



Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No. : CSE4108
Course Name : Artificial Intelligence Lab

Term Assignment : 01

Submitted By:

Name : Faiza Anan Noor
ID No. : 17 01 04 093
Session : Spring - 2020
Section : B (B2)

Predicate:

A predicate is an expression of one or more variables defined on some specific domain. A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable.

Predicate Logic:

Predicate Logic deals with predicates, which are propositions containing variables.

Examples of predicates:

1. Let $E(x, y)$ denote " $x = y$ "
2. Let $X(a, b, c)$ denote " $a + b + c = 0$ "
3. Let $M(x, y)$ denote " x is married to y "

Resolution:

Resolution is a rule of inference leading to a refutation theorem-proving technique for sentences in propositional logic and first-order logic. In other words, iteratively applying the resolution rule in a suitable way allows for telling whether a propositional formula is satisfiable and for proving that a first-order formula is unsatisfiable. Attempting to prove a satisfactory first-order formula as unsatisfiable may result in a nonterminating computation; this problem doesn't occur in propositional logic. The clause produced by a resolution rule is sometimes called a resolvent.

Resolution is a rule of inference resulting sentences in PL and FOL/FOPL. We have to worry about the arguments to predicates, so it is harder to know when two literals match and can be used by resolution. For example, does the literal Brother (Bill, Chelsea) match Brother(x, y)? The answer depends on how we substitute values for variables (unification).

Resolution Rule:

The resolution rule in propositional logic is a single valid inference rule that produces a new clause implied by two clauses containing complementary literals. A *literal* is either an atomic sentence or a negation of an atomic sentence. For example, if p is a logical constant, the following sentences are both literals.

$$p$$

$$\neg p$$

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).

Major steps of processing:

Let's now analyze the steps using a real life example.

i) KB in Natural Language:

1. Every guest receives at least one gift.
2. Faiza is a guest.

ii) KB in in FOL:

1. $\forall x (\text{Guest}(x) \Rightarrow \exists y (\text{Gift}(y) \wedge \text{Receives}(x, y)))$
(It implies that there is at least one gift and receives gift)
2. $\text{Guest}(\text{Faiza})$

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)

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

III. Skolemize

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

IV. Drop all \forall s.

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

V. Simplify further to get CNF.

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

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

iv) KB in CNF of FOL:

1. $\neg \text{Guest}(x) \vee \text{Gift}(\text{GiftFor}(x))$
2. $\neg \text{Guest}(x) \vee \text{Receives}(x, \text{GiftFor}(x))$
3. $\text{Guest}(\text{Faiza})$

iv) KB in CNF of FOL:

1. $\neg \text{Guest}(x) \vee \text{Gift}(\text{GiftFor}(x))$
 2. $\neg \text{Guest}(x) \vee \text{Receives}(x, \text{GiftFor}(x))$
 3. $\text{Guest}(\text{Karim})$
- d) Query: Does Faiza receive a gift?
- $\exists x (\text{Gift}(x) \wedge \text{Receives}(\text{Faiza}, x))$

- 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 (\square) by applying the Resolution rule repeatedly.
- iii) If \square 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 $\neg \text{Gift}(y) \vee \neg \text{Receives}(\text{Faiza}, y)$

$$\begin{aligned} &\Rightarrow \exists x (\text{Gift}(x) \wedge \text{Receives}(\text{Faiza}, x)) \\ &\Rightarrow \neg \exists x (\text{Gift}(x) \wedge \text{Receives}(\text{Faiza}, x)) \\ &\Rightarrow \forall x \neg (\text{Gift}(x) \wedge \text{Receives}(\text{Faiza}, x)) \\ &\Rightarrow \forall x (\neg \text{Gift}(x) \vee \neg \text{Receives}(\text{Faiza}, x)) \\ &\Rightarrow \neg \text{Gift}(x) \vee \neg \text{Receives}(\text{Faiza}, x) \end{aligned}$$

f) The Kb after negation of the query:

1. $\neg \text{Guest}(x) \vee \text{Gift}(\text{GiftFor}(x))$
2. $\neg \text{Guest}(x) \vee \text{Receives}(x, \text{GiftFor}(x))$
3. $\text{Guest}(\text{Faiza})$

$$\text{T1. } \neg \text{Gift}(y) \vee \neg \text{Receives}(\text{Faiza}, y)$$

g) Finding the answer to the query:

