Name - Vasu Kalariya
Roll - PE29
Sub - AI

Lab Assignment - 4

Title: Implementation of Unification algorithm.

Aim: To implement Unification algorithm.

Objective: To study and implement Unification algorithm.

Theory:

→ Unification Algorithm
In logic and compute science unification is an algo. proces of solving equations between symbolic expressions. A unification algorithm should compute for a given problem a complete and minimal substitution set that is a set covering all its solutions and containing no redundant members.

→ Resolution as proof ~~procedure~~ procedure.
Resolution is a ~~theory~~ theorem proving technique that proceeds by building refutation proofs, i.e proofs by contradictions. It was invented by a mathematician John Alan in 1965. Resolution is used if there are various statement are given and we need to prove a conclusion of ~~these~~ those statements Unification is a key concept in proofs by resolution Resolution is a single inference rule which can efficiently operate on the conjuctive normal form or ~~to~~ clausal form.

→ Steps:
→ Conversion of facts into first order logic.
→ Convert ~~for~~ FOL statements into CNF

→ Negate the statements into ~~CNF~~ which needs to prove.
→ Draw resolution graph (unification)

Input: Two literals L1 & L2

Output: A set of substitution

Algorithm: Unification algorithm

FAQ

1 Why Resolution is required
Resolution is used if there are various statements are given and we need to prove a conclusion of those statement unification is a key concept in proofs by resolutions

2 What are pre-requisites for applying unification algorithm
1] Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified
2] Number of arguments in both expression must be identical.
3] Unification will fail if there are two similar variables present in the same expression

3 What are the applications of unification algorithm
1] Logical programming
2] Programming language type system implementation
3] Cryptographic Protocol analysis.
4] Term rewriting algorithm
5] SMT solvers.

```python
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

    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")]),{}))
```