

## 1. Funciones de haskell

```

foldr, foldl :: Foldable t => (a -> b -> b) -> b -> t a -> b
foldr1, foldl1 :: Foldable t => (a -> a -> a) -> t a -> a

map :: (a -> b) -> [a] -> [b]
zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]

(+++) :: [a] -> [a] -> [a]
(!!) :: [a] -> Int -> a
head, last :: [a] -> a
init, tail :: [a] -> [a]
length :: Foldable t => t a -> Int

reverse :: [a] -> [a]
concat :: Foldable t => t [a] -> [a]
union :: Eq a => [a] -> [a] -> [a]

all, any :: Foldable t => (a -> Bool) -> t a -> Bool

null :: Foldable t => t a -> Bool -- Es vacío
elem :: (Eq a, Foldable t) => a -> t a -> Bool

nub :: Eq a => [a] -> [a] -- Elimina duplicados
sort :: Ord a => [a] -> [a] -- Ordena la lista

concatMap :: Foldable t => (a -> [b]) -> t a -> [b]
find :: (a -> Bool) -> [a] -> Maybe a
filter :: (a -> Bool) -> [a] -> [a]
iterate :: (a -> a) -> a -> [a]
span :: (a -> Bool) -> [a] -> ([a], [a])
replicate :: Int -> a -> [a]
take, drop :: Int -> [a] -> [a]
takeWhile, dropWhile :: (a -> Bool) -> [a] -> [a]

and, or :: Foldable t => t Bool -> Bool
maximum, minimum :: (Ord a, Foldable t) => t a -> a
sum :: (Num a, Foldable t) => t a -> a

max, min :: Ord a => a -> a -> a
rem :: Integral a => a -> a -> a
ord :: Char -> Int
chr :: Int -> Char

fromJust :: Maybe a -> a
isNothing :: Maybe a -> Bool
lookup :: Eq a => a -> [(a, b)] -> Maybe b
maybe :: b -> (a -> b) -> Maybe a -> b

```

## 2. Esquemas de recursión

```

map :: (a -> b) -> [a] -> [b]
map _ [] = []
map f (x:xs) = (f x) : (map f xs)

filter :: (a -> Bool) -> [a] -> [a]
filter _ [] = []
filter p (x:xs) | (p x)      = x : (filter p xs)
                 | otherwise = filter p xs

foldr :: (a -> b -> b) -> b -> [a] -> b
foldr _ z [] = z
foldr f z (x:xs) = f x (foldr f z xs)

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

recr :: b -> (a -> [a] -> b -> b) -> [a] -> b
recr z _ [] = z
recr z f (x:xs) = f x xs (recr z f xs)

type DivideConquer a b = (a -> Bool) -> (a -> b) -> (a -> [a]) ->
                           ([b] -> b) -> a -> b

divideConquerListas :: DivideConquer [a] b
-- Esto significa que DivideConquerLista es de tipo
-- ([a] -> Bool) -> ([a] -> b) -> ([a] -> [[a]]) -> ([b] -> b)
-- -> [a] -> b

divideConquerListas esTrivial resolver repartir combinar l =
    if (esTrivial l) then resolver l
    else combinar (map dc (repartir l))
where dc = divideConquerListas esTrivial resolver repartir combinar

```

### 3. Cálculo lambda $\lambda^{bn}$

#### Función Free Variables

$$\begin{aligned}
 FV(x) &\stackrel{def}{=} x \\
 FV(true) = FV(false) &\stackrel{def}{=} \emptyset \\
 FV(if\ M\ then\ P\ else\ Q) &\stackrel{def}{=} FV(M) \cup FV(P) \cup FV(Q) \\
 FV(M\ N) &\stackrel{def}{=} FV(M) \cup FV(N) \\
 FV(\lambda x : \sigma. M) &\stackrel{def}{=} FV(M) \setminus \{x\}
 \end{aligned}$$

#### Sustitución

$$\begin{aligned}
 x\{x \leftarrow N\} &\stackrel{def}{=} N \\
 a\{x \leftarrow N\} &\stackrel{def}{=} a \text{ si } a \in \{true, false\} \cup \mathcal{X} \setminus \{x\} \\
 (if\ M\ then\ P\ else\ Q)\{x \leftarrow N\} &\stackrel{def}{=} if\ M\{x \leftarrow N\}\ then\ P\{x \leftarrow N\}\ else\ Q\{x \leftarrow N\} \\
 (M_1\ M_2)\{x \leftarrow N\} &\stackrel{def}{=} M_1\{x \leftarrow N\}\ M_2\{x \leftarrow N\} \\
 (\lambda y : \sigma. M)\{x \leftarrow N\} &\stackrel{def}{=} \lambda y : \sigma. M\{x \leftarrow N\} \quad x \neq y, \ y \notin FV(N)
 \end{aligned}$$

#### 3.0.1. Tipos

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

#### Expresiones

$$\begin{aligned}
 M, P, Q ::= & true \mid false \mid if\ M\ then\ P\ else\ Q \\
 & \mid M\ N \mid \lambda x : \sigma. M \\
 & \mid x \mid 0 \mid succ(M) \mid pred(M) \mid isZero(M)
 \end{aligned} \tag{1}$$

#### 3.0.2. Reglas de tipado

$$\begin{aligned}
 &\overline{\Gamma \triangleright true : Bool}^{(T-True)} && \overline{\Gamma \triangleright false : Bool}^{(T-False)} \\
 &\frac{x : \sigma \in \Gamma}{\Gamma \triangleright x : \sigma}^{(T-Var)} && \frac{\Gamma \triangleright M : Bool \quad \Gamma \triangleright P : \sigma \quad \Gamma \triangleright Q : \sigma}{\Gamma \triangleright if\ M\ then\ P\ else\ Q : \sigma}^{(T-If)} \\
 &\frac{\Gamma, x : \sigma \triangleright M : \tau}{\Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau}^{(T-Abs)} && \frac{\Gamma \triangleright M : \sigma \rightarrow \tau \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M\ N : \tau}^{(T-App)}
 \end{aligned}$$

$$\frac{}{\Gamma \triangleright 0 : Nat} (T\text{-Zero})$$

$$\frac{\Gamma \triangleright M : Nat}{\Gamma \triangleright succ(M) : Nat} (T\text{-Succ}) \quad \frac{\Gamma \triangleright M : Nat}{\Gamma \triangleright pred(M) : Nat} (T\text{-Pred})$$

$$\frac{\Gamma \triangleright M : Nat}{\Gamma \triangleright isZero(M) : Bool} (T\text{-IsZero})$$

### Valores

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

### Reglas de semánticas

$$\frac{}{if \textcolor{red}{true} then M_1 else M_2 \rightarrow M_1} (E\text{-IfTrue})$$

$$\frac{}{if \textcolor{red}{false} then M_1 else M_2 \rightarrow M_2} (E\text{-IfFalse})$$

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

$$\frac{M_1 \rightarrow M'_1}{M_1 M_2 \rightarrow M'_1 M_2} (E\text{-App1} / \mu) \quad \frac{M_2 \rightarrow M'_2}{\textcolor{red}{V}_1 M_2 \rightarrow \textcolor{red}{V}_1 M'_2} (E\text{-App2} / v)$$

$$\frac{}{(\lambda x : \sigma. M) \textcolor{red}{V} \rightarrow M\{x \leftarrow \textcolor{red}{V}\}} (E\text{-App3} / \beta)$$

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

$$\frac{}{pred(0) \rightarrow 0} (E\text{-PredZero}) \quad \frac{}{pred(succ(\underline{n})) \rightarrow \underline{n}} (E\text{-PredSucc})$$

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

$$\frac{}{isZero(0) \rightarrow true} \text{(E-IsZeroZero)} \qquad \frac{}{isZero(succ(\underline{n})) \rightarrow false} \text{(E-isZeroSucc)}$$

$$\frac{M_1 \rightarrow M'_1}{isZero(M_1) \rightarrow isZero(M'_1)} \text{(E-isZero)}$$

## 4. Extensión con memoria $\lambda^{bnu}$

### Tipos

$$\sigma, \tau ::= \text{Bool} \mid \text{Nat} \mid \text{Unit} \mid \text{Ref } \sigma \mid \sigma \rightarrow \tau$$

