TD corrigé partiel logique

1 Nombre de variables

Soit ϕ une formule logique n'utilisant pas $\neg$. Soit n le nombre de connecteurs logiques ($\vee$ et $\neg$) dans ϕ.
Exprimer le nombre d'**occurrences** (c'est à dire compté avec multiplicité) de variables de ϕ, en fonction de n.
► On peut représenter ϕ par un arbre (voir cours/résumé) où les noeuds internes sont les connecteurs logiques et les feuilles sont les occurrences variables.
S'il n'y a pas de $\neg$, tous les connecteurs logiques sont des binaires donc l'arbre est binaire strict.
D'après la formule vue dans le cours sur les arbres, le nombre de feuilles de cet arbre est égal à 1 + le nombre de noeuds internes.
Donc le nombre d'occurrences de variables dans ϕ est $\boxed{n+1}$.
On peut le vérifier sur un exemple: si $\phi = (x \wedge z) \vee (z \vee y)$ alors ϕ a 3 connecteurs logiques et $4 = 3 + 1$ occurrences de variables.
Remarque: on peut aussi le démontrer par récurrence (la formule sur les arbres est aussi démontrée par récurrence).

2 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.
► On note a, b, c des variables booléennes indiquant si A, B, C prend un dessert (par exemple, $a = 1 \longleftrightarrow A$ prend un dessert).
Les 4 conditions deviennent:

- $a \implies b$ (qui est équivalent à $\neg a \vee b$)
- $(b \vee c) \wedge \neg(b \wedge c)$ (qui est équivalent à $(b \wedge \neg c) \vee (\neg b \wedge c)$)
- $a \vee c$
- $c \implies a$ (qui est équivalent à $\neg c \vee a$)

En écrivant les valeurs de vérités de ces 4 formules:

a	b	c	$\neg a \vee b$	$(b \wedge \neg c) \vee (\neg b \wedge c)$	$a \vee c$	$\neg c \vee a$
0	0	0	1	0	0	1
0	0	1	1	1	1	0
0	1	0	1	1	0	1
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	1	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

(on peut vérifier qu'on a bien $2^3 = 8$ lignes)
On trouve qu'une seule possibilité pour que les 4 conditions soit réunies simultanément: $a = 1$, $b = 1$, $c = 0$ c'est à dire A et B prennent un dessert mais pas C.

3 Calcul booléen

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

1. $\phi_1 = c(b+c) + (a+d)\overline{(a\overline{d}+c)}$
   ► $\phi_1 \equiv cb + c + (a+d)((\overline{a}+d)\overline{c}) \equiv c + (a\overline{a} + ad + d\overline{a} + dd)\overline{c} \equiv c + (0 + (a+\overline{a})d + d)\overline{c} \equiv c + (d+d)\overline{c} \equiv c + d\overline{c}$.

2. $\phi_2 = ab + c + \overline{b}\overline{c} + \overline{a}\,\overline{c}$
   ► $\phi_2 \equiv ab + c + (\overline{b} + \overline{a})\overline{c} \equiv ab + c + \overline{ab}\overline{c} \equiv ab + c + \overline{ab + c} \equiv 1$: ϕ_2 est une tautologie.
3. $\phi_3 = \neg(a \wedge b) \wedge (a \vee \neg b) \wedge (a \vee b)$

