



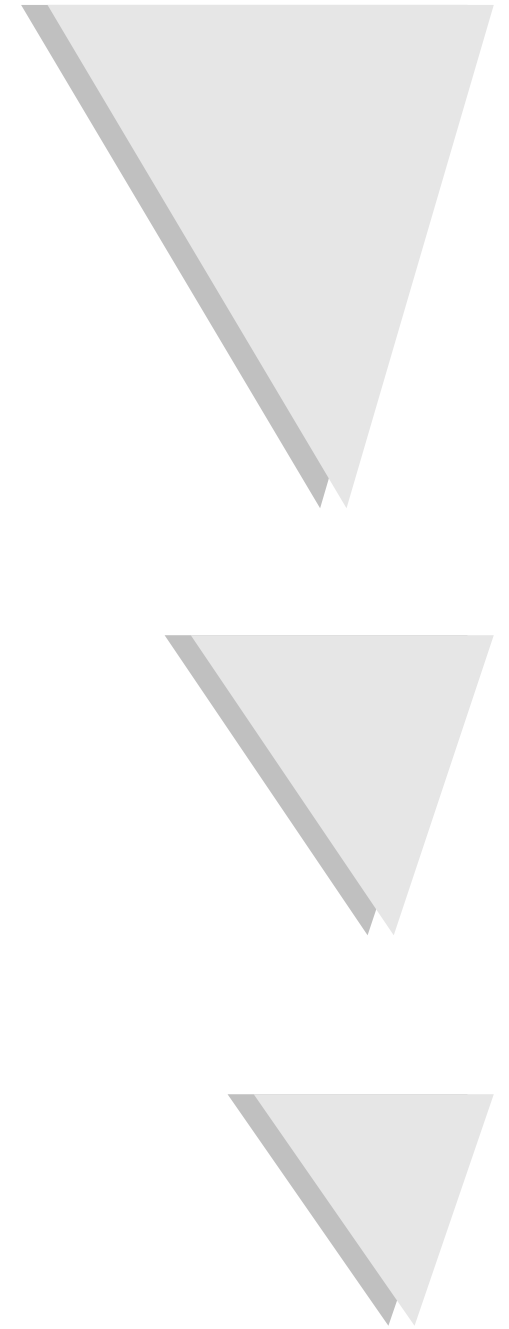
PROGRAMACIÓN FUNCIONAL

Modelo Funcional: Sistema de Tipos



Sistema de tipos

- ◆ Definición de tipo
- ◆ Inferencia y asignación de tipos.
- ◆ Ventajas y desventajas.
- ◆ Sistema de tipos a la Hindley-Milner.
- ◆ Polimorfismo.



Tipos

- ◆ ¿Cómo determinamos si una expresión está bien formada?
 - ◆ Reglas sintácticas
 - ◆ Reglas de asignación de tipo
- ◆ ¿Qué es un tipo?
 - ◆ Conjunto de valores con propiedades comunes
 - ◆ **Ejs:** enteros, valores de verdad, caracteres, funciones de enteros en enteros

Tipos

- ◆ Toda expresión válida denota un valor
- ◆ Todo valor pertenece a un conjunto
- ◆ Los tipos denotan conjuntos
- ◆ Entonces...

TODA EXPRESIÓN DEBE TENER UN
TIPO PARA SER VÁLIDA

(si una expresión no tiene tipo, es inválida)

- ◆ TIPADO FUERTE (STRONG TYPING)

Asignación de Tipos

◆ Notación: $e :: A$

◆ se lee "la expresión e tiene tipo A "

◆ significa que el valor denotado por e pertenece al conjunto de valores denotado por A

◆ Ejemplos:

$2 :: \text{Int}$

$\text{False} :: \text{Bool}$

$'a' :: \text{Char}$

$\text{doble} :: \text{Int} \rightarrow \text{Int}$

$[\text{sqr}, \text{doble}] :: [\text{Int} \rightarrow \text{Int}]$

Asignación de tipos

- ◆ Se puede deducir el tipo de una expresión a partir de su constitución
- ◆ Algunas reglas
 - ◆ si $e1 :: A$ y $e2 :: B$, entonces $(e1, e2) :: (A, B)$
 - ◆ si $m, n :: \text{Int}$, entonces $(m+n) :: \text{Int}$
 - ◆ si $f :: A \rightarrow B$ y $e :: A$, entonces $f\ e :: B$
 - ◆ si $d = e$ y $e :: A$, entonces $d :: A$

Asignación de tipos

- ◆ Ejemplo: $\text{doble } x = x + x$
 $\text{twice}' (f, y) = f (f y)$
 - ◆ $x + x :: \text{Int}$, y entonces sólo puede ser que $x :: \text{Int}$
 - ◆ $\text{doble } x :: \text{Int}$ y $x :: \text{Int}$, entonces sólo puede ser que $\text{doble} :: \text{Int} \rightarrow \text{Int}$
 - ◆ si $y :: A$ y $f :: A \rightarrow A$, entonces $f y :: A$, $f (f y) :: A$
 - ◆ como $\text{twice}' (f, y) :: A$, y $(f, y) :: (A \rightarrow A, A)$, sólo puede ser que $\text{twice}' :: (A \rightarrow A, A) \rightarrow A$

