Artificial Intelligence
[Lab Manual]

Academic Year 2010-11
Spring Semester

Indira Gandhi Institute of Technology
Guru Gobind Singh Indraprastha University
Kashmere Gate, Delhi

Version No.	Comments	Modification Date	Modification Done By
1.0	Created Lab Manual – added list of experiments related to Prolog.	8th Feb. 2011	Rishabh Kaushal, Assistant Professor, Information Technology Dept., IGIT.
2.0	Suggested experiments related to topics like Fuzzy Logic, Neural Networks and Genetic Programming	21st Feb. 2011	Dr. Devendra Tayal, Associate Professor & Head, Computer Science Dept., IGIT.

Table: Modification History

Table of Contents

0. Guidelines

This document titled ‘Artificial Intelligence - Lab Manual’ is written based on following guidelines.

1. Lab work is divided into 12 weeks. Each week has certain objectives to be accomplished.
2. Each week has 2 hours of Lab work.
3. Time limit for each experiment is one week unless otherwise stated in the experiment.
4. The stipulated 2 hours of Lab work time is only a *necessary* condition but may not be *sufficient* to solve the experiment exercises. Therefore, students are strongly encouraged and advised to install the necessary software(s) in their personal computing environments.
5. Each experiment would be supplemented by relevant reference(s), exercise questions could only be solved by a thorough reading of these reference(s) and not otherwise. Soft-copy format of the books mentioned in reference(s) shall be provided.
6. Students are expected to perform one experiment every week even in the case when lab session could not be held in that week for any reason whatsoever. If deemed necessary, students can approach the Lab Instructor to fix an alternate time-slot for the orientation on the scheduled experiment.

1. Syllabus

As taken from www.ipu.ac.in website*.

**Academics → Academics → Scheme Syllabus (Affiliated Institutes, w.e.f. 2005-06)*

UNIT – I

Scope of AI: Games, theorem proving, natural language processing, vision and speech processing, robotics, expert systems, AI techniques-search knowledge, abstraction.

Problem Solving (Blind): State space search; production systems, search space control; depth-first, breadth-first search.

Heuristic Based Search: Heuristic search, Hill climbing, best-first search, branch and bound, Problem Reduction, Constraint Satisfaction End, Means-End Analysis.

[No. of Hrs.: 12]

UNIT – II

Game Playing: Game Tree, Minimax Algorithm, Alpha Beta Cutoff, Modified Minimax Algorithm, Horizon Effect, Futility Cut-off.

Knowledge Representation: Predicate Logic: Unificatioin, Modus Ponens, Modus Tolens, Resolution in Predicate Logic, Conflict Resolution Forward Chaining, Backward Chaining, Declarative and Procedural Representation, Rule based Systems.

Structured Knowledge Representation: Semantic Nets: Slots, exceptions and default frames, conceptual dependency, scripts.

[No. of Hrs.: 12]

UNIT – III

Handling Uncertainty: Non-Monotonic Reasoning, Probabilistic reasoning, use of certainty factors, fuzzy logic.

Natural Language Processing: Introduction, Syntactic Processing, Semantic Processing, Pragmatic Processing.

[No. of Hrs.: 10]

UNIT – IV

Learning: Concept of learning, learning automation, genetic algorithm, learning by inductions, neural nets.

Expert Systems: Need and justification for expert systems, knowledge acquisition, Case Studies: MYCIN, RI. **[No. of Hrs.: 10]**

TEXT BOOKS:

1. E. Rich and K. Knight, "Artificial Intelligence", TMH, 2nd Ed., 1992.
2. N. J. Nilsson, "Principles of AI", Narosa Publ. House, 1990.
3. M. N. Hoda, "Foundation Course in Artificial Intelligence", Vikas Pub., 2004.

REFERENCES BOOKS:

1. P. H. Winston, "Artificial Intelligence", Pearson Education, 3rd Edition, 2002.

2. D. W. Patterson, “Introduction to AI and Expert Systems”, PHI, 1992.
3. R. J. Schalkoff, “Artificial Intelligence – An Engineering Approach”, McGraw Hill Int. Ed. Singapore, 1992.
4. M. Sasikumar, S. Ramani, “Rule Based Expert Systems”, Narosa Publishing House, 1994.
5. Tim Johns, “Artificial Intelligence, Application Programming”, Wiley Dreamtech, 2005.

