

# Esquemas de recursión sobre listas: FoldR

```
foldr :: (a -> b -> b) -> b -> [a] -> b
```

La función `foldr` nos permite realizar recursión estructural sobre una lista.

O, hablando en francés, la función `foldr`

- Toma una función que representa el paso recursivo y un valor que representa el caso base,
- Y nos devuelve una función que sabe como reducir listas de **a** a un valor **b**.

```
foldr f z [] = z
foldr f z (x:xs) = f x (foldr f z xs)
```

## Definir utilizando foldr

- `longitud :: [a] -> Int`
- `concatenar :: [[a]] -> [a]`
- `suma :: [Int] -> Int`

## Esquemas de recursión sobre listas: FoldL

La función `foldl` es muy similar a `foldr` pero *acumula desde la izquierda*. Se define de la siguiente forma:

### FoldL

```
foldl :: (b -> a -> b) -> b -> [a] -> b
foldl f z [] = z
foldl f z (x : xs) = foldl f (f z x) xs
```

# Esquemas de recursión sobre listas: FoldR, FoldL y las listas infinitas

¿Qué sucede con las listas infinitas al usar `foldr` o `foldl`?

## Usando foldr

```
suma [1..]
---> foldr (+) 0 [1..]
---> 1 + (foldr (+) 0 [2..])
---> 1 + (2 + (foldr (+) 0 [3..]))
---> 1 + (2 + (3 + (foldr (+) 0 [4..])))
```

## Usando foldl

```
suma [1..]
---> foldl (+) 0 [1..]
---> foldl (+) (0 + 1) [2..]
---> foldl (+) ((0 + 1) + 2) [3..]
---> foldl (+) (((0 + 1) + 2) + 3) [4..]
```

## Otros esquemas de recursión sobre listas: Divide & Conquer

La técnica de Divide & Conquer consiste en dividir un problema en problemas más fáciles de resolver y luego combinando los resultados parciales, lograr obtener un resultado general.

Para generalizar la técnica, crearemos el tipo `DivideConquer` definido como:

<code>type DivideConquer a b</code>	
= ( <code>a</code> → <code>Bool</code> )	– determina si es o no el caso trivial
→ ( <code>a</code> → <code>b</code> )	– resuelve el caso trivial
→ ( <code>a</code> → [ <code>a</code> ])	– parte el problema en sub-problemas
→ ([ <code>b</code> ] → <code>b</code> )	– combina resultados
→ <code>a</code>	– input
→ <code>b</code>	– resultado

# Machete: Tipos y Términos

Las expresiones de tipos (o simplemente tipos) son

$$\sigma ::= \text{Bool} \mid \text{Nat} \mid \sigma \rightarrow \rho$$

Sea  $\mathcal{X}$  un conjunto infinito enumerable de variables y  $x \in \mathcal{X}$ . Los términos están dados por

$$\begin{aligned} M ::= & \quad x \\ | & \quad \text{true} \\ | & \quad \text{false} \\ | & \quad \text{if } M \text{ then } M \text{ else } M \\ | & \quad \lambda x : \sigma. \, M \\ | & \quad M \, M \\ | & \quad 0 \\ | & \quad \text{succ}(M) \\ | & \quad \text{pred}(M) \\ | & \quad \text{iszzero}(M) \end{aligned}$$

# Machete: Axiomas y reglas de tipado

$$\frac{}{\Gamma \vdash \text{true} : \text{Bool}} \text{ (T-TRUE)} \quad \frac{}{\Gamma \vdash \text{false} : \text{Bool}} \text{ (T-FALSE)}$$

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

$$\frac{\Gamma \vdash M : \text{Bool} \quad \Gamma \vdash P : \sigma \quad \Gamma \vdash Q : \sigma}{\Gamma \vdash \text{if } M \text{ then } P \text{ else } Q : \sigma} \text{ (T-IF)}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma. M : \sigma \rightarrow \tau} \text{ (T-ABS)} \quad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \text{ (T-APP)}$$

# Machete: Axiomas y reglas de tipado

$$\frac{}{\Gamma \vdash 0 : Nat} \text{ (T-ZERO)}$$

$$\frac{\Gamma \vdash M : Nat}{\Gamma \vdash \text{succ}(M) : Nat} \text{ (T-SUCC)}$$

$$\frac{\Gamma \vdash M : Nat}{\Gamma \vdash \text{pred}(M) : Nat} \text{ (T-PRED)}$$

$$\frac{\Gamma \vdash M : Nat}{\Gamma \vdash \text{iszero}(M) : Bool} \text{ (T-IsZERO)}$$

# Machete: Semántica operacional

$V ::= \text{true} | \text{false} | \lambda x : \sigma. M | \underline{n}$   
donde  $\underline{n}$  abrevia  $\text{succ}^n(0)$ .

