

## \* Unification Algorithm (Example)

⇒ Let's take simple example to understand unification

Eg:

$$\frac{P(x, b(y))}{\textcircled{1}}, \quad \frac{P(a, b(g(z)))}{\textcircled{2}}$$

⇒ Unification is all about making the expressions look identical. So for the above both expressions to make them look identical we need to do substitution.

⇒ Formal Notation for substitution

- we will substitute  $x$  with  $a$  in expression  $\textcircled{1}$ , so it is represented as  $a/x$  ie.  $a/x$  and also  $g(z)/y$

- With both the substitutions we can make expression  $\textcircled{1}$  look like expression  $\textcircled{2}$

we get,  $[a/x, g(z)/y]$

⇒ Let's take some more examples

$$Q(a, g(x, a), f(y)) = Q(a, g(f(b), a), x)$$

⇒ Here in the above example the variable  $x$  should be replaced with  $f(b)$

$$\therefore f(b)/x$$

⇒ The expression looks like

$$Q(a, g(f(b), a), f(b))$$

But Here  $y$  and  $b$  are not identical so as we can see in expression ① we have  $f(y)$

∴ we replace  $b$  for  $y$  i.e.  $b/y$

Now we get finally

$$Q(a, g(f(b), a), f(b))$$

Substitutions :

$$[f(b)/x, b/y]$$

⇒ Example :

Consider  $P(x, g(x))$  :

⇒ Solutions

- $P(z, y)$  : unifies with  $[x/z, g(x)/y]$
- $P(z, g(z))$  : unifies with  $[x/z \text{ or } z/x]$
- $P(\text{socrates}, g(\text{socrates}))$  : unifies, with  $[\text{socrates}/x]$
- $P(g(y), z)$  : unifies with  $[g(y)/x, g(g(y))/z]$
- $P(\text{socrates}, f(\text{socrates}))$  : does not unify  
( $f$  and  $g$  does not match)
- $P(g(y), y)$  : does not unify . no substitution works.

### \* Substitutions

Unification will produce a set of substitutions that make two literals the same. A

substitution  $t_i/v_i$  specifies substitution of term  $t_i$  and variable  $v_i$ .

## \* Unification Algorithm

⇒ Algorithm : Unify ( $L_1, L_2$ )

1. If  $L_1$  or  $L_2$  is a variable or constant, then :

a) If  $L_1$  and  $L_2$  are identical return NIL

b) Else if  $L_1$  is a variable, then if  $L_1$  occurs in  $L_2$  then return FAIL, else return  $\{L_2/L_1\}$

c) Else if  $L_2$  is a variable, then if  $L_2$  occurs in  $L_1$  then return FAIL, else return  $\{L_1/L_2\}$

d) Else return FAIL

2. If the initial predicate symbols in  $L_1$  and  $L_2$  are not identical, then return FAIL

3. If  $L_1$  and  $L_2$  have a different number of arguments, then return FAIL

4. Set SUBST to NIL

5. for  $i \leftarrow 1$  to number of arguments in  $L_1$ :

a) Call unify with the  $i^{\text{th}}$  argument of  $L_1$  and the  $i^{\text{th}}$  argument of  $L_2$  putting result in  $S$

b) If  $S = \text{FAIL}$  then return FAIL

c) If  $S$  is not equal to NIL then:

i) Apply  $S$  to the remainder of both  $L_1$  and  $L_2$

ii)  $\text{SUBST} = \text{APPEND}(S, \text{SUBST})$

6. Return SUBST

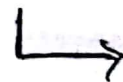
### \* Unification Implementation

1. Initialize the substitution set to be empty.  
A NIL set indicates failure.

2. Recursively unify expressions:

- Identical Item match

- If one item is a variable  $v_i$  and the other is a term  $t_i$  not containing that variable, then:



1. Substitute  $t_i/v_i$  in the existing substitutions
2. Add  $t_i/v_i$  to the substitution set.
3. If both items are functions, the function names must be identical and all arguments must unify. Substitutions are made in the rest of the expressions as unification proceeds.