2. List of Experiments

This section gives an outline of the experiments to be performed each week. For conceptual details of each experiment, please refer the respective chapters from the books mentioned in the references.

A. Prolog related Experiments.

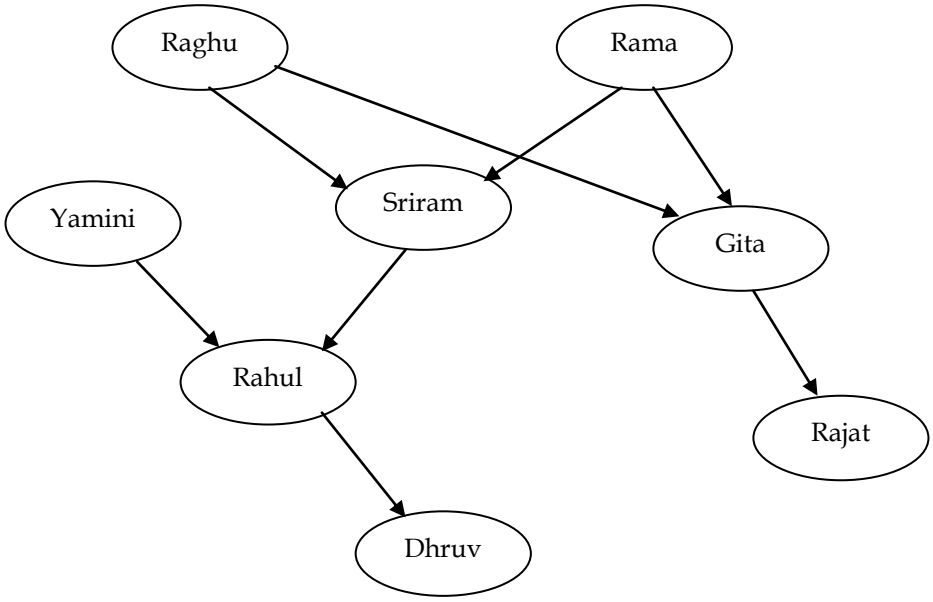
2.1 Prolog: Tutorial Introduction.

Objectives:

1. Familiarity with the *Prolog* programming environment.
2. Tutorial introduction to *Prolog* programming

Exercises:

Given below is a family relation. Answer the following questions using 'Prolog' programming:-



1. Assuming that an arrow (X ⟶ Y) in Figure 1.1 depicts a *parent* relation from X to Y, in other words, X is parent of Y. Answer the following:-
   a. Enlist the various facts in terms of *clauses.*
   b. Is Sriram parent of Rahul?
   c. Is Lakshman parent of Gita?
   d. Is Rahul parent of Yamini?
   e. Find all children of Raghu.
   f. Enlist all parent-child relations.
   g. Who is grand-parent of Rahul?

h. Who are grand-children of Rama?
i. Are Sriram and Gita siblings?
j. What are the results of the following *prolog statements*?
   i.?- parent(X,Sriram).
   ii.?- parent(Gita,X).
   iii.?- parent(X,Y), parent(Yamini,X), parent(Y,Dhruv).

2. Create the following new "relations" (write new *prolog rules* for the following) and draw their *definition graphs:-*
   a. Male
   b. Female
   c. Mother
   d. Father
   e. Grandfather
   f. Grandchild
   g. Child
   h. Sibling
   i. Brother
   j. Sister
   k. Aunt

3. What is the use of *different* relation, illustrate with an example.

4. Add *predecessor* relation to the *prolog program* constructed above.

5. Illustrate the working of the **goal** *predecessor (Rama, Dhruv)* using execution trace.

Notes:

1. Prolog programs consist of *clauses*, each terminated with a full-stop (.).

2. A relation can be defined in form of n-tuple.

3. Arguments to the relation can be of two types – *atoms* and *variables*.

4. *Clauses* are of three types – *facts*, *rules* and *questions*. *Facts* declare statements that are unconditionally true, *rules* are statements that are true based on certain conditions and through *questions*, user can ask what is true and what is false.