## Reglas de Evaluación en un paso

$$\frac{M_1 \rightarrow M'_1}{M_1 M_2 \rightarrow M'_1 M_2} (\text{E-APP1 o } \mu)$$

$$\frac{M_2 \rightarrow M'_2}{V_1 M_2 \rightarrow V_1 M'_2} (\text{E-APP2 o } \nu)$$

$$\frac{}{(\lambda x : \sigma. M) V \rightarrow M\{x \leftarrow V\}} (\text{E-APPABS o } \beta)$$

# Machete: Semántica operacional

$V ::= \text{true} | \text{false} | \lambda x : \sigma. M | \underline{n}$   
donde  $\underline{n}$  abrevia  $\text{succ}^n(0)$ .

## Reglas de Evaluación en un paso

$$\frac{}{\text{if } \text{true} \text{ then } M_2 \text{ else } M_3 \rightarrow M_2} (\text{E-IFTRUE})$$

$$\frac{}{\text{if } \text{false} \text{ then } M_2 \text{ else } M_3 \rightarrow M_3} (\text{E-IFFALSE})$$

$$\frac{M_1 \rightarrow M'_1}{\text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rightarrow \text{if } M'_1 \text{ then } M_2 \text{ else } M_3} (\text{E-IF})$$

# Machete: Semántica operacional

## Reglas de Evaluación en un paso

$$\frac{M_1 \rightarrow M'_1}{\text{succ}(M_1) \rightarrow \text{succ}(M'_1)} \quad (\text{E-SUCC})$$

$$\frac{}{\text{pred}(0) \rightarrow 0} \quad (\text{E-PREDZERO}) \qquad \frac{}{\text{pred}(\text{succ}(\underline{n})) \rightarrow \underline{n}} \quad (\text{E-PREDSUCC})$$

$$\frac{M_1 \rightarrow M'_1}{\text{pred}(M_1) \rightarrow \text{pred}(M'_1)} \quad (\text{E-PRED})$$

$$\frac{}{\text{iszzero}(0) \rightarrow \text{true}} \quad (\text{E-ISZEROZERO}) \qquad \frac{}{\text{iszzero}(\text{succ}(\underline{n})) \rightarrow \text{false}} \quad (\text{E-ISZEROSUCC})$$

$$\frac{M_1 \rightarrow M'_1}{\text{iszzero}(M_1) \rightarrow \text{iszzero}(M'_1)} \quad (\text{E-ISZERO})$$

# Sintaxis para cálculo λ con pares

¿Qué hay que agregar?

- ...términos para representar el constructor y los observadores

$$M ::= \dots | \langle M, N \rangle | \pi_1(M) | \pi_2(M)$$

- ...y un tipo para estas nuevas expresiones

$$\sigma ::= \dots | \sigma \times \tau$$

# Reglas de tipado para pares

¿Qué hay que agregar?

- Al menos una regla por cada forma nueva de sintaxis, porque cada una de ellas precisa poder ser tipada.
- Notar que, de no hacerlo, sería imposible construir términos tipables (útiles) con dicha forma.

# Regla de tipado para el constructor

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : \tau}{\Gamma \triangleright \langle M, N \rangle : \sigma \times \tau}$$

# Reglas de tipado para las proyecciones

$$\frac{\Gamma \triangleright M : \sigma \times \tau}{\Gamma \triangleright \pi_1(M) : \sigma}$$

$$\frac{\Gamma \triangleright N : \sigma \times \tau}{\Gamma \triangleright \pi_2(N) : \tau}$$

## Semántica para pares

¿Qué reglas hay que agregar?

- Necesitamos reducir todos los pares *con sentido* que no sean valores.

¿Cuáles son los valores?

- Empecemos por ahí entonces...

