Práctica N^{o} 3 - Inferencia de Tipos

Aclaraciones:

- Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.
- Usaremos las expresiones de tipos y términos vistas en clase, con los tipos Bool, Nat y funciones ya definidos.
- Para esta práctica será necesario utilizar los axiomas y reglas de tipado e inferencia vistos en clase (tanto en las teóricas como en las prácticas).
- Siempre que se pide definir extensiones, se asume que el algoritmo de unificación (`MGU`) y el de borrado (`Erase`) ya se encuentran correctamente extendidos, de manera que sólo es necesario extender el algoritmo $\mathbb{W}$ (también conocido como `Principal Typing`).

Gramáticas a tener en cuenta:

- Términos **anotados**
  $M ::= x \mid \lambda x\colon \sigma.M \mid M\ M \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M \mid 0 \mid \mathsf{succ}(M) \mid \mathsf{pred}(M) \mid \mathsf{isZero}(M)$ Donde la letra x representa un *nombre de variable* arbitrario. Tales nombres se toman de un conjunto infinito dado $\mathfrak{X} = \{w, w_1, w_2, \dots, x, x_1, x_2, \dots, y, y_1, y_2, \dots, f, f_1, f_2, \dots\}$
- Términos **sin anotaciones**
  $M' ::= x \mid \lambda x.M' \mid M'\ M' \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ M'\ \mathsf{then}\ M'\ \mathsf{else}\ M' \mid 0 \mid \mathsf{succ}(M') \mid \mathsf{pred}(M') \mid \mathsf{isZero}(M')$
- Tipos
  $\sigma ::= \mathsf{Bool} \mid \mathsf{Nat} \mid \sigma \to \sigma \mid \mathbf{s}$
  Donde la letra $\mathbf{s}$ representa una *variable de tipos* arbitraria. Tales nombres se toman de un conjunto infinito dado $\mathfrak{T} = \{\mathbf{s}, \mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{t}, \mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{a}, \mathbf{b}, \mathbf{c}, \dots\}$

Ejercicio 1

Determinar el resultado de aplicar la sustitución `S` a las siguientes expresiones

I. $S = \{\mathbf{t} \leftarrow \mathsf{Nat}\}$ $\quad S(\{x : \mathbf{t} \to \mathsf{Bool}\})$

II. $S = \{\mathbf{t}_1 \leftarrow \mathbf{t}_2 \to \mathbf{t}_3,\ \mathbf{t} \leftarrow \mathsf{Bool}\}$ $\quad S(\{x : \mathbf{t} \to \mathsf{Bool}\}) \rhd S(\lambda x\colon \mathbf{t}_1 \to \mathsf{Bool}.x) \colon S(\mathsf{Nat} \to \mathbf{t}_2)$

Ejercicio 2 ★

Determinar el resultado de aplicar el `MGU` ("most general unifier") sobre las ecuaciones planteadas a continuación. En caso de tener éxito, mostrar la sustitución resultante.

I. `MGU` $\{\mathbf{t}_1 \to \mathbf{t}_2 \doteq \mathsf{Nat} \to \mathsf{Bool}\}$

II. `MGU` $\{\mathbf{t}_1 \to \mathbf{t}_2 \doteq \mathbf{t}_3\}$

III. `MGU` $\{\mathbf{t}_1 \to \mathbf{t}_2 \doteq \mathbf{t}_2\}$

IV. `MGU` $\{(\mathbf{t}_2 \to \mathbf{t}_1) \to \mathsf{Bool} \doteq \mathbf{t}_2 \to \mathbf{t}_3\}$

V. `MGU` $\{\mathbf{t}_2 \to \mathbf{t}_1 \to \mathsf{Bool} \doteq \mathbf{t}_2 \to \mathbf{t}_3\}$

VI. `MGU` $\{\mathbf{t}_1 \to \mathsf{Bool} \doteq \mathsf{Nat} \to \mathsf{Bool}, \mathbf{t}_1 \doteq \mathbf{t}_2 \to \mathbf{t}_3\}$

VII. `MGU` $\{\mathbf{t}_1 \to \mathsf{Bool} \doteq \mathsf{Nat} \to \mathsf{Bool}, \mathbf{t}_2 \doteq \mathbf{t}_1 \to \mathbf{t}_1\}$

VIII. `MGU` $\{\mathbf{t}_1 \to \mathbf{t}_2 \doteq \mathbf{t}_3 \to \mathbf{t}_4, \mathbf{t}_3 \doteq \mathbf{t}_2 \to \mathbf{t}_1\}$

Ejercicio 3

Unir con flechas los tipos que unifican entre sí (entre una fila y la otra). Para cada par unificable, exhibir el *mgu* ("most general unifier").

