Devoir de Logique – Licence - IUP2 2000 / 2001

Votre fichier contenant votre programme doit être envoyé par e-mail à vos chargés de TD le 22 décembre à 15 heures au plus tard. Pour cela :

1. **Créer un répertoire séparé qui ne contient que votre devoir.**
2. **Pour envoyer votre devoir à votre chargé de TD, se placer dans le répertoire contenant votre devoir et lancer la commande "~treinen/envoiedevoir". Cette commande envoye tout le contenu du répertoire actuel à votre chargé de TD.**

Le devoir doit être fait en binôme. Le fichier doit avoir comme nom le nom du binôme des auteurs, qui doit aussi apparaître en commentaire. En tête, indiquer aussi en commentaire le langage de programmation utilisé.

Attention : aucun devoir en retard ne sera corrigé. Le 22-12-2000, après 15 heures, un corrigé sera disponible sur la page web du cours de logique.

Exercice 1
Le but de ce devoir est l'implantation d'un algorithme de décision du problème de la satisfiabilité d'une formule de la logique booléenne fondé sur un système de preuve dit des *tableaux* (R. Smullyan, M. Fitting).

Le système de preuve considéré, que nous appellerons ici Tab, est un système de *réfutation.* Il manipule des *ensembles de formules en forme normale de négation* écrites à l'aides des connecteurs $\neg$, $\vee$, $\wedge$ et ne contenant pas (pour simplifier) les constantes logiques $\mathcal{V}$ et $\mathcal{F}$.

Une formule propositionnelle A est dite être en *forme normale de négation (fnn)* si et seulement si les seuls connecteurs sont $\neg$, $\vee$, $\wedge$ et toute occurrence éventuelle du connecteur $\neg$ s'applique seulement à des sous-formules de A qui sont atomiques.

Une *déduction* dans Tab d'un ensemble de formules $Conc$ à partir d'un ensemble d'ensembles de formules Hyp est un arbre binaire tel que :

1. tout nœud est un ensemble de formules en forme normale de négation ;
2. la racine est l'ensemble $Conc$;
3. les feuilles sont des éléments de Hyp ou bien des axiomes de Tab ;
4. si un noeud n a un seul fils n_1, alors n est un ensemble de formules Γ tel que
$$\frac{\Gamma_1}{\Gamma}$$
est une instance d'une règle d'inférence de Tab, et n_1 est Γ_1 ;
5. si un noeud n a deux fils n_1 et n_2 alors n est une ensemble de formules Γ tel que
$$\frac{\Gamma_1,\ \Gamma_2}{\Gamma}$$
est une instance d'une règle d'inférence binaire du système Tab, n_1 est Γ_1 et n_2 est Γ_2.

Un tel arbre est dit être un *tableau pour Conc*.

Une branche d'un tableau est dite *close* si sa feuille est un axiome. Un tableau pour $Conc$ est *clos* si toute branche est close. Une *réfutation* par tableaux d'une formule A est un tableau clos pour $\{A\}$.

Un *axiome* de Tab a la forme

$$\frac{}{\Gamma, p, \ \neg p}\ Ax$$

où Γ est un est *ensemble* fini (éventuellement vide) de formules en *fnn* et p est une lettre propositionnelle quelconque.

Les autres règles d'inférence de Tab sont donnés dans la figure 1, où Γ dénote un ensemble fini (éventuellement vide) de formules en fnn, A et B des formules quelconque en fnn :

$\dfrac{\Gamma, A, B}{\Gamma, A \wedge B}\ (\alpha)$	$\dfrac{\Gamma, A \quad \Gamma, B}{\Gamma, A \vee B}\ (\beta)$

FIG. 1 – Règles pour $\wedge$ et $\vee$ du Calcul Tab

Dans toute règle d'inférence, si Γ est un ensemble de formules, et A est une formule, la notation Γ, A indique l'ensemble de formules $\Gamma \cup \{A\}$.

Voici un exemple de réfutation de la formule (insatisfiable) $(p \ \wedge \ q) \ \ \wedge \ (\neg p \ \vee \ \neg q)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p, \ \ q, \ \ \neg p}\ Ax \qquad \dfrac{}{p, \ \ q, \ \ \neg q}\ Ax}{p, \ \ q, \ \ \ \neg p \ \vee \ \neg q}\ \beta}{p \ \wedge \ q, \ \ \ \neg p \ \vee \ \neg q}\ \alpha}{(p \ \wedge \ q) \ \ \wedge \ (\neg p \ \vee \ \neg q)}\ \alpha$$

Il facile de démontrer que, étant donnée une formule quelconque A en fnn, A est insatisfiable si et seulement si il existe une réfutation par tableaux de A (voir les deux derniers exercices du partiel de logique).

