

Paradigmas de Programación

Unificación Inferencia de tipos

1er cuatrimestre de 2025

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Introducción

Algoritmo de unificación

Algoritmo de inferencia de tipos

Corrección del algoritmo de unificación

Problema de inferencia de tipos

Notación

Términos **sin** anotaciones de tipos:

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

Términos **con** anotaciones de tipos:

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

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

Ejemplo: $\text{erase}((\lambda x : \text{Bool}. x) \text{True}) = (\lambda x. x) \text{True}$.

Problema de 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 $\text{erase}(M) = U$ 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 $\text{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

Veremos un algoritmo para resolver este problema.

Problema de inferencia de tipos

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

Ejemplo — tipos parcialmente conocidos

- ▶ En $x \text{ True}$ sabemos que $x : \text{Bool} \rightarrow X_1$.
- ▶ En $\text{if } x \text{ y then True else False}$ sabemos que $x : X_2 \rightarrow \text{Bool}$.

Incorporamos *incógnitas* (X_1, X_2, X_3, \dots) a los tipos.

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

Ejemplo — ecuaciones entre tipos

- ▶ $(X_1 \rightarrow \text{Bool}) \stackrel{?}{=} ((\text{Bool} \rightarrow \text{Bool}) \rightarrow X_2)$
tiene solución: $X_1 := (\text{Bool} \rightarrow \text{Bool})$ y $X_2 := \text{Bool}$.
- ▶ $(X_1 \rightarrow X_1) \stackrel{?}{=} ((\text{Bool} \rightarrow \text{Bool}) \rightarrow X_2)$
tiene solución: $X_1 := (\text{Bool} \rightarrow \text{Bool})$ y $X_2 := (\text{Bool} \rightarrow \text{Bool})$.
- ▶ $(X_1 \rightarrow \text{Bool}) \stackrel{?}{=} X_1$
no tiene solución.

Introducción

Algoritmo de unificación

Algoritmo de inferencia de tipos

Corrección del algoritmo de unificación

Unificación

Suponemos fijado un conjunto finito de constructores de tipos:

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

Los tipos se forman usando incógnitas y constructores:

$$\tau ::= X_n \mid C(\tau_1, \dots, \tau_n)$$

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.

Notamos:

$$\{X_{k_1} := \tau_1, \dots, X_{k_n} := \tau_n\}$$

a la sustitución **S** tal que $S(X_{k_i}) = \tau_i$ para cada $1 \leq i \leq n$ y $S(X_k) = X_k$ para cualquier otra incógnita.

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

Ejemplo — aplicación de una sustitución a un tipo

Si $S = \{X_1 := \text{Bool}, X_3 := (X_2 \rightarrow X_2)\}$, entonces:

$$S((X_1 \rightarrow \text{Bool}) \rightarrow X_3) = ((\text{Bool} \rightarrow \text{Bool}) \rightarrow (X_2 \rightarrow X_2))$$

Unificación

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

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

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

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

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

...

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

Unificación

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

Ejemplo — problema de unificación con infinitas soluciones

$$\{X_1 \stackrel{?}{=} X_2\}$$

tiene infinitos unificadores:

- ▶ $\{X_1 := X_2\}$
- ▶ $\{X_2 := X_1\}$
- ▶ $\{X_1 := X_3, X_2 := X_3\}$
- ▶ $\{X_1 := \text{Bool}, X_2 := \text{Bool}\}$
- ▶ $\{X_1 := (\text{Bool} \rightarrow \text{Bool}), X_2 := (\text{Bool} \rightarrow \text{Bool})\}$
- ▶ ...

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:

$$E = \{(\mathbf{X}_1 \rightarrow \text{Bool}) \stackrel{?}{=} \mathbf{X}_2\}$$

las siguientes sustituciones son unificadores:

- ▶ $\mathbf{S}_1 = \{\mathbf{X}_1 := \text{Bool}, \mathbf{X}_2 := (\text{Bool} \rightarrow \text{Bool})\}$
- ▶ $\mathbf{S}_2 = \{\mathbf{X}_1 := \text{Int}, \mathbf{X}_2 := (\text{Int} \rightarrow \text{Bool})\}$
- ▶ $\mathbf{S}_3 = \{\mathbf{X}_1 := \mathbf{X}_3, \mathbf{X}_2 := (\mathbf{X}_3 \rightarrow \text{Bool})\}$
- ▶ $\mathbf{S}_4 = \{\mathbf{X}_2 := (\mathbf{X}_1 \rightarrow \text{Bool})\}$

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

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.
- ▶ De lo contrario, la regla es de la forma $E \rightarrow_{\mathbf{S}} E'$.
La resolución del problema E se reduce a resolver otro problema E' , aplicando la sustitución \mathbf{S} .

