Name: Shabbar Adamjee  
Roll No.: PB57  
PRN: 1032221508

*AIES ASSIGNMENT 4*

*Unification Algorithm*

*Code*

from dataclasses import dataclass

from typing import Dict, List, Optional, Union

@dataclass

class Constant:

    value: str

@dataclass

class Variable:

    value: str

@dataclass

class Function:

    name: str

    args: List["Term"]

Term = Union[Constant, Variable, Function]

@dataclass

class Relation:

    predicate: str

    args: List[Term]

    def \_\_str\_\_(self):

        args\_str = ", ".join(str(arg) for arg in self.args)

        return f"{self.predicate}({args\_str})"

def occurs\_check(var: Variable, term: Term, substitution: Dict[str, Term]) -> bool:

    if isinstance(term, Variable):

        if var.value == term.value:

            return True

        elif term.value in substitution:

            return occurs\_check(var, substitution[term.value], substitution)

    elif isinstance(term, Constant):

        return False

    elif isinstance(term, Function):

        return any(occurs\_check(var, arg, substitution) for arg in term.args)

    return False

def apply\_substitution(term: Term, substitution: Dict[str, Term]) -> Term:

    if isinstance(term, Variable) and term.value in substitution:

        return apply\_substitution(substitution[term.value], substitution)

    elif isinstance(term, Function):

        return Function(

            term.name, [apply\_substitution(arg, substitution) for arg in term.args]

        )

    return term

def unify\_terms(

    term1: Term, term2: Term, substitution: Dict[str, Term]

) -> Optional[Dict[str, Term]]:

    term1 = apply\_substitution(term1, substitution)

    term2 = apply\_substitution(term2, substitution)

    if term1 == term2:

        return substitution

    elif isinstance(term1, Variable):

        if occurs\_check(term1, term2, substitution):

            return None

        substitution[term1.value] = term2

        return substitution

    elif isinstance(term2, Variable):

        return unify\_terms(term2, term1, substitution)

    elif isinstance(term1, Function) and isinstance(term2, Function):

        if term1.name != term2.name or len(term1.args) != len(term2.args):

            return None

        for arg1, arg2 in zip(term1.args, term2.args):

            substitution = unify\_terms(arg1, arg2, substitution)

            if substitution is None:

                return None

        return substitution

    elif isinstance(term1, Constant) or isinstance(term2, Constant):

        return None

    else:

        return None

def unifier(r1: Relation, r2: Relation) -> Optional[Dict[str, Term]]:

    if r1.predicate != r2.predicate or len(r1.args) != len(r2.args):

        return None

    substitution = {}

    for arg1, arg2 in zip(r1.args, r2.args):

        result = unify\_terms(arg1, arg2, substitution)

        if result is None:

            return None

        substitution = result

    return substitution

def main():

    # Example usage

    relation1 = Relation("Knows", [Constant("Raj"), Variable("X")])

    relation2 = Relation("Knows", [Variable("Y"), Constant("Seeta")])

    result = unifier(relation1, relation2)

    if result:

        print("Unification successful:")

        for var, term in result.items():

            print(f"{var} = {term}")

    else:

        print("Unification failed")

    # Additional test cases

    relation3 = Relation("Likes", [Variable("X"), Variable("Y")])

    relation4 = Relation("Likes", [Constant("John"), Constant("Pizza")])

    result2 = unifier(relation3, relation4)

    if result2:

        print("\nUnification successful:")

        for var, term in result2.items():

            print(f"{var} = {term}")

    else:

        print("\nUnification failed")

    # Test case with nested functions

    relation5 = Relation(

        "Father", [Function("Parent", [Variable("X")]), Constant("John")]

    )

    relation6 = Relation(

        "Father", [Function("Parent", [Constant("Mary")]), Variable("Y")]

    )

    result3 = unifier(relation5, relation6)

    if result3:

        print("\nUnification successful:")

        for var, term in result3.items():

            print(f"{var} = {term}")

    else:

        print("\nUnification failed")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

*Output*

