

## Inferencia de tipos

# Inferencia de tipos

- ▶ Problema que consiste en transformar términos **sin** información de tipos o con información de tipos **parcial** en términos **tipables**
- ▶ Para ello debe **inferirse** la información de tipos faltante
- ▶ Beneficio para lenguajes con tipos
  - ▶ el programador puede obviar algunas declaraciones de tipos
  - ▶ en general, evita la sobrecarga de tener que declarar y manipular **todos** los tipos
  - ▶ todo ello sin desmejorar la performance del programa: la inferencia de tipos se realiza en tiempo de **compilación**

# El problema de la inferencia de tipos

Primero modificamos la sintaxis de los términos de LC **eliminando** toda anotación de tipos

$$\begin{array}{lcl} M & ::= & x \\ & | & \textit{true} \mid \textit{false} \mid \textit{if } M \textit{ then } P \textit{ else } Q \\ & | & 0 \mid \textit{succ}(M) \mid \textit{pred}(M) \mid \textit{iszero}(M) \\ & | & \lambda x.M \mid M \ N \mid \\ & | & \textit{fix } M \end{array}$$

Denotamos este conjunto de términos con  $\Lambda$ .

# Función de borrado

Llamaremos  $\text{ERASE}(\cdot)$  a la función que dado un término de LC **elimina** las anotaciones de tipos de las abstracciones.

$\text{ERASE}(\cdot): \Lambda_{\tau} \rightarrow \Lambda$  se define de la manera esperada.

## Ejemplo

$$\text{ERASE}(\lambda x : \text{Nat}. \lambda f : \text{Nat} \rightarrow \text{Nat}. f\ x) = \lambda x. \lambda f. f\ x$$

# El problema de la inferencia - Definición

Dado un término  $U$  sin anotaciones de tipo, hallar un término estándar (i.e. con anotaciones de tipos)  $M$  tal que

1.  $\Gamma \triangleright M : \sigma$ , para algún  $\Gamma$  y  $\sigma$ , y
2.  $\text{ERASE}(M) = U$

## Ejemplos

- ▶ Para  $U = \lambda x. \text{succ}(x)$  tomamos  $M = \lambda x : \text{Nat}. \text{succ}(x)$   
(observar que no hay otra posibilidad)
- ▶ Para  $U = \lambda x. \lambda f. f\ x$  tomamos  $M_{\sigma, \tau} = \lambda x : \sigma. \lambda f : \sigma \rightarrow \tau. f\ x$   
(hay un  $M_{\sigma, \tau}$  por cada  $\sigma, \tau$ )
- ▶ Para  $U = xx$  no existe ningún  $M$  con la propiedad deseada

# El problema del chequeo de tipos

chequeo de tipos  $\neq$  inferencia de tipos

## Chequeo de tipos

Dado un término estándar  $M$  determinar si existe  $\Gamma$  y  $\sigma$  tales que  $\Gamma \triangleright M : \sigma$  es derivable.

- ▶ Es mucho más fácil que el problema de la inferencia
- ▶ Consiste simplemente en seguir la estructura sintáctica de  $M$  para reconstruir una derivación del juicio
- ▶ Es esencialmente equivalente a determinar, dados  $\Gamma$  y  $\sigma$ , si  $\Gamma \triangleright M : \sigma$  es derivable.

# Variables de tipo



- ▶ Dado  $\lambda x. \lambda f. f (f x)$ , para cada  $\sigma$ ,  
 $M_\sigma = \lambda x : \sigma. \lambda f : \sigma \rightarrow \sigma. f (f x)$  es un solución posible
- ▶ ¿De qué manera podemos escribir una **única** expresión que englobe a todas ellas? Usando **variables de tipo**
  - ▶ Todas las soluciones se pueden representar con
$$\lambda x : s. \lambda f : s \rightarrow s. f (f x)$$
  - ▶ “s” es una variable de tipos que representa una expresión de tipos arbitraria
  - ▶ Si bien esta expresión **no** es una solución en sí misma, la **sustitución** de s por cualquier expresión de tipos **sí** arroja una solución válida

# Variables de tipo

- ▶ Extendemos las expresiones de tipo de LC con variables de tipo  $s, t, u, \dots$

$$\sigma ::= s \mid Nat \mid Bool \mid \sigma \rightarrow \tau$$

- ▶ Denotamos con  $\mathcal{V}$  al conjunto de variables de tipo.
- ▶ Denotamos con  $\mathcal{T}$  al conjunto de tipos así definidos.

## Ejemplos

- ▶  $s \rightarrow t$
- ▶  $Nat \rightarrow Nat \rightarrow t$
- ▶  $Bool \rightarrow t$



# Sustitución de tipos (o simplemente sustitución)

- Función que mapea variables de tipo en expresiones de tipo.

