LOGICAL PROGRAMMING WITH PROLOG

INTRODUCTION

- Logic Programming is a paradigm that uses formal logic to express computation.
- In Prolog, you define a knowledge base using **facts** and **rules**, and solve problems by querying this base.
- Instead of specifying how to compute, you describe what relationships hold and let Prolog derive answers through **unification** and **backtracking**.

WHAT IS PROLOG?

- Prolog = **PRO**gramming in **LOG**ic.
- It's a logic programming language based on formal logic.
- Instead of telling the computer how to do something, you tell it what is true (**facts**) and what **rules** hold.
- Prolog then answers **queries** using **unification** and **backtracking**.

WAYS TO RUN PROLOG

- **Using Prolog on Linux**
- **Using Prolog on Windows** (download exe from https://www.swi-prolog.org/Download.html and install)
- **Using the online swish tool for Prolog** (https://swish.swi-prolog.org/)

KEY CONCEPTS - FACTS

- Facts are statements that are always true.
- Syntax: predicate(arguments).
- Examples:

```
father(john, mary).   % John is the father of Mary
likes(alice, pizza).  % Alice likes pizza
```

KEY CONCEPTS - RULES

- Rules define **relationships or logical implications**.
- Syntax: head :- body. → "head is true if body is true"
- Example:

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

- **Recursion** in rules allows multi-level reasoning:

```
ancestor(X,Z) :- parent(X,Z).          % Base case
ancestor(X,Z) :- parent(X,Y), ancestor(Y,Z).  % Recursive case
```

KEY CONCEPTS - QUERIES

- Queries ask Prolog to find **values that satisfy facts/rules**.
- Syntax: ?- predicate(arguments).
- Example:

  ?- father(john, X). % Who are John's children?

  X = mary ;

  X = mark.

  - Multiple answers are accessed using ; (backtracking).

KEY CONCEPTS - VARIABLES VS CONSTANTS

- **Variables**: Start with capital letters.

  Example: X, Person, Movie.

- **Constants/Literals**: Start with lowercase letters.

  Example: john, mary, pizza.

KEY CONCEPTS - UNIFICATION

- Prolog tries to **match the query with facts/rules**.
- Variables get **bound** to values that make the query true.
- Example:

```
?- father(john, X).  % X unifies with mary and mark
```

KEY CONCEPTS - BACKTRACKING

- Prolog explores all possible solutions using **depth-first search**.
- If one path fails, it goes back and tries another.
- Used in **finding multiple matches**:

```
?- pair(X,Y).  % All combinations from two lists
```

KEY CONCEPTS - NEGATION

- Represented as \+ Predicate (negation by failure).
- True if Prolog **cannot prove the predicate**.
- **\+ Goal** succeeds **if Goal cannot be proven true**.
- Example:

```
can_eat(Person, Food) :- likes(Person, Food), \+ dislikes(Person, Food).
```

KEY CONCEPTS - CUT OPERATOR (!)

- Prevents backtracking past a certain point.
- Useful for **decision-making** to avoid unnecessary checks.
- Example:

```
grade_pass(Student) :- grade(Student, a), !.   % automatically passes if grade is 'a'
grade_pass(Student) :- grade(Student, b), check_other_conditions(Student).
```

KEY CONCEPTS - LISTS

- Lists store multiple values.
- Syntax:

```
[]          % empty list
[H|T]       % head H and tail T
[a,b,c]     % list of elements
```

- Example rules:

```
my_member(X, [X|_]).          % X is head
my_member(X, [_|Tail]) :- my_member(X, Tail).  % search tail
```

KEY CONCEPTS - RECURSIVE RULES

- Recursion is heavily used in Prolog:
  - Ancestor/descendant relationships
  - Generating sequences or lists
- Base case + recursive case.

KEY CONCEPTS - ARITY

- Number of arguments a predicate takes.
- Example:

  father/2 → father(X,Y)

  likes/2 → likes(Person, Food)