BVSK Kaushik
CSE 1 - 6th Semester (2021-2025)
02015002721

Artificial Intelligence

S.No.	Name of Experiment	Page No.	Date	Sign

Experiment - 1

Aim - Study of Prolog

Prolog or PROgramming in LOGics is a logical and declarative programming language. It is one major example of the fourth generation language that supports the declarative programming paradigm. This is particularly suitable for programs that involve symbolic or non-numeric computation. This is the main reason to use Prolog as the programming language in Artificial Intelligence, where symbol manipulation and inference manipulation are the fundamental tasks.

Knowledge Base in Logic Programming

Well, as we know there are three main components in logic programming − Facts, Rules and Queries. Among these three if we collect the facts and rules as a whole then that forms a Knowledge Base. So we can say that the knowledge base is a collection of facts and rules.

Facts

We can define fact as an explicit relationship between objects, and properties these objects might have. So facts are unconditionally true in nature.

Syntax

The syntax for facts is as follows −

relation(object1,object2...).

Rules

We can define rule as an implicit relationship between objects. So facts are conditionally true. So when one associated condition is true, then the predicate is also true.

Syntax

rule_name(object1, object2, ...) :- fact/rule(object1,
object2, ...)

Suppose a clause is like :

P :- Q;R.

This can also be written as

P :- Q.

P :- R.

If one clause is like :

P :- Q,R;S,T,U.

Is understood as
P :- (Q,R);(S,T,U).
Or can also be written as:
P :- Q,R.
P :- S,T,U.

Queries

Queries are some questions on the relationships between objects and object properties.
Syntax
| ? - relation(object1, object2, …)

The above typed in the console is a query and returns whether it is true or false.

Atoms

Atoms are one variation of constants. They can be any names or objects. There are few rules that should be followed when we are trying to use Atoms as given below −
Strings of letters, digits and the underscore character, '_', starting with a lower-case letter. For example −

azahar
b59
b_59
b_59AB
b_x25
antara_sarkar

Strings of special characters

We have to keep in mind that when using atoms of this form, some care is necessary as some strings of special characters already have a predefined meaning; for example ':-'.

<--->
=======>
...
.:.
::=

Strings of characters enclosed in single quotes.

This is useful if we want to have an atom that starts with a capital letter. By enclosing it in quotes, we make it distinguishable from variables −

'Rubai'
'Arindam_Chatterjee'
'Sumit Mitra'

Numbers

Another variation of constants is the Numbers. So integer numbers can be represented as 100, 4, -81, 1202. In Prolog, the normal range of integers is from -16383 to 16383.

Prolog also supports real numbers, but normally the use-case of floating point number is very less in Prolog programs, because Prolog is for symbolic, non-numeric computation. The treatment of real numbers depends on the implementation of Prolog. Example of real numbers are 3.14159, -0.00062, 450.18, etc.

The variables come under the Simple Objects section. Variables can be used in many such cases in our Prolog program, that we have seen earlier. So there are some rules of defining variables in Prolog.

We can define Prolog variables, such that variables are strings of letters, digits and underscore characters. They start with an upper-case letter or an underscore character. Some examples of Variables are −

X
Sum
Memer_name
Student_list
Shoppinglist
_a50
_15

Anonymous Variables in Prolog

Anonymous variables have no names. The anonymous variables in prolog is written by a single underscore character '_'. And one important thing is that each individual anonymous variable is treated as different. They are not same.

Comparison Operators

Comparison operators are used to compare two equations or states. Following are different comparison operators −

Operator	Meaning
X > Y	X is greater than Y
X < Y	X is less than Y
X >= Y	X is greater than or equal to Y

X =< Y	X is less than or equal to Y
X =:= Y	the X and Y values are equal
X =\= Y	the X and Y values are not equal

You can see that the '=<' operator, '=:=' operator and '=\=' operators are syntactically different from other languages. Let us see some practical demonstration to this.

