Lenguajes de programación 2020-1
Tarea Examen Parcial 4

Favio E. Miranda Perea Javier Enríquez Mendoza

21 de Noviembre de 2019
Fecha de entrega: 2 de diciembre de 2019
Facultad de Ciencias UNAM

Esta tarea vale 8 puntos sobre el parcial 4, la calicación se completa con una pregunta presencial el 29 de noviembre de 2019.

1. (1pt.) Se desea implementar una función `ct` que reciba un árbol heterogéneo de naturales o booleanos y devuelva la conjunción de sus elementos siempre y cuando todos sean booleanos y en otro caso devuelva el valor $n+1$ donde n es el primer natural encontrado en el árbol. Para este propósito defina una función `ctaux` y una expresión e tal que la función `ct` quede implementada como

   ```
   ct t = handle ctaux t with x => e
   ```

   Bosqueje la evaluación de la expresión

   ```
   Node (iszero 9) (Node False Void Void) (Node 5 Void Void)
   ```

   en la máquina $\mathcal{K}$. Puede omitir varios pasos pero no los que involucran el manejo de excepciones.

   Sugerencia: Es más facil si define `ctaux` a partir de una función binaria `vand` que realice la conjunción si sus argumentos son booleanos y en caso contrario lance una excepción adecuada de forma que sea manejada por e. Puede suponer que existe una función unaria `isbool` que verifica si su argumento es o no un booleano.

2. (1pt.) Considere el siguiente programa, donde suponemos que el lenguaje contiene un operador primitivo `not` para la negación booleana.

   ```
   not letcc k2 in
        iszero(2 + letcc k1 in
                    3 + if x = pred 8 then
                           4 * pred (continue k2 6)
                          else 5 * suc (continue k1 false)
                  end
              )
            end
       end
   ```

   a) ¿Cuáles son los tipos de `k1` y `k2`?

   b) ¿A qué continuaciones se ligan las variables `k1,k2`?

   c) ¿A qué se evalua el programa para $x = 7$ y para $x \neq 7$?

3. (1.5pts.) Considere la siguiente función `N` que depende de ciertas funciones dadas `f,g,h`

```
N 0 = 17

N 1 = f (1+13)

N 2 = 22 + (2-3) + 2

N 3 = 22 + (f 3) + 37

N 4 = g 22 (f 4)

N 5 = 22 + (f 4) + 33 + (g y)

N x = h (f x) (44-y) (g y)   cuando x>5
```

Defina la versión cps de `N`, denotada `cpsN`. Para esto puede suponer definidas las versiones cps de `f,g,h` denotadas `cpsf, cpsg, cpsh`. *Atención:* Las operaciones aritméticas deben permanecer sin cambios, es decir, no se piden las versiones cps de + y -.

4. (1.5pts) Considerense las siguientes cuatro características para un lenguaje de programación:

   a) Existe un tipo de cadenas String y se cumple Int $\leq$ String mediante una conversión implícita que transforma a un entero en una cadena digamos, por ejemplo 123 a "123".

   b) Existe un tipo de cadenas String y se cumple String $\leq$ Int mediante una conversión implícita que transforma a una cadena en un entero ignorando los caractéres que no son dígitos, excepto el caracter '-' inicial, por ejemplo "-0aw23r4/" corresponde a -234.

   c) Existe un operador binario + que denota a ambas la suma de enteros y la concatenación de cadenas.

   d) Existe un operador binario = que denota a ambas la igualdad de enteros y de cadenas.

   Para cada par de estas características $(a, b), (a, c) \ldots (c, d)$, discuta si se violan o no los principios fundamentales del subtipado. En caso afirmativo escriba un ejemplo de un programa simple que cause un comportamiento ambiguo o contraintuitivo y que muestre porque se violan los principios de subtipado.

5. (1pt.) Usando la definición de números naturales en Java Peso Pluma vista en clase.

   a) Agregue métodos `pot` y `leq` para las operaciónes potencia y el orden $\leq$.

   b) Modelar la clase `Boolean` para el manejo de valores booleanos. La clase debe de extender de `Object` y el constructor recibirá un objeto de la clase `Nat` para definir su valor, si se recibe un cero, el objeto `Boolean` representará al valor `false` y si se recibe uno, el objeto representará `true`. Además se deben construir los métodos `true` y `false` que regresan una instancia de `Boolean` segun el caso.

