Feuille 4 - Calcul propositionnel : Formes normales

Exercice 1 On introduit le connecteur $\diamond$ (nor) dont la table de vérité est donnée par

x	y	$x \diamond y$
V	V	F
V	F	F
F	V	F
F	F	V

1. Donner une forme normale conjonctive et une forme normale disjonctive de la formule $x \diamond y$.
2. Donner les tables de vérité des formules $(x \diamond x)$, $((x \diamond x) \diamond x)$ et $(x \diamond y) \diamond (x \diamond y)$
3. Donner des formules équivalentes à $\top$, $\bot$, $\neg x$, $x \wedge y$, $x \vee y$ et $x \Rightarrow y$ qui n'utilisent que l'opérateur $\diamond$ et possiblement les variables x et y. On pourra justifier le résultat soit par des tables de vérité, soit en utilisant des équivalences avec les formules propositionnelles usuelles.
4. Définir par des équations récursives une fonction `nor` qui étant donnée une formule propositionnelle construite sur les connecteurs habituels, la transforme en une formule qui ne contient que des variables propositionnelles et le connecteur $\diamond$.
5. Donner le résultat de $\texttt{nor}(x \vee (y \vee z))$.

Exercice 2 *Formes normales à partir d'une table de vérité.*

Donner les formes normales conjonctives et disjonctives des formules P et Q dont les tables de vérité sont les suivantes :

	a	b	P
1	V	V	F
2	V	F	V
3	F	V	V
4	F	F	F

	a	b	c	Q
1	V	V	V	F
2	V	V	F	V
3	V	F	V	V
4	V	F	F	V
5	F	V	V	F
6	F	V	F	F
7	F	F	V	V
8	F	F	F	F

Exercice 3 *Formes normales.*

1. A l'aide des lois de de Morgan et des lois de distributivité, mettre sous forme normale conjonctive et sous forme normale disjonctive les formules suivantes :
   (a) $(\neg p \vee \neg q \vee r) \Rightarrow (r \vee s)$
   (b) $p \Rightarrow ((\neg q \vee r) \Rightarrow s)$
   (c) $\neg(p \vee \neg q) \wedge (q \Rightarrow (p \vee r))$
   (d) $(p \Rightarrow q) \wedge \neg q \wedge \neg(q \Rightarrow p)$
   (e) $(\neg p \Rightarrow q) \wedge (q \Rightarrow p)$
2. Donner pour chacune des formules ci-dessus un modèle.
3. Utiliser les exemples précédents pour répondre aux questions suivantes
   — Est-il possible d'avoir une formule qui soit simultanément en forme normale conjonctive et disjonctive ?

— Est-ce vrai que toutes les formes normales conjonctives d'une formule donnée ont le même nombre de clauses ?
— Est-ce vrai si on ne permet pas de répétitions de clauses dans une forme normale conjonctive, ni de clause qui se simplifie trivialement ?

4. (optionnel) Montrer par récurrence structurelle que toute formule P admet une forme normale disjonctive. On considèrera seulement le cas où la formule P est en forme normale négative (utilise uniquement les connecteurs $\bot$, $\top$, $\vee$ et $\wedge$ et des littéraux).

Exercice 4 On reprend le type Ocaml vu en cours pour représenter les formules propositionnelles

```
type connecteur = Impl | Et | Ou
type form = Var of int | Bot | Top
          | Neg of form | Bin of form * connecteur * form
```

Compléter les fonctions suivantes afin d'implanter les fonctions `clauseb` et `fncb` qui testent respectivement si une formule est une clause et si une formule est en forme normale conjonctive.

```
let rec clauseb = function
    Var n ->
  | Bot
  | Top ->
  | Neg f ->
  | Bin(f,Or,g) ->
  | Bin(f,Et,g) ->
  | Bin(f,Impl,g) ->
```

```
let rec fncb = function
    Var n ->
  | Bot
  | Top ->
  | Neg f ->
  | Bin(f,Or,g) ->
  | Bin(f,Et,g) ->
  | Bin(f,Impl,g) ->
```

Exercice 5 *Propriétés théoriques des clauses*

On se donne n variables propositionnelles $x_1, \ldots, x_n$. On pourra commencer par regarder le cas $n = 3$. On rappelle qu'une clause est une disjonction de littéraux avec deux cas "limites" à savoir la clause vide $\bot$ et la clause triviale $\top$.

