PLP - Primer Parcial - 1er cuatrimestre de 2017

Este examen se aprueba obteniendo al menos **65 puntos** en total, y al menos **5 puntos** por cada tema. Poner nombre, apellido y número de orden en cada hoja, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

Ejercicio 1 - Programación funcional (35 puntos)
Las matrices infinitas pueden ser representadas como funciones:
`type MatrizInfinita a = Int->Int->a`
donde el primer argumento corresponde a la fila, el segundo a la columna y el resultado al valor contenido en la celda correspondiente.
Por ejemplo, las siguientes definiciones:

```
identidad = \i j->if i==j then 1 else 0
cantor = \x y->(x+y)*(x+y+1)`div`2+y
pares = \x y->(x,y)
```

corresponden a las matrices:

1	0	0	⋯
0	1	0	⋯
0	0	1	⋯
⋮	⋮	⋮	⋱

identidad

0	2	5	⋯
1	4	8	⋯
3	7	12	⋯
⋮	⋮	⋮	⋱

cantor

(0,0)	(0,1)	(0,2)	⋯
(1,0)	(1,1)	(1,2)	⋯
(2,0)	(2,1)	(2,2)	⋯
⋮	⋮	⋮	⋱

pares

Definir las siguientes funciones:

a. `fila::Int->MatrizInfinita a->[a]` y `columna::Int->MatrizInfinita a->[a]` que, dado un índice, devuelven respectivamente la fila o la columna correspondiente en la matriz (en forma de lista infinita). Por ejemplo, `fila 0 identidad` devuelve la lista con un 1 seguido de infinitos 0s.

b. `trasponer::MatrizInfinita a->MatrizInfinita a`, que dada una matriz devuelve su traspuesta.

c. `mapMatriz::(a->b)->MatrizInfinita a->MatrizInfinita b`,
`filterMatriz::(a->Bool)->MatrizInfinita a->[a]` y
`zipWithMatriz::(a->b->c)->MatrizInfinita a->MatrizInfinita b->MatrizInfinita c`, que se comportan como `map`, `filter` y `zipWith` respectivamente, pero aplicadas a matrices infinitas. En el caso de `filterMatriz` no importa el orden en el que se devuelvan los elementos, pero se debe pasar una y sólo una vez por cada posición de la matriz.

d. `zipMatriz::MatrizInfinita a->MatrizInfinita b->MatrizInfinita (a,b)`.

e. `suma::Num a=>[[a]] -> MatrizInfinita a-> [[a]]`, que dada una matriz, representada como la lista de sus filas, y una matriz infinita, devuelva la suma de las matrices representada como la lista de sus filas. Recordamos que la suma de matrices se define como la suma celda a celda.

Ejercicio 2 - Cálculo Lambda Tipado (35 puntos)

El objetivo del ejercicio es extender este lenguaje para soportar tipos enumerados.

Los conjuntos de tipos y términos se extienden de la siguiente manera:

$$\sigma \ ::= \ ... \mid \mathsf{enum}(c_1, ..., c_n) \quad M \ ::= \ c_1 \mid ... \mid c_n \mid \mathsf{switch}_{\mathsf{enum}(c_1,...,c_n)} \ M \colon c_i \rightsquigarrow M; \ ...; \ c_j \rightsquigarrow M \ \mathsf{default}\colon M$$

donde $c_1, ..., c_n$ y $c_i, ..., c_j$ son constantes que representan los elementos del tipo enumerado. Cada constante pertenece a un único enumerado.

Pueden definirse macros para nombrar tipos enumerados. Por ejemplo:

$\mathsf{día} \stackrel{\mathsf{def}}{=} \mathsf{enum}(\mathit{Lunes}, \mathit{Martes}, \mathit{Miércoles}, \mathit{Jueves}, \mathit{Viernes}, \mathit{Sábado}, \mathit{Domingo})$

Se desea definir una funcionalidad de switch al estilo de C, Java, etc., que funcione de manera similar a los case definidos para otros tipos pero sea más flexible, permitiendo poner los casos en cualquier orden e incluso omitir casos, dando un resultado por defecto si el término analizado no coincide con ninguno de los casos. Por ejemplo:

