TP9 : Logique propositionnelle

Christophe Rose

Vendredi 6 juin 2008

christophe.rose@ens.fr

http ://www.eleves.ens.fr/home/rose/caml

Dans ce TP, nous allons implémenter la logique propositionnelle pour résoudre quelques problèmes.

1 Propositions logiques

Pour créer des propositions logiques, on utilise le type suivant.

```
type prop = Var of int
| True
| False
| Not of prop
| Or of prop * prop
| And of prop * prop;;
```

Ainsi, la proposition $\neg(x_1 \vee x_2) \wedge x_0$ s'écrit `And (Not (Or (Var 1, Var 2)), Var 0)`.

Dans tout le reste du TP, on supposera que les indices des variables sont tous supérieurs ou égaux à 1.

Les propositions logiques étant peu lisibles, nous allons écrire une fonction pour les visualiser.

On rappelle que `str1 ^ str2` est le concaténé de deux chaînes de caractères, et que `string_of_int` transforme un entier en une chaîne de caractères.

Question 1. Écrivez une fonction `show` qui a le comportement suivant. On devra obtenir ceci :

```
#show (Or(Var 1,False)) ;;
- string = "(x1 v F)"
#show (And(Var 1,Not(Var 2)));;
- string = "(x1 ^ !x2)"
```

Question 2. Écrivez une fonction `multi_or` de type `prop list -> prop` qui à une liste de propositions logiques $P_1, \ldots, P_n$ renvoie la proposition $P_1 \vee \ldots \vee P_n$. Faites-en de même pour la fonction `multi_and`, qui renvoie $P_1 \wedge \ldots \wedge P_n$.

Question 3. Écrivez la fonction `imp` de type `prop * prop -> prop` qui à partir de deux propositions P_1 et P_2 donnent une proposition logiquement équivalente à $P_1 \Rightarrow P_2$. Même chose pour `equiv` et `xor` qui donnent respectivement $P_1 \Leftrightarrow P_2$ et $P_1 \otimes P_2$.

Question 4. Écrivez une fonction récursive `eval` de type `prop -> bool vect -> bool` qui évalue une proposition logique lorsque les valeurs des variables sont données par le tableau.

2 Tables de vérité

Question 5. Écrivez une fonction récursive `indice_max` qui renvoie l'indice de variable maximal rencontré dans une proposition logique.

Pour pouvoir prendre en compte tous les tableaux possibles (avec x_0 faux), nous allons utiliser une fonction `incremente_tab`, qui modifie un tableau sans renvoyer sa valeur.

Ainsi, on passe successivement du tableau `[|false;false;false|]` aux tableaux `[|false;true;false|]`, `[|false;false;true|]`, puis `[|false;true;true|]`. En effet, on commence par modifier les indices les plus petits.

Pour savoir quand on a fini, cette fonction renvoie `false` lorsqu'on l'utilise sur le dernier tableau possible, constitué d'un `false` en 0 et de `true` partout ailleurs. Elle renvoie `true` sinon.

```
let incremente_tab tab =
  let i = ref 1 in
    while (!i < vect_length tab) && tab.(!i) do
      tab.(!i)<- false;
      incr i;
    done;
    if !i < vect_length tab
    then (tab.(!i)<-true;true)
    else (false) ;;
```

Question 6. Écrivez une fonction `possibles` qui renvoie la liste des tableaux satisfaisant une proposition logique. Le premier tableau sera évalué hors de la boucle. On copiera les tableaux à l'aide de la fonction `copy_vect`.

Question 7. Écrivez deux fonctions `satisfiable` et `tautologie` qui déterminent respectivement si une proposition logique est satisfiable et si c'est une tautologie.

Question 8. Vérifier que les propositions logiques suivantes sont des tautologies :

- $\neg\neg x_1 \Rightarrow x_1$,
- $(x_1 \Rightarrow x_2) \Leftrightarrow (\neg x_2 \Rightarrow \neg x_1)$,
- $(x_1 \Rightarrow x_2) \wedge (x_2 \Rightarrow x_3) \Rightarrow (x_1 \Rightarrow x_3)$.

Question 9. Écrire une fonction récursive `une_seule` de type `prop list -> prop`, qui à une liste de propositions logiques renvoie une proposition qui est vraie si et seulement s'il y a exactement une des propositions de la liste qui est vraie.

3 Problèmes

Un chevalier doit partir délivrer des princesses. Arrivé à une intersection, il a le choix entre trois chemins, chacun précédé d'un panneau. Le gardien des lieux lui déclare : « Parmi ces troix chemins, l'un mène à une princesse, et son panneau dit la vérité. Quant aux deux autres, ils aboutissent à une mort certaine. Au moins l'un des panneaux ment. » Voici ce qui est écrit à l'entrée de chaque chemin :

1. Le deuxième chemin mène à une mort certaine.
2. Ce chemin mène à une mort certaine.
3. Le premier chemin mène à une mort certaine.

Question 10. Les variables x_1, x_2 et x_3 sont utilisées pour savoir si un chemin mène à une princesse. Les variables x_4, x_5 et x_6 sont utilisées pour savoir si un panneau dit la vérité. Utilisez les fonctions abordées précédemment pour trouver quel est le chemin à prendre.

Après avoir trouvé la première princesse, le chevalier continue son périple et se retrouve à une autre intersection, avec cinq chemins cette fois-ci. Le gardien des lieux déclare : « L'un des chemins mène à une princesse et son panneau dit la vérité. Tous les autres mènent à une mort certaine. » Voici ce qui est écrit à l'entrée de chaque chemin :

1. Le premier chemin mène à une mort certaine et pas le quatrième.
2. C'est un chemin de numéro impair qui mène à la princesse.
3. Le deuxième panneau dit la vérité ou le cinquième ment.
4. Ce panneau ment mais pas le premier.
5. Le troisième panneau ment.

Question 11. Quel est le chemin à prendre ?

Le chevalier se retrouve à une intersection à neuf chemins. Le gardien des lieux déclare la même chose que le dernier. Voici ce qui est écrit :

1. C'est un chemin de numéro impair qui mène à la princesse.
2. Ce chemin mène à la princesse.
3. Le cinquième panneau dit la vérité ou le septième ment. De plus, le neuvième ment.
4. Le premier panneau ment.
5. Parmi le deuxième et le quatrième panneau, il y en a au moins un qui dit la vérité.
6. Le troisième panneau ment.
7. Le premier chemin mène à une mort certaine.
8. Ce panneau ment et le neuvième chemin mène à la princesse.
9. Le sixième panneau ment.

Question 12. Quel est le chemin à prendre ?

Question 13. Quelle est la complexité de la fonction `possibles` ?