AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY DHAKA-1208, BANGLADESH.

Department of Computer Science and Engineering
Spring 2019

Program: Bachelor of Science in Computer Science and Engineering
Course No: CSE 4108
Course Title: Artificial Intelligence Lab

Term Project: 01
Topic: Unification with Predicate Logic

Date of Submission: September 22, 2019

Submitted to:
Dr. S.M. Abdullah Al-Mamun Professor, Department of CSE, AUST.

Md. Siam Ansary
Adjunct Faculty, Department of CSE, AUST.

Submitted By:
Name: Syed Sanzam
ID: 16.01.04.042
Group: A2

Preface:

In computer science and logic, unification is the algorithmic procedure used in solving equations involving symbolic expressions. It is the process of making two different logical atomic expressions identical by finding a substitution. In other words, by replacing certain sub-expression variables with other expressions, unification tries to identify two symbolic expressions.

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:

The problem of **Unification** is the following: given two atoms, determine if they unify, and, if they do, return an MGU of them.

```
Procedure Unify($t_1$,$t_2$)
    Inputs
        $t_1$,$t_2$: atoms Output
        most general unifier of $t_1$ and $t_2$ if it exists or $\bot$ otherwise
    Local
        $E$: a set of equality statements
        $S$: substitution
    $E \leftarrow \{t_1=t_2\}$
    $S=\{\}$
    while ($E \neq \{\}$ )
        select and remove $x=y$ from $E$
        if ($y$ is not identical to $x$) then
            if ($x$ is a variable) then
                replace $x$ with $y$ everywhere in $E$ and $S$
                $S \leftarrow \{x/y\} \cup S$
            else if ($y$ is a variable) then
                replace $y$ with $x$ everywhere in $E$ and $S$
                $S \leftarrow \{y/x\} \cup S$
            else if ($x$ is $f(x_1,\ldots,x_n)$ and $y$ is $f(y_1,\ldots,y_n)$) then
                $E \leftarrow E \cup \{x_1=y_1,\ldots,x_n=y_n\}$
            else
                return $\bot$
    return $S$
```

Here, *E* is a set of equality statements implying the unification, and *S* is a set of equalities of the correct form of a substitution. In this algorithm, if *x/y* is in the substitution *S*, then, by construction, *x* is a variable that does not appear elsewhere in *S* or in *E*. In *line 19*, *x* and *y* must have the same predicate and the same number of arguments; otherwise the unification fails.

Python Implementation:

unification.py

```python
# Author : Syed Sanzam
# Topic : Unification of Predicate Logic
# Date: 21.09.19

def create():
    global names
    global totArgs
    global args

    names = []
    totArgs = []
    args = []

    for i in range(2):
        t = str(input('Name: '))
        names.append(t)
        t = int(input('Total Number of Arguments: '))
        totArgs.append(t)

        l = []
        for j in range(totArgs[i]):
            t = str(input("Args: "))
            l.append(t)

        args.append(l)
        print('\n')


def display():
    print("The Expressions are : ")
    for i in range(2):
        print(names[i], end = "")
        print('(', end = "")
        for j in range(totArgs[i]):
            print(args[i][j], end="")
            if(j != totArgs[i] - 1):
                print(',',end = "")
        print(')',end = "")
        print('\n')
```

```
def isUnifiable():
    sameNames = False
    sameArgs = False

    for i in range(len(names) - 1):
        if(names[i] == names [i + 1]):
            sameNames = True
            break

    for i in range(len(totArgs) - 1):
        if(totArgs[i] == totArgs[i+1]):
            sameArgs = True
            break

    if(sameNames and sameArgs):
        return True
    else:
        return False


def unify():
    global mgu # Most General Unifier
    global substitution # Set of substitution
    global equalityStatements # Set of Equality Statements

    equalityStatements = []
    substitution = []

    for i in range(totArgs[0]):
        l = []
        l.append(args[0][i])
        l.append(args[1][i])
        equalityStatements.append(l)

    loopCount = 0
    while(loopCount <= len(totArgs)):
        #print("While loop e dhukse!")
        l = []
        arg1 = equalityStatements[0][0]
        arg2 = equalityStatements[0][1]
        del equalityStatements[0]

        l.append(arg1)
        l.append(arg2)

        for i in range(len(equalityStatements)):
            for j in range(len(equalityStatements)):
                if(equalityStatements[i][j] == arg1):
                    equalityStatements[i][j] = arg2

        for i in range(len(substitution)):
            for j in range(len(substitution)):
                if(substitution[i][j] == arg1):
                           substitution[i][j] = arg2

               substitution.append(l)
               loopCount = loopCount + 1




```

```
def printResult():
    file = open("database.txt","a") #To Store the result of the Unification
    print("Most General Unifier (MGU) is : ", end = " ")
    print('[', end = "")
    file.write("[")
    for i in range(len(substitution)):
        print(substitution[i][0] + "/" + substitution[i][1], end = "")
        file.write(substitution[i][0])
        file.write("/")
        file.write(substitution[i][1])
        if(i != len(substitution) - 1):
            print(',',end = " ")
            file.write(",")
    print(']', end = "")
    file.write("]")
    file.write("\n")
    print('\n')




def evaluate():
    if(isUnifiable()):
        unify()
        printResult()
    else:
        print("The Expressions are not Unifiable")



#Main
create()   # Takes input and creates the expressions
display()  # Displays the input expressions
evaluate() # Evaluates and prints the result
```

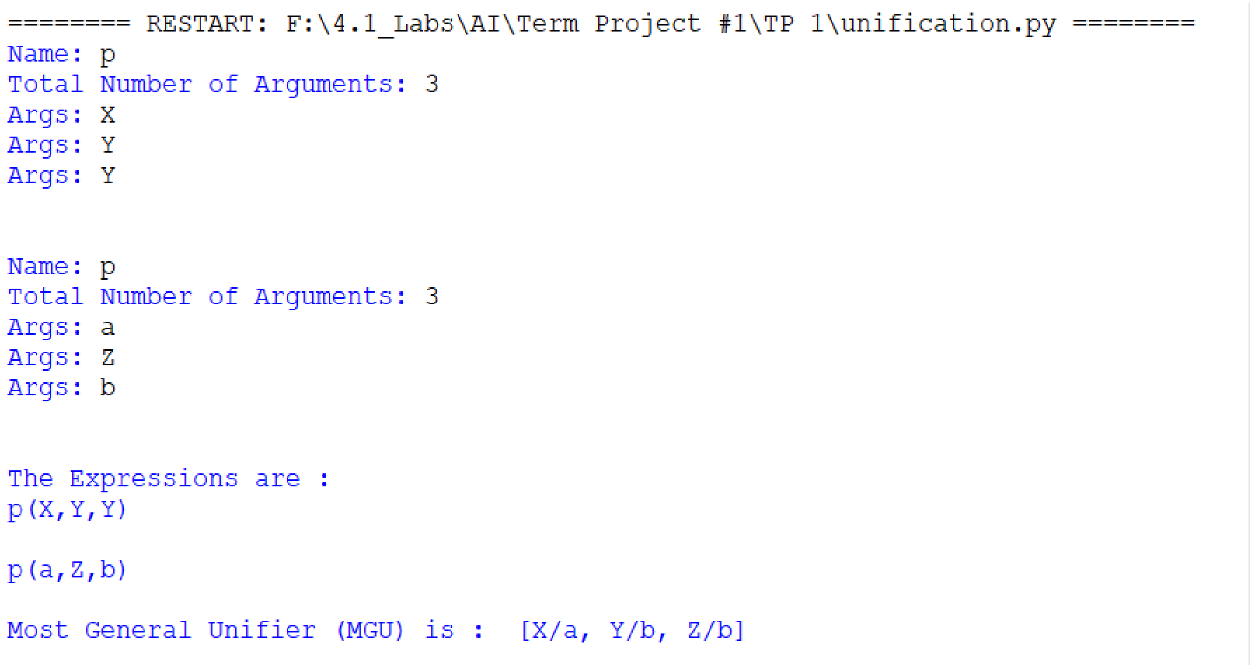
Algorithm Overview:

This implementation was solely based on the aforementioned algorithm and seems to work just fine. If we consider the following example,

Suppose we want to unify *p(X,Y,Y)* with *p(a,Z,b)*.
Initially *E* is *{p(X,Y,Y)=p(a,Z,b)}*.
The first time through the while loop,
E becomes *{X=a,Y=Z,Y=b}*. Suppose *X=a* is selected next.
Then *S* becomes *{X/a}* and *E* becomes *{Y=Z,Y=b}*. Suppose *Y=Z* is selected.
Then *Y* is replaced by *Z* in *S* and *E*. *S* becomes *{X/a,Y/Z}* and *E* becomes *{Z=b}*.
Finally *Z=b* is selected, *Z* is replaced by *b*, *S* becomes *{X/a,Y/b,Z/b}*, and *E* becomes empty. The substitution then, is returned as an MGU.
So, the Most General Unifier for the this case will be, *{X/a,Y/b,Z/b}*.

Input and Output:

If we run and execute the *unification.py* and put the expressions in terms of **Names, Number of Arguments** and **Arguments**, we will see,



