

Examen – 13 Septembre 2004

Durée : 3 heures.

Documents autorisés

Exercice 1 En utilisant l'isomorphisme de Curry-Howard, définir une fonction OCaml `foo` représentant une preuve de la proposition $((A \wedge B) \Rightarrow (C \wedge D)) \Rightarrow ((B \wedge A) \Rightarrow (D \wedge C))$. On rappelle que si e_1 et e_2 sont des expressions OCaml de type A et B , alors la paire (e_1, e_2) est une expression représentant la preuve de $A \wedge B$. Que fait cette fonction ?

Exercice 2

Les entiers peuvent être représentés en λ -calcul par des itérateurs de fonction. Cette méthode, due à Church, consiste à “coder” un entier n en un terme, appelé entier de Church, noté \underline{n} , et défini comme suit :

$$\underline{n} \triangleq \lambda f. \lambda x. \underbrace{(f (f (\dots (f x))))}_{n \text{ applications}} = \lambda f. \lambda x. (f^n x)$$

En utilisant le langage OCaml, \underline{n} s'écrit donc :

$$\text{let } \underline{n} = \text{function } f \rightarrow \text{function } x \rightarrow \underbrace{(f (f (\dots (f x))))}_{n \text{ applications}}$$

1. Typer les expressions suivantes (détailler l'arbre d'inférence de typage en explicitant les règles utilisées, le système de contraintes de typage associé à cet arbre, et la résolution de ce système) :

```
let zero = function f -> function x -> x;;
let un = function f -> function x -> (f x);;
let deux = function f -> function x -> (f (f x));;
```

2. On définit la fonction successeur comme suit :

$$\text{succ} \triangleq \lambda n. \lambda f. \lambda x. ((n f) (f x))$$

En utilisant le langage OCaml, `succ` s'écrit donc :

```
let succ = function nc -> function f -> function x -> ((nc f) (f x));;
```

- (a) Typer en détail la fonction `succ`.
 - (b) Montrer que pour tout entier n , le λ -terme $(\text{succ } \underline{n})$ se β -réduit en $\underline{n+1}$.
 - (c) Soit \underline{n} un entier de Church dont la valeur est connue dans l'environnement d'évaluation (`zero`, `un`, ou `deux`, par exemple). Détailler l'évaluation OCaml de l'application $(\text{succ } \underline{n})$.
3. Montrer que pour tout entier n , le type de \underline{n} peut s'écrire $('a \rightarrow 'a) \rightarrow 'a \rightarrow 'a$. Cette preuve s'obtient :
 - en raisonnant par récurrence sur n ,
 - en utilisant les propriétés établies dans les questions précédentes,

- en utilisant la propriété (admise) que le typage est préservé lors de l'évaluation, c'est à dire que si une expression e est de type t , alors le résultat de l'évaluation de e est aussi de type t .
4. Définir une fonction Ocaml `int_to_church` qui, étant donné un entier n , calcule l'entier de Church \underline{n} . Typer en détail cette fonction.
 5. Typer en détail l'expression `(int_to_church 4)`.