Lenguajes de Programación
Tarea 3

Bravo García Marco Antonio
Herrera Brito Juan José

Problema I

Fibonacci

$$\frac{\Gamma \vdash \text{fib: (number} \leftarrow \text{number)} \quad \Gamma \vdash \text{n:number}}{\Gamma \vdash [\text{fib} \leftarrow \text{number}] \quad \Gamma \vdash \{\text{fib n}\}\text{:number}} \quad [A]$$

$$\Gamma \vdash \{\text{rec } \{\text{fib: number}\ \{\text{fun (n:number):number}\ \{\text{if (<= n 1)}\ 1\ \text{(+ (fib (- n 1)) (fib (- n 2)))}\}\}\}\}\{\text{fib n}\}$$

$$\frac{\dfrac{\dfrac{\Gamma \vdash \text{n:number} \quad \Gamma \vdash \text{1:number}}{\Gamma \vdash \text{(<= n 1):bool}} \quad \dfrac{\Gamma \vdash \text{n:number} \quad \Gamma \vdash \text{1:number}}{\Gamma \vdash \text{(fib (- n 1))}} \quad \dfrac{\Gamma \vdash \text{n:number} \quad \Gamma \vdash \text{2:number}}{\Gamma \vdash \text{(fib (- n 2))}}}{\Gamma \vdash \text{(<= n 1):bool} \quad \Gamma \vdash \text{1:number} \quad \Gamma \vdash \text{(+ (fib (- n 1)) (fib (- n 2)))}}}{\Gamma, [\text{fib} \leftarrow \text{number}] \vdash \text{(if (<= n 1) 1 (+ (fib (- n 1)) (fib (- n 2))))}}$$

$$[A] = \Gamma, [\text{fib} \leftarrow \text{number}] \vdash \text{(fun (n:number):number : number} \rightarrow \text{number } \{\text{if (<= n 1) 1 (+ (fib (- n 1)) (fib (- n 2)))}\})$$

Empty List

$$\frac{\Gamma \vdash \text{empty?:(list} \rightarrow \text{bool)} \quad \Gamma \vdash \text{l:list}}{\Gamma \vdash \text{(empty? l)}}$$

Problema II

❶(+ ❷1 ❸(first ❹(cons ❺ true ❻empty)))

[❶] = (+ 1 (first (cons true empty)))

[❶] = number y [❷] = [❸] = number

[❷] = [1] = number

[❸] = [❹] → [❺]

[❹] = list y [❺] = number [❻] = list

[❺] = [true] = bool

Aquí hay una inconsistencia en los tipos y da un error

Problema III

❶{fun {f : C1 } : C2
  ❷{fun {x : C3 }: C4
    ❸{fun {y : C5 }: C6
      {❹ cons ❺ x {❻f {❼f ❽y}}}}}}

[❶] = [C1] → [❷]
[❷] = [C3] → [❸]
[❸] = [C5] → [❹]
[❹] = [list] y [❺] = number , [❻] = list
[❺] = number
[❻] = [f] → [❼]
[❼] = [❷]
[❽] = list

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

Problema IV

No cambia por que los juicios de tipo sirven para revisar semánticamente que los tipos sean correctos, en la definición de funciones es exactamente lo mismo ya que tendría que checar los tipos y después evaluar, y en la aplicación de funciones es lo mismo ya que el checador de tipos entra en compilación y la app de funciones en ejecución.

Problema V

Ventajas:

Al tener el polimorfismo podemos sobrecargar una función para que sirva con diferentes tipos, eso nos ahorra tener una función que haga lo mismo para cada tipo y es mas facil usarla.

Desventajas:

La sobrecarga de operadores ya que se puede perder un poco el “significado” de una función, en el polimorfismo implícito si el programador no da por echo puede que su programa no haga lo que el esperaba por que podría ocurrir que en lugar de tirar un error el programa siguiera funcionado

Problema VI

Lenguajes de Dominio Especifico (DSL)

Ventajas:

Son especializados en un problema o un domino es en especifico, lo que nos garantiza que esta bien definido y que siempre tendremos una solución óptima al problema que resolvemos con ellos

Desventajas:

Al ser de domino especifico solo sirven para hacer una “cosa” en especial, por lo tanto no pueden ser utilizados para hacer cualquier cosa o tienen limitaciones.

Lenguajes de Propósito General

Ventajas:

Al ser de propósito general se podría modelar cualquier cosa en ellos y dar solución, mientras no salga del paradigma del lenguaje

Desventajas:

Al ser de propósito general se convierten en mas extensos los programas que el de dominio especifico.

MySQL

Lenguaje para bases de datos, consultas, modificaciones, actualizaciones y creación

```
CREATE TABLE contacts (/* Creamos la tabla contacts*/
 ID_Contact mediumint(8) unsigned default null auto_increment,/*Tiene un campo ID de maximo 8 digitos, positivo, vacio al no poner nada, y que se autoincrementa*/
 Name varchar(50) not null,/*Tiene un campo Name de maximo 50 caracteres y no puede estar vacio*/
 Email varchar(50) not null,/**Tiene un campo Email de maximo 50 caracteres y no puede estar vacio*/
 Phone varchar(15) not null,/**Tiene un campo Phone de maximo 15 caracteres y no puede estar vacio*/
 PRIMARY KEY (ID_Contact)/*Agregamos una restriccion donde ID es la llave primaria de la tabla*/
);
```

PHP

Lenguaje para desarrollo web, mayormente incrustado en un HTML

```
<html>
	<head>
		<title>Prueba de PHP</title><!--Titulo que muestra el navegador web-->
	</head>
	<body>
		<?php echo'<p>Hola Mundo</p>';?> <!--Despliega en la pantalla “Hola Mundo”-->
	</body>
```

```
</html>
```

XML

Es un lenguaje de marcado de etiquetas ,en general sirve para representar objetos y sus propiedades y características por jerarquía

```
<comida>
        <nombre>Waffles</nombre>
        <precio>$2.00</precio>
        <descripcion>Waffles baratos</descripcion>
        <calorias>650</calorias>
 </comida>
```

Tenemos una comida, que cuenta con Nombre,Precio,Descripción y Calorías, en cada campo se especifica el valor.