NAME :- DIVYANG BAGLA
PANEL :- C PC-33
C-3

PAGE NO.:

AI LAB ASSIGNMENT - 4

AIM :- Implement unification algoridm.

OBJECTIVE :- To study and impement unification algo.

THEORY :-

Unification Algoidm :-

It is used when we need to determine contradiction It computes 2 literals and decides whether there exists a set of substitutions that make them identical. In this case, literal is represented as a list, where 1st element is name of a predicate.

Resolution ~~as~~proof procedure :-

Allows compate interference mechanism in prepositional logic, the procedure to proof by resolution of proposition them to select 2 clauses and calculate each parent clause. It's resolide is empty clause the contradection is found.

INPUT :- two literals L1 & L2

OUTPUT :- A set of substitution.

FAQ's

Q) Why resolution is required?

Resolution procedures interference algorithm when put with any complete search rules. It works by using the principle of proof by contradiction

The procedure continues until no new clauses can be added or an application of resolution rule derives the empty clause.

Q.2 Pre requisites of applying unification Algorithm
1) Predicate symbol must be same.
2) No. of arguments in both expression must be identical.

Q.3 What are applications of unification Algorithm
Automated reasoning is a main application of this algorithm. It is also used in logic programs & cryptographic analysis.

AI - Lab4 Code

```
import random
class Variable:
    def __init__(self,value):
        self.value = value
    def __eq__(self, other):
        return self.value == other.value
class Constant:
    def __init__(self,value):
        self.value = value
    def __eq__(self, other):
        return self.value == other.value
class Rel:
    def __init__(self,name,args):
        #This is a list
        self.name = name
        self.value = str(self.name)+str([i.value for i in args])
        self.args = args



def Unify(L1,L2,testset):
    '''
    L1 and L2 are Rel types, variables or constants
    '''
    #If both are variable or constants
    if(isinstance(L1,Variable) or isinstance(L2,Variable) or 
isinstance(L1,Constant) or isinstance(L2,Constant)):
        if L1 == L2:
            return None
        elif isinstance(L1,Variable):
            if isinstance(L2,Variable):
                print("Both missmatching variables")
                return False
            else:
                if L1.value not in testset.values():
                    return [L2,L1]
                else:
                    print("Ambigious Variable")
                    return False
        elif isinstance(L2,Variable):
            if isinstance(L1,Variable):
                print("Both missmatching variables")
                return False
            else:
                if L2.value not in testset.values():
                    return [L1,L2]
                else:
                    print("Ambigious Variable")
                    return False
        else:
            print("Missmatch")
            return False

    #Ensuring the functions are the same
    elif L1.name != L2.name:
        print("Relation Missmatch")
        return False
    #Ensuring the functions have the same number of arguments
    elif len(L1.args) != len(L2.args):
        print("length does not match")
        return False

    SUBSET = {}

    for i in range(len(L1.args)):
        S = Unify(L1.args[i],L2.args[i],SUBSET)
        if S==False:
            return False
        if S != None:
            SUBSET[S[0].value] = S[1].value

    return SUBSET


if __name__ == "__main__":


print(Unify(Rel("Knows",[Constant("Raj"),Variable("X")]),Rel("Knows",[Variable("Y")
,Rel("Sister",[Variable("Y")])]),{}))
    print()

print(Unify(Rel("Knows",[Constant("Raj"),Variable("X")]),Rel("Knows",[Variable("Y")
,Constant("Seeta")]),{}))
    print()

print(Unify(Rel("Knows",[Constant("Raj"),Variable("X")]),Rel("Knows",[Variable("X")
,Constant("Seeta")]),{}))



'''
OUTPUT:-


{'Raj': 'Y', "Sister['Y']": 'X'}

{'Raj': 'Y', 'Seeta': 'X'}
```

```
Ambigious Variable
False


'''
```