Université Paris 7 - DEUG-MIAS Année 2004-2005

TD de *Logique et Circuits* n° 8

Calcul Propositionnel

Exercice 1 On note P, M, J les propositions "Pierre est étudiant", "Marie est étudiante" et "Jean est étudiant". Traduire en formules de calcul propositionnel :

- Pierre et Marie sont étudiants
- Marie est étudiante ou Pierre n'est pas étudiant
- Pierre, Marie, et Jean ne sont pas tous les trois étudiants
- Ni Pierre ni Marie ne sont étudiants
- Si Pierre est étudiant, alors Marie et Jean sont étudiants
- Pierre est étudiant si Marie et Jean sont étudiants
- Marie est étudiante si et seulement si Pierre l'est

Exercice 2 Une contrée est peuplée de bons, qui disent toujours la vérité et de méchants, qui mentent toujours.

1. Un méchant peut-il dire "je suis un méchant" ?
2. A déclare : "Je suis méchant ou B est bon". Que sont A et B ?
3. A déclare : "Nous sommes tous méchants". B déclare : "Un seul de nous trois est bon". Que sont A, B et C ?
4. X se tient à l'embranchement de deux routes. Une va au paradis, l'autre en enfer. Quelle question doit-on poser à X pour savoir où est le paradis ?

Exercice 3 Quelle est la formule correspondant à la fonction suivante :

a	b	c	f(a,b,c)
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	F
F	V	V	V
F	V	F	F
F	F	V	V
F	F	F	V

Exercice 4

1. Définir en OCaml un type `formule` correspondant aux formules logiques avec les connecteurs $\wedge, \vee, \neg, \rightarrow$.
2. Écrire en Ocaml une fonction de type `'a -> 'a list list -> 'a list list` qui ajoute un élément à chaque liste d'une liste. Par exemple,

```
# ajout 1 [[2; 3] ; [4; 5]];;
- : int list list = [[1; 2; 3]; [1; 4; 5]]
```

3. Écrire une fonction `combi : int -> bool list list` qui renvoie toutes les combinaisons possibles de n valeurs booléennes. Par exemple, pour $n = 3$, on doit retrouver toutes les combinaisons possibles de a, b, c de l'exercice précédent :

```
# combi 3;;
- : bool list list =
[[true; true; true]; [true; true; false]; [true; false; true];
 [true; false; false]; [false; true; true]; [false; true; false];
 [false; false; true]; [false; false; false]]
```

4. On définit une fonction par une liste de booléens, correspondant aux valeurs que prend la fonction pour les n-uplets dans l'ordre fourni par `combi`. Par exemple, la fonction f de l'exercice précédent s'écrit :

   ```
   val f : bool list = [true; true; true; false; true; false; true; true]
   ```

   Écrire en OCaml la fonction `liste_faux` qui prend en argument une liste de combinaisons (résultat de `combi`) et une fonction et qui renvoie les n-uplets des paramètres pour lesquels f prend la valeur $false$. Par exemple, pour la fonction f de l'exercice précédent :

   ```
   # liste_faux (combi 3) f;;
   - : bool list list = [[false; true; false]; [true; false; false]]
   ```

5. Question subsidiaire pour ceux qui finissent les feuilles de Td ! Écrire une fonction qui prend un nombre de variables et une fonction booléenne comme définie à la question 4 et qui renvoie la formule correspondant à cette fonction. Par exemple,

   ```
   # en_formule 3 f;;
   - : formule =
   N (Ou, N (Et, N (Et, Non (F 0), F 1), Non (F 2)),
    N (Et, N (Et, F 0, Non (F 1)), Non (F 2)))
   ```