

# Taller de Inferencia de Tipos

## Machete

Paradigmas de Lenguajes de Programación

### 1. Algoritmo de inferencia

- $\mathbb{W}(x) \stackrel{\text{def}}{=} \{x : s\} \triangleright x : s$ ,  $s$  variable fresca
- $\mathbb{W}(\theta) \stackrel{\text{def}}{=} \emptyset \triangleright \theta : \text{nat}$
- $\mathbb{W}(\text{true}) \stackrel{\text{def}}{=} \emptyset \triangleright \text{true} : \text{bool}$
- $\mathbb{W}(\text{false}) \stackrel{\text{def}}{=} \emptyset \triangleright \text{false} : \text{bool}$
- $\mathbb{W}(\text{succ}(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S \text{succ}(M) : \text{nat}$  donde
  - $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
  - $S = \text{MGU}\{\tau \doteq \text{nat}\}$
- $\mathbb{W}(\text{pred}(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S \text{pred}(M) : \text{nat}$  donde
  - $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
  - $S = \text{MGU}\{\tau \doteq \text{nat}\}$
- $\mathbb{W}(\text{iszero}(U)) \stackrel{\text{def}}{=} S\Gamma \triangleright S \text{iszero}(M) : \text{bool}$  donde
  - $\mathbb{W}(U) = \Gamma \triangleright M : \tau$
  - $S = \text{MGU}\{\tau \doteq \text{nat}\}$
- $\mathbb{W}(\text{if } U \text{ then } V \text{ else } W) \stackrel{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3 \triangleright S(\text{if } M \text{ then } P \text{ else } Q) : S\sigma$  donde
  - $\mathbb{W}(U) = \Gamma_1 \triangleright M : \rho$
  - $\mathbb{W}(V) = \Gamma_2 \triangleright P : \sigma$
  - $\mathbb{W}(W) = \Gamma_3 \triangleright Q : \tau$
  - $S = \text{MGU}\{\sigma \doteq \tau, \rho \doteq \text{bool}\} \cup \{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_i, x : \sigma_2 \in \Gamma_j, i \neq j\}$
- $\mathbb{W}(\lambda x.U) \stackrel{\text{def}}{=} \Gamma' \triangleright \lambda x : \tau'. M : \tau' \rightarrow \rho$  donde
  - $\mathbb{W}(U) = \Gamma \triangleright M : \rho$
  - $\tau' = \begin{cases} \alpha & \text{si } x : \alpha \in \Gamma \\ s & \text{con } s \text{ variable fresca en otro caso} \end{cases}$
  - $\Gamma' = \Gamma \ominus \{x\}$
- $\mathbb{W}(UV) \stackrel{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \triangleright S(MN) : St$  donde
  - $\mathbb{W}(U) = \Gamma_1 \triangleright M : \tau$
  - $\mathbb{W}(V) = \Gamma_2 \triangleright N : \rho$
  - $t$  variable fresca
  - $S = \text{MGU}\{\tau \doteq \rho \rightarrow t\} \cup \{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_1, x : \sigma_2 \in \Gamma_2\}$

## 2. Código auxiliar

### 2.1. Expresiones

```
data Exp a = VarExp Symbol |
            ZeroExp |
            SuccExp (Exp a) |
            PredExp (Exp a) |
            IsZeroExp (Exp a) |
            TrueExp |
            FalseExp |
            IfExp (Exp a) (Exp a) (Exp a) |
            LamExp Symbol a (Exp a) |
            AppExp (Exp a) (Exp a)
```

```
type Symbol = String
type PlainExp = Exp ()
type AnnotExp = Exp Type
```

### 2.2. Tipos

```
data Type = TVar Int | TNat | TBool | TFun Type Type
```

### 2.3. Contexto

```
emptyContext :: Context
extendC :: Context -> Symbol -> Type -> Context
removeC :: Context -> Symbol -> Context
evalC :: Context -> Symbol -> Type
joinC :: [Context] -> Context
domainC :: Context -> [Symbol]
```

### 2.4. Sustituciones

```
emptySubst :: Subst
extendS :: Int -> Type -> Subst -> Subst

class Substitutable a where
    (<.>) :: Subst -> a -> a
    instance Substitutable Type -- subst <.> t
    instance Substitutable Context -- subst <.> c
    instance Substitutable Exp -- subst <.> e
```

### 2.5. Unificación

```
type UnifGoal = (Type, Type)
data UnifResult = UOK Subst | UError Type Type
mgu :: [UnifGoal] -> UnifResult
```