Lenguajes de Programación 2016-1
Tarea 3

Ricardo Garcia Garcia   Juan Carlos López López   Luis Rodrigo Rojo Morales

25 de noviembre de 2015
Facultad de Ciencias UNAM

Problema I

Haga el juicio de tipo para la función fibonacci y el predicado empty?

```
(define fibonacci
  (lambda (n)
    (cond
      [(<= n 2) 1]
      [else (+ (fibonacci (- n 1)) (fibonacci (- n 2)))])))
```

$$\frac{\surd}{\Gamma \vdash n : number} \qquad \frac{\surd}{\Gamma \vdash 1 : number} \qquad \frac{\surd}{\Gamma \vdash n : number} \qquad \frac{\surd}{\Gamma \vdash 2 : number}$$

$$\frac{\Gamma \vdash n : number \quad \Gamma \vdash 1 : number}{\Gamma \vdash (- \quad n \quad 1) : number} \qquad \frac{\Gamma \vdash n : number \quad \Gamma \vdash 2 : number}{\Gamma \vdash (- \quad n \quad 2) : number}$$

$$\frac{\Gamma \vdash (- \quad n \quad 1) : number}{\Gamma(fibonacci(number \to number) \vdash \Gamma \vdash (- \quad n \quad 1) : number} \qquad \frac{\Gamma \vdash (- \quad n \quad 2) : number}{\Gamma(fibonacci(number \to number) \vdash \Gamma \vdash (- \quad n \quad 2) : number}$$

$$\frac{\Gamma(fibonacci(number \to number) \vdash \Gamma \vdash (- \quad n \quad 1) : number}{\Gamma \vdash (fibonacci(- \quad n \quad 1)) : number} \qquad \frac{\Gamma(fibonacci(number \to number) \vdash \Gamma \vdash (- \quad n \quad 2) : number}{\Gamma \vdash (fibonacci(- \quad n \quad 2)) : number}$$

$$\frac{\surd}{\Gamma \vdash n : number} \qquad \frac{\surd}{\Gamma \vdash 2 : number}$$

$$\frac{\Gamma \vdash n : number \quad \Gamma \vdash 2 : number}{\Gamma \vdash (<= \quad n \quad 2) : bool} \qquad \frac{\surd}{\Gamma \vdash 1 : number} \qquad \frac{\Gamma \vdash (fibonacci(- \quad n \quad 1)) : number \quad \Gamma \vdash (fibonacci(- \quad n \quad 2)) : number}{\Gamma \vdash (+(fibonacci(- \quad n \quad 1)) \quad (fibonacci(- \quad n \quad 2))) : number}$$

$$\frac{\Gamma \vdash (<= \quad n \quad 2) : bool \quad \Gamma \vdash 1 : number \quad \Gamma \vdash (+(fibonacci(- \quad n \quad 1)) \quad (fibonacci(- \quad n \quad 2))) : number}{\Gamma[n \leftarrow number] \vdash (cond \quad [(<= \quad n \quad 2) \quad 1] \quad [else \quad (+(fibonacci(- \quad n \quad 1)) \quad (fibonacci(- \quad n \quad 2)))]) : number}$$

$$\frac{\Gamma[n \leftarrow number] \vdash (cond \quad [(<= \quad n \quad 2) \quad 1] \quad [else \quad (+(fibonacci(- \quad n \quad 1)) \quad (fibonacci(- \quad n \quad 2)))]) : number}{\Gamma[fibonacci \leftarrow number] \vdash \{(lambda(n : number)) : number \quad (cond \quad [(<= \quad n \quad 2) \quad 1] \quad [else \quad (+(fibonacci(- \quad n \quad 1)) \quad (fibonacci(- \quad n \quad 2)))])\} : number \to number}$$

$$\frac{\text{l:list}}{\text{(empty? l):bool}}$$

Problema II

Considera el siguiente programa:

```
(+ 1 (first (cons true empty)))
```

Este programa tiene un error de tipos.

