Contents

Abstract

Keywords:PROLOG, Inference engine

Checkers is a two-player abstract strategy board game. This project has the goal of developing a program that takes the configuration of the checkers board as input and generate moves that can defeat a human opponent. It relies on the inference engine of PROLOG, a logic programming language. The user interface has been implemented using Java.

Figure 1: A Checkers Board

1 INTRODUCTION

Checkers is a strategy game for two players on the opposite sides of the gameboard involving alternate moves. The board size in general is 8×8, and for this size the checkers board is the same as that of chess.

Abstract strategy games such as chess and checkers have too many possibilities at any instant of the game for a human to completely analyze. However, all the information for making the best move is present in the board and random generators such as a dice are absent. The players decide their moves based on their knowledge of the game and a search through the possible legal moves. Such games have been of interest to people in the artificial intelligence community, who wish to see if sufficient knowledge of the game and proper search skills could be programmed into a computer to make it capable of beating a expert human opponent.

The rules for 8×8 checkers are as follows.

1. Each player has twelve pieces placed on the twelve dark squares closest to him.
2. Each piece can move diagonally and capture diagonally.
3. A player can capture an opponent piece by jumping over it to the empty square immediately behind it.
4. A piece is crowned if it reaches the last file of the oppositions ranks. A crowned piece can move and capture in forward as well as backward direction.
5. A player is declared the winner if he has captured all his opponents pieces or if his opponent has no legal move in his turn.

This project involves making a checkers program with a variation in the original rules : The goal of a player is to crown one of his pieces before his opponent crowns one of his, or to leave his opponent with no legal moves in his turn (either by capturing all of his pieces or blocking all of their movements). This variation on the rules helps to focus on the strategic aspect of the game with the pieces. The project uses the inference mechanism of PROLOG to generate the moves for the computer.

2 LITERATURE REVIEW

In a logic programming language, programs consist of a set of axioms or rules that specify relations between objects. In such a programming language, a computation is infact a proof, and the output of the program is extracted from the proof.[1] So from a computational point of view, the problem is to find term values for variables that make the formula valid. Sterlin and Shapiro[2] suggest that the computation can be viewed as a deduction of consequences of the program.

The game of checkers is the largest game to be solved by a computer till date. Chinook, the program which solved the game and the Man-Machine Champion, is the first program to become World Champion in any non-trivial game of skill. It uses deep searches with alpha-beta pruning, evaluation functions with 25 heuristic components and opening and endgame databases.[3] Various kinds of traps and many possible endgame scenarios have been systematically analysed in checkers literature.[4]

3 PROBLEM STATEMENT

Given a checkers board of size 2n×2n, and the following rules and winning configurations, write a checkers program that defends white's position against a human opponent playing black.

Rules :
White moves first and the players move alternately.

Captures are compulsory.

Winning Configuration:
1. If a player can make no legal moves and its his turn, then he loses the game and his opponent is declared the winner.
2. If a player captures all of his opponents pieces then he is declared the winner.
3. If a white piece reaches the $2n^{th}$ row, white is declared the winner.
4. If a black piece reaches the 1^{st} row, black is declared the winner.

4 DESIGN

Initial Configuration :

Given a square board of size 2n such that for 1≤ i , j ≤ 2n

$$\begin{aligned} colour(i,j) &= black \quad if\, i+j \,=\, even. \\ &= white \quad if\, i+j \,=\, odd. \\ piececol(i,j) &= black \quad if\, a\, black\, piece\, occupies\, it. \\ &= white \quad if\, a\, white\, piece\, occupies\, it. \\ &= empty \quad if\, no\, piece\, occupies\, it. \end{aligned}$$

The initial configuration of the board is :

$$\begin{aligned} piececol(i,j) &= white \quad 1 \le i \le n-1 \; and \; colour(i,j) = black. \\ &= black \quad n+2 \le i \le 2n \; and \; colour(i,j) = black. \\ &= empty \quad otherwise \\ turn &= white \end{aligned}$$

Legal Moves:

Normal Move:
A white pawn can move from position (i, j) to

$$\begin{aligned} &(i+1, j+1) \;\; if \; i+1 \le 2n, \;\; j+1 \le 2n \; and \; piececol(i+1, j+1) = empty. \\ &(i+1, j-1) \;\; if \; i+1 \le 2n, \;\; j-1 \ge 1 \; and \; piececol(i+1, j-1) = empty. \end{aligned}$$

A black pawn can move from position (i, j) to

$$\begin{aligned} &(i-1, j+1) \;\; if \; i-1 \ge 1, \;\; j+1 \le 2n \; and \; piececol(i-1, j+1) = empty. \\ &(i-1, j-1) \;\; if \; i-1 \ge 1, \;\; j-1 \ge 1 \; and \; piececol(i-1, j-1) = empty. \end{aligned}$$

Capture Move:
A white pawn can move from position (i, j) to

$$\begin{aligned} (i+2, j+2) \quad & if \; i+2 \le 2n, \; j+2 \le 2n, \; piececol(i+2, j+2) = empty \; and \\ & piececol(i+1, j+1) = black. \\ (i+2, j-2) \quad & if \; i+2 \le 2n, \; j-2 \ge 1, \; piececol(i+2, j-2) = empty \; and \\ & piececol(i+1, j-1) = black. \end{aligned}$$

A black pawn can move from position (i, j) to

$(i-2, j+2)$ $\quad if\ i-2 \geq 1,\ j+2 \leq 2n,\ piececol(i-2, j+2) = empty\ and$
$piececol(i-1, j+1) = white.$

$(i-2, j-2)$ $\quad if\ i-2 \geq 1,\ j-2 \geq 1,\ piececol(i-2, j-2) = empty\ and$
$piececol(i-1, j-1) = white.$

Rules :

If one capture move is possible, then the player has to make that move.

