I Exercice CCP

Rappelons les règles de déduction naturelle suivantes, où A et B sont des formules logiques et Γ un ensemble de formules logiques quelconques :

$$\frac{}{\Gamma, A \vdash A}\text{AX} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}\bot_e \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\to_i \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash A \to B}{\Gamma \vdash B}\to_e \qquad \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\neg_i \qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\neg_e$$

1. Montrer que le séquent $\vdash \neg A \to (A \to \bot)$ est dérivable, en explicitant un arbre de preuve.
2. Montrer que le séquent $\vdash (A \to \bot) \to \neg A$ est dérivable, en explicitant un arbre de preuve.
3. Donner une règle correspondant à l'introduction du symbole $\wedge$ ainsi que deux règles correspondant à l'élimination du symbole $\wedge$. Montrer que le séquent $\vdash (\neg A \to (A \to \bot)) \wedge ((A \to \bot) \to \neg A)$ est dérivable.
4. On considère la formule $P = ((A \to B) \to A) \to A$ appelée loi de Peirce. Montrer que $\models P$, c'est-à-dire que P est une tautologie.
5. Pour montrer que le séquent $\vdash P$ est dérivable, il est nécessaire d'utiliser la règle d'absurdité classique $\bot_c$ (ou une règle équivalente), ce que l'on fait ci-dessous (il n'y aura pas besoin de réutiliser cette règle). Terminer la dérivation du séquent $\vdash P$, dans laquelle on pose $\Gamma = \{(A \to B) \to A, \neg A\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{\Gamma \vdash A}\,? \qquad \dfrac{}{\Gamma \vdash \neg A}\,\text{AX}}{\Gamma = (A \to B) \to A, \neg A \vdash \bot}\,\neg_i}{(A \to B) \to A) \vdash A}\,\bot_c}{\vdash ((A \to B) \to A) \to A}\,\to_i$$

II Lois de de Morgan

En utilisant la logique minimale (sans utiliser le raisonnement par l'absurde raa ni $\bot_e$) :

1. Prouver le séquent $\neg P \vee \neg Q \vdash \neg(P \wedge Q)$.
2. Prouver le séquent $\neg(P \vee Q) \vdash \neg P \wedge \neg Q$.
3. Prouver le séquent $\neg P \wedge \neg Q \vdash \neg(P \vee Q)$.

En utilisant la logique classique (avec raa) :

4. Prouver le séquent $\neg(P \wedge Q) \vdash \neg P \vee \neg Q$.

III Démonstrations

1. Prouver le séquent $\vdash P \to (Q \to P)$.
2. Prouver le séquent $\vdash P \to (P \to Q) \to Q$.
3. Prouver le séquent $\vdash \neg(P \wedge \neg P)$.

Exercice 1. Nombre de variables
Soit ϕ une formule logique. Soit n le nombre de connecteurs logiques ($\vee$ et $\wedge$) dans ϕ.
Exprimer le nombre de terminaux (variables, T ou F) de ϕ, en fonction de n.

Exercice 2. Tautologie et équivalence
Soient ϕ_1, ϕ_2, ϕ_3 des formules. Montrer que les formules suivantes sont des tautologies :

1. $\phi_1 \longrightarrow (\phi_2 \longrightarrow \phi_1)$
2. $(\phi_1 \longrightarrow \phi_2) \vee (\phi_2 \longrightarrow \phi_3)$
3. $(\neg\neg\phi_1) \longrightarrow \phi_1$

Pour chacune des équivalences suivantes, la prouver ou trouver un contre-exemple :

4. $(\phi_1 \longrightarrow \phi_2) \longrightarrow \phi_3 \equiv \phi_1 \longrightarrow (\phi_2 \longrightarrow \phi_3)$
5. $(\phi_1 \wedge \phi_2) \longrightarrow \phi_3 \equiv \phi_1 \longrightarrow (\phi_2 \longrightarrow \phi_3)$

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. $\phi_1 = c(b+c) + (a+d)\overline{(a\overline{d}+c)}$
2. $\phi_2 = ab + c + \overline{b}\overline{c} + \overline{a}\ \overline{c}$
3. $\phi_3 = \neg(a \wedge b) \wedge (a \vee \neg b) \wedge (a \vee b)$

Exercice 5. Système complet logique
On définit les opérateurs NAND, NOR, XOR par leurs tables de vérité :

x	y	$x\ NAND\ y$	$x\ NOR\ y$	$x\ XOR\ y$
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 $\vee$, $\wedge$, $\neg$.
2. Montrer que $\{\wedge, \neg\}$ est complet.
3. Montrer que $\{NAND\}$ est complet. (c'est pour cette raison que le NAND est très utilisé en électronique)
4. Montrer que $\{NOR\}$ est complet.
5. Montrer que $\{XOR\}$ 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). Par exemple, $(a \vee b) \wedge (b \vee c \vee d) \wedge c$ est une FNC.
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).

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. Donner une FNC et une FND équivalente à $\neg(x \longrightarrow (\neg y \wedge z)) \vee (z \longrightarrow y)$.
3. Écrire une fonction `fnc : 'a formula -> 'a formula` mettant une forme en FNC.