Genera restricciones para este programa. Aísla el conjunto mas pequeño de estas restricciones tal que, resultas juntas, identifiquen el error de tipos.

Siéntete libre de etiquetar las sub-expresiones del programa con superíndices para usarlos cuando escribas y resuelvas tus restricciones.

Respuesta:

$\boxed{1}$(+ $\boxed{2}$1 $\boxed{3}$ (first $\boxed{4}$ (cons $\boxed{5}$ true $\boxed{6}$ empty)))

Conjunto de Restricciones:

Si e1 y e2 expresiones cuales quiera dentro del lenguaje entonces:
1-[(+ e1 e2)]= number si [e1]= number y [e2]= number
2-Si e es un numero entonces [e]= number
3-[(first e)]= number si [e]= nlist
4-[(cons e1 e2)]= nlist si [e1]= number y [e2]= nlist
5- si e=true o e=false entonces [e]=boolean

Inferencia:

Tenemos que:
A)Para [$\boxed{1}$] =[(+ 1 (first (cons true empty)))] = number si [1] = number y [(first (cons true empty))]= number (1)
B)Para [$\boxed{2}$] = [1] number (2)
C)Para [$\boxed{3}$] = [(first (cons true empty))]= number si [(cons true empty)]=nlist (3)
D)Para [$\boxed{4}$] = [(cons true empty)]=nlist si [true]= number y [empty] = nlist (4)
E)Para [$\boxed{5}$] = [true]!=number sino que [true] = boolean (5)
Por E) tenemos que el programa no puede continuar debido al error de tipos dadas las reglas del conjunto de restricciones

Problema III

Considera la siguiente expresión con tipos:

```
{fun {f : C1 } : C2
  {fun {x : C3 } : C4
    {fun {y : C5 } : C6
      {cons x {f {f y}}}}}}
```

Dejamos los tipos sin especificar (Cn) para que sean llenados por el proceso de inferencia de tipos. Deriva restricciones de tipos para el programa anterior. Luego resuelve estas restricciones. A partir de estas soluciones, rellena los valores de las Cn. Asegúrate de mostrar todos los pasos especificados por los algoritmos (i.e., escribir la respuesta basándose en la intuición o el conocimiento es insuficiente). Deberás usar variables de tipo cuando sea necesario. Para no escribir tanto, puedes etiquetar cada expresión con una variable de tipos apropiada, y presentar el resto del algoritmo en términos solamente de estas variables de tipos.

```
{ 1 fun {f}
  { 2 fun {x}
    { 3 fun {y}
      { 4 cons 5 x  6 {f 7 {f y}}}}}}
```

Si e1 y e2 expresiones cuales quiera dentro del lenguaje entonces:
1.- [(cons e1 e2)]= nlist si [e1]= number y [e2]= nlist

Restricciones:

$[\boxed{1}] = [f] \rightarrow [\boxed{1}]$

$[\boxed{2}] = [x] \rightarrow [\boxed{2}]$

$[\boxed{3}] = [y] \rightarrow [\boxed{3}]$

$[\boxed{4}] = [\{cons\ x\ \{f\ \{f\ y\}\}\}] = nlist$ si $[x] = number$ y $[\{f\ \{f\ y\}\}] = nlist$

$[\boxed{5}] = [x] = number$

$[\boxed{6}] = [f\{f\ y\}] = nlist$, $[f] = number$ $[\boxed{7}] = nlist$

$[\boxed{7}] = [\{f\ y\}] = nlist$, $[f] = number$ $[y] = number$

Gracias a estas restricciones podemos sacar los tipos C1...C6

```
{fun {f : number } : nlist
  {fun {x : number } : nlist
    {fun {y : number } : nlist
      {cons x {f {f y}}}}}}
```

Problema IV

Considera los juicios de tipos discutidos en clase para un lenguaje glotón (en el capitulo de **Juicios de Tipos** del libro de Shriram). Considera ahora la versión perezosa del lenguaje. Pon especial atención a las reglas de tipado para:

- definición de funciones
- aplicación de funciones

Para cada una de estas, si crees que la regla original no cambia, explica por que no (Si crees que ninguna de las dos cambia, puedes responder las dos partes juntas). Si crees que algún otro juicio de tipos debe cambiar, menciónalo también.

Respuesta:

No cambia porque el juicio de tipos lo que te va a dar son los tipos sin evaluar la función y los tipos en definición de funciones y aplicación de funciones son los mismos, no importa si se evalua de forma perezosa o glotona.

Problema V

¿Cuáles son las ventajas y desventajas de tener polimorfismo explícito e implícito en los lenguajes de programación?

Respuesta:

Polimorfismo Explicito

Ventajas	**Desventajas**
-Permitir el reciclamiento de código. -Puedes crear nuevos tipos sin modificar las clases ya existentes	-Poca legibilidad de código debido al hecho de poder tener diversas funciones con el mismo nombre.

Polimorfismo Implicito

Ventajas	**Desventajas**
-No repites código	-Puede que tengas que hacer cast del tipo especifico que necesites

Problema VI

Da las ventajas y desventajas de tener lenguajes de dominio especifico (DSL) y de propósito general. También da al menos tres ejemplos de lenguajes DSL, cada ejemplo debe indicar el propósito del DSL y un ejemplo documentando su uso.

Lenguajes de Dominio Especifico (DSL)

Ventajas	**Desventajas**
-Proporciona apropiadas abstracciones y anotaciones.	-Aprenderlo para que solo pueda resolver un problema espepecifico.
-Nos permiten seguridad en nivel de dominio, mientras los metodos del lenguaje esten seguros esto nos permitira seguridad cada vez que los usemos.	-Encontrar, ajustar o mantener un alcance adecuado.
-Es mas sencillo desarrollar programas en un area en especifico para programadores que no sean expertos en ella.	-Gente no experta en el lenguaje no puede modificar o crear codigo facilmente.

Lenguajes de Proposito General (GPL)

Ventajas	**Desventajas**
-Nos ayuda a resolver problemas de diferentes areas.	-Por ese mismo motivo, puede causar que el programador use un lenguaje poco apto para resolver algún problema.
-Actualmente cuentan con una amplia gama de bibliotecas y documentación de respaldo.	-Si quieres resolver un problema en especifico puede que sea mejor usar un dsl.

Ejemplos DSL:

1. SQL: Este lenguaje fue creado para acceder a bases de datos relacionales facilmente.
   Ejemplo:

   ```
   SELECT * FROM TABLA;
   ```

   nos da toda la info alamacenada en TABLA.

2. Fortran (**For**mula **Tran**slating System): es un lenguaje de programación de alto nivel, procedimental e imperativo, que está especialmente adaptado al cálculo numérico y a la computación científica.

   ```
    program circle
   real r, area, pi

   write (*,*) 'Radio r:'
   read (*,*) r
   pi = atan(1.0e0)*4.0e0
   ```

```
 area = pi*r*r
 write (*,*) 'Area = ', area
end
```

Este programa recibe un numero real n e imprime el area de un circulo con radio n

3. Algol (**Algo**rithmic **L**anguage): es un lenguaje de programacion que fue diseñado especialmente para el computo cientifico. Fue usado principalmente por cientificos en computacion en Estados Unidos y Europa. ALGOL 60 se convirtio en el estandar para la publicacion de algoritmos y tuvo un gran impacto en el desarrollo de futuros lenguajes.

```
PROC print fibo = (INT n) VOID :
  BEGIN
      INT a := 0, b := 1;
      FOR i TO n DO
          print((whole(i, 0), " => ", whole(b, 0), new line));
          INT c = a + b;
          a := b;
          b := c
      OD
  END;

INT i;
print("Fibonacci hasta? ");
read((i, new line));
print fibo(i)
```

Ejemplo en Algol que nos da la serie de fibonacci hasta n.

4. CSS: es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML Ejemplo:

```
  p {
    text-align: center;
    color: red;
}
```

Alinea la etiqueta p y le pone color rojo.