Práctica N° 2 - Introducción al cálculo lambda tipado

Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.

A menos que se especifiquen las extensiones a utilizar, trabajaremos con el cálculo λ con los tipos Bool, Nat y funciones.

Notación para este segmento de la materia:

- las letras $M, N, O, P, \ldots$ denotan términos.
- las letras $V, W, Y, \ldots$ denotan valores.
- las letras griegas $\sigma, \tau, \rho, \pi, \ldots$ denotan tipos.

Gramáticas a tener en cuenta:

- Términos
  $M ::= x \mid \lambda x\colon \sigma.\ M \mid M\ M \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M \mid 0 \mid \mathsf{succ}(M) \mid \mathsf{pred}(M) \mid \mathsf{isZero}(M)$

  Donde la letra x representa un *nombre de variable* arbitrario. Tales nombres se toman de un conjunto infinito dado $\mathfrak{X} = \{w, w_1, w_2, \ldots, x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots\}$
- Tipos
  $\sigma ::= \mathsf{Bool} \mid \mathsf{Nat} \mid \sigma \to \sigma$

Sintaxis

Ejercicio 1 ★

Determinar qué expresiones son sintácticamente válidas (es decir, pueden ser generadas con las gramáticas presentadas) y determinar a qué categoría pertenecen (expresiones de términos o expresiones de tipos):

1. x
2. $x\ x$
3. M
4. $M\ M$
5. $\mathsf{true}\ \mathsf{false}$
6. $\mathsf{true}\ \mathsf{succ}(\mathsf{false}\ \mathsf{true})$
7. $\lambda x.\mathsf{isZero}(x)$
8. $\lambda x\colon \sigma.\ \mathsf{succ}(x)$
9. $\lambda x\colon \mathsf{Bool}.\ \mathsf{succ}(x)$
10. $\lambda x\colon \mathsf{if}\ \mathsf{true}\ \mathsf{then}\ \mathsf{Bool}\ \mathsf{else}\ \mathsf{Nat}.\ x$
11. σ
12. Bool
13. $\mathsf{Bool} \to \mathsf{Bool}$
14. $\mathsf{Bool} \to \mathsf{Bool} \to \mathsf{Nat}$
15. $(\mathsf{Bool} \to \mathsf{Bool}) \to \mathsf{Nat}$
16. $\mathsf{succ}\ \mathsf{true}$
17. $\lambda x\colon \mathsf{Bool}.\ \mathsf{if}\ 0\ \mathsf{then}\ \mathsf{true}\ \mathsf{else}\ 0\ \mathsf{succ}(\mathsf{true})$

Ejercicio 2

Mostrar un término que utilice al menos una vez **todas** las reglas de generación de la gramática y exhibir su *árbol sintáctico*.

Ejercicio 3 ★

1. Marcar las ocurrencias del término x como subtérmino en $\lambda x\colon \mathsf{Nat}.\ \mathsf{succ}((\lambda x\colon \mathsf{Nat}.\ x)\ x)$.
2. ¿Ocurre x_1 como subtérmino en $\lambda x_1\colon \mathsf{Nat}.\ \mathsf{succ}(x_2)$?
3. ¿Ocurre $x\ (y\ z)$ como subtérmino en $u\ x\ (y\ z)$?

Ejercicio 4 ★

Para los siguientes términos:

1. $u\ x\ (y\ z)\ (\lambda v\colon \mathsf{Bool}.\ v\ y)$
2. $(\lambda x\colon \mathsf{Bool} \to \mathsf{Nat} \to \mathsf{Bool}.\ \lambda y\colon \mathsf{Bool} \to \mathsf{Nat}.\ \lambda z\colon \mathsf{Bool}.\ x\ z\ (y\ z))\ u\ v\ w$
3. $w\ (\lambda x\colon \mathsf{Bool} \to \mathsf{Nat} \to \mathsf{Bool}.\ \lambda y\colon \mathsf{Bool} \to \mathsf{Nat}.\ \lambda z\colon \mathsf{Bool}.\ x\ z\ (y\ z))\ u\ v$

Se pide:

1. Insertar todos los paréntesis de acuerdo a la convención usual.
2. Dibujar el árbol sintáctico de cada una de las expresiones.
3. Indicar en el árbol cuáles ocurrencias de variables aparecen ligadas y cuáles libres.
4. ¿En cuál de los términos anteriores ocurre la siguiente expresión como subtérmino?
   $(\lambda x\colon \mathsf{Bool} \to \mathsf{Nat} \to \mathsf{Bool}.\ \lambda y\colon \mathsf{Bool} \to \mathsf{Nat}.\ \lambda z\colon \mathsf{Bool}.\ x\ z\ (y\ z))\ u$

Tipado

Ejercicio 5 *(Derivaciones ★)*

Demostrar o explicar por qué no puede demostrarse cada uno de los siguientes juicios de tipado.

1. $\emptyset \triangleright \mathsf{if\ true\ then}\ 0\ \mathsf{else\ succ}(0)\ :\ \mathsf{Nat}$
2. $\{x : \mathsf{Nat}, y : \mathsf{Bool}\} \triangleright \mathsf{if\ true\ then\ false\ else}\ (\lambda z\colon \mathsf{Bool}.\ z)\ \mathsf{true}\ :\ \mathsf{Bool}$
3. $\emptyset \triangleright \mathsf{if}\ \lambda x\colon \mathsf{Bool}.\ x\ \mathsf{then}\ 0\ \mathsf{else\ succ}(0)\ :\ \mathsf{Nat}$
4. $\{x : \mathsf{Bool} \to \mathsf{Nat}, y : \mathsf{Bool}\} \triangleright x\ y\ :\ \mathsf{Nat}$

Ejercicio 6

Determinar qué tipo representa σ en cada uno de los siguientes juicios de tipado.

1. $\emptyset \triangleright \mathsf{succ}(0)\ :\ \sigma$
2. $\emptyset \triangleright \mathsf{isZero}(\mathsf{succ}(0))\ :\ \sigma$
3. $\emptyset \triangleright \mathsf{if}\ (\mathsf{if\ true\ then\ false\ else\ false})\ \mathsf{then}\ 0\ \mathsf{else\ succ}(0)\ :\ \sigma$

Ejercicio 7 ★

Determinar qué tipos representan σ y τ en cada uno de los siguientes juicios de tipado. Si hay más de una solución, o si no hay ninguna, indicarlo.

1. $\{x\colon \sigma\} \triangleright \mathsf{isZero}(\mathsf{succ}(x))\ :\ \tau$
2. $\emptyset \triangleright (\lambda x\colon \sigma.\ x)(\lambda y\colon \mathsf{Bool}.\ 0)\ :\ \sigma$
3. $\{y\colon \tau\} \triangleright \mathsf{if}\ (\lambda x\colon \sigma.\ x)\ \mathsf{then}\ y\ \mathsf{else\ succ}(0)\ :\ \sigma$
4. $\{x\colon \sigma\} \triangleright x\ y\ :\ \tau$
5. $\{x\colon \sigma, y\colon \tau\} \triangleright x\ y\ :\ \tau$
6. $\{x\colon \sigma\} \triangleright x\ \mathsf{true}\ :\ \tau$
7. $\{x\colon \sigma\} \triangleright x\ \mathsf{true}\ :\ \sigma$
8. $\{x\colon \sigma\} \triangleright x\ x\ :\ \tau$

Ejercicio 8

Mostrar un término que no sea tipable y que no tenga variables libres ni abstracciones.

Ejercicio 9

Mostrar un juicio de tipado que sea demostrable en el sistema actual pero que no lo sea al cambiar (T-ABS) por la siguiente regla. Mostrar la demostración del juicio original.

$$\frac{\Gamma \triangleright M \ : \ \tau}{\Gamma \triangleright \lambda x\colon \sigma.\ M \ : \ \sigma \to \tau}\ \text{T-ABS2}$$

Semántica

Ejercicio 10 ★

Sean $\sigma,\ \tau,\ \rho$ tipos. Según la definición de sustitución, calcular:

1. $(\lambda y\colon \sigma.\ x\ (\lambda x\colon \tau.\ x))\{x \leftarrow (\lambda y\colon \rho.\ x\ y)\}$
2. $(y\ (\lambda v\colon \sigma.\ x\ v))\{x \leftarrow (\lambda y\colon \tau.\ v\ y)\}$

Renombrar variables en ambos términos para no cambiar el significado del término.

