

CSE 259 - Logic in Computer Science

Recitation-2

Basic Building Blocks

Waqar Hassan Khan



What is Prolog?

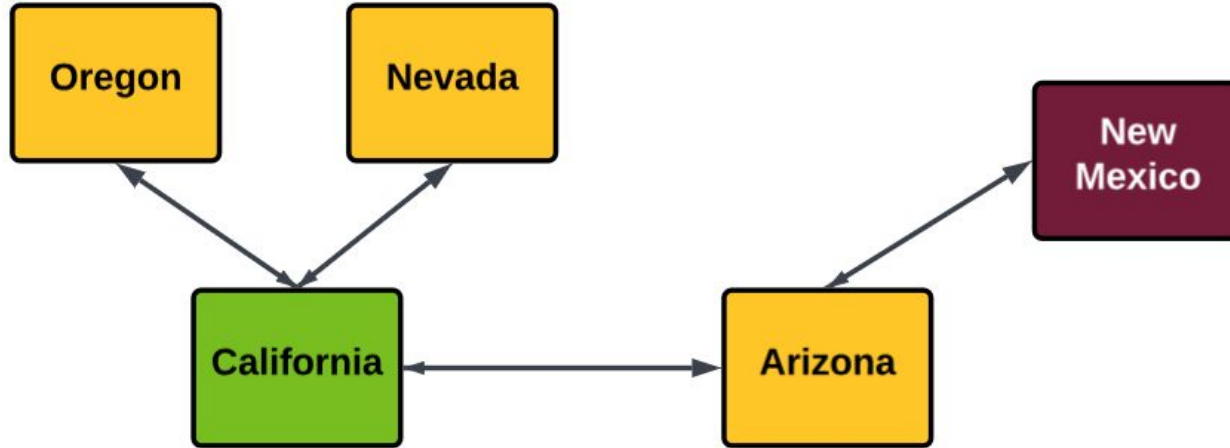
- Prolog is a **logic programming language**
- Prolog is intended primarily as a **declarative programming language**

Declarative Programming Language

Declarative programming is when you write your code in such a way that it describes what you want to do, and not how you want to do it. It is left up to the compiler to figure out the how.

Example: SQL, Prolog (Obviously :D)

An example



Suppose, we are in California. Which states can we visit from here? (We can travel if the states are adjacent)

Sample code for the example - Java

```
1 public class StateTravel {
2
3     public static void main(String[] args) {
4         boolean oregonConnected = true;
5         boolean nevadaConnected = true;
6         boolean arizonaConnected = true;
7         boolean newMexicoConnected = false; // Arizona only connects to New Mexico
8
9         // Individual if cases for each state
10        if (oregonConnected) {
11            System.out.println("Can travel to Oregon from California.");
12        } else {
13            System.out.println("Cannot travel to Oregon from California.");
14        }
15
16        if (nevadaConnected) {
17            System.out.println("Can travel to Nevada from California.");
18        } else {
19            System.out.println("Cannot travel to Nevada from California.");
20        }
21
22        if (arizonaConnected) {
23            System.out.println("Can travel to Arizona from California.");
24        } else {
25            System.out.println("Cannot travel to Arizona from California.");
26        }
27
28        if (arizonaConnected && newMexicoConnected) {
29            System.out.println("Can travel to New Mexico from California (via Arizona).");
30        } else {
31            System.out.println("Cannot travel to New Mexico from California.");
32        }
33    }
34 }
35
```

Sample code for the example - Prolog

```

1      next_to(oregon, california).
2      next_to(california, oregon).
3
4      next_to(california, nevada).
5      next_to(nevada, california).
6
7      next_to(california, arizona).
8      next_to(arizona, california).
9
10     next_to(arizona, new_mexico).
11     next_to(new_mexico, arizona).
12
13     travel(A, C) :- (next_to(A, C) ; (next_to(A, B), next_to(B, C), A \= C)).

```

Basic Syntax

Term

- Basic building blocks of programs and data structures
- Similar to how variables, constants, and expressions are in other programming languages.

Basic Syntax

Different Terms

- **Variable:** starts with an **uppercase letter** or with an **underscore**. Example: **A**, **Ab**, **_a**
- **Constant :** **atom** or **number**. Atom starts with **lowercase**. Example: **john**, **apple**, **23**, **45**, etc.
- **Compound term :** formed by combining other terms using functors and parentheses. A functor is an atom that represents a function or relation symbol, and arguments are terms separated by commas and enclosed in parentheses. Example: **person(john, 25)** “**person**” is a **functor**. “**john**” and “**25**” are atoms (**term!!**)

Basic Syntax

Predicate

- Fundamental concept used to define relations
- Represent statements or propositions that can be true or false
- Predicate name should be an **atom**
- There can be 0 or more arguments. Example: **likes(john, apple)**

Basic Syntax - Practice!

Which one of these are a variable?

1. X
2. y
3. _y
4. Fun

Basic Syntax - Practice!

Which one of these are a variable?

1. X

2. y

3. _y

4. Fun

Basic Syntax - Practice!

Which one of these are an atom?

1. X
2. y
3. _y
4. Fun

Basic Syntax - Practice!

Which one of these are an atom?

1. X

2. y

3. _y

4. Fun

Basic Syntax - Practice!

Which one of these are a predicate?

1. X
2. y
3. $_y$
4. $\text{Fun}(\text{car})$.
5. $\text{fun}(\text{Car})$

Basic Syntax - Practice!

Which one of these are a predicate?

1. X

2. y

3. _y

4. Fun(car).

5. fun(Car)

Basic Syntax

Rule

- Contains **four** parts:

Head, :-, Body, and a dot (.)

fun(X) :- red(X), car(X).

This example means - if X is a car and is red then it is fun.

- Symbols used:

Implication :-

Conjunction , (**and**)

Disjunction ; (**or**)

Basic Syntax

Facts

- Represents a relation between items
- Should **always begin with a lowercase letter** and **end with a full stop**. The facts themselves can consist of any letter or number

Lets solve a problem!

1. Define some facts

- “ana”, “casey”, “grace” are mothers
- “bob”, “dan”, “esion”, “frank” are fathers

2. Define two simple rules

- If someone(X) is a mother then she is a female
- If someone(Y) is a father then he is a male

3. Ask the following questions-

- Is “ana” a female or male?
- Is “frank” a female or male?

Lets solve a problem!

```
1  % facts
2  mother(ana).
3  mother(casey).
4  mother(grace).
5
6  father(bob).
7  father(dan).
8  father(esion).
9  father(frank).
10
11 % rules
12 female(X) :- mother(X).
13 male(Y) :- father(Y).
```

Lets solve a problem!



SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)

File Edit Settings Run Debug Help

Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run `?- license.` for legal details.

For online help and background, visit <https://www.swi-prolog.org>
For built-in help, use `?- help(Topic).` or `?- apropos(Word).`

`?-`

`% e:/Programming/TA/ASU-CSE-259-Prolog/old/Recitation-2/family-relationship.pl compiled 0.02 sec, 9 clauses`

`?- female(ana).`

true.

`?- male(ana).`

false.

`?- female(franc).`

false.

`?- male(franc).`

true.

`?-`

Lets solve a problem!

Running the code

- [Windows] Open SWI Prolog -> File -> consult -> select the file
- [Mac] Open SWI Prolog -> write `consult("full path of the file").`

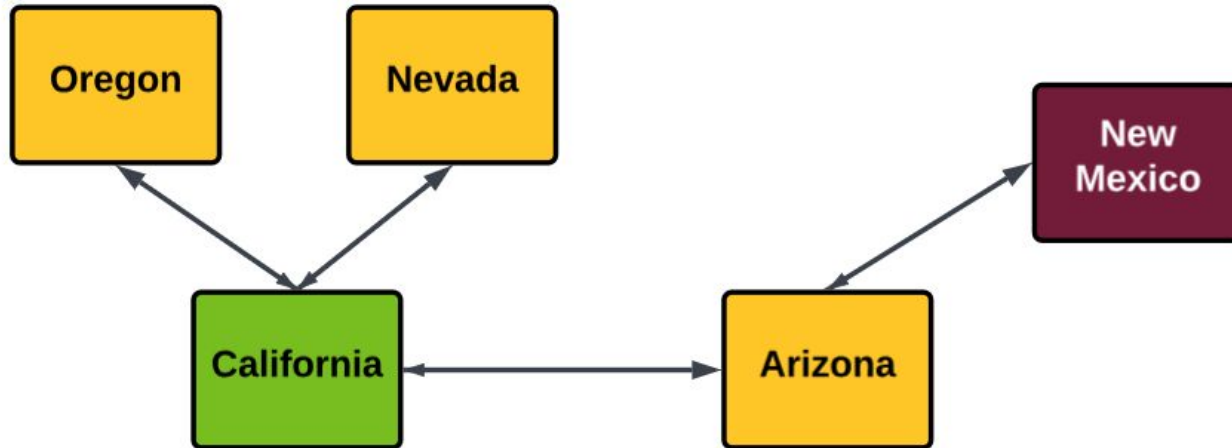
Inside consult, don't forget the double quotes!

Understanding the result

- True or false will be displayed.

Lets solve another problem!

Where can we travel from California?



Lets solve another problem!

```
1  % facts
2  next_to(oregon, california).
3  next_to(california, oregon).
4
5  next_to(california, nevada).
6  next_to(nevada, california).
7
8  next_to(california, arizona).
9  next_to(arizona, california).
10
11 next_to(arizona, new_mexico).
12 next_to(new_mexico, arizona).
13
14 % rule - when can we travel from state A to state B?
15 √ travel(A, C) :- (
16     | next_to(A, C);
17     | (next_to(A, B), next_to(B, C), A \= C)).
```

Lets solve another problem!

```
% e:/Programming/TA/ASU-CSE-259-Prolog/Recitation-2/travel.pl compiled 0.00 sec, 9 clauses
?- travel(california, arizona).
true .

?- travel(oregon, california).
true .

?- travel(oregon, new_mexico).
false.

?- travel(oregon, arizona).
true.

?-
```

Lets solve another problem!

Problem

- `travel(oregon, new_mexico).` did not work!