Type system

rem

2024-09-20

Contents

1 Unification logic

During unification we substitute all *type variables* with concrete types. These concrete types are resolved based on the substitution rules below.

$$\frac{\text{Input terms and conditions}}{\text{Result of the rule}}$$

1.1 Rules

Let $Unify(T_1, T_2)$ be the unification function within the environment θ. Let S be the set of types: {Int, Float, Char, String, ...}.

Unification of identical types: $\theta = \{\}$

$$\frac{}{\text{Unify}(T_1, T_2) = \{\}}$$

Unification of two primitive types:

$$\frac{T_1 = T_2 \text{ and } T_1, T_2 \in S}{\text{Unify}(T_1, T_2) = \{\}}$$

Unification of type variables:

- $FV(T)$ refers to the *free variables* within T (the occurs check).

Occurs check prevents *infinite type expansion* which is *very* bad.

- If α does not occur within T, we can substitute α with T

$$\frac{\alpha \notin FV(T)}{Unify(\alpha, T) = \{\alpha \mapsto T\}}$$

Unification of arrays and pointers: Recursive substitution:

- If T_1 and T_2 can be unified with substitution θ then, $\text{Array}(T_1)$ and $\text{Array}(T_2)$

can also be unified with θ.

$$\frac{\text{Unify}(T_1, T_2) = \theta}{\text{Unify}(\text{Array}(T_1), \text{Array}(T_2)) = \theta}$$

$$\frac{\text{Unify}(T_1, T_2) = \theta}{\text{Unify}(\text{Pointer}(T_1), \text{Pointer}(T_2)) = \theta}$$

Unification of function types: To unify two function types $T_1 \rightarrow T_2$ and $T'_1 \rightarrow T'_2$, we unify the input types T_1 with T'_1 and T'_2 (the output types).

$$\frac{\text{Unify}(T_1, T'_1) = \theta_1 \text{Unify}(T_2[\theta_1], T'_2[\theta_1]) = \theta_2}{\text{Unify}(T_1 \rightarrow T_2, T_{'1 \rightarrow T'_2) = \theta_1 \circ \theta_2}}$$

Where, in the above equation:

- θ_1 and θ_2 are the *substitutions* from unifying the argument and return types
- The result is the *composition* of both substitutions (as in $\theta_1 \circ \theta_2$)