### Términos

$$M ::= \dots \mid \text{unit} \mid \text{ref } M \mid !M \mid M := N \mid l$$

### Axiomas y reglas de tipado

$$\frac{}{\Gamma \mid \Sigma \triangleright \text{unit} : \text{Unit}} (\text{T-Unit}) \quad \frac{\Gamma \mid \Sigma \triangleright M_1 : \sigma}{\Gamma \mid \Sigma \triangleright \text{ref } M_1 : \text{Ref } \sigma} (\text{T-Ref})$$

$$\frac{\Gamma \mid \Sigma \triangleright M_1 : \text{Ref } \sigma}{\Gamma \triangleright !M_1 : \sigma} (\text{T-DeRef})$$

$$\frac{\Gamma \mid \Sigma \triangleright M_1 : \text{Ref } \sigma \quad \Gamma \mid \Sigma \triangleright M_2 : \sigma}{\Gamma \triangleright M_1 := M_2 : \text{Unit}} (\text{T-Assing})$$

$$\frac{\Sigma(l) = \sigma}{\Gamma \mid \text{Signa} \triangleright l : \text{Ref } \sigma} (\text{T-Loc})$$

### Valores

$$V ::= \dots \mid \text{unit} \mid l$$

### Axiomas y reglas semánticas

$$\frac{M_1 \mid \mu \rightarrow M'_1 \mid \mu'}{M_1 \mid M_2 \mid \mu \rightarrow M'_1 \mid M_2 \mid \mu'} (\text{E-App1}) \quad \frac{M_2 \mid \mu \rightarrow M'_2 \mid \mu'}{\textcolor{red}{V}_1 \mid M_2 \mid \mu \rightarrow \textcolor{red}{V}_1 \mid M'_2 \mid \mu'} (\text{E-App2})$$

$$\frac{}{(\lambda x : \sigma. M) \mid \textcolor{red}{V} \mid \mu \rightarrow M \{x \leftarrow \textcolor{red}{V}\} \mid \mu'} (\text{E-AppAbs})$$

$$\frac{M_1 \mid \mu \rightarrow M'_1 \mid \mu'}{!M_1 \mid \mu \rightarrow !M'_1 \mid \mu'} (\text{E-DeRef}) \quad \frac{\mu(l) = \textcolor{red}{V}}{!l \mid \mu \rightarrow \textcolor{red}{V} \mid \mu} (\text{E-DerefLoc})$$

$$\frac{M_1 \mid \mu \rightarrow M'_1 \mid \mu'}{M_1 := M_2 \mid \mu \rightarrow M'_1 := M_2 \mid \mu'} (\text{E-Assign1})$$

$$\frac{M_2 \mid \mu \rightarrow M'_2 \mid \mu'}{\textcolor{red}{V} := M_2 \mid \mu \rightarrow \textcolor{red}{V} := M'_2 \mid \mu'} (\text{E-Assign2})$$

$$\frac{}{l := \textcolor{red}{V} | \mu \rightarrow \textit{unit} | \mu [l \rightarrow \textcolor{red}{V}]} \text{(E-Assign)}$$

$$\frac{M_1 | \mu \rightarrow M'_1 | \mu'}{\textit{ref } M_1 | \mu \rightarrow \textit{ref } M'_1 | \mu'} \text{(E-Ref)}$$

$$\frac{l \notin \textit{Dom}(\mu)}{\textit{ref } \textcolor{red}{V} | \mu \rightarrow l | \mu \oplus (l \rightarrow \textcolor{red}{V})} \text{(E-RefV)}$$

## 5. Extensión con recursión $\lambda^{\cdot r}$

**Términos**

$$M ::= \dots \mid \text{fix } M$$

**Regla de tipado**

$$\frac{\Gamma \triangleright M : \sigma \rightarrow \sigma}{\Gamma \triangleright \text{fix } M : \sigma} (\text{T-Fix})$$

**Reglas de evaluación**

$$\frac{M_1 \rightarrow M'_1}{\text{fix } M_1 \rightarrow \text{fix } M'_1} (\text{E-Fix})$$

$$\frac{}{\text{fix } (\lambda x : \sigma. M) \rightarrow M\{x \leftarrow \text{fix } \lambda x : \sigma. M\}} (\text{E-FixBeta})$$

## 6. Extensión con Declaraciones Locales ( $\lambda^{\dots let}$ )

Con esta extensión, agregamos al lenguaje el término  $\text{let } x : \sigma = M \text{ in } N$ , que evalúa  $M$  a un valor, liga  $x$  a  $V$  y, luego, evalúa  $N$ . Este término solo mejora la legibilidad de los programas que ya podemos definir con el lenguaje hasta ahora definido.

**Términos**

$$M ::= \dots \mid \text{let } x : \sigma = M \text{ in } N$$

**Axiomas y reglas de tipado**

$$\frac{\Gamma \triangleright M : \sigma_1 \quad \Gamma, x : \sigma_1 \triangleright N : \sigma_2}{\Gamma \triangleright \text{let } x : \sigma_1 = M \text{ in } N : \sigma_2} (\text{T-Let})$$

**Axiomas y reglas de evaluación**

$$\frac{M_1 \rightarrow M'_1}{\text{let } x : \sigma = M_1 \text{ in } M_2 \rightarrow \text{let } x : \sigma = M'_1 \text{ in } M_2} (\text{E-Let})$$

$$\frac{}{\text{let } x : \sigma = \mathbf{V_1} \text{ in } M_2 \rightarrow M_2\{x \leftarrow \mathbf{V_1}\}} (\text{E-LetV})$$

### 6.0.1. Construcción $\text{let}$ recursivo (Letrec)

Una construcción alternativa para definir funciones recursivas es

$$\text{letrec } f : \sigma \rightarrow \sigma = \lambda x : \sigma. M \text{ in } N$$

Y  $\text{letRec}$  se puede definir en base a  $\text{let}$  y  $\text{fix}$  (definido en ??) de la siguiente forma:

$$\text{let } f : \sigma \rightarrow \sigma = (\text{fix } \lambda f : \sigma \rightarrow \sigma. \lambda x : \sigma. M) \text{ in } N$$



## 7. Extensión con Registros $\lambda^{\dots r}$

### Tipos

$$\sigma, \tau ::= \dots \mid \{l_i : \sigma_i \mid i \in 1..n\}$$

El tipo  $\{l_i : \sigma_i \mid i \in 1..n\}$  representan las estructuras con  $n$  atributos tipados, por ejemplo:  $\{\text{nombre} : \text{String}, \text{edad} : \text{Nat}\}$

### Términos

$$M ::= \dots \mid \{l_i = M_i \mid i \in 1..n\} \mid M.l$$

Los términos significan:

- El registro  $\{l_i = M_i \mid i \in 1..n\}$  evalúa  $\{l_i = V_i \mid i \in 1..n\}$  donde  $V_i$  es el valores al que evalúa  $M_i$  para  $i \in 1..n$ .
- $M.l$ : Proyecta el valor de la etiqueta  $l$  del registro  $M$

### Axiomas y reglas de tipado

$$\frac{\Gamma \triangleright M_i : \sigma_i \text{ para cada } i \in 1..n}{\Gamma \triangleright \{l_i = M_i \mid i \in 1..n\} : \{l_i : \sigma_i \mid i \in 1..n\}} \text{(T-RCD)}$$

$$\frac{\Gamma \triangleright M : \{l_i : \sigma_i \mid i \in 1..n\}}{\Gamma \triangleright M.l_j : \sigma_j} \text{(T-Proj)}$$

### Valores

$$V ::= \dots \mid \{l_i = V_i \mid i \in 1..n\}$$

### Axiomas y reglas de evaluación

$$\frac{j \in 1..n}{\{l_i = V_i \mid i \in 1..n\}.l_j \rightarrow V_j} \text{(E-ProjRcd)}$$

$$\frac{M \rightarrow M'}{M.l \rightarrow M'.l} \text{(E-Proj)}$$

$$\frac{M_j \rightarrow M'_j}{\{l_i = V_i \mid i \in 1..j-1, l_j = M_j, l_i = M_i \mid i \in j+1..n\} \rightarrow \{l_i = V_i \mid i \in 1..j-1, l_j = M'_j, l_i = M_i \mid i \in j+1..n\}} \text{(E-RCD)}$$

## 8. Extensión con tuplas

### Tipos

$$\sigma, \tau ::= \dots \mid \sigma \times \tau$$

### Términos

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

### Axiomas y reglas de tipado

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright N : \tau}{\Gamma \triangleright \langle M, N \rangle : \sigma \times \tau} (\text{T-Tupla})$$

$$\frac{\Gamma \triangleright M : \sigma \times \tau}{\Gamma \triangleright \pi_1(M) : \sigma} (\text{T-}\pi_1) \qquad \frac{\Gamma \triangleright M : \sigma \times \tau}{\Gamma \triangleright \pi_2(M) : \tau} (\text{T-}\pi_2)$$

