Paradigmas de Programación

Inferencia de tipos

1er cuatrimestre de 2024

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Inferencia de tipos

Inferencia de tipos

Notación

Términos **sin** anotaciones de tipos:

$$U ::= x \mid \lambda x.\, U \mid U\, U \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ U\ \mathsf{then}\ U\ \mathsf{else}\ U$$

Inferencia de tipos

Notación

Términos **sin** anotaciones de tipos:

$$U ::= x \mid \lambda x.\, U \mid U\, U \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ U\ \mathsf{then}\ U\ \mathsf{else}\ U$$

Términos **con** anotaciones de tipos:

$$M ::= x \mid \lambda x : \tau.\, M \mid M\, M \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M$$

Inferencia de tipos

Notación

Términos **sin** anotaciones de tipos:

$$U ::= x \mid \lambda x.\, U \mid U\, U \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ U\ \mathsf{then}\ U\ \mathsf{else}\ U$$

Términos **con** anotaciones de tipos:

$$M ::= x \mid \lambda x : \tau.\, M \mid M\, M \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M$$

Notamos $\mathtt{erase}(M)$ al término sin anotaciones de tipos que resulta de borrar las anotaciones de tipos de M.

Inferencia de tipos

Notación

Términos **sin** anotaciones de tipos:

$$U ::= x \mid \lambda x.\, U \mid U\, U \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ U\ \mathsf{then}\ U\ \mathsf{else}\ U$$

Términos **con** anotaciones de tipos:

$$M ::= x \mid \lambda x : \tau.\, M \mid M\, M \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M$$

Notamos $\texttt{erase}(M)$ al término sin anotaciones de tipos que resulta de borrar las anotaciones de tipos de M.

Ejemplo: $\texttt{erase}((\lambda x : \mathsf{Bool}.\, x)\, \mathsf{True}) = (\lambda x.\, x)\, \mathsf{True}$.

Inferencia de tipos

Definición

Un término U sin anotaciones de tipos es **tipable** sii existen:

un contexto de tipado Γ

un término con anotaciones de tipos M

un tipo τ

tales que $\mathtt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

Inferencia de tipos

Definición

Un término U sin anotaciones de tipos es **tipable** sii existen:

un contexto de tipado Γ

un término con anotaciones de tipos M

un tipo τ

tales que $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

El **problema de inferencia de tipos** consiste en:

- Dado un término U, determinar si es tipable.
- En caso de que U sea tipable:
  hallar un contexto Γ, un término M y un tipo τ
  tales que $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

Inferencia de tipos

Definición

Un término U sin anotaciones de tipos es **tipable** sii existen:

un contexto de tipado Γ

un término con anotaciones de tipos M

un tipo τ

tales que $\mathtt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

El **problema de inferencia de tipos** consiste en:

- Dado un término U, determinar si es tipable.
- En caso de que U sea tipable:
  hallar un contexto Γ, un término M y un tipo τ
  tales que $\mathtt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

Veremos un algoritmo para resolver este problema.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En $x\,\mathsf{True}$ sabemos que $x : \mathsf{Bool} \to \mathtt{X1}$.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En $x\,\mathrm{True}$ sabemos que $x : \mathrm{Bool} \to \mathtt{X1}$.
- En $\mathrm{if}\ x\,y\ \mathrm{then}\ \mathrm{True}\ \mathrm{else}\ \mathrm{False}$ sabemos que $x : \mathtt{X2} \to \mathrm{Bool}$.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \mathsf{Bool} \to \mathtt{X1}$.
- En if $x\ y$ then True else False sabemos que $x : \mathtt{X2} \to \mathsf{Bool}$.

Incorporamos *incógnitas* ($\mathtt{X1}, \mathtt{X2}, \mathtt{X3}, \ldots$) a los tipos.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \mathrm{Bool} \to \mathtt{X1}$.
- En if $x\,y$ then True else False sabemos que $x : \mathtt{X2} \to \mathrm{Bool}$.

Incorporamos *incógnitas* ($\mathtt{X1}, \mathtt{X2}, \mathtt{X3}, \ldots$) a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \text{Bool} \to \texttt{X1}$.
- En if $x\,y$ then True else False sabemos que $x : \texttt{X2} \to \text{Bool}$.

Incorporamos *incógnitas* ($\texttt{X1}, \texttt{X2}, \texttt{X3}, \ldots$) a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\texttt{X1} \to \text{Bool}) \stackrel{?}{=} ((\text{Bool} \to \text{Bool}) \to \texttt{X2})$

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En $x\,\mathsf{True}$ sabemos que $x : \mathsf{Bool} \to \mathtt{X1}$.
- En $\mathsf{if}\ x\ y\ \mathsf{then\ True\ else\ False}$ sabemos que $x : \mathtt{X2} \to \mathsf{Bool}$.

Incorporamos *incógnitas* $(\mathtt{X1}, \mathtt{X2}, \mathtt{X3}, \ldots)$ a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\mathtt{X1} \to \mathsf{Bool}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$
  tiene solución: $\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool})$ y $\mathtt{X2} := \mathsf{Bool}$.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \mathsf{Bool} \to \mathtt{X1}$.
- En if $x\ y$ then True else False sabemos que $x : \mathtt{X2} \to \mathsf{Bool}$.

Incorporamos *incógnitas* ($\mathtt{X1}, \mathtt{X2}, \mathtt{X3}, \ldots$) a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\mathtt{X1} \to \mathsf{Bool}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$
  tiene solución: $\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool})$ y $\mathtt{X2} := \mathsf{Bool}$.
- $(\mathtt{X1} \to \mathtt{X1}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \mathsf{Bool} \to \mathtt{X1}$.
- En if $x\,y$ then True else False sabemos que $x : \mathtt{X2} \to \mathsf{Bool}$.

Incorporamos *incógnitas* ($\mathtt{X1}, \mathtt{X2}, \mathtt{X3}, \ldots$) a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\mathtt{X1} \to \mathsf{Bool}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$
  tiene solución: $\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool})$ y $\mathtt{X2} := \mathsf{Bool}$.
- $(\mathtt{X1} \to \mathtt{X1}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$
  tiene solución: $\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool})$ y $\mathtt{X2} := (\mathsf{Bool} \to \mathsf{Bool})$.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En $x\,\mathsf{True}$ sabemos que $x : \mathsf{Bool} \to \mathtt{X1}$.
- En $\mathsf{if}\ x\,y\ \mathsf{then}\ \mathsf{True}\ \mathsf{else}\ \mathsf{False}$ sabemos que $x : \mathtt{X2} \to \mathsf{Bool}$.

Incorporamos *incógnitas* $(\mathtt{X1}, \mathtt{X2}, \mathtt{X3}, \ldots)$ a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\mathtt{X1} \to \mathsf{Bool}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$
  tiene solución: $\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool})$ y $\mathtt{X2} := \mathsf{Bool}$.
- $(\mathtt{X1} \to \mathtt{X1}) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to \mathtt{X2})$
  tiene solución: $\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool})$ y $\mathtt{X2} := (\mathsf{Bool} \to \mathsf{Bool})$.
- $(\mathtt{X1} \to \mathsf{Bool}) \stackrel{?}{=} \mathtt{X1}$

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \text{Bool} \to \texttt{X1}$.
- En if $x\,y$ then True else False sabemos que $x : \texttt{X2} \to \text{Bool}$.

Incorporamos *incógnitas* ($\texttt{X1}, \texttt{X2}, \texttt{X3}, \ldots$) a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\texttt{X1} \to \text{Bool}) \stackrel{?}{=} ((\text{Bool} \to \text{Bool}) \to \texttt{X2})$
  tiene solución: $\texttt{X1} := (\text{Bool} \to \text{Bool})$ y $\texttt{X2} := \text{Bool}$.
- $(\texttt{X1} \to \texttt{X1}) \stackrel{?}{=} ((\text{Bool} \to \text{Bool}) \to \texttt{X2})$
  tiene solución: $\texttt{X1} := (\text{Bool} \to \text{Bool})$ y $\texttt{X2} := (\text{Bool} \to \text{Bool})$.
- $(\texttt{X1} \to \text{Bool}) \stackrel{?}{=} \texttt{X1}$
  no tiene solución.

Unificación

Suponemos fijado un conjunto finito de constructores de tipos:

- Tipos constantes: Bool, Int, $\ldots$.
- Constructores unarios: (List $\bullet$), (Maybe $\bullet$), $\ldots$.
- Constructores binarios: $(\bullet \to \bullet)$, $(\bullet \times \bullet)$, (Either $\bullet$ $\bullet$), $\ldots$.
- (Etcétera).

Los tipos se forman usando incógnitas y constructores:

$$\tau ::= \mathtt{X}n \mid C(\tau_1, \ldots, \tau_n)$$

Unificación

Suponemos fijado un conjunto finito de constructores de tipos:

- Tipos constantes: Bool, Int, ….
- Constructores unarios: (List •), (Maybe •), ….
- Constructores binarios: $(\bullet \to \bullet)$, $(\bullet \times \bullet)$, (Either • •), ….
- (Etcétera).

Los tipos se forman usando incógnitas y constructores:

$$\tau ::= \mathtt{X}n \mid C(\tau_1, \dots, \tau_n)$$

La **unificación** es el problema de resolver sistemas de ecuaciones entre tipos con incógnitas.

Unificación

Suponemos fijado un conjunto finito de constructores de tipos:

- Tipos constantes: Bool, Int, ...
- Constructores unarios: (List $\bullet$), (Maybe $\bullet$), ...
- Constructores binarios: $(\bullet \to \bullet)$, $(\bullet \times \bullet)$, (Either $\bullet$ $\bullet$), ...
- (Etcétera).

