Karan Goddami
PB40
B3
AI

lab- assignment- 4

- Aim

To implement Unification Algorithm

- Objective

To study and implement Unification algorithm.

- Theory

→ Unification Algo :-

Unification is a process of making two different logical atomic expression identical by finding a substitution. The UNIFY algorithm is used for unification which takes two atomic sentences and returns a unifier for those sentences. It returns fail if the expressions do not match with each oth The substitution variables are called most general unifier or MGU. resolution as proof procedure.

This technique uses proof by contradiction and is based on the fact that any sentence in propositional logic can be transformed into an equivalent sentence in conjuctive normal form.

- Input: Two Literals L1 & L2
- Output: A set of substitutions
- Algorithm: Unification Algorithm
- Platform: Windows
- FAQ's

Q.1 Why resolution is required?

A It is used if there are various statements given and we need to prove a conclusion of these statements

Q2 What are the prerequisites for applying a unification algorithm?

A i) Predicate Symbol must be same, atoms or expression with diff. predicate symbol can never be unified
ii) Number of arguments in both expressions must be identical.
iii) Unification will fail if there are two similar variables present in same expression.

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",[Vari
able("Y"),Rel("Sister",[Variable("Y")])]),{}))
    print()

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

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

OUTPUT:

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

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

Ambigious Variable
False