Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE4108

Artificial Intelligence

TERM PROJECT 01

Topic no:05

Topic Name: Resolution with predicate logic

Submitted By:

Mayisha Farzana 16.02.04.028

Date of Submission: **30 August, 2020**

Resolution with predicate logic:

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.

Steps for Resolution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

Example of conversion:

i) KB in Natural Language:

1. Every guest receives at least one gift.

2. Karim is a guest.

ii) KB in in FOL:

1. $\forall x\ (Guest(x) \Rightarrow \exists y\ (Gift(y) \wedge Receives(x, y)))$

(It implies that there is at least one gift and receives gift)

2. Guest(Karim)

iii) Conversion to CNF:

I. Standardize variables. [Use separate variables for different quantifiers.]

II. Eliminate $\Rightarrow$ and $\Leftrightarrow$, and move $\neg$ inward.

($p \Rightarrow q$, $\neg p \Rightarrow q$ equivalent)

$\forall x\ (\neg Guest(x) \vee (\exists y(Gift(y) \wedge Receives(x, y))))$

III. Skolemize

$\forall x\ (\neg Guest(x) \vee (Gift(GiftFor(x)) \wedge Receives(x, GiftFor(x))))$

IV. Drop all $\forall$s.

$(\neg Guest(x) \vee (Gift(GiftFor(x)) \wedge Receives(x, GiftFor(x))))$

V. Simplify further to get CNF.

$(\neg Guest(x) \vee Gift(GiftFor(x))) \wedge (\neg Guest(x) \vee Receives(x, GiftFor(x)))$

(Literal or literal) and (literal or literal) $\Rightarrow$ CNF

iv) KB in CNF of FOL:

1. $\neg Guest(x) \vee Gift(GiftFor(x))$

2. $\neg Guest(x) \vee Receives(x, GiftFor(x))$

3. Guest(Karim)

iv) KB in CNF of FOL:

1. $\neg Guest(x) \vee Gift(GiftFor(x))$

2. $\neg Guest(x) \vee Receives(x, GiftFor(x))$

3. Guest(Karim)

d) Query: Does Karim receive a gift?

$\exists x\ (Gift(x) \wedge Receives(Karim, x))$

i) To answer / prove, add the negation of the sentence to the KB and try to derive a contradiction.

ii) Equivalently, Add the CNF of the negation of the query to the KB, and try to derive an empty clause ([]) by applying the Resolution rule repeatedly.

iii) If [] can't be derived, it means that the sentence (query) is False. This is known as the Resolution-Refutation completeness of KB.

e) Conversion of the negation of the query:

In this statement, we will apply negation to the conclusion statements, which will be written as $\neg Gift(y) \vee \neg Receives(Karim, y)$

$\exists x$ (Gift(x) $\wedge$ Receives(Karim, x))

$\Rightarrow \neg\exists x$ (Gift(x) $\wedge$ Receives(Karim, x))

$\Rightarrow \forall x \neg$(Gift(x) $\wedge$ Receives(Karim, x))

$\Rightarrow \forall x(\neg$Gift(x) $\vee \neg$Receives(Karim, x))

$\Rightarrow \neg$Gift(x) $\vee \neg$Receives(Karim, x)

Code Start:

1. $\neg$Guest(x) $\vee$ Gift(GiftFor(x))

2. $\neg$Guest(x) $\vee$ Receives(x, GiftFor(x))

3. Guest(Karim)

T1.$\neg$Gift(y) $\vee \neg$Receives(Karim, y)

f) Finding the answer to the query:

Resolving T1 and 1 with θ={y/GiftFor(x)}

T2. $\vee \neg$Receives(Karim, GiftFor(x)))$\vee \neg$Guest(x)

Resolving T1 and 2 with θ= {x / Karim, y / GiftFor(Karim)}

T3. $\neg$Gift(GiftFor(Karim) $\vee \neg$Guest(Karim)

Resolving T2 and 2 with θ= {x / Karim}

T4. $\neg$Guest(Karim)

Resolving T2 and 3 with θ= {x / Karim}

T5. $\neg$Receives (Karim,GiftFor(Karim))

Resolving T3 and 1 with θ= {x / Karim}

T6. ~~$\neg$Guest(Karim)~~

Resolving T3, 3 with θ={ }

T7. ¬Gift(GiftFor(Karim))

Resolving T4 and 3 with θ={ }

T8. [ ] (stop)

Answer ‘Yes’.

