Prolog and AI

Karthik Reddy(IMT2018011), Suchith (IMT2018019), Gopal (IMT2018083),

Programming Languages, May 2021

Introduction

AI:

- Artificial intelligence is the getting of computers to do things that seem to be intelligent. The hope is that more intelligent computers can be more helpful to us–better able to respond to our needs and wants, and more clever about satisfying them.
- AI is accomplished by studying how human brain thinks and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.
- It has gained prominence recently due, in part, to big data, or the increase in speed, size and variety of data businesses are now collecting. AI can perform tasks such as identifying patterns in the data more efficiently than humans, enabling businesses to gain more insight out of their data.

AI includes:

- Getting computers to remember complicated interrelated facts, and draw conclusions from them (inference), Getting computers to plan sequences of actions to accomplish goals (planning), Getting computers to offer us advice based on complicated rules for various situations (expert systems).
- AI has applications in all fields of human study.
  - Machine Learning
  - Natural Language Processing
  - Expert Systems
  - Robotics
  - Game playing and many more

Prolog:

- Prolog is a logical and declarative programming language. The name itself, Prolog, is short for PROgramming in LOGic.
- Prolog is intended primarily as a declarative programming language.
- In prolog, logic is expressed as relations (called Facts and Rules). The core heart of prolog lies at the logic being applied.
- Formulation or Computation is carried out by running a query over these relations.
- In prolog, We declare some facts. These facts constitute the Knowledge Base of the system.

- We can query against the Knowledge Base. We get output as affirmative if our query is already in the knowledge base or it is implied by Knowledge Base, otherwise, we get output as negative.
- So, Knowledge Base can be considered similar to a database, against which we can query.
- Prolog facts are expressed in a definite pattern. Facts contain entities and their relation.

Prolog Syntax

$$
\begin{array}{lll}
\text{const} & \rightarrow & \textit{atom} \mid \textit{number} \\
\text{var} & \rightarrow & \textit{ucase}\ \ \textit{string} \mid \_\textit{string} \\
\text{atom} & \rightarrow & \textit{lcase}\ \ \textit{string} \\
\text{letter} & \rightarrow & \textit{ucase} \mid \textit{lcase} \\
\text{string} & \rightarrow & \textit{epsilon} \mid \textit{letter}\ \ \textit{string} \\
 & & \mid \textit{number}\ \ \textit{string} \mid \_\textit{string} \\
\text{ucase} & \rightarrow & A \mid \ldots \mid Z \\
\text{lcase} & \rightarrow & a \mid \ldots \mid z \\
\text{number} & \rightarrow & \ldots
\end{array}
$$

Prolog programming

Facts: **English meanings**

food(burger). # burger is a food

food(sandwich). # sandwich is a food

food(pizza). # pizza is a food

lunch(sandwich). #sandwich is a lunch

dinner(pizza). # pizza is a dinner

Rules:

meal(X) :- food(X). # Every food is a meal or Anything is a meal if it is a food

Queries / Goals:

?- food(pizza). # Is pizza a food?

Prolog returns :- Yes

?- meal(X), lunch(X). # Which food is meal and lunch?

Prolog returns X := sandwich.

?- dinner(sandwich). # Is sandwich a dinner?

Prolog returns :- No

Prolog and AI

Prolog has a clear contribution to solving a series of AI problems. There are many features that make Prolog suitable as a programming language for developing AI applications. Some of them are:

Declarative nature:

- Logical programming removes the imperative nature of other languages and allows programmers to solve a problem by describing the problem itself rather than defining a solution.
- The programmer writes programs by declaring the facts and the rules that apply to the problem at hand and then makes queries in order for Prolog to return valid solutions.
- This high level of abstraction makes Prolog suitable for AI applications where the programmers should give emphasis on the problem.

