Lenguajes de Programación
Tarea III

Andrea Itzel González Vargas
Karla Esquivel Guzmán
Carlos Gerardo Acosta Hernández

Facultad de Ciencias UNAM

Problema I

$$\frac{\dfrac{\dfrac{\Gamma\ [x \leftarrow \text{number}], [\text{fib (number} \rightarrow \text{number})] \vdash \text{(- x 1)} : \text{number}\ \surd \quad \Gamma\ [x \leftarrow \text{number}], [\text{fib (number} \rightarrow \text{number})] \vdash \text{(- x 2)} : \text{number}\ \surd}{\Gamma\ [x \leftarrow \text{number}] \vdash x : \text{number}\ \Gamma\ [x \leftarrow \text{number}] \vdash 1 : \text{number}\ \surd \quad \Gamma\ [x \leftarrow \text{number}] \vdash \text{(fib (- x 1))} : \text{number} \quad \Gamma\ [x \leftarrow \text{number}] \vdash \text{(fib (- x 2))} : \text{number}} \quad \dfrac{\Gamma\ [x \leftarrow \text{number}] \vdash x : \text{number}\ \surd \quad \Gamma\ [x \leftarrow \text{number}] \vdash 0 : \text{number}\ \surd}{\Gamma\ [x \leftarrow \text{number}] \vdash \text{(= x 1)} : \text{bool}\ \Gamma\ [x \leftarrow \text{number}] \vdash 1 : \text{number}\ \surd \quad \Gamma\ [x \leftarrow \text{number}] \vdash \text{(+ (fib (- x 1)) (fib (- x 2)))} : \text{number}}}{\Gamma\ [x \leftarrow \text{number}] \vdash \text{(= x 0)} : \text{boolean} \quad \Gamma\ [x \leftarrow \text{number}] \vdash 0 : \text{number}\ \surd \quad \Gamma\ [x \leftarrow \text{number}] \vdash \text{(if (= x 1) 1 (+ (fib (- x 1) fib (- x 2))))} : \text{number}}$$

$$\frac{\Gamma\ [x \leftarrow \text{number}] \vdash \text{(if (= x 0) 0 (if (= x 1) 1 (+ (fib (- x 1) fib (- x 2)))))} : \text{number}}{\Gamma \vdash \text{fib (x : number)} : \text{number (if (= x 0) 0 (if (= x 1) 1 (+ (fib (- x 1) fib (- x 2)))))} : \text{(number} \rightarrow \text{number)} \quad \Gamma \vdash \text{empty?: (list} \rightarrow \text{bool)}\ \Gamma \vdash \text{l} : \text{list}\ \surd}{\Gamma \vdash \text{(empty? l)} : \text{bool}}$$

Problema II

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

Retricciones

$[[\boxed{1}]]$ = number si $[[\boxed{2}]] = [[\boxed{3}]]$ = number
$[[\boxed{2}]]$ = number
$[[\boxed{3}]]$ = number si $[[\boxed{4}]]$ = nlist
$[[\boxed{4}]]$ = nlist si $[[\boxed{5}]]$ = number y $[[\boxed{6}]]$ = nlist
$[[\boxed{5}]]$ = number si $[[\boxed{5}]]$ contiene un numeral, pero $[[\boxed{5}]]$ = boolean!!
Por lo tanto esta mal formado el programa

Problema III

$\boxed{1}$ {fun {f : C1} : C2
$\boxed{2}$ {fun {x : C3} : C4
$\boxed{3}$ {fun {y : C5} : C6
{$\boxed{4}$ cons x $\boxed{5}$ {f $\boxed{6}$ {f y}}}}}}}

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

Paso 5	[[x]] = number [[$\boxed{5}$]] = list [[$\boxed{4}$]] = list [[f]] = [[$\boxed{6}$]] → [[$\boxed{5}$]] [[f]] = [[y]] → [[$\boxed{6}$]]	[[$\boxed{1}$]] ↦ [[f]] → [[x]] → [[y]] → [[$\boxed{4}$]] [[$\boxed{2}$]] ↦ [[x]] → [[y]] → [[$\boxed{4}$]] [[$\boxed{3}$]] ↦ [[y]] → [[$\boxed{4}$]]
Paso 3	[[$\boxed{5}$]] = list [[$\boxed{4}$]] = list [[f]] = [[$\boxed{6}$]] → [[$\boxed{5}$]] [[f]] = [[y]] → [[$\boxed{6}$]]	[[$\boxed{1}$]] ↦ [[f]] → number → [[y]] → [[$\boxed{4}$]] [[$\boxed{2}$]] ↦ number → [[y]] → [[$\boxed{4}$]] [[$\boxed{3}$]] ↦ [[y]] → [[$\boxed{4}$]] [[x]] ↦ number
Paso 3	[[$\boxed{4}$]] = list [[f]] = [[$\boxed{6}$]] → list [[f]] = [[y]] → [[$\boxed{6}$]]	[[$\boxed{1}$]] ↦ [[f]] → number → [[y]] → [[$\boxed{4}$]] [[$\boxed{2}$]] ↦ number → [[y]] → [[$\boxed{4}$]] [[$\boxed{3}$]] ↦ [[y]] → [[$\boxed{4}$]] [[x]] ↦ number [[$\boxed{5}$]] ↦ list
Paso 4	[[f]] = [[$\boxed{6}$]] → list [[f]] = [[y]] → [[$\boxed{6}$]]	[[$\boxed{1}$]] ↦ [[f]] → number → [[y]] → list [[$\boxed{2}$]] ↦ number → [[y]] → list [[$\boxed{3}$]] ↦ [[y]] → list [[x]] ↦ number [[$\boxed{5}$]] ↦ list [[$\boxed{4}$]] ↦ list
Paso 3	[[$\boxed{6}$]] → list = [[y]] → [[$\boxed{6}$]]	[[$\boxed{1}$]] ↦ [[$\boxed{6}$]] → list → number → [[y]] → list [[$\boxed{2}$]] ↦ number → [[y]] → list [[$\boxed{3}$]] ↦ [[y]] → list [[x]] ↦ number [[$\boxed{5}$]] ↦ list [[$\boxed{4}$]] ↦ list [[f]] ↦ [[$\boxed{6}$]] → list
Paso 5	[[$\boxed{6}$]] = [[y]] list = [[$\boxed{6}$]]	[[$\boxed{1}$]] ↦ [[$\boxed{6}$]] → list → number → [[$\boxed{6}$]] → list [[$\boxed{2}$]] ↦ number → [[$\boxed{6}$]] → list

