

**Exercice 1. Nombre de variables**

Soit  $\varphi$  une formule logique n'utilisant pas  $\neg$ . Soit  $n$  le nombre de connecteurs logiques ( $\vee$  et  $\neg$ ) dans  $\varphi$ .

Exprimer le nombre d'**occurrences** (c'est à dire compté avec multiplicité) de variables de  $\varphi$ , en fonction de  $n$ .

**Exercice 2. Tautologies**

Soient  $p, q, r$  des formules. Montrer que les formules suivantes sont des tautologies :

1.  $p \implies (q \implies p)$
2.  $(p \implies q) \vee (q \implies r)$
3.  $(\neg\neg p) \implies p$

**Exercice 3. Énigme gastronomique**

Trois personnes (nommée  $A, B, C$ ) mangent ensemble. On sait que :

- si  $A$  prend un dessert,  $B$  aussi
- soit  $B$ , soit  $C$  prennent un dessert, mais pas les deux
- $A$  ou  $C$  prend un dessert
- si  $C$  prend un dessert,  $A$  aussi

Déterminer qui prend un dessert, en utilisant une table de vérité.

**Exercice 4. Calcul booléen**

Donner des formules équivalentes les plus simples possibles pour les formules suivantes (en utilisant le moins de littéraux possible) :

1.  $\varphi_1 = c(b + c) + (a + d)(\overline{ad} + c)$
2.  $\varphi_2 = ab + c + \overline{b}\overline{c} + \overline{a}\overline{c}$
3.  $\varphi_3 = \neg(a \wedge b) \wedge (a \vee \neg b) \wedge (a \vee b)$

**Exercice 5. Système complet logique**

On définit les opérateurs NAND, NOR, XOR par leurs tables de vérité :

$x$	$y$	$x \text{ NAND } y$	$x \text{ NOR } y$	$x \text{ XOR } y$
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

On dit qu'un ensemble  $S$  d'opérateurs logiques est **complet** si toute formule logique est équivalente à une formule qui n'utilise que des opérateurs dans  $S$ .

1. Exprimer NAND, NOR, XOR, à l'aide de  $\vee, \wedge, \neg$ .
2. Montrer que  $\{\wedge, \neg\}$  est complet.
3. Montrer que  $\{\text{NAND}\}$  est complet. (c'est pour cette raison que le NAND est très utilisé en électronique)
4. Montrer que  $\{\text{NOR}\}$  est complet.
5. Montrer que  $\{\text{XOR}\}$  n'est pas complet.

**Exercice 6. Forme normale conjonctive/disjonctive**

On rappelle qu'une forme normale conjonctive (FNC) est une conjonction ( $\wedge$ ) de disjonctions ( $\vee$ ) de littéraux (chaque littéral étant une variable ou sa négation).

On rappelle qu'une forme normale disjonctive (FND) est une disjonction ( $\vee$ ) de conjonctions ( $\wedge$ ) de littéraux (chaque littéral étant une variable ou sa négation).

Par exemple,  $(a \vee b) \wedge (b \vee c \vee d) \wedge c$  est une FNC.

1. Montrer par induction/récurrence que toute formule logique (construite avec  $\wedge, \vee, \neg$ ) est équivalente à une FNC ainsi qu'à une FND.
2. On rappelle que  $x \implies y$  est une notation pour  $\neg x \vee y$ .  
Donner une FNC et une FND équivalente à  $\neg(x \implies (\neg y \wedge z)) \vee (z \implies y)$ .

### Exercice 7. Extrait Mines-Pont 2010

On appelle **variable booléenne** une variable qui ne peut prendre que les valeurs 0 (synonyme de faux) ou 1 (synonyme de vrai). Si  $x$  est une variable booléenne, on note  $\bar{x}$  le complémenté (ou négation) de  $x$  :  $x$  vaut 1 si  $x$  vaut 0 et  $x$  vaut 0 si  $x$  vaut 1.

On appelle **littéral** une variable booléenne ou son complémenté.

