Master informatique 1 8 janvier 2014

# Fondements de la programmation, partiel 2

## 1 Inférence de type en lambda-calcul simplement typé

Les exercices suivants font références au lambda-calcul simplement typé avec paires.

### 1.1 Typage sans justification

Sans justification, donner un type à chacun des termes suivants (lorsque c'est possible) et désigner les termes non typables :

1. $\lambda x.x$
2. $\lambda xf.(f\ (f\ (f\ x)))$
3. $((\lambda x.x)\ (\lambda x.x))$
4. $\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$
5. $\lambda xy.\langle x, y\rangle$
6. $\lambda xf.\langle(f\pi_1 x), (f\pi_2 x)\rangle$

### 1.2 Typage et $\beta$-réduction

Donner un exemple de terme non typable qui se réduit en un terme typable.

### 1.3 Terme d'un type donné

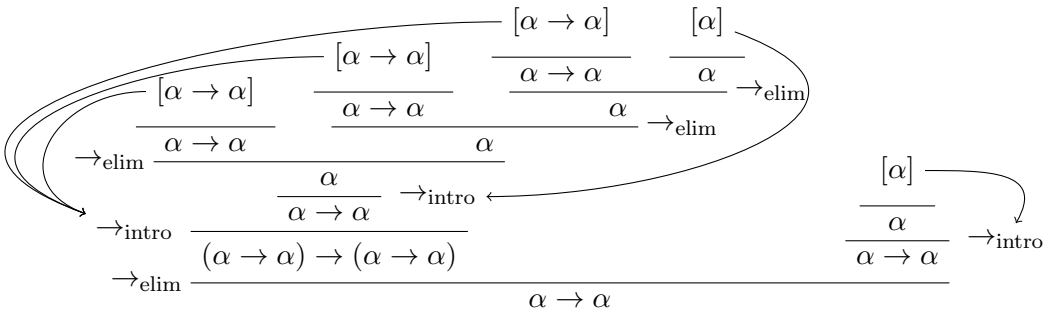
Donner un terme clos (sans variable libre) de type $\alpha \to ((\alpha \to \beta) \to \beta)$.

### 1.4 Inférences de type

1. Montrer par une inférence de type que $\lambda x.x$ est typable de type $\alpha \to \alpha$.
2. Montrer par une inférence de type que $(((\lambda x.x)(\lambda y.y))(\lambda z.z))$ est typable.
3. Montrer par une inférence de type que $\lambda xf.\langle(f\pi_1 x), (f\pi_2 x)\rangle$ est typable.

## 2 Correspondance preuves programmes

Soit la preuve suivante en déduction naturelle.




1. La traduire en un terme du lambda-calcul simplement typé;
2. effectuer la réduction du terme jusqu'à trouver une forme normale;
3. traduire le lambda terme normal obtenu en déduction naturelle.

# 3 Concepts et langages de programmation

## 3.1 Stratégie d'appel

Soit un programme C définissant les fonctions suivantes :

```
int deux() {
  printf("deux()\n");
  return 2;
}

int carre(int n) {
  printf("carre()\n");
  return n * n;
}

int quatre(int x) {
  printf("quatre()\n");
  return 4;
}
```

Si la fonction principale (main) contient seulement l'instruction `carre(deux());` qu'est ce qui sera affiché ? Et si la fonction principale contient seulement l'instruction `quatre(deux());` ? Citer un langage dans lequel l'affichage serait différent et expliquer pourquoi.

## 3.2 First-class functions

Qu'est-ce qui s'affichera dans la console (en javascript) ? Expliquer.

```
function f(y) {
  function g(x) {
      return x + y;
  };
  return g;
}
h = f(2);
t = f(1);
console.log(h(1));
```

# 4 Calculabilité

## 4.1 Analyse syntaxique

Est-il toujours possible, quel que soit le langage de programmation, de décider si un programme source correspond à un arbre syntaxique du langage ? (Expliquer)

## 4.2 Complétude

Est-ce que le lambda-calcul simplement typé permet de représenter toutes les fonctions calculables au sens de Turing ?