1. Combien de fonctions booléennes différentes y-a-t-il en utilisant seulement ces variables ?
2. Montrer que si une clause C contient deux littéraux qui portent sur la même variable propostionnelle alors il existe une clause C' plus courte (avec moins de littéraux) qui a la même table de vérité. On appellera dans la suite clause réduite, une telle clause.
3. Combien de clauses différentes réduites peut-on construire qui n'ont pas la même table de vérité ?
4. Soit deux clauses C et D réduites telles que $C \models D$ et $D \neq \top$, comment se comparent l'ensemble des littéraux de C et ceux de D ?
5. Soit une clause C réduite qui comporte des littéraux sur les n variables. Combien de lignes de la table de la vérité ont la valeur vrai ?
6. Soit une clause C réduite qui comporte p des n variables. Combien de lignes de la table de la vérité ont la valeur vrai ?

Exercice 6 *Mise en forme clausale efficace*

On a vu en cours que la mise en forme clausale d'une formule P (qui correspond à la forme normale conjonctive) pouvait aboutir à faire grossir exponentiellement la taille de la formule. On montre ici que si on s'intéresse juste à savoir si P est satisfiable alors on peut trouver un ensemble de clauses qui est satisfiable exactement lorsque P est satisfiable et dont la taille est linéaire en la taille de P.

Le principe de la décomposition est le suivant. On part d'une formule P qui est en forme normale de négation et qui ne contient que des conjonctions et des disjonctions. On construit un ensemble $\texttt{clause}(P)$ qui contient éventuellement plus de variables.

On procède par récurrence sur la structure de la formule.

— Si $P = P_1 \wedge P_2$ alors il suffit de décomposer P_1 et P_2 en clauses et de les rassembler : $\texttt{clause}(P) = \texttt{clause}(P_1) \cup \texttt{clause}(P_2)$

— Si P est un littéral ou une disjonction de littéraux alors P est déjà une clause et $\texttt{clause}(P) = \{P\}$
— Si $P = P_1 \vee P_2$ mais que P n'est pas une disjonction de littéraux, cela signifie que P contient au moins une conjonction. La formule P est logiquement équivalente à $l_1 \vee \ldots \vee l_n \vee (R_1 \wedge Q_1) \vee \ldots \vee (R_p \wedge Q_p)$. On remplace chaque conjonction $R_i \wedge Q_i$ par une nouvelle variable propositionnelle x_i et on ajoute des formules qui disent que $x_i \Rightarrow (R_i \wedge Q_i)$.

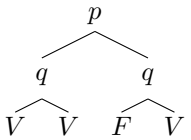
$$\begin{aligned}\texttt{clause}(P) = \; & \{l_1 \vee \ldots \vee l_n \vee x_1 \vee \ldots \vee x_p\} \\ & \cup \texttt{clause}(\neg x_1 \vee R_1) \cup \texttt{clause}(\neg x_1 \vee Q_1) \cup \ldots \\ & \cup \texttt{clause}(\neg x_p \vee R_p) \cup \texttt{clause}(\neg x_p \vee Q_p)\end{aligned}$$

1. appliquer la méthode précédente à la formule
   $P \stackrel{\text{def}}{=} (a_1 \wedge \neg b_1) \vee (a_2 \wedge b_2) \vee (\neg a_3 \wedge (b_3 \vee (a_4 \wedge b_4))) \vee \neg a_5$.
2. Expliquer pourquoi la procédure précédente termine
3. (optionnel) Montrer que si $\texttt{clause}(P)$ est vraie pour une interprétation I alors P est aussi vraie pour cette interprétation et si P est vraie pour une interprétation I alors il existe une interprétation I' qui coincide avec I sur les variables de P et pour laquelle $\texttt{clause}(P)$ est vraie.

Exercice 7 *Connecteur propositionnel IF.* On considère des formules construites à partir de variables propositionnelles, des constantes $\top$ et $\bot$ et d'un seul connecteur logique ternaire $\mathbf{IF}(P, Q, R)$. On appelle $\textsc{Prop}_{\text{IF}}$ l'ensemble des formules ainsi construites. La valeur de $\mathbf{IF}(P, Q, R)$ est égale à la valeur de Q lorsque la valeur de P est vraie et à la valeur de R sinon.

