Prolog Tutorial and Lab Exercises

Mathematical and Computational Logic
Yachay Tech University

1. Introduction to Prolog

Prolog (**PROgramming in LOGic**) is a **declarative programming language** based on formal logic. Instead of giving step-by-step instructions, you state **facts** and **rules**, and then ask **queries** to see what the system can infer.

Prolog is widely used in **Artificial Intelligence**, **natural language processing**, **knowledge representation**, and **reasoning systems**.

2. Basic Concepts

2.1 Predicates

A **predicate** describes a relation among objects. It is identified by its **name** and its **arity** (number of arguments).

Example:

```
parent(john, mary).
```

- Predicate: parent/2 (two arguments).
- Meaning: John is Mary's parent.

2.2 Facts

A **fact** is something unconditionally true.

Example:

```
likes(mary, pizza).
likes(john, pasta).
```

This says Mary likes pizza, and John likes pasta.

2.3 Rules

A **rule** defines a relation in terms of other facts or rules. General form:

```
head :- body.
```

Meaning: head is true if body is true.

Example:

```
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

- X is a grandparent of Y if X is a parent of Z and Z is a parent of Y.

2.4 Queries

Queries ask Prolog if something is true, or what values satisfy a condition.

Example:

```
?- parent(john, mary).
true.

?- parent(john, Who).
Who = mary.
```

2.5 Variables

- Variables start with a capital letter (X , Y , Person).
- They represent unknowns that Prolog will solve for.

2.6 Backtracking

Prolog tries to satisfy queries by searching through rules and facts, and if one path fails, it **backtracks** to try another.

Example:

```
likes(mary, pizza).
likes(mary, pasta).

?- likes(mary, Food).
Food = pizza ;
Food = pasta.
```

2.7 Built-in Predicates

- write/1 : prints text.
- nl/0 : prints a newline.
- is/2 : arithmetic evaluation.
- =/2 : equality test.

2.8 Lists

Lists are fundamental in Prolog.

- Empty list: []
- Non-empty: [Head | Tail]

Example: defining membership

```
member(X, [X | _]).
member(X, [_ | T]) :- member(X, T). Query:
```

```
?- member(3, [1,2,3,4]).
true.
```

3. Hello World in Prolog

```
hello_world :-
        write('Hello, world!'), nl.
```

Run it in Prolog:

```
?- hello_world.
Hello, world!
true.
```

Explanation:

- hello_world/0 is a predicate with arity 0.
- Defined by a rule: it is true if write('Hello, world!') and nl succeed.
- write/1 prints text, nl/0 prints a newline.

4. Hands-on Mini Lab

This mini lab introduces Prolog step by step. Estimated duration: \~1 hour.

Step 1: Facts

Create a file family.pl with:

```
parent(john, mary).
parent(mary, susan).
parent(mary, bob).
parent(susan, alice).
```

Try queries:

```
?- parent(john, mary).
?- parent(mary, Who).
```

Step 2: Rules

Add a rule:

```
grandparent(X, Y) :- parent(X, Z), parent(Z, Y). Query:
```

```
?- grandparent(john, susan).
?- grandparent(john, Who).
```

Step 3: Variables & Backtracking

```
likes(mary, pizza).
likes(mary, pasta).
likes(john, pizza).
```

Query:

```
?- likes(mary, Food).
```

Observe how Prolog finds multiple solutions with ; .

Step 4: Built-ins

```
?- X is 2 + 3.
```

```
X = 5.

?- 3 = 3.
true.
```

Step 5: Lists

Add list membership:

```
member(X, [X | _]).
member(X, [_ | T]) :- member(X, T). Query:


?- member(2, [1,2,3,4]).
?- member(X, [a,b,c]).
```

5. Laboratory - Prolog

This lab combines the concepts above into a larger project.

Goal: Build a Knowledge Base and Query It

1. **Family Tree Knowledge Base**

2. Define at least 10 parent/2 facts with multiple generations.

3. Add rules for:
   - grandparent/2
   - sibling/2
   - ancestor/2 (recursive)

4. **Food Preferences**
5. Define facts like likes(alicia, pizza)
6. Add a rule food_friend(X, Y) that is true if X and Y like the same food.
7. **Math Utility**
8. Define factorial recursively
9. Define a rule sum_list(List, Sum) to compute the sum of a list.

10 **List Processing**

11. Implement length_list(List, Length) .

12. Implement append_list(List1, List2, Result) .

13. **Queries to Run**

14. Who are the ancestors of a specific person?

15. Who are siblings in your family tree?
16. Who are food friends?

17. What is the factorial of 6?
18. What is the sum of [2,4,6,8] ?
19. What is the length of [a,b,c,d] ?
20. Append [1,2] and [3,4] .

Summary

- **Predicates** are the heart of Prolog.
- **Facts** define unconditional truths.
- **Rules** define relationships based on conditions. • **Queries** let us ask questions.
- **Backtracking** finds multiple solutions.
- **Lists and recursion** provide powerful data processing.

With these building blocks, you can begin solving logical problems in Prolog.