Inferencia de Tipos

PLP

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

26 de abril de 2018

Motivación

Ejemplo

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

$$\emptyset \rhd \lambda x : \text{Nat} \,.\, \text{isZero}(x) : \text{Nat} \to \text{Bool}$$

¿Cómo hicimos? ¿Se puede automatizar? ¿Podríamos no escribir el : 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?

- $\emptyset \rhd \lambda x{:}\mathsf{Nat}.\ \mathsf{succ}(x) : \mathsf{Nat} \to \mathsf{Nat}$
- $\{y{:}\mathsf{Nat}\} \rhd \lambda x{:}\square{:}t.\ \mathsf{succ}(y) :\square : t \to \mathsf{Nat}$
- $(\lambda x.\mathsf{isZero}(x))\ \mathsf{true}$ no tiene tipo.
- $\emptyset \rhd \lambda x{:}\ \mathsf{Nat}.\ x : \mathsf{Nat} \to \mathsf{Nat}$ no es lo más general.
- $\emptyset \rhd \lambda x{:}\ t.\ x : t \to t$ 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 $\emptyset \rhd \lambda x\colon t\,.\,x : t \to t$. Por ejemplo:

- $\emptyset \rhd \lambda x : \mathsf{Nat}.\ x : \mathsf{Nat} \to \mathsf{Nat}$
- $\emptyset \rhd \lambda x : \mathsf{Bool}.\ x : \mathsf{Bool} \to \mathsf{Bool}$
- $\{y\colon \mathsf{Bool}\} \rhd \lambda x : r \to \mathsf{Nat}.\ x : (r \to \mathsf{Nat}) \to (r \to \mathsf{Nat})$
- ...

Recordemos algunas reglas de tipado

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : \sigma}\ (\text{T-Var}) \qquad \frac{\Gamma \cup \{x : \sigma\} \rhd M : \tau}{\Gamma \rhd \lambda x : \sigma . M : \sigma \to \tau}\ (\text{T-Abs})$$

$$\frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\, N : \tau}\ (\text{T-App})$$

Tipado vs. Inferencia

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

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

Tipado vs. Inferencia

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

Otra forma de escribirlo:

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

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

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

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

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

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

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

Tipado vs. Inferencia

$$\frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\ N : \tau}\ (\text{T-App})$$

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

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

- Entonces

$$\mathbb{W}(U\ V) \stackrel{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \rhd S(M\ N) : St$$

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

$$M ::= \ldots \mid \lambda\langle x, y\rangle : \langle \sigma \times \tau \rangle . M$$

$$M' ::= \ldots \mid \lambda\langle x, y\rangle . M'$$

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

Extender el algoritmo

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

donde

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

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

Extensiones del lenguaje

Listas

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

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

$$\frac{}{\Gamma \rhd [\,]_\sigma : [\sigma]} \qquad \frac{\Gamma \rhd M : \sigma \qquad \Gamma \rhd N : [\sigma]}{\Gamma \rhd M :: N : [\sigma]}$$

$$\frac{\Gamma \rhd M : [\sigma] \qquad \Gamma \rhd N : \tau \qquad \Gamma \cup \{h : \sigma, t : [\sigma]\} \rhd O : \tau}{\Gamma \rhd \mathit{Case}\ M\ \mathit{of}\ [\,] \rightsquigarrow N\ ; h :: t \rightsquigarrow O : \tau}$$

Extender el algoritmo

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

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

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

Ahora usémoslo

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

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:

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

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

La regla de tipado es la siguiente:

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

Otra extensión del lenguaje

Letrec

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

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

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:

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

Extendemos el algoritmo

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

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

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

Otra forma de escribirlo

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

$\mathbb{W}(\text{letrec } f = U_1 \text{ in } U_2) \stackrel{\text{def}}{=} S\,\Gamma_1' \cup S\,\Gamma_2' \rhd S\,(\text{letrec } f = M \text{ in } N) : S\,\sigma$
donde

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

PLP ▷ fin clase: consultas

¿? ¿? ¿? ¿? ¿? ¿? ¿? ¿? ¿? ¿? ¿? ¿? ¿?