As an empty clause is resolved, the sentence(query) is proved true, that is the answer is

‘Yes, Karim Receives a gift’.

Python Code:

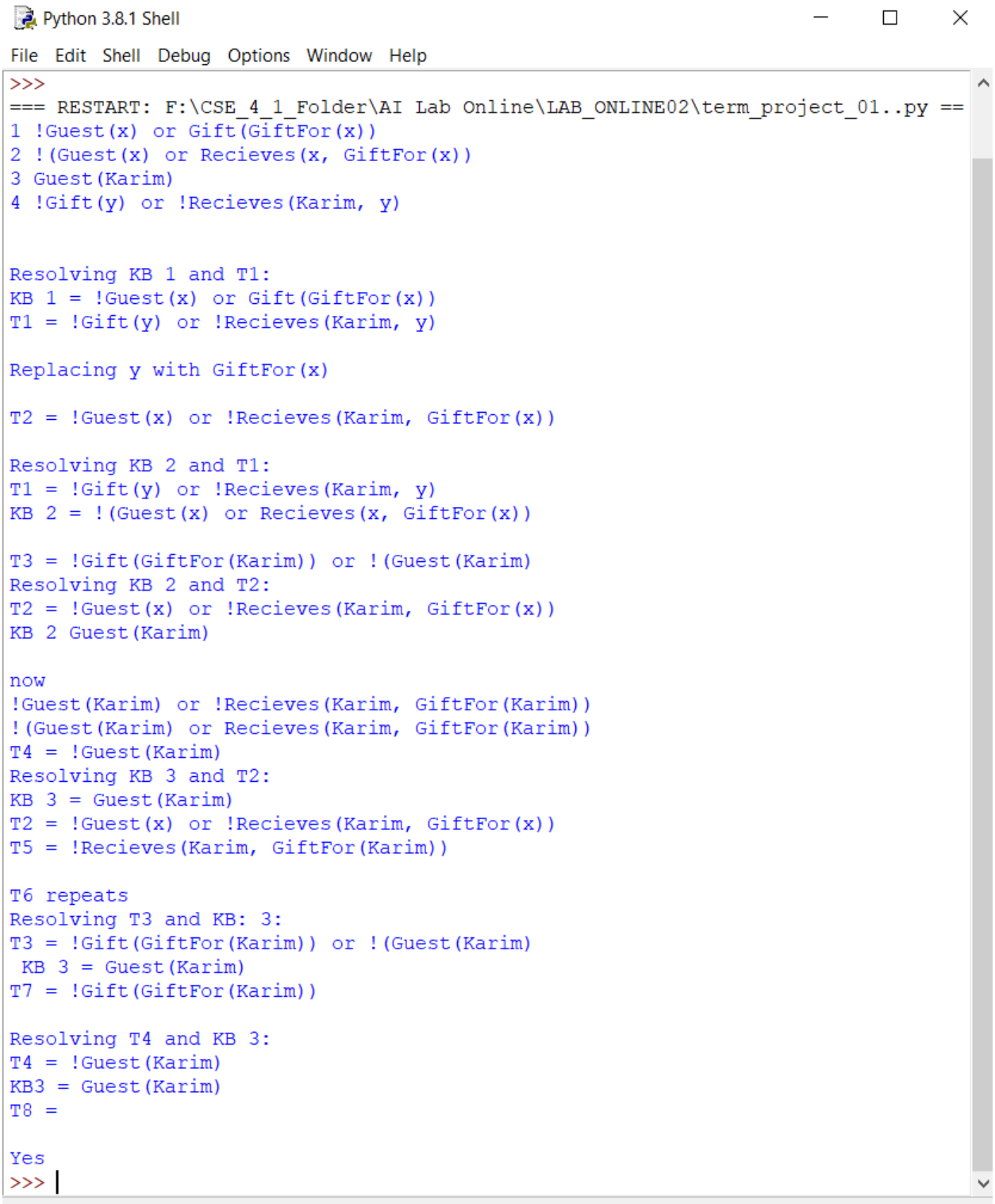
```
kb = ['!Guest(x) or Gift(GiftFor(x))',
      '!(Guest(x) or Recieves(x, GiftFor(x))',
      'Guest(Karim)',
      '!Gift(y) or !Recieves(Karim, y)']
def predicate_logic():
    for i in range(len(kb)):
        print(str(i + 1) + " " + kb[i])
    print("\n")
    T1 = kb[3]
    tmp = T1
    one = kb[0]
    print("Resolving KB 1 and T1: ")
    print("KB 1 = " + one + "\nT1 = " + T1 + "\n")
    print("Replacing y with GiftFor(x)\n")
```

```
tmp = tmp.replace("y", "GiftFor(x)")

one = one.replace(" or Gift(GiftFor(x))", "")

tmp = tmp.replace("!Gift(GiftFor(x)) or ", "")

T2 = one + " or " + tmp

print("T2 = " + T2 + "\n")

print("Resolving KB 2 and T1:")

two = kb[1]

print("T1 = " + T1 + "\n" + "KB 2 = " + two + "\n")

tmp = T1

tmp = tmp.replace("y", "GiftFor(x)")

tmp = tmp.replace("x", "Karim")

two = two.replace("x", "Karim")

tmp = tmp.replace(" or !Recieves(Karim, GiftFor(Karim))", "")

two = two.replace(" or Recieves(Karim, GiftFor(Karim))", "")

T3 = tmp + " or " + two

print("T3 = " + T3)

print("Resolving KB 2 and T2:")

print("T2 = " + T2 + "\n" + "KB 2 " + kb[2] + "\n")

tmp = T2;

two = kb[1]

tmp = tmp.replace("x", "Karim")

two = two.replace("x", "Karim")
```

```
print("now\n" + tmp + "\n" + two)

tmp = tmp.replace(" or !Recieves(Karim, GiftFor(Karim))", "")

two = two.replace(" or Recieves(Karim, GiftFor(Karim))", "")

T4 = tmp

print("T4 = " + T4)

print("Resolving KB 3 and T2:")

print("KB 3 = " + kb[2])

print("T2 = " + T2)

tmp = T2

tmp = tmp.replace("x", "Karim")

tmp = tmp.replace("!Guest(Karim) or ", "")

T5 = tmp

print("T5 = " + T5 + "\n")

print("T6 repeats")

print("Resolving T3 and KB: 3:")

print("T3 = " + T3 + "\n KB 3 = " + kb[2])

tmp = T3

tmp = tmp.replace(" or !(Guest(Karim)", "")

T7 = tmp

print("T7 = " + tmp + "\n")

print("Resolving T4 and KB 3:")

print("T4 = " + T4 + "\nKB3 = " + kb[2])
```

```
    T8 = ""
    print("T8 = " + T8 + "\n")
    if (len(T8) <= 0):
        return True
    return False
#main
if (predicate_logic()):
    print("Yes")
else :
    print("No")
```

