11/27/2020

Prolog Programming

By Connor Bartal & Sean Rogers

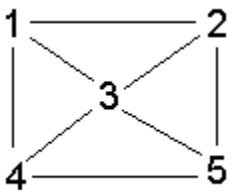
PROLOG

What is Prolog:

Prolog stands for "Programming in logic. A person typically uses a programming language to tell a computer to perform whatever task they desire. In other languages the programmer tells that the computer what a variable is. For example, you can define int a = 3 (telling the computer the variable a is the integer 3). The program than can set, evaluate, or change that variable based on the programs following execution. However, in prolog the computer is told less how to complete the task. In prolog a set of facts are defined, and these facts could be used to define characteristics, such as ("Smith is the father of Adam"). The Prolog program can than ask the computer about the facts already given and the computer will return the provided answers. Prolog allows people to use recursion instead of iteration, this creates unique simplicity for solving problems, making Prolog programs smaller and easier to understand when coping with large real-life problems.

Prolog Graphs

The most interesting thing we found is that prolog can easily represent graphs with its rule system. For example, the graph on the right could be represented with the following code.




```
edge(1,2).
edge(1,4).
edge(1,3).
edge(2,3).
edge(2,5).
edge(3,4).
edge(3,5).
edge(4,5).

connected(X,Y) :- edge(X,Y) ; edge(Y,X).
```

With the use of this disjunction, (;) It is the same as writing the following code. Which connects the graph both ways, this can of course be changed to connect one way or to include lengths to find the shortest distance.

```
connected(X,Y) :- edge(X,Y).
connected(X,Y) :- edge(Y,X).
```

This allows a connection between all the edges allowing for powerful algorithms to find the paths, graphing, or even mathematical formulas. Prolog sets up a powerful and easy system to build graphs, with nodes, edges, lengths because of its easy to use rule system of predefined facts. Graphs are a powerful mathematical tool in some cases and prolog just makes it easier to use.

We were able to take advantage of graphs throughout this project. When determining whether the interviewer could see a user's posts, we created a graph with directional connections. This allowed us to properly implement blocking. Once we got this working properly, it came down to simply asking: "Is there a path between A and B?" If there was not, simply terminate the program. If there was, then we need to determine if there are any bad posts the user needs to review.

Ease of Use and Understanding

We decided that we like prolog's easy to read fact system and found it interesting that you could write English statements out than define it in prolog facts. For example, "John is the father of Jason" could be written in prolog in the following form

```
father_child(John, Jason)
parent_child(X, Y) :- father_child(X, Y).
```

Even those with very little programming experience could look at that fact and understand the English statement that was made earlier. We believe that the system could even be used to teach new programmers, programming logic as it is easier to read and more compact than other languages.

Prolog in AI

We believe that prolog could be a powerful tool when used with Artificial Intelligence programming. While researching this we even found out that Prolog is used in IBM Watson a powerful AI platform for business. Why we believe prolog would be powerful in AI problems is its use of backtracking.

Backtracking is the idea that even if a search path ends at a dead end, the prolog Backtracking mechanism allow prolog to retreat down a search path and attempt to find another path. It allows AI to find more than one solution to a problem if backtracking is forced after finding the first solution. This could assist AI in finding more efficient solutions. It could also provide a wealth of solutions to the problem so that options are available to the programmer or those he is working for.

Another strength that prolog has in AI problems is its declarative nature. By being a declarative programming language, it allows the programmer to solve a problem by describing the problem rather than defining the solution. The steps in this situation would be that the programmer writes programs by declaring the facts at hand and the rules that apply to those facts and then would make queries for Prolog to return valid solutions. This creates high level abstraction which can be suitable for AI applications especially with the focus should be on the problem itself.