# Extensión de los valores

$V ::= \dots | \langle V, W \rangle$

# Reglas de semántica para pares

Ahora sí, las reglas

$$\frac{M \rightarrow M'}{< M, N > \rightarrow < M', N >} \qquad \frac{N \rightarrow N'}{< V, N > \rightarrow < V, N' >}$$

## Reglas de semántica para las proyecciones

$$\frac{M \rightarrow M'}{\pi_1(M) \rightarrow \pi_1(M')}$$

$$\frac{M \rightarrow M'}{\pi_2(M) \rightarrow \pi_2(M')}$$

$$\pi_1(< \textcolor{red}{V}, \textcolor{red}{W} >) \rightarrow \textcolor{red}{V}$$

$$\pi_2(< \textcolor{red}{V}, \textcolor{red}{W} >) \rightarrow \textcolor{red}{W}$$

# Sintaxis para cálculo $\lambda$ con árboles binarios

¿Qué hay que agregar?

- ...términos para representar los constructores y observadores  
 $M ::= \dots \mid Nil_\sigma \mid Bin(M, N, O) \mid root(M) \mid right(M) \mid left(M) \mid isNil(M)$
- ...y un tipo para estas nuevas expresiones

$$\sigma ::= \dots \mid AB_\sigma$$

# Reglas de tipado para los constructores

$$\frac{}{\Gamma \triangleright Nil_\sigma : AB_\sigma}$$

$$\frac{\Gamma \triangleright M : AB_\sigma \quad \Gamma \triangleright O : AB_\sigma \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright Bin(M, N, O) : AB_\sigma}$$

- $Nil_\sigma$  es una constante diferente según el tipo  $\sigma$ .
  - ¡No tenemos polimorfismo!
- Para  $Bin$ , en cambio, el tipo queda determinado por el tipo de los subterminos.

# Reglas de tipado para los observadores

$$\frac{\Gamma \triangleright M : AB_\sigma}{\Gamma \triangleright \text{root}(M) : \sigma}$$

$$\frac{\Gamma \triangleright M : AB_\sigma}{\Gamma \triangleright \text{isNil}(M) : \text{Bool}}$$

$$\frac{\Gamma \triangleright M : AB_\sigma}{\Gamma \triangleright \text{left}(M) : AB_\sigma}$$

$$\frac{\Gamma \triangleright M : AB_\sigma}{\Gamma \triangleright \text{right}(M) : AB_\sigma}$$

# Semántica para árboles binarios

- Primero, empecemos por los valores:

$$V ::= \dots \mid Nil_\sigma \mid Bin(V, W, Y)$$

# Reglas de semántica para los constructores

$$\frac{M \rightarrow M'}{\text{Bin}(M, N, O) \rightarrow \text{Bin}(M', N, O)}$$

$$\frac{N \rightarrow N'}{\text{Bin}(\textcolor{red}{V}, N, O) \rightarrow \text{Bin}(\textcolor{red}{V}, N', O)}$$

$$\frac{O \rightarrow O'}{\text{Bin}(\textcolor{red}{V}, \textcolor{red}{W}, O) \rightarrow \text{Bin}(\textcolor{red}{V}, \textcolor{red}{W}, O')}$$

## Reglas de semántica para los observadores (1/2)

$$\frac{M \rightarrow M'}{\text{left}(M) \rightarrow \text{left}(M')}$$

$$\frac{M \rightarrow M'}{\text{right}(M) \rightarrow \text{right}(M')}$$

$$\frac{M \rightarrow M'}{\text{root}(M) \rightarrow \text{root}(M')}$$

$$\frac{M \rightarrow M'}{\text{isNil}(M) \rightarrow \text{isNil}(M')}$$

## Reglas de semántica para los observadores (2/2)

$$\overline{isNil(Nil_{\sigma}) \rightarrow true}$$

$$\overline{isNil(Bin(V, W, Y)) \rightarrow false}$$

$$\overline{left(Bin(V, W, Y)) \rightarrow V}$$

$$\overline{right(Bin(V, W, Y)) \rightarrow Y}$$

$$\overline{root(Bin(V, W, Y)) \rightarrow W}$$

# Sintaxis para cálculo $\lambda$ con árboles binarios bis

- Los tipos quedan igual que en el caso anterior:

$$\sigma ::= \dots \mid AB_\sigma$$

- Y los términos,

$$M ::= \dots \mid Nil_\sigma \mid Bin(M, N, O) \mid$$

$$Case_{AB_\sigma} M \ of \ Nil \rightsquigarrow N ; Bin(m, n, o) \rightsquigarrow O$$

Aquí las minúsculas ( $m, n, o$ ) representan **variables**.

