Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No. : CSE4108

Course Title: Artificial Intelligence Lab

Term Assignment: 01

Submitted by

Name: Rizeya Rabbi Reyad

ID: 160104082

Lab Group: B2

Question: Reasoning with Propositional Logic

Answer:

Introduction:

We have some known data, which are the fact and rules. From rules it is possible to identify new solution or fact. So, bellow the code we will query a value to find a solution from the fact and rules.

Major steps of processing:

1. At first the program will try to find the fact in the initial KB.
2. Then it will try to find new facts or rules using Modus Ponens, Modus Tollens and Simplification etc.
3. Then it will check if the given input can be derived or not.
4. If it finds the expected fact in the knowledge Base, it will give the output: "X is found in KB"
5. If it does not find the expected fact **say X** in the appended knowledge Base, it will give output: "X is not found in KB"

KB that has been used:

1. A
2. B
3. C
4. D
5. P -> Q
6. C AND L ->P
7. D AND M ->P
8. B AND L ->M
9. A AND P ->L
10. A AND B ->L
11. A AND D ->G
12. G AND B ->C

Python code:

```
while True:
    fact = ['A', 'B', 'C', 'D']
lists = [('P', 'Q')]
    rules = [('C', 'L', 'P'), ('D', 'M', 'P'), ('B', 'L', 'M'),
             ('A', 'P', 'L'), ('A', 'B', 'L'), ('A', 'D', 'G'), ('G', 'B', 'C')]
     with open('output.txt', 'w') as
f1:
        loop1,   loop2   =   0,   0
result = 'a'         x = input('Enter
a query: ')
                 def
simplification():
            global loop1, loop2, result
j = 0             index = len(fact)-1
index2 = len(fact)             while j
< index:
                print(f'check {fact[j]}')
if (loop1 == 1):
                    break                 k = 0                 while k <
index2:                     if (loop2 == 1):
loop1 = 1                         break                     for l in
range(len(rules)):                         if ((fact[j] == rules[l][0]) &
(fact[k] == rules[l][1])):                             if (rules[l][2] not in
fact):                                  fact.append(rules[l][2])
index = index+1                                  index2 = index2 + 1
print(fact, end='\n', file=f1)                                  if
(rules[l][2] == x):
                                     loop2 = 1
result = x
break                         else:
continue                     k += 1
j += 1
         def
modPon():
            global loop1, result
            for j in range(len(fact)):
if (loop1 == 1):
                    break                 for l in
range(len(lists)):                     if (fact[j] ==
lists[l][0]):                         if (lists[l][1] not
in fact):
fact.append(lists[l][1])
print(fact, end='\n', file=f1)
if (lists[l][1] == x):
```

```
                                loop1 = 1
result = x
break
         def
modTol():
            global loop1, result
for j in range(len(fact)):
if (loop1 == 1):
                    break                 for l in
range(len(lists)):                     if (fact[j] ==
'!'+lists[l][1]):                         if
('!'+lists[l][0] not in fact):
fact.append('!'+lists[l][0])
print(fact, end='\n', file=f1)
if ('!'+lists[l][0] == x):
                                loop1 = 1
result = x
break         for j in range(len(fact)):
if (fact[j] == x):                 result
= x         if (result != x):
modPon()         if (result != x):
modTol()         if (result != x):
simplification()
     if (result ==
x):
        print(f'\n{x} is found in KB')
print(f'The facts are {fact}')
else:
        print(f'\n{x} is not found in KB')     decision =
input('Do you want to check again? (y/n) :')     if
(decision == 'n'):
            break
```

Output:

```
Enter a query: M
check A
check B
check C

M is found in KB
The facts are ['A', 'B', 'C', 'D', 'L', 'G', 'M']
Do you want to check again? (y/n) :y
Enter a query: L
check A
check B

L is found in KB
The facts are ['A', 'B', 'C', 'D', 'L']
Do you want to check again? (y/n) :y
Enter a query: G
check A
check B

G is found in KB
The facts are ['A', 'B', 'C', 'D', 'L', 'G']
Do you want to check again? (y/n) :y
Enter a query: C

C is found in KB
The facts are ['A', 'B', 'C', 'D']
Do you want to check again? (y/n) :n
E:\4.1 lecture\AI\Lab\T-Assignment>
```

Saved data in output.txt file:

output.txt - Notepad

File Edit Format View Help

```
['A', 'B', 'C', 'D', 'L']
['A', 'B', 'C', 'D', 'L', 'G']
['A', 'B', 'C', 'D', 'L', 'G', 'M']
```