Resolving T1 and 1 with $\theta = \{y / \text{GiftFor}(x)\}$

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

Resolving T1 and 2 with $\theta = \{x / \text{Faiza}, y / \text{GiftFor}(\text{Faiza})\}$

$$\text{T3. } \neg \text{Gift}(\text{GiftFor}(\text{Faiza})) \vee \neg \text{Guest}(\text{Faiza})$$

Resolving T2 and 2 with $\theta = \{x / \text{Faiza}\}$

$$\text{T4. } \neg \text{Guest}(\text{Karim})$$

Resolving T2 and 3 with $\theta = \{x / \text{Faiza}\}$

T5. \neg Receives (Karim, GiftFor(Faiza))
Resolving T3 and 1 with $\theta = \{x / \text{Faiza}\}$

T6. \neg Guest(Faiza)
Resolving T3, 3 with $\theta = \{ \}$

T7. \neg Gift(GiftFor(Faiza))
Resolving T4 and 3 with $\theta = \{ \}$

T8. [] (stop) Answer 'Yes'. As an empty clause is resolved, the sentence(query) is proved true, that is the answer is 'Yes, Faiza Receives a gift'.

Major Steps of resolution rule:

1. From the KB in CNF of FOL,

Step 1: 1 with $\theta = \{y / \text{GiftFor}(x)\}$

Step 2: 2 with $\theta = \{x / \text{Faiza}\}$

Step 3: 2 with $\theta = \{x / \text{Faiza}, y / \text{GiftFor}(\text{Faiza})\}$

2. From the KB in CNF of FOL,

Step 1: 1 with $\theta = \{y / \text{CourseFor}(x)\}$

Step 2: 2 with $\theta = \{x / \text{Faiza}\}$

Step 3: 2 with $\theta = \{x / \text{Faiza}, y / \text{CourseFor}(\text{Faiza})\}$

An empty clause is resolved so the query is proved true.

Implementation in Python:

```
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():
```

```
    f = open("savedDb.txt", "a")  
    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("\nKB 1 = " + one + "\nT1 = " + T1 + "\n")  
    print("Replacing y with GiftFor(x)\n")
```

```
    f.write("Resolving KB 1 and T1: ")  
    f.write("\nKB 1 = " + one + "\nT1 = " + T1 + "\n")  
    f.write("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:")
```

```
    f.write("\nT2 = " + T2 + "\n")  
    f.write("Resolving KB 2 and T1:")
```

```

two = kb[1]
print("T1 = " + T1 + "\n" + "KB 2 = " + two + "\n")
f.write("T1 = " + T1 + "\n" + "KB 2 = " + two + "\n")
tmp = T1
tmp = tmp.replace("y", "GiftFor(x)")
tmp = tmp.replace("x", "Faiza")
two = two.replace("x", "Faiza")
tmp = tmp.replace(" or !Recieves(Faiza, GiftFor(Faiza))", "")
two = two.replace(" or Recieves(Faiza, GiftFor(Faiza))", "")
T3 = tmp + " or " + two
print("T3 = " + T3)
print("Resolving KB 2 and T2:")
print("T2 = " + T2 + "\n" + "KB 2 " + kb[2] + "\n")

```

```

f.write("T3 = " + T3)
f.write("Resolving KB 2 and T2:")
f.write("T2 = " + T2 + "\n" + "KB 2 " + kb[2] + "\n")

```

```

tmp = T2;
two = kb[1]
tmp = tmp.replace("x", "Faiza")
two = two.replace("x", "Faiza")

```

```

print("now\n" + tmp + "\n" + two)
f.write("now\n" + tmp + "\n" + two)
tmp = tmp.replace(" or !Recieves(Faiza, GiftFor(Faiza))", "")
two = two.replace(" or Recieves(Faiza, GiftFor(Faiza))", "")
T4 = tmp
print("T4 = " + T4)
print("Resolving KB 3 and T2:")
print("KB 3 = " + kb[2])
print("T2 = " + T2)

```

```

f.write("\nT4 = " + T4)

```



```
f.write("Resolving KB 3 and T2:")
```

```
f.write("KB 3 = " + kb[2])
```

```
f.write("T2 = " + T2)
```

```
tmp = T2
```

```
tmp = tmp.replace("x", "Faiza")
```

```
tmp = tmp.replace("!Guest(Faiza) or ", "")
```

```
T5 = tmp
```

```
print("\nT5 = " + T5 + "\n")
```

```
print("T6 repeats")
```