### Valores

$$V ::= \dots \mid \langle V, V \rangle$$

### Axiomas y reglas de evaluación

$$\frac{M \rightarrow M'}{\langle M, N \rangle \rightarrow \langle M', N \rangle} (\text{E-Tuplas}) \qquad \frac{N \rightarrow N'}{\langle \textcolor{red}{V}, N \rangle \rightarrow \langle \textcolor{red}{V}, N' \rangle} (\text{E-Tuplas1})$$

$$\frac{M \rightarrow M'}{\pi_1(M) \rightarrow \pi_1(M')} (\text{E-}\pi_1) \qquad \frac{}{\pi_1(\langle \textcolor{red}{V}_1, \textcolor{red}{V}_2 \rangle) \rightarrow \textcolor{red}{V}_1} (\text{E-}\pi'_1)$$

$$\frac{M \rightarrow M'}{\pi_2(M) \rightarrow \pi_2(M')} (\text{E-}\pi_2) \qquad \frac{}{\pi_2(\langle \textcolor{red}{V}_1, \textcolor{red}{V}_2 \rangle) \rightarrow \textcolor{red}{V}_2} (\text{E-}\pi'_2)$$

## 9. Extensión con árboles binarios

### Tipos

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

### Términos

$$M, N ::= \dots \mid \text{Nil}_\sigma \mid \text{Bin}(M, N, O) \mid \text{raiz}(M) \mid \text{der}(M) \mid \text{izq}(M) \mid \text{esNil}(M)$$

### Axiomas y reglas de tipado

$$\frac{}{\Gamma \triangleright \text{Nil}_\sigma : AB_\sigma} (\text{T-Nil}) \quad \frac{\Gamma \triangleright M : AB_\sigma \quad \Gamma \triangleright N : \sigma \quad \Gamma \triangleright O : AB_\sigma}{\Gamma \triangleright \text{Bin}(M, N, O) : AB_\sigma} (\text{T-Bin})$$

$$\frac{\Gamma \triangleright M : AB_\sigma}{\Gamma \triangleright \text{raiz}(M) : \sigma} (\text{T-raiz}) \quad \frac{\Gamma \triangleright M : AB_\sigma}{\Gamma \triangleright \text{der}(M) : AB_\sigma} (\text{T-der})$$

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

### Valores

$$V ::= \dots \mid \text{Nil} \mid \text{Bin}(V, V, V)$$

### Axiomas y reglas de evaluación

$$\frac{M \rightarrow M'}{\text{Bin}(M, N, O) \rightarrow \text{Bin}(M', N, O)} (\text{E-Bin1}) \quad \frac{N \rightarrow N'}{\text{Bin}(V, N, O) \rightarrow \text{Bin}(V, N', O)} (\text{E-Bin2})$$

$$\frac{O \rightarrow O'}{\text{Bin}(V_1, V_2, O) \rightarrow \text{Bin}(V_1, V_2, O')} (\text{E-Bin3})$$

$$\frac{M \rightarrow M'}{\text{raiz}(M) \rightarrow \text{raiz}(M')} (\text{E-Raiz1}) \quad \frac{}{\text{raiz}(\text{Bin}(V_1, V_2, V_3)) \rightarrow V_2} (\text{E-Bin3})$$

$$\frac{M \rightarrow M'}{\text{der}(M) \rightarrow \text{der}(M')} (\text{E-Der1}) \quad \frac{}{\text{der}(\text{Bin}(V_1, V_2, V_3)) \rightarrow V_3} (\text{E-Der2})$$

$$\frac{M \rightarrow M'}{\text{izq}(M) \rightarrow \text{izq}(M')} (\text{E-Izq1}) \quad \frac{}{\text{izq}(\text{Bin}(V_1, V_2, V_3)) \rightarrow V_1} (\text{E-Izq2})$$

$$\frac{}{\text{isNil}(M) \rightarrow \text{izq}(M')} (\text{E-isNil1}) \quad \frac{}{\text{isNil}(\text{Bin}(V_1, V_2, V_3)) \rightarrow \text{false}} (\text{E-isNilBin})$$

$$\frac{}{\text{isNil}(\text{Bin}(V_1, V_2, V_3)) \rightarrow \text{true}} (\text{E-isNilNil})$$