Los tipos se forman usando incógnitas y constructores:

$$\tau ::= \mathtt{X}n \mid C(\tau_1, \ldots, \tau_n)$$

La **unificación** es el problema de resolver sistemas de ecuaciones entre tipos con incógnitas.

Veremos primero un algoritmo de unificación.
Luego lo usaremos para dar un algoritmo de inferencia de tipos.

Unificación

Una **sustitución** es una función que a cada incógnita le asocia un tipo.

Unificación

Una **sustitución** es una función que a cada incógnita le asocia un tipo.

Notamos:

$$\{\mathtt{X}k_1 := \tau_1, \ldots, \mathtt{X}k_n := \tau_n\}$$

a la sustitución $\mathbf{S}$ tal que $\mathbf{S}(\mathtt{X}k_i) = \tau_i$ para cada $1 \leq i \leq n$ y $\mathbf{S}(\mathtt{X}k) = \mathtt{X}k$ para cualquier otra incógnita.

Unificación

Una **sustitución** es una función que a cada incógnita le asocia un tipo.

Notamos:

$$\{\mathtt{X}k_1 := \tau_1, \ldots, \mathtt{X}k_n := \tau_n\}$$

a la sustitución $\mathbf{S}$ tal que $\mathbf{S}(\mathtt{X}k_i) = \tau_i$ para cada $1 \leq i \leq n$ y $\mathbf{S}(\mathtt{X}k) = \mathtt{X}k$ para cualquier otra incógnita.

Si τ es un tipo, escribimos $\mathbf{S}(\tau)$ para el resultado de reemplazar cada incógnita de τ por el valor que le otorga $\mathbf{S}$.

Unificación

Una **sustitución** es una función que a cada incógnita le asocia un tipo.

Notamos:

$$\{\mathtt{X}k_1 := \tau_1, \ldots, \mathtt{X}k_n := \tau_n\}$$

a la sustitución $\mathbf{S}$ tal que $\mathbf{S}(\mathtt{X}k_i) = \tau_i$ para cada $1 \leq i \leq n$ y $\mathbf{S}(\mathtt{X}k) = \mathtt{X}k$ para cualquier otra incógnita.

Si τ es un tipo, escribimos $\mathbf{S}(\tau)$ para el resultado de reemplazar cada incógnita de τ por el valor que le otorga $\mathbf{S}$.

Ejemplo — aplicación de una sustitución a un tipo

Si $\mathbf{S} = \{\mathtt{X1} := \mathrm{Bool},\ \mathtt{X3} := (\mathtt{X2} \to \mathtt{X2})\}$, entonces:

$$\mathbf{S}((\mathtt{X1} \to \mathrm{Bool}) \to \mathtt{X3}) =$$

Unificación

Una **sustitución** es una función que a cada incógnita le asocia un tipo.

Notamos:

$$\{\mathtt{X}k_1 := \tau_1, \ldots, \mathtt{X}k_n := \tau_n\}$$

a la sustitución $\mathbf{S}$ tal que $\mathbf{S}(\mathtt{X}k_i) = \tau_i$ para cada $1 \leq i \leq n$ y $\mathbf{S}(\mathtt{X}k) = \mathtt{X}k$ para cualquier otra incógnita.

Si τ es un tipo, escribimos $\mathbf{S}(\tau)$ para el resultado de reemplazar cada incógnita de τ por el valor que le otorga $\mathbf{S}$.

Ejemplo — aplicación de una sustitución a un tipo

Si $\mathbf{S} = \{\mathtt{X1} := \mathsf{Bool},\ \mathtt{X3} := (\mathtt{X2} \to \mathtt{X2})\}$, entonces:

$$\mathbf{S}((\mathtt{X1} \to \mathsf{Bool}) \to \mathtt{X3}) = ((\mathsf{Bool} \to \mathsf{Bool}) \to (\mathtt{X2} \to \mathtt{X2}))$$

Unificación

Un **problema de unificación** es un conjunto finito E de ecuaciones entre tipos que pueden involucrar incógnitas:

$$E = \{\tau_1 \stackrel{?}{=} \sigma_1, \tau_2 \stackrel{?}{=} \sigma_2, \dots, \tau_n \stackrel{?}{=} \sigma_n\}$$

Unificación

Un **problema de unificación** es un conjunto finito E de ecuaciones entre tipos que pueden involucrar incógnitas:

$$E = \{\tau_1 \stackrel{?}{=} \sigma_1, \tau_2 \stackrel{?}{=} \sigma_2, \ldots, \tau_n \stackrel{?}{=} \sigma_n\}$$

Un **unificador** para E es una sustitución $\mathbf{S}$ tal que:

$$\mathbf{S}(\tau_1) = \mathbf{S}(\sigma_1)$$

$$\mathbf{S}(\tau_2) = \mathbf{S}(\sigma_2)$$

$$\ldots$$

$$\mathbf{S}(\tau_n) = \mathbf{S}(\sigma_n)$$

Unificación

En general, la solución a un problema de unificación no es única.