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

```
print("\nT3 = " + T3 + "\n KB 3 = " + kb[2])
```

```
f.write("\nT5 = " + T5 + "\n")
```

```
f.write("\nT6 repeats")
```

```
f.write("\nResolving T3 and KB: 3:")
```

```
f.write("T3 = " + T3 + "\n KB 3 = " + kb[2])
```

```
tmp = T3
```

```
tmp = tmp.replace(" or !(Guest(Faiza)", "")
```

```
T7 = tmp
```

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

```
print("\nResolving T4 and KB 3:\n")
```

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

```
f.write("\nT7 = " + tmp + "\n")
```

```
f.write("Resolving T4 and KB 3:")
```

```
f.write("\nT4 = " + T4 + "\nKB3 = " + kb[2])
```

```
T8 = ""
```

```
print("T8 = " + T8 + "\n")
```

```
f.write("\nT8 = " + T8 + "\n")
```

```
if (len(T8) <= 0):
```

```

        f.write("Yes")
        f.close()
        return True
    return False
#main
if (predicate_logic()):
    print("Yes")

else :
    print("No")

```

Input:

The following things in the knowledge base act as input for our code in python:

```

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)

```

Here the knowledge base is provided as input on which we later on apply the resolution rule.

Output:

```
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(Faiza)) or !Recieves(Karim, GiftFor(Faiza)) or !(Guest(Faiza))
Resolving KB 2 and T2:
T2 = !Guest(x) or !Recieves(Karim, GiftFor(x))
KB 2 Guest(Karim)

now
!Guest(Faiza) or !Recieves(Karim, GiftFor(Faiza))
!(Guest(Faiza) or Recieves(Faiza, GiftFor(Faiza)))
T4 = !Guest(Faiza) or !Recieves(Karim, GiftFor(Faiza))
Resolving KB 3 and T2:
KB 3 = Guest(Karim)
T2 = !Guest(x) or !Recieves(Karim, GiftFor(x))

T5 = !Recieves(Karim, GiftFor(Faiza))

T6 repeats
Resolving T3 and KB: 3:

T3 = !Gift(GiftFor(Faiza)) or !Recieves(Karim, GiftFor(Faiza)) or !(Guest(Faiza))
KB 3 = Guest(Karim)
T7 = !Gift(GiftFor(Faiza)) or !Recieves(Karim, GiftFor(Faiza))

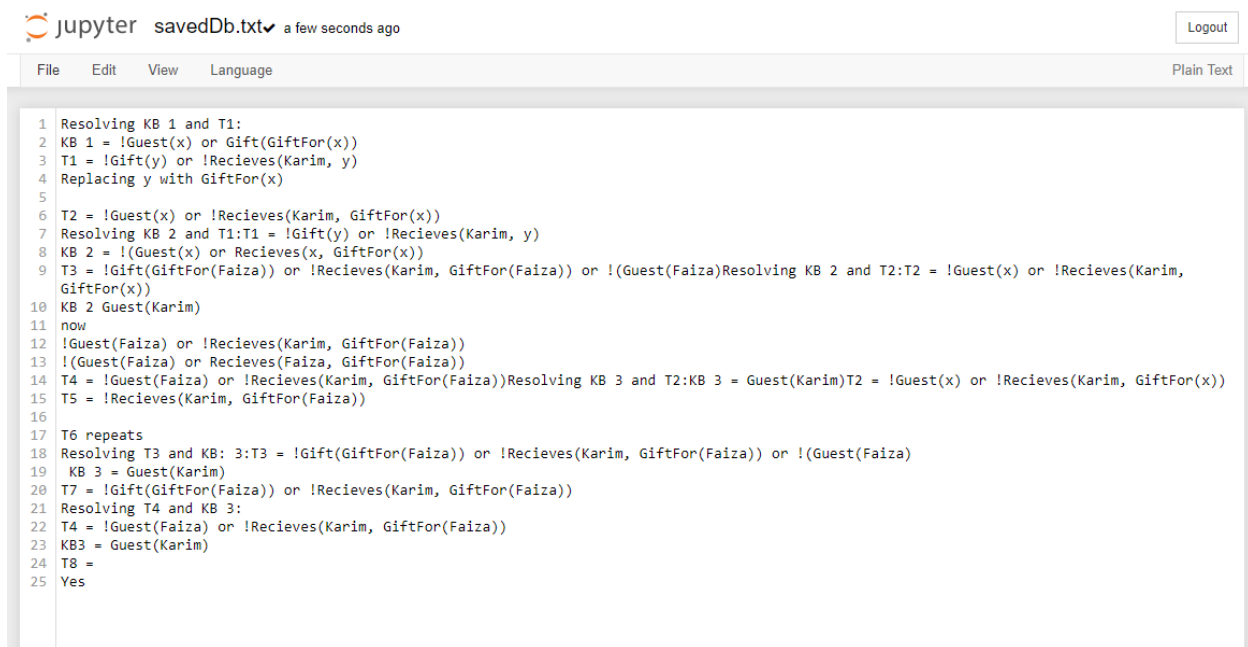
Resolving T4 and KB 3:

T4 = !Guest(Faiza) or !Recieves(Karim, GiftFor(Faiza))
KB3 = Guest(Karim)
T8 =

Yes
```