## 10. Algoritmo de Martelli-Montanari

### 1. Descomposición

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

### 2. Eliminación de par trivial

$$\{Nat \dot{=} Nat\} \cup G \mapsto G$$

$$\{Bool \dot{=} Bool\} \cup G \mapsto G$$

$$\{s \dot{=} s\} \cup G \mapsto G$$

### 3. Swap Si $\sigma$ no es una variable,

$$\{\sigma \dot{=} s\} \cup G \mapsto \{s \dot{=} \sigma\} \cup G$$

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

$$\{s \dot{=} \sigma\} \cup G \mapsto_{\sigma/s} G[\sigma/s]$$

### 5. Falla

$\{\sigma \dot{=} \tau\} \cup G \mapsto \text{falla}$ , con  $(\sigma, \tau) \in T \cup T^{-1}$  y  $T = \{(Bool, Nat), (Nat, \sigma_1 \rightarrow \sigma_2), (Bool, \sigma_1 \rightarrow \sigma_2)\}$ .  
 Aquí, la notación  $T^{-1}$  se refiere al conjunto con cada tupla de  $T$  invertida.

### 6. Occur Check Si $s \neq \sigma$ y $s \in FV(\sigma)$

$$\{s \dot{=} \sigma\} \cup G \mapsto \text{falla}$$

## 11. Función $\mathbb{W}$

### Constantes y variables

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

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

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

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

### Caso *succ*

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

$$\blacksquare \mathbb{W}(U) = \Gamma \triangleright M : \tau$$

$$\blacksquare S = MGU\{\tau \dot{=} Nat\}$$

### Caso *pred*

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

$$\blacksquare \mathbb{W}(U) = \Gamma \triangleright M : \tau$$

$$\blacksquare S = MGU\{\tau \dot{=} Nat\}$$

### Caso *isZero*

$$\mathbb{W}(\text{isZero}(U)) \stackrel{def}{=} S\Gamma \triangleright S \text{ isZero}(M) : Bool$$

$$\blacksquare \mathbb{W}(U) = \Gamma \triangleright M : \tau$$

$$\blacksquare S = MGU\{\tau \dot{=} Nat\}$$

**Caso *ifThenElse***

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

- $\mathbb{W}(U) = \Gamma_1 \triangleright M : \rho$
- $\mathbb{W}(V) = \Gamma_2 \triangleright P : \sigma$
- $\mathbb{W}(W) = \Gamma_3 \triangleright Q : \tau$
- $S = MGU\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_i \wedge x : \sigma_2 \in \Gamma_j, i \neq j\} \cup \{\sigma \doteq \tau \mid \rho \doteq Bool\}$

**Caso aplicación**

$$\mathbb{W}(U \ V) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \triangleright S \textit{ (M N) : St}$$

- $\mathbb{W}(U) = \Gamma_1 \triangleright M : \tau$
- $\mathbb{W}(V) = \Gamma_2 \triangleright N : \rho$
- $S = MGU\{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_i \wedge x : \sigma_2 \in \Gamma_j, i \neq j\} \cup \{\tau \doteq \rho \rightarrow t\}$  con  $t$  variable fresca

**Caso abstracción**

Sea  $\mathbb{W}(U) = \Gamma \triangleright M : \rho$ , si  $\Gamma$  tiene información de tipos para  $x$ , es decir  $x : \tau \in \Gamma$  para algún  $\tau$ , entonces:

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

Si  $\Gamma$  no tiene información de tipos para  $x$  ( $x \notin \text{Dom}(\Gamma)$ ), entonces elegimos una variable fresca  $s$  y

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

**Caso *fix***

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

- $\mathbb{W}(U) = \Gamma_1 \triangleright M : \tau$
- $S = MGU\{\tau \doteq t \rightarrow t\}$  con  $t$  variable fresca