$\mathbf{t} \to \mathbf{u}$	Nat	$\mathbf{u} \to \mathsf{Bool}$	$\mathbf{a} \to \mathbf{b} \to \mathbf{c}$
$\mathbf{t}$	$\mathsf{Nat} \to \mathsf{Bool}$	$(\mathsf{Nat} \to \mathbf{u}) \to \mathsf{Bool}$	$\mathsf{Nat} \to \mathbf{u} \to \mathsf{Bool}$

Ejercicio 4

I. Mostrar, si es posible, dos términos $M_1 \neq M_2 \in \lambda^{bn}$ tal que:
   - $\emptyset \vdash M_1 : \sigma$, $\emptyset \vdash M_2 : \sigma'$, y $\mathbb{W}(Erase(M_1)) = \mathbb{W}(Erase(M_2))$.
   - $\emptyset \vdash M_1 : \sigma$, $\emptyset \vdash M_2 : \sigma$, y $\mathbb{W}(Erase(M_1)) = \mathbb{W}(Erase(M_2))$.

II. Mostrar, si es posible, un término $M \in \lambda^{bn}$ tal que $\emptyset \vdash M : \sigma$ y $\mathbb{W}(Erase(M)) = \Gamma \vdash M' : \rho$ y
   a) $\sigma \neq \rho$;
   b) $\sigma = \rho$ y $M \neq M'$.

Ejercicio 5

Decidir, utilizando el método del árbol, cuáles de las siguientes expresiones son tipables. Mostrar qué reglas y sustituciones se aplican en cada paso y justificar por qué no son tipables aquéllas que fallan.

I. $\lambda z.$ if z then 0 else $\mathsf{succ}(0)$
II. $\lambda y.$ succ$((\lambda x.x)\ y)$
III. $\lambda x.$ if isZero(x) then x else (if x then x else x)
IV. $\lambda x.\lambda y.$ if x then y else succ(0)
V. if True then $(\lambda x.\ 0)\ 0$ else $(\lambda x.\ 0)$ False
VI. $(\lambda f.$ if True then $f\ 0$ else f $\mathsf{False})$ $(\lambda x.\ 0)$
VII. $\lambda x.\lambda y.\lambda z.$ if z then y else $\mathsf{succ}(x)$
VIII. fix $(\lambda x.$ pred$(x))$

Para el punto VIII, asumir extentido el algoritmo de inferencia con $\mathbb{W}(\text{fix}) = \emptyset \triangleright \text{fix}_{\mathbf{a}} : (\mathbf{a} \to \mathbf{a}) \to \mathbf{a}$ donde $\mathbf{a}$ es una variable fresca.

Ejercicio 6 ★

Utilizando el árbol de inferencia, inferir el tipo de las siguientes expresiones o demostrar que no son tipables. En cada paso donde se realice una unificación, mostrar el conjunto de ecuaciones a unificar y la sustitución obtenida como resultado de la misma.

- $\lambda x.\ \lambda y.\ \lambda z.\ (z\ x)\ y\ z$
- $\lambda x.\ x\ (w\ (\lambda y.w\ y))$
- $\lambda x.\lambda y.\ xy$
- $\lambda x.\lambda y.\ yx$
- $\lambda x.(\lambda x.\ x)$
- $\lambda x.(\lambda y.\ y)x$
- $(\lambda z.\lambda x.\ x\ (z\ (\lambda y.\ z)))$ True

Ejercicio 7

I. Utilizar el algoritmo de inferencia sobre la siguiente expresión: $\lambda y.(x\ y)\ (\lambda z.x_2)$

II. Una vez calculado, demostrar (utilizando chequeo de tipos) que el juicio encontrado es correcto.

III. ¿Qué ocurriría si x_2 fuera x?

Ejercicio 8 ★

Tener en cuenta un nuevo tipo par definido como: $\sigma ::= \ldots \mid \sigma \times \sigma$

Con expresiones nuevas definidas como: $M ::= \ldots \mid \langle M,\ M\rangle \mid \pi_1(M) \mid \pi_2(M)$

Y las siguientes reglas de tipado:

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : \tau}{\Gamma \triangleright \langle M, N\rangle : \sigma \times \tau} \qquad \frac{\Gamma \triangleright M : \sigma \times \tau}{\Gamma \triangleright \pi_1(M) : \sigma} \qquad \frac{\Gamma \triangleright M : \sigma \times \tau}{\Gamma \triangleright \pi_2(M) : \tau}$$

I. Adaptar el algoritmo de inferencia para que funcione sobre esta versión extendida.

II. Tipar la expresión $(\lambda \mathrm{f}.\langle \mathrm{f}, \underline{2}\rangle)\ (\lambda \mathrm{x}.\mathrm{x}\ \underline{1})$ utilizando la versión extendida del algoritmo.

