© Paul Delannoy freepol@free.fr

Une approche de l'algèbre booléenne

Lorsque l'on découvre les nombres entiers «naturels» on apprend très vite à les classer en 2 parties «égales» : les nombres sont PAIRS (2 les divise, la division par 2 «tombe juste», son reste est 0) ou IMPAIRS (le reste de leur division par 2 est 1, seule valeur possible autre que 0).

On décide de noter $\overline{0}$ l'ensemble des nombres pairs et $\overline{1}$ celui des nombres impairs : $\begin{cases}\overline{0}=\{2k, k\in\mathbb{N}\}\\ \overline{1}=\{2k+1, k\in\mathbb{N}\}\end{cases}$

Ceci prend de l'intérêt si l'on poursuit l'étude en s'intéressant aux opérations + et *:

si deux nombres n_1 et n_2 sont pairs, leur somme l'est aussi, de même que leur produit : $\begin{cases}2k_1+2k_2=2(k_1+k_2)\\ 2k_1 2k_2=2(2k_1k_2)\end{cases}$

Si n_1 est pair et n_2 impair, la somme est impaire et le produit pair : $\begin{cases}2k_1+2k_2+1=2(k_1+k_2)+1\\ 2k_1(2k_2+1)=2(2k_2k_1+k_1)\end{cases}$

le cas $n_1 \in \overline{1}$ et $n_2 \in \overline{0}$ donne le même résultat

Si les 2 sont impairs, leur somme est paire et leur produit impair : $\begin{cases}2k_1+1+2k_2+1=2(k_1+k_2+1)\\ (2k_1+1)(2k_2+1)=2(2k_1k_2+k_1+k_2)+1\end{cases}$

Il n'y a pas d'autres cas possibles, donc nous pouvons résumer ces calculs par deux «tables d'opération» qui donnent la valeur dans l'ensemble $\{\overline{0},\overline{1}\}$ du résultat de l'opération entre les couples (ligne,colonne) :

Addition	$\overline{0}$	$\overline{1}$
$\overline{0}$	0	1
$\overline{1}$	1	0

Lire 0+0=0=1+1, 1+0=0+1=1

Multiplication	$\overline{0}$	$\overline{1}$
$\overline{0}$	0	0
$\overline{1}$	0	1

Lire 0*0=0*1=1*0=0, 1*1=1

Boole a vu que ce que nous venons de faire définit deux opérations **+** et **x** dans l'ensemble que l'on note $\mathbb{Z}$ /2.

La logique «courante»

Dans la vie courante nous employons les expressions OU et ET. Si p et q sont des phrases (vraies ou fausses), la phrase $pOUq$ sera vraie ou fausse selon les *«valeurs de vérité»* de p et de q. De même pour la phrase $pETq$. Une réflexion rapide et simple donne de ces opérations les tables suivantes :

pOUq	p fausse	p vraie
q fausse	FAUX	VRAI
q vraie	VRAI	VRAI

Lire pVq : vrai si l'un est vrai au moins

pETq	p fausse	p vraie
q fausse	FAUX	FAUX
q vraie	FAUX	VRAI

Lire p^q : vrai si les 2 sont vrais

Une simple comparaison des tables permet de voir que l'opération ET et le produit **x** sont identiques. Par contre le OU et l'opération **+** ne le sont pas. Pour décrire le **+** il faudra établir la table de vérité de l'assemblage «OU exclusif» qui est faux si les 2 sont vrais.

Soyons un peu systèmatiques !

Comme un *opérateur logique* de la langue courante peut être décrit, comme nous venons de le faire pour ET et OU, grâce à sa *table de vérité* qui n'est autre qu'un tableau de 4 valeurs, chacune valant 0 ou 1, nous pouvons affirmer qu'il y a 2*2*2*2=16 opérations logiques et 16 seulement, dés lors que les propositions ne peuvent prendre que les valeurs VRAI (1) ou FAUX (0). Notons que ce choix est arbitraire et que le choix inverse est tout à fait possible...

En fait, il n'y en a pas 16, mais 8 !

Une opération courante dans la vie est de «nier» ou «dire le contraire». Il est donc une opération encore plus simple, qui consiste à définir la phrase *Nonp* qui prend la valeur que n'a pas p : $\neg p$ ou $\overline{p}$ vaut 0 si p vaut 1 et 1 si p vaut 0.
Avec cette fonction appelée *Négation* la recherche des 16 possibilités va se réduire à 8, puisque chaque tableau peut être associé à son «contraire» par négation. Par exemple avec le OU :

pVq	p fausse	p vraie
q fausse	FAUX	VRAI
q vraie	VRAI	VRAI

Lire pVq : vrai si l'un est vrai au moins

¬(pVq)	p fausse	p vraie
q fausse	VRAI	FAUX
q vraie	FAUX	FAUX

Lire ¬p^¬q : vrai si les 2 sont faux

Et on peut même encore réduire ce nombre !

Certaines tables comme celle-ci qui est clairement égale à $\neg p$: $\begin{pmatrix} & p & \neg p \\ q & 0 & 1 \\ \neg q & 0 & 1 \end{pmatrix}$
ont des valeurs qui ne dépendent pas d'une des deux entrées.
On trouvera donc 4 tables de cette sorte, qui sont des opérations «projection» : $p, \neg p, q, \neg q$

Il y a aussi les opérateurs «constants» : $\begin{pmatrix} & p & \neg p \\ q & 0 & 0 \\ \neg q & 0 & 0 \end{pmatrix}$ et sa négation $\begin{pmatrix} & p & \neg p \\ q & 1 & 1 \\ \neg q & 1 & 1 \end{pmatrix}$

Au total, des 16 initiaux, ne restent que 10, appariés en deux groupes de 5 par la négation.

Ces 5 là sont $p \wedge q$, $p \vee q$ déjà rencontrés et liés par les lois de DeMorgan $\begin{cases} \neg(p \wedge q) = \neg p \vee \neg q \\ \neg p \wedge \neg q = \neg(p \vee q) \end{cases}$

Mais aussi les deux $p \Rightarrow q$ et $q \Rightarrow p$, et leur assemblage $p \Leftrightarrow q$. Commencons par cette «*équivalence*» : sa table est bien sûr $\begin{pmatrix} & p & \neg p \\ q & 1 & 0 \\ \neg q & 0 & 1 \end{pmatrix}$: les deux sont faux ou les deux sont vrais, les *valeurs de vérité* sont égales.
C'est la multiplication dans $\mathbb{Z}/2\mathbb{Z}$, c'est le *«OU exclusif»,* appelé XOR en informatique et électronique. Remarquons que *pXORp* a exactement pour valeur $\neg p$, ce qui explique que TOUS les circuits logiques d'un ordinateur peuvent être réalisés en assemblant UNIQUEMENT des *«portes logiques»* réalisant XOR.

Du coup $\neg(p \Leftrightarrow q)$ a pour table $\begin{pmatrix} & p & \neg p \\ q & 0 & 1 \\ \neg q & 1 & 0 \end{pmatrix}$ c'est $\neg p \vee \neg q$ et $p \Rightarrow q$ a pour table $\begin{pmatrix} & p & \neg p \\ q & 1 & 1 \\ \neg q & 0 & 1 \end{pmatrix}$ qui est clairement celle de $\neg p \vee q$ («Toto mange ta soupe ou tu auras la fessée» ne veut pas dire que manger sa soupe suffit à éviter ladite fessée).

On aura donc pour $q \Rightarrow p$ la table $\begin{pmatrix} & p & \neg p \\ q & 1 & 0 \\ \neg q & 1 & 1 \end{pmatrix}$ qui est celle de $\neg q \vee p$. Et calculer $(p \Rightarrow q) \wedge (q \Rightarrow p)$ nous méne à $(\neg p \vee q) \wedge (p \vee \neg q)$ *id-est* à la table de $p \Leftrightarrow q$: $\begin{pmatrix} & p & \neg p \\ q & 1 & 0 \\ \neg q & 0 & 1 \end{pmatrix}$ ce qui est le moins que l'on pouvait attendre.