If more than one capture moves are possible, the player can choose one of the capture moves.

If further capturing is possible with a piece used to make a capture, then those capture moves are compulsorily made in this turn.

Winning Configuration:

If a player can make no legal moves and its his turn, then he loses the game and his opponent is declared the winner.

$$(\forall i, j : piececol(i, j) = white \vee piececol(i, j) = empty,$$
$$where\ 1 \leq i, j \leq 2n) \Longrightarrow winner = white.$$
$$(\forall i, j : piececol(i, j) = black \vee piececol(i, j) = empty,$$
$$where\ 1 \leq i, j \leq 2n) \Longrightarrow winner = black.$$
$$If\ \exists j\ : piececol(2n, j) = white,\ then\ winner = white.$$
$$If\ \exists j\ : piececol(1, j) = black,\ then\ winner = black.$$

This mathematical specification of the configuration of the board is the design of the problem. This mathematical specification can be passed to PROLOG as the set of facts and rules and can be used to generate a move using PROLOG's inference mechanism.

5 IMPLEMENTATION

The implementation of the game for the board of size 8×8 has been completed. It uses uses java version 1.6.0_31, OpenJDK 7 and swi-prolog version 6.6.5. The graphical interface has been made in Java Swing and the computer moves are generated by a prolog program. The program uses the number of vulnerable pieces at the end of the move as a heuristic to decide which move to play. Even with this simple heuristic, the program performs reasonably well. For a 4×4 board, the first player has a winning sequence of moves, and the computer is able to find it.

The move the program makes are, in order of preference:
1. An immediate winning move, i.e. queening of a piece, or capturing the last of the opponent's pieces.
2. A move that leaves as few of its pieces vulnerable to capture as possible.

The user makes a move by clicking a piece and then clicking the square to which the piece should be moved. The computer checks if the suggested move is valid. The state of the board is then represented as a string and written to a file. The prolog program is then called which reads the file and determines the state of the board. Then it searches for all possible moves and selects the move with the best value for the heuristic used. This move is read by the Java program using an object of the InputStreamReader class. The Java program then makes the suggested move on the board. If any player wins the game, a message window is displayed.

The link between the Prolog engine and the Java program has been established using file read and write operations. Alternatives, such as using sockets for communication between the programs were considered but file read and write were relatively easier to implement and hence preferred.

As captures are compulsory, the prolog program searches for possible captures first and considers normal moves only if no capture moves are possible.

The source code for the project along with the documentation will be made available at the repository
`https://github.com/sureshsudhakaran/checkers-ls2` as per the instructions of my guide.

6 CRITICAL ANALYSIS

The program lacks certain features and requires work in the following areas :

1. Because of the simple nature of the heuristic (the number of pieces left vulnerable after a move) , a human player can figure out what move the computer is likely to play in a given situation and plan his strategies accordingly.

2. The program is incapable of learning from its previous games. So the same sequence of moves can be used to defeat the computer over and over.

3. The heuristic becomes irrelevant when there is a definite win in a few moves. Many interesting tactics that lead to a win in checkers revolve around sacrificing a piece. Because offering a sacrifice increases the number of vulnerable pieces, the program can miss several winning opportunities.

7 FUTURE WORK

The following features can be added to improve the program:

1. Increasing the ply-depth of the program will improve its play drastically. It will also enable it to recognize traps.

2. Strategies specific to queening can be explored and implemented. More heuristics can be added to enable the incorporation of these strategies in the program.

3. The program can be extended to a standard checkers game that continues after queening.

4. Features such as undoing a move and suggesting a move for the human player can be added.

8 CONCLUSION

The project uses the mathematical formalisation of the checkers board's configuration and PROLOG's inference mechanism to generate moves on behalf of the computer. The generated moves are based on the number of pieces vulnerable to capture after that move. It can be improved further by increasing the ply-depth and choosing better heuristics. On adding the moves of the crowned pieces, it can be further enhanced to a full-fledged checker's program. By providing options for forward and backward jumps for normal and crowned pieces and making the required changes in the program, different variations of checkers which are popular (such as Italian dama, Russian Checkers etc) can be implemented using the same program as the base.

References

[1] Jean H. Gallier. *Logic for Computer Science Foundations of Automatic Theorem Proving*. Second edition, 2003.

[2] Leon Sterlin and Ehud Shapiro. *The Art of Prolog*. The MIT Press.

[3] Robert Lake Jonathan Schaeffer. *Solving the Game of Checkers*. MSRI Publications, 1996.

[4] Fred Reinfeld. *How to Win at Checkers*. Wilshire Book Company, 1983.