III. Intentar tipar la siguiente expresión utilizando la versión extendida del algoritmo.
$(\lambda f.\langle f\ \underline{2},\ f\ \mathsf{True}\rangle)\ (\lambda x.x)$
Mostrar en qué punto del mismo falla y por qué motivo.

Ejercicio 9

a) Extender el sistema de tipado y el algoritmo de inferencia con las reglas necesarias para introducir los tipos `Either` $\sigma\ \sigma$ y
`Maybe` σ, cuyos términos son análogos a los de Haskell.

b) Utilizando estas reglas y el método del árbol, tipar la expresión:
λx`.if` x `then Just (Left 0) else Nothing`

Ejercicio 10 ★

a) Extender el algoritmo de inferencia para soportar la inferencia de tipos de árboles binarios. En esta extensión del algoritmo sólo se considerarán los *constructores* del árbol.

La sintaxis de esta extensión es la siguiente:

$\sigma ::= \ldots \mid AB_\sigma \qquad M ::= \ldots \mid Nil_\sigma \mid Bin(M, N, O)$

Y sus reglas de tipado, las siguientes:

$$\frac{}{\Gamma \triangleright Nil_\sigma : AB_\sigma} \qquad \frac{\Gamma \triangleright M : AB_\sigma \quad \Gamma \triangleright O : AB_\sigma \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright Bin(M, N, O) : AB_\sigma}$$

Nota: la función *Erase*, que elimina la información de tipos que el inferidor se encargará de inferir, se extiende de manera acorde para la sintaxis nueva:

$$\begin{aligned} Erase(Nil_\sigma) &= Nil \\ Erase(Bin(M, N, O)) &= Bin(Erase(M), Erase(N), Erase(O)) \end{aligned}$$

Recordar que una entrada válida para el algoritmo es un pseudo término con la información de tipos eliminada. Por ejemplo:

$$(\lambda x.Bin(Nil, 5, Bin(Nil, x, Nil)))\ 5$$

b) Escribir la regla de tipado para el case de árboles binarios, y la regla análoga en el algoritmo de inferencia.

Ejercicio 11 ★

Extender el algoritmo de inferencia $\mathbb{W}$ para que soporte el tipado del *switch* de números naturales, similar al de C o C++. La extensión de la sintaxis es la siguiente:

$$M = \ldots \mid \texttt{switch}\ M\ \{\texttt{case}\ \underline{n_1} :\ M_1\ \ \ldots\ \texttt{case}\ \underline{n_k} :\ M_k\ \texttt{default} :\ M_{k+1}\}$$

donde cada $\underline{n_i}$ es un numeral (un *valor* de tipo Nat, como 0, $\mathsf{succ}(0)$, $\mathsf{succ}(\mathsf{succ}(0))$, etc.). Esto forma parte de la sintaxis y no hace falta verificarlo en el algoritmo.
La regla de tipado es la siguiente:

$$\frac{\Gamma \triangleright M : \mathsf{Nat} \quad \forall i, j (1 \leq i, j \leq k \wedge i \neq j \Rightarrow n_i \neq n_j) \quad \Gamma \triangleright N_1 : \sigma \ldots \Gamma \triangleright N_k : \sigma \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright \texttt{switch}\ M\ \{\texttt{case}\ \underline{n_1} :\ N_1 \ldots \texttt{case}\ \underline{n_k} :\ N_k\ \texttt{default} : N\} : \sigma}$$

Por ejemplo, una expresión como:

$$\lambda x.\ \texttt{switch}\ (x)\ \{\texttt{case}\ 0 :\ \mathsf{True}\ \texttt{default} :\ \mathsf{False}\}$$

debería tipar a $\mathsf{Nat} \to \mathsf{Bool}$. En cambio, la expresión:

$$\texttt{switch}\ \underline{3}\ \{\texttt{case}\ \underline{1} :\ \underline{1}\ \texttt{case}\ \underline{2} :\ 0\ \texttt{default} :\ \mathsf{False}\}$$

no tiene tipo, pues entre los casos hay números y booleanos. Y finalmente, la expresión:

$$\texttt{switch}\ \underline{3}\ \{\texttt{case}\ \underline{1} :\ \underline{1}\ \texttt{case}\ \underline{2} :\ \underline{2}\ \texttt{case}\ \underline{1} :\ \underline{3}\ \texttt{default} :\ 0\}$$

tampoco tiene tipo, ya que el número 1 se repite entre los casos.

Ejercicio 12

En este ejercicio extenderemos el algoritmo de inferencia para soportar operadores binarios. Dichos operadores se comportan de manera similar a las funciones, excepto que siempre tienen 2 parámetros y su aplicación se nota de manera infija. Para esto extenderemos la sintaxis y el sistema de tipos del cálculo lambda tipado de la siguiente manera:

$$M\ ::=\ \ldots\ \mid\ \varphi x : \sigma\ y : \tau.M\ \mid\ \langle M\ N\ O \rangle \qquad \sigma\ ::=\ \ldots\ \mid\ \mathrm{Op}(\sigma, \tau \to \upsilon)$$

Aquí φ es el constructor de operadores que liga las variables x (parámetro anterior al operador) e y (parámetro posterior) y $\langle M\ N\ O \rangle$ es la aplicación del operador N a los parámetros M y O (lo ponemos entre $\langle$ y $\rangle$ para evitar problemas de ambigüedad con la aplicación estándar). $\mathrm{Op}(\sigma, \tau \to \upsilon)$, por otro lado, representa el tipo de los operadores cuyo parámetro anterior es de tipo σ, el posterior de tipo τ y dan como resultado un tipo υ.

Las reglas de tipado que se incorporan son las siguientes:

$$\frac{\Gamma \cup \{x : \sigma,\ y : \tau\} \triangleright M : \upsilon}{\Gamma \triangleright \varphi x : \sigma\ y : \tau.M : \mathrm{Op}(\sigma, \tau \to \upsilon)}$$

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : \mathrm{Op}(\sigma, \tau \to \upsilon) \quad \Gamma \triangleright O : \tau}{\Gamma \triangleright \langle M\ N\ O \rangle : \upsilon}$$

I. Dar la extensión al algoritmo necesaria para soportar el tipado de las nuevas expresiones. Recordar que el parámetro de entrada es un término **sin anotaciones de tipos**.

II. Aplicar el algoritmo extendido con el método del árbol para tipar: $\langle(\lambda x.\mathsf{succ}(x))\ (\varphi xy.xy)\ 0\rangle$

Ejercicio 13

Considerar el algoritmo de inferencia extendido para soportar listas:

$\mathbb{W}([\ ]) \stackrel{def}{=} \emptyset \triangleright [\ ]_{\mathbf{t}} : [\mathbf{t}]$, con $\mathbf{t}$ variable fresca.

$\mathbb{W}(M : N) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \triangleright S(U : V) : [S\sigma]$, con:

$$\mathbb{W}(M) = \Gamma_1 \triangleright U : \sigma$$

$$\mathbb{W}(N) = \Gamma_2 \triangleright V : \tau$$

$$S = \mathtt{MGU}(\{\tau \doteq [\sigma]\} \cup \{\alpha \doteq \beta / x : \alpha \in \Gamma_1 \wedge x : \beta \in \Gamma_2\})$$

I. Extender el algoritmo de inferencia para soportar expresiones de la forma "$\exists x$ in M/N".

$$\frac{\Gamma \cup \{x : \sigma\} \triangleright N : \mathsf{Bool} \qquad \Gamma \triangleright M : [\sigma]}{\Gamma \triangleright \exists x \text{ in } M/N : \mathsf{Bool}}$$

II. Aplicar el algoritmo extendido con el método del árbol para tipar las siguientes expresiones. Si alguna de ellas no tipa, indicar el motivo.

I) $(\lambda x.\exists y \text{ in } x/y)(0 : [\ ])$
II) $(\lambda x.\exists y \text{ in } x/y)(\text{iszero}(z) : [\ ])$
III) $\exists x \text{ in } [\ ]/\mathsf{True}$
IV) $\exists x \text{ in } [\ ]/(\lambda y.\mathsf{True})$
V) $\exists x \text{ in } (0 : [\ ])/\text{iszero}(x)$

Ejercicio 14

Se desea diseñar un algoritmo de inferencia de tipos para el cálculo λ extendido con fórmulas proposicionales de la siguiente manera:

$$M ::= \cdots \mid \neg M \mid M \supset M \mid \mathsf{esTautolog\acute{i}a}(M)$$

$$\sigma := \cdots \mid \mathsf{Prop}$$

Las reglas de tipado son:

$$\frac{\Gamma \triangleright M : \mathsf{Prop}}{\Gamma \triangleright \neg M : \mathsf{Prop}}\textsc{tNeg} \qquad \frac{\Gamma \triangleright M : \mathsf{Prop} \qquad \Gamma \triangleright N : \mathsf{Prop}}{\Gamma \triangleright M \supset N : \mathsf{Prop}}\textsc{tImp}$$

$$\frac{\Gamma, x_1 : \mathsf{Prop}, \ldots, x_n : \mathsf{Prop} \triangleright M : \mathsf{Prop} \qquad \mathsf{fv}(M) = \{x_1, \ldots, x_n\}}{\Gamma \triangleright \mathsf{esTautolog\acute{i}a}(M) : \mathsf{Bool}}\textsc{tTaut}$$

Notar que $\mathsf{esTautolog\acute{i}a}(M)$ liga **todas** las variables libres de M. Por ejemplo, $\mathsf{esTautolog\acute{i}a}(p \supset (q \supset p))$ es un término cerrado y bien tipado (de tipo Bool).

I. Extender el algoritmo de inferencia para admitir las expresiones incorporadas al lenguaje, de tal manera que implemente las reglas de tipado tNeg, tImp y tTaut.

II. Aplicar el algoritmo extendido con el método del árbol para tipar las siguientes expresiones (exhibiendo siempre las sustituciones utilizadas). Si alguna de ellas no tipa, indicar el motivo.

- $\lambda y.\neg((\lambda x.\neg x)(y \supset y))$
- $(\lambda x.\mathsf{esTautología}(\mathsf{if}\ x\ \mathsf{then}\ y\ \mathsf{else}\ z))\mathsf{True}$

Ejercicio 15 ★

En este ejercicio modificaremos el algoritmo de inferencia para incorporar la posibilidad de utilizar letrec en nuestro cálculo.

- $M ::= \ldots \mid \mathsf{letrec}\ f = M\ \mathsf{in}\ N$
- letrec permite por ejemplo representar el factorial de 10 de la siguiente manera:

$$\mathsf{letrec}\ f = \ (\lambda\, x : \mathsf{Nat}\,.\,\mathsf{if}\ \mathsf{isZero}(x)\ \mathsf{then}\ 1\ \mathsf{else}\ x \times f\ (\mathsf{pred}(x)))\ \mathsf{in}\ f\ 10$$

Para ello se agrega la siguiente regla de tipado:

$$\frac{\Gamma \cup \{f : \pi \to \tau\} \triangleright M : \pi \to \tau \qquad \Gamma \cup \{f : \pi \to \tau\} \triangleright N : \sigma}{\Gamma \triangleright \mathsf{letrec}\ f = M\ \mathsf{in}\ N : \sigma}$$

Suponiendo que se propone el siguiente pseudocódigo:

$\mathbb{W}(\mathsf{letrec}\ f = M\ \mathsf{in}\ N) \stackrel{def}{=} \Gamma \triangleright S\,(\mathsf{letrec}\ f = M'\ \mathsf{in}\ N') : S\,\sigma$
donde

- $\mathbb{W}(M) = \Gamma_1 \triangleright M' : \pi \to \tau$
- $\mathbb{W}(N) = \Gamma_2 \triangleright N' : \sigma$
- $\tau_1 = \rho / f : \rho \in \Gamma_1$
- $\tau_2 = \delta / f : \delta \in \Gamma_2$
- S = $\mathtt{MGU}\ \{\tau_1 \doteq \tau_2$, COMPLETAR$\}$
- $\Gamma = S\,\Gamma_1 \cup S\,\Gamma_2$

I. Explicar cuál es el error en los llamados recursivos. Dar un ejemplo que debería tipar y no lo hace debido a este error.

II. Explicar cuál es el error en el pseudocódigo con respecto la definición de τ_1 y τ_2. Dar un ejemplo que debería tipar y no lo hace debido a este error.

III. El contexto Γ ¿puede contener a f? ¿Es un comportamiento deseable? Mostrar un ejemplo donde esto trae conflictos (ayuda: usar letrec dentro de un término más grande).

IV. Reescribir el pseudocódigo para que funcione correctamente (corregir los errores y completar la definición de S).