AISC

Class: B.E COMP

Experiment 06: Solve a reasoning problem using unification

Learning Objective:

Basic Experiments

Solve a reasoning problem using unification

Tools: Python

Theory:

What is Unification?

Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.

It takes two literals as input and makes them identical using substitution.

Let Ψ1 and Ψ2 be two atomic sentences and σ **be a unifier such that,** $\Psi 1\sigma = \Psi 2\sigma$, **then it can be expressed as** UNIFY(Ψ1, Ψ2).

Example: Find the MGU for Unify{King(x), King(John)}

Let Ψ1 = King(x), Ψ2 = King(John),

Substitution θ = {John/x} **is a unifier for these atoms and applying this substitution, and both expressions will be identical.**

The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist).

Unification is a key component of all first-order inference algorithms.

It returns fail if the expressions do not match with each other.

The substitution variables are called Most General Unifier or MGU.

E.g. **Let's say there are two different expressions,** P(x, y), and P(a, f(z)).

In this example, we need to make both above statements identical to each other. For this, we will perform the substitution.

P(x, y)......... (i)
P(a, f(z))......... (ii)

Substitute x with a, and y with f(z) in the first expression, and it will be represented as a/x **and f(z)/y.**

With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: [a/x, f(z)/y].

Conditions for Unification:

Following are some basic conditions for unification:

Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.

Number of Arguments in both expressions must be identical.

Unification will fail if there are two similar variables present in the same expression.

Unification Algorithm:

Algorithm: Unify(Ψ1, Ψ2)

Step. 1: If Ψ1 or Ψ2 is a variable or constant, then:

a) If Ψ1 or Ψ2 are identical, then return NIL.

b) Else if Ψ1is a variable,

a. then if Ψ1 occurs in Ψ2, then return FAILURE

b. Else return { (Ψ2/ Ψ1)}.

c) Else if Ψ2 is a variable,

a. If Ψ2 occurs in Ψ1 then return FAILURE,

b. Else return {(Ψ1/ Ψ2)}.

d) Else return FAILURE.

Step.2: If the initial Predicate symbol in Ψ1 and Ψ2 are not same, then return FAILURE.

Step. 3: IF Ψ1 and Ψ2 have a different number of arguments, then return FAILURE.

Step. 4: Set Substitution set(SUBST) to NIL.

Step. 5: For i=1 to the number of elements in Ψ1.

a) Call Unify function with the ith element of Ψ1 and ith element of Ψ2, and put the result into S.

b) If S = failure then returns Failure

c) If S ≠ NIL then do,

a. Apply S to the remainder of both L1 and L2.

b. SUBST= APPEND(S, SUBST).

Step.6: Return SUBST.

Implementation of the Algorithm

Step.1: **Initialize the substitution set to be empty.**

Step.2: **Recursively unify atomic sentences:**

Check for Identical expression match.

If one expression is a variable vi, and the other is a term ti which does not contain variable vi, then:

Substitute ti / vi in the existing substitutions

Add ti /vi to the substitution setlist.

If both the expressions are functions, then function name must be similar, and the number of arguments must be the same in both the expression.

For each pair of the following atomic sentences find the most general unifier (If exist).

1. Find the MGU of {p(f(a), g(Y)) and p(X, X)}

Sol: S0 => Here, Ψ1 = p(f(a), g(Y)), and Ψ2 = p(X, X)
SUBST θ= {f(a) / X}
S1 => Ψ1 = p(f(a), g(Y)), and Ψ2 = p(f(a), f(a))
SUBST θ= {f(a) / g(y)}, Unification failed**.**

Unification is not possible for these expressions.

2. Find the MGU of {p(b, X, f(g(Z))) and p(Z, f(Y), f(Y))}

Here, Ψ1 = p(b, X, f(g(Z))) , and Ψ2 = p(Z, f(Y), f(Y))
S0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}
SUBST θ={b/Z}

S1 => { p(b, X, f(g(b))); p(b, f(Y), f(Y))}
SUBST θ={f(Y) /X}

S2 => { p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))}
SUBST θ= {g(b) /Y}

S2 => { p(b, f(g(b)), f(g(b)); p(b, f(g(b)), f(g(b))} Unified Successfully.
And Unifier = { b/Z, f(Y) /X , g(b) /Y}**.**

3. Find the MGU of {p (X, X), and p (Z, f(Z))}

Here, Ψ1 = {p (X, X), and Ψ2 = p (Z, f(Z))
S0 => {p (X, X), p (Z, f(Z))}
SUBST θ= {X/Z}
S1 => {p (Z, Z), p (Z, f(Z))}
SUBST θ= {f(Z) / Z}, Unification Failed**.**

Hence, unification is not possible for these expressions.

4. Find the MGU of UNIFY(prime (11), prime(y))

Here, Ψ1 = {prime(11) , and Ψ2 = prime(y)}
S0 => {prime(11) , prime(y)}
SUBST θ= {11/y}

S1 => {prime(11) , prime(11)} , Successfully unified.
Unifier: {11/y}.

5. Find the MGU of Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)}

Here, Ψ1 = Q(a, g(x, a), f(y)), and Ψ2 = Q(a, g(f(b), a), x)
S0 => {Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)}
SUBST θ= {f(b)/x}
S1 => {Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))}

SUBST θ= {b/y}
S1 => {Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))}, Successfully Unified.

Unifier: [a/a, f(b)/x, b/y].

6. UNIFY(knows(Richard, x), knows(Richard, John))

Here, Ψ1 = knows(Richard, x), and Ψ2 = knows(Richard, John)
S0 => { knows(Richard, x); knows(Richard, John)}
SUBST θ= {John/x}
S1 => { knows(Richard, John); knows(Richard, John)}, Successfully Unified.
Unifier: {John/x}.

Implimentation:

SOURCE CODE:

```
from unification import *

@unifiable
class Account(object):
    def __init__(self, id, name, balance):
        self.id = id
        self.name = name
        self.balance = balance

data = [Account(1, 'John', 100),
        Account(2, 'Josh', 0),
        Account(2, 'Gwen', 0),
        Account(2, 'Dany', 400),
        Account(2, 'Rose', 500)]


id, name, balance = var('id'), var('name'), var('balance')
a=[unify(Account(id, name, balance), acct) for acct in data]
print("\nEntered data: \n")
print(*a,sep='\n')
b=[unify(Account(id, name, 0), acct) for acct in data]
```

```
print("\nApplying Unification(Account(id, name, 0), acct): \n")
print(*b,sep='\n')
```

OUTPUT:

```
Entered data:

{~id: 1, ~name: 'John', ~balance: 100}
{~id: 2, ~name: 'Josh', ~balance: 0}
{~id: 2, ~name: 'Gwen', ~balance: 0}
{~id: 2, ~name: 'Dany', ~balance: 400}
{~id: 2, ~name: 'Rose', ~balance: 500}

Applying Unification(Account(id, name, 0), acct):

False
{~id: 2, ~name: 'Josh'}
{~id: 2, ~name: 'Gwen'}
False
False
```

Learning Outcomes: Students should have the ability to

LO1: Understand the problem formulation

Course Outcomes: Upon completion of the course students will be able to understand problem formulation in IIS.

Conclusion: Hence we have successfully understood the concept of unification and also implemented the same using a reasoning problem in python.

Viva Questions:

Ques.1 What do you understand by Unification?

Ques. 2. Explain the basic steps in unification?

Ques. 3. Write types of problem discussed in detail.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				