

Práctica N° 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**
 $M ::= x \mid \lambda x: \sigma. M \mid M \ M \mid \text{True} \mid \text{False} \mid \text{if } M \text{ then } M \text{ else } M \mid 0 \mid \text{succ}(M) \mid \text{pred}(M) \mid \text{isZero}(M)$ 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**
 $M' ::= x \mid \lambda x. M' \mid M' \ M' \mid \text{True} \mid \text{False} \mid \text{if } M' \text{ then } M' \text{ else } M' \mid 0 \mid \text{succ}(M') \mid \text{pred}(M') \mid \text{isZero}(M')$
- Tipos
 $\sigma ::= \text{Bool} \mid \text{Nat} \mid \sigma \rightarrow \sigma \mid \mathbf{s}$
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. $S = \{\mathbf{t} \leftarrow \text{Nat}\} \quad S(\{x: \mathbf{t} \rightarrow \text{Bool}\})$
- II. $S = \{\mathbf{t}_1 \leftarrow \mathbf{t}_2 \rightarrow \mathbf{t}_3, \mathbf{t} \leftarrow \text{Bool}\} \quad S(\{x: \mathbf{t} \rightarrow \text{Bool}\}) \triangleright S(\lambda x: \mathbf{t}_1 \rightarrow \text{Bool}. x): S(\text{Nat} \rightarrow \mathbf{t}_2)$

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 $\{\mathbf{t}_1 \rightarrow \mathbf{t}_2 \doteq \text{Nat} \rightarrow \text{Bool}\}$
- II. MGU $\{\mathbf{t}_1 \rightarrow \mathbf{t}_2 \doteq \mathbf{t}_3\}$
- III. MGU $\{\mathbf{t}_1 \rightarrow \mathbf{t}_2 \doteq \mathbf{t}_2\}$
- IV. MGU $\{(\mathbf{t}_2 \rightarrow \mathbf{t}_1) \rightarrow \text{Bool} \doteq \mathbf{t}_2 \rightarrow \mathbf{t}_3\}$
- V. MGU $\{\mathbf{t}_2 \rightarrow \mathbf{t}_1 \rightarrow \text{Bool} \doteq \mathbf{t}_2 \rightarrow \mathbf{t}_3\}$
- VI. MGU $\{\mathbf{t}_1 \rightarrow \text{Bool} \doteq \text{Nat} \rightarrow \text{Bool}, \mathbf{t}_1 \doteq \mathbf{t}_2 \rightarrow \mathbf{t}_3\}$
- VII. MGU $\{\mathbf{t}_1 \rightarrow \text{Bool} \doteq \text{Nat} \rightarrow \text{Bool}, \mathbf{t}_2 \doteq \mathbf{t}_1 \rightarrow \mathbf{t}_1\}$
- VIII. MGU $\{\mathbf{t}_1 \rightarrow \mathbf{t}_2 \doteq \mathbf{t}_3 \rightarrow \mathbf{t}_4, \mathbf{t}_3 \doteq \mathbf{t}_2 \rightarrow \mathbf{t}_1\}$

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”).

$$\begin{array}{cccc} \mathbf{t} \rightarrow \mathbf{u} & \text{Nat} & \mathbf{u} \rightarrow \text{Bool} & \mathbf{a} \rightarrow \mathbf{b} \rightarrow \mathbf{c} \\ \mathbf{t} & \text{Nat} \rightarrow \text{Bool} & (\text{Nat} \rightarrow \mathbf{u}) \rightarrow \text{Bool} & \text{Nat} \rightarrow \mathbf{u} \rightarrow \text{Bool} \end{array}$$

Ejercicio 4

- I. Mostrar, si es posible, dos términos $M_1 \neq M_2 \in \lambda^{bn}$ tal que:
 - $\emptyset \vdash M_1 : \sigma, \emptyset \vdash M_2 : \sigma', \text{ y } \mathbb{W}(\text{Erase}(M_1)) = \mathbb{W}(\text{Erase}(M_2))$.
 - $\emptyset \vdash M_1 : \sigma, \emptyset \vdash M_2 : \sigma, \text{ y } \mathbb{W}(\text{Erase}(M_1)) = \mathbb{W}(\text{Erase}(M_2))$.
- II. Mostrar, si es posible, un término $M \in \lambda^{bn}$ tal que $\emptyset \vdash M : \sigma$ y $\mathbb{W}(\text{Erase}(M)) = \Gamma \vdash M' : \rho$ 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. $\lambda z. \text{if } z \text{ then } 0 \text{ else succ}(0)$
- II. $\lambda y. \text{succ}((\lambda x.x) y)$
- III. $\lambda x. \text{if isZero}(x) \text{ then } x \text{ else } (\text{if } x \text{ then } x \text{ else } x)$
- IV. $\lambda x. \lambda y. \text{if } x \text{ then } y \text{ else succ}(0)$
- V. $\text{if True then } (\lambda x. 0) 0 \text{ else } (\lambda x. 0) \text{ False}$
- VI. $(\lambda f. \text{if True then } f 0 \text{ else } f \text{ False}) (\lambda x. 0)$
- VII. $\lambda x. \lambda y. \lambda z. \text{if } z \text{ then } y \text{ else succ}(x)$
- VIII. $\text{fix } (\lambda x. \text{pred}(x))$

Para el punto [VIII](#), asumir extendido el algoritmo de inferencia con $\mathbb{W}(\text{fix}) = \emptyset \triangleright \text{fix}_{\mathbf{a}} : (\mathbf{a} \rightarrow \mathbf{a}) \rightarrow \mathbf{a}$ 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.

