AI Coursework

Thomas Rooney

March 5, 2013

```
Introduction to Artificial Intelligence Coursework[1]

(Submission by groups of two students allowed/welcome)
          Due date: Thursday 7  March 2013
Electronic submission (code + this worksheet) on CATE
```

1. Introduction

We will be looking at a two player board game called "war of life" which will be played on an 8x8 board. Player 1 will start with a random configuration of 12 blue pieces and player 2 will start with a similar random configuration of 12 red pieces. An example initial configuration might be (where b stands for "blue piece" and r stands for "red piece"):

| | The game terminates as follows: If at some stage no (red or blue) pieces at all are left on the board, then the game is drawn. If, when it is his/her turn, a player cannot move anywhere, then the game is declared a stalemate and is drawn. If one player has no pieces left on the board, then that player loses and the other player wins. If the game lasts for 250 moves without a winner, then it is declared an exhausted draw.

Part 1 – Getting to know the Game

Question 1

| |

Board states are represented in the program as pairs of lists, where the first list contains the co-ordinates of all the alive blue pieces and the second list contains the co-ordinates of all the alive red pieces. For example, this is a simple board state with two alive blues and one alive red:

[[[3,4],[5,7]],[[8,8]]]

Question 2

In the box below, write down the Prolog representation for the initial board state given in question 1.

| |

Use this representation, and the predicate

next_generation(+BoardState, -NextGenerationBoardState)

to check whether your answer for question 1 was correct. If it is, then run a further two generations, and put the resulting board states in the tables below:

| |1 | |Number of wins for player 1 (blue) | | |Number of wins for player 2 (red) | | |Longest (non-exhaustive) game | | |Shortest game | | |Average game length (including | | |exhaustives) | | |Average game time | |

Question 4

Does it look like there is any advantage to playing first if both players choose moves randomly? Answer in the space below.

Part 2 – Implementing Strategies

In this part, we will be implementing search strategies in order to improve a war of life playing agent's performance. They already have one strategy: random, which chooses any piece randomly and moves it randomly. The question is: can we implement any strategy which out-performs this one?

We will look at four strategies:

Bloodlust: This strategy chooses the next move for a player to be the one which (after Conway's crank) produces the board state with the fewest number of opponent's pieces on the board (ignoring the player's own pieces).

Self Preservation: This strategy chooses the next move for a player to be the one which (after Conway's crank) produces the board state with the largest number of that player's pieces on the board (ignoring the opponent's pieces).

Land Grab: This strategy chooses the next move for a player to be the one which (after Conway's crank) produces the board state which maximises this function: Number of Player's pieces – Number of Opponent's pieces.

Minimax: This strategy looks two-ply ahead using the heuristic measure described in the Land Grab strategy. It should follow the minimax principle and take into account the opponent's move after the one being chosen for the current player.

In your file my_wol.pl, write five predicates:

bloodlust(+PlayerColour, +CurrentBoardState, -NewBoardState, -Move). self_preservation(+PlayerColour, +CurrentBoardState, -NewBoardState, -Move). land_grab(+PlayerColour, +CurrentBoardState, -NewBoardState, -Move). minimax(+PlayerColour, +CurrentBoardState, -NewBoardState, -Move).

These predicates will implement the four strategies described above by choosing the next move for a player. They will all do the same thing: choose the next move for the player. PlayerColour will either be the constant b for blue or r for red. The CurrentBoardState will be the state of the board upon which the move choice is going to be made. The predicate will produce a NewBoardState, in the usual representation (pair of lists) which will represent the board state after the move, but before Conway's crank is turned. The predicate will also return the Move that changed the current to the new board state. This will be represented as a list [r1,c1,r2,c2] where the move chosen is to move the piece at co-ordinate (r1, c1) to the empty cell at co-ordinate (r2, c2).

It should be possible to run these strategies in the same way as the random strategy, using the constants bloodlust, self_preservation, land_grab. For example, if you load war_of_life.pl and my_wol.pl into Prolog, then type the query:

play(verbose, bloodlust, land_grab, NumberOfMoves, WinningPlayer).

this will play a game in which player 1 uses the bloodlust strategy and player 2 uses the land_grab strategy. Check that this is working OK by running a few games with different strategy pairings.

Part 3 – A Tournament

Question 6

We want to know which, if any, strategy is the best for playing this game, and we'll do this by using a tournament. Play as many games as time will allow for each pairing of strategies, and fill in the table over the page.

Question 7

If both players have the same strategy, for which strategies does it appear that playing first is an advantage/disadvantage? Answer in the space below:

Question 8

Imagine playing football in a gale force wind and where the pitch is extremely muddy. Here, it is hardly worth the players kicking the ball, because the environment plays too big a factor in the game. What evidence do you have against the claim that the environment plays a bigger part than the movement of pieces in the war of life game? The environment here is Conway's Crank, which is beyond the control of the players. Answer in the space below:

Submitting Your Work

Electronic submission: submit a zipped directory WOL.zip containing 1) a my_wol.pl file and 2) a file worksheet.pdf (your completed version of this worksheet).

Note: my_wol.pl should run under Sicstus on the lab machines – do not use SWI or other Prologs!

|P1 strategy |P2 strategy |Games |P1 |P2 |Draws |Av. |Av. Game| | | |Played|wins |wins | |Game |Time | | | | | | | |length | | |Random |Bloodlust | | | | | | | |Random |Self Preservation| | | | | | | |Random |Land Grab | | | | | | | |Random |Minimax | | | | | | | |Bloodlust |Random | | | | | | | |Bloodlust |Bloodlust | | | | | | | |Bloodlust |Self Preservation| | | | | | | |Bloodlust |Land Grab | | | | | | | |Bloodlust |Minimax | | | | | | | |Self Preservation|Random | | | | | | | |Self Preservation|Bloodlust | | | | | | | |Self Preservation|Self Preservation| | | | | | | |Self Preservation|Land Grab | | | | | | | |Self Preservation|Minimax | | | | | | | |Land Grab |Random | | | | | | | |Land Grab |Bloodlust | | | | | | | |Land Grab |Self Preservation| | | | | | | |Land Grab |Land Grab | | | | | | | |Land Grab |Minimax | | | | | | | |Minimax |Random | | | | | | | |Minimax |Bloodlust | | | | | | | |Minimax |Self Preservation| | | | | | | |Minimax |Land Grab | | | | | | | |Minimax |Minimax | | | | | | |