## 12. Subtipado

$$\frac{}{Nat <: Float} (S\text{-NatFloat}) \quad \frac{}{Int <: Float} (S\text{-IntFloat}) \quad \frac{}{Bool <: Nat} (S\text{-BoolNat})$$

$$\frac{\sigma' <: \sigma \quad \tau <: \tau'}{\sigma \rightarrow \tau <: \sigma' \rightarrow \tau'} (S\text{-Func})$$

$$\frac{}{\sigma <: \sigma} (S\text{-Refl}) \quad \frac{\sigma <: \tau \quad \tau <: \rho}{\sigma <: \rho} (S\text{-Trans})$$

$$\frac{\sigma <: \tau \quad \tau <: \sigma}{Ref \ \tau <: Ref \ \sigma}$$

$$\frac{\sigma <: \tau}{Source \ \sigma <: Source \ \tau} (S\text{-Source}) \quad \frac{\tau <: \sigma}{Sink \ \sigma <: Sink \ \tau} (S\text{-Sink})$$

$$\frac{}{Ref \ \tau <: Source \ \tau} (S\text{-RefSource}) \quad \frac{}{Ref \ \tau <: Sink \ \tau} (S\text{-RefSink})$$

### 12.1. Reglas de reduccion con subtipado

$$\frac{x : \sigma \in \Gamma}{\Gamma \mapsto x : \sigma} (T\text{-Var})$$

$$\frac{\Gamma, x : \sigma \mapsto M : \tau}{\Gamma \mapsto \lambda x : \sigma. M : \sigma \rightarrow \tau} (T\text{-Abs}) \quad \frac{\Gamma \mapsto M : \sigma \rightarrow \tau \quad \Gamma \mapsto N : \rho \quad \rho <: \sigma}{\Gamma \mapsto M \ N : \tau} (T\text{-App})$$

## 13. Objetos

### 13.0.1. Sintaxis

$a, b ::=$	$x$	Variables
	$  [l_i = \varsigma(x_i)b_i^{i \in 1..n}]$	Objetos
	$  a.l$	Selección/ Envío de mensajes
	$  a.l \Leftarrow \varsigma(x)b$	Redefinición de un método.

### 13.1. Variables libres

$\text{fv}(\varsigma(x)b)$	$= \text{fv}(b) \setminus \{x\}$
$\text{fv}(x)$	$= \{x\}$
$\text{fv}([l_i = \varsigma(x_i)b_i^{i \in 1..n}])$	$= \bigcup_{i \in 1..n} \text{fv}(\varsigma(x_i)b_i)$
$\text{fv}(a.l)$	$= \text{fv}(a)$
$\text{fv}(a.l \Leftarrow \varsigma(x)b)$	$= \text{fv}(a.l) \cup \text{fv}(\varsigma(x)b)$

### 13.2. Sustitución

$x\{x \leftarrow c\}$	$= c$	
$y\{x \leftarrow c\}$	$= y$	si $x \neq y$
$([l_i = \varsigma(x_i)b_i^{i \in 1..n}])\{x \leftarrow c\}$	$= [l_i = (\varsigma(x_i)b_i)\{x \leftarrow c\}^{i \in 1..n}]$	
$(a.l)\{x \leftarrow c\}$	$= (a\{x \leftarrow c\}).l$	
$(a.l \Leftarrow \varsigma(x)b)\{x \leftarrow c\}$	$= (a\{x \leftarrow c\}).l \Leftarrow (\varsigma(x)b)\{x \leftarrow c\}$	
$(\varsigma(y)b)\{x \leftarrow c\}$	$= (\varsigma(y')(b\{y \leftarrow y'\}\{x \leftarrow c\}))$	si $y' \notin \text{fv}(\varsigma(y)b) \cup \text{fv}(c) \cup \{x\}$

### 13.3. Semantica operacional

$$V ::= [l_i = \varsigma(x_i)b_i^{i \in 1..n}]$$

$$\frac{}{v \longrightarrow v} [\text{Obj}]$$

$$\frac{a \longrightarrow v' \quad v' \equiv [l_i = \varsigma(x_i)b_i^{i \in 1..n}] \quad b_j\{x_j \leftarrow v'\} \longrightarrow v \quad j \in 1..n}{a.l_j \longrightarrow v} [\text{Sel}]$$

$$\frac{a \longrightarrow [l_i = \varsigma(x_i)b_i^{i \in 1..n}] \quad j \in 1..n}{a.l_j \Leftarrow \varsigma(x)b \longrightarrow [l_j = \varsigma(x)b, l_i = \varsigma(x_i)b_i^{i \in 1..n - \{j\}}]} [\text{Upd}]$$

