Interrogation : Logique et Programmation Fonctionnelle

Licence d'informatique 3.1

——

▸ Exercice 1. Formules

Simplifier le plus possible les formules suivantes (en donnant les équivalences intermédiaires) et repérer les tautologies.

1. $\neg(F \vee G) \rightarrow \neg G$
2. $\neg(F \wedge G) \rightarrow ((\neg F \rightarrow H) \rightarrow (G \wedge \neg F))$
3. $(F \rightarrow G) \wedge (G \vee H \vee F) \wedge (H \rightarrow F)$
4. $\neg(\forall y(\exists z(\neg A(z, y) \wedge B(y))))$

▸ Exercice 2. Séquents

On rappelle les règles de réduction des séquents.

$G_\vee$:	$\frac{\Gamma, F \vDash \Delta \quad \Gamma, G \vDash \Delta}{\Gamma, F \vee G \vDash \Delta}$	$D_\vee$:	$\frac{\Gamma \vDash \Delta, F, G}{\Gamma \vDash \Delta, F \vee G}$	
$G_\wedge$:	$\frac{\Gamma, F, G \vDash \Delta}{\Gamma, F \wedge G \vDash \Delta}$	$D_\wedge$:	$\frac{\Gamma \vDash \Delta, F \quad \Gamma \vDash \Delta, G}{\Gamma \vDash \Delta, F \wedge G}$	
$G_\rightarrow$:	$\frac{\Gamma \vDash \Delta, F \quad \Gamma, G \vDash \Delta}{\Gamma, F \rightarrow G \vDash \Delta}$	$D_\rightarrow$:	$\frac{\Gamma, F \vDash \Delta, G}{\Gamma \vDash \Delta, F \rightarrow G}$	
$G_\neg$:	$\frac{\Gamma \vDash \Delta, F}{\Gamma, \neg F \vDash \Delta}$	$D_\neg$:	$\frac{\Gamma, G \vDash \Delta}{\Gamma \vDash \Delta, \neg G}$	
$G_\forall$:	$\frac{\Gamma, A(t), \forall x A(x) \vDash \Delta}{\Gamma, \forall x A(x) \vDash \Delta}$	$D_\forall$:	$\frac{\Gamma \vDash A(a), \Delta}{\Gamma \vDash \forall x A(x), \Delta}$	où a est une variable fraîche
$G_\exists$:	$\frac{\Gamma, A(a) \vDash \Delta}{\Gamma, \exists x A(x) \vDash \Delta}$	$D_\exists$:	$\frac{\Gamma \vDash A(t), \exists x A(x)}{\Gamma \vDash \Delta, \exists x A(x)}$	et t est un terme

Prouver les formules suivantes par la méthode des séquents.

1. $F \rightarrow ((\neg F \vee \neg G) \rightarrow (F \wedge \neg G))$
2. $(F \rightarrow G) \rightarrow ((\neg G \wedge F) \rightarrow H)$
3. $(\exists x \forall y A(y, x)) \rightarrow (\forall x \exists y A(x, y))$
4. $(\exists x \forall y (A(x, y) \wedge B(y))) \rightarrow (\forall x (B(x) \wedge \exists y A(y, x)))$

▸ **Exercice 3. Récursion**

Écrire le code CAML des fonctions récursives suivantes.

1. la factorielle d'un entier : `val fact : int -> int`.
2. la puissance d'un entier par un autre : `val puiss : int -> int -> int`.
3. la suite de Fibonacci, définie par $u_0 = u_1 = 1$ et $u_n = u_{n-1} + u_{n-2}$ pour $n > 1$: `val fibo : int -> int`.
4. une fonction qui renvoie un couple composé des deux termes consécutifs (u_n, u_{n+1}) de la suite de Fibonacci : `val fibo2 : int -> int * int`.

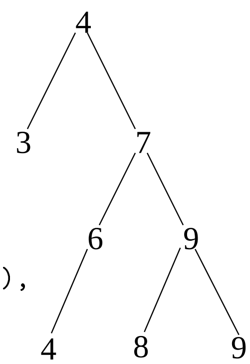
▸ **Exercice 4. Arbres**

On définit le type suivant d'arbres binaires étiquetés par des entiers :

```
type arbre = Vide | Noeud of int * arbre * arbre
```

Ainsi, l'arbre a représenté ci-contre est défini par :

```
let a = Noeud (4, Noeud (3, Vide, Vide),
                  Noeud (7, Noeud (6, Noeud (4, Vide, Vide), Vide),
                            Noeud (9, Noeud (8, Vide, Vide),
                                      Noeud (9, Vide, Vide))))
```



1. On définit la hauteur d'un arbre de la façon suivante : la hauteur de l'arbre vide est 0; celle d'un autre arbre est 1 plus la hauteur maximale de ses deux sous-arbres. Par exemple, la hauteur de a est 4. Écrire une fonction récursive renvoyant la hauteur d'un arbre : `val hauteur : arbre -> int`.
2. Supposons que notre arbre soit trié, c'est à dire que l'étiquette d'un nœud soit strictement supérieure à toutes celles de son fils gauche et inférieure ou égale à celle de son fils droit (c'est le cas dans l'exemple). Écrire une fonction qui renvoie vrai si un entier donné est présent dans un arbre, faux sinon : `val cherche : int -> arbre -> bool`.
3. Écrire une fonction récursive qui renvoie la somme de tous les entiers présents dans l'arbre : `val somme : arbre -> int`.
4. Écrire une fonction récursive qui prend en argument une fonction et un arbre, applique la fonction à chaque nœud de l'arbre, et renvoie l'arbre correspondant : `val map : (int -> int) -> arbre -> arbre`. Par exemple, `map factorielle a` renverrait :

   ```
   Noeud (24, Noeud (6, Vide, Vide),
              Noeud (5040, Noeud (720, Noeud (24, Vide, Vide), Vide),
                           Noeud (362880, Noeud (40320, Vide, Vide),
                                          Noeud (362880, Vide, Vide))))
   ```