Hay dos posibilidades:

1. $E = E_0 \rightarrow_{\mathbf{S}_1} E_1 \rightarrow_{\mathbf{S}_2} E_2 \rightarrow \dots \rightarrow_{\mathbf{S}_n} E_n \rightarrow_{\mathbf{S}_{n+1}}$ falla
En tal caso el problema de unificación E no tiene solución.
2. $E = E_0 \rightarrow_{\mathbf{S}_1} E_1 \rightarrow_{\mathbf{S}_2} E_2 \rightarrow \dots \rightarrow_{\mathbf{S}_n} E_n = \emptyset$
En tal caso el problema de unificación E tiene solución.

Algoritmo de unificación de Martelli–Montanari

$$\{\mathbf{X}_n \stackrel{?}{=} \mathbf{X}_n\} \cup E \xrightarrow{\text{Delete}} E$$

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

$$\{\tau \stackrel{?}{=} \mathbf{X}_n\} \cup E \xrightarrow{\text{Swap}} \{\mathbf{X}_n \stackrel{?}{=} \tau\} \cup E$$

si τ no es una incógnita

$$\{\mathbf{X}_n \stackrel{?}{=} \tau\} \cup E \xrightarrow{\text{Elim}}_{\{\mathbf{X}_n := \tau\}} E' = \{\mathbf{X}_n := \tau\}(E)$$

si \mathbf{X}_n no ocurre en τ

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

si $C \neq C'$

$$\{\mathbf{X}_n \stackrel{?}{=} \tau\} \cup E \xrightarrow{\text{Occurs-Check}} \text{falla}$$

si $\mathbf{X}_n \neq \tau$
y \mathbf{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 .
2. Si E no tiene solución, el algoritmo llega a una falla.
3. Si E tiene solución, el algoritmo llega a \emptyset :

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

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

Además, dicho unificador es el *más general* posible.

(Salvo renombre de incógnitas).

Definición (Unificador más general)

Notamos $\text{mgu}(E)$ 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:

- ▶ $\{(X_2 \rightarrow (X_1 \rightarrow X_1)) \stackrel{?}{=} ((\text{Bool} \rightarrow \text{Bool}) \rightarrow (X_1 \rightarrow X_2))\}$
- ▶ $\{X_1 \stackrel{?}{=} (X_2 \rightarrow X_2), X_2 \stackrel{?}{=} (X_1 \rightarrow X_1)\}$

Introducción

Algoritmo de unificación

Algoritmo de inferencia de tipos

Corrección del algoritmo de unificación

Algoritmo \mathcal{I} — Inferencia de tipos

El algoritmo \mathcal{I} recibe un término U sin anotaciones de tipos.

Consta de los siguientes pasos:

1. **Rectificación** del término.
2. **Anotación** del término con variables de tipo frescas.
3. **Generación de restricciones** (ecuaciones entre tipos).
4. **Unificación de las restricciones**.

Algoritmo \mathcal{I} — Paso 1: rectificación

Decimos que un término está *rectificado* si:

1. No hay dos variables ligadas con el mismo nombre.
2. No hay una variable ligada con el mismo nombre que una variable libre.

Ejemplo – Términos rectificados

$x (\lambda x. x x) (\lambda y. y x)$	no está rectificado
$x (\lambda z. z z) (\lambda y. y x)$	está rectificado
$\lambda x. \lambda x. x y$	no está rectificado
$\lambda x. \lambda z. z y$	está rectificado

Observación

Siempre se puede rectificar un término α -renombrándolo.

Algoritmo \mathcal{I} — Paso 2: anotación

Tenemos un término U , que suponemos ya rectificado.

Producimos un contexto Γ_0 y un término M_0 :

1. El contexto Γ_0 le da tipo a todas las variables libres de U .
El tipo de cada variable es una incógnita *fresca*.
2. El término M_0 está anotado de tal modo que $\text{erase}(M_0) = U$.
Todas las anotaciones son incógnitas frescas.

Ejemplo – Anotación del término

Dado el término rectificado $U = (\lambda x. y \ x \ x) (\lambda z. w)$, producimos:

1. $\Gamma_0 = (y : \mathbf{X}_1, w : \mathbf{X}_2)$
2. $M_0 = (\lambda x : \mathbf{X}_3. y \ x \ x) (\lambda z : \mathbf{X}_4. w)$

Algoritmo \mathcal{I} — Paso 3: generación de las restricciones

Tenemos un contexto Γ y un término M con anotaciones de tipos.

Recursivamente calculamos:

1. Un tipo τ , que corresponde al tipo de M .
2. Un conjunto de ecuaciones E .

Representan restricciones para que M esté bien tipado.