Unificación

En general, la solución a un problema de unificación no es única.

Ejemplo — problema de unificación con infinitas soluciones

$$\{\mathtt{X1} \stackrel{?}{=} \mathtt{X2}\}$$

tiene infinitos unificadores:

- $\{\mathtt{X1} := \mathtt{X2}\}$
- $\{\mathtt{X2} := \mathtt{X1}\}$
- $\{\mathtt{X1} := \mathtt{X3},\ \mathtt{X2} := \mathtt{X3}\}$
- $\{\mathtt{X1} := \mathsf{Bool},\ \mathtt{X2} := \mathsf{Bool}\}$
- $\{\mathtt{X1} := (\mathsf{Bool} \to \mathsf{Bool}),\ \mathtt{X2} := (\mathsf{Bool} \to \mathsf{Bool})\}$
- $\ldots$

Unificación

Una sustitución $\mathbf{S}_A$ es **más general** que una sustitución $\mathbf{S}_B$ si existe una sustitución $\mathbf{S}_C$ tal que:

$$\mathbf{S}_B = \mathbf{S}_C \circ \mathbf{S}_A$$

es decir, $\mathbf{S}_B$ se obtiene instanciando variables de $\mathbf{S}_A$.

Unificación

Una sustitución $\mathbf{S}_A$ es **más general** que una sustitución $\mathbf{S}_B$ si existe una sustitución $\mathbf{S}_C$ tal que:

$$\mathbf{S}_B = \mathbf{S}_C \circ \mathbf{S}_A$$

es decir, $\mathbf{S}_B$ se obtiene instanciando variables de $\mathbf{S}_A$.

Para el siguiente problema de unificación:

$$E = \{(\mathtt{X1} \to \mathsf{Bool}) \stackrel{?}{=} \mathtt{X2}\}$$

las siguientes sustituciones son unificadores:

- $\mathbf{S}_1 = \{\mathtt{X1} := \mathsf{Bool},\ \mathtt{X2} := (\mathsf{Bool} \to \mathsf{Bool})\}$
- $\mathbf{S}_2 = \{\mathtt{X1} := \mathsf{Int},\ \mathtt{X2} := (\mathsf{Int} \to \mathsf{Bool})\}$
- $\mathbf{S}_3 = \{\mathtt{X1} := \mathtt{X3},\ \mathtt{X2} := (\mathtt{X3} \to \mathsf{Bool})\}$
- $\mathbf{S}_4 = \{\mathtt{X2} := (\mathtt{X1} \to \mathsf{Bool})\}$

¿Qué relación hay entre ellas? (¿Cuál es más general que cuál?).

Algoritmo de unificación de Martelli–Montanari

Algoritmo de unificación de Martelli–Montanari

Dado un problema de unificación E (conjunto de ecuaciones):

Algoritmo de unificación de Martelli–Montanari

Dado un problema de unificación E (conjunto de ecuaciones):

- Mientras $E \neq \varnothing$, se aplica sucesivamente alguna de las seis reglas que se detallan más adelante.

Algoritmo de unificación de Martelli–Montanari

Dado un problema de unificación E (conjunto de ecuaciones):

- Mientras $E \neq \emptyset$, se aplica sucesivamente alguna de las seis reglas que se detallan más adelante.
- La regla puede resultar en una `falla`.

Algoritmo de unificación de Martelli–Montanari

Dado un problema de unificación E (conjunto de ecuaciones):

- Mientras $E \neq \varnothing$, se aplica sucesivamente alguna de las seis reglas que se detallan más adelante.
- La regla puede resultar en una `falla`.
- De lo contrario, la regla es de la forma $E \to_{\mathbf{S}} E'$.
  La resolución del problema E se reduce a resolver otro problema E', aplicando la sustitución $\mathbf{S}$.

Algoritmo de unificación de Martelli–Montanari

Dado un problema de unificación E (conjunto de ecuaciones):

- Mientras $E \neq \varnothing$, se aplica sucesivamente alguna de las seis reglas que se detallan más adelante.
- La regla puede resultar en una `falla`.
- De lo contrario, la regla es de la forma $E \to_{\mathbf{S}} E'$.
  La resolución del problema E se reduce a resolver otro problema E', aplicando la sustitución $\mathbf{S}$.

Hay dos posibilidades:

1. $E = E_0 \to_{\mathbf{S}_1} E_1 \to_{\mathbf{S}_2} E_2 \to \ldots \to_{\mathbf{S}_n} E_n \to_{\mathbf{S}_{n+1}} \texttt{falla}$
   En tal caso el problema de unificación E no tiene solución.
2. $E = E_0 \to_{\mathbf{S}_1} E_1 \to_{\mathbf{S}_2} E_2 \to \ldots \to_{\mathbf{S}_n} E_n = \varnothing$
   En tal caso el problema de unificación E tiene solución.