Arithmetic Operators in Prolog

Arithmetic operators are used to perform arithmetic operations. There are few different types of arithmetic operators as follows −

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
//	Integer Division
mod	Modulus

Decision Making

The decision statements are If-Then-Else statements. So when we try to match some condition, and perform some task, then we use the decision making statements. The basic usage is as follows −

If <condition> is true, Then <do this>, Else

In some different programming languages, there are If-Else statements, but in Prolog we have to define our statements in some other manner. Following is an example of decision making in Prolog.
Program

```
% If-Then-Else statement

gt(X,Y) :- X >= Y,write('X is greater or equal').
gt(X,Y) :- X < Y,write('X is smaller').

% If-Elif-Else statement

gte(X,Y) :- X > Y,write('X is greater').
gte(X,Y) :- X =:= Y,write('X and Y are same').
gte(X,Y) :- X < Y,write('X is smaller').
```

Loops
Loop statements are used to execute the code block multiple times. In general, for, while, do-while are loop constructs in programming languages (like Java, C, C++).
Code block is executed multiple times using recursive predicate logic. There are no direct loops in some other languages, but we can simulate loops with few different techniques.
Program

```
count_to_10(10) :- write(10),nl.
count_to_10(X) :-
  write(X),nl,
  Y is X + 1,
  count_to_10(Y).
```

Conjunction
Conjunction (AND logic) can be implemented using the comma (,) operator. So two predicates separated by comma are joined with AND statement. Suppose we have a predicate, parent(jhon, bob), which means “Jhon is parent of Bob”, and another predicate, male(jhon), which means “Jhon is male”. So we can make another predicate that father(jhon,bob), which means “Jhon is father of Bob”. We can define predicate father, when he is parent AND he is male.

Disjunction
Disjunction (OR logic) can be implemented using the semi-colon (;) operator. So two predicates separated by semi-colon are joined with OR statement. Suppose we have a predicate, father(jhon, bob). This tells that “Jhon is father of Bob”, and another predicate, mother(lili,bob), this tells that “lili is mother of bob”. If we create another predicate as child(), this will be true when father(jhon, bob) is true OR mother(lili,bob) is true.
Program

```
parent(jhon,bob).
parent(lili,bob).
```

```
male(jhon).
female(lili).

% Conjunction Logic
father(X,Y) :- parent(X,Y),male(X).
mother(X,Y) :- parent(X,Y),female(X).

% Disjunction Logic
child_of(X,Y) :- father(X,Y);mother(X,Y).
```

Experiment - 2

Aim - Write simple facts for statements using PROLOG.

a) Ram likes mango
b) Seema is a girl
c) Bill likes Cindy
d) Rose is red
e) John owns gold

Program:

```
likes(ram, mango).
likes(bill, cindy).
isgirl(seema).
isred(rose).
owns(john, gold).
```

Output:

```
| ?- consult('C:/Users/students/Documents/exp2.pl').
compiling C:/Users/students/Documents/exp2.pl for byte code...
C:/Users/students/Documents/exp2.pl compiled, 4 lines read - 742 bytes written, 6 ms

yes
| ?- likes(bill, cindy).

yes
| ?- likes(ram, mango).

yes
| ?- owns(john, gold).

yes
| ?- isgirl(seema).

yes
| ?- isred(rose).

yes
```

Experiment - 3

Aim - Write predicates, one converts centigrade temperatures to Fahrenheit, the other checks if temperature is below freezing using PROLOG.

Program:

```
1  ctof(C) :-
2  write('Fahreinheit temperature is '),
3  F is (9 / 5 * C) + 32,
4  write(F).
5
6  isbelowFreezing(C) :- C < 0.
```

Output:

```
| ?- consult('C:/Users/students/Documents/exp3.pl').
compiling C:/Users/students/Documents/exp3.pl for byte code...
C:/Users/students/Documents/exp3.pl compiled, 5 lines read - 893 bytes written, 6 ms

yes
| ?- isbelowFreezing(-10).

yes
| ?- isbelowFreezing(0).

no
| ?- ctof(20)
.
Fahreinheit temperature is 68.0

yes
| ?-
```

Experiment - 4

Aim - Write a program to implement Breadth First Search Traversal.