Prolog and AI

Backtracking:

- When a search path ends at a dead end, the backtracking mechanism of Prolog retreats back down the search path to try another path. This feature makes Prolog exceptionally suitable for a number of search problems that AI faces. It also helps in finding more than one solution to the problem. In that case, backtracking is forced after finding the first solution.
- Because a great number of AI problems can be represented as a problem of finding the right path in the search space, the built-in depth-first mechanism of Prolog, accompanied by backtracking, becomes very powerful.

Prolog and AI

Pattern matching and Unification:

- The use of unification to find the most general common instance of two formulas or patterns makes pattern matching a build-in feature of Prolog.
- This intelligent feature can assist AI problem solving where in many cases the decisions that are made are based on situation matching.
- This Prolog ability can be found really helpful in specific areas of AI, such as natural language processing, intelligent database search, and some others.

Prolog and AI

List handling mechanism:

- The data structure of List is very important for handling AI problems (e.g., LISP which is also used for solving AI problems, stands for " LIST Processing").
- Lists are built-in in Prolog while in most other languages they are not, making faster and easier the writing of Prolog programs that require list handling.
- List's recursive nature allows the extensive use of recursion in problem-solving, providing an additional advantage for solving AI problems with Prolog.

Prolog and AI

Don't care, and don't know nondeterminism:

- In the execution of a Prolog program, the nondeterminism feature is really apparent. Although the rule that will execute is the first one that matches the goal, we can ask for more than one solution.
- Using the backtracking mechanism, alternative rules will apply and other valid solutions will be found.
- This introduces a) "don't know nondeterminism" implying that all possible ways to find the solutions will be followed because the execution does not know how to find the solution,
- b) "don't care non-determinism" meaning that we just need one solution and we do not care which solution is that among the many that exist.
- Both of these types of nondeterminism are considered useful in AI applications, especially for those dealing with logic.

Applications

Problem solving is a process of generating solutions from observed data. And is defined by its 'elements' and their 'relations'.
The problem can then be solved by using the rules, in combination with an appropriate control strategy, to move through the problem space until a path from an initial state to a goal state is found. This is known as 'search'.

examples:

- Learning Neural Network - Perceptron
- Supervised Learning - linear classifier
- Expert Systems

Expert Systems

Expert System :- Knowledge + problem-solving.

- **Knowledge Base**
- **Inference Engine**
  - Forward chaining
  - Backward Chaining
- **User Interface**

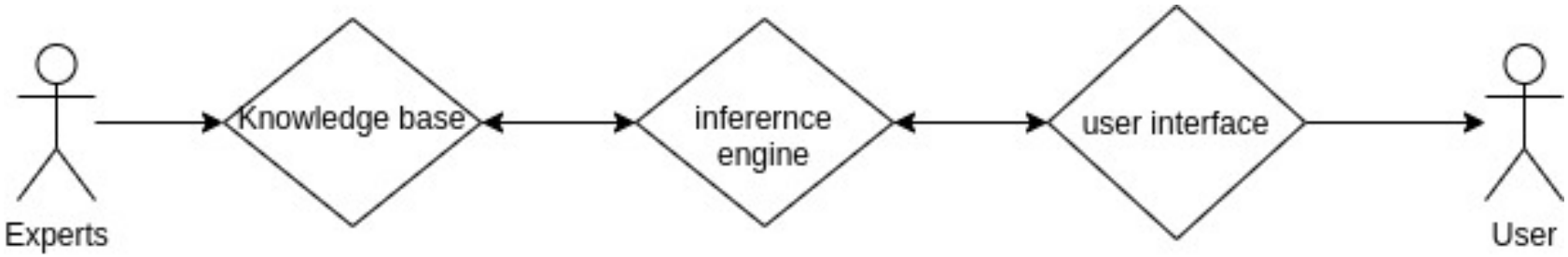

Figure: Flow Diagram

References

- http://www.sci.brooklyn.cuny.edu/k̃opec/Publications/Artificial%20Intelligence -Search%20Methods.htm
- https://www.cet.edu.in/noticefiles/271_AI%20Lect%20Notes.pdf
- http://www.let.rug.nl/bos/lpn//

Thank You