D'autre part, toute formule A écrite à l'aide des 5 connecteurs habituels (sans utiliser ni $\mathcal{V}$ ni $\mathcal{F}$) peut se réécrire en une formule logiquement équivalente A^{nn} qui est en fnn (traduire $\rightarrow$ et $\leftrightarrow$ et voir le quatrième exercice du partiel).

Or, puisque toute construction d'un tableau pour A termine forcement (pourquoi ? Indiquez le en commentaire, dans votre fichier), on peut utiliser le système Tab pour décider si une formule A (écrite à l'aide des 5 connecteurs habituels, sans utiliser ni $\mathcal{V}$ ni $\mathcal{F}$) est insatisfiable ou pas : tout d'abord, on calcule une A^{nn} équivalente, puis on teste s'il existe ou pas une réfutation par tableaux de A^{nn}.

Dans ce devoir, on vous demande de programmer, dans un langage de programmation de votre choix parmi **C, Caml, Java**, les fonctions suivantes :

1. la fonction *fnn*, qui, étant donnée une formule A quelconque écrite à l'aides des connecteurs $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ et ne contenant pas les constantes logiques $\mathcal{V}$ et $\mathcal{F}$, retourne une

formule A^{nn} (contenant exclusivement les connecteurs $\neg$, $\vee$, $\wedge$) **en fnn** et logiquement équivalente à A ;

2. la fonction *réfutable ?*, qui, étant donnée une formule (contenant exclusivement les connecteurs $\neg$, $\vee$, $\wedge$) A en *fnn*, teste s'il existe ou pas une réfutation par tableaux de A ;
3. la fonction *sat ?*, qui, étant donnée une formule A quelconque écrite à l'aides des connecteurs $\neg$, $\vee$, $\wedge$ et ne contenant pas les constantes logiques $\mathcal{V}$ et $\mathcal{F}$, teste si A est satisfiable ou pas (en utilisant les deux premières fonctions) ;
4. la fonction *modèle* qui, étant donnée une formule A quelconque écrite à l'aides des connecteurs $\neg$, $\vee$, $\wedge$ et ne contenant pas les constantes logiques $\mathcal{V}$ et $\mathcal{F}$, construit un tableau pour A et il en extrait un modèle de A *dans le cas où A est satisfiable*, (et seulement dans ce cas la !).

5. Votre programme lira sur son entrée standard une formule respectant la grammaire suivante :
```
F  ::=  ( F | F )
     |  ( F & F )
     |  ( F -> F )
     |  ( F <-> F )
     |  ( ! F )
     |  ident
```
   où *ident* représente les identificateurs bâtis sur les lettres (majuscules et minuscules), les chiffres et le caractère «`_`». Les blancs, tabulations et retour chariots peuvent êtres utilisés comme séparateur (en particulier, la formule peut être écrite sur plusieurs lignes).
6. Il construira une représentation de cette formule et en calculera une forme normale de négation.
7. Il écrira sur sa sortie standard la chaîne «`FNN`» suivie d'un retour chariot puis de la forme normale de négation calculée suivie d'un retour chariot
8. Il testera ensuite avec la méthode des tableaux si cette formule est satisfiable ou non.
9. Il écrira ensuite sur sa sortie standard la chaîne «`INSATISFIABLE`» suivie d'un retour chariot si celle-ci n'est pas satisfiable et la chaîne «`SATISFIABLE`» si elle n'est pas insatisfiable.
10. Si la formule est satisfiable, alors il calculera une liste $(l_1, v_1), \ldots (l_n, v_n)$ avec l_i lettre propositionnelle et v_i valeur de vérité (c'est-à-dire «`V`» ou «`F`») telle que si pour tout i, on met l_i à la valeur v_i, la formule donnée initialement est vraie. Il affichera cette liste en écrivant chaque l_i suivi d'un espace puis de v_i puis d'un retour chariot sur sa sortie standard.

Dans votre fichier, indiquez en commentaire l'algorithme suivi par la fonction *modèle* et justifiez sa correction par rapport à la spécification donnée. *En général, donnez des commentaires claires et pertinentes, spécifiant toute fonction (ou procédure) définie et justifiant vos choix de structures de données et d'algorithmes.*

ATTENTION : il est très important de respecter à la lettre les spécifications ci-dessus, car vos programmes vont être testés de façon automatique. La composante de la fonction du point 6 ci-dessus, permettant d'analyser une chaîne de caractères et de construire la représentation correspondante vous est **fournie***, tant en C, qu'en Caml qu'en Java : consulter la page web du cours :*
`http : //www.lri.fr/ENSEIGNANTS/LM/logique/`
En général, consulter cette page pour toute autre information éventuelle concernant ce dévoir.