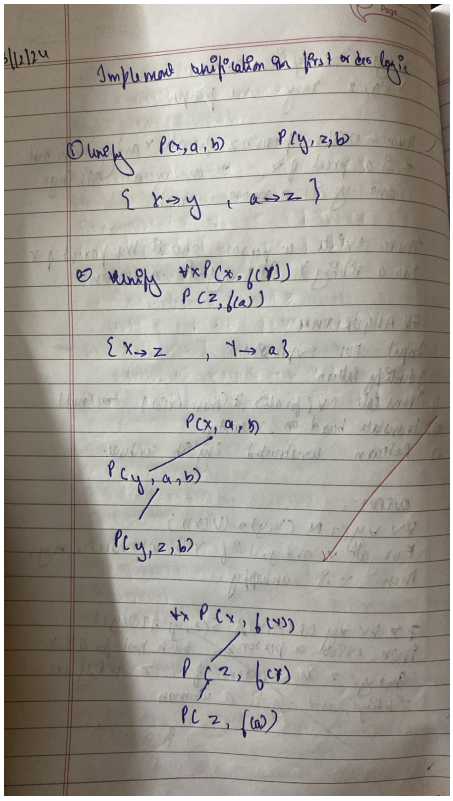
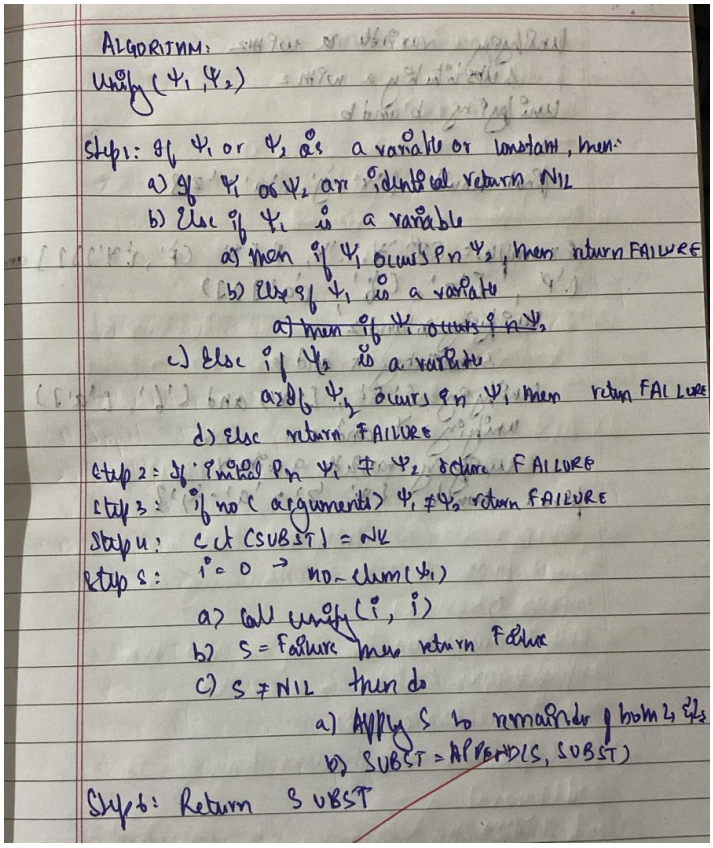
Date:3/12/24
Program Title:Implement unification in first order logic.
Algorithm


Implement unification on first order logic
① Unify P(x, a, b) P(y, z, b)
{ x → y , a → z }
② Unify ∀x P(x, f(y))
P(z, f(a))
{ x → z , y → a }
P(x, a, b)
P(y, a, b)
P(y, z, b)
∀x P(x, f(y))
P(z, f(y))
P(z, f(a))


ALGORITHM:
Unify (Ψ1, Ψ2)
Step 1: If Ψ1 or Ψ2 is a variable or constant, then:
a) If Ψ1 and Ψ2 are identical return NIL
b) Else if Ψ1 is a variable
a) then if Ψ1 occurs in Ψ2, then return FAILURE
(b) Else if Ψ2 is a variable
a) then if Ψ2 occurs in Ψ1
c) Else if Ψ2 is a variable
a) If Ψ2 occurs in Ψ1 then return FAILURE
d) Else return FAILURE
Step 2: If predicate Ψ1 ≠ Ψ2 return FAILURE
Step 3: if no (arguments) Ψ1 ≠ Ψ2 return FAILURE
Step 4: Set SUBST = NIL
Step 5: i = 0 → no. elem(Ψ1)
a) Call unify(i, i)
b) S = failure then return failure
c) S ≠ NIL then do
a) Apply S to remainder of both Ψ1, Ψ2
b) SUBST = APPEND(S, SUBST)
Step 6: Return SUBST

Code:

```
class UnificationError(Exception):
    pass
def unify(term1, term2, substitution=None, level=0):
    if substitution is None:
        substitution = {}
    indent = " " * level
    print(f"{indent}Unifying: {term1} and {term2}")
    if term1 == term2:
        print(f"{indent}Terms are identical, no substitution.")
        return substitution
    elif is_variable(term1):
        print(f"{indent}Unifying variable {term1} with {term2}")
        return unify_variable(term1, term2, substitution, level + 1)
    elif is_variable(term2):
        print(f"{indent}Unifying variable {term2} with {term1}")
        return unify_variable(term2, term1, substitution, level + 1)
    elif isinstance(term1, tuple) and isinstance(term2, tuple):
        if term1[0] != term2[0]:
            raise UnificationError(f"Cannot unify {term1} with {term2}, different function names.")
        for t1, t2 in zip(term1[1], term2[1]):
            substitution = unify(t1, t2, substitution, level + 1)
        return substitution
    else:
        raise UnificationError(f"Cannot unify {term1} with {term2}, they are incompatible.")
def unify_variable(var, term, substitution, level):
    indent = " " * level
    if var in substitution:
        print(f"{indent}Variable {var} is already substituted as {substitution[var]}")
        return unify(substitution[var], term, substitution, level + 1)
    if term == var:
        print(f"{indent}Variable {var} is the same as the term, no substitution.")
        return substitution
    if is_variable(term) and var in get_variables(term):
        raise UnificationError(f"{indent}Cannot unify variable {var} with {term} (circular unification).")
    print(f"{indent}Substituting {var} with {term}")
    substitution[var] = term
    return substitution
```

```
def is_variable(term):
    return isinstance(term, str) and term.islower()
def get_variables(term):
    if is_variable(term):
        return {term}
    elif isinstance(term, tuple):
        variables = set()
        for arg in term[1]:
            variables.update(get_variables(arg))
        return variables
    else:
        return set()
if __name__ == "__main__":
    print("\nExample 1: Unifying P(x, a, b) and P(y, z, b)\n")
    term1 = ('P', ['x', 'a', 'b'])
    term2 = ('P', ['y', 'z', 'b'])
    try:
        substitution = unify(term1, term2)
        print("\nFinal Unification Result:", substitution)
    except UnificationError as e:
        print("Unification failed:", e)
    print("\nExample 2: Unifying P(x, f(Y)) and P(Z, f(a))\n")
    term3 = ('P', ['x', ('f', ['Y'])])
    term4 = ('P', ['Z', ('f', ['a'])])
    try:
        substitution = unify(term3, term4)
        print("\nFinal Unification Result:", substitution)
    except UnificationError as e:
        print("Unification failed:", e)
```

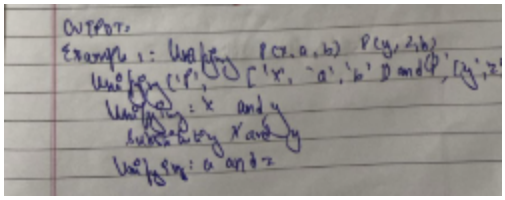
class UnificationError(Exception):

