Taller de Inferencia de Tipos
Machete

Paradigmas de Lenguajes de Programación

1. Algoritmo de inferencia

- $\mathbb{W}(x) \overset{\text{def}}{=} \{x : s\} \rhd x : s, \quad s$ variable fresca
- $\mathbb{W}(\mathit{0}) \overset{\text{def}}{=} \emptyset \rhd \mathit{0} : nat$
- $\mathbb{W}(true) \overset{\text{def}}{=} \emptyset \rhd true : bool$
- $\mathbb{W}(false) \overset{\text{def}}{=} \emptyset \rhd false : bool$
- $\mathbb{W}(succ(U)) \overset{\text{def}}{=} S\Gamma \rhd S\, succ(M) : nat$ donde
  - $\mathbb{W}(U) = \Gamma \rhd M : \tau$
  - $S = MGU\{\tau \doteq nat\}$
- $\mathbb{W}(pred(U)) \overset{\text{def}}{=} S\Gamma \rhd S\, pred(M) : nat$ donde
  - $\mathbb{W}(U) = \Gamma \rhd M : \tau$
  - $S = MGU\{\tau \doteq nat\}$
- $\mathbb{W}(iszero(U)) \overset{\text{def}}{=} S\Gamma \rhd S\, iszero(M) : bool$ donde
  - $\mathbb{W}(U) = \Gamma \rhd M : \tau$
  - $S = MGU\{\tau \doteq nat\}$
- $\mathbb{W}(if\ U\ then\ V\ else\ W) \overset{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3 \rhd S(if\ M\ then\ P\ else\ Q) : S\sigma$ donde
  - $\mathbb{W}(U) = \Gamma_1 \rhd M : \rho$
  - $\mathbb{W}(V) = \Gamma_2 \rhd P : \sigma$
  - $\mathbb{W}(W) = \Gamma_3 \rhd Q : \tau$
  - $S = MGU\{\sigma \doteq \tau, \rho \doteq bool\} \cup \{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_i, x : \sigma_2 \in \Gamma_j, i \neq j\}$
- $\mathbb{W}(\lambda x.U) \overset{\text{def}}{=} \Gamma' \rhd \lambda x : \tau'.M : \tau' \to \rho$ donde
  - $\mathbb{W}(U) = \Gamma \rhd M : \rho$
  - $\tau' = \begin{cases} \alpha \text{ si } x : \alpha \in \Gamma \\ s \text{ con } s \text{ variable fresca en otro caso} \end{cases}$
  - $\Gamma' = \Gamma \ominus \{x\}$
- $\mathbb{W}(U\, V) \overset{\text{def}}{=} S\Gamma_1 \cup S\Gamma_2 \rhd S(M\, N) : St$ donde
  - $\mathbb{W}(U) = \Gamma_1 \rhd M : \tau$
  - $\mathbb{W}(V) = \Gamma_2 \rhd N : \rho$
  - t variable fresca
  - $S = MGU\{\tau \doteq \rho \to t\} \cup \{\sigma_1 \doteq \sigma_2 \mid x : \sigma_1 \in \Gamma_1, x : \sigma_2 \in \Gamma_2\}$

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) |
             EmptyListExp a |
             ConsExp (Exp a) (Exp a) |
             ZipWithExp (Exp a) (Exp a) Symbol Symbol (Exp a)

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

2.2. Tipos

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