Université Laval
Faculté des sciences et de génie
Département d'informatique et de génie logiciel
GLO-66811

Danny Dubé
Hiver 2007
Version : 7 février

Réponse à la série d'exercices #3
Implantation de la récursion

1. À l'aide de la méthode des blocs d'activation explicites :

```
LAM_f:
  px = arg1; py = arg2;
  if (px == 0) goto THEN_1 else goto ELSE_1;
THEN_1:
  ret_val = py - 2;
  goto do_return;
ELSE_1:
  fct2call = &&LAM_g;
  arg1 = px - 1; arg2 = 2 * py + 1;
  goto do_invoke;

LAM_g:
  pa = arg1; pb = arg2;
  if (pa % 2 == 1) goto THEN_2 else goto ELSE_2;
THEN_2:
  fct2call = &&LAM_g;
  arg1 = pa - 1; arg2 = pb + 2;
  goto do_invoke;
ELSE_2:
  fct2call = &&LAM_h;
  arg1 = pb;
  bp = new BLOC(&&RET_PT1, bp, pa);
  goto do_invoke;
RET_PT1:
  pa = ...; bp = ...;
  fct2call = LAM_f;
  arg1 = pa; arg2 = ret_val;
  goto do_invoke;

LAM_h:
  pt = arg1;
  ret_val = 3 * pt - 1;
  goto do_return;
```

À l'aide de la méthode "trampoline" :

```
(define (f x y)
  (if (= x 0)
      (TNTreturn (- y 2))
      (TNTterm-apply g (- x 1) (+ (* 2 y) 1))))
(define (g a b)
  (if (= (modulo a 2) 1)
      (TNTterm-apply g (- a 1) (+ b 2))
      (TNTterm-apply f a (TNTnon-term-apply h b))))
(define (h t)
  (TNTreturn (- (* 3 t) 1)))
```

2. Le résultat brut de la transformation en forme CPS est :

$$
\begin{array}{l}
(\lambda \mathrm{t}_1. \\
\quad \mathrm{op\_car}\ \mathrm{t}_1 \\
\quad\quad (\lambda \mathrm{t}_2. \\
\quad\quad\quad \text{if } \mathrm{t}_2 \text{ then} \\
\quad\quad\quad\quad (\lambda \mathrm{t}_3. \\
\quad\quad\quad\quad\quad (\lambda \mathrm{t}_4. \\
\quad\quad\quad\quad\quad\quad (\lambda \mathrm{t}_5. \\
\quad\quad\quad\quad\quad\quad\quad \mathrm{t}_4\ \mathrm{t}_5 \\
\quad\quad\quad\quad\quad\quad\quad\quad (\lambda \mathrm{t}_6. \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \mathrm{t}_3\ \mathrm{t}_6\ \mathcal{K}_0)) \\
\quad\quad\quad\quad\quad\quad \mathrm{n}) \\
\quad\quad\quad\quad\quad \mathrm{h}) \\
\quad\quad\quad\quad \mathrm{g} \\
\quad\quad\quad \text{else} \\
\quad\quad\quad\quad \mathcal{K}_0 \\
\quad\quad\quad\quad (\lambda \mathrm{x}\ \mathrm{k}_1. \\
\quad\quad\quad\quad\quad \mathrm{k}_1\ \mathrm{n}))) \\
\mathrm{n}
\end{array}
$$

Si on élimine les λ-expressions purement administratives, on obtient :

$$
\begin{array}{l}
\mathrm{op\_car}\ \mathrm{n} \\
\quad (\lambda \mathrm{t}_2. \\
\quad\quad \text{if } \mathrm{t}_2 \text{ then} \\
\quad\quad\quad \mathrm{h}\ \mathrm{n} \\
\quad\quad\quad\quad (\lambda \mathrm{t}_6. \\
\quad\quad\quad\quad\quad \mathrm{g}\ \mathrm{t}_6\ \mathcal{K}_0) \\
\quad\quad \text{else} \\
\quad\quad\quad \mathcal{K}_0 \\
\quad\quad\quad (\lambda \mathrm{x}\ \mathrm{k}_1. \\
\quad\quad\quad\quad \mathrm{k}_1\ \mathrm{n}))
\end{array}
$$

3. La forme générale de la conversion en forme CPS de l'expression est :

$$
\begin{array}{rl}
 & (\lambda \mathrm{t}_1.\ \text{if } \mathrm{t}_1 \text{ then } (\lambda \mathrm{u}_1.\ C_2)\ 1 \text{ else } (\lambda \mathrm{v}_1.\ C_2)\ 2)\ \mathrm{y}1 \\
\text{où} & C_2 = (\lambda \mathrm{t}_2.\ \text{if } \mathrm{t}_2 \text{ then } (\lambda \mathrm{u}_2.\ C_3)\ 1 \text{ else } (\lambda \mathrm{v}_2.\ C_3)\ 2)\ \mathrm{y}2 \\
\text{où} & C_3 = \ldots \\
\text{où} & C_i = (\lambda \mathrm{t}_i.\ \text{if } \mathrm{t}_i \text{ then } (\lambda \mathrm{u}_i.\ C_{i+1})\ 1 \text{ else } (\lambda \mathrm{v}_i.\ C_{i+1})\ 2)\ \mathrm{y}i \\
\text{où} & C_{i+1} = \ldots \\
\text{où} & C_n = (\lambda \mathrm{t}_n.\ \text{if } \mathrm{t}_n \text{ then } (\lambda \mathrm{u}_n.\ C_{n+1})\ 1 \text{ else } (\lambda \mathrm{v}_n.\ C_{n+1})\ 2)\ \mathrm{y}n \\
\text{où} & C_{n+1} = \mathcal{K}_0\ 3
\end{array}
$$

   Clairement, si on n'utilisait pas artificiellement les abréviations C_i, on aurait un code de taille dans $O(2^n)$.

4. Nous devons commencer par montrer que si K ne comporte aucun appel non-terminal, alors CPS$[\![e]\!][\![K]\!]$ ne comporte aucun appel non-terminal peu importe le λ-terme e. Il faut procéder par induction sur la taille de e. La preuve est très simple à obtenir. Il suffit de considérer chacune des 10 formes de syntaxe possibles pour e. Il peut être nécessaire d'utiliser l'hypothèse d'induction jusqu'à 3 fois pour certaines formes et il faut faire une sous-preuve par induction dans le cas de l'appel (n arguments).

   Ensuite, il est trivial de conclure que CPS_main$[\![e]\!]$ ne comporte aucun appel non-terminal, peu importe e.