Chapter 8

Prolog

1. Introduction to Prolog

Prolog (Programming in Logic) is a logic programming language associated with artificial intelligence and computational linguistics. It is declarative in nature, meaning you describe *what* you want rather than *how* to compute it. The program logic is expressed in terms of relations, represented as facts and rules.

Prolog programs consist of:

- **Facts** – things that are unconditionally true.
- **Rules** – conditional truths, dependent on other facts or rules.
- **Queries** – questions asked to the system to infer answers using existing facts and rules.

2. Converting English to Facts and Rules

Example 1 (Fact):

English Statement: "Ravi is a boy."

Prolog Fact:

```
boy(ravi).
```

Example 2 (Rule):

English Statement: "X is a grandfather of Y if X is the father of Z and Z is the parent of Y."

Prolog Rule:

```
grandfather(X, Y) :- father(X, Z), parent(Z, Y).
```

Example 3 (Query):

Query: "Is Ravi a boy?"

```
?- boy(ravi).
```

Output: true.

3. Prolog Terminology

- Atom: Constant values (e.g., ravi, apple, x).
- Variable: Identifiers starting with uppercase or underscore (e.g., X, Person, _Age).
- Predicate: Represents a relationship, followed by arguments (e.g., father(ram, shyam)).
- Clause: A fact or a rule.
- Goal: A query or sub-query Prolog tries to satisfy.

4. Variables

Variables in Prolog start with a capital letter or an underscore.

```
likes(ram, X).
```

Here, X is a variable and can match any value.

Example:

```
likes(ram, ice_cream).

likes(ram, chocolate).

?- likes(ram, What).
```

Output:

```
What = ice_cream ;

What = chocolate.
```

5. Control Structures

a. Conjunction (AND):

```
happy(X) :- rich(X), healthy(X).
```

Means X is happy if X is rich *and* healthy.

b. Disjunction (OR):

```
pass(X) :- studies(X).

pass(X) :- bribe(X).
```

Means X can pass if X studies *or* bribes.

6. Arithmetic Operators

Arithmetic operations in Prolog are handled using the is keyword.

Operators:

- +, -, *, /, mod, //, **

Example:

```
area(R, A) :- A is 3.14 * R * R.


?- area(5, A).
```

Output:

```
A = 78.5.
```

7. Matching (Unification)

Unification is the process by which Prolog matches terms. It is not assignment; it's more like pattern matching.

Example:

```
parent(john, mary).


?- parent(john, X).
```

Output: X = mary.

Invalid Unification Example:

```
?- parent(X, mary), X = bob.
```

If there is no such fact, it will return false.

8. Backtracking

Prolog tries to satisfy goals from left to right. If one path fails, it backtracks and tries another.

Example:

```
likes(john, pizza).

likes(john, burger).

likes(john, pasta).


?- likes(john, X).
```

Output:

```
X = pizza ;

X = burger ;

X = pasta.
```

Prolog tries each option using backtracking until all solutions are exhausted.

9. Cuts (!)

The cut operator ! is used to control backtracking. When encountered, it commits to the choices made so far and prevents Prolog from considering alternative solutions.

Example without cut:

```
grade(X, 'A') :- marks(X, M), M > 80.

grade(X, 'B') :- marks(X, M), M > 60.
```

With cut:

```
grade(X, 'A') :- marks(X, M), M > 80, !.

grade(X, 'B') :- marks(X, M), M > 60, !.

grade(X, 'C').
```

Now once a condition is met, Prolog won’t backtrack to check other grades.

Conclusion

Prolog provides a powerful mechanism to represent knowledge and reason about it using facts, rules, and queries. Its unique features like unification, backtracking, and cut operator give developers fine-grained control over logic-based problem solving, making it suitable for domains like expert systems, natural language processing, and AI.