PROBLEMA I)

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

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash n:num\ \ \Gamma \vdash 0:num\ \ \Gamma \vdash 0:num\Gamma}\ n:num,\ 0:num}{\Gamma \vdash (eq?\ n\ 0):bool\ \Gamma \vdash 0:num}}{\Gamma \vdash [(eq?\ n\ 0)0]:num} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash n:num\ \Gamma \vdash 1:num\ \Gamma \vdash 1:num}\ n:num\ 1:num}{\Gamma \vdash (eq?n\ 1):bool\ \Gamma \vdash 1:num}}{\Gamma \vdash [(eq?\ n\ 1)1]:num} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash n:num\ \ \Gamma \vdash 1:num}\ n:num\ 1:num}{\Gamma \vdash (-\ n\ 1):\ num}}{\Gamma \vdash fibon(num<\text{-}num)} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash n:num\ \ \Gamma \vdash 2:num}\ n:num\ 2:num}{\Gamma \vdash (-n\ 2):\ num}}{\Gamma \vdash fibon(num<\text{-}num)}}{\Gamma \vdash (fibon(-\ n\ 1)):num\ \ \Gamma \vdash (fibon(-\ n\ 2)):num}}{\Gamma \vdash [+(fibon\ (-\ n\ 1))(fibon(-\ n\ 2))]:num}$$

$$\dfrac{\Gamma \vdash fibon:(num <\text{-} num)\ \ \Gamma \vdash n :num \qquad . \qquad \Gamma[n:num] \vdash body\text{-}cond:num}{\Gamma\ [fibon <\text{-} num] \vdash \{fibon\ n:num\} \qquad \Gamma[fibon <\text{-} num] \vdash \{fun\ (n:num):num(cond(body\text{-}cond))\}:(num<\text{-}num)}$$

```
Γ ⊢ {rec ( fibon : num
         (fun ( n:num):num
              (cond
                   [(eq? n 0) 0]
body-cond          [(eq? n 1) 1]
                   [(+ (fibon (- n 1))(fibon (- n 2))])))}
```

$$\dfrac{l:lista}{\Gamma \vdash l:lista \quad .}$$

Γ ⊢ (empty? l) : booleano

PROBLEMA II)

Inferencia de tipos.

(① + ②1 (③first (④ cons true empty)))

[| ① |] = [| (+1 (first(cons true empty))) |]
[| ① |] = number

[| ② |] = [| 1|] = number

[| ③ |] = [| 4|] -> τ (En este caso τ es tipo booleano pues lo que nos regresa first es true)

[| ④ |] = ([|true|] x [|empty|]) -> lista
[|true|] = booleano
[|empty|]=lista

Obtenemos un error semántico porque no se puede sumar un número con un booleano, ya que la suma solo está definida para números.

PROBLEMA III)

Deriva restricciones de tipos para el programa anterior. Luego resuelve estas restricciones. A partir de estas soluciones, rellena los valores de las Cn.

Etiquetar subexpresiones

①{fun {f : C1 } : C2
②{fun {x : C3 } : C4
③{fun {y : C5 } : C6
④{cons x {f {f y}}}}}}

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

[|①|] = [| C1|] -> [|②|]

c2=[|②|] = [|C3|] -> [|③|]

c4=[|③|] = [|C5|] -> [|④|]

c6= [|④|] = ([|C3|] x [| {f {f y}} |])-> lista (por def de cons.)
[|④|] = (τ x lista) -> lista
[|C3|]= τ
[| {f {f y}} |] =lista

[| {f {f y}} |] = [| {f y} |] -> [| {f {f y}} |]
[| {f y} |] = [| C5 |]-> [| {f y} |]

[| f |]= [| f y |] -> [|f (f y)|]
pero [| f |] = [| y |] -> [| f y|]

entonces [| f y |] = [| y |]
[| f(f y)|] = [| f y|]

por lo tanto, [|y|] = [|f y|] = [| f(f y)|] = lista

PROBLEMA IV)

Considera los juicios de tipos discutidos en clase para un lenguaje glotón (en el capítulo 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 qué no (Si crees que ninguna de las dos cambia, puedes responder las dos partes juntas). Si crees que debe cambiar algún otro juicio de tipos, menciónalo también.

Las reglas para los juicios de tipo no cambian puesto que su comportamiento es el mismo,lo único que cambia es el árbol que se forma, porque cargas la expresión completa, no solo el valor, por lo tanto, solo se realizan más pasos para llegar a las hojas del arbol, que corresponden a las variables de la expresión.

PROBLEMA V)

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

Ventajas del polimorfismo explícito: Permite definir una interfaz común y un aspecto externo idéntico para una serie de clases.

Desventajas del polimorfismo explícito: Al declarar y explicitar uno por uno los tipos a usar, es una desventaja para el programador

Ventajas del polimorfismo implícito: Al no incluir especificación previa sobre el tipo de datos sobre el que se trabaja , nos da la ventaja de usarlo en todo tipo de dato compatible.

Desventajas del polimorfismo implícito: La ventaja anterior también es una desventaja ya que al no saber que tipo de datos son con los uqe se trabajan

PROBLEMA VI)

Da las ventajas y desventajas de tener lenguajes de dominio específico (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.

Utilizar un lenguaje de dominio específico nos permite dar una solución que se encuentre más cerca del dominio del problema.

- Como un DSL se especializa en resolver cierto tipo de problemas, utilizarlo cuando se presente alguno de estos, te ayudará a resolverlo de manera más eficiente puesto que contiene funciones específicas para este trabajo. Contiene las construcciones que se ajustan a exactamente el espacio del problema, consta de los elementos y las relaciones que representan directamente la lógica del espacio del problema.
- Los lenguajes específicos suelen estar optimizados para realizar ese tipo específico de operaciones.

- Una desventaja con la que nos topamos al utilizar en DSL es tener que aprender un lenguaje para resolver solamente cierto tipo de problemas.
- Construir un DSL es costoso pues es construir un lenguaje de programación nuevo, con su compilador o intérprete, pues se necesita un analisis lexico, sintactico, semántico, etc.
- Cuando se necesita hacer un programa que requiera múltiples funcionalidades, por ejemplo, un sistema de inventario y cobro para una tienda, en la que se tiene que llevar cuentas, precios, fechas, y el sistema de cobro, y que requiere hacer muchas otras cosas que son considerablemente diferentes entre sí.

- SQL. Para definición, manipulación y control de bases de datos relacionales
- CSS para definir interfaces de usuario a nivel de presentación lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML
- JSP crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos.