* Unification Algorithm (Example)

⇒ Let's take simple example to understand unification

Eg:

$$\underbrace{P(x, f(y))}_{①} \quad , \quad \underbrace{P(a, f(g(z)))}_{②}$$

⇒ 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 ①, so it is represented as a for x ie. a/x and also $g(z)/y$

- With both the substitutions we can make expression ① look like expression ②

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

⇒ Let's take some more examples

$$\underbrace{Q(a, g(x,a), f(y))}_{1} = Q(a, \underbrace{g(f(b), a), x}_{2})$$

⇒ 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 $f(y)$

$\therefore$ we replace b for y ie. b/y

Now we get finally

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

<u>Substitutions</u> :

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

⇒ Example :

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

⇒ Solutions

- $P(z, y)$: unifies with $[x/z, g(x)/y]$
- $P(z, g(z))$: unifies with $[x/z$ or $z/x]$
- $P(socrates, g(socrates))$: unifies, ~~with~~ $[socrates/x]$
- $P(g(y), z)$: unifies with $[g(y)/x, g(g(y))/z]$
- $P(socrates, f(socrates))$: 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 (L1, L2)

1. If L1 or L2 is a variable or constant, then :
   a) If L1 and L2 are identical return NIL
   b) Else if L1 is a variable, then if L1 occurs in L2 then return FAIL, else return {(L2/L1)}
   c) Else if L2 is a variable, then if L2 occurs in L1 then return FAIL, else return {(L1/L2)}
   d) Else return FAIL

2. If the initial predicate symbols in L1 and L2 are not identical, then return FAIL

3. If L1 and L2 have a different number of arguments, then return FAIL

4. Set SUBST to NIL

5. for i ← 1 to number of arguments in L1 :

a) Call unify with the i^{th} argument of L1 and the i^{th} argument of L2 putting result in S

b) If S = FAIL then return FAIL

c) If S is not equal to NIL then :

i) Apply S to the remainder of both L1 and L2

ii) SUBST = APPEND (S, 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.