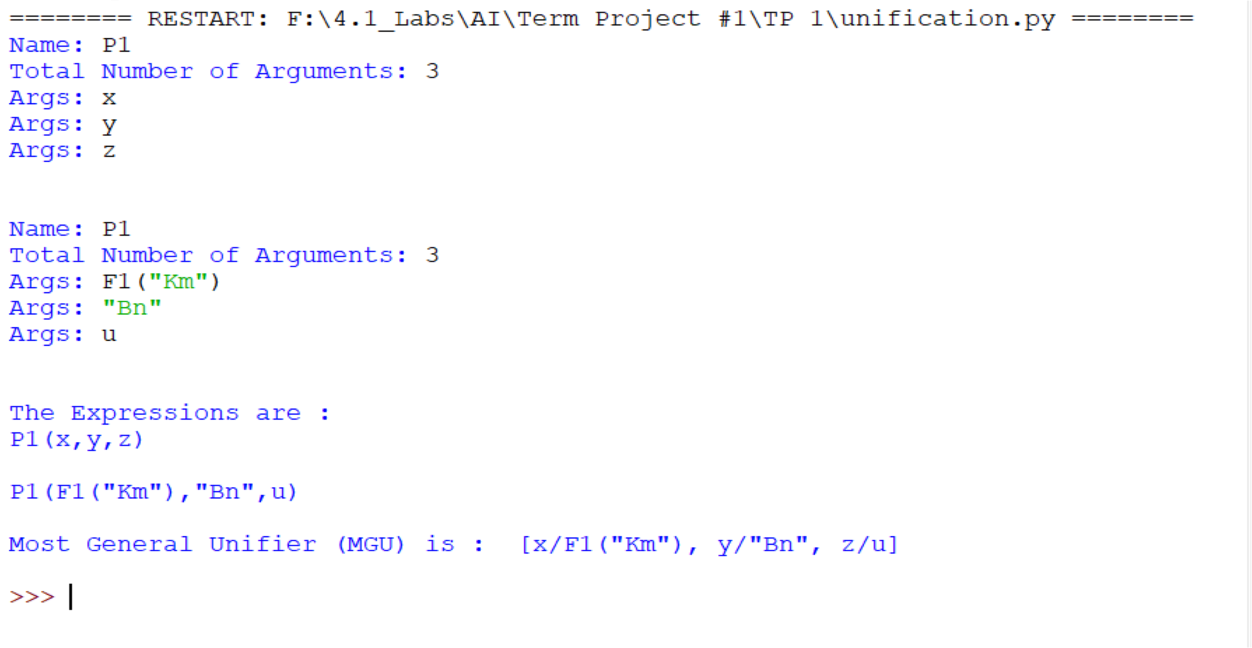
```
======== RESTART: F:\4.1_Labs\AI\Term Project #1\TP 1\unification.py ========
Name: p
Total Number of Arguments: 3
Args: X
Args: Y
Args: Y


Name: p
Total Number of Arguments: 3
Args: a
Args: Z
Args: b


The Expressions are :
p(X,Y,Y)

p(a,Z,b)

Most General Unifier (MGU) is :  [X/a, Y/b, Z/b]
```

So, the output of the implementation is as expected according to the algorithm.

Let us consider another example. Two same-predicate atomic sentences are given in a set, S.

S = {P1(x, y, z), P1(F1(“Km”), “Bn”, u)}

The output for this scenario will be,

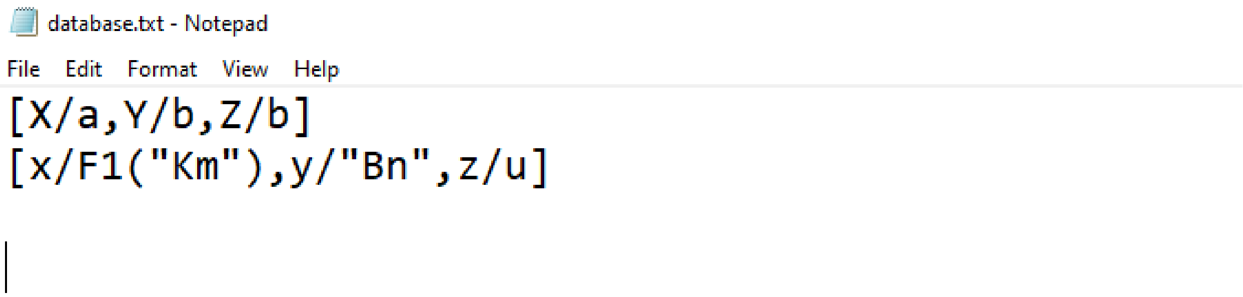


```
======== RESTART: F:\4.1_Labs\AI\Term Project #1\TP 1\unification.py ========
Name: P1
Total Number of Arguments: 3
Args: x
Args: y
Args: z


Name: P1
Total Number of Arguments: 3
Args: F1("Km")
Args: "Bn"
Args: u


The Expressions are :
P1(x,y,z)

P1(F1("Km"),"Bn",u)

Most General Unifier (MGU) is :  [x/F1("Km"), y/"Bn", z/u]

>>> |
```

Implementation Overview:

The implementation is divided into multiple functions. The *create()* and *display()* functions are entirely used for taking inputs and displaying the expressions for clarification purpose. The most important function, *unify()* starts finding substitution for certain sub-expression variables. This function only executes if the conditions are met, which is determined from the *isUnifiable()* function. Finally, the *printResult()* function displays the desired Most General Unifier (MGU) and stores the result in *database.text* file for further inspection.

The Contents of *database.txt* after executing the program for the aforementioned scenarios are,



```
[X/a,Y/b,Z/b]
[x/F1("Km"),y/"Bn",z/u]
```

References:

- https://www.javatpoint.com/ai-unification-in-first-order-logic
- https://www.techopedia.com/definition/22735/unification
- https://artint.info/html/ArtInt_287.html

My GitHub:

https://github.com/sanzamsyed/Artificial-Intelligence