5. Questions are given as *goals* to be satisfied whose answers are "yes" or "no".

6. Prolog *clauses* consist of *head* and *body*, *body* consist of list of *goals* separated by comma (,) where a comma is a conjunction.

7. *Facts* are prolog *clauses* with empty *body*.

8. Predecessor relation (formed using recursive rule) is composed of two rules. First rule determines *direct* relation while second rule determines *indirect* relation.

9. Prolog programs explains *declarative* and *procedural* meanings, *declarative* decides the output of the program while *procedural* decides how the output is obtained.

10. Emphasis upon the *declarative style* of programming.

References:

1. Prolog Programming for Artificial Intelligence by Ivan Bratko (Chapter-1)

2. An Introduction to Prolog Programming (Lecture Notes) by Ulle Endriss (Chapter-1)

2.2 Prolog: A Systematic Study.

Objectives:

1. Systematic introduction to *Prolog* programming constructs

2. Learning basic concepts of *Prolog* through illustrative examples and small exercises

Exercises:

Answer the following questions.

1. What are the various data objects used in *Prolog* programming? Explain them using few examples of each category.

2. All structures objects in *Prolog* are trees. Draw tree diagrams for the following expressions:-
   /(*(x,y),+(z,1))

3. Assume that a rectangle is represented by the term *rectangle(P1,P2,P3,P4)* where P's are vertices of the rectangle. Define the relation *regular(R)* which is true if R is a rectangle whose sides are vertical and horizontal. What will be the changes if rectangle is replaced with a square.

4. How are conjunction and disjunction used in *Prolog* programming? Explain with an example for each using relations from the family tree.

5. Trace the steps involved in concluding "Seema is talkative" from the facts "All girls are talkative" and "Seem is a girl".

6. Will the following query succeed? (Give reasons to support your answer)
   ?- hates(rahul, sheetal) = hates(Rahul, Sheetal)

7. What do you mean by single underscore (_)?

8. Why is order of clauses important? Illustrate with possibility of formation of infinite loop.

Notes:

1. There are *four* ways of organizing data in *Prolog* namely
   a. Atoms: same as *constants (never starts with capital letter)*
   b. Numbers: stores numerical data
   c. Variables: starts with *capital letter or underscore (_)*
   d. Compound Term: consist of a *functor* and arguments

2. All *terms* can be represented as tree data structure.

3. Two *terms* are said to match if they are identical or if they can be made identical by means of variable instantiation.

References:

1. Prolog Programming for Artificial Intelligence by Ivan Bratko (Chapter-2)

2. An Introduction to Prolog Programming (Lecture Notes) by Ulle Endriss (Chapter-1)

2.3 Operations on List data structure in Prolog.

Objectives:

1. Understanding *list* data structure in *Prolog*.

2. Writing illustrative examples and solving exercises related to various operations that can be performed on *lists* in *Prolog*.

Exercises:

Answer the following questions.

1. Write your own rules (don't use any pre-defined relations in *Prolog* language) for each of the following operations:-
   a. Adding element X in the list L at start and at end
   b. Deleting element X from the list L
   c. Checking if element X is present in list L
   d. Concatenating two lists L1 and L2 into a list L

2. Enlist with examples at least three pre-defined operations in *Prolog* for *list* which are not covered in above Question.

3. List can be considered as a set, assuming that there is no duplicate entry in it. Write rules for the relation *powerSet(P,L)* such that P is a list consisting of power set of given list L. Hint: Power set of S is a set of all possible subsets of S.

4. Define the following relations:-
   a. reverse(L,R) where R is the reverse of the given list L.
   b. palindrome(L) which outputs "Yes" if the given list L is palindrome.
   c. last(L,X) where X is the last element of the given list L.
   d. first(L,X) where X is the first element of the given list L.

