IENAC-S-99 Octobre 2000

Examen de logique formelle et calculabilité

Durée : 2 h
Tous documents autorisés

Ce sujet comporte 2 pages.

Logique

1. Soit le système de preuve PM[1] (analogue à la « méthode axiomatique ») constitué des axiomes

$$(p \vee p) \rightarrow p \tag{1}$$
$$q \rightarrow (p \vee q) \tag{2}$$
$$(p \vee q) \rightarrow (q \vee p) \tag{3}$$
$$(q \rightarrow r) \rightarrow ((p \vee q) \rightarrow (p \vee r)) \tag{4}$$

et de la règles d'inférence (*modus ponens*)

$$\frac{\vdash A \quad \vdash A \rightarrow B}{\vdash B}$$

Une démonstration est une suite de formules qui sont soit des instances des axiomes, soit conclusion de la règle d'inférence.

Montrer que le système PM est correct, i.e. que toutes les formules démontrables avec PM sont des tautologies.

2. Montrer en utilisant le principe de résolution et le calcul des séquents que c est déductible des formules suivantes :

$$a \rightarrow (b \rightarrow c) \tag{5}$$
$$a \rightarrow (d \rightarrow b) \tag{6}$$
$$a \tag{7}$$
$$d \tag{8}$$

3. Soit la signature $\Sigma = \{i_2, a_0, b_0\}$ et le symbole de prédicat p_1. Montrer avec la méthode de démonstration de votre choix que $p(i(a, i(i(a, b), b)))$ est conséquence logique des formules suivantes :

$$\forall x\, \forall y\, (p(x) \rightarrow p(y)) \rightarrow p(i(x, y)) \tag{9}$$
$$\forall x\, \forall y\, (p(i(x, y)) \wedge p(x)) \rightarrow p(y) \tag{10}$$

Calculabilité

1. Donner un terme du λ-calcul réalisant la division par deux (2) pour les entiers de CHURCH. Indication : utiliser (en justifiant) une itération sur une paire constituée d'un booléen et d'un entier :

$$< FALSE, n > \quad \rightsquigarrow \quad < TRUE, n >$$
$$< TRUE, n > \quad \rightsquigarrow \quad < FALSE, n+1 >$$

[1] Il s'agit du système proposé dans les *Principia Mathematica* de A. N. WHITEHEAD et B. RUSSELL, publié en 1910

2. Comparer les quatres fonctions suivantes de calcul de la factorielle (`goedel` est un itérateur qui calcule $f(n, f(n-1, ..., f(1, x)))$; quelles sont les versions les plus facile à analyser, i.e. qu'il faut préférer ?

```
let rec goedel = fun n f x -> if n = 0 then x else f n (goedel (n-1) f x);;

let rec fact1 = fun n ->
  if n = 0 then 1 else n * fact1 (n-1);;
let fact2 = fun n ->
  goedel n (fun x y -> x * y) 1;;
let fact3 = fun n ->
  let f = ref 1 in
  for i = 2 to n do f := !f * i done;
  !f;;
let fact4 = fun n ->
  let f = ref 1 and i = ref 2 in
  while !i <= n do
    f := !f * !i;
    incr i
  done;
  !f;;
```