6. (2pts.) El objetivo de este ejercicio es definir el siguiente mini lenguaje de expresiones aritméticas y booleanas MinEAB

$$\mathtt{e} ::= n \mid true \mid false \mid e + e \mid e < e$$

en Java Peso Pluma. Atendiendo los siguientes lineamientos:

   a) Defina una clase `Expr` que incluya los siguientes métodos:

   - `isAtom` que devuelve `true` si la expresión no tiene subexpresiones propias.
   - `lsub` que devuelve la subexpresión izquierda de una expresión no atómica.
   - `rsub` que devuelve la subexpresión derecha de una expresión no atómica.
   - `eval` que devuelve el valor de la expresión.

   Esta clase debe ser abstracta en el sentido de que no tiene atributos y por lo tanto sus métodos no hacen nada pero deben ser definidos. Es decir, se considera a `Expr` como una interfaz.

b) Defina una clase `NumExpr` que extienda a `Expr` y de forma que sus instancias correspondan a números.

c) Defina una clase `BoolExpr` que extienda a `Expr` y de forma que sus instancias correspondan a booleanos.

d) Defina una clase `SumExpr` que extienda a `Expr` y de forma que sus instancias correspondan a sumas.

e) Defina una clase `LTExpr` que extienda a `Expr` y de forma que sus instancias correspondan a comparaciones de orden.

f) Dé ejemplos de instancias de cada una de las clases.

g) Se extiende al lenguaje MinEAB con expresiones de las formas $-e$, *iszero* e. Modifique la definición de la clase `Expr` para poder modelar a las nuevas expresiones. Defina después clases `NegExpr` y `IsZExpr` cuyas instancias sean expresiones de las nuevas formas $-e$ e *iszero* e respectivamente, dando ejemplos de instancias de cada clase. ¿Cómo se modifican las subclases de `Expr` definidas en los puntos anteriores?

Puede suponer definida una clase `Value` cuyas instancias sean los valores del lenguaje, ya sea naturales o booleanos. Es decir, Value es esencialmente el tipo Nat + Bool. Ademas de otras clases primitivas con los metodos que requiera, especificandolos previamente. También puede utilizar convenientemente la constante de error en cualquier método.

7. **(hasta 2 pts extra): Privacidad en Java Peso Pluma**
Java proporciona mecanismos para controlar el acceso. Un método o atributo de una clase pueden declararse como *publico, protegido o privado*. En Java Peso pluma podemos hacer lo mismo como sigue:

$$\texttt{C} \;::=\; \texttt{class C extends C}\,\{\vec{p}\,\vec{\texttt{C}}\,\vec{f}\,;\,\texttt{K}\,\vec{\texttt{M}}\} \quad -\textit{declaracion de clases}$$

$$\texttt{M} \;::=\; p\,\texttt{C m}\,(\vec{\texttt{C}}\,\vec{x}\,)\,\{\texttt{return}\,e;\,\} \quad -\textit{declaracion de metodos}$$

$$p \;::=\; \texttt{public} \mid \texttt{protected} \mid \texttt{private} \quad -\textit{modificador de privacidad}$$

El significado intuitivo de los modificadores de privacidad es el siguiente:

- Si un método o atributo de una clase `C` se declara `public`, entonces se permite el acceso desde cualquier lugar.
- Si un método o atributo de una clase `C` se declara `protected`, entonces se permite el acceso únicamente desde métodos de `C` y subclases de `C`.
- Si un método o atributo de una clase `C` se declara `private`, entonces se permite el acceso únicamente desde métodos de `C`.

Extienda la semántica estática para filtrar programas que contengan violaciones de privacidad de acuerdo a las reglas dadas arriba. Especifique claramente cuales reglas de tipado originales se eliminan o se sustituyen por nuevas y cuales se mantienen.
Sugerencia: Cuando se verifica si un método está bien formado en una clase `C` se debe verificar si la expresión en el cuerpo del método no se refiere a métodos o atributos en otra clase `D` que seán privados o protegidos si $\texttt{C} \not\leq \texttt{D}$. Podría necesitarse el paso de algo más que el contexto Γ en el juicio de tipado para expresiones.