Programación Funcional

Ejercicios de Práctica Nro. 14

Lambda cálculo

Aclaraciones:

- Los ejercicios siguen un orden de complejidad creciente, y cada uno puede servir a los siguientes. No se recomienda saltear ejercicios sin consultar antes a un docente.
- Recordar que se pueden aprovechar en todo momento las funciones ya definidas, tanto las de esta misma práctica como las de prácticas anteriores.
- Probar todas las implementaciones, al menos en una consola interactiva.
- Es sumamente aconsejable resolver los ejercicios utilizando primordialmente los conceptos y metodologías vistos en clase, dado que los exámenes de la materia evalúan principalmente este aspecto. Para utilizando formas alternativas al resolver los ejercicios consultar a los docentes.

Parte I

Ejercicio 1) Escribir 10 expresiones lambda de diferente complejidad. Al menos una debe tener al menos 3 lambdas y no tener aplicaciones; al menos una debe tener al menos 3 lambdas y tener aplicaciones; al menos una debe tener al menos 3 aplicaciones; al menos una debe tener 4 lambdas no anidados; al menos una debe tener un lambda utilizado en una aplicación, pero dentro de un lambda mayor.

Ejercicio 2) Indicar cuáles de las siguientes expresiones son expresiones lambda según la definición inductiva y cuáles no. Justificar.

a. *(λx.(λy.x y) (z (λw.w)))*
b. *(f x y)*
c. *(λb.λm.λn.((b m) n))*
d. *(λf.(λx.f (f x)))*
e. *(λf z.f (λy.y) z) (λx.x) (λg h w.g (h w))*
f. *(λa.(λb.(b c)))*
g. *(λq u i e n.n (e u q u e n))*
h. *(λx.x x x) (λx.x x x) (λx.x x x)*
i. *(λf.(λx.f x))*
j. *(h (λf.(λx.(λy.((f x) y))))*
k. *(((λf.(λz.((f (λy.y)) z))) (λx.x)) (λg.(λh.(λw.(g (h w))))))*
l. *(x (y z))*
m. *((λx.x) x)*
n. *(λx.x) (λx.x) ((λx.(λy.x)) w)*

Ejercicio 3) Decir cuáles de las expresiones lambda del punto anterior son expresiones cerradas. Justificar.

Ejercicio 4) Dar las expresiones lambda correctas según las convenciones de notación para aquellas expresiones del ejercicio 3 que no lo sean.

Ejercicio 5) Reducir las expresiones lambda del ejercicio 3 que puedan ser reducidas, hasta su forma normal (si existe).

Ejercicio 6) Decir cuáles de las siguientes expresiones lambda son equivalentes entre sí, por renombre de variables (*alfa conversión*).

a. *(λx.x)*
b. *(λf x.f (f x))*
c. *(λf.λx.f x)(λz.z)*
d. *x*
e. *(λw.z)*
f. *(λf. (λy.y) z)*
g. *λy.y*
h. *(λx y.x (x y))*
i. *z*
j. *((λa.(λb.(a b)))(λf.f))*
k. *(λw.x)*
l. *(λf.z)*
m. *(λx.x) z*

Parte II

ACLARACIÓN: A partir de este punto, cuando se piden dar expresiones, solamente se deben utilizar expresiones cerradas (pueden utilizarse las convenciones de notación).

Ejercicio 7) Dar expresiones lambda en forma normal que se puedan usar como definiciones para las siguientes funciones de las prácticas 1 y 2.

a. **compose**
b. **twice**
c. **flip**
d. **subst**
e. **cuatroVeces** (equivalente a `(twice twice)` y `(many 4)`; tener en cuenta que no se pueden utilizar números en esta parte, por no estar definidos aún)
f. **appDup'** (equivalente a `(appDup . uncurry)`; tener en cuenta que no se pueden utilizar pares en esta parte, por no estar definidos aún)

Ejercicio 8) Reducir las siguientes lambda expresiones a forma normal.

a. **apply twice twice**
b. **flip twice**
c. **id** *(***flip id***)*
d. **compose twice** *(λs z.s (s z)))*

Ejercicio 9) Dar las definiciones en lambda cálculo de **not** y **or** para **Booleanos.**

Ejercicio 10) Para los **pares**:

a. Validar que la representación canónica cumple la especificación.
b. Definir las funciones **curry**, **uncurry** y **swap**.

Ejercicio 11) Definir **tuplas** de 3 y 4 elementos en lambda cálculo siguiendo el mismo procedimiento que para pares.

Ejercicio 12) Para las listas:

a. Validar que la representación canónica cumple la especificación.
b. Definir en lambda cálculo **length**, **sum**, **prod**, **append**, **reverse** y **filter.**
c. Definir en lambda cálculo **recr**, **head**, **tail**, **null.**

Ejercicio 13) Para ExpA:

a. Validar que la representación canónica cumple la especificación.
b. Definir en lambda cálculo **esSuma**, **esProd**, **simplEA**, **evalExpA** para **ExpA.**

Ejercicio 14) Definir en lambda cálculo la especificación de los tipos **Tree** y **TG**, dar representaciones canónicas y validar que cumplen las especificaciones dadas.

Ejercicio 15) Definir en lambda cálculo la especificación de los tipos **Maybe**, **Color**, **Dir** y otros enumerativos a elección (**Meses**, **Días**, etc.), dar representaciones canónicas y validar que cumplen las especificaciones dadas.

Ejercicio 16) Para los números naturales:

a. Validar que la representación canónica cumple la especificación.
b. Definir en lambda cálculo **mult**, **isZero**, **recNat**, **pred**.

Ejercicio 17) (DIFÍCIL) Para **árboles generales**:

a. Definir en lambda cálculo la especificación de los mismos, dar la representación canónica y validar que cumple las especificaciones dadas.
b. Definir el lambda cálculo las funciones **sumAG**, **mirrorAG**, **depthAG** para **árboles generales.**

Ejercicio 18) (MUY DIFÍCIL) Para el tipo **LIS** (y sus auxiliares):

a. Definir en lambda cálculo la especificación de los tipos necesarios, dar las representaciones canónicas y validar que cumplen las especificaciones dadas.
   AYUDA: require recursión mutua expresada como recursión estructural (no visto).
   AYUDA 2: para las variables usar números naturales en lugar de **Strings**.
b. Dar la función de **evalPrograma**.
   AYUDA: requiere recursión general.