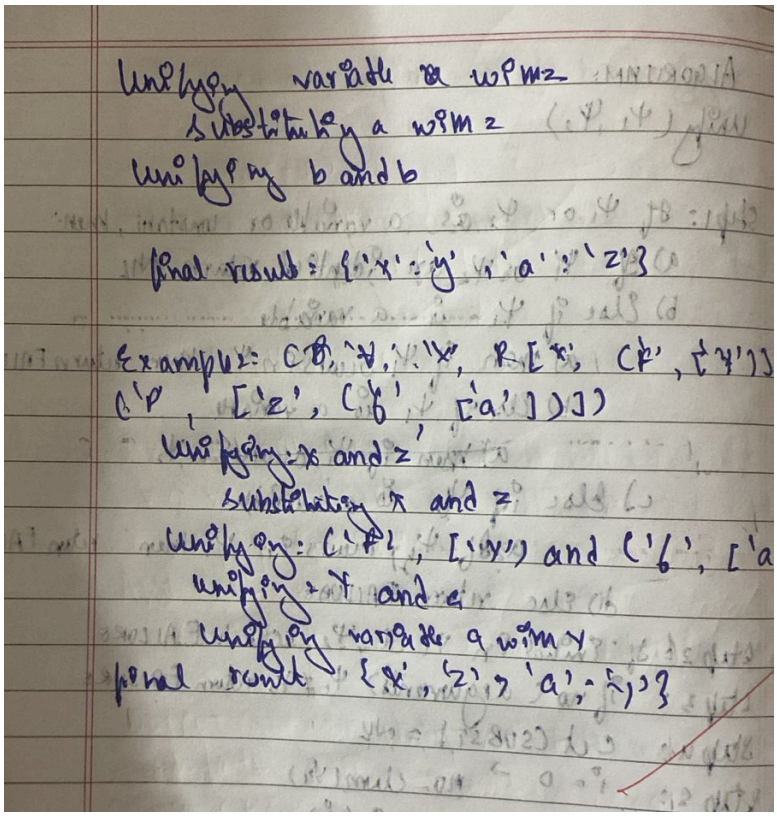
```
    pass

def unify(term1, term2, substitution=None, level=0):
    if substitution is None:
        substitution = {}
    indent = " " * level
    print(f"{indent}Unifying: {term1} and {term2}")
    if term1 == term2:
        print(f"{indent}Terms are identical, no substitution.")
        return substitution
    elif is_variable(term1):
        print(f"{indent}Unifying variable {term1} with {term2}")
        return unify_variable(term1, term2, substitution, level + 1)
    elif is_variable(term2):
        print(f"{indent}Unifying variable {term2} with {term1}")
        return unify_variable(term2, term1, substitution, level + 1)
    elif isinstance(term1, tuple) and isinstance(term2, tuple):
        if term1[0] != term2[0]:
            raise UnificationError(f"Cannot unify {term1} with {term2}, different function names.")
        for t1, t2 in zip(term1[1], term2[1]):
            substitution = unify(t1, t2, substitution, level + 1)
        return substitution
    else:
        raise UnificationError(f"Cannot unify {term1} with {term2}, they are incompatible.")

def unify_variable(var, term, substitution, level):
    indent = " " * level
    if var in substitution:
        print(f"{indent}Variable {var} is already substituted as {substitution[var]}")
        return unify(substitution[var], term, substitution, level + 1)
    if term == var:
        print(f"{indent}Variable {var} is the same as the term, no substitution.")
        return substitution
    if is_variable(term) and var in get_variables(term):
        raise UnificationError(f"{indent}Cannot unify variable {var} with {term} (circular
unification).")
    print(f"{indent}Substituting {var} with {term}")
    substitution[var] = term
    return substitution
```

```
def is_variable(term):
    return isinstance(term, str) and term.islower()

def get_variables(term):
    if is_variable(term):
        return {term}
    elif isinstance(term, tuple):
        variables = set()
        for arg in term[1]:
            variables.update(get_variables(arg))
        return variables
    else:
        return set()

if __name__ == "__main__":
    print("\nExample 1: Unifying P(x, a, b) and P(y, z, b)\n")
    term1 = ('P', ['x', 'a', 'b'])
    term2 = ('P', ['y', 'z', 'b'])
    try:
        substitution = unify(term1, term2)
        print("\nFinal Unification Result:", substitution)
    except UnificationError as e:
        print("Unification failed:", e)

    print("\nExample 2: Unifying P(x, f(Y)) and P(Z, f(a))\n")
    term3 = ('P', ['x', ('f', ['Y'])])
    term4 = ('P', ['Z', ('f', ['a'])])
    try:
        substitution = unify(term3, term4)
        print("\nFinal Unification Result:", substitution)
    except UnificationError as e:
        print("Unification failed:", e)
```

Snapshot of the output:

```
Example 1: Unifying P(x, a, b) and P(y, z, b)

Unifying: ('P', ['x', 'a', 'b']) and ('P', ['y', 'z', 'b'])
 Unifying: x and y
 Unifying variable x with y
  Substituting x with y
 Unifying: a and z
 Unifying variable a with z
  Substituting a with z
 Unifying: b and b
 Terms are identical, no substitution.

Final Unification Result: {'x': 'y', 'a': 'z'}

Example 2: Unifying P(x, f(Y)) and P(Z, f(a))

Unifying: ('P', ['x', ('f', ['Y'])]) and ('P', ['Z', ('f', ['a'])])
 Unifying: x and Z
 Unifying variable x with Z
  Substituting x with Z
 Unifying: ('f', ['Y']) and ('f', ['a'])
  Unifying: Y and a
  Unifying variable a with Y
   Substituting a with Y

Final Unification Result: {'x': 'Z', 'a': 'Y'}
```


OUTPUT:
Example 1: Unifying P(x, a, b) P(y, z, b)
Unifying ('P', ['x', 'a', 'b'] and ('P', ['y', 'z'
Unifying: x and y
Substituting x and y
Unifying: a and z


Unifying variable a with z
Substituting a with z
Unifying b and b
final result: {'x': 'y', 'a': 'z'}

Example: ('P', ['x', 'x', ('P', ['x', ('P', ['y'])
('P', ['z', ('f', ['a'])])])
Unifying: x and z
Substituting x and z
Unifying: ('P', ['y']) and ('f', ['a
Unifying: y and a
Unifying variable a with y
final result: {'x', 'z': 'a', 'y': 'a'}