Usamos  $S$ ,  $T$ , etc. para sustituciones.

Formalmente,  $S : \mathcal{V} \rightarrow \mathcal{T}$

Solo nos interesan las  $S$  tales que  $\{t \in \mathcal{V} \mid St \neq t\}$  es finito.



- Una sustitución  $S$  puede aplicarse (de la manera esperada) a
  1. una expresión de tipos  $\sigma$  (escribimos  $S\sigma$ )
  2. un término  $M$  (escribimos  $SM$ )
  3. un contexto de tipado  $\Gamma = \{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$  (escribimos  $S\Gamma$  y lo definimos como sigue)

$$S\Gamma \stackrel{\text{def}}{=} \{x_1 : S\sigma_1, \dots, x_n : S\sigma_n\}$$

## Sustitución - Nociones adicionales

- ▶ El conjunto  $\{t \mid St \neq t\}$  se llama **soporte** de  $S$ .
- ▶ El soporte representa las variables que  $S$  “afecta”.
- ▶ Usamos la notación  $\{\sigma_1/t_1, \dots, \sigma_n/t_n\}$  para la sustitución con soporte  $\{t_1, \dots, t_n\}$  definida de la manera obvia.
- ▶ La sustitución cuyo soporte es  $\emptyset$  es la **sustitución identidad** y la notamos  $Id$ .

# Instancia de un juicio de tipado

Un juicio de tipado  $\Gamma' \triangleright M' : \sigma'$  es una **instancia** de  $\Gamma \triangleright M : \sigma$  si existe una sustitución de tipos  $S$  tal que

$$\Gamma' \supseteq S\Gamma, M' = SM \text{ y } \sigma' = S\sigma$$

## Propiedad

Si  $\Gamma \triangleright M : \sigma$  es derivable, entonces cualquier instancia del mismo también lo es.

# Función de Inferencia $\mathbb{W}(\cdot)$

Definir una función  $\mathbb{W}(\cdot)$  que dado un término  $U$  sin anotaciones verifica

**Corrección**  $\mathbb{W}(U) = \Gamma \triangleright M : \sigma$  implica

- ▶  $\text{ERASE}(M) = U$  y
- ▶  $\Gamma \triangleright M : \sigma$  es derivable

**Compleitud** Si  $\Gamma \triangleright M : \sigma$  es derivable y  $\text{ERASE}(M) = U$ , entonces

- ▶  $\mathbb{W}(U)$  tiene éxito y
- ▶ produce un juicio  $\Gamma' \triangleright M' : \sigma'$  tal que  $\Gamma \triangleright M : \sigma$  es instancia del mismo (se dice que  $\mathbb{W}(\cdot)$  computa un tipo principal)

## Algoritmo de inferencia (caso constantes y variables)

$$\mathbb{W}(0) \stackrel{\text{def}}{=} \emptyset \triangleright 0 : \textit{Nat}$$

$$\mathbb{W}(\textit{true}) \stackrel{\text{def}}{=} \emptyset \triangleright \textit{true} : \textit{Bool}$$

$$\mathbb{W}(\textit{false}) \stackrel{\text{def}}{=} \emptyset \triangleright \textit{false} : \textit{Bool}$$

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

## Algoritmo de inferencia (caso *succ*)

$$\mathbb{W}(\textcolor{red}{succ}(U)) \stackrel{\text{def}}{=}$$

- ▶ Sea  $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
- ▶ Necesitamos saber si  $\tau$  puede ser un *Nat*

# Unificación

- ▶ ¿El tipo  $s \rightarrow t$  es compatible o unificable con  $Nat \rightarrow u$ ? Sí
  - ▶ Basta tomar la sustitución  $S \stackrel{\text{def}}{=} \{Nat/s, u/t\}$
  - ▶ Y observar que  $S(s \rightarrow t) = Nat \rightarrow u = S(Nat \rightarrow u)$
- ▶ ¿El tipo  $s$  es compatible o unificable con  $Nat$ ? Sí
  - ▶ La sustitución antedicha es tal que  $Ss = SNat$

El proceso de determinar si existe una sustitución  $S$  tal que dos expresiones de tipos  $\sigma, \tau$  son unificables (ie.  $S\sigma = S\tau$ ) se llama **unificación**

# Composición de sustituciones

La **composición** de  $S$  y  $T$ , denotada  $S \circ T$ , es la sustitución que se comporta como sigue:

$$(S \circ T)(\sigma) = S(T\sigma)$$

## Ejemplo

Sea  $S = \{u \rightarrow Bool/t, Nat/s\}$  y  $T = \{v \times Nat/u, Nat/s\}$ , entonces  $T \circ S = \{(v \times Nat) \rightarrow Bool/t, v \times Nat/u, Nat/s\}$

- ▶ Decimos que  $S = T$  si tienen el mismo soporte y  $St = Tt$  para todo  $t$  en el soporte de  $S$
- ▶  $S \circ Id = Id \circ S = S$
- ▶  $S \circ (T \circ U) = (S \circ T) \circ U$



# Preorden sobre sustituciones

Una sustitución  $S$  es **más general** que  $T$  si existe  $U$  tal que  $T = U \circ S$ .

