

Lenguajes de Programación 2020-1

Facultad de Ciencias UNAM

Ejercicio Semanal 12

Sandra del Mar Soto Corderi
Edgar Quiroz Castañeda

November 13, 2019

1. Evalúe el resultado de la siguiente expresión

$(\text{letcc } k \text{ in } 10 * (\text{continue } k \ 3)) + 2$

En caso donde las expresiones antes de evaluar ya son valores y no forman parte de una expresión `continue`, se saltará directamente a la siguiente evaluación.

```

□ > (letcc[Nat](k.10 * continue(k, 3))) + 2
→κ sum(_ + 2) > letcc[Nat](k.10 * continue(k, 3))
→κ sum(_ + 2) > (10 * continue(k, 3))[k := cont(sum(_ + 2))]
→β sum(_ + 2) > 10 * continue(cont(sum(_ + 2)), 3)
→κ* mul(10, _), sum(_ + 2) > continue(cont(sum(_ + 2)), 3)
→κ continue(_, 3), mul(10, _), sum(_ + 2) > cont(sum(_ + 2))
→κ continue(_, 3), mul(10, _), sum(_ + 2) < cont(sum(_ + 2))
→κ continue(cont(sum(_ + 2)), _), mul(10, _), sum(_ + 2) > 3
→κ continue(cont(sum(_ + 2)), _), mul(10, _), sum(_ + 2) < 3
→κ sum(_ + 2) < 3
→κ* □ < 5

```

2. Transforme la siguiente función a CPS

```

-- función filter
filter _ [] = []
filter f (x:xs)
  | f x == x ==> True ==> x:xs'
  | otherwise ==> xs'
  where xs' = filter f xs

```

Hay que agregar un acumulador de aplicaciones de funciones así como alterar las llamadas recursivas para que siempre sean lo último que sea llamado.

```

-- función filter en CPS
cpsfilter _ [] k = k []
cpsfilter f (x:xs) k
  | f x ==> True ==> cpsfilter f xs (\v -> k (x:v))
  | otherwise ==> cpsfilter f xs k

```

y utilice su definición para calcular `cpsfilter cpseven [0..10]` paso a paso.