1. Donner des formules équivalentes à $\neg P$, $P \wedge Q$, $P \vee Q$ et $P \Rightarrow Q$ qui n'utilisent que le connecteur **IF**.
2. Décrire par des équations récursives une fonction `valIF` qui étant donnée une interprétation des variables propositionnelles I et une formule P de $\textsc{Prop}_{\text{IF}}$ calcule la valeur de vérité de cette formule.
3. Montrer que les formules $\mathbf{IF}(\mathbf{IF}(P, Q, R), S, T)$ et $\mathbf{IF}(P, \mathbf{IF}(Q, S, T), \mathbf{IF}(R, S, T))$ sont équivalentes.
4. Décrire par des équations récursives une fonction de normalisation `norm` qui étant donnée une formule P de $\textsc{Prop}_{\text{IF}}$ renvoie une formule équivalente dans laquelle toute expression de la forme $\mathbf{IF}(P, Q, R)$ est telle que P est une variable propositionnelle (on dira que la formule est en forme normale).
   On commencera par introduire une fonction `IFn` qui étant données trois formules P, Q et R de $\textsc{Prop}_{\text{IF}}$, en supposant que Q et R sont déjà en forme normale, calcule une formule *en forme normale* équivalente à $\mathbf{IF}(P, Q, R)$.
5. Soit la formule propositionnelle $A \stackrel{\text{def}}{=} ((p \Rightarrow q) \Rightarrow p) \Rightarrow p$.
   (a) Transformer la formule en `IF`-expression.
   (b) Donner une forme normale équivalente.
   (c) Représenter la formule en forme normale comme un arbre binaire dont les nœuds internes portent une étiquette de la forme $\texttt{IF}(x)$ avec x la variable propositionnelle qui apparaît en condition.
   (d) En utilisant la représentation précédente, montrer que A est valide.
6. En déduire une méthode pour décider si une formule de $\textsc{Prop}_{\text{IF}}$ est valide.

Exercice 8 *Arbre de décision binaire.* Un *arbre de décision binaire* (BDT) est un arbre binaire dont les nœuds sont étiquetés par des variables propositionnelles et les feuilles sont formées des constantes booléennes V ou F.



Les variables propositionnelles sont ordonnées et on impose de plus que les variables propositionnelles des fils sont strictement plus grandes que celle du père.

L'arbre peut se voir comme la table de vérité d'une formule propositionnelle (ou encore une fonction booléenne) : lorsque l'on passe un nœud étiqueté par la variable x, le sous arbre gauche correspond au cas où x a la valeur V et le sous-arbre droit au cas où x a la valeur F, dans l'exemple :

p	q	
V	V	V
V	F	V
F	V	F
F	F	V

On peut représenter l'arbre en utilisant un constructeur `Bool` pour les feuilles, associé à une valeur booléenne V ou F et un constructeur à trois arguments $\mathbf{IF}(x, P, Q)$ avec x une variable propositionnelle et P et Q des arbres de décision binaire.

Dans l'exemple cela donne $\mathbf{IF}(p, \mathbf{IF}(q, \texttt{Bool}(V), \texttt{Bool}(V)), \mathbf{IF}(q, \texttt{Bool}(F), \texttt{Bool}(V)))$

1. Donner les équations récursives qui définissent la valeur booléenne $\texttt{vald}(I, t)$ de l'arbre de décision binaire t pour une interprétation I des variables propositionnelles.

$$\begin{aligned} \texttt{vald}(I, \texttt{Bool}(b)) &= \\ \texttt{vald}(I, \mathbf{IF}(x, P, Q)) &= \ldots \texttt{vald}(I, P) \ldots \texttt{vald}(I, Q) \ldots \end{aligned}$$

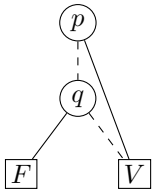
2. Donner les équations récursives qui définissent une formule $\texttt{form}(t)$ avec t un BDT, qui a la même table de vérité et qui de plus est en forme normale de négation et n'utilise que des littéraux, des conjonctions et des disjonctions.
3. Définir par des équations récursives une fonction `notd` qui prend en argument un BDT t qui représente une formule A et renvoie un autre BDT qui représente la formule $\neg A$.
4. On généralise la construction précédente au cas d'une opération binaire `op` sur les formules propositionnelles (comme la conjonction, l'implication).

   Définir par des équations récursives une fonction `opd` qui prend en argument deux BDT t et u qui représentent respectivement les formules A et B et renvoie un autre BDT qui représente la formule $A \,\texttt{op}\, B$. On fera attention à respecter la contrainte sur l'ordre des variables propositionnelles dans les BDT.
5. Que peut-on-dire des feuilles d'un BDT qui correspond à une formule valide ? satisfiable ? insatisfiable ?
6. Proposer un algorithme pour trouver un modèle d'une formule.

Exercice 9 *Diagramme de décision binaire.* Les arbres de décision binaire ont deux défauts : des arbres différents peuvent représenter des formules équivalentes (par exemple $\mathbf{IF}(x, P, P) \equiv P$) et leur taille est en général exponentielle en le nombre de variables propositionnelles.

Un *diagramme de décision binaire* (BDD) est obtenu à partir d'un arbre de décision en assurant de plus les conditions suivantes :

1. *réduction :* aucun nœud n'a deux fils identiques, cela correspond au fait que l'expression $\mathbf{IF}(p, A, A)$ est logiquement équivalente à A et donc on peut remplacer un nœud qui a deux fois le même fils par ce fils ;
2. *partage :* l'arbre est transformé en graphe orienté acyclique (DAG), c'est-à-dire que deux sous-arbres identiques sont partagés. Dans un graphe, il n'y a plus de notion de sous-arbre gauche et de sous-arbre droit : on distingue par un arcs plein le cas où la variable vaut V et par un arc pointillé le cas où la variable vaut F.