Algoritmo de unificación de Martelli–Montanari

$$\{\mathtt{X}n \stackrel{?}{=} \mathtt{X}n\} \cup E \quad \xrightarrow{\mathtt{Delete}} \quad E$$

Algoritmo de unificación de Martelli–Montanari

$$\{\mathtt{X}n \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\texttt{Delete}} E$$

$$\{C(\tau_1, \ldots, \tau_n) \stackrel{?}{=} C(\sigma_1, \ldots, \sigma_n)\} \cup E \xrightarrow{\texttt{Decompose}} \{\tau_1 \stackrel{?}{=} \sigma_1, \ldots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$$

Algoritmo de unificación de Martelli–Montanari

$$\{\mathtt{X}n \stackrel{?}{=} \mathtt{X}n\} \cup E \quad \xrightarrow{\texttt{Delete}} \quad E$$

$$\{C(\tau_1, \ldots, \tau_n) \stackrel{?}{=} C(\sigma_1, \ldots, \sigma_n)\} \cup E \quad \xrightarrow{\texttt{Decompose}} \quad \{\tau_1 \stackrel{?}{=} \sigma_1, \ldots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$$

$$\{\tau \stackrel{?}{=} \mathtt{X}n\} \cup E \quad \xrightarrow{\texttt{Swap}} \quad \{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E$$

si τ no es una incógnita

Algoritmo de unificación de Martelli–Montanari

$$\{\mathtt{X}n \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\mathtt{Delete}} E$$

$$\{C(\tau_1, \dots, \tau_n) \stackrel{?}{=} C(\sigma_1, \dots, \sigma_n)\} \cup E \xrightarrow{\mathtt{Decompose}} \{\tau_1 \stackrel{?}{=} \sigma_1, \dots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$$

$$\{\tau \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\mathtt{Swap}} \{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E$$

si τ no es una incógnita

$$\{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E \xrightarrow{\mathtt{Elim}}_{\{\mathtt{X}n := \tau\}} E' = \{\mathtt{X}n := \tau\}(E)$$

si $\mathtt{X}n$ no ocurre en τ

Algoritmo de unificación de Martelli–Montanari

$$\{\mathtt{X}n \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\texttt{Delete}} E$$

$$\{C(\tau_1, \ldots, \tau_n) \stackrel{?}{=} C(\sigma_1, \ldots, \sigma_n)\} \cup E \xrightarrow{\texttt{Decompose}} \{\tau_1 \stackrel{?}{=} \sigma_1, \ldots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$$

$$\{\tau \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\texttt{Swap}} \{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E$$

si τ no es una incógnita

$$\{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E \xrightarrow{\texttt{Elim}}_{\{\mathtt{X}n := \tau\}} E' = \{\mathtt{X}n := \tau\}(E)$$

si $\mathtt{X}n$ no ocurre en τ

$$\{C(\tau_1, \ldots, \tau_n) \stackrel{?}{=} C'(\sigma_1, \ldots, \sigma_m)\} \cup E \xrightarrow{\texttt{Clash}} \texttt{falla}$$

si $C \neq C'$

Algoritmo de unificación de Martelli–Montanari

$$\{\mathtt{X}n \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\texttt{Delete}} E$$

$$\{C(\tau_1, \ldots, \tau_n) \stackrel{?}{=} C(\sigma_1, \ldots, \sigma_n)\} \cup E \xrightarrow{\texttt{Decompose}} \{\tau_1 \stackrel{?}{=} \sigma_1, \ldots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$$

$$\{\tau \stackrel{?}{=} \mathtt{X}n\} \cup E \xrightarrow{\texttt{Swap}} \{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E$$

si τ no es una incógnita

$$\{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E \xrightarrow{\texttt{Elim}}_{\{\mathtt{X}n := \tau\}} E' = \{\mathtt{X}n := \tau\}(E)$$

si $\mathtt{X}n$ no ocurre en τ

$$\{C(\tau_1, \ldots, \tau_n) \stackrel{?}{=} C'(\sigma_1, \ldots, \sigma_m)\} \cup E \xrightarrow{\texttt{Clash}} \texttt{falla}$$

si $C \neq C'$

$$\{\mathtt{X}n \stackrel{?}{=} \tau\} \cup E \xrightarrow{\texttt{Occurs-Check}} \texttt{falla}$$

si $\mathtt{X}n \neq \tau$
y $\mathtt{X}n$ ocurre en τ

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.
2. Si E no tiene solución, el algoritmo llega a una `falla`.

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.
2. Si E no tiene solución, el algoritmo llega a una `falla`.
3. Si E tiene solución, el algoritmo llega a $\varnothing$:

$$E = E_0 \rightarrow_{\mathbf{s}_1} E_1 \rightarrow_{\mathbf{s}_2} E_2 \rightarrow \ldots \rightarrow_{\mathbf{s}_n} E_n = \varnothing$$

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.
2. Si E no tiene solución, el algoritmo llega a una `falla`.
3. Si E tiene solución, el algoritmo llega a $\varnothing$:

$$E = E_0 \rightarrow_{\mathbf{S}_1} E_1 \rightarrow_{\mathbf{S}_2} E_2 \rightarrow \ldots \rightarrow_{\mathbf{S}_n} E_n = \varnothing$$

   Además, $\mathbf{S} = \mathbf{S}_n \circ \ldots \circ \mathbf{S}_2 \circ \mathbf{S}_1$ es un unificador para E.

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.
2. Si E no tiene solución, el algoritmo llega a una `falla`.
3. Si E tiene solución, el algoritmo llega a $\emptyset$:

   $$E = E_0 \to_{\mathbf{S}_1} E_1 \to_{\mathbf{S}_2} E_2 \to \ldots \to_{\mathbf{S}_n} E_n = \emptyset$$

   Además, $\mathbf{S} = \mathbf{S}_n \circ \ldots \circ \mathbf{S}_2 \circ \mathbf{S}_1$ es un unificador para E.

   Además, dicho unificador es el *más general* posible.
   (Salvo renombre de incógnitas).

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.
2. Si E no tiene solución, el algoritmo llega a una `falla`.
3. Si E tiene solución, el algoritmo llega a $\varnothing$:

   $$E = E_0 \to_{\mathbf{S}_1} E_1 \to_{\mathbf{S}_2} E_2 \to \ldots \to_{\mathbf{S}_n} E_n = \varnothing$$

   Además, $\mathbf{S} = \mathbf{S}_n \circ \ldots \circ \mathbf{S}_2 \circ \mathbf{S}_1$ es un unificador para E.

   Además, dicho unificador es el *más general* posible.
   (Salvo renombre de incógnitas).

Definición (Unificador más general)

Notamos $\mathrm{mgu}(E)$ al unificador más general de E, si existe.

Algoritmo de unificación de Martelli–Montanari

Ejemplo

Calcular unificadores más generales para los siguientes problemas de unificación:

- $\{(\mathtt{X2} \to (\mathtt{X1} \to \mathtt{X1})) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to (\mathtt{X1} \to \mathtt{X2}))\}$
- $\{\mathtt{X1} \stackrel{?}{=} (\mathtt{X2} \to \mathtt{X2}),\ \mathtt{X2} \stackrel{?}{=} (\mathtt{X1} \to \mathtt{X1})\}$

Algoritmo $\mathbb{W}$ de inferencia de tipos

Algoritmo $\mathbb{W}$ de inferencia de tipos

El algoritmo $\mathbb{W}$ recibe un término U sin anotaciones de tipos.

Procede recursivamente sobre la estructura de U:

- Puede fallar, indicando que U no es tipable.
- Puede tener éxito.
  En tal caso devuelve una tripla (Γ, M, τ),
  donde $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$ es válido.

Algoritmo $\mathbb{W}$ de inferencia de tipos

El algoritmo $\mathbb{W}$ recibe un término U sin anotaciones de tipos.

Procede recursivamente sobre la estructura de U:

- Puede fallar, indicando que U no es tipable.
- Puede tener éxito.
  En tal caso devuelve una tripla (Γ, M, τ),
  donde $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$ es válido.

Escribimos $\mathbb{W}(U) \rightsquigarrow \Gamma \vdash M : \tau$ para indicar que el algoritmo de inferencia tiene éxito cuando se le pasa U como entrada y devuelve una tripla (Γ, M, τ).

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\overline{\mathbb{W}(\mathsf{True}) \rightsquigarrow \qquad\qquad\qquad\qquad}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\mathsf{True}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{True} : \mathsf{Bool}}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\mathsf{True}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{True} : \mathsf{Bool}}$$

$$\frac{}{\mathbb{W}(\mathsf{False}) \;\rightsquigarrow\; \phantom{\varnothing \vdash \mathsf{False} : \mathsf{Bool}}}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\mathsf{True}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{True} : \mathsf{Bool}}$$

$$\frac{}{\mathbb{W}(\mathsf{False}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{False} : \mathsf{Bool}}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\mathsf{True}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{True} : \mathsf{Bool}}$$

$$\frac{}{\mathbb{W}(\mathsf{False}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{False} : \mathsf{Bool}}$$

$$\frac{}{\mathbb{W}(x) \;\rightsquigarrow\; \qquad\qquad\qquad}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\mathsf{True}) \ \rightsquigarrow\ \varnothing \vdash \mathsf{True} : \mathsf{Bool}}$$

$$\frac{}{\mathbb{W}(\mathsf{False}) \ \rightsquigarrow\ \varnothing \vdash \mathsf{False} : \mathsf{Bool}}$$

$$\frac{\mathtt{X}k \text{ es una incógnita fresca}}{\mathbb{W}(x) \ \rightsquigarrow\ x : \mathtt{X}k \vdash x : \mathtt{X}k}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\text{if } U_1 \text{ then } U_2 \text{ else } U_3) \rightsquigarrow}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\begin{array}{rcl} \mathbb{W}(U_1) & \leadsto & \Gamma_1 \vdash M_1 : \tau_1 \\ \mathbb{W}(U_2) & \leadsto & \Gamma_2 \vdash M_2 : \tau_2 \\ \mathbb{W}(U_3) & \leadsto & \Gamma_3 \vdash M_3 : \tau_3 \end{array}$$

$\mathbb{W}(\mathsf{if}\ U_1\ \mathsf{then}\ U_2\ \mathsf{else}\ U_3) \leadsto$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$
\frac{\begin{array}{rcl}\mathbb{W}(U_1) & \rightsquigarrow & \Gamma_1 \vdash M_1 : \tau_1 \\ \mathbb{W}(U_2) & \rightsquigarrow & \Gamma_2 \vdash M_2 : \tau_2 \\ \mathbb{W}(U_3) & \rightsquigarrow & \Gamma_3 \vdash M_3 : \tau_3 \\ \multicolumn{3}{c}{\mathbf{S} = \mathrm{mgu}(\{\tau_1 \stackrel{?}{=} \mathrm{Bool}, \tau_2 \stackrel{?}{=} \tau_3\})}\end{array}}{\mathbb{W}(\text{if } U_1 \text{ then } U_2 \text{ else } U_3) \rightsquigarrow}
$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$
\frac{\begin{array}{rcl}\mathbb{W}(U_1) & \rightsquigarrow & \Gamma_1 \vdash M_1 : \tau_1 \\ \mathbb{W}(U_2) & \rightsquigarrow & \Gamma_2 \vdash M_2 : \tau_2 \\ \mathbb{W}(U_3) & \rightsquigarrow & \Gamma_3 \vdash M_3 : \tau_3 \end{array} \qquad \mathbf{S} = \mathrm{mgu}\left(\begin{array}{l} \{\tau_1 \stackrel{?}{=} \mathsf{Bool}, \tau_2 \stackrel{?}{=} \tau_3\} \cup \\ \{\Gamma_i(x) \stackrel{?}{=} \Gamma_j(x) \mid i, j \in \{1,2,3\},\ x \in \Gamma_i \cap \Gamma_j\} \end{array} \right)}{\mathbb{W}(\mathsf{if}\ U_1\ \mathsf{then}\ U_2\ \mathsf{else}\ U_3) \rightsquigarrow}
$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\begin{array}{rcl} \mathbb{W}(U_1) & \rightsquigarrow & \Gamma_1 \vdash M_1 : \tau_1 \\ \mathbb{W}(U_2) & \rightsquigarrow & \Gamma_2 \vdash M_2 : \tau_2 \\ \mathbb{W}(U_3) & \rightsquigarrow & \Gamma_3 \vdash M_3 : \tau_3 \end{array}$$

