# Unification and resolution

* Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.
* It takes two literals as input and makes them identical using substitution.
* Let Ψ1 and Ψ2 be two atomic sentences and  **Ψ12**, then it can be expressed as **UNIFY(Ψ1, Ψ2)**.

**Algorithm: Unify(Ψ1, Ψ2)**

Step. 1: If Ψ1 or Ψ2 is a variable or constant, then:

a) If Ψ1 or Ψ2 are identical, then return NIL.

b) Else if Ψ1is a variable,

a. then if Ψ1 occurs in Ψ2, then return FAILURE

b. Else return { (Ψ2/ Ψ1)}.

c) Else if Ψ2 is a variable,

a. If Ψ2 occurs in Ψ1 then return FAILURE,

b. Else return {( Ψ1/ Ψ2)}.

d) Else return FAILURE.

Step.2: If the initial Predicate symbol in Ψ1 and Ψ2 are not same, then return FAILURE.

Step. 3: IF Ψ1 and Ψ2 have a different number of arguments, then return FAILURE.

Step. 4: Set Substitution set(SUBST) to NIL.

Step. 5: For i=1 to the number of elements in Ψ1.

a) Call Unify function with the ith element of Ψ1 and ith element of Ψ2, and put the result into S.

b) If S = failure then returns Failure

c) If S ≠ NIL then do,

a. Apply S to the remainder of both L1 and L2.

b. SUBST= APPEND(S, SUBST).

Step.6: Return SUBST.

**CODE**:

def get\_index\_comma(string):

"""

Return index of commas in string

"""

index\_list = list()

# Count open parentheses

par\_count = 0

for i in range(len(string)):

if string[i] == ',' and par\_count == 0:

index\_list.append(i)

elif string[i] == '(':

par\_count += 1

elif string[i] == ')':

par\_count -= 1

return index\_list

def is\_variable(expr):

"""

Check if expression is variable

"""

for i in expr:

if i == '(':

return False

return True

def process\_expression(expr):

"""

input: - expression:

'Q(a, g(x, b), f(y))'

return: - predicate symbol:

Q

- list of arguments

['a', 'g(x, b)', 'f(y)']

"""

# Remove space in expression

expr = expr.replace(' ', '')

# Find the first index == '('

index = None

for i in range(len(expr)):

if expr[i] == '(':

index = i

break

# Return predicate symbol and remove predicate symbol in expression

predicate\_symbol = expr[:index]

expr = expr.replace(predicate\_symbol, '')

# Remove '(' in the first index and ')' in the last index

expr = expr[1:len(expr) - 1]

# List of arguments

arg\_list = list()

# Split string with commas, return list of arguments

indices = get\_index\_comma(expr)

if len(indices) == 0:

arg\_list.append(expr)

else:

arg\_list.append(expr[:indices[0]])

for i, j in zip(indices, indices[1:]):

arg\_list.append(expr[i + 1:j])

arg\_list.append(expr[indices[len(indices) - 1] + 1:])

return predicate\_symbol, arg\_list

def get\_arg\_list(expr):

"""

input: expression:

'Q(a, g(x, b), f(y))'

return: full list of arguments:

['a', 'x', 'b', 'y']

"""

\_, arg\_list = process\_expression(expr)

flag = True

while flag:

flag = False

for i in arg\_list:

if not is\_variable(i):

flag = True

\_, tmp = process\_expression(i)

for j in tmp:

if j not in arg\_list:

arg\_list.append(j)

arg\_list.remove(i)

return arg\_list

def check\_occurs(var, expr):

"""

Check if var occurs in expr

"""

arg\_list = get\_arg\_list(expr)

if var in arg\_list:

return True

return False

def unify(expr1, expr2):

# Step 1:

if is\_variable(expr1) and is\_variable(expr2):

if expr1 == expr2:

return 'Null'

else:

return False

elif is\_variable(expr1) and not is\_variable(expr2):

if check\_occurs(expr1, expr2):

return False

else:

tmp = str(expr2) + '/' + str(expr1)

return tmp

elif not is\_variable(expr1) and is\_variable(expr2):

if check\_occurs(expr2, expr1):

return False

else:

tmp = str(expr1) + '/' + str(expr2)

return tmp

else:

predicate\_symbol\_1, arg\_list\_1 = process\_expression(expr1)

predicate\_symbol\_2, arg\_list\_2 = process\_expression(expr2)

# Step 2

if predicate\_symbol\_1 != predicate\_symbol\_2:

return False

# Step 3

elif len(arg\_list\_1) != len(arg\_list\_2):

return False

else:

# Step 4: Create substitution list

sub\_list = list()

# Step 5:

for i in range(len(arg\_list\_1)):

tmp = unify(arg\_list\_1[i], arg\_list\_2[i])

if not tmp:

return False

elif tmp == 'Null':

pass

else:

if type(tmp) == list:

for j in tmp:

sub\_list.append(j)

else:

sub\_list.append(tmp)

# Step 6

return sub\_list

if \_\_name\_\_ == '\_\_main\_\_':

#Data 1

f1 = 'p(b(A), X, g(X), f(g(Z)))'

f2 = 'p(Z, f(Y), Y, f(Y))'

result = unify(f1, f2)

if not result:

print('Unification failed!')

else:

print('Unification successfull!')

print(result)

INPUT: 'Q(a, g(x, b), f(y))'

OUTPUT:

