Lenguajes de Programación

Tarea 03

Hector Enrique Gómez Morales

Jonathan Abrego Alvarez , 308043305
Ascencio Espindola Jorge Eduardo, 309043511

25 de noviembre de 2015

Problema I

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

Función fibicacci = fib. Nota algunas cosas se acortaron el nombre para reducir espacio y que cupieran las cosas lo mas decentemente posible

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \texttt{fib:(number}\rightarrow\texttt{number)}}\quad \dfrac{}{\Gamma \vdash \texttt{1:number}}}{\Gamma \vdash [\texttt{fib}\leftarrow\texttt{number}] \vdash \texttt{\{fib 1\}:number}}\quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{n:num}}}{\Gamma \vdash \texttt{n:number}}\quad \dfrac{\dfrac{}{\texttt{0:num}}}{\Gamma \vdash \texttt{0:number}}}{\Gamma \vdash \texttt{(= n 0):boolean}}\quad \dfrac{}{\texttt{0:number}}}{\Gamma \vdash \texttt{(= n 0):boolean}}\quad \dfrac{}{\Gamma \vdash \texttt{0:number}}\ \texttt{**)}}{\Gamma \vdash [\texttt{fib}\leftarrow\texttt{number}] \vdash \texttt{(if(= n 0) 0 (if(= n 1) 1 (+(fib(- n 1))(fib(- n 2))))): number}}}{\Gamma \vdash [\texttt{fib}\leftarrow\texttt{number}] \vdash \texttt{\{fun (n:number)number (if(= n 0) 0 (if(= n 1) 1 (+ (fib(- n 1))(fib(- n 2))))):number}\rightarrow\texttt{number\}}}}{\Gamma \vdash \texttt{rec\{fib: number (fun(n:number): number (if (= n 0) 0 (if (= n 1) 1 (+ (fib (- n 1)) (fib(- n 2))))))\}\{fib 1\}}}$$

Continuación de los asteriscos.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\texttt{n:num}}}{\Gamma \vdash \texttt{n:number}}\quad \dfrac{\dfrac{}{\texttt{1:num}}}{\Gamma \vdash \texttt{1:number}}}{\Gamma \vdash \texttt{(= n 1):boolean}}\quad \dfrac{}{\texttt{1:number}}}{\Gamma \vdash \texttt{(= n 1):boolean}}\quad \dfrac{}{\Gamma \vdash \texttt{1:number}}\quad \dfrac{\dfrac{}{\Gamma \vdash \texttt{fib:(num}\rightarrow\texttt{num)}}\quad \dfrac{}{\Gamma \vdash \texttt{(- n 1):num}}}{\Gamma \vdash \texttt{(fib(- n 1))}}\quad \dfrac{\dfrac{}{\Gamma \vdash \texttt{(num}\rightarrow\texttt{num)}}\quad \dfrac{}{\Gamma \vdash \texttt{(- n 2):num}}}{\Gamma \vdash \texttt{(fib(- n 2))}}}{\Gamma \vdash \texttt{(+(fib(- n 1))(fib(- n 2))))): number}}}{\texttt{**)}\ \Gamma \vdash \texttt{(if(= n 1) 1 (+(fib(- n 1))(fib(- n 2))))): number}}$$

Predicado empty?

$$\dfrac{\Gamma \vdash \texttt{lst :list}}{\Gamma \vdash \texttt{\{empty? lst\}: boolean}}$$

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.

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

Por lo que sabemos para realiza una suma se necesita dos expresiones $e1$ & $e2$, donde ambas deben de ser de tipo number:

$[e1] = number$

$[e2] = number$

$\Rightarrow [(+\ e1\ e2)] = number$

Restricciones:

$\boxed{1}$ =[(+ 1 (first (cons true empty)))]= $number$

$\boxed{2}$ = [1] = $number$

$\boxed{3}$ =[(first(cons true emprty))] = $number$

$\boxed{4}$ = [(cons true emprty)]=number

$\boxed{5}$ =[true]$\neq$ $number$ pero como sabemos $[true] = boolean$

Viendo las restricciones anteriores podemos ver que habrá un error de tipos ya que se por lo que se menciono al principio para una suma se necesitan de tipo number ambas expresiones

