Feuille 5 - - Skolémisation, BDD

Exercice 1 Mettre sous forme clausale l'ensemble de formules $E = \{F_1, F_2, F_3\}$ où

1. $F_1 = \forall x, (p(x) \Rightarrow \exists y, \forall x, q(x, y))$
2. $F_2 = ((\exists x, (p(x) \Rightarrow r(x))) \vee \forall y,\ p(y)) \wedge \forall x, \exists y, (r(y) \Rightarrow p(x))$
3. $F_3 = ((\forall x, (p(x) \Rightarrow \exists y, q(y))) \Rightarrow \exists z, r(z)) \Rightarrow \exists u, s(u)$

Exercice 2 *Satisfiabilité, examen 2 2013*
Soit la formule $A \stackrel{\text{def}}{=} (\exists x, \neg P(x)) \Rightarrow ((\forall x, P(x) \vee Q(x)) \Rightarrow \forall x, Q(x))$

1. Mettre la formule A en forme normale de négation (les négations ne portent que sur les prédicats et il n'y a plus de connecteur d'implication).
2. Soit B la formule obtenue à la question précédente, dire si les affirmations suivantes sont vraies ou fausses (sans les justifier, c'est du cours !)
   — $A \models B$
   — $B \models A$
   — A est satisfiable si et seulement si B l'est.
3. Eliminer dans B les quantificateurs existentiels ($\exists$) en utilisant la skolémisation et mettre les quantificateurs universels ($\forall$) en tête de la formule.
4. Soit C la formule obtenue à la question précédente, dire si les affirmations suivantes sont vraies ou fausses (sans les justifier, c'est du cours !)
   — $B \models C$
   — $C \models B$
   — B est satisfiable si et seulement si C l'est.
5. La formule C est-elle vraie dans un modèle dont le domaine a exactement un élément ? Même question pour A, justifier vos réponses.
6. La formule A est-elle satisfiable ?
7. Mettre la formule $\neg A$ en forme clausale.
8. Indiquer le domaine et la base de Herbrand pour la forme clausale ainsi obtenue.
9. La formule $\neg A$ est-elle satisfiable ? La formule A est-elle valide ?

Exercice 3 *Forme de Herbrand*

On rappelle que le théorème de Skolem dit que pour toute formule close A, il existe une formule B de la forme $\forall x_1 \dots x_n, C$ avec C sans quantificateur telle que les trois propriétés suivantes soient vérifiées

1. $B \models A$,
2. si A est satisfiable alors B est satisfiable
3. si A est insatisfiable alors il existe des substitutions closes $\sigma_1, \dots, \sigma_k$ telles que $\{C[\sigma_1], \dots, C[\sigma_k]\}$ soit insatisfiable.

A partir de ce résultat, et de la correspondance entre validité d'une formule et insatisfiabilité de sa négation, montrer le résultat dual suivant

1. Pour une formule P close quelconque, montrer qu'il existe une formule Q (appelée forme de Herbrand de P) qui s'écrit $\exists x_1 \dots x_n, R$ avec R sans quantificateur telle que les trois propriétés suivantes sont vérifiées
   (a) $P \models Q$,

(b) si Q est valide alors P est valide

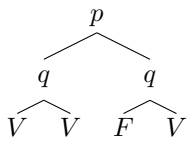
(c) si P est valide alors il existe des substitutions closes $\sigma_1, \ldots, \sigma_k$ telles que la formule $R[\sigma_1] \vee \ldots \vee R[\sigma_k]$ soit valide.

2. Montrer que $R[\sigma_1] \vee \ldots \vee R[\sigma_k] \Rightarrow Q$ est une formule valide.
3. Soit la formule $P \stackrel{\text{def}}{=} (T(a) \vee T(b)) \Rightarrow \exists x, T(x)$, trouver la forme de Herbrand associée ainsi que des substitutions closes $\sigma_1, \ldots, \sigma_k$ telles que $R[\sigma_1] \vee \ldots \vee R[\sigma_k]$ soit valide. Peut-on trouver une solution à ce problème avec une seule substitution.

Exercice 4 *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 5 *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.

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

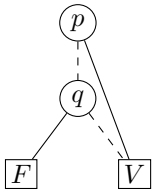
2. Donner les équations récursives qui définissent une formule $\texttt{form}(t)$ avec t un BDT, qui a la même table de vérité et qui de plus est en forme normale de négation et n'utilise que des littéraux, des conjonctions et des disjonctions.
3. Définir par des équations récursives une fonction `notd` 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 6 *Diagramme de décision binaire.* Les arbres de décision binaire ont deux défauts : des arbres différents peuvent représenter des formules équivalentes (par exemple $\textbf{IF}(x, P, P) \equiv P$) 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 $\textbf{IF}(p, A, A)$ 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 $P[x]\top$ pour la branche "vrai" et l'OBDD P_F correspondant à la formule $P[x]\bot$ 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.

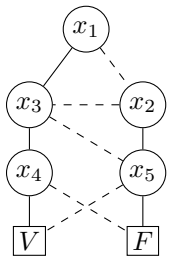
(a) Exprimer à l'aide des fonctions ci-dessus, les propriétés sur la table G qui assurent que le graphe correspond bien à un OBDD : partage maximal, réduction et ordre sur les variables propositionnelles.

(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 5 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`$(G', k) = \texttt{form}(G, k)$ 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.