4 Système complet logique

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

x	y	$x\ NAND\ y$	$x\ NOR\ y$	$x\ XOR\ y$
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 $\vee$, $\wedge$, $\neg$.
   ► $x\ NAND\ y \equiv \neg(x \wedge y)$, $x\ NOR\ y \equiv \neg(x \vee y)$, $xXORy \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$ $(\equiv (x \vee y) \wedge (\neg x \vee \neg y))$.
2. Montrer que $\{\wedge, \neg\}$ est complet.
   ► $x \vee y \equiv \neg(\neg x \wedge \neg y)$. Toute formule avec des $\vee$, $\wedge$, $\neg$ peut donc être réécrite avec que des $\wedge$ et $\neg$.
3. Montrer que $\{NAND\}$ est complet. (c'est pour cette raison que le NAND est très utilisé en électronique)
   ► Si $x = 1$ alors $x\ NAND\ x = 0$ et si $x = 0$ alors $x\ NAND\ x = 1$. D'où $\neg x \equiv x\ NAND\ x$.
   Vu la table vérité de $NAND$, $x \wedge y = \neg(x\ NAND\ y) = (x\ NAND\ y)\ NAND\ (x\ NAND\ y)$.
   Comme $\{\wedge, \neg\}$ est complet (d'après la question précédente), $\{NAND\}$ l'est aussi.
4. Montrer que $\{NOR\}$ est complet.
   ► $\neg x\ NOR\ \neg y = 1$ ssi $x = 1$ et $y = 1$. D'où $x\ NAND\ y = \neg(\neg x\ NOR\ \neg y)$.
   De plus $\neg x = x\ NOR\ x$. Donc $x\ NAND\ y$ peut s'écrire avec que des NOR. Comme $NAND$ est complet, NOR est complet.
5. Montrer que $\{XOR\}$ n'est pas complet.
   ► Nous allons trouver une propriété (qu'on appelle parfois un invariant) vérifiée par une formule n'utilisant que des XOR mais qui n'est pas vraie pour les formules en général.
   Soit ϕ une formule n'utilisant que des variables et des XOR.
   Soit d l'évaluation donnant la valeur 0 à toutes les variables de ϕ.
   On peut démontrer par récurrence sur la taille de ϕ que $[\![\phi]\!]_d = 0$:

   - si ϕ est une variable (cas de base) alors $[\![\phi]\!]_d = 0$ par définition de d
   - si $\phi = \phi_1\ XOR\ \phi_2$ alors, par hypothèse de récurrence (car ϕ_1 et ϕ_2 sont de tailles plus petites que ϕ et ne contiennent que des XOR), $[\![\phi_1]\!]_d = 0$ et $[\![\phi_2]\!]_d = 0$ donc $[\![\phi]\!]_d = 0$ (d'après la table de vérité de XOR).

   On ne peut donc pas obtenir la formule $\neg x$ (par exemple) avec que des XOR puisque cette formule est vraie quand $x = 0$.

5 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.
   ► On définit la taille d'une formule comme son nombre de connecteurs logiques ($\wedge$, $\neg$, $\vee$) et de variables.
   Montrons $P(n)$: « si ϕ est une formule de taille n alors ϕ est équivalente à une FND ainsi qu'à une FNC ».
   $P(1)$ est vraie car une formule de taille 1 est une variable qui est à la fois une FND et une FNC.
   Soit $n \in \mathbb{N}^*$. Supposons $P(k)$ pour tout $k \leq n$.
   Soit ϕ une formule de taille $n+1$:

   - Si ϕ est la négation d'une formule, i.e $\phi = \neg\psi$. ψ est de taille n donc d'après $P(n)$, $\psi \equiv \psi'$ et $\psi \equiv \psi''$, où ψ' est une FNC et ψ'' une FND.
     Alors, en appliquant les lois de De Morgan sur $\neg\psi'$ on obtient une FND et en appliquant les lois de De Morgan sur $\neg\psi''$ on obtient une FNC.

- Si $\phi = \psi \wedge \psi'$: soient (par hypothèse de récurrence) f et f' des FNC équivalentes à ψ et ψ', respectivement. Clairement, $f \wedge f'$ est une FNC équivalente à ϕ.
  Soient $g = c_1 \vee c_2 \vee ... \vee c_k$ et $g' = c'_1 \vee c'_2 \vee ... \vee c'_k$ des FND équivalentes à ψ et ψ', respectivement.
  Alors $g \wedge g' = (c_1 \vee c_2 \vee ... \vee c_k) \wedge (c'_1 \vee c'_2 \vee ... \vee c'_k) = \bigvee_{i,j} (c_i \wedge c'_j)$ est une FND.

- Démonstration similaire si $\phi = \psi \vee \psi'$...