Ejercicio 11 *(Valores)* ★

Dado el conjunto de valores visto en clase:

$$V := \lambda x\colon \sigma.\ M \mid \mathsf{true} \mid \mathsf{false} \mid 0 \mid \mathsf{succ}(V)$$

Determinar si cada una de las siguientes expresiones es o no un valor:

1. $(\lambda x\colon \mathsf{Bool}.\ x)\ \mathsf{true}$
2. $\lambda x\colon \mathsf{Bool}.\ \underline{2}$
3. $\lambda x\colon \mathsf{Bool}.\ \mathsf{pred}(\underline{2})$
4. $\lambda y\colon \mathsf{Nat}.\ (\lambda x\colon \mathsf{Bool}.\ \mathsf{pred}(\underline{2}))\ \mathsf{true}$
5. x
6. $\mathsf{succ}(\mathsf{succ}(0))$

Ejercicio 12 *(Programa, Forma Normal)* ★

Para el siguiente ejercicio, considerar el cálculo **sin** la regla $\mathsf{pred}(0) \to 0$

Un *programa* es un término que tipa en el contexto vacío (es decir, no puede contener variables libres).

Para cada una de las siguientes expresiones

(a) Determinar si puede ser considerada un **programa**.

(b) Si vale (a), ¿Cuál es el resultado de su evaluación? Determinar si se trata de una forma normal, y en caso de serlo, si es un **valor** o un **error**.

1. $(\lambda x\colon \mathsf{Bool}.\ x)\ \mathsf{true}$
2. $\lambda x\colon \mathsf{Nat}.\ \mathsf{pred}(\mathsf{succ}(x))$
3. $\lambda x\colon \mathsf{Nat}.\ \mathsf{pred}(\mathsf{succ}(y))$
4. $(\lambda x\colon \mathsf{Bool}.\ \mathsf{pred}(\mathsf{isZero}(x)))\ \mathsf{true}$
5. $(\lambda f\colon \mathsf{Nat} \to \mathsf{Bool}.\ f\ 0)\ (\lambda x\colon \mathsf{Nat}.\ \mathsf{isZero}(x))$
6. $(\lambda f\colon \mathsf{Nat} \to \mathsf{Bool}.\ x)\ (\lambda x\colon \mathsf{Nat}.\ \mathsf{isZero}(x))$
7. $(\lambda f\colon \mathsf{Nat} \to \mathsf{Bool}.\ f\ \mathsf{pred}(0))\ (\lambda x\colon \mathsf{Nat}.\ \mathsf{isZero}(x))$

8. $\mathsf{fix}\ (\lambda y\colon \mathsf{Nat}.\ \mathsf{succ}(y))$
9. $\mathsf{letrec}\ f = \lambda x\colon \mathsf{Nat}.\ \mathsf{succ}(f\ x)\ \mathsf{in}\ f\ 0$

Ejercicio 13 *(Determinismo)*

1. ¿Es cierto que la relación definida $\to$ es determinística (o una función parcial)?
   Más precisamente, ¿pasa que si $M \to N$ y $M \to N'$ entonces $N = N'$?
2. ¿Vale lo mismo con muchos pasos? Es decir, ¿es cierto que si $M \twoheadrightarrow M'$ y $M \twoheadrightarrow M''$ entonces $M' = M''$?
3. ¿Acaso es cierto que si $M \to M'$ y $M \twoheadrightarrow M''$ entonces $M' = M''$?

Ejercicio 14

1. ¿Da lo mismo evaluar $\mathsf{succ}(\mathsf{pred}(M))$ que $\mathsf{pred}(\mathsf{succ}(M))$? ¿Por qué?
2. ¿Es verdad que para todo M vale que $\mathsf{isZero}(\mathsf{succ}(M)) \twoheadrightarrow \mathsf{false}$? Si no lo es, ¿para qué términos vale?
3. ¿Para qué términos M vale que $\mathsf{isZero}(\mathsf{pred}(M)) \twoheadrightarrow \mathsf{true}$? (Hay infinitos).

Ejercicio 15

Al agregar la siguiente regla para las abstracciones:

$$\frac{M \to M'}{\lambda x\colon \tau.\ M \to \lambda x\colon \tau.\ M'}\ E-ABS$$

1. Repensar el conjunto de valores para respetar esta modificación, pensar por ejemplo si $(\lambda x\colon \mathsf{Bool}.\ Id_{bool}\ \mathsf{true})$ es o no un valor.
2. ¿Qué reglas deberían modificarse para no perder el determinismo?
3. Utilizando la nueva regla y los valores definidos, reducir la siguiente expresión
   $(\lambda x\colon \mathsf{Nat} \to \mathsf{Nat}.\ x\ \underline{23})\ (\lambda x\colon \mathsf{Nat}.\ 0)$
   ¿Qué se puede concluir entonces? ¿Es seguro o no agregar esta regla?

