** sont associatifs :

$$P\ \mathbf{or}\ (Q\ \mathbf{or}\ R) = (P\ \mathbf{or}\ Q)\ \mathbf{or}\ R$$

• L'opérateur **and** est distributif par rapport à l'opérateur **or :**

$$P \textbf{ and } (Q \textbf{ or } R) = (P \textbf{ and } Q) \textbf{ or } (P \textbf{ and } R)$$

• L'opérateur **or** est distributif par rapport à l'opérateur **and :**

$$P \textbf{ and } (Q \textbf{ or } R) = (P \textbf{ and } Q) \; or \; (P \textbf{ and } R)$$

• Lois de De Morgan :

$$\textbf{not}(P \textbf{ and } Q) = \big(\textbf{not}(P)\big) \textbf{ or } (\textbf{not}(Q))$$
$$\textbf{not}(P \textbf{ or } Q) = \big(\textbf{not}(P)\big) \textbf{ and } (\textbf{not}(Q))$$

I.4 Le caractère « séquentiel » des opérateurs and et or

Dans certains cas, il n'est pas nécessaire d'évaluer la valeur de vérité des deux propositions P et Q pour déterminer la valeur de vérité de P and Q ou de P or Q.

• Si P est faux, alors $P \textbf{ and } Q$ est faux quelle que soit la valeur de vérité de Q. Ainsi, si on constate que P est faux, il n'est pas utile d'évaluer la valeur de vérité de Q pour en déduire la valeur de $P \textbf{ and } Q$.

• De même, si P est vrai alors $P \textbf{ or } Q$ est vrai quelle que soit la valeur de vérité de Q. Dans ce cas non plus, on n'aura pas besoin d'évaluer la valeur de vérité de Q pour en déduire la valeur de $P \textbf{ or } Q$.

On dit que ces deux opérateurs **and** et **or** sont « séquentiels ».

II Matérialisation électronique des opérations logiques et arithmétiques

Comment réalise-t-on des opérations logiques avec un ordinateur ?

II.1 Portes logiques

Dans les circuits électriques qui composent un ordinateur deux états électriques possibles d'un point du circuit, en « tension basse » ou en « tension haute », sont associés respectivement aux chiffres 0 et 1. D'autre part, aux chiffres 0 et 1 sont eux-mêmes associés respectivement aux valeurs de vérité « Faux » et « Vrai ».

Avec un composant électronique élémentaire appelé « transistor » on réalise une situation dans laquelle un certain état électrique d'un point « en entrée » a pour conséquence que l'état électrique d'un point « en sortie » sera opposé : 1 si l'entrée est à 0 et 0 si l'entrée est à 1. On a donc le tableau suivant :

État de l'entrée	**Etat de la sortie**
0	1
1	0

On voit donc que l'on a réalisé ainsi la table de vérité de l'opérateur **not(...)**.

En combinant plusieurs transistors on peut également réaliser des circuits électrique pour lesquels l'état électrique d'un point en sortie dépend de l'état électrique de deux points en entrée. On construit ainsi des **portes logiques** qui réalisent de façon électronique différents opérateurs logiques binaires et notamment les opérateurs **and**, **or** et **xor**. Dans la représentation schématique d'un circuit ces portes logiques sont associées aux schémas suivants :

II.2 Addition binaire

En combinant ces différentes portes logiques, on peut réaliser des calculs arithmétiques élémentaires. Voyons par exemple comment on peut ainsi réaliser l'addition binaire de deux entrées e_0 et e_1 chacune formée d'un seul bit, pour obtenir un résultat sur deux bits s_0 et s_1. On doit obtenir le tableau suivant :

$\boldsymbol{e_0}$	$\boldsymbol{e_1}$	$\boldsymbol{s_1}$	$\boldsymbol{s_0}$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

En associant les 0 et les 1 à « faux » et « vrai », on peut donc écrire les expressions suivantes :

$$\boldsymbol{s_0 = e_0 \; xor \; e_1}$$

et $\boldsymbol{s_1 = e_0 \; and \; e_1}$

Le circuit d'addition binaire pourra donc être représenté de la façon suivante :