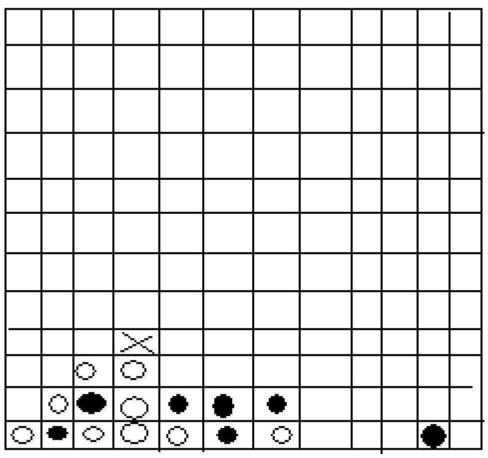
Autumn 2004
Instructor: B. Ross

COSC 3P71 Artificial Intelligence : Assignment 2

Due date: 12:00 noon, Tuesday November 2.
Lates: 12:00 noon, Friday November 5 (-25% deduction). No lates after this date.
Goal: Game playing with minimax search.
Languages: Any programming language of your choice.
Hand in: A listing of your source code, and a listing of your program execution for a representative sample of matches. Electronic submission of your code and data.



Connect 4. Connect-4 is played on a 12 by 12 grid set vertically on a table. Black and white chips are dropped into the grid, over one of the 12 columns, which fall and rest on the bottom of the grid, or the top chip in that column. When a column is full (12 chips in it), chips cannot be dropped down it. There are 2 players – black and white. Players take turns dropping their chip into the grid. The first player to have 4 chips of their colour connect either horizontally, vertically or diagonally is the winner.

For example, here's one possible game configuration. If the white player drops a chip in the 4th column, it will land on the X, and white will win (that column is white, as is the diagonal).

Write a program that plays Connect-4. The program should use the minimax procedure as discussed in class and the text (the use of alpha-beta pruning is optional). The program should accept the maximum ply P for the minimax search. Your program should play an external 'interactive' player (you!), who enters moves from the keyboard.

In order for your program to behave intelligently, you need to design your own static evaluator function. Any heuristic used should account for the connected pieces of both players' colours, the size of the connected sequences, and the number of such connected sequences. Experiment to find a good heuristic formula!

Some design considerations:

• **Architecture**: Remember to make your search engine as abstract as possible. It should call a Generate routine to generate all the possible transitions/moves from a given one, for whichever player is to make a move at that level in the tree. Your static evaluator should evaluate the heuristic score for a given configuration. Once these and other routines are appropriately modularized, the game playing mechanism becomes quite straight-forward. The textbook outlines the basic minimax procedure upon which you can base your design.

• **Loops**: Cycles in games are not as important an issue as with general search problems. Luckily, looping is impossible with Connect-4, because once a chip is placed in the grid it is never removed.

Hence each move adds a new piece to the board, and therefore each transition creates a new configuration.

• **Communication**: Your program should converse by generating the grid numbers of moves made by itself; you should enter grid numbers to make your moves. You might also like to dump out a representation of the board, in text format. That would be easier to follow than mere coordinates.

• **Creativity**: Feel free to design any new clever search optimizations, heuristics, or efficiency considerations that you like!

Bonus! [+10%] **Alpha-beta**: Add the alpha-beta pruning procedure to your algorithm!