# Reglas de tipado para árboles binarios bis

- Para los constructores son las que ya teníamos.

$$\overline{\Gamma \triangleright Nil_\sigma : AB_\sigma}$$

$$\frac{\Gamma \triangleright M : AB_\sigma \quad \Gamma \triangleright O : AB_\sigma \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright Bin(M, N, O) : AB_\sigma}$$

# Regla de tipado para el Case

$$\frac{\Gamma \triangleright M : AB_\sigma \quad \Gamma \triangleright N : \tau}{\Gamma \cup \{m : AB_\sigma, n : \sigma, o : AB_\sigma\} \triangleright O : \tau}$$

---

$$\Gamma \triangleright \text{Case}_{AB_\sigma} M \text{ of } Nil \rightsquigarrow N ; \text{Bin}(m, n, o) \rightsquigarrow O : \tau$$

## Semántica para los árboles binarios bis

- Tenemos los mismos valores que antes:

$$V ::= \dots \mid Nil_\sigma \mid Bin(V, W, Y)$$

# Reglas de semántica para los constructores

- Análogas a las que ya teníamos.

$$\frac{M \rightarrow M'}{\text{Bin}(M, N, O) \rightarrow \text{Bin}(M', N, O)}$$

$$\frac{N \rightarrow N'}{\text{Bin}(\textcolor{red}{V}, N, O) \rightarrow \text{Bin}(\textcolor{red}{V}, N', O)}$$

$$\frac{O \rightarrow O'}{\text{Bin}(\textcolor{red}{V}, \textcolor{red}{W}, O) \rightarrow \text{Bin}(\textcolor{red}{V}, \textcolor{red}{W}, O')}$$

# Reglas de semántica para el Case

$$\frac{M \rightarrow M'}{\text{Case}_{AB_\sigma} M \text{ of } \text{Nil} \rightsquigarrow N ; \text{Bin}(m, n, o) \rightsquigarrow O}$$

$\rightarrow$

$$\text{Case}_{AB_\sigma} M' \text{ of } \text{Nil} \rightsquigarrow N ; \text{Bin}(m, n, o) \rightsquigarrow O$$

---

$$\text{Case}_{AB_\sigma} \text{Nil}_\sigma \text{ of } \text{Nil} \rightsquigarrow N ; \text{Bin}(m, n, o) \rightsquigarrow O \rightarrow N$$

---

$$\begin{aligned} & \text{Case}_{AB_\sigma} \text{Bin}(\textcolor{red}{V}, \textcolor{red}{W}, \textcolor{red}{Y}) \text{ of } \text{Nil} \rightsquigarrow N ; \text{Bin}(m, n, o) \rightsquigarrow O \\ & \rightarrow O\{m \leftarrow \textcolor{red}{V}, n \leftarrow \textcolor{red}{W}, o \leftarrow \textcolor{red}{Y}\} \end{aligned}$$

# 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 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. Falla

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

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

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

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

## Ejemplo de secuencia exitosa

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

- El MGU es

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

## Ejemplo de secuencia fallida

$\vdash^1 \{r \rightarrow (s \rightarrow r) \doteq s \rightarrow ((r \rightarrow \text{Nat}) \rightarrow r)\}$   
 $\vdash^{1,4}_{s/r} \{r \doteq s, s \rightarrow r \doteq (r \rightarrow \text{Nat}) \rightarrow r\}$   
 $\vdash^{4,1}_{s/r} \{s \rightarrow s \doteq (s \rightarrow \text{Nat}) \rightarrow s\}$   
 $\vdash^1 \{s \doteq s \rightarrow \text{Nat}, s \doteq s\}$   
 $\vdash^6 \text{falla}$

Inferencia

Unificación

Algoritmo de inferencia

Algoritmo de inferencia

Ejemplos

# Algoritmo de inferencia (caso constantes y variables)

$$\begin{aligned} \mathbb{W}(0) &\stackrel{\text{def}}{=} \emptyset \triangleright 0 : \text{Nat} \\ \mathbb{W}(\text{true}) &\stackrel{\text{def}}{=} \emptyset \triangleright \text{true} : \text{Bool} \\ \mathbb{W}(\text{false}) &\stackrel{\text{def}}{=} \emptyset \triangleright \text{false} : \text{Bool} \\ \mathbb{W}(x) &\stackrel{\text{def}}{=} \{x : s\} \triangleright x : s, \quad s \text{ variable fresca} \end{aligned}$$