Ejercicio 16

La variante del Cálculo Lambda vista en clase utiliza el modo de reducción call-by-value: para reducir una aplicación a forma normal, se reduce primero el argumento y luego se ejecuta la aplicación. ¿Cómo cambiaría el cálculo si en lugar de esto se utilizara la estrategia call-by-name (es decir, reduciendo la aplicación antes que el argumento)[1]? Mencionar qué reglas se modifican, y reescribirlas para adaptarlas a esta estrategia.

Reducir el siguiente término a forma normal utilizando la estrategia:

$comp\ (\lambda x\colon \mathsf{Nat}.\ \mathsf{succ}(x))\ (\lambda x\colon \mathsf{Nat}.\ \mathsf{succ}(x))\ \underline{5}$

donde $comp \stackrel{\text{def}}{=} \lambda f\colon \mathsf{Nat} \to \mathsf{Nat}.\ \lambda g\colon \mathsf{Nat} \to \mathsf{Nat}.\ \lambda x\colon \mathsf{Nat}.\ f\ (g\ x)$

Extensiones

En esta sección puede asumirse, siempre que sea necesario, que el cálculo ha sido extendido con la suma de números naturales $(M + N)$, con las siguiente reglas de tipado y semántica:

$$\frac{\Gamma \rhd M\colon \mathsf{Nat} \quad \Gamma \rhd N\colon \mathsf{Nat}}{\Gamma \rhd M + N\colon \mathsf{Nat}}\ \text{T-+}$$

$$\frac{M \to M'}{M + N \to M' + N}\ \text{E-+}_1 \qquad \frac{N \to N'}{V + N \to V + N'}\ \text{E-+}_2 \qquad \frac{}{V + 0 \to V}\ \text{E-+0} \qquad \frac{}{V_1 + \mathsf{succ}(V_2) \to \mathsf{succ}(V_1) + V_2}\ \text{E-+Succ}$$

[1] Pista: la idea de la reducción call-by-name consiste en lo siguiente: en el caso de un término con la forma $(\lambda x\colon \sigma.M)\ N$, en lugar de reducir primero N (que es lo que haría la reducción call-by-value), se resuelve la aplicación directamente sobre el parámetro sin reducir; es decir, se reduce $(\lambda x\colon \sigma.M)\ N$ a $M\{x \leftarrow N\}$.

Ejercicio 17 ★

Este ejercicio extiende el Cálculo Lambda tipado con listas. Comenzamos ampliando el conjunto de tipos:

$$\sigma \quad ::= \quad \ldots \mid [\sigma]$$

donde $[\sigma]$ representa el tipo de las listas cuyas componentes son de tipo σ. El conjunto de términos ahora incluye:

$$M, N, O \quad ::= \quad \ldots \mid [\,]_\sigma \mid M :: N \mid \texttt{case}\, M \;\texttt{of}\; \{[\,] \rightsquigarrow N \mid h :: t \rightsquigarrow O\} \mid \texttt{foldr}\, M\, \text{base} \rightsquigarrow N; \text{rec}(h, r) \rightsquigarrow O$$

donde

- $[\,]_\sigma$ es la lista vacía cuyos elementos son de tipo σ;
- $M :: N$ agrega M a la lista N;
- $\texttt{case}\ M\ \texttt{of}\ \{[\,] \rightsquigarrow N \mid h :: t \rightsquigarrow O\}$ es el observador de listas. Por su parte, los nombres de variables que se indiquen luego del $\mid$ (h y t en este caso) son variables que pueden aparecer libres en O y deberán ligarse con la cabeza y cola de la lista respectivamente;
- $\texttt{foldr}\ M\ \ \text{base} \rightsquigarrow N;\ \text{rec}(h, r) \rightsquigarrow O$ es el operador de recursión estructural (no currificado). Los nombres de variables indicados entre parentesis (h y r en este caso) son variables que pueden aparecer libres en O y deberán ser ligadas con la cabeza y el resultado de la recursión respectivamente.

Por ejemplo,

- $\texttt{case}\ 0 :: \mathsf{succ}(0) :: [\,]_{\mathsf{Nat}}\ \texttt{of}\ \{[\,] \rightsquigarrow \mathsf{false} \mid x :: xs \rightsquigarrow \mathsf{isZero}(x)\} \twoheadrightarrow \mathsf{true}$
- $\texttt{foldr}\, \underline{1} :: \underline{2} :: \underline{3} :: (\lambda x\colon [\mathsf{Nat}].\ x)\ [\,]_{\mathsf{Nat}}\ \text{base} \rightsquigarrow 0;\ \text{rec}(head, rec) \rightsquigarrow head + rec \twoheadrightarrow \underline{6}$

1. Mostrar el árbol sintáctico para los dos ejemplos dados.
2. Agregar reglas de tipado para las nuevas expresiones.
3. Demostrar el siguiente juicio de tipado (recomendación: marcar variables libres y ligadas en el término antes de comenzar).

   $\{x : \mathsf{Bool}, y : [\mathsf{Bool}]\} \triangleright \texttt{foldr}\, x :: x :: y\ \ \text{base} \rightsquigarrow y;\ \text{rec}(y, x) \rightsquigarrow \mathsf{if}\ y\ \mathsf{then}\ x\ \mathsf{else}\ [\,]_{\mathsf{Bool}}\ :\ [\mathsf{Bool}]$
4. Mostrar cómo se extiende el conjunto de valores. Estos deben reflejar la forma de las listas que un programa podría devolver.
5. Agregar los axiomas y reglas de reducción asociados a las nuevas expresiones.

Ejercicio 18 ★

A partir de la extensión del ejercicio 17, definir una nueva extensión que incorpore expresiones de la forma $\texttt{map}(M, N)$, donde N es una lista y M una función que se aplicará a cada uno de los elementos de N.
Importante: tener en cuenta las anotaciones de tipos al definir las reglas de tipado y semántica.

Ejercicio 19 ★

La aplicación parcial sobre funciones currificadas es una de las ventajas de los lenguajes funcionales, como el cálculo lambda tipado. Sin embargo, el mecanismo del cálculo lambda (que se repite en la mayoría de los lenguajes funcionales como Haskell) es limitado, ya que la aplicación parcial debe hacerse siempre en el orden de los argumentos. Por ejemplo, si tenemos la función potencia, podemos usarla con aplicación parcial para definir la función cuadrado, si su primer parámetro es el exponente, o la función dosALa si su primer parámetro es la base, pero no podemos hacer ambas cosas con la misma función potencia.

Para solucionar este problema introduciremos el cálculo μ, que es igual al cálculo lambda en todo, excepto en que el mecanismo para construir funciones ($\lambda x.M$) y el mecanismo para aplicarlas ($M\ N$) serán sustituidos por un nuevo mecanismo de construcción ($\mu x_1, ..., x_n.M$) y de aplicación ($M\ \#_i\ N$). Estos cambios también introducen un cambio en el sistema de tipos: en lugar de tener $\sigma \to \tau$ tendremos $\{\sigma_1, ..., \sigma_n\} \to \tau$. Notar que $\{\sigma_1, ..., \sigma_n\}$ no es un nuevo tipo, sino sólo una parte del nuevo tipo para funciones.

La sintaxis del cálculo μ y su conjunto de tipos, entonces, serán los siguientes:

$$M, N \ ::= \ ... \mid \mu x_1 : \sigma_1, ..., x_n : \sigma_n.M \mid M\ \#_i\ N \qquad \sigma_1 \ldots \sigma_n, \tau \ ::= \ ... \mid \{\sigma_1, ..., \sigma_n\} \to \tau$$

El término $\mu x_1 : \sigma_1, ..., x_n : \sigma_n.M$ sirve para construir una nueva función de n parámetros ordenados y el operador $\#_i$ sirve para aplicar el i-ésimo parámetro. Notar que si la cantidad de parámetros de una función es mayor a 1, al aplicarla se obtiene una nueva función con un parámetro menos, pero si la cantidad de parámetros es exactamente 1, al aplicarla se obtiene su valor de retorno. Notar además que que el orden de los tipos de los argumentos es importante: por ejemplo, $\{nat, nat, bool\} \to nat$ y $\{bool, nat, nat\} \to nat$ no son el mismo tipo.

