TD n° 3

Calcul Propositionnel - Équivalences, algorithmes, modélisation

Équivalences

Soient $\varphi, \theta \in \mathscr{F}_{\mathrm{cp}}$ et $q \in \mathrm{PROP}$; la substitution, dans la formule φ, de la variable q par la formule θ, notée $\varphi_{[q \leftarrow \theta]}$, se définit par induction sur la structure d'une formule selon les cas suivants :

$$\begin{aligned} p_{[q\leftarrow\theta]} &:= \begin{cases} \theta\,, & \text{si } p = q\,, \\ p\,, & \text{sinon}\,, \end{cases} \\ (\psi \circ \psi')_{[q\leftarrow\theta]} &:= \psi_{[q\leftarrow\theta]} \circ \psi'_{[q\leftarrow\theta]}\,,\ \text{ou} \circ \in \{\wedge, \vee, \Rightarrow\}\,, \\ (\neg\psi)_{[q\leftarrow\theta]} &:= \neg(\psi_{[q\leftarrow\theta]})\,. \end{aligned}$$

Exercice 3.1. Ecrivez explicitement $\varphi_{[p\leftarrow\theta]}, \varphi_{[q\leftarrow\theta]}$, et $\varphi_{[r\leftarrow\theta]}$, où

1. $\varphi := [(q \vee \neg p) \Rightarrow r] \wedge [r \Rightarrow (\neg p \vee q)]$ et $\theta := \neg\neg q \vee \neg p$,
2. $\varphi := [(q \vee \neg p) \Rightarrow (\neg\neg q \vee \neg p)] \wedge [(\neg\neg q \vee \neg p) \Rightarrow r]$ et $\theta := \neg p \vee q$.

Exercice 3.2. 1. Argumentez que la relation $\equiv$ entre formules propositionnelles est réflexive symétrique et transitive (c'est-à-dire, c'est une relation d'équivalence).

2. En utilisant une suite d'équivalences, montrez que la formule

$$\varphi := [(q \vee \neg p) \Rightarrow (\neg\neg q \vee \neg p)] \wedge \neg[(\neg\neg p \vee \neg q) \wedge (q \wedge \neg p)]$$

est équivalente à $\top$ (c'est-à-dire, elle est une tautologie). Justifiez votre calcul, en faisant appel, à chaque équivalence, à une des équivalences classiques entre formules propositionnelles vues en cours.

Exercice 3.3. Considérez cet enoncé : *si $\theta \equiv \theta'$, alors $\varphi_{[q\leftarrow\theta]} \equiv \varphi_{[q\leftarrow\theta']}$*. Prouvez que l'enoncé est vrai, pour tout $\varphi, \theta, \theta' \in \mathscr{F}_{\mathrm{cp}}$ et $q \in \mathrm{PROP}$. (Conseil : par induction sur la structure de φ, ...)

Formes normales

Exercice 3.4. Calculer une forme clausale (conjonctive) de chacune des formules suivantes :

1. $\psi_1 = (p \wedge \neg((q \vee r) \Rightarrow p)) \vee s$;
2. $\psi_2 = (p_1 \wedge q_1) \vee (p_2 \wedge q_2)$;
3. $\psi_3 = \neg((p \Leftrightarrow q) \Rightarrow (r \Rightarrow s))$.

Algorithmes

Exercice 3.5. *(Algorithme de Quine).* Transformez la formule suivante :

$$\varphi = (p \Rightarrow ((q \vee r) \wedge s)) \wedge \neg(q \Rightarrow (r \wedge (p \vee s)))$$

en forme normale conjonctive et appliquez ensuite l'algorithme de Quine pour trouver les modèles de φ.

Remarque : une forme normale de φ :

$$(\neg p \vee q \vee r) \wedge (\neg p \vee s) \wedge q \wedge (\neg r \vee \neg p) \wedge (\neg r \vee \neg s)\,.$$

Exercice 3.6. *(Algorithme DPLL).* Transformez la formule suivante :

$$\varphi = \neg[(p \Rightarrow s) \Rightarrow ((q \Rightarrow r) \Rightarrow ((p \vee q) \Rightarrow (s \wedge q \wedge r \wedge \neg p)))]$$

en FNC et appliquez ensuite l'algorithme de DPLL pour trouver un modèle. Répétez ensuite l'exercice avec la formule de l'exercice 3.5.

Remarque : une forme normale de φ :

$$(\neg p \vee s) \wedge (\neg q \vee r) \wedge (p \vee q) \wedge (\neg s \vee \neg q \vee \neg r \vee p)\,.$$

COMPLÉMENTS SUR LES ALGORITHMES

Exercice 3.7. Une clause C_0 subsume une clause C_1 quand tout littéral de C_0 est aussi un littéral de C_1. On écrit alors $C_0 \sqsubseteq C_1$.

— Montrez $C_0 \sqsubseteq C_1$ implique $C_0 \models C_1$.
— Montrez que, lorsqu'on veut calculer un modèle de $\mathscr{C}$, on peut retirer de $\mathscr{C}$ toute clause C_1 telle qu'il existe une clause $C_0 \in \mathscr{C}$ telle que $C_0 \sqsubseteq C_1$.