$\mathsf{esFinde} \stackrel{\mathsf{def}}{=} \lambda x \colon \mathsf{día}.\mathsf{switch}_{\mathsf{día}} \ x \colon \mathit{Sábado} \rightsquigarrow \mathsf{True}; \mathit{Domingo} \rightsquigarrow \mathsf{True} \ \mathsf{default}\colon \mathsf{False}$

Restricción: para poder usar un switch, todos los casos deben ser constantes pertenecientes al enumerado, y ser distintas entre sí. Algunos ejemplos que no deberían admitirse son:

$\mathsf{switch}_{\mathsf{día}} \ \mathit{Lunes} \colon \mathit{Sábado} \rightsquigarrow \mathsf{True}; \mathit{Sábado} \rightsquigarrow \mathsf{True} \ \mathsf{default}\colon \mathsf{False}$ (tiene dos constantes iguales).

$\mathsf{switch}_{\mathsf{día}} \ \mathit{Lunes} \colon \mathit{Sábado} \rightsquigarrow \mathsf{True}; (\lambda x : \mathsf{Nat}.\mathit{Domingo})0 \rightsquigarrow \mathsf{True} \ \mathsf{default}\colon \mathsf{False}$ (El segundo caso no es una constante).

$\mathsf{switch}_{\mathsf{día}} \ \mathit{Lunes} \colon \mathit{Marzo} \rightsquigarrow \mathsf{True}; \mathit{Sábado} \rightsquigarrow \mathsf{True} \ \mathsf{default}\colon \mathsf{False}$ (Marzo no es un día de la semana).

a) Introducir las reglas de tipado para la extensión propuesta.

b) Utilizar las reglas de tipado definidas para chequear el siguiente juicio de tipado: $\emptyset \triangleright \mathsf{esFinde} \colon \mathsf{día} \to \mathsf{Bool}$

c) Indicar formalmente como se modifica el conjunto de valores y dar la semántica operacional de a un paso para la extensión propuesta.

d) Explicar qué problemas traería introducir un switch similar al actual, pero sin opción por defecto.

Ejercicio 3 - Inferencia de tipos (30 puntos)

En este ejercicio extenderemos el algoritmo de inferencia para tipar listas alternadas.

La sintaxis y la extensión del conjunto de tipos para la extensión propuesta son las siguientes:

$$M, N, O \ ::= \ ... \mid M \mathbin{:} N \mid [\,]_{\sigma,\tau} \mid \mathtt{foldr}_{\mathtt{ALT}} \, M \, \mathrm{base} \rightsquigarrow N; \mathrm{rec}(h, r) \rightsquigarrow O \qquad\qquad \sigma \ ::= \ ... \mid [\sigma, \tau]$$

Y las reglas de tipado son la siguientes:

$$\frac{}{\Gamma \triangleright [\,]_{\sigma,\tau} : [\sigma, \tau]} \qquad \frac{\Gamma \triangleright M : \sigma \qquad \Gamma \triangleright N : [\tau, \sigma]}{\Gamma \triangleright M \mathbin{:} N : [\sigma, \tau]} \qquad \frac{\Gamma \triangleright M : [\sigma, \tau] \qquad \Gamma \triangleright N : \rho \qquad \Gamma \cup \{h : \sigma, r : \rho\} \triangleright O : \rho}{\Gamma \triangleright \mathtt{foldr}_{\mathtt{ALT}} \ M \ \mathrm{base} \rightsquigarrow N; \mathrm{rec}(h, r) \rightsquigarrow O : \rho}$$

a) Extender el algoritmo de inferencia para soportar el tipado de la nueva estructura. Recordar que los subíndices de $[\,]_{\sigma,\tau}$, al igual que todas las anotaciones de tipos, no aparecen en la entrada del algoritmo.

b) Aplicar el algoritmo extendido con el método del árbol para tipar la siguiente expresión:

$$\mathtt{foldr}_{\mathtt{ALT}} \ 0 \mathbin{:} \mathsf{True} \mathbin{:} [\,] \ \mathrm{base} \rightsquigarrow 0; \mathrm{rec}(h, r) \rightsquigarrow \mathsf{Succ}(r)$$