Universida Nacional Autonoma de Mexico
Lenguajes de Programación 2016-1
Facultad Ciencias
Tarea 2

Rodrigo Rivera Paz

Facultad de Ciencias UNAM

Problema I

En teoria y laboratorio hemos visto el lenguaje FAE, que es un lenguaje con expresiones aritmeticas, funciones y aplicaciones de funciones. ¿Es FAE un lenguaje Turing-Completo?. Debes proveer una respuesta breve e inambigua, seguida de una justificacion mas extensa de tu respuesta. Hint: Investiguen sobre el combinador Y.

Respuesta: FAE es un Turing completo ya que posee estructuras de control y debido al combinador Y obtiene la simulación de recursión

Problema II

¿Java es glotón o perezoso? Escribe un programa para determinar la respuesta a esta pregunta. El mismo programa, ejecutado en cada uno de los dos regímenes, debe producir resultados distintos. Puedes usar todas las características de Java que gustes, pero debes mantener el programa relativamente corto: **penalizaremos cualquier programa que consideremos excesivamente largo o confuso** (Hint: es posible resolver este problema con un programa de unas cuantas docenas de lineas).

Respuesta: Java es glotón debido a que ejecuta todas las operaciones posibles como se ve en el programa ejecuta la división entre cero aunque aún no es realmente necesaria. Ese código en evaluación perezosa daría True, debido a que no pide nada en la función foo para regresar true y en evaluación glotona da error.

Problema III

En nuestro intérprete perezoso, identificamos 3 puntos en el lenguaje donde necesitamos forzar la evaluación de las expresiones closures (invocando a la función `strict`): la posición de la función de una aplicación, la expresión de prueba de una condicional, y las primitivas aritméticas. Doug Oord, un estudiante algo sedentario, sugiere que podemos reducir la cantidad de código reemplazando todas las invocaciones de `strict` por una sola. En el interprete visto en el capitulo 8 del libro de Shriram elimino todas las instancias de `strict` y reemplazo

```
[id (v) (lookup v env)]
```

por

```
[id (v) (strict (lookup v env))]
```

El razonamiento de Doug es que el único momento en que el interprete regresa una expresión closure es cuando busca un identificador en el ambiente. Si forzamos esta evaluación, podemos estar seguros de que en ninguna otra parte del interprete tendremos un closure de expresiones, y eliminando las otras invocaciones de `strict` no causaremos ningún daño. Doug evita razonar en la otra dirección, es decir si esto resultara o no en un interprete mas glotón de lo necesario.

Escribe un programa que produzca diferentes resultados sobre el interprete original y el de Doug. Escribe el resultado bajo cada interprete e identifica claramente cual interprete producirá cada resultado. Asume un lenguaje interpretado con características aritméticas, funciones de primera clase, with, if0 y rec (aunque algunas no se encuentren en nuestro interprete perezoso). Hint: Compara este comportamiento contra el interprete perezoso que vimos en clase y no contra el comportamiento de Haskell.

Si no puedes encontrar un programa como el que se pide, defiende tus razones de por que no puede existir, luego considera el mismo lenguaje con cons, first y rest añadidos.

Respuesta:

Problema IV

Ningún lenguaje perezoso en la historia ha tenido operaciones de estado (tales como la mutación de valores en cajas o asignación de valores a variables) ¿Por que no?

La mejor respuesta a esta pregunta incluiría dos cosas: un pequeño programa (que asume la evaluación perezosa) el cual usara estado y una breve explicación de cual es el problema que ilustra la ejecución de dicho programa. Por favor usa la noción original (sin cache) de perezosos sin cambio alguno. Si presentas un ejemplo lo suficientemente ilustrativo (el cual no necesita ser muy largo), tu explicación sera muy pequeña.