3. *variables ordonnées :* comme dans les BDT, on va demander que les variables soient ordonnées : la variable d'un fils est toujours strictement supérieure à la variable du père, on parle alors de diagramme de décision binaire *ordonné* (ou OBDD).

Pour construire un OBDD à partir d'une formule propositionnelle P, il suffit de partir de la plus petite variable x qui apparaît dans la formule et de construire récursivement l'OBDD P_V correspondant à la formule $P[x \leftarrow \top]$ pour la branche "vrai" et l'OBDD P_F correspondant à la formule $P[x \leftarrow \bot]$ pour la branche "faux". Si les deux OBDDs sont les mêmes alors P_V est le résultat attendu, sinon on introduit le nœud x avec un arc plein vers P_V et un arc pointillé vers P_F.

1. Construire les OBDDs des formules $(z \Leftrightarrow t)$ puis $(x \Leftrightarrow y) \wedge (z \Leftrightarrow t)$ en prenant comme ordre $x < y < z < t$.
2. En partant des feuilles, annoter chaque noeud dans les deux OBDD précédents par une formule équivalente qui n'utilise que des littéraux et les symboles $\vee$ et $\wedge$.
3. Montrer que deux formules sont équivalentes si et seulement si elles sont représentées par le même OBDD. En déduire comment, étant donné un OBDD, on peut savoir si la formule associée est valide, satisfiable ou insatisfiable.
4. Pour représenter un OBDD, on associe un numéro à chaque sommet du graphe et on utilise une table G qui permet de stocker les variables et les arêtes. On dispose des opérations suivantes :
   — `noeud`(G, n) : teste si n est un sommet du graphe qui correspond à un noeud interne ;
   — `feuille`(G, n) : teste si n est un sommet du graphe qui correspond à une feuille (V ou F) ;
   — pour un nœud interne : `var`(G, n) donne la variable propositionnelles associée, `vrai`(G, n) est le numéro du sommet qui correspond au cas où la variable est vraie et `faux`(G, n) est le numéro du sommet qui correspond au cas où la variable est fausse ;
   — pour une feuille : `valeur`(G, n) renvoie un booléen correspondant à la valeur V ou F de la feuille.

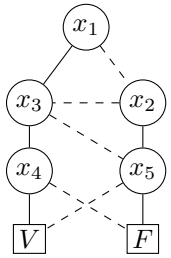
   (a) Exprimer à l'aide des fonctions ci-dessus, les propriétés sur la table G qui assurent que le graphe correspond bien à un OBDD : partage maximal, réduction et ordre sur les variables propositionnelles.

   (b) Soit un graphe G correspondant à un OBDD et un sommet n, comment transformer la fonction `form` de la question 2 de l'exercice 8 en une fonction `form`(G, n) qui calcule la formule logique associée ? comment faire pour exploiter le partage et ne pas recalculer deux fois le même résultat ?

   (c) Expliquer comment construire une fonction `if` qui étant donnés un graphe G correspondant à un OBDD, une variable propositionnelle x et deux sommets n et m de G, renvoie un graphe G' et un noeud p, tel que `form`$(G', p) \equiv \mathbf{IF}(x, \texttt{form}(G, n), \texttt{form}(G, m))$ et `form`$(G', k) =$ `form`(G, k) pour tous les sommets de G (G' contient possiblement un sommet de plus que G, mais ne modifie pas les sommets de G). On utilisera une table T qui permet de retrouver si une formule est déjà représentée dans le graphe : il suffit de donner le nom de la variable et les sommets correspondant au cas vrai et au cas faux, si le nœud apparait dans le graphe, la table donnera le sommet correspondant.

   (d) On veut compter le nombre de modèles d'une formule représentée par un OBDD.

      i. Soit l'OBDD suivant sur les variables propositionnelles x_1, x_2, x_3, x_4, x_5 :



      annoter chaque nœud n du graphe pour donner le nombre d'interprétations concernant les variables plus grandes que celle du nœud n qui satisfont la formule associée à n.

      ii. On suppose donnée une fonction `indice` qui calcule pour chaque sommet de l'OBDD un entier : si le sommet est un nœud, l'indice est la position de la variable correspondante dans l'ordre (la plus petite variable a le numéro 1) et si le sommet est une feuille, l'indice est $n + 1$ avec n le nombre total de variables.
      Construire une fonction `nbsat` qui étant donné un OBDD représenté par un graphe G et un sommet n, retourne le nombre de modèles de la formule en ne considérant que les variables plus grandes que celle du sommet.

      iii. Sachant que la variable racine d'un OBDD n'est pas forcément la plus petite variable, en déduire le nombre de modèles de la formule en considérant toutes les variables.