On représente la **disjonction** (« ou » logique) par le symbole  $\vee$  et la **conjonction** (« et » logique) par le symbole  $\wedge$ .

On appelle **clause** une disjonction de littéraux. De plus, il ne doit pas y avoir deux fois la même variable dans une clause.

On appelle **formule logique sous forme normale conjonctive** une conjonction de clauses.

On appelle **valuation** des variables d'une formule logique une application de l'ensemble de ces variables dans l'ensemble  $\{0, 1\}$ .

Une clause vaut 1 si au moins un de ses littéraux vaut 1 et 0 sinon. Une clause est dite **satisfaite** par une valuation des variables si elle vaut 1 pour cette valuation. Une formule logique sous forme normale conjonctive vaut 1 si toutes ses clauses valent 1 et 0 sinon. Une formule logique est dite **satisfaite** par une valuation des variables si elle vaut 1 pour cette valuation. Une formule logique est dite **satisfiable** s'il existe une valuation de ses variables qui la satisfait.

Étant donnée une formule logique  $f$  sous forme normale conjonctive, on note dans ce problème **max( $f$ )** le nombre maximum de clauses de  $f$  pouvant être satisfaites par une même valuation.

En notant  $m$  le nombre de clauses de  $f$ , on remarque que  $f$  est satisfiable si et seulement si  $\max(f) = m$ .

On considère la formule  $f_1$  (sous forme normale conjonctive) dépendant des variables  $x, y, z$  :

$$f_1 = (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{x} \vee \bar{z}) \wedge (x \vee \bar{y} \vee z)$$

1. Indiquer si  $f_1$  est satisfiable ou non et, si elle est satisfiable, donner l'ensemble des solutions de  $f_1$ .

Une **instance de 3-SAT** est une formule logique sous forme normale conjonctive dont toutes les clauses contiennent 3 littéraux.

2. Déterminer une instance  $f_2$  de 3-SAT non satisfiable et possédant exactement 8 clauses; indiquer  $\max(f_2)$  en justifiant la réponse.

On considère une instance  $f$  de 3-SAT définie sur  $n$  variables booléennes.

On note  $V$  l'ensemble des  $2^n$  valuations des variables de  $f$ .

Soit  $val$  une valuation des  $n$  variables. Si  $C$  est une clause, on note  $\varphi(C, val)$  la valeur de  $C$  pour la valuation  $val$  et on note  $\psi(f, val)$  le nombre de clauses de  $f$  qui valent 1 pour la valuation  $val$ .

On a :  $\psi(f, val) = \sum_{C \text{ clause de } f} \varphi(C, val)$  et  $\max(f) = \max_{val \in V} \psi(f, val)$ .

3. Soit  $C$  une clause de  $f$ . Donner une expression simple de  $\sum_{val \in V} \varphi(C, val)$ , en fonction de  $n$ .

4. Soit  $m$  le nombre de clauses dont  $f$  est la conjonction.

En considérant la somme  $\sum_{C \text{ clause de } f} \sum_{val \in V} \varphi(C, val)$ , donner en fonction de  $m$  un minorant de  $\max(f)$ .

5. Donner le nombre minimum de clauses d'une instance de 3-SAT non satisfiable.

### Exercice 8. Réduction de 3-SAT à d'autres problèmes

Le problème 3-SAT consiste à déterminer si une formule en forme normale conjonctive dont chaque clause contient 3 littéraux est satisfiable.

Une des questions les plus importantes en informatique est de savoir s'il est possible de résoudre 3-SAT en temps polynomial (en la taille de la formule). On pense qu'il n'en existe pas (c'est la fameuse conjecture  $P \neq NP$ ).

1. On considère le problème CLIQUE : étant donné un graphe  $G$  et un entier  $k$ , existe-t-il un sous-graphe complet (une *clique*) à  $k$  sommets dans  $G$ ?  
Montrer que si on peut résoudre CLIQUE en temps polynomial alors on peut résoudre 3-SAT en temps polynomial.