Facultad de Ciencias, UNAM
Lenguajes de Programación
Ejecuciones

Rubí Rojas Tania Michelle

21 de enero de 2021

Realiza la ejecución paso a paso de:

1. `(make-list 5)`

   Usando la implementación de CPS

```
(define (make-list n)
  (make-list/k n (lambda (x) x)))

(define (make-list/k n k)
  (if (zero? n)
      (k '())
      (make-list/k (sub1 n) (lambda (v) (k (cons n v))))))
```

   Solución:

```
(make-list 5) = (make-list/k 5 (λ(x) x))
              = (make-list/k 4 (λ(v) ((λ(x) x) (cons 5 v))))
              = (make-list/k 3 (λ(v) ((λ(v) ((λ(x) x) (cons 5 v))) (cons 4 v))))
              = (make-list/k 2 (λ(v) ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v))) (cons 4 v)))
                                     (cons 3 v))))
              = (make-list/k 1 (λ(v) ((λ(v) ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v)))
                                     (cons 4 v))) (cons 3 v))) (cons 2 v))))
              = (make-list/k 0 (λ(v) ((λ(v) ((λ(v) ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v)))
                                     (cons 4 v))) (cons 3 v))) (cons 2 v))) (cons 1 v))))
              = ((λ(v) ((λ(v) ((λ(v) ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v)))
                (cons 4 v))) (cons 3 v))) (cons 2 v))) (cons 1 v))) '())
              = ((λ(v) ((λ(v) ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v))) (cons 4 v)))
                (cons 3 v))) (cons 2 v))) '(1))
              = ((λ(v) ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v))) (cons 4 v))) (cons 3 v)))
                '(2 1))
              = ((λ(v) ((λ(v) ((λ(x) x) (cons 5 v))) (cons 4 v))) '(3 2 1))
              = ((λ(v) ((λ(x) x) (cons 5 v)))  '(4 3 2 1))
              = ((λ(x) x) '(5 4 3 2 1))
              = '(5 4 3 2 1)
```

2. `(map add1 '(1 2 3))`

   Usando la implementación de CPS

```
(define (map f lst)
  (map2 f lst (lambda (x) x)))

(define (map2 f lst k)
  (if (null? lst)
      (k '())
      (f (car lst))
         (lambda (v) (map2 f (cdr lst) (lambda (v2) (k (cons v v2)))))))
```

   Solución: Redefinimos la función `map2` de la siguiente manera

```
(define (map2 f lst k)
  (if (null? lst)
      (k '())
      (map2 f (cdr lst) (lambda (v) (k (cons (f (car lst)) v))))))
```

   pues al ejecutar la función original de `map2` obtenemos un error de compilación. De cualquier forma, esta nueva versión utiliza CPS.

$$
\begin{aligned}
\texttt{(map add1 '(1 2 3))} &= \texttt{(map2 add1 '(1 2 3) (}\lambda\texttt{(x) x))} \\
&= \texttt{(map2 add1 '(2 3) (}\lambda\texttt{(v) ((}\lambda\texttt{(x) x) (cons (add1 1) v))))} \\
&= \texttt{(map2 add1 '(3) (}\lambda\texttt{(v) ((}\lambda\texttt{(v) ((}\lambda\texttt{(x) x) (cons (add1 1) v)))} \\
&\quad \texttt{(cons (add1 2) v))))} \\
&= \texttt{(map2 add1 '() (}\lambda\texttt{(v) ((}\lambda\texttt{(v) ((}\lambda\texttt{(v) ((}\lambda\texttt{(x) x) (cons (add1 1) v)))} \\
&\quad \texttt{(cons (add1 2) v))) (cons (add1 3) v))))} \\
&= \texttt{((}\lambda\texttt{(v) ((}\lambda\texttt{(v) ((}\lambda\texttt{(v) ((}\lambda\texttt{(x) x) (cons (add1 1) v)))} \\
&\quad \texttt{(cons (add1 2) v))) (cons (add1 3) v))) '())} \\
&= \texttt{((}\lambda\texttt{(v) ((}\lambda\texttt{(v) ((}\lambda\texttt{(x) x) (cons (add1 1) v)))} \\
&\quad \texttt{(cons (add1 2) v))) '(4))} \\
&= \texttt{((}\lambda\texttt{(v) ((}\lambda\texttt{(x) x) (cons (add1 1) v))) '(3 4))} \\
&= \texttt{((}\lambda\texttt{(x) x) '(2 3 4))} \\
&= \texttt{'(2 3 4)}
\end{aligned}
$$