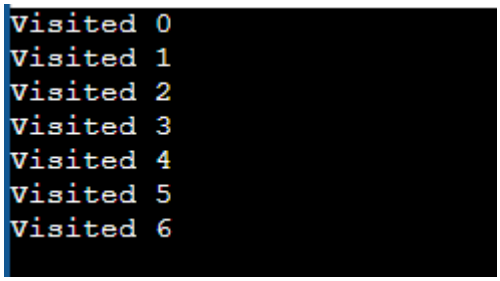
Program:

```
#include <iostream>
#include <list>
using namespace std;
class Graph{
   int nVertices;
   list<int> *adjLists;
   public:
   Graph(int n) {
      nVertices = n;
      adjLists = new list<int>[n];
   }
```

```
    void addEdge(int src, int dest) {
        adjLists[src].push_back(dest);
        adjLists[dest].push_back(src);
    }
    void BFS(int start) {
        bool *visited = new bool[nVertices];
        for (int i = 0; i < nVertices; i++)
            visited[i] = false;

        list<int> queue;

        visited[start] = true;
        queue.push_back(start);

        list<int>::iterator i;

        while (!queue.empty()) {
            int currVertex = queue.front();
            cout << "Visited " << currVertex << "\n";
            queue.pop_front();

            for (i = adjLists[currVertex].begin(); i != adjLists[currVertex].end(); ++i) {
                int adjVertex = *i;
                if (!visited[adjVertex]) {
                    visited[adjVertex] = true;
                    queue.push_back(adjVertex);
                }
            }
        }
    }
};

int main() {
    Graph g(7);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 3);
    g.addEdge(1, 4);
    g.addEdge(2, 5);
    g.addEdge(2, 6);

    g.BFS(0);

    return 0;
```

```
}
```

Output:



Experiment - 5

Aim - Write a program to implement Water Jug problem.

Program:

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int, int> pii;
void printpath(map<pii, pii> mp, pii u)
{
    if (u.first == 0 && u.second == 0) {
        cout << 0 << " " << 0 << endl;
        return;
    }
    printpath(mp, mp[u]);
    cout << u.first << " " << u.second << endl;
}
void BFS(int a, int b, int target)
```

```
{
    map<pii, int> m;
    bool isSolvable = false;
    map<pii, pii> mp;

    queue<pii> q;

    q.push(make_pair(0, 0));
    while (!q.empty()) {

        auto u = q.front();
        // cout<<u.first<<" "<<u.second<<endl;
        q.pop();
        if (m[u] == 1)
            continue;

        if ((u.first > a || u.second > b || u.first < 0
             || u.second < 0))
            continue;
        // cout<<u.first<<" "<<u.second<<endl;

        m[{ u.first, u.second }] = 1;

        if (u.first == target || u.second == target) {
            isSolvable = true;

            printpath(mp, u);
            if (u.first == target) {
                if (u.second != 0)
                    cout << u.first << " " << 0 << endl;
            }
            else {
                if (u.first != 0)
                    cout << 0 << " " << u.second << endl;
            }
            return;
        }
        // completely fill the jug 2
        if (m[{ u.first, b }] != 1) {
            q.push({ u.first, b });
            mp[{ u.first, b }] = u;
        }
         // completely fill the jug 1
        if (m[{ a, u.second }] != 1) {
```

```
    q.push({ a, u.second });
    mp[{ a, u.second }] = u;
}
 // transfer jug 1 -> jug 2
int d = b - u.second;
if (u.first >= d) {
    int c = u.first - d;
    if (m[{ c, b }] != 1) {
        q.push({ c, b });
        mp[{ c, b }] = u;
    }
}
else {
    int c = u.first + u.second;
    if (m[{ 0, c }] != 1) {
        q.push({ 0, c });
        mp[{ 0, c }] = u;
    }
}
// transfer jug 2 -> jug 1
d = a - u.first;
if (u.second >= d) {
    int c = u.second - d;
    if (m[{ a, c }] != 1) {
        q.push({ a, c });
        mp[{ a, c }] = u;
    }
}
else {
    int c = u.first + u.second;
    if (m[{ c, 0 }] != 1) {
        q.push({ c, 0 });
        mp[{ c, 0 }] = u;
    }
}
 // empty the jug 2
if (m[{ u.first, 0 }] != 1) {
    q.push({ u.first, 0 });
    mp[{ u.first, 0 }] = u;
}
 // empty the jug 1
if (m[{ 0, u.second }] != 1) {
    q.push({ 0, u.second });
    mp[{ 0, u.second }] = u;
```

```
        }
    }
    if (!isSolvable)
        cout << "No solution";
}
int main()
{
    int Jug1 = 4, Jug2 = 3, target = 2;
    cout << "Path from initial state "
         "to solution state ::\n";
    BFS(Jug1, Jug2, target);
    return 0;
}
```

Output:

```
Path from initial state to solution state ::
0 0
0 3
3 0
3 3
4 2
0 2
```

Experiment - 6

Aim - Write a program to remove punctuations from given string.

Program:

```
#include <iostream>
#include <string>
#include <ctype.h>

using namespace std;

int main() {
    string s;
    getline(cin, s);
    char ns[s.length() + 1];
    int i = 0;
    int k = 0;
    while (i < s.length()) {
        if (isalnum(s[i]) || s[i] == ' ') {
```

```
            ns[k] = s[i];
            ++k;
        }
        ++i;
    }
    ns[k] = '\0';
    cout << ns;
    return 0;
}
```

Output:

```
Raining, cats-and dogs.?
Raining catsand dogs
```

Experiment - 7

Aim - Write a program to sort a sentence in alphabetical order.

Program:

```
sentence = input("Enter sentence: ")
sort = sorted(sentence)
sortsent = "".join(sort)
print("Sentence sorted by char: ", sortsent)

def cmpfnc(a):
    return a.lower()

words = sentence.split()
sort = sorted(words, key=cmpfnc)
sortsent = " ".join(sort)
print("Sentence sorted by words: ", sortsent)
```

Output:

```
Enter sentence: A cat Runs fast
Sentence sorted by char:     ARaacfnssttu
Sentence sorted by words:  A cat fast Runs
```

Experiment - 8

Aim - Write a program to implement a Hangman game using python.

Program:

```
import random

words = ["lamberouge", "ikigai", "ayurveda", "shadow", "divine"]
word = random.choice(words)
turns = 10
guesses = []
print(f"Guess the characters in the word of length {len(word)}")
while (turns > 0):
    suc = True
    guess = input("Guess the character: ")
    guesses.append(guess)
    for char in word:
        if char in guesses:
```

```
            print(char, end=' ')
        else:
            suc = False
            print('_', end=' ')
    print()
    if (suc):
        print(f"You guessed {word} with {turns} turns remaining. You win!")
        break
    if guess not in word:
        turns -= 1
        print("Wrong guess")
else:
    print("_________\n|      |     \n|      0     \n|     /|\\     \n|     / \\\n|")
    print("You lose!")
```

Output:

```
Guess the characters in the word of length 6
Guess the character: i
i _ i _ _ i
Guess the character: k
i k i _ _ i
Guess the character: g
i k i g _ i
Guess the character: a
i k i g a i
You guessed ikigai with 10 turns remaining. You win!
```

```
Guess the characters in the word of length 10
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
Guess the character: d

_ _ _ _ _ _ _ _ _ _
Wrong guess
________
|      |
|      0
|     /|\
|     / \
|
You lose!
```

Experient - 9

Aim - Write a program to implement the Tic-Tac-Toe game.

Program:

```
board = [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
player = 1

# Win Flags
Win = 1
Draw = -1
Running = 0
Stop = 1

Game = Running
Mark = 'X'


def DrawBoard():
    print(" %c | %c | %c " % (board[1], board[2], board[3]))
    print("___|___|___")
    print(" %c | %c | %c " % (board[4], board[5], board[6]))
    print("___|___|___")
    print(" %c | %c | %c " % (board[7], board[8], board[9]))
    print()


def CheckPosition(x):
    if board[x] == ' ':
        return True
    else:
        return False


def CheckWin():
    global Game

    if board[1] == board[2] and board[2] == board[3] and board[1] != ' ':
        Game = Win
    elif board[4] == board[5] and board[5] == board[6] and board[4] != ' ':
        Game = Win
    elif board[7] == board[8] and board[8] == board[9] and board[7] != ' ':
        Game = Win
```

```
    elif board[1] == board[4] and board[4] == board[7] and board[1] != ' ':
        Game = Win
    elif board[2] == board[5] and board[5] == board[8] and board[2] != ' ':
        Game = Win
    elif board[3] == board[6] and board[6] == board[9] and board[3] != ' ':
        Game = Win
    elif board[1] == board[5] and board[5] == board[9] and board[5] != ' ':
        Game = Win
    elif board[3] == board[5] and board[5] == board[7] and board[5] != ' ':
        Game = Win
    elif board[1] != ' ' and board[2] != ' ' and board[3] != ' ' and \
            board[4] != ' ' and board[5] != ' ' and board[6] != ' ' and \
            board[7] != ' ' and board[8] != ' ' and board[9] != ' ':
        Game = Draw
    else:
        Game = Running


print("Player 1 [X] --- Player 2 [O]\n")
DrawBoard()
while Game == Running:
    if player % 2 != 0:
        print("Player 1's chance")
        Mark = 'X'
    else:
        print("Player 2's chance")
        Mark = 'O'

    choice = int(input("Enter the position between [1-9] where you want to mark: "))
    if CheckPosition(choice):
        board[choice] = Mark
        player += 1
        CheckWin()