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

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: $\sigma ::= \dots \mid \sigma \times \sigma$

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

Y las siguientes reglas de tipado:

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

- I. Adaptar el algoritmo de inferencia para que funcione sobre esta versión extendida.
- II. Tipar la expresión $(\lambda f.\langle f, 2 \rangle) \ (\lambda x.x \ 1)$ utilizando la versión extendida del algoritmo.
- III. Intentar tipar la siguiente expresión utilizando la versión extendida del algoritmo.
 $(\lambda f.\langle f \ 2, f \ \text{True} \rangle) \ (\lambda x.x)$
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:
 $\lambda x.\text{if } x \text{ then Just (Left 0) else Nothing}$

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:

$$\sigma ::= \dots \mid AB_\sigma \quad M ::= \dots \mid Nil_\sigma \mid Bin(M, N, O)$$

Y sus reglas de tipado, las siguientes:

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

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:

$$(\lambda x. \text{Bin}(\text{Nil}, 5, \text{Bin}(\text{Nil}, x, \text{Nil}))) 5$$

- 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:

$$M = \dots | \text{switch } M \{ \text{case } \underline{n}_1 : M_1 \dots \text{case } \underline{n}_k : M_k \text{ default} : M_{k+1} \}$$

donde cada \underline{n}_i es un numeral (un *valor* de tipo **Nat**, como 0, $\text{succ}(0)$, $\text{succ}(\text{succ}(0))$, etc.). Esto forma parte de la sintaxis y no hace falta verificarlo en el algoritmo.

La regla de tipado es la siguiente:

$$\frac{\Gamma \triangleright M : \text{Nat} \quad \forall i, j (1 \leq i, j \leq k \wedge i \neq j \Rightarrow n_i \neq n_j) \quad \Gamma \triangleright N_1 : \sigma \dots \Gamma \triangleright N_k : \sigma \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright \text{switch } M \{ \text{case } \underline{n}_1 : N_1 \dots \text{case } \underline{n}_k : N_k \text{ default} : N \} : \sigma}$$

Por ejemplo, una expresión como:

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

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

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

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

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

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:

$$M ::= \dots | \varphi x : \sigma y : \tau. M \quad | \langle M \ N \ O \rangle \quad \sigma ::= \dots | \text{Op}(\sigma, \tau \rightarrow v)$$

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 $\langle \ \rangle$ para evitar problemas de ambigüedad con la aplicación estándar). $\text{Op}(\sigma, \tau \rightarrow v)$, 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 v .

Las reglas de tipado que se incorporan son las siguientes:

$$\frac{\Gamma \cup \{x : \sigma, y : \tau\} \triangleright M : v}{\Gamma \triangleright \varphi x : \sigma y : \tau. M : \text{Op}(\sigma, \tau \rightarrow v)}$$

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : \text{Op}(\sigma, \tau \rightarrow v) \quad \Gamma \triangleright O : \tau}{\Gamma \triangleright \langle M \ N \ O \rangle : v}$$

1. 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: $\langle (\lambda x.\text{succ}(x)) (\varphi xy.xy) 0 \rangle$

Ejercicio 13

Considerar el algoritmo de inferencia extendido para soportar listas:

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

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

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

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

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

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

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

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 } [\]/\text{True}$

IV) $\exists x \text{ in } [\]/(\lambda y.\text{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:

$$M ::= \dots \mid \neg M \mid M \supset M \mid \text{esTautología}(M)$$

$$\sigma ::= \dots \mid \text{Prop}$$

Las reglas de tipado son:

$$\frac{\Gamma \triangleright M : \text{Prop}}{\Gamma \triangleright \neg M : \text{Prop}} \text{TNEG} \quad \frac{\Gamma \triangleright M : \text{Prop} \quad \Gamma \triangleright N : \text{Prop}}{\Gamma \triangleright M \supset N : \text{Prop}} \text{TIMP}$$

$$\frac{\Gamma, x_1 : \text{Prop}, \dots, x_n : \text{Prop} \triangleright M : \text{Prop} \quad \text{fv}(M) = \{x_1, \dots, x_n\}}{\Gamma \triangleright \text{esTautología}(M) : \text{Bool}} \text{TTAUT}$$

Notar que $\text{esTautología}(M)$ liga **todas** las variables libres de M . Por ejemplo, $\text{esTautología}(p \supset (q \supset p))$ 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.

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

Ejercicio 15 ★

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

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

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

Para ello se agrega la siguiente regla de tipado:

$$\frac{\Gamma \cup \{f : \pi \rightarrow \tau\} \triangleright M : \pi \rightarrow \tau \quad \Gamma \cup \{f : \pi \rightarrow \tau\} \triangleright N : \sigma}{\Gamma \triangleright \text{letrec } f = M \text{ in } N : \sigma}$$

Suponiendo que se propone el siguiente pseudocódigo:

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

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

- 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).