

Inferencia de Tipos

PLP

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

12 de septiembre de 2017

- 1 Introducción
- 2 Algoritmo de inferencia
- 3 Extensiones

Motivación

Ejemplo

Dada la expresión $\lambda x: \text{Nat}. \text{isZero}(x)$, ¿qué tipo tiene?

$$\emptyset \triangleright \lambda x: \text{Nat}. \text{isZero}(x) : \text{Nat} \rightarrow \text{Bool}$$

¿Cómo hicimos? ¿Se puede automatizar? ¿Podríamos no escribir el $: \text{Nat}$?

¿Y para $\lambda x. \lambda y. (x \ y) \ (\lambda z. x)$?

Inferencia

Dada una expresión, ¿tiene tipo? ¿Cuál es este tipo? ¿Es el más general? ¿Qué necesitamos saber del contexto?

Más ejemplos *a ojo*

¿Qué tipo tiene? ¿En qué contexto? ¿En qué contexto?

¿Es lo más general posible?

- $\emptyset \triangleright \lambda x:\text{Nat}. \text{succ}(x) : \text{Nat} \rightarrow \text{Nat}$
- $\{y:\text{Nat}\} \triangleright \lambda x:t. \text{succ}(y) : t \rightarrow \text{Nat}$
- $(\lambda x.\text{isZero}(x)) \text{ true}$ no tiene tipo.
- $\emptyset \triangleright \lambda x:\text{Nat}. x : \text{Nat} \rightarrow \text{Nat}$ no es lo más general.
- $\emptyset \triangleright \lambda x:t. x : t \rightarrow t$ es lo más general.

Generalidad

Dijimos que queremos el juicio *más general*. ¿Qué significa ser el más general?

Todos los juicios derivables para $\lambda x.x$ son instancias de

$\emptyset \triangleright \lambda x : t . x : t \rightarrow t$. Por ejemplo:

- $\emptyset \triangleright \lambda x : \text{Nat} . x : \text{Nat} \rightarrow \text{Nat}$
- $\emptyset \triangleright \lambda x : \text{Bool} . x : \text{Bool} \rightarrow \text{Bool}$
- $\{y : \text{Bool}\} \triangleright \lambda x : r \rightarrow \text{Nat} . x : (r \rightarrow \text{Nat}) \rightarrow (r \rightarrow \text{Nat})$
- ...

Recordemos algunas reglas de tipado

$$\frac{x : \sigma \in \Gamma}{\Gamma \triangleright x : \sigma} \text{ (T-VAR)} \quad \frac{\Gamma \cup \{x : \sigma\} \triangleright M : \tau}{\Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau} \text{ (T-ABS)}$$
$$\frac{\Gamma \triangleright M : \sigma \rightarrow \tau \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M N : \tau} \text{ (T-APP)}$$

Tipado vs. Inferencia

$$\frac{x : \sigma \in \Gamma}{\Gamma \triangleright x : \sigma} \text{ (T-VAR)}$$

$$\mathbb{W}(x) \stackrel{\text{def}}{=} \{x : t\} \triangleright x : t, \quad t \text{ variable fresca}$$

Tipado vs. Inferencia

$$\frac{\Gamma \cup \{x : \sigma\} \triangleright M : \tau}{\Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau} \text{ (T-Abs)}$$

Otra forma de escribirlo:

- Sea $\mathbb{W}(U) = \Gamma \triangleright M : \rho$
- Si el contexto tiene información de tipos para x (i.e. $x : \tau \in \Gamma$ para algún τ), entonces

$$\mathbb{W}(\lambda x. U) \stackrel{\text{def}}{=} \Gamma \setminus \{x : \tau\} \triangleright \lambda x : \tau. M : \tau \rightarrow \rho$$

- Si el contexto no tiene información de tipos para x (i.e. $x \notin \text{Dom}(\Gamma)$) elegimos una variable fresca t y entonces

$$\mathbb{W}(\lambda x. U) \stackrel{\text{def}}{=} \Gamma \triangleright \lambda x : t. M : t \rightarrow \rho$$

$$\tau = \begin{cases} \alpha & \text{si } x : \alpha \in \Gamma \\ \text{variable fresca en otro caso.} \end{cases}$$

$$\Gamma' = \Gamma \ominus \{x\}$$

$$\mathbb{W}(\lambda x. U) \stackrel{\text{def}}{=} \Gamma' \triangleright \lambda x : \tau. M : \tau \rightarrow \rho$$

Tipado vs. Inferencia

$$\frac{\Gamma \triangleright M : \sigma \rightarrow \tau \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M N : \tau} \text{ (T-APP)}$$

- Sea

- $\mathbb{W}(U) = \Gamma_1 \triangleright M : \tau$
- $\mathbb{W}(V) = \Gamma_2 \triangleright N : \rho$

- Sea

$$S = \text{MGU}\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_1 \wedge x : \sigma_2 \in \Gamma_2\} \cup \{\tau \doteq \rho \rightarrow t\} \text{ con } t \text{ una variable fresca}$$

- Entonces

$$\mathbb{W}(UV) \stackrel{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \triangleright S(MN) : St$$

Apliquémoslo

Utilizar el algoritmo \mathbb{W} para las siguientes expresiones:

- $\lambda f . \lambda x . f(f\ x)$
- $x(\lambda x . x)$
- $\lambda x . x\ y\ x$

Extensiones al algoritmo

En general

- Agregar casos nuevos al algoritmo.
- Menos frecuentemente, modificar casos existentes.

Para incorporar nuevos términos

- Nuevas reglas de tipado \Rightarrow nuevos casos del algoritmo \mathbb{W} .
- Anotar las expresiones con sus tipos.

Extensión del lenguaje

Abstracciones sobre pares

$$M ::= \dots | \lambda \langle x, y \rangle : \langle \sigma \times \tau \rangle . M$$

$$M' ::= \dots | \lambda \langle x, y \rangle . M'$$

$$\frac{\Gamma, x: \sigma, y: \tau \triangleright M: \rho}{\Gamma \triangleright \lambda \langle x, y \rangle : \langle \sigma \times \tau \rangle . M: \langle \sigma \times \tau \rangle \rightarrow \rho}$$

Extender el algoritmo

$$\mathbb{W}(\lambda \langle x, y \rangle . U) \stackrel{\text{def}}{=} \Gamma' \triangleright \lambda \langle x, y \rangle : \langle \tau_x \times \tau_y \rangle . M: \langle \tau_x \times \tau_y \rangle \rightarrow \rho$$

donde

$$\tau_x = \begin{cases} \alpha & \text{si } x: \alpha \in \Gamma \\ \text{variable fresca si no.} & \end{cases} \quad \tau_y = \begin{cases} \beta & \text{si } y: \beta \in \Gamma \\ \text{variable fresca si no.} & \end{cases}$$

$$\Gamma' = \Gamma \ominus \{x, y\}$$

Extensiones del lenguaje

Listas

$$\sigma ::= \dots \mid [\sigma]$$

$$M, N, O ::= \dots \mid []_{\sigma} \mid M :: N \mid \text{Case } M \text{ of } [] \rightsquigarrow N ; h :: t \rightsquigarrow O$$

$$\frac{}{\Gamma \triangleright []_{\sigma} : [\sigma]} \quad \frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : [\sigma]}{\Gamma \triangleright M :: N : [\sigma]}$$

$$\frac{\Gamma \triangleright M : [\sigma] \quad \Gamma \triangleright N : \tau \quad \Gamma \cup \{h : \sigma, t : [\sigma]\} \triangleright O : \tau}{\Gamma \triangleright \text{Case } M \text{ of } [] \rightsquigarrow N ; h :: t \rightsquigarrow O : \tau}$$

Extender el algoritmo

$$\mathbb{W}([]) \stackrel{\text{def}}{=} ?$$

$$\mathbb{W}(U_1 :: U_2) \stackrel{\text{def}}{=} ?$$

$$\mathbb{W}(\text{Case } U_1 \text{ of } [] \rightsquigarrow U_2 ; h :: t \rightsquigarrow U_3) \stackrel{\text{def}}{=} ?$$

Ahora usémoslo

$$\mathbb{W}(\text{Case succ}(0) :: x \text{ of } [] \rightsquigarrow x ; x :: y \rightsquigarrow \text{succ}(x) :: []) = ?$$

