Tarea 3
CC4101 - Lenguajes de Programación

Éric Tanter
Auxiliar: Richard Ibarra

Fecha de Entrega. Viernes 14 de Mayo *

Listas y Currificación

1. **(1 pto)** En esta parte, deberá extender el lenguaje FAE de manera que soporte listas como valores de primera clase.

   El siguiente programa debe ser válido en su lenguaje.

   ```
   {with {l {list 1 2 3}}
      {+ {car l} {car {cdr l}}}}
   => 3
   ```

2. **(1 pto)** Las funciones del lenguaje FAE admiten sólo un parámetro.

   (a) **(0.5 pto)** Modifique su lenguaje de manera que ahora las funciones reciban una cantidad arbitraria de parámetros.

   ```
   {with {add {fun {x y}} {+ x y}}
      {add 3 4}}
   => 7
   ```

   (b) **(0.2 pto)** Agregue además a su lenguaje la primitiva `fold`, para poder iterar sobre una lista:

   ```
   {fold {fun {x y} {+ x y}} 0 {list 1 2 3 4}}
   => 10
   ```

   (c) **(0.3 pto)** Su lenguaje permite definir identificadores mediante el uso de `with`.

   - ¿De qué manera el hecho de tener funciones de múltiples argumentos nos permite modifcar `with` de manera que permita definir más de un identificador?
   - Modifique su lenguaje para que ahora `with` permita definir múltiples identificadores. Medite dónde debe hacer esta modificación.

*Ver las reglas de entrega y evaluación en `http://pleiad.cl/teaching/cc4101/reglas`.

3. **(3 ptos)** En clases ha visto que el uso de currificación permite definir abstracciones de mayor nivel reutilizando el código al entregarle información parcial a la aplicación de una función.

   (a) **(1 pto)** Modifique su intérprete de manera que sea posible entregarle sólo algunos de los parámetros que espera. Note que la aplicación no necesariamente es parámetro a parámetro.

   ```
   {with {add {fun {x y z} {+ {+ x y} z}}
      {with {add7 {add 3 4}}
          {add7 3}}}
   => 10
   ```

   (b) **(1 pto)** Es posible optimizar su intérprete de manera que las funciones sean preevaluadas cuando se aplica currificación[1]. La técnica de preevaluación reduce todas las expresiones que son posibles.

   ```
   add7 <=> {fun {z} {+ 7 z}}
   ```

   (c) **(0.5 pto)** Ilustre el funcionamiento de su currificación con evaluación parcial con el siguiente ejemplo: una función `in-range` que, dado una lista, una cota inferior y una cota superior, retorna `true` si todos los elementos de la lista estan en el rango de las cotas, `false` sino:

   ```
   {in-range {list 2 8 4 6} 1 9}
   => true
   {in-range {list 2 8 4 6} 3 9}
   => false
   ```

   (d) **(0.5 pto)** Su implementación de currificación requiere que los parámetros de la funciones se apliquen en el orden que fueron definidas. Esto impide que la técnica sea usada cuando no se conoce el primero parámetro pero sí alguno de los siguientes.

   Modifique su intérprete para que ahora permita la aplicación de funciones usando el wildcard `?`. Este wildcard indica que el parámetro no se conoce, sin embargo, alguno de los siguentes será entregado.

   ```
   {with {sub {fun {x y}} {- x y}}
      {with {sub5 {sub ? 5}}}
         {sub5 5}}}
   => 0

   {with {in-range {fun {lst inf sup} ...}}
     {with {in-range-1-10 {in-range ? 1 10}}
       {in-range-1-10 {list 2 4}}}}
   => true
   ```

[1] Esa técnica aplicable en varios contextos se conoce como evaluación parcial: http://en.wikipedia.org/wiki/Partial_evaluation