    DrawBoard()

    if Game == Draw:
        print("Game Draw")
    elif Game == Win:
        player -= 1
        if player % 2 != 0:
            print("Player 1 Won")
        else:
            print("Player 2 Won")
```

Output:

```
Player 1 [X] --- Player 2 [O]

   |   |
___|___|___
   |   |
___|___|___
   |   |

Player 1's chance
Enter the position between [1-9] where you want to mark: 1
 X |   |
___|___|___
   |   |
___|___|___
   |   |

Player 2's chance
Enter the position between [1-9] where you want to mark: 3
 X |   | O
___|___|___
   |   |
___|___|___
   |   |

Player 1's chance
Enter the position between [1-9] where you want to mark: 5
 X |   | O
___|___|___
   | X |
___|___|___
   |   |

Player 2's chance
Enter the position between [1-9] where you want to mark: 6
 X |   | O
___|___|___
   | X | O
___|___|___
   |   |

Player 1's chance
Enter the position between [1-9] where you want to mark: 9
 X |   | O
___|___|___
   | X | O
___|___|___
   |   | X

Player 1 Won
```

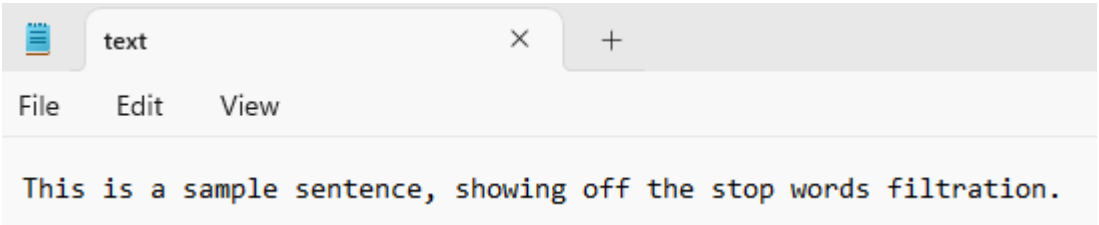
Experiment - 10

Aim - Write a program to remove stopwords for a given passage from a text file using NLTK.

Program:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

sent = ""
with open('text.txt') as file:
    sent = file.read()

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(sent)
filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]

print("Sample sentence: ", sent)
print()
print("Filtered sentence: ", ' '.join(filtered_sentence))
```

Output:




```
Sample sentence:  This is a sample sentence, showing off the stop words filtration.

Filtered sentence:  sample sentence , showing stop words filtration .
```

Experiment - 11

Aim - Write a program to implement stemming for a given sentence using NLTK.

Program:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from functools import reduce

ps = PorterStemmer()

sentence = "Programmers program with programming languages"
words = word_tokenize(sentence)

stemmed_sentence = reduce(lambda x, y: x + " " + ps.stem(y), words, "")

print("Sentence: ", sentence)
print()
print("Stemmed sentence: ", stemmed_sentence)
```

Output:

```
Sentence:  Programmers program with programming languages

Stemmed sentence:   programm program with program languag
```

Experiment - 12

Aim - Write a program to implement POS(part of speech) tagging for the given sentence using NLTK.

Program:

```
import nltk
from nltk.tokenize import word_tokenize

sentence = "The girl wearing the yellow dress is my new neighbour."
wordsList = nltk.word_tokenize(sentence)
tagged = nltk.pos_tag(wordsList)

print("Sentence: ", sentence)
print()
print("POS tagged list: ", tagged)
```

Output:

```
Sentence:  The girl wearing the yellow dress is my new neighbour.

POS tagged list:  [('The', 'DT'), ('girl', 'NN'), ('wearing', 'VBG'), ('the', 'DT'),
('yellow', 'JJ'), ('dress', 'NN'), ('is', 'VBZ'), ('my', 'PRP$'), ('new', 'JJ'), ('ne
ighbour', 'NN'), ('.', '.')]
```

Experiment - 13

Aim - Write a program to implement Lemmatization using NLTK.

Program:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos="a"))
```

Output:

```
rocks : rock
corpora : corpus
better : good
```

Experiment - 14

Aim - Write a program for Text Classification for the given sentence using NLTK.

Program: