AI LAB

CODE

#program unification in First Order Logic

def unify(x1, x2):

    """

    Unify two expressions (x1 and x2) based on the given unification algorithm.

    Returns a substitution set (SUBST) or FAILURE if unification is not possible.

    """

    if is\_variable\_or\_constant(x1) or is\_variable\_or\_constant(x2):

        if x1 == x2:

            return []

        elif is\_variable(x1):

            if occurs\_check(x1, x2):

                return "FAILURE"

            else:

                return [(x2, x1)]

        elif is\_variable(x2):

            if occurs\_check(x2, x1):

                return "FAILURE"

            else:

                return [(x1, x2)]

        else:

            return "FAILURE"

    if not is\_same\_predicate(x1, x2):

        return "FAILURE"

    if len(x1) != len(x2):

        return "FAILURE"

    subst = []

    for i in range(len(x1)):

        s = unify(x1[i], x2[i])

        if s == "FAILURE":

            return "FAILURE"

        elif s:

            subst.extend(s)

            apply\_substitution(s, x1[i+1:])

            apply\_substitution(s, x2[i+1:])

    return subst

def is\_variable\_or\_constant(expr):

    """Check if the expression is a variable or a constant."""

    return isinstance(expr, str) and expr.isalnum()

def is\_variable(expr):

    """Check if the expression is a variable."""

    return isinstance(expr, str) and expr.islower()

def occurs\_check(var, expr):

    """Check if the variable occurs in the expression."""

    if var == expr:

        return True

    elif isinstance(expr, (list, tuple)):

        return any(occurs\_check(var, sub\_expr) for sub\_expr in expr)

    return False

def is\_same\_predicate(x1, x2):

    """Check if the initial predicate symbols of x1 and x2 are the same."""

    if isinstance(x1, (list, tuple)) and isinstance(x2, (list, tuple)):

        return x1[0] == x2[0]

    return False

def apply\_substitution(subst, expr):

    """Apply the substitution set to the given expression."""

    for old, new in subst:

        if expr == old:

            return new

        elif isinstance(expr, (list, tuple)):

            return [apply\_substitution(subst, sub\_expr) for sub\_expr in expr]

    return expr

def parse\_input(expr):

    """Parse user input into a list or tuple representing the predicate."""

    try:

        return eval(expr)

    except Exception as e:

        print(f"Error in input format: {e}")

        return None

print("Enter two expressions to unify. Use list/tuple format.")

print("Example: ['P', 'x', 'a'] represents P(x, a)")

expr1\_input = input("Enter the first expression: ")

expr2\_input = input("Enter the second expression: ")

expr1 = parse\_input(expr1\_input)

expr2 = parse\_input(expr2\_input)

if expr1 is not None and expr2 is not None:

    result = unify(expr1, expr2)

    print("Unification Result:", result)

else:

    print("Invalid input format. Please try again.")

print("tamanna  - 1BM22CS301")

OUTPUT