Asignación de tipos

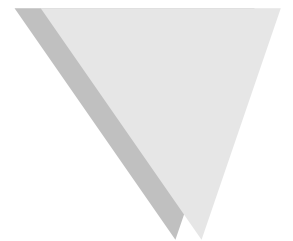
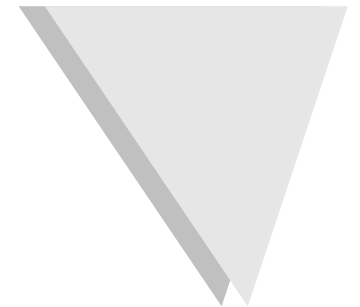
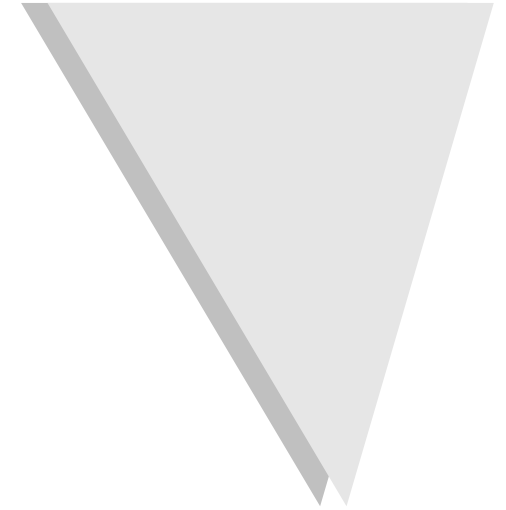
- ◆ Propiedades deseables
 - ◆ que sea automática (que haya un programa)
 - ◆ que le dé tipo al mayor número posible de expresiones con sentido
 - ◆ que no le dé tipo al mayor número posible de expresiones sin sentido
 - ◆ que se conserve por reducción
 - ◆ que los tipos sean descriptivos y razonablemente sencillos de leer

Asignación de tipos

- ◆ Inferencia de tipos
 - ◆ dada una expresión e , determinar si tiene tipo o no según las reglas, y cuál es ese tipo
- ◆ Chequeo de tipos
 - ◆ dada una expresión e y un tipo A , determinar si $e :: A$ según las reglas, o no
- ◆ Sistema de tipado fuerte (strong typing)
 - ◆ sistema que acepta una expresión si, y sólo si ésta tiene tipo según las reglas

Sistema de tipos

- ◆ ¿Para qué sirven los tipos?
 - ◆ detección de errores comunes
 - ◆ documentación
 - ◆ especificación rudimentaria
 - ◆ oportunidades de optimización en compilación
- ◆ Es una buena práctica en programación empezar dando el tipo del programa que se quiere escribir.



Sistema Hindley-Milner

◆ Tipos básicos

- ◆ enteros Int
- ◆ caracteres Char
- ◆ booleanos Bool

◆ Tipos compuestos

- ◆ tuplas (A,B)
- ◆ listas [A]
- ◆ funciones (A->B)

◆ Polimorfismo

Polimorfismo

- ◆ ¿Qué tipo tendrá la siguiente función?

`id :: ??`

`id x = x`

`(id 3) :: Int`

`(id 'a') :: Char`

`(id True) :: Bool` `(id doble) :: Int -> Int`

- ◆ ¿Es una expresión con sentido?

- ◆ ¿Debería tener un tipo?

- ◆ En realidad:

`id :: A -> A`, cualquiera sea `A`

Polimorfismo paramétrico

- ◆ Solución: ¡variables de tipo!

$\text{id} :: a \rightarrow a$

se lee: "id es una función que dado un elemento de *algún tipo* a , retorna un elemento de ese mismo tipo"

- ◆ La identidad es una función *polimórfica*
 - ◆ el tipo de su argumento puede ser *instanciado* de diferentes maneras en diferentes usos

$(\text{id } 3) :: \text{Int}$

y aquí

$\text{id} :: \text{Int} \rightarrow \text{Int}$

$(\text{id } \text{True}) :: \text{Bool}$

y aquí

$\text{id} :: \text{Bool} \rightarrow \text{Bool}$

Polimorfismo paramétrico

◆ Polimorfismo

- ◆ Característica del sistema de tipos
- ◆ Dada una expresión que puede ser tipada de infinitas maneras, el sistema puede asignarle un tipo que sea más general que todos ellos, y tal que en cada uso pueda transformarse en uno particular.

◆ Ej: $\text{id} :: a \rightarrow a$ ← más general ← particulares
 $\text{id} :: \text{Int} \rightarrow \text{Int}$ $\text{id} :: [\text{Int}] \rightarrow [\text{Int}] \dots$

- ◆ Reemplazando a por Int , por ejemplo, se obtiene un tipo particular
- ◆ Se llama “paramétrico” pues a es el *parámetro*.

Polimorfismo paramétrico

• ¿Tienen tipo las siguientes expresiones?

¿Cuáles?

(Recordar: $\text{twice } f = g \text{ where } g \ x = f (f \ x)$)

`twice :: ??`

`(id doble) (id 3) :: ??`

`(id twice') (id doble, id 3) :: ??`

`(id id) (id doble) :: ??`

`id id :: ??`

`twice id :: ??`

Resumen

- ◆ Inferencia y asignación de tipos.
- ◆ Sistema de tipado fuerte.
- ◆ Sistema Hindley-Milner.
- ◆ Polimorfismo.