**Indefinido:**  $[a = \varsigma(x)x.a].a$

**13.3.1. Codificación de funciones**

$$\begin{aligned}
[[x]] &\stackrel{def}{=} x \\
[[M \ N]] &\stackrel{def}{=} [[M]].arg := [[N]] \\
[[\lambda x.M]] &\stackrel{def}{=} [val = \varsigma(y)[[M]]\{x \leftarrow y.arg\}, \ arg = \varsigma(y)y.arg]
\end{aligned}$$



## 14. Resolución

### 14.1. Lógica propocisional

$$\frac{C_1 = \{A_1, \dots, A_m, L\} \quad C_2 = \{B_1, \dots, B_m, \bar{L}\}}{C = \{A_1, \dots, A_m, B_1, \dots, B_n\}}$$

### 14.2. Lógica de primer orden

Transformar la formula:

1. Eliminar las implicaciones, es decir, si aparece una clausula de la forma  $(A \supset B)$ , reescribirla como  $(\neg A \vee B)$ .
2. Pasar a **forma normal negada**.
3. Pasar a **forma normal prenexa**.
4. Pasar a **forma normal de Skolem**.
5. Pasar a **forma normal conjuntiva**.
6. **Distribuir** cuantificadores universales.

#### 14.2.1. Skolemización

Sea  $A$  una sentencia rectificada en forma normal negada, la **forma normal de Skolem de A** ( $\mathbf{SK}(A)$ ) se define recursivamente como sigue:

Sea  $A'$  cualquier subfórmula de  $A$ ,

- Si  $A'$  es una fórmula atómica o su negación,  $\mathbf{SK}(A') = A'$ .
- Si  $A'$  es de la forma  $(B \star C)$  con  $\star \in \{\wedge, \vee\}$ , entonces  $\mathbf{SK}(A') = (\mathbf{SK}(B) \star \mathbf{SK}(C))$ .
- Si  $A'$  es de la forma  $\forall x.B$ , entonces  $\mathbf{SK}(A') = \forall x.\mathbf{SK}(B)$ .
- Si  $A'$  es de la forma  $\exists x.B$  y  $\{x, y_1, \dots, y_m\}$  son las variables libres de  $B$ , entonces:
  1. Si  $m > 0$ , crear un **símbolo de función de Skolem**,  $f_x$  de aridad  $m$  y definir:

$$\mathbf{SK}(A') = \mathbf{SK}(B\{x \leftarrow f(y_1, \dots, y_m)\})$$

2. Si  $m = 0$ , crear una nueva **constante de Skolem**  $c_x$  y

$$\mathbf{SK}(A') = \mathbf{SK}(B\{x \leftarrow c_x\})$$

### 14.3. Reglas de resolucion de primer orden

$$\frac{\{B_1, \dots, B_k, A_1, \dots, A_n\} \quad \{\neg D_1, \dots, \neg D_k, A_1, \dots, A_n\}}{\sigma(\{A_1, \dots, A_m, C_1, \dots, C_n\})}$$

donde  $\sigma$  es el **unificador más general** (MGU) de  $\{B_1, \dots, B_k, \neg D_1, \dots, \neg D_k\}$  y  $\sigma(\{A_1, \dots, A_m, C_1, \dots, C_n\})$  es el **resolvente**.

## 14.4. Regla de resolución binaria y factorización

$$\frac{\frac{\{B_1, A_1, \dots, A_n\} \quad \{-D_1, A_1, \dots, A_n\}}{\sigma(\{A_1, \dots, A_m, C_1, \dots, C_n\})}}{\frac{\{B_1, \dots, B_k, A_1, \dots, A_n\}}{\sigma(\{B_1, A_1, \dots, A_m\})}}$$

## 15. Prolog predicados

**Predicados:** =, sort, msort, length, nth1, nth0, member, append, last, between, is\_list, list\_to\_set, is\_set, union, intersection, subset, subtract, select, delete, reverse, atom, number, numlist, sumlist, flatten, help

**Operaciones extra-lógicas :** is, \ =, ==, ==, = \ =, >, <, =<, >=, abs, max, min, gcd, var, nonvar, ground, trace, notrace

**Metapredicados:** bagof, setof, maplist, include, not, forall, assert, retract, listing