Notes:

1. List data structures are internally organized as trees.

2. List consist of two parts namely *head* and *tail*, where *head* can be any data object while *tail* must always be a list (it is either empty or in turn can be represented as *head* and *tail*).

3. Some of the common operations that can be performed over *lists* are checking an object for membership, concatenating two lists, adding new objects to lists or deleting the existing ones, finding length of list and so on.

References:

1. Prolog Programming for Artificial Intelligence by Ivan Bratko (Chapter-3)

2. An Introduction to Prolog Programming (Lecture Notes) by Ulle Endriss (Chapter-2)

2.4 Understanding operators in Prolog.

Objectives:

1. Understanding the concept of *arithmetic expressions* in *Prolog.*

2. Writing illustrative examples and solving exercises related to various *user-defined* operators in *Prolog*.

Exercises:

Answer the following questions.

1. 8 *is* 3+5 works but other way around doesn't, comment.

2. Write a predicate *factorial(N,X)* which finds the factorial (X) of the given number (N).

3. Assuming the following operator definitions:-
   :- op(300, xfx, likes).
   :- op(200, xfy, and).
   :- op(100, yfx, or).
   How are the terms given below interpreted by Prolog, what are the principal functors and what are their corresponding tree structures?
   Term1 = Ram *likes* football.
   Term2 = Raghu *likes* tennis *and* football *or* volleyball.
   Term3 = Football *likes* Ram *and* Raghu.

5. What are the candidates for operators in sentences of the following type? For each of the candidate operator selected, define the operator definitions and ask questions related to them as given below.
   Sentences:-
   Ram is good.
   Good is selected.
   Raghu is not good.
   Questions:-
   Who is good.
   Who is not selected.
   Who is not good.
   Who is not selected.

6. Write rules for following predicates (related to operations on lists):-
   a. *length(L,N)* which finds length (N) of the given list (L).
   b. *average(L,X)* which finds average (X) of the given list (L).
   c. *maxList(L,M)* which finds maximum (M) of the given list (L).

Notes:

1. The *is* operator is used to operate upon arithmetic operations.
   a. The *is* operator takes two arguments, of which second has to be a valid arithmetic expression with all variables instantiated, while first can be a variable representing a number or a number itself.
   b. Operation succeeds if result of second argument *matches* the first one.

2. Arithmetic operators are categorized into *functions* and *relations*, few examples are:-
   a. Functions: *max/2, min/2, abs/1*
   b. Relations: <, >, =:=

3. New operators are defined by the *op* directive, stating name of the variable its type and precedence.

References:

1. Prolog Programming for Artificial Intelligence by Ivan Bratko (Chapter-3)

2. An Introduction to Prolog Programming (Lecture Notes) by Ulle Endriss (Chapter-3 and Chapter-4)

2.5 Programming Challenge: Developing my own Prolog interpreter.

Objectives:

1. Understanding the *under the hood* workings of *Prolog* interpreter.
2. Developing a procedural code which *imitates* the declarative style of *Prolog* programming by focusing on *matching* algorithm followed by *Prolog* interpreter.

Exercise: (Problem Statement)

You are employed in a company *Engineering Illustrated Pvt. Ltd.* and are in the team which develops software for engineering colleges in order to facilitate and enhance learning by demystifying the internal workings of some of the tools being used in Labs. One such tool is *SWI-Prolog* which is used in a college named *Indira Gandhi Institute of Technology*. The college has approached your company to help them in understanding the tool (SWI-Prolog). After *using* the tool (SWI-Prolog) for few weeks, your team has realized that at the heart of SWI-Prolog's working is its amazing algorithm for *matching* using which it answers questions posed by user in presence of an initial knowledge fed to it (in form of a file, say "file.pl"). While *using* SWI-Prolog, you have also realized that unlike procedural languages like C, SWI-Prolog follows an entirely different approach i.e. declarative style of programming and as a result of which, it becomes quite a steep learning curve for the students who are beginners. Therefore, to reduce the learning time, you have decided to develop your own *Prolog Interpreter* which shall convert the *declarative* style of SWI-Prolog programs (i.e. knowledge input in form of facts and rules, say in "file.pl") into *procedural* style of programs (which are more closer to the approach in which most students are quite proficient). And in doing so, you believe that it would help students understand the SWI-Prolog programs better and quicker.

