

7. Implement unification in first order logic.

```
def unify(s1, s2, theta={}):

    if theta is None:
        return None

    if s1 == s2:
        return theta

    if isinstance(s1, str) and s1.islower():
        return unify_var(s1, s2, theta)

    if isinstance(s2, str) and s2.islower():
        return unify_var(s2, s1, theta)

    if isinstance(s1, tuple) and isinstance(s2, tuple) and len(s1) == len(s2):
        return unify(s1[1:], s2[1:], unify(s1[0], s2[0], theta))

    return None


def unify_var(var, x, theta):
    if var in theta:
        return unify(theta[var], x, theta)
    elif x in theta:
        return unify(var, theta[x], theta)
    elif occurs_check(var, x, theta):
        return None
    else:
        theta[var] = x
        return theta


def occurs_check(var, x, theta):
    if var == x:
        return True
    elif isinstance(x, str) and x.islower() and x in theta:
        return occurs_check(var, theta[x], theta)
    elif isinstance(x, tuple):
        for arg in x:
            if occurs_check(var, arg, theta):
                return True
    return False
```

```

s1 = ('p', 'x', ('f', 'x'), ('y'))
s2 = ('p', 'a', 'y', ('f', 'x'))

substitution = unify(s1, s2)

if substitution:
    print("Unification successful:")
    print(f"Substitution: {substitution}")
else:
    print("Unification failed.")

```

Output:

⇒ Unification successful:
Substitution: {'x': 'a', 'y': ('f', 'x')}

Observation

15/11/24
Simulated Annealing

★ Implement Simulated Annealing to solve N-Queens problem.

Function calculate_conflicts(board):
Initialize Conflict = 0
Calculate Conflicts = No of queens attacking each other
Return Conflict

Function simulated_annealing(n):
current_board = random board of size n
current_cost = calculate_conflicts(current_board)
temperature = 1000

while temperature > 0.001:
new_board = generate random neighbours of current_board
new_cost = calculate_conflicts(new_board)

If new_cost < current_cost or $\text{random() < exp}((\text{current_cost} - \text{new_cost}) / \text{temperature})$:

current_board = new_board
current_cost = new_cost
temperature *= 0.99

Return current_board

Output

Enter the no of queens 4
Enter the initial positions of the queens as a list of row indices (0-indexed):
3 1 2 0

Iteration 0 : Cost = 3, Temperature = 1000.00
[2, 1, 2, 0]

Iteration 1 : Cost = 3, Temperature = 990.00
[2, 1, 2, 0]

Iteration 2 : Cost = 2, Temperature = 980.10
[2, 0, 2, 0]

Solution: [1, 3, 0, 2]