- La idea es que  $S$  es más general que  $T$  porque  $T$  se obtiene instanciando  $S$

# Unificador

Una **ecuación de unificación** es una expresión de la forma  $\sigma_1 \doteq \sigma_2$ .  
Una sustitución  $S$  es una **solución** de un conjunto de ecuaciones de unificación  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$  si  $S\sigma_1 = S\sigma'_1, \dots, S\sigma_n = S\sigma'_n$

## Ejemplos

- ▶ La sustitución  $\{Bool/v, Bool \times Nat/u\}$  es solución de  $\{v \times Nat \rightarrow Nat \doteq u \rightarrow Nat\}$
- ▶  $\{Bool \times Bool/v, (Bool \times Bool) \times Nat/u\}$  también!
- ▶  $\{v \times Nat/u\}$  también!
- ▶  $\{Nat \rightarrow s \doteq t \times u\}$  **no** tiene solución
- ▶  $\{u \rightarrow Nat \doteq u\}$  **no** tiene solución

# Unificador más general (MGU)

Una sustitución  $S$  es un **MGU** de  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$  si

1. es solución de  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$
2. es más general que cualquier otra solución de  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$

## Ejemplos

- ▶ La sustitución  $\{Bool/v, Bool \times Nat/u\}$  es solución de  $\{v \times Nat \rightarrow Nat \doteq u \rightarrow Nat\}$  pero no es un MGU pues es instancia de la solución  $\{v \times Nat/u\}$
- ▶  $\{v \times Nat/u\}$  es un MGU del conjunto

# Algoritmo de unificación

## Teorema

Si  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$  tiene solución, existe un MGU y además es único salvo renombre de variables

- ▶ Entrada:

- ▶ Conjunto de ecuaciones de unificación  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$

- ▶ Salida:

- ▶ **MGU**  $S$  de  $\{\sigma_1 \doteq \sigma'_1, \dots, \sigma_n \doteq \sigma'_n\}$ , si tiene solución

- ▶ **falla**, en caso contrario

# Algoritmo de Martelli-Montanari

- ▶ Vamos a presentar un algoritmo no-determinístico
- ▶ Consiste en **reglas de simplificación** que simplifican conjuntos de pares de tipos a unificar (*goals*)

$$G_0 \mapsto G_1 \mapsto \dots \mapsto G_n$$

- ▶ Las secuencias que terminan en el goal vacío son **exitosas**; aquellas que terminan en **falla** son **fallidas**
- ▶ Algunos pasos de simplificación llevan a una sustitución que representa una solución parcial al problema

$$G_0 \mapsto G_1 \mapsto_{S_1} G_2 \mapsto \dots \mapsto_{S_k} G_n$$

- ▶ Si la secuencia es exitosa el MGU es  $S_k \circ \dots \circ S_1$

# Reglas del algoritmo de Martelli-Montanari

## 1. Descomposición

$$\{\sigma_1 \rightarrow \sigma_2 \doteq \tau_1 \rightarrow \tau_2\} \cup G \mapsto \{\sigma_1 \doteq \tau_1, \sigma_2 \doteq \tau_2\} \cup G$$

$$\{Nat \doteq Nat\} \cup G \mapsto G$$

$$\{Bool \doteq Bool\} \cup G \mapsto G$$

## 2. Eliminación de par trivial

$$\{s \doteq s\} \cup G \mapsto G$$

## 3. Swap: si $\sigma$ no es una variable

$$\{\sigma \doteq s\} \cup G \mapsto \{s \doteq \sigma\} \cup G$$

## 4. Eliminación de variable: si $s \notin FV(\sigma)$

$$\{s \doteq \sigma\} \cup G \mapsto_{\{\sigma/s\}} \{\sigma/s\} G$$

## 5. Colisión

$$\{\sigma \doteq \tau\} \cup G \mapsto \text{falla}, \text{ con } (\sigma, \tau) \in T \cup T^{-1} \text{ y}$$

$$T = \{(Bool, Nat), (Nat, \sigma_1 \rightarrow \sigma_2), (Bool, \sigma_1 \rightarrow \sigma_1)\}$$

## 6. Occur check: si $s \neq \sigma$ y $s \in FV(\sigma)$

$$\{s \doteq \sigma\} \cup G \mapsto \text{falla}$$

## Ejemplo de secuencia exitosa

$$\begin{array}{ll} \vdash^1 & \{(Nat \rightarrow r) \rightarrow (r \rightarrow u) \doteq t \rightarrow (s \rightarrow s) \rightarrow t\} \\ \vdash^3 & \{Nat \rightarrow r \doteq t, r \rightarrow u \doteq (s \rightarrow s) \rightarrow t\} \\ \vdash^4_{Nat \rightarrow r/t} & \{t \doteq Nat \rightarrow r, r \rightarrow u \doteq (s \rightarrow s) \rightarrow t\} \\ \vdash^1 & \{r \rightarrow u \doteq (s \rightarrow s) \rightarrow (Nat \rightarrow r)\} \\ \vdash^4_{s \rightarrow s/r} & \{r \doteq s \rightarrow s, u \doteq Nat \rightarrow r\} \\ \vdash^4_{Nat \rightarrow (s \rightarrow s)/u} & \{u \doteq Nat \rightarrow (s \rightarrow s)\} \\ & \emptyset \end{array}$$

► El MGU es

$$\begin{aligned} & \{Nat \rightarrow (s \rightarrow s)/u\} \circ \{s \rightarrow s/r\} \circ \{Nat \rightarrow r/t\} = \\ & \{Nat \rightarrow (s \rightarrow s)/t, s \rightarrow s/r, Nat \rightarrow (s \rightarrow s)/u\} \end{aligned}$$

## Ejemplo de secuencia fallida

$$\begin{array}{ll} & \{r \rightarrow (s \rightarrow r) \doteq s \rightarrow ((r \rightarrow \text{Nat}) \rightarrow r)\} \\ \mapsto^1 & \{r \doteq s, s \rightarrow r \doteq (r \rightarrow \text{Nat}) \rightarrow r\} \\ \mapsto_{s/r}^4 & \{s \rightarrow s \doteq (s \rightarrow \text{Nat}) \rightarrow s\} \\ \mapsto^1 & \{s \doteq s \rightarrow \text{Nat}, s \doteq s\} \\ \mapsto^6 & \text{falla} \end{array}$$



# Propiedades del algoritmo

## Teorema

- ▶ El algoritmo de Martelli-Montanari siempre termina
- ▶ Sea  $G$  un conjunto de pares
  - ▶ si  $G$  tiene un unificador, el algoritmo termina exitosamente y retorna un MGU
  - ▶ si  $G$  no tiene unificador, el algoritmo termina con **falla**

## Algoritmo de inferencia (caso *succ*)

- ▶ Sea  $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
- ▶ Sea  $S = MGU\{\tau \doteq Nat\}$
- ▶ Entonces

$$\mathbb{W}(\textcolor{red}{succ}(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S \textcolor{red}{succ}(M) : Nat$$

- ▶ Nota: Caso *pred* es similar

## Algoritmo de inferencia (caso *iszero*)

- ▶ Sea  $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
- ▶ Sea  $S = MGU\{\tau \doteq Nat\}$
- ▶ Entonces

$$\mathbb{W}(\textcolor{red}{iszero}(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S \textcolor{blue}{iszero}(M) : \textcolor{blue}{Bool}$$

# Algoritmo de inferencia (caso ifThenElse)

► Sea

- $\mathbb{W}(U) = \Gamma_1 \triangleright M : \rho$
- $\mathbb{W}(V) = \Gamma_2 \triangleright P : \sigma$
- $\mathbb{W}(W) = \Gamma_3 \triangleright Q : \tau$

► Sea

$$S = MGU(\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_i \wedge x : \sigma_2 \in \Gamma_j, i \neq j\} \\ \cup \\ \{\sigma \doteq \tau, \rho \doteq Bool\})$$

► Entonces

$$\mathbb{W}(\textit{if } U \textit{ then } V \textit{ else } W) \stackrel{\text{def}}{=} \\ S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3 \triangleright S(\textit{if } M \textit{ then } P \textit{ else } Q) : S\sigma$$

# Algoritmo de inferencia (caso aplicación)

- ▶ Sea

- ▶  $\mathbb{W}(U) = \Gamma_1 \triangleright M : \tau$

- ▶  $\mathbb{W}(V) = \Gamma_2 \triangleright N : \rho$

- ▶ Sea

$$S = MGU(\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_1 \wedge x : \sigma_2 \in \Gamma_2\} \\ \cup$$

$\{\tau \doteq \rho \rightarrow t\})$  con  $t$  una variable fresca

- ▶ Entonces

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

## Algoritmo de inferencia (caso abstracción)

- ▶ 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  $\tau$ ), 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  $s$  y entonces

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

## Algoritmo de inferencia (caso *fix*)

- ▶ Sea  $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
- ▶ Sea  $S = MGU\{\tau \doteq t \rightarrow t\}$ ,  $t$  variable fresca

$$\mathbb{W}(\textcolor{red}{fix}(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S \textcolor{blue}{fix}(M) : St$$