Otra extensión

Switch de naturales

Switch

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:

$$\frac{\begin{array}{c} \Gamma \triangleright M : \text{Nat} \quad \forall i, j (1 \leq i, j \leq k \wedge i \neq j \Rightarrow n_i \neq n_j) \\ \Gamma \triangleright N_1 : \sigma \quad \dots \quad \Gamma \triangleright N_k : \sigma \quad \Gamma \triangleright N : \sigma \end{array}}{\Gamma \triangleright \text{switch } M \{ \text{case } \underline{n_1} : N_1 \dots \text{case } \underline{n_k} : N_k \text{ default} : N \} : \sigma}$$

Otra extensión del lenguaje

Letrec

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

$M ::= \dots \mid \text{letrec } f = M \text{ in } N$

Permite por ejemplo representar el factorial de 10 de la siguiente manera:

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

Para ello se agrega la siguiente regla de tipado:

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

Extendemos el algoritmo

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

$\mathbb{W}(\text{letrec } f = U_1 \text{ in } U_2) \stackrel{\text{def}}{=} S \Gamma'_1 \cup S \Gamma'_2 \triangleright S(\text{letrec } f = M_1 \text{ in } M_2) : S \tau_2$
 donde

- $\mathbb{W}(U_1) = \Gamma_1 \triangleright M_1 : \tau_1$
- $\mathbb{W}(U_2) = \Gamma_2 \triangleright M_2 : \tau_2$
- $\tau_{f1} = \begin{cases} \alpha_1 & \text{si } f : \alpha_1 \in \Gamma_1 \\ \text{variable fresca en otro caso.} \end{cases}$
- $\tau_{f2} = \begin{cases} \alpha_2 & \text{si } f : \alpha_2 \in \Gamma_2 \\ \text{variable fresca en otro caso.} \end{cases}$
- $\Gamma'_1 = \Gamma_1 \ominus \{f\}$ y $\Gamma'_2 = \Gamma_2 \ominus \{f\}$
- $S = \text{mgu} \{ \tau_{f1} \doteq \tau_{f2}, \tau_1 \doteq t_1 \rightarrow t_2, \tau_1 \doteq \tau_{f1} \}$
 $\cup \{ \sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma'_1, x : \sigma_2 \in \Gamma'_2 \}$
 t_1 y t_2 variables frescas

Otra forma de escribirlo

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

$\mathbb{W}(\text{letrec } f = U_1 \text{ in } U_2) \stackrel{\text{def}}{=} S \Gamma'_1 \cup S \Gamma'_2 \triangleright S(\text{letrec } f = M \text{ in } N) : S \sigma$
 donde

- $\mathbb{W}(U_1) = \Gamma_1 \triangleright M : \gamma$
- $\mathbb{W}(U_2) = \Gamma_2 \triangleright N : \sigma$
- $\tau_f = \begin{cases} \alpha_1 & \text{si } f : \alpha_1 \in \Gamma_1 \\ \alpha_2 & \text{si } f \notin \text{dom}(\Gamma_1) \text{ y } f : \alpha_2 \in \Gamma_2 \\ \text{variable fresca en otro caso.} \end{cases}$
- $\Gamma'_1 = \Gamma_1 \ominus \{f\}$ y $\Gamma'_2 = \Gamma_2 \ominus \{f\}$
- $S = \text{mgu } \{\gamma \doteq t_1 \rightarrow t_2, \gamma \doteq \tau_f\} \cup \{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_1, x : \sigma_2 \in \Gamma_2\}$
 t_1 y t_2 variables frescas

Moraleja

Algunas conclusiones

- Los llamados recursivos devuelven un contexto, un término anotado y un tipo. **No podemos asumir nada sobre ellos.**
- Cuando la regla tiene tipos iguales o tipos con una forma específica: unificar.
- Si hay contextos repetidos en las premisas, unificarlos.
- Cuando la regla liga variables:
 - Obtener su tipo del Γ obtenido recursivamente.
 - Si no figuran: variable fresca.
 - Sacarlas del Γ del resultado (y del que se vaya a unificar).
- Decorar los términos según corresponda.
- Si la regla tiene restricciones adicionales, se incorporan como posibles casos de falla.

i? i? i? i? i? i? i? i? i? i? i? i?