UNIFICATION

- unifies two terms
- used for pattern matching and type inference

ex int * x

y * (bool * bool)

are unifiable for x = bool * bool

y = int

ex int * int

int * bool

are not unifiable.

SUBSTITUTION

⟨type⟩ ::=

int |

float |

bool |

⟨type⟩ → ⟨type⟩ |

⟨type⟩ * ⟨type⟩ |

variable

⟨term⟩ ::=

constant |

variable |

f (⟨term⟩, ..., ⟨term⟩)

- the essential task of unification is to find a substitution that makes the two terms equal.

$$f(x, h(x, y))\ \{x \mapsto g(y),\ y \mapsto z\}$$

$$= f(g(y), h(g(y), z))$$

- the terms t_1 and t_2 are unifiable if there exists a substitution S such that

$$t_1 S = t_2 S$$

ex $t_1 = (x, g(y))$

$t_2 = (g(z), w)$

$S = \{x \mapsto g(z),\ w \mapsto g(y)\}$

MOST GENERAL UNIFIERS (MGU).

- it is possible that no unifier for given two terms exist.

x | $f(x)$ cannot be unified.

- there may be several unifiers.

$t_1 = f(x, g(y))$

$t_2 = f(g(z), w)$

$S = \{ x \mapsto g(z),\ y \mapsto w,\ w \mapsto g(w) \}$

$S' = \{ x \mapsto g(f(a,b)),$

$y \mapsto f(b,a),$

$z \mapsto f(a,b),$

$w \mapsto g(f(b,a)) \}.$

- when a unifier exists, there is always a most general unifier (mgu) that is unique up to renaming.

- S is the most general unifier of t_1 and t_2 if
  - it is a unifier of t_1, t_2.
  - for every other unifier S' of t_1, t_2, there exists a refinement of S to give S'.

- mgu can be efficiently computed.

$t_1 = f(x, g(y))$

$t_2 = f(g(z), w)$

$mgu = \{ x \mapsto g(z),\ y \mapsto w,\ w \mapsto g(w) \}$

$S' = \{ y \mapsto g(w) \} \cup mgu$

$= \{ y \mapsto g(w),\ x \mapsto g(z),\ w \mapsto g(g(w)) \}$

ex fun f ⇒ f (f "hi")

type of expression:

$\lambda f : T_1 \cdot apply(f : T_1,$

$apply(f : T_1,$

$"hi" : string) : T_2) : T_3$

$mgu(T_1, string \to T_2)$

$= \{T_1 \mapsto string \to T_2\} = S$

$mgu(S, T_1, T_2 \to T_3)$

$= \{T_1 \mapsto string \to T_2,\ T_2 \mapsto string,\ T_3 \mapsto string\}$