1. Introducir las reglas de tipado para la extensión propuesta.
2. Dar formalmente la extensión de los valores e introducir las reglas de semántica para la extensión propuesta.
3. Escribir las construcciones básicas del cálculo lambda (λ y aplicación) como macros del cálculo μ para mostrar que este último puede emularlo.

Ejercicio 20

Definir una extensión que permita "unir" un registro $\{x_1 = M_1, \ldots, x_m = M_m\}$ con otro registro $\{y_1 = N_1, \ldots, y_n = N_n\}$, de manera tal que el registro resultante contenga todas las etiquetas de ambos, con los mismos valores y en el mismo orden.

Restricción: los registros a unir **no deben** tener etiquetas en común.

Ejercicio 21 *(Conectivos booleanos)*

Definir como macros (azúcar sintáctica) los términos **Not**, **And**, **Or**, **Xor**, que simulen desde la reducción los conectivos clásicos usuales, por ej. $And\ M\ N \twoheadrightarrow \mathsf{true} \Leftrightarrow M \twoheadrightarrow \mathsf{true}$ y $N \twoheadrightarrow \mathsf{true}$.

Notar que definir una macro no es lo mismo que hacer una extensión. Por ejemplo, definir el término $I_\sigma \stackrel{\text{def}}{=} \lambda x : \sigma.x$, que es la función identidad del tipo σ, es distinto de extender la sintaxis del lenguaje con términos de la forma $I(M)$, lo cual además requeriría agregar nuevas reglas de tipado y de evaluación.

Ejercicio 22 ★

Se desea extender el cálculo lambda tipado agregando *unión de funciones*. Para ello, extenderemos el conjunto de términos y el de tipos de la siguiente manera:

$$M_1 \ldots M_k ::= \cdots \mid [(M_1, \ldots, M_k)] \qquad \sigma ::= \cdots \mid \mathsf{Union}(\sigma_1, \ldots, \sigma_k)_\tau$$

Cada M_i dentro de "[()]" es una función con distinto dominio del resto pero con la misma imagen.

En el tipo $\mathsf{Union}(\sigma_1, \ldots, \sigma_k)_\tau$, cada σ_i representa el tipo del dominio de M_i (la función en la posición i), y τ el tipo de la imagen de todas las funciones.

Al aplicarse esta unión sobre un valor de tipo σ, el término reduce utilizando la función de esta unión cuyo tipo para el dominio sea σ. Es decir, aplicando la función que corresponda según el dominio.

Por ejemplo, sea

$$l \stackrel{def}{=} [(\lambda\, x : \mathsf{Nat}\,.\ x + 2,\ \lambda\, x : \mathsf{Bool}\,.\ \mathsf{if}\ x\ \mathsf{then}\ 4\ \mathsf{else}\ 3,\ \lambda\, f : \mathsf{Bool} \to \mathsf{Nat}\,.\,(f\ \mathsf{true})\ +\ 3)]$$

l tiene tipo $\mathsf{Union}(\mathsf{Nat}, \mathsf{Bool}, \mathsf{Bool} \to \mathsf{Nat})_{\mathsf{Nat}}$.

Luego, $l\ (\lambda\, b : Bool.\ \mathsf{if}\ b\ \mathsf{then}\ 3\ \mathsf{else}\ 4) \longrightarrow\!\!\!\!\rightarrow ((\lambda b : \mathsf{Bool}.\ \mathsf{if}\ b\ \mathsf{then}\ 3\ \mathsf{else}\ 4)\ \mathsf{true}) + 3) \longrightarrow\!\!\!\!\rightarrow 6$.

Se pide:

1. Extender las reglas de tipado acordemente.
2. Mostrar el árbol de derivación para el juicio: $\{y : \mathsf{Nat}\} \triangleright [(\lambda x : \mathsf{Bool}.\ y, \lambda x : \mathsf{Nat}.\ x)]\ y : \mathsf{Nat}$.
3. Indicar cómo se modifica el conjunto de valores. Justificar.
4. Modificar o extender las reglas de semántica operacional para la extensión propuesta.

Ejercicio 23

Definir las siguientes funciones en Cálculo Lambda con Listas (visto en el ejercicio 17). Pueden definirse como macros o como extensiones al cálculo.