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

Problema IV

En ninguno de los dos casos cambia la regla de tipado. Para justificarlo, primero recordemos las reglas correspondientes.

- Definición de funciones:

$$\frac{\Gamma[i \leftarrow \tau_1\ ] \vdash b : \tau_2}{\Gamma \vdash \{fun\ \{i : \tau_1\} : \tau_2\ b\} : (\tau_1 \to \tau_2)}$$

- Aplicación de funciones:

$$\frac{\Gamma \vdash f : (\tau_1 \to \tau_2) \qquad \Gamma \vdash a : \tau_1}{\Gamma \vdash \{f\ a\} : \tau_2}$$

Veamos que para determinar el tipo de las expresiones, nunca se hace alusión directa a la evaluación. En las funciones se habla en cambio de tipos de entrada y tipos de salida que no restringimos a ser el mismo. Podríamos pensar para nuestro lenguaje perezoso que en la aplicación de función si recibimos una expresión en lugar de un valor del tipo de entrada esperado tendríamos que hacer ciertas modificaciones a la regla de tipado, sin embargo ya sabemos que uno de los puntos estrictos de un lenguaje perezoso ocurre precisamente en una aplicación de función, por lo que se está forzando la evaluación y la función recibe el resultado de dicha evaluación que debe cumplir el tipo especificado. Esto para la entrada. Para el tipo de salida, si exigimos un resultado, aún con un lenguaje perezoso no podemos devolver una expresión sin evaluar, por lo que no necesitamos modificar la regla de tipado. Tanto el argumento de la función como el resultado tendrá la misma forma para ambas evaluaciones. Por lo tanto, no es necesario cambiar la regla original para la versión perezosa del lenguaje.

Problema V

Explícito

En el polimorfismo explícito los tipos de los objetos deben ser declarados en su inicialización para cada uno antes de ser operados. Esto nos lleva a una programación repetitiva, pesada y mucho más restrictiva. El control del código puede disminuir con el uso de este polimorfismo por la obligatoriedad de hacer presente el tipo. Sin embargo, en su uso se encuentran ciertas ventajas. Para empezar,la creación de nuevas clases de objetos sin afectar la forma ya existente de la que procede. También la facilidad con la que se recicla el código es mayor y por lo general los lenguajes proveen de ciertas herramientas para evitar la presencia explícita del tipo más que al momento de inicializarlo.

Implícito

Uno de los ejes en los lenguajes de programación con polimorfismo implícito es la utilización de código en común si la abstracción semántica lo permite. Ello se traduce en una de las ventajas más grandes sobre el otro, la simplicidad y la versatilidad del código. Es resumido en "nunca repitas el código". Si encontramos una o más secciones de código repetidas es porque el autor del código falló en identificar una abstracción en común para diferentes problemas. Sin embargo, para vislumbrar un par de desventajas podemos considerar el siguiente escenario: Bajo el contexto de programación en *Java*, consideremos una lista con capacidad para almacenar objetos de cualquier tipo. Para evitar la repetición del código, es posible hacer uso de la jerarquía de tipos de acuerdo al supertipo más general que es *Object*. Ahora, al momento de tomar elementos de la lista para operar con ellos, nos veremos obligados a hacer un *cast* a su tipo específico para el contexto particular en el que será utilizado. Por supuesto, obtendremos un código desordenado y perderemos precisión

sobre las tareas que los objetos son capaces de realizar, de hecho puede que nos veamos afectados por un nuevo factor de incertidumbre. Sin embargo, este tipo de desventajas es para situaciones o circunstancias menos generales, por lo que no significa una contraparte fuerte en comparación con las ventajas que ofrece.

Problema VI