Tecnológico de Monterrey

Course
Programming Languages

Final project's Architecture Document
Tic-tac-toe Game (Human vs Computer)

Teacher
Obed Muñoz

Student
Jorge Padilla
A00570894

Date
November 22nd 2018

Before we dig in ourselves in this document, it is important to highlight the fact that this project was possible to be done thanks to the next reference I found on the internet: https://www.cpp.edu/~jrfisher/www/prolog_tutorial/contents.html. To be honest, I had a really bad time trying to figure out how to even begin this project, because I am obviously not a Prolog expert. It is really complicated to use not only a different programming language, but a different programming paradigm. But at the end, I found Prolog and the Logic Programming Paradigm to be really interesting and this reference would help me understand better the core concepts of this programming language and paradigm. In addition, I thought it was really cool and interesting to see how a connection between a Java program and a Prolog program can be made using threads, which will be seen later in this document.

Tic-tac-toe Game (Human vs Computer)

For this project, a tic-tac-toe game was made which implements the alpha-beta algorithm, so the user can play against the computer. For the GUI (Graphical User Interface), the programming language used was Java, while the programming language used for the computer's logic was Prolog. In order to successfully connect the Java GUI program and the Prolog logic, a second Java program was implemented, which all it does is getting a server up and running so the Java GUI program and the prolog logic can communicate with each other.

Java Progam

The logic of this program consists on creating a thread with the connector (the bridge program between the Java GUI program and the Prolog logic program), which allows all programs to connect to the same port (In this case, port 8080 was used). When this happens, all connected programs have an input and an output, therefore everything the user writes (Java GUI program) is broadcasted as an output so the other programs can read it.

The java GUI program then is all about declaring the variables for the GUI . After that, the thread does its thing and the connection to the port is established. Once the connection is made, the Prolog program connects to the port as well.

All nine spaces of the game have (X, Y) coordinates and each value is assigned to the corresponding GUI button. What this means is that when the user clicks on a button, the program broadcasts that button's coordinates and the Prolog program receives it as a string through its input sream. Then, the computer (Prolog program) defines its next move based on the algorithm mentioned earlier. Finally, the Java program reads the computer's answer and so on until either the player or the computer wins or all of the spaces have been filled (tie).

Prolog program

Here we will cover the most important facts, rules, predicates and general aspects of the program. Having said that, what the program does is:

- Define the game board as a list and each space is numbered from 1 through 9.
- Use *mark(player, board, X, Y).* to generate all possible movements that can be made.
- Use the next rule to keep the game board updated:

- *record(Player, X, Y) :- retract(board(B)), mark(Player, B, X, Y), assert(board(B)).*

- Implement *win(Board, o).* to declare the possible options or moves to win.
- Implement *open(Board, o).* to declare all available spaces in the game board.
- Implement he alpha-beta algorithm.
- Connect to the server.

% The connection to the port is made

```
connect(Port) :-
        tcp_socket(Socket),
        gethostname(Host), % local host
        tcp_connect(Socket,Host:Port),
        tcp_open_socket(Socket,INs,OUTs),
        assert(connectedReadStream(INs)),
        assert(connectedWriteStream(OUTs)).

:- connet(8080).

ttt :-
        connectedReadStream(IStream),
        % The user's move (Java input) is read
        read(IStream,(X,Y)),
        % The game board is updated with the user's recent move
        record(x,X,Y),
        board(B),
        % Find the best move with the alpha-beta algorithm
        alpha_beta(o,2,B,-200,200,(U,V),_Value),
        %  The game board is updated with the computer's recent move
        record(o,U,V),
        connectedWriteStream(OStream),
        % The computer's move (Prolog input) is sent to the Java program
        write(OStream,(U,V)),
        nl(OStream), flush_output(OStream),
        ttt.

:- ttt.
```

Conclusions

Throughout this course I realized there are a lot of programming languages I have not even heard about in my life, and that each an every one of them was meant to be made for something specific. Although you can mess around with them and do different things with all of them. The goal here is to identify the advantages and disadvantages of the programming language you intend to use, and analyze whether the programming language you are considering is in fact your best option.