2. On rappelle que $x \implies y$ est une notation pour $\neg x \vee y$.
   Donner une FNC et une FND équivalente à $\neg(x \implies (\neg y \wedge z)) \vee (z \implies y)$.
   ▶ $\neg(x \implies (\neg y \wedge z)) \vee (z \implies y) \equiv \neg(\neg x \vee (\neg y \wedge z)) \vee (\neg z \vee y) \equiv (x \wedge (y \vee \neg z)) \vee \neg z \vee y$.
   Pour trouver une FND, on peut utiliser la distributivité de $\wedge$ dans $x \wedge (y \vee \neg z)$: $(x \wedge (y \vee \neg z)) \vee \neg z \vee y = \boxed{(x \wedge y) \vee (x \wedge \neg z) \neg z \vee y}$.
   Pour trouver une FNC, on peut utiliser la distributivité de $\vee$ dans $(x \wedge (y \vee \neg z)) \vee (\neg z \vee y)$:
   $(x \wedge (y \vee \neg z)) \vee (\neg z \vee y) = (x \vee (\neg z \vee y)) \wedge ((y \vee \neg z) \vee (\neg z \vee y)) \equiv \boxed{(x \vee (\neg z \vee y)) \wedge (y \vee \neg z)}$.

6 Extrait Mines-Pont 2010

1. En construisant la table de vérité de f_1, on trouve qu'elle est satisfiable pour $(x, y, z) \in \{(1,0,0), (0,1,1), (0,0,1)\}$

x	y	z	f_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

2. Soit f_2 la conjonction de toutes les différentes clauses de taille 3 (au nombre de $2^3 = 8$) sur x, y, z, c'est à dire:

$$f_2 = (x \vee y \vee z) \wedge (x \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y \vee z) \wedge (\overline{x} \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{y} \vee \overline{z})$$

   Alors chaque valuation de x, y, z satisfait toutes les clauses de f_2 sauf une seule: par exemple, $(x, y, z) = (1, 0, 1)$ satisfait toues ses clauses sauf $\overline{x} \vee y \vee \overline{z}$.
   On en déduit que f_2 n'est pas satisfiable et que $\max(f_2) = 7$.

3. Une clause C est satisfaite par toute valuation sauf celles qui donne un triplet de valeurs booléennes particulier aux trois variables qui apparaissent dans C (les 2^{n-3} autres variables ont alors une valeur quelconque):

$$\sum_{val \in V} \varphi(C, val) = 2^n - 2^{n-3} = \frac{7}{8} 2^n$$

4.

$$\sum_{C \text{ clause de } f} \sum_{val \in V} \varphi(C, val) = \frac{7m}{8} 2^n = \sum_{val \in V} \sum_{C \text{ clause de } f} \varphi(C, val) = \sum_{val \in V} \psi(f, val) \leq 2^n \max(f)$$

   Donc $\max(f) \geq \frac{7}{8} m = m - \frac{m}{8}$ et, comme $\max(f)$ est un entier: $\max(f) \geq \lceil \frac{7}{8} m \rceil = m - \lfloor \frac{m}{8} \rfloor$.

5. Supposons f non satisfiable. Alors $\max(f) < m$ donc, d'après la question précédente, $m - \lfloor \frac{m}{8} \rfloor < m$ donc $\lfloor \frac{m}{8} \rfloor \geq 1$ et $m \geq 8$.
   Une formule non satisfiable contient donc au moins 8 clauses.
   D'après la question 2, 8 est donc le nombre minimum de clauses d'une instance de 3-SAT non satisfiable.

7 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.
   ▶ Soit $\phi = c_1 \wedge ... \wedge c_k$ une formule sous forme normale conjonctive, où $c_i = \ell_1^i \vee \ell_2^i \vee \ell_3^i$.

Soit G le graphe dont les sommets sont tous les littéraux ℓ_j^i apparaissant dans ϕ et tel que deux littéraux sont adjacents ssi ils ne sont pas dans la même clause et ils ne sont pas la négation l'un de l'autre.
Montrons que ϕ est satisfiable ssi G possède une clique de taille k.
Supposons ϕ satisfiable par une valuation d. Chaque clause c_i doit posséder un littéral v_i satisfait par d. Alors l'ensemble $\{v_i \mid 1 \leq i \leq k\}$ des sommets correspondant à ces littéraux forme une clique de taille k dans G.
Supposons que G possède une clique de taille k et soit S l'ensemble de ses sommets. Comme 2 sommets d'une même clause ne sont pas être reliés entre eux dans G, S possède un littéral dans chaque clause de ϕ. On peut alors donner alors comme valeur de vérité 0 à chaque x tel que $\neg x \in S$ et 1 si $x \in S$. On obtient ainsi une valuation satisfaisant ϕ.