Exercice 3.8. Considérez l'algorithme suivant pour résoudre le problème SAT :

INPUT : un ensemble de clauses $\mathscr{C}$ OUTPUT : vrai si l'ensemble $\mathscr{C}$ est satisfaisable, faux sinon

Soit $Prop(\mathscr{C})$ l'ensemble des variables propositionnelles dans $\mathscr{C}$.
Pour toute valuation $v : Prop(\mathscr{C}) \to \{0,1\}$, faire :
    pour tout $C \in \mathscr{C}$, faire
        si $v(C) = 0$, retourner faux
retourner vrai

Est ce que l'algorithme de Quine a un quelque avantage par rapport à l'algorithme décrit ci-dessus ? Idée : si $Prop(\mathscr{C}) = n$, combien d'itérations sont faites par l'algorithme ci-dessus ? Par ailleurs, combien sont les noeuds de l'arbre de Herbrad de profondeur maximum n ?

Exercice 3.9. Pour tout $n > m \geq 1$, soit

$$\mathscr{C}_{n,m} := \{\, p_n, p_n \Rightarrow p_{n-1}, \ldots, p_{m+1} \Rightarrow p_m \,\}\,.$$

Nous souhaitons estimer la performance de l'algorithme de Quine avec $\mathscr{C}_{n,1}$ en entrée. Nous allons donc compter le nombre des noeuds de l'arbre de Herbrand visités par l'algorithme de Quine avec entrée $\mathscr{C}_{n,m}$.

1. Donnez une formule pour la fonction $f(k)$ qui compte les noeuds de l'arbre de Herbrand visités par l'algorithme avec entrée $\mathscr{C}_{n,n-k} \cup \{\, \neg p_{n-k} \,\}$.
2. Donnez une formule pour la fonction $g(k)$ qui compte les noeuds de l'arbre de Herbrand visités par l'algorithme avec entrée $\mathscr{C}_{n,n-k}$.

MODÉLISATION

Exercice 3.10. Quatre cartes sont placées en carré. Elles ont quatre valeurs différentes : as, roi, dame, valet et quatre couleurs différentes : trèfle, carreau, coeur, pique, mais pas nécessairement dans cet ordre. Il s'agit d'associer à chaque valeur une couleur et un emplacement, en respectant les contraintes suivantes :

(a) L'as est en haut à gauche.
(b) Le pique est en bas à droite.
(c) La dame est en haut à droite.
(d) Le roi est un roi de trèfle.
(e) L'as n'est pas un as de carreau.

1. Représenter le problème sous un ensemble de clauses. Vous choisirez et définirez vous-même les propositions.
2. Déterminer s'il existe un modèle de cet ensemble de clauses.

Algorithme de Quine
entrée : un ensemble de clauses $\mathcal{C}$ sortie : $vrai$ si $\mathcal{C}$ est satisfaisable ou $faux$ sinon
simplifier l'ensemble de clauses (cf. Remarque 1.70) ; si $\mathcal{C} = \varnothing$ retourner $vrai$ si $\mathcal{C}$ contient la clause $\perp$ retourner $faux$ choisir le prochain $p \in \textsc{Prop}$ apparaissant dans une clause si $Quine(\mathcal{C}\langle p \leftarrow \perp\rangle) = vrai$ alors retourner vrai sinon retourner $Quine(\mathcal{C}\langle p \leftarrow \top\rangle)$

Algorithme $DPLL$	
entrée : un ensemble de clauses $\mathcal{C}$ sortie : $vrai$ si $\mathcal{C}$ est satisfaisable ou $faux$ sinon	
simplifier l'ensemble de clauses (cf. Remarque 1.70) ; si $\mathcal{C} = \varnothing$ retourner $vrai$ si $\mathcal{C}$ contient la clause $\perp$ retourner $faux$	
si $\mathcal{C}$ contient la clause unitaire ℓ retourner $DPLL(\mathcal{C}\langle \ell \leftarrow \top\rangle)$	(propagation des contraintes booléennes)
choisir un littéral ℓ depuis une clause et $x, y \in \{\top, \perp\}$, $x \neq y$ **avec la bonne heuristique !**	(décision des littéraux)
si $DPLL(\mathcal{C}\langle \ell \leftarrow x\rangle) = vrai$ alors retourner vrai	
sinon retourner $DPLL(\mathcal{C}\langle \ell \leftarrow y\rangle)$	(backtracking)

Heuristiques. Les heuristiques sont très importantes car elles permettent de réduire rapidement la taille de l'arbre de recherche. Parmi les heuristiques possibles :

Littéraux purs. Si une variable propositionnelle apparaît seulement sous forme positive ou seulement sous forme négative alors ses littéraux sont dits purs. On choisit un littéral pur ℓ parmi ceux qui apparaissent dans le plus de clauses, et on choisit x tel que $\ell[\ell \leftarrow x] = \top$.

Littéraux fréquents. On appelle cette heuristique DLIS, acronyme pour l'anglais « dynamic largest individual sum of literals ». Elle consiste à choisir un littéral parmi ceux apparaissant les plus ; choisir $x = \top$.

Littéraux "courts". On appelle cette heuristique MOM's, acronyme pour l'anglais « maximum occurrence in clauses of minimum size ». Elle consiste à choisir un littéral parmi ceux apparaissant les plus, dans les clauses les plus courtes ; choisir $x = \perp$.

Fréquence des variables. Choisir un symboles propositionnels parmi ceux qui apparaissent le plus, dans les clauses les plus courtes ; choisir $x = \perp$.