$$\frac{\mathbf{S} = \mathrm{mgu}\left(\begin{array}{l} \{\tau_1 \stackrel{?}{=} \mathsf{Bool}, \tau_2 \stackrel{?}{=} \tau_3\} \cup \\ \{\Gamma_i(x) \stackrel{?}{=} \Gamma_j(x) \mid i, j \in \{1,2,3\},\ x \in \Gamma_i \cap \Gamma_j\} \end{array} \right)}{\begin{array}{l} \mathbb{W}(\mathsf{if}\ U_1\ \mathsf{then}\ U_2\ \mathsf{else}\ U_3) \ \rightsquigarrow\ \mathbf{S}(\Gamma_1) \cup \mathbf{S}(\Gamma_2) \cup \mathbf{S}(\Gamma_3) \vdash \\ \qquad\qquad \mathbf{S}(\mathsf{if}\ M_1\ \mathsf{then}\ M_2\ \mathsf{else}\ M_3) : \mathbf{S}(\tau_2) \end{array}}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(U\,V) \rightsquigarrow}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\begin{array}{rcl} \mathbb{W}(U) & \rightsquigarrow & \Gamma_1 \vdash M : \tau \\ \mathbb{W}(V) & \rightsquigarrow & \Gamma_2 \vdash N : \sigma \end{array}}{\mathbb{W}(U\,V) \rightsquigarrow}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\dfrac{\begin{array}{c} \mathbb{W}(U) \rightsquigarrow \Gamma_1 \vdash M : \tau \\ \mathbb{W}(V) \rightsquigarrow \Gamma_2 \vdash N : \sigma \\ \mathtt{X}k \text{ es una incógnita fresca} \\ \mathbf{S} = \mathrm{mgu}\{\tau \stackrel{?}{=} \sigma \to \mathtt{X}k\} \end{array}}{\mathbb{W}(U\,V) \rightsquigarrow}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\begin{array}{c} \mathbb{W}(U) \ \leadsto \ \Gamma_1 \vdash M : \tau \\ \mathbb{W}(V) \ \leadsto \ \Gamma_2 \vdash N : \sigma \\ \mathtt{X}k \text{ es una incógnita fresca} \\ \mathbf{S} = \mathrm{mgu}\{\tau \overset{?}{=} \sigma \to \mathtt{X}k\} \cup \{\Gamma_1(x) \overset{?}{=} \Gamma_2(x) : x \in \Gamma_1 \cap \Gamma_2\} \end{array}}{\mathbb{W}(U\,V) \ \leadsto}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$
\frac{
\begin{array}{c}
\mathbb{W}(U) \ \leadsto \ \Gamma_1 \vdash M : \tau \\
\mathbb{W}(V) \ \leadsto \ \Gamma_2 \vdash N : \sigma \\
\mathtt{X}k \text{ es una incógnita fresca} \\
\mathbf{S} = \mathrm{mgu}\{\tau \stackrel{?}{=} \sigma \to \mathtt{X}k\} \cup \{\Gamma_1(x) \stackrel{?}{=} \Gamma_2(x) : x \in \Gamma_1 \cap \Gamma_2\}
\end{array}
}{
\mathbb{W}(U\,V) \ \leadsto \ \mathbf{S}(\Gamma_1) \cup \mathbf{S}(\Gamma_2) \vdash \mathbf{S}(M\,N) : \mathbf{S}(\mathtt{X}k)
}
$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\lambda x.\, U) \;\rightsquigarrow}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\mathbb{W}(U) \;\rightsquigarrow\; \Gamma \vdash M : \tau}{\mathbb{W}(\lambda x.\, U) \;\rightsquigarrow\;}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\mathbb{W}(U) \;\rightsquigarrow\; \Gamma \vdash M : \tau}{\mathbb{W}(\lambda x.\, U) \;\rightsquigarrow\; \Gamma \qquad\quad \vdash \lambda x : \sigma.\, M : \sigma \to \tau}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\mathbb{W}(U) \;\rightsquigarrow\; \Gamma \vdash M : \tau}{\mathbb{W}(\lambda x.\, U) \;\rightsquigarrow\; \Gamma \ominus \{x\} \vdash \lambda x : \sigma.\ M : \sigma \to \tau}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\mathbb{W}(U) \;\rightsquigarrow\; \Gamma \vdash M : \tau}{\mathbb{W}(\lambda x.\, U) \;\rightsquigarrow\; \Gamma \ominus \{x\} \vdash \lambda x : \sigma.\ M : \sigma \to \tau}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\mathbb{W}(U) \ \leadsto\ \Gamma \vdash M : \tau \qquad \sigma = \begin{cases} \Gamma(x) & \text{si } x \in \Gamma \\ \text{una incógnita fresca } \mathtt{X}k & \text{si no} \end{cases}}{\mathbb{W}(\lambda x.\, U) \ \leadsto\ \Gamma \ominus \{x\} \vdash \lambda x : \sigma.\ M : \sigma \to \tau}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

Teorema (Corrección del algoritmo $\mathbb{W}$)

1. Si U no es tipable, $\mathbb{W}(U)$ falla al resolver alguna unificación.

Algoritmo $\mathbb{W}$ de inferencia de tipos

Teorema (Corrección del algoritmo $\mathbb{W}$)

1. Si U no es tipable, $\mathbb{W}(U)$ falla al resolver alguna unificación.
2. Si U es tipable, $\mathbb{W}(U) \rightsquigarrow \Gamma \vdash M : \tau$,
   donde $\mathtt{erase}(M) = U$ y $\Gamma \vdash M : \tau$ es un juicio válido.

Algoritmo $\mathbb{W}$ de inferencia de tipos

Teorema (Corrección del algoritmo $\mathbb{W}$)

1. Si U no es tipable, $\mathbb{W}(U)$ falla al resolver alguna unificación.
2. Si U es tipable, $\mathbb{W}(U) \rightsquigarrow \Gamma \vdash M : \tau$,
   donde $\mathtt{erase}(M) = U$ y $\Gamma \vdash M : \tau$ es un juicio válido.

   Además, $\Gamma \vdash M : \tau$ es el juicio de tipado más general posible.
   Más precisamente, si $\Gamma' \vdash M' : \tau'$ es un juicio válido y
   $\mathtt{erase}(M') = U$, existe una sustitución $\mathbf{S}$ tal que:

$$\begin{array}{rcl} \Gamma' & \supseteq & \mathbf{S}(\Gamma) \\ M' & = & \mathbf{S}(M) \\ \tau' & = & \mathbf{S}(\tau) \end{array}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

Ejercicio. Aplicar el algoritmo de inferencia sobre los siguientes términos:

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

¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ? ? ? ? ? ? ? ? ?