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}}}}}}
```

	Stack	Sustitución
Comienzo	$[[\boxed{1}]] = [[f]] \to [[\boxed{2}]]$ $[[\boxed{2}]] = [[x]] \to [[\boxed{3}]]$ $[[\boxed{3}]] = [[y]] \to [[\boxed{4}]]$ $[[cons]] = [[\boxed{5}]] \times [[\boxed{6}]] \to [[\boxed{4}]] = number \times list \to list$ $[[\boxed{5}]] = [[x]]$ $[[f]] = [[\boxed{7}]] \to [[\boxed{6}]]$ $[[f]] = [[y]] \to [[\boxed{7}]]$	
Paso 3	$[[\boxed{2}]] = [[x]] \to [[\boxed{3}]]$ $[[\boxed{3}]] = [[y]] \to [[\boxed{4}]]$ $[[cons]] = [[\boxed{5}]] \times [[\boxed{6}]] \to [[\boxed{4}]] = number \times list \to list$ $[[\boxed{5}]] = [[x]]$ $[[f]] = [[\boxed{7}]] \to [[\boxed{6}]]$ $[[f]] = [[y]] \to [[\boxed{7}]]$	$[[\boxed{1}]] \mapsto [[f]] \to [[\boxed{2}]]$
Paso 3	$[[\boxed{3}]] = [[y]] \to [[\boxed{4}]]$ $[[cons]] = [[\boxed{5}]] \times [[\boxed{6}]] \to [[\boxed{4}]] = number \times list \to list$	$[[\boxed{1}]] \mapsto [[f]] \to [[x]] \to [[\boxed{3}]]$ $[[\boxed{2}]] \mapsto [[x]] \to [[\boxed{3}]]$

	$[[\boxed{5}]] = [[x]]$ $[[f]] = [[\boxed{7}]] \to [[\boxed{6}]]$ $[[f]] = [[y]] \to [[\boxed{7}]]$	
Paso 3	$[[cons]] = [[\boxed{5}]] \times [[\boxed{6}]] \to [[\boxed{4}]] = number \times list \to list$ $[[\boxed{5}]] = [[x]]$ $[[f]] = [[\boxed{7}]] \to [[\boxed{6}]]$ $[[f]] = [[y]] \to [[\boxed{7}]]$	$[[\boxed{1}]] \mapsto [[f]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{2}]] \mapsto [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{3}]] \mapsto [[y]] \to [[\boxed{4}]]$
Paso 5	$[[\boxed{5}]] = number$ $[[\boxed{6}]] = list$ $[[\boxed{4}]] = list$ $[[\boxed{5}]] = [[x]]$ $[[f]] = [[\boxed{7}]] \to [[\boxed{6}]]$ $[[f]] = [[y]] \to [[\boxed{7}]]$	$[[\boxed{1}]] \mapsto [[f]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{2}]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{3}]] \to [[y]] \to [[\boxed{4}]]$
Paso 3	$[[\boxed{6}]] = list$ $[[\boxed{4}]] = list$ $number = [[x]]$ $[[f]] = [[\boxed{7}]] \to [[\boxed{6}]]$ $[[f]] = [[y]] \to [[\boxed{7}]]$	$[[\boxed{1}]] \mapsto [[f]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{2}]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{3}]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{5}]] \to number$
Paso 3	$[[\boxed{4}]] = list$ $number = [[x]]$ $[[f]] = [[\boxed{7}]] \to list$ $[[f]] = [[y]] \to [[\boxed{7}]]$	$[[\boxed{1}]] \mapsto [[f]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{2}]] \to [[x]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{3}]] \to [[y]] \to [[\boxed{4}]]$ $[[\boxed{5}]] \to number$ $[[\boxed{6}]] \to list$

:
:
:

De lo anterior, tratando de intuir un poco el resto que por lo que se dijo era mucho y quién sabe cuando acabariamos

```
{fun {f : list} : list
```

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

Restricciones

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.

No cambian en los lenguajes perezosos, ya que el juicio de tipo esta dado por los tipos que están dado en la función. Es decir lo tipos de juicio los vamos a determiar

Problema V

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

Explícito

Ventajas

✓ Reciclamiento de código.

✓ Creación de nuevos tipos sin alterar clases existentes.

Desventajas

✗ Dificulta le legibilidad del código.

Implícito

Ventajas

✓ No se repite código.

Desventajas

✗ En ciertos casos se requiere hacer el cast del tipo que se necesite.

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.

Lenguaje de dominio especifico

Ventajas

✓ No están pensados para describir la totalidad del sistema

✓ Ofrece un mayor nivel de abstracción

✓ Facilita el desarrollo de programas en un área específicamente, ayudando a los programadores que no tengan mucha experiencia.

✓ Proporciona apropiadas abstracciones y anotaciones.

✓ Permite seguridad en nivel de domino.

Desventajas

✗ Gente con poca experiencia tiene dificultades para modificar o crear código.

✗ Encontrar/Ajustar un alcance adecuado.

Lenguaje de propósito general

Ventajas

✓ Sirven para resolver casi cualquier porblema

✓ Ayuda a resolver problemas de diferentes áreas.

Desventajas

✗ No permite desarrollar más allá de para lo que está pensado y diseñado.

Ejemplos de DSL

- SQL: para definir querys.

Ejemplo 1: SELECT * FROM table1;
No proporciona toda la información de la table1

- HTML: para definir interfaces de usuario. Curiosamente este DSL se ha ido "extendiendo" para ir convirtiéndolo cada vez más en un lenguaje generalista. No es de extrañar que sea difícil definir aplicaciones usando enteramente HTML.

Ejemplo 1:
```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```
Es un ejemplo sencillo de un .html en donde se definen las partes suficientes con las que se puede generar un archivo de este tipo

- CSS: para definir interfaces de usuario a nivel de presentación pura.

Ejemplo 1:
```
h1{
    color: orange;
    text-align: center;
}
```
Da un color naranja a todo el texto que este dentro de la etiqueta h1, además de alinear al centro dicho texto