In order to track the progress of development of your *Prolog Interpreter*, following three milestones are devised:-

1. Input Knowledge ("file.pl") *only* consist of facts.

Eg.

```
parent(bob,john).                    # fact
male(bob).                           # fact
capital(delhi,india).
good(john).
```

2. Input Knowledge ("file.pl") consist of facts and rules.

Eg.

```
parent(bob,john).
male(bob).
father(X,Y):-parent(X,Y),male(X).    # rule
capital(delhi,india).
```

```
city(X):-capital(X,_).
good(john).
better(X):-parent(X,_),good(X).
```

3. Input Knowledge ("file.pl") consist of *recursive* rules.

Eg.

```
parent(bob,john).
male(bob).
father(X,Y):-parent(X,Y),male(X).
capital(delhi,india).
city(X):-capital(X,_).
good(john).
better(X):-parent(X,_),good(X).
predecessor(X,Y):-parent(X,Y).
predecessor(X,Y):-parent(X,Y),predecessor(Y,Z).     # recursive rule
```

Notes:

1. Parsing of the input knowledge in form of "file.pl" needs to be done.
2. Relevant information to be extracted saved in *appropriate* data structures.
3. Mechanism for matching the questions posed in a file, say "questions.txt" with the saved information in data structures is to be devised.

References:

1. Prolog Programming for Artificial Intelligence by Ivan Bratko (Chapter-2)
2. An Introduction to Prolog Programming (Lecture Notes) by Ulle Endriss (Chapter-1 and Chapter-2)

B. Fuzzy Logic related Experiments.

2.6 Kohonen Self Organizing Feature Map.

Exercise: (Problem Description)

Write a program to create Kohonen Self Organizing Feature Map. A provision should be made for the user to enter inputs and initialize the weights of his choice.

2.7 Fuzzy Sets.

Exercise: (Problem Description)

Write a program to implement the various primitive operations of fuzzy sets. The program should provide a feature to enter number of elements in the Universe of Discourse and the two fuzzy sets may be defined on it.

2.8 Tournament Selection Method – Genetic Algorithm.

Exercise: (Problem Description)

Using Tournament Selection Method of the Genetic Algorithms,

Maximize $F(x_1,x_2) = 4x_1 + 3x_2$, subject to the following constraints

$2*x_1 + 3*x_2 <= 6$

$-3*x_1 + 2*x_2 <= 3$

$2*x_1 + x_2 <= 4$

$0 <= x <= 2$

Your program should accomplish the following steps:-

a. Generate initial population using random number generation.

b. Use the Tournament Selection Method to select any two parents

c. Generate the offspring using the following crossover operation:-

$x_1 = a*x_1 + (1-a)*x_2$

$x_2 = a*x_2 + (1-a)*x_1$

d. Calculate the maximum fitness value by applying this operation separately for each iteration. Show your outputs.

2.9 Travelling Salesman Problem – Genetic Algorithm.

Exercise: (Problem Description)

Solve the Travelling Salesman Problem using Genetic Algorithms. The salesman travels in cities and return to the starting city with minimal cost. He is not allowed to cross a city more than

once. In this problem, assume that all the cities are interconnected. The cost indicates the distance between the two cities.

Hints:

1. Use Genetic Algorithm to select the cities randomly, which becomes the initial population for this problem.
2. Fitness function may be defined as minimum cost. Initially fitness function is set to maximum value and for each travel, the cost is calculated and compared with the fitness function. The new fitness value is assigned to the minimum cost. Chose initial population randomly and that becomes a parent. For next generation, apply cyclic crossover operation.
3. Solve the problem for size N.

Appendix A: Bibliography

1. Ivan Bratko, Prolog Programming for Artificial Intelligence.

2. Ulle Endriss, An Introduction to Prolog Programming (Lecture Notes).