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 $\overline{x}$ 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 :

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

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. 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 : $\psi(f, val) = \sum\limits_{C \text{ clause de } f} \varphi(C, val)$ et $\max(f) = \max\limits_{val \in V} \psi(f, val)$.

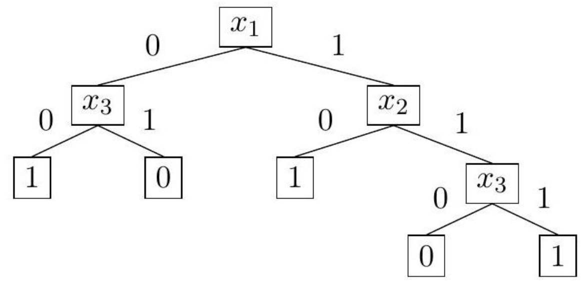
3. Soit C une clause de f. Donner une expression simple de $\sum\limits_{val \in V} \varphi(C, val)$, en fonction de n.
4. Soit m le nombre de clauses dont f est la conjonction. En considérant la somme $\sum\limits_{C \text{ clause de } f} \sum\limits_{val \in V} \varphi(C, val)$, donner en fonction de m un minorant de $\max(f)$.
5. Donner le nombre minimum de clauses d'une instance de 3-SAT non satisfiable.

Exercice 8. Arbre de décision (Oral ENS info)

On considère un ensemble de variables propositionnelles $\mathcal{X} = \{x_1, \ldots, x_n\}$ muni de l'ordre total où $x_i < x_j$ si et seulement si $i < j$. Un arbre de décision sur $\mathcal{X}$ est un arbre binaire dont les feuilles sont étiquetées par 0 ou par 1 , et dont les nœuds internes sont étiquetés par une variable de $\mathcal{X}$ et ont deux enfants appelés enfant 0 et enfant 1 . On impose que si un nœud interne n étiqueté par x_i a un descendant n' qui est un nœud interne étiqueté par x_j alors $x_i < x_j$.

Un arbre de décision T sur $\mathcal{X}$ décrit une fonction booléenne Φ_T qui à toute valuation $\nu : \mathcal{X} \to \{0, 1\}$ associe une valeur de vérité calculée comme suit : si T consiste exclusivement d'une feuille étiquetée $b \in \{0, 1\}$ alors la fonction Φ_T s'évalue à b quelle que soit ν. Sinon, on considère le nœud racine n de T et la variable x_i qui l'étiquette, on regarde la valeur $\nu(x_i) \in \{0, 1\}$ que ν donne à x_i, et le résultat de l'évaluation de Φ_T sous ν est celui de l'évaluation de $\Phi_{T'}$ sous ν, où T' est le sous-arbre de T enraciné en l'enfant b de n.

1. On considère l'arbre de décision T_0 suivant et la fonction Φ_{T_0} qu'il définit. Évaluer cette fonction pour la valuation donnant à x_1, x_2, x_3 respectivement les valeurs $0, 1, 0$. Donner un exemple de valuation sous laquelle cette formule s'évalue en 0 .



2. On considère la formule de la logique propositionnelle $(x_1 \wedge x_2) \vee \neg(x_1 \wedge \neg x_3)$. Construire un arbre de décision sur les variables $x_1 < x_2 < x_3$ qui représente la même fonction.
3. Quels arbres de décision représentent des tautologies? des fonctions satisfiables?
4. Étant donné un arbre de décision représentant une fonction Φ, expliquer comment construire un arbre de décision représentant sa négation $\neg\Phi$.
5. Étant donné un arbre de décision représentant une fonction Φ, donner le pseudocode d'un algorithme qui calcule une représentation de Φ sous la forme d'une formule de la logique propositionnelle. Analyser sa complexité en temps et en espace.
6. Étant donné deux arbres de décision représentant des formules Φ_1 et Φ_2 sur le même ensemble de variables et avec le même ordre, expliquer comment construire un arbre de décision représentant $\Phi_1 \wedge \Phi_2$. Quelle est sa complexité en temps et la taille de l'arbre ainsi obtenu?
7. Étant donné un arbre de décision et la séquence de variables $x_1, \ldots, x_n$, donner le pseudocode d'un algorithme qui calcule combien de valuations satisfont la fonction booléenne qu'il capture. Quelle est sa complexité en temps?
8. Peut-on efficacement récrire une formule quelconque de la logique propositionnelle en arbre de décision?

   On se propose dans les quelques prochaines questions de démontrer formellement l'intuition de la question 7.

7a. Une formule booléenne en forme normale disjonctive, également appelée DNF, est une formule qui est une disjonction de conjonctions de littéraux (c'est-à-dire une variable ou sa négation).

Montrer qu'étant donné un arbre de décision on peut efficacement calculer une formule en DNF qui représente la même fonction.

7b. Montrer que, pour tout n, la formule $\bigwedge\limits_{1 \leq i \leq n} (x_i \vee y_i)$ n'admet pas de représentation concise sous la forme d'une DNF.

7c. Conclure.