AI
LAB 7

Unification

Unification in First-Order Logic deals with finding a common substitution for variables in different terms to make them
match. The goal of unification is to make two expressions identical by assigning values to variables in a way that preserves their
meanings.
It involves substituting terms for variables in a way that the resulting expressions match.

```
# Unification in First-Order Logic
# Date: 16 November 2024
# College: BMSCE

def is_variable(x):
    """Check if a term is a variable (starts with lowercase and is a
single letter)."""
    return isinstance(x, str) and x[0].islower()

def unify(x, y, substitution=None):
    """Main Unification function."""
    if substitution is None:
        substitution = {}

    if x == y:
        return substitution

    # If x is a variable
    if is_variable(x):
        return unify_var(x, y, substitution)

    # If y is a variable
    if is_variable(y):
        return unify_var(y, x, substitution)

    # If both are compound expressions (predicates or functions)
    if isinstance(x, list) and isinstance(y, list):
        if x[0] != y[0] or len(x) != len(y):  # Different predicate
names or argument counts
            return "FAIL"
        for i in range(1, len(x)):
            substitution = unify(x[i], y[i], substitution)
```

```
            if substitution == "FAIL":
                return "FAIL"
        return substitution

    return "FAIL"


def unify_var(var, x, substitution):
    """Handles unification when a variable is encountered."""
    if var in substitution:
        return unify(substitution[var], x, substitution)
    elif x in substitution:
        return unify(var, substitution[x], substitution)
    elif occurs_check(var, x, substitution):
        return "FAIL"
    else:
        substitution[var] = x
        return substitution


def occurs_check(var, x, substitution):
    """Checks if var occurs in x (to avoid circular references)."""
    if var == x:
        return True
    elif isinstance(x, list):
        return any(occurs_check(var, xi, substitution) for xi in x)
    return False


def pretty(expr):
    """Converts a list form expression back into readable string."""
    if isinstance(expr, list):
        return f"{expr[0]}({', '.join(pretty(x) for x in expr[1:])})"
    return expr


# --- Example ---
# Unify Eats(x, Apple) with Eats(Riya, y)

expr1 = ["Eats", "x", "Apple"]
expr2 = ["Eats", "Riya", "y"]

result = unify(expr1, expr2)

print("Expression 1:", pretty(expr1))
print("Expression 2:", pretty(expr2))
print("\nResult of Unification:")

if result == "FAIL":
```

```
    print("Unification failed.")
else:
    print("Substitutions:", result)
    unified_expr = ["Eats", result.get("x", "x") if "x" in result else
expr1[1],
                            result.get("y", "y") if "y" in result else
expr1[2]]
    print("Unified Expression:", pretty(unified_expr))
```

Output:

```
Expression 1: Eats(x, Apple)
Expression 2: Eats(Riya, y)

Result of Unification:
Substitutions: {'x': 'Riya', 'y': 'Apple'}
Unified Expression: Eats(Riya, Apple)
```