Nota: en este ejercicio usamos la notación $M : \sigma$ para decir que la expresión M a definir debe tener tipo σ en cualquier contexto.

1. $head_\sigma : [\sigma] \to \sigma$ y $tail_\sigma : [\sigma] \to [\sigma]$ (asumir que $\bot_\sigma \stackrel{\text{def}}{=} fix\ \lambda x : \sigma.x$).

2. $iterate_\sigma$:$(\sigma \to \sigma) \to \sigma \to [\sigma]$ que dadas f y x genera la lista infinita $x :: f\ x :: f(f\ x) :: f(f(f\ x)) :: \ldots$ (y no termina).

3. $zip_{\rho,\sigma} : [\rho] \to [\sigma] \to [\rho \times \sigma]$ que se comporta como la función homónima de Haskell.

4. $take_\sigma : nat \to ([\sigma] \to [\sigma])$ que se comporta como la función homónima de Haskell.

Ejercicio 24 ★

Se desea extender el cálculo lambda tipado para tener un mayor control sobre el proceso de reducción. Para esto, se introducen expresiones capaces de detener la reducción de un término, o de continuar una reducción que estaba detenida.

El conjunto de tipos será: σ ::= ... | $\texttt{det}(\sigma)$ donde $\texttt{det}(\sigma)$ es el tipo de los términos que resultan de detener la reducción de términos de tipo σ.

El conjunto de términos será: M ::= ... | $\texttt{detener}(M)$ | $\texttt{continuar}(M)$

El comportamiento de estas expresiones es el siguiente: sea M un término tipable cualquiera, $\texttt{detener}(M)$ detiene la reducción de M. Es decir, no reduce por más que M pueda reducirse. Por otro lado, si N es un término detenido, $\texttt{continuar}(N)$ reanuda la reducción de N.

Por ejemplo, $\texttt{continuar}((\lambda x\colon \texttt{det}(\mathsf{Nat}).x)\ \texttt{detener}(\mathsf{pred}(\mathsf{succ}(0)))) \to \texttt{continuar}(\texttt{detener}(\mathsf{pred}(\mathsf{succ}(0)))) \to \mathsf{pred}(\mathsf{succ}(0)) \to 0$.

Además, las funciones que esperan argumentos detenidos, pueden recibir argumentos del tipo correspondiente sin detener. En ese caso, en lugar de reducir el argumento hasta obtener un valor, lo detienen. Esto permite definir funciones que toman parámetros por nombre (call-by-name). Por ejemplo:

$(\lambda x\colon \texttt{det}(\mathsf{Nat}).\mathsf{if}\ \mathsf{true}\ \mathsf{then}\ \texttt{continuar}(x)\ \mathsf{else}\ 0)\ \mathsf{succ}(\mathsf{pred}(\mathsf{succ}(0))) \to$
$(\lambda x\colon \texttt{det}(\mathsf{Nat}).\mathsf{if}\ \mathsf{true}\ \mathsf{then}\ \texttt{continuar}(x)\ \mathsf{else}\ 0)\ \texttt{detener}(\mathsf{succ}(\mathsf{pred}(\mathsf{succ}(0)))) \to$
$\mathsf{if}\ \mathsf{true}\ \mathsf{then}\ \texttt{continuar}(\texttt{detener}(\mathsf{succ}(\mathsf{pred}(\mathsf{succ}(0))))))\ \mathsf{else}\ 0 \to$
$\texttt{continuar}(\texttt{detener}(\mathsf{succ}(\mathsf{pred}(\mathsf{succ}(0))))) \to \mathsf{succ}(\mathsf{pred}(\mathsf{succ}(0))) \to \mathsf{succ}(0)$.

1. Introducir las reglas de tipado para la extensión propuesta.

2. Exhibir la derivación de tipado para el siguiente juicio:
$\{y\colon \mathsf{Bool}\} \triangleright (\lambda x\colon \texttt{det}(\mathsf{Bool}).\mathsf{if}\ y\ \mathsf{then}\ \texttt{continuar}(x)\ \mathsf{else}\ \mathsf{false})\ \mathsf{isZero}(0)\ :\ \mathsf{Bool}$.

3. Indicar formalmente cómo se modifica el conjunto de valores, y dar la semántica operacional de a un paso para la extensión propuesta. Notar que puede ser necesario modificar alguna de las reglas preexistentes.