Definimos un algoritmo recursivo:

$$\underbrace{\mathcal{I} \left(\underbrace{\Gamma}_{\text{contexto}} \mid \underbrace{M}_{\text{término}} \right)}_{\text{entrada}} = \underbrace{\left(\underbrace{\tau}_{\text{tipo}} \mid \underbrace{E}_{\text{restricciones}} \right)}_{\text{salida}}$$

con la precondition de que Γ le da tipo a todas las variables de M .

Algoritmo \mathcal{I} — Paso 3: generación de las restricciones

1. $\mathcal{I}(\Gamma \mid \text{True}) = (\text{Bool} \mid \emptyset)$
2. $\mathcal{I}(\Gamma \mid \text{False}) = (\text{Bool} \mid \emptyset)$
3. $\mathcal{I}(\Gamma \mid x) = (\tau \mid \emptyset)$ si $(x : \tau) \in \Gamma$
4. $\mathcal{I}(\Gamma \mid \text{if } M_1 \text{ then } M_2 \text{ else } M_3) =$
 $(\tau_2 \mid \{\tau_1 \stackrel{?}{=} \text{Bool}, \tau_2 \stackrel{?}{=} \tau_3\} \cup E_1 \cup E_2 \cup E_3)$
donde $\mathcal{I}(\Gamma \mid M_1) = (\tau_1 \mid E_1)$
 $\mathcal{I}(\Gamma \mid M_2) = (\tau_2 \mid E_2)$
 $\mathcal{I}(\Gamma \mid M_3) = (\tau_3 \mid E_3)$
5. $\mathcal{I}(\Gamma \mid M_1 M_2) = (\mathbf{x}_k \mid \{\tau_1 \stackrel{?}{=} (\tau_2 \rightarrow \mathbf{x}_k)\} \cup E_1 \cup E_2)$
donde \mathbf{x}_k es una incógnita fresca
 $\mathcal{I}(\Gamma \mid M_1) = (\tau_1 \mid E_1)$
 $\mathcal{I}(\Gamma \mid M_2) = (\tau_2 \mid E_2)$
6. $\mathcal{I}(\Gamma \mid \lambda x : \tau. M) = (\tau \rightarrow \sigma \mid E)$
donde $\mathcal{I}(\Gamma, x : \tau \mid M) = (\sigma \mid E)$

Algoritmo \mathcal{I} — Paso 4: unificación de las restricciones

Recordemos: Γ_0 y M_0 resultan de anotar un término rectificado U .

Una vez calculado $\mathcal{I}(\Gamma_0 \mid M_0) = (\tau \mid E)$:

1. Calculamos $\mathbf{S} = \text{mgu}(E)$.
2. Si no existe el unificador, el término U no es tipable.
3. Si existe el unificador, el término U es tipable y vale:

$$\mathbf{S}(\Gamma_0) \vdash \mathbf{S}(M_0) : \mathbf{S}(\tau)$$

Algoritmo \mathcal{I} — Corrección

Sean Γ_0 y M_0 el resultado de anotar un término rectificado U .
Supongamos que $\mathcal{I}(\Gamma_0 \mid M_0) = (\tau \mid E)$. Entonces:

Teorema (Corrección del algoritmo \mathcal{I})

1. Si U no es tipable, no hay unificador para E .
2. Si U es tipable, existe $\mathbf{S} = \text{mgu}(E)$.

Además, $\mathbf{S}(\Gamma_0) \vdash \mathbf{S}(M_0) : \mathbf{S}(\tau)$ es un juicio de tipado válido.

Más aún, el juicio de tipado es el más general posible para U .

Más precisamente, si $\Gamma' \vdash M' : \tau'$ es un juicio válido y $\text{erase}(M') = U$, existe una sustitución \mathbf{S}' tal que:

$$\begin{aligned}\Gamma' &\supseteq \mathbf{S}'(\Gamma_0) \\ M' &= \mathbf{S}'(M_0) \\ \tau' &= \mathbf{S}'(\tau)\end{aligned}$$

donde además \mathbf{S} es más general que \mathbf{S}' .

Algoritmo \mathcal{I} de inferencia de tipos

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

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

Introducción

Algoritmo de unificación

Algoritmo de inferencia de tipos

Corrección del algoritmo de unificación

Recordemos: algoritmo de unificación

$$\{x \stackrel{?}{=} x\} \cup E \xrightarrow{\text{Delete}} E$$

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

$$\{\tau \stackrel{?}{=} x\} \cup E \xrightarrow{\text{Swap}} \{x \stackrel{?}{=} \tau\} \cup E$$

si τ no es una variable

$$\{x \stackrel{?}{=} \tau\} \cup E \xrightarrow{\text{Elim}}_{\{x := \tau\}} E\{x := \tau\}$$

si $x \notin \tau$

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

si $C \neq C'$

$$\{x \stackrel{?}{=} \tau\} \cup E \xrightarrow{\text{Occurs-Check}} \text{falla}$$

si $x \neq \tau$ y $x \in \tau$

Terminación del algoritmo de unificación

Dado un conjunto de ecuaciones de unificación E , definimos:

- ▶ n_1 : cantidad de variables distintas en E
- ▶ n_2 : tamaño de E , calculado como $\sum_{(\tau \stackrel{?}{=} \sigma) \in E} |\tau| + |\sigma|$
- ▶ n_3 : cantidad de ecuaciones de la forma $\tau \stackrel{?}{=} x$ en E

Podemos observar que las reglas que no producen falla achican la tripla (n_1, n_2, n_3) , de acuerdo con el *orden lexicográfico*:

	n_1	n_2	n_3
Elim	>		
Decompose	=	>	
Delete	\geq	>	
Swap	=	=	>

Corrección del algoritmo de unificación

Recordemos

1. Una **sustitución** es una función **S** que le asocia un término **S**(*x*) a cada variable *x*.
2. **S** es un **unificador** de *E* si para cada $(\tau \stackrel{?}{=} \sigma) \in E$ se tiene que **S**(τ) = **S**(σ).
3. **S** es **más general** que **S'** si existe **T** tal que **S'** = **T** \circ **S**.
4. **S** es un **m.g.u.** de *E* si **S** es un unificador de *E* y para todo unificador **S'** de *E* se tiene que **S** es más general que **S'**.
Técnicamente, nos interesan los m.g.u. **idempotentes**, es decir **S**(**S**(τ)) = **S**(τ) para todo término τ .

Corrección del algoritmo de unificación

Lema — corrección de la regla Delete

\mathbf{S} m.g.u. de $E \implies \mathbf{S}$ m.g.u. de $\{x \stackrel{?}{=} x\} \cup E$.

Lema — corrección de la regla Swap

\mathbf{S} m.g.u. de $\{\tau \stackrel{?}{=} \sigma\} \cup E \implies \mathbf{S}$ m.g.u. de $\{\sigma \stackrel{?}{=} \tau\} \cup E$.

Lema — corrección de la regla Decompose

\mathbf{S} m.g.u. de $\{\tau_1 \stackrel{?}{=} \sigma_1, \dots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$
 $\implies \mathbf{S}$ m.g.u. de $\{C(\tau_1, \dots, \tau_n) \stackrel{?}{=} C(\sigma_1, \dots, \sigma_n)\} \cup E$.

Lema — corrección de la regla Elim

\mathbf{S} m.g.u. de $E\{x := \tau\}$ y $x \notin \tau$
 $\implies \mathbf{S} \circ \{x := \tau\}$ m.g.u. de E .

Usar el hecho de que si $\mathbf{S}(x) = \tau$ entonces $\mathbf{S}(\sigma\{x := \tau\}) = \mathbf{S}(\sigma)$.

Corrección del algoritmo de unificación

Probemos la corrección del algoritmo en caso de éxito.

Sea $E_0 \rightarrow_{\mathbf{s}_1} E_1 \rightarrow_{\mathbf{s}_n} E_2 \rightarrow \dots \rightarrow_{\mathbf{s}_n} E_n = \emptyset$.

Veamos que $\mathbf{S}_n \circ \dots \circ \mathbf{S}_1$ es un m.g.u. de E .

Por inducción en n :

1. Si $n = 0$, la sustitución identidad es un m.g.u. de \emptyset .
2. Si $n > 0$, se tiene:

$$E_0 \rightarrow_{\mathbf{s}_1} E_1 \quad E_1 \rightarrow_{\mathbf{s}_2} \dots \rightarrow_{\mathbf{s}_n} E_n = \emptyset$$

Por HI, $\mathbf{S}_n \circ \dots \circ \mathbf{S}_2$ es un m.g.u. de E_1 .

Aplicando alguno de los lemas anteriores, se concluye que

$\mathbf{S}_n \circ \dots \circ \mathbf{S}_2 \circ \mathbf{S}_1$ es un m.g.u. de E_0 .

Corrección del algoritmo de unificación

La corrección en caso de falla se prueba de manera similar, con lemas que van “hacia adelante” en lugar de “hacia atrás”.

ι ι ι ι ι ι ι ι ι ι ? ? ? ? ? ? ? ?

Lectura recomendada

Capítulo 22 del libro de Pierce.

Benjamin C. Pierce. *Types and Programming Languages*.
The MIT Press, 2002.

Extra: teoría detrás del método de unificación

Sección 4.5 del libro de Baader & Nipkow.

Franz Baader y Tobias Nipkow. *Term Rewriting and All That*.
Cambridge University Press, 1998.