Lenguajes de programación 2016-1
Tarea 1

José Ricardo Rodríguez Abreu
Profesora: Karla Ramírez Pulido
Ayudante: Héctor Enrique Gómez Morales

October 11, 2015
Facultad de Ciencias UNAM

1. Problema I

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

La respuesta es SI:

La Tesis Church-Turing nos da la equivalencia entre los conceptos de una función computable con las máquinas de Turing, por lo que sólo hay que ver que nuestro lenguaje es capaz de procesar todas las funciones recursivas. Debemos probar que nuestro lenguaje tiene funciones de recursión primivita:

0. if y ciclos:

Tenemos el if0S y podemos emular el for con una función recursiva usando fun y condición de if0S hasta que la variable i sea igual a cero.

1. Función cero:

Definimos la función zero como: '{{fun {n} {0}} X} Y podemos ver claramente que siempre regresa numV 0.

2. Función sucesor:

Definimos la función sucesor como:
'{{fun {n} $+n1$} X} Y es la función que nos da numV x+1.

3. Proyección y composición:

Por definición de FAES podemos ver que podemos hacer funciones de composición como funciones anidadas n veces. También podemos ver con el With*s que podemos hacer proyección con distintos argu-

mentos.

El combinador Y nos regala la definición de recursión en nuestro lenguaje:

Nos dice que Y g = g (Y g).

2. Problema II

Â¿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). Debes anexar tanto el código fuente de tu programa (en un archivo aparte al PDF de la tarea) así como una respuesta a la pregunta de si Java es glotón o perezoso, y una explicación de porque su programa determina esto. Es decir, deben proveer una respuesta breve e inambigua (p.ej. Java es perezoso) seguida de una descripción del resultado que obtendrías bajo cada régimen, junto con una breve explicación de por que ese régimen generaría tal resultado.

Java es glotón pero tiene algo llamado evaluación de corto circuito que podemos ver en el programa anexado ya que probamos las dos cosas: La evaluación glotona la podemos ver durante la llamada a la función de fTrue ya que nunca la usamos pero se evalua cada una de las partes de la función. El corto circuito lo vemos en el if ya que si evaluara todo entraría a el factorial que claramente fallará al acabarnos la memoria.

En el programa anexado se demuestra lo siguiente:
1. El programa tiene definido una función factorial y es llamada 2 veces con un número muy grande para demostrar que nunca es llamada. Esto ocurre porque Java hace uso de un tipo de evaluación perezosa llamada "de corto circuito" que sólo evalúa los argumentos que necesitan algunos operadores booleanos. Por ejemplo el operador lógico && sólo evalua hasta encontrar uno que sea falso ya que hace que el AND sea falso. Equivalente con el OR de java: Sólo evalua hasta encontrar uno que sea true.
2. Java es glotón porque cada que el usuario manda a llamar a una método lo evalua y lo vemos en el caso de fTrue y fFalse: Aunque sólo son llamadas sin uso, java las evalúa.

3.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...

Si creamos la función identidad:
{{fun {x} x} x}

De acuerdo a Doug, en nuestro programa deberia decir que no existe la función x, sin embargo en el

programa original nos regresa el id X y aqui difiere lo que dice Doug del programa del Shriram.

4. Problema IV

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

Supongamos que si lo hace: Entonces le asignamos valores a las variables y llamamos a una función con esas variables, pero no necesitamos evaluar la función hasta después de un cambio de valores en las variables. Entonces hacemos la llamada a la función pero al ser llamada como le asignamos valores a las variables y las cambiamos, nos regresará algo que no esperábamos.