Saved Database of Intermediate activities:

Let's analyze the saved database of intermediate activities for one of our examples. We have saved the intermediate activities in one text file called "savedDb.txt" which is shown below.



```
1 Resolving KB 1 and T1:
2 KB 1 = !Guest(x) or Gift(GiftFor(x))
3 T1 = !Gift(y) or !Receives(Karim, y)
4 Replacing y with GiftFor(x)
5
6 T2 = !Guest(x) or !Receives(Karim, GiftFor(x))
7 Resolving KB 2 and T1:T1 = !Gift(y) or !Receives(Karim, y)
8 KB 2 = !(Guest(x) or Receives(x, GiftFor(x)))
9 T3 = !Gift(GiftFor(Faiza)) or !Receives(Karim, GiftFor(Faiza)) or !(Guest(Faiza)Resolving KB 2 and T2:T2 = !Guest(x) or !Receives(Karim, GiftFor(x))
10 KB 2 Guest(Karim)
11 now
12 !Guest(Faiza) or !Receives(Karim, GiftFor(Faiza))
13 !(Guest(Faiza) or Receives(Faiza, GiftFor(Faiza)))
14 T4 = !Guest(Faiza) or !Receives(Karim, GiftFor(Faiza))Resolving KB 3 and T2:KB 3 = Guest(Karim)T2 = !Guest(x) or !Receives(Karim, GiftFor(x))
15 T5 = !Receives(Karim, GiftFor(Faiza))
16
17 T6 repeats
18 Resolving T3 and KB: 3:T3 = !Gift(GiftFor(Faiza)) or !Receives(Karim, GiftFor(Faiza)) or !(Guest(Faiza)
19 KB 3 = Guest(Karim)
20 T7 = !Gift(GiftFor(Faiza)) or !Receives(Karim, GiftFor(Faiza))
21 Resolving T4 and KB 3:
22 T4 = !Guest(Faiza) or !Receives(Karim, GiftFor(Faiza))
23 KB3 = Guest(Karim)
24 T8 =
25 Yes
```

Explanation of the process:

The KB in Natural Language of the first example:

1. Every guest receives at least one gift.
2. Faiza is a guest.

The intermediate activities :

KB in in FOL:

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

Conversion to CNF:
$$(\neg \text{Guest}(x) \vee \text{Gift}(\text{GiftFor}(x))) \wedge (\neg \text{Guest}(x) \vee \text{Receives}(x, \text{GiftFor}(x)))$$
KB in CNF of FOL:

1. $\neg \text{Guest}(x) \vee \text{Gift}(\text{GiftFor}(x))$
2. $\neg \text{Guest}(x) \vee \text{Receives}(x, \text{GiftFor}(x))$
3. $\text{Guest}(\text{Faiza})$

Conversion of the negation of the query:

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

$$\Rightarrow \neg \text{Gift}(x) \vee \neg \text{Receives}(\text{Faiza}, x)$$
The KB after negation of the query:

1. $\neg \text{Guest}(x) \vee \text{Gift}(\text{GiftFor}(x))$
2. $\neg \text{Guest}(x) \vee \text{Receives}(x, \text{GiftFor}(x))$
3. $\text{Guest}(\text{Faiza})$

Finding answer to query:

- T1. $\neg \text{Gift}(y) \vee \neg \text{Receives}(\text{Faiza}, y)$
- T2. $\vee \neg \text{Receives}(\text{Karim}, \text{GiftFor}(x)) \vee \neg \text{Guest}(x)$
- T3. $\neg \text{Gift}(\text{GiftFor}(\text{Faiza})) \vee \neg \text{Guest}(\text{Faiza})$
- T4. $\neg \text{Guest}(\text{Faiza})$
- T5. $\neg \text{Receives}(\text{Faiza}, \text{GiftFor}(\text{Faiza}))$
- T6. $\neg \text{Guest}(\text{Faiza})$
- T7. $\neg \text{Gift}(\text{GiftFor}(\text{Faiza}))$
- T8. [] (stop)

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