Partiel de fondements de la programmation
mercredi 7 janvier 2015

1 Lambda-calcul simplement typé (6 points)

Pour chacun des termes suivants, typer le terme à l'aide d'une inférence de type ou expliquer pourquoi il n'est pas typable.

1. $\lambda xyz.(x\ (y\ z))$
2. $\lambda xy.(x\ (y\ x))$
3. $\lambda xy.((x\ y)\ x)$
4. $\lambda x.(x\ \lambda y.y)$

2 Extension du lambda-calcul (2 point)

1. Le terme `let id` $= (\lambda x.x)$ `in (id unit);` `(id` $\lambda xy.y)$ utilise des extensions du lambda-calcul simplement typé. Lesquelles ?
2. Sans effectuer de réduction, supprimer le sucre syntaxique de ce terme de façon à obtenir un terme du lambda-calcul avec pour seule extension une constante `unit` de type Unit.

3 Machine abstraite de Krivine (7 points)

On rappelle le fonctionnement de la machine abstraite de Krivine (qui est par nom) et de la machine abstraite de Krivine par valeur (un variante de la précédente).

3.1 KAM par nom

On définit les termes, les environnements, les clôtures et les piles en posant:

- terme $t ::= x \mid (t\ t) \mid \lambda x.t$
- environnement $e ::= \emptyset \mid e, x = c$
- clôture $c ::= (t, e)$
- pile $\pi = \varepsilon \mid c : \pi$

L'état de la machine est représenté par un triplet fait d'un terme d'un environnement et d'une pile. Intialement on se donne le terme et l'environnement et la pile sont vides. Une étape de réduction fait passer d'un état à l'état suivant à l'aide des trois règles suivantes :

$$\frac{(t\ u) \quad e \quad \pi}{t \quad e \quad (u, e) : \pi}\ \text{push} \qquad \frac{\lambda x.t \quad e \quad c : \pi}{t \quad e, x = c \quad \pi}\ \text{pop} \qquad \frac{x \quad e, x = (t, e') \quad \pi}{t \quad e' \quad \pi}\ \text{deref}$$

Par exemple, l'évaluation du terme $((\lambda x.x)\ y)$ sur la KAM par nom se déroule ainsi:

$$\dfrac{\dfrac{\dfrac{((\lambda x.x)\ y) \quad \emptyset \quad \varepsilon}{\lambda x.x \quad \emptyset \quad (y,\emptyset)}\ \text{push}}{x \quad x=(y,\emptyset) \quad \varepsilon}\ \text{pop}}{y \quad \emptyset \quad \varepsilon}\ \text{deref}$$

Le résultat est y (plus aucune règle ne s'applique).

3.1.1 Question

Donner l'exécution de la KAM par nom sur le terme $(((\lambda xy.x)\ z)\ z')$.

3.2 KAM par valeur

La KAM par valeur est définie à partir de la KAM par nom en modifiant les piles de façon à ce qu'elles contiennent deux sortes d'éléments (des fonctions ou des arguments) :

- pile $\pi = \varepsilon \mid Fc : \pi \mid Ac : \pi$

et en appliquant les règles suivantes (par priorités décroissantes) :

$$\dfrac{(t\ u) \quad e \quad \pi}{t \quad e \quad A(u,e):\pi}\ \text{push} \qquad \dfrac{\lambda x.t \quad e \quad A(u,e'):\pi}{u \quad e' \quad F(\lambda x.t,e):\pi}\ \text{swap}$$

$$\dfrac{x \quad e, x=(t,e') \quad \pi}{t \quad e' \quad \pi}\ \text{deref} \qquad \dfrac{v \quad e \quad F(\lambda x.t,e'):\pi}{t \quad e', x=(v,e) \quad \pi}\ \text{pop}$$

Dans cette dernière règle, v désigne une valeur c'est à dire ici un lambda-terme qui n'est pas une application (donc une variable ou un lambda).

Par exemple, l'évaluation du terme $((\lambda x.x)\ y)$ sur la KAM par valeur se déroule ainsi:

$$\dfrac{\dfrac{\dfrac{\dfrac{((\lambda x.x)\ y) \quad \emptyset \quad \varepsilon}{\lambda x.x \quad \emptyset \quad A(y,\emptyset)}\ \text{push}}{y \quad \emptyset \quad F(\lambda x.x,\emptyset)}\ \text{swap}}{x \quad x=(y,\emptyset) \quad \varepsilon}\ \text{pop}}{y \quad \emptyset \quad \varepsilon}\ \text{deref}$$

3.2.1 Question

Donner l'exécution de la KAM par valeur sur le terme $(((\lambda xy.x)\ z)\ z')$.

3.2.2 Questions

On veut comparer le temps d'exécution de la KAM par nom et de la KAM par valeur. Pour mesurer ce temps on compte le nombre de règles appliquées dans chaque exécution mais **sans compter** les applications la règle *swap*. Ainsi les exécutions des deux machines sur le terme $((\lambda x.x)\ y)$ prennent autant de temps (3 règles hors règle *swap*).

1. Est-ce encore le cas sur le terme $(((\lambda xy.x)\ z)\ z')$?
2. Les deux machines ont-elles toujours les mêmes temps d'exécution ou bien pouvez-vous trouver un terme pour lequel la KAM par nom est plus rapide et/ou un terme pour lequel la KAM par valeur est plus rapide ? Justifier par un raisonnement ou en donnant des exemples de termes et leurs exécutions.

4 Concepts et langages de programmation

4.1 Questions sur les concepts (3 points)

1. Donner un exemple usuel de fonction d'ordre supérieur, dans le langage de votre choix.
2. Donner un exemple usuel de fonction polymorphe de façon paramétrique.
3. Citer un langage dans lequel on ne peut pas programmer de fonctions récursives.

4.2 Étude de programmes (4 points)

Pour chacun des programmes suivants, déterminer ce qui sera affiché à l'exécution et expliquer pourquoi.

4.2.1 Le programme C carre.c

```
int un() {
  printf("un\n");
  return 1;
}

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

int carre(int x){
  printf("carre de %d\n", x);
  return x*x;
}

int main () {
  if (un() == 1) {
    carre(deux());
  } else {
    carre(un());
  }
  return EXIT_SUCCESS;
}
```

4.2.2 Le programme javascript exp.js

```
function curry(f, x) {
    function g(y) {
      return f(x, y);
    }
   return g;
}

function exp(base, exposant) {
  var res = 1;
  for (var i = 0; i < exposant; i += 1) {
    res *= base;
  }
 return res;
}

g = curry(exp, 2);
h = curry(exp, 3);
j = curry(exp, 4);
/* affichage de la valeur de h(2) */
console.log(h(2));
```

4.2.3 Le programme Racket mono.rkt

```
#lang racket
> (define bkpk "valeur quelconque")
> (define bkp-done #f) ; faux
> (define monomaniac
    (lambda (x)
      (call/cc
       (lambda (k)
         (if bkp-done
             ; then
             (bkpk x)
             ; else
             (begin
               (set! bkpk k)
               (set! bkp-done #t)
               (k x)))))))

> (display
   (+ (* 3 (monomaniac 5)) 1))
; affichage ?
> (display
   (* 100 (monomaniac 2)))
; affichage ?
```