## Algoritmo de inferencia (caso *succ*)

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

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

- ▶ 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

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

## 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

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

► Entonces

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

# Algoritmo de inferencia (caso aplicación)

► Sea

$$\begin{array}{lcl} \triangleright \mathbb{W}(U) & = & \Gamma_1 \triangleright M : \tau \\ \triangleright \mathbb{W}(V) & = & \Gamma_2 \triangleright N : \rho \end{array}$$

► Sea

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

► 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

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

## Ejemplo (2/4)

*if true then succ(x y) else x (succ(y))*

$$\mathbb{W}(x) = \{x : s\} \triangleright x : s$$

$$\mathbb{W}(y) = \{y : t\} \triangleright y : t$$

$$\mathbb{W}(xy) = \{x : t \rightarrow r, y : t\} \triangleright xy : r$$

donde  $S = MGU(\{s \doteq t \rightarrow r\}) = \{t \rightarrow r/s\}$

$$\mathbb{W}(succ(xy)) = \{x : t \rightarrow \text{Nat}, y : t\} \triangleright \text{succ}(xy) : \text{Nat}$$

donde  $S = MGU(\{r \doteq \text{Nat}\}) = \{\text{Nat}/r\}$

## Ejemplo (3/4)

*if true then succ(x y) else x (succ(y))*

$$\begin{aligned}\mathbb{W}(y) &= \{y : v\} \triangleright y : v \\ \mathbb{W}(\text{succ}(y)) &= \{y : \text{Nat}\} \triangleright \text{succ}(y) : \text{Nat} \\ &\quad \text{donde } MGU(\{v \doteq \text{Nat}\}) = \{\text{Nat}/v\} \\ \mathbb{W}(x) &= \{x : u\} \triangleright x : u \\ \mathbb{W}(x \text{succ}(y)) &= \{x : \text{Nat} \rightarrow w, y : \text{Nat}\} \triangleright x \text{succ}(y) : w \\ &\quad \text{donde } MGU(\{u \doteq \text{Nat} \rightarrow w\}) = \{\text{Nat} \rightarrow w/u\}\end{aligned}$$

## Ejemplo (4/4)

$$M = \text{if } \text{true} \text{ then } \text{succ}(x\ y) \text{ else } x\ (\text{succ}(y))$$

- ▶  $\mathbb{W}(\text{true}) = \emptyset \triangleright \text{true} : \text{Bool}$
- ▶  $\mathbb{W}(\text{succ}(x\ y)) = \{x : t \rightarrow \text{Nat}, y : t\} \triangleright \text{succ}(x\ y) : \text{Nat}$
- ▶  $\mathbb{W}(x\ \text{succ}(y)) = \{x : \text{Nat} \rightarrow w, y : \text{Nat}\} \triangleright x\ \text{succ}(y) : w$

$$\mathbb{W}(M) = \{x : \text{Nat} \rightarrow \text{Nat}, y : \text{Nat}\} \triangleright M : \text{Nat}$$

donde  $S = MGU(\{t \rightarrow \text{Nat} \doteq \text{Nat} \rightarrow w, t \doteq \text{Nat}, \text{Nat} \doteq w\}) = \{\text{Nat}/t, \text{Nat}/w\}$

# Un ejemplo de falla

$M = \text{if } \text{true} \text{ then } x \underline{2} \text{ else } x \text{ true}$

$$\mathbb{W}(x) = \{x : s\} \triangleright x : s$$

$$\mathbb{W}(\underline{2}) = \emptyset \triangleright \underline{2} : Nat$$

$$\mathbb{W}(x \underline{2}) = \{x : Nat \rightarrow t\} \triangleright x \underline{2} : t$$

$$\mathbb{W}(x) = \{x : u\} \triangleright x : u$$

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

$$\mathbb{W}(x \text{ true}) = \{x : Bool \rightarrow v\} \triangleright x \text{ true} : v$$

$$\mathbb{W}(M) = \text{falla}$$

no existe el  $MGU(\{Nat \rightarrow t \doteq Bool \rightarrow v\})$