OUTPUT:


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
=== RESTART: F:\CSE_4_1_Folder\AI Lab Online\LAB_ONLINE02\term_project_01..py ==
1 !Guest(x) or Gift(GiftFor(x))
2 !(Guest(x) or Recieves(x, GiftFor(x))
3 Guest(Karim)
4 !Gift(y) or !Recieves(Karim, y)
Resolving KB 1 and T1:
KB 1 = !Guest(x) or Gift(GiftFor(x))
T1 = !Gift(y) or !Recieves(Karim, y)
Replacing y with GiftFor(x)
T2 = !Guest(x) or !Recieves(Karim, GiftFor(x))
Resolving KB 2 and T1:
T1 = !Gift(y) or !Recieves(Karim, y)
KB 2 = !(Guest(x) or Recieves(x, GiftFor(x))
T3 = !Gift(GiftFor(Karim)) or !(Guest(Karim)
Resolving KB 2 and T2:
T2 = !Guest(x) or !Recieves(Karim, GiftFor(x))
KB 2 Guest(Karim)
now
!Guest(Karim) or !Recieves(Karim, GiftFor(Karim))
!(Guest(Karim) or Recieves(Karim, GiftFor(Karim))
T4 = !Guest(Karim)
Resolving KB 3 and T2:
KB 3 = Guest(Karim)
T2 = !Guest(x) or !Recieves(Karim, GiftFor(x))
T5 = !Recieves(Karim, GiftFor(Karim))
T6 repeats
Resolving T3 and KB: 3:
T3 = !Gift(GiftFor(Karim)) or !(Guest(Karim)
KB 3 = Guest(Karim)
T7 = !Gift(GiftFor(Karim))
Resolving T4 and KB 3:
T4 = !Guest(Karim)
KB3 = Guest(Karim)
T8 =
Yes
>>>