

ARTIFICIAL INTELLIGENCE

EXPERIMENT NO: 7

IMPLEMENTATION OF UNIFICATION AND RESOLUTION

Nikith Kumar Seemakurthi

RA1911003020480

AIM:

To implement Unification and Resolution using python.

ALGORITHM:

1. If Ψ_1 or Ψ_2 is a variable or constant, then:
 - a) If Ψ_1 or Ψ_2 are identical, then return NULL.
 - b) Else if Ψ_1 is a variable:
 - Then if Ψ_1 occurs in Ψ_2 , then return False
 - Else return (Ψ_2/Ψ_1)
 - c) Else if Ψ_2 is a variable:
 - then if Ψ_2 occurs in Ψ_1 , then return False
 - Else return (Ψ_1/Ψ_2)
 - d) Else return False
2. If the initial Predicate symbol in Ψ_1 and Ψ_2 are not same, then return False.
3. If Ψ_1 and Ψ_2 have a different number of arguments, then return False.
4. Create Substitution list.
5. For $i=1$ to the number of elements in Ψ_1 .
 - a) Call Unify function with the i th element of Ψ_1 and with element of Ψ_2 , and put the result into S.
 - b) If $S = \text{False}$ then returns False.
 - c) If $S \neq \text{Null}$ then append to Substitution list.
6. Return Substitution list.

SOURCE CODE:

```
def get_index_comma(string):  
    index_list = list()  
    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):
```

```
    for i in expr:
        if i=='(':
            return False
    return True
```

```
def process_expression(expr):
```

```
    expr = expr.replace(",")
    index = None
    for i in range(len(expr)):
        if expr[i]=='(':
            index=i
            break
    predicate_symbol = expr[:index]
    expr = expr.replace(predicate_symbol,"")
    expr = expr[1:len(expr)-1]
    arg_list = list()
    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):
```

```
    _,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):
    arg_list = get_arg_list(expr)
    if var in arg_list:
        return True
    return False

```

```

def unify(expr1,expr2):
    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)
    if predicate_symbol_1 != predicate_symbol_2:
        return False
    elif len(arg_list_1) != len(arg_list_2):
        return False
    else:
        sub_list = list()
        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)
        return sub_list

```

```

if __name__=='__main__':

```

```

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

```

```

f2 = 'p(Z,f(Y),f(Y))'
result = unify(f1,f2)
if not result:
    print('Unification failed!')
else:
    print('Unification successfully!')
    print(result)

```

OUTPUT:

The screenshot shows a Python IDE with a file named 'main.py'. The code defines three functions: `get_index_comma`, `is_variable`, and `process_expression`. The `get_index_comma` function iterates through a string to find comma indices, adjusting for parentheses. The `is_variable` function checks if a character is a variable (not a comma). The `process_expression` function replaces commas with spaces. The output window shows the message 'Unification successfully!' followed by the substitution list `['b(A)/Z', 'f(Y)/X', 'g(Z)/Y']`.

```

main.py
1- def get_index_comma(string):
2-     index_list = list()
3-     par_count = 0
4-     for i in range(len(string)):
5-         if string[i] == ',' and par_count == 0:
6-             index_list.append(i)
7-         elif string[i] == '(':
8-             par_count += 1
9-         elif string[i] == ')':
10-            par_count -= 1
11-     return index_list
12-
13- def is_variable(expr):
14-     for i in expr:
15-         if i == ',':
16-             return False
17-     return True
18-
19- def process_expression(expr):
20-     expr = expr.replace(',', ' ')
21-     index = None
22-     for i in range(len(expr)):
23-         if expr[i] == '(':

```

Unification successfully!
['b(A)/Z', 'f(Y)/X', 'g(Z)/Y']

RESULT:

Hence, the Unification and Resolution algorithm is implemented successfully.