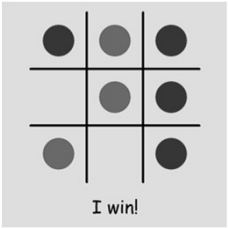
Unbeatable Noughts and Crosses – by Recursion

Recursive Analysis in Noughts and Crosses

Online UNIQ Summer School, July 2020

Professor Peter Millican
Faculties of Philosophy and Computer Science
Hertford College, Oxford



1

Finding the Program

- The program is in the "computer science" part of the *Turtle* Examples menu:



2

2

The *Evaluate* Procedure

- In the Pascal version of the program, the ***evaluate*** procedure runs from line 23 until line 61-64 (depending on how it's word-wrapped), and contains only 20 substantial instructions (here counting both an "if" and "then" as individual instructions).
  - *This procedure does the entire analysis of each noughts and crosses move*, analysing ahead from the current position and considering every relevant sequence of possibilities until the end of the game!
  - The only long instruction (at line 34) is conceptually very simple: it just checks every possible line of 3, to see whether the player who has just moved has filled a line.

3

3

Procedure Parameters

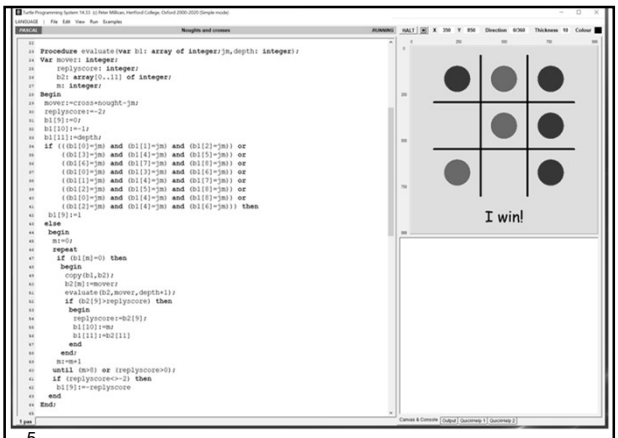
jm number representing player who has just moved: "cross" = 33 (royal blue); "nought" = 2 (red)

depth how deep is this position within the analysis tree?

b1 array of integers representing position to analyse:
b1[0], b1[1], b1[2], … b1[8] hold the contents of the nine cells of the board
b1[9] returns the value of the position (from the point of view of the player who just moved)
b1[10] returns the best move found so far
b1[11] holds the depth of the current analysis

4

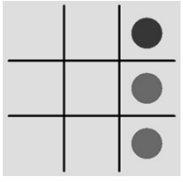
4



5

5

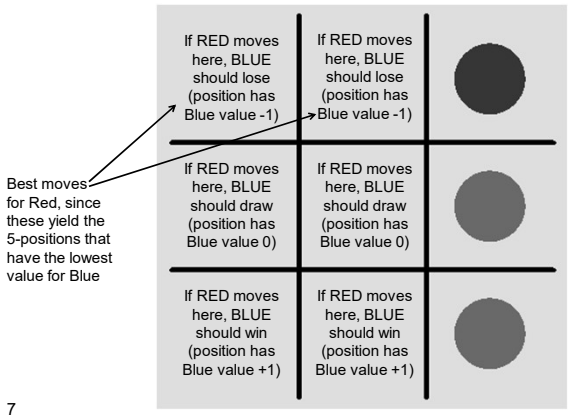
- Playing infallibly requires a reliable method for assessing the "value" of any position (+1=winning; 0=drawing; -1=losing).
- Consider this position, with Red to move:
  - It's a "6-space-position" (let's say "6-position" for short).
  - After Red makes her move, her opponent, Blue, will be faced with a 5-position in which to make his reply.
  - Suppose there is a method for calculating the value (to Blue) of any 5-position: how can Red use this *now* to work out her best move?



6

6

Unbeatable Noughts and Crosses – by Recursion



7

7

"Minimaxing"

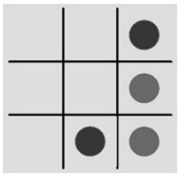
- Red winning is the same as Blue losing (and vice-versa).
  - The value of a position to Red is the inverse of its value to Blue (e.g. -1 as opposed to +1).
  - So Red tries to find moves which *maximise* the value to her, and *minimise* the value to Blue.
  - Red, faced with a 6-position, chooses the move which produces the 5-position which is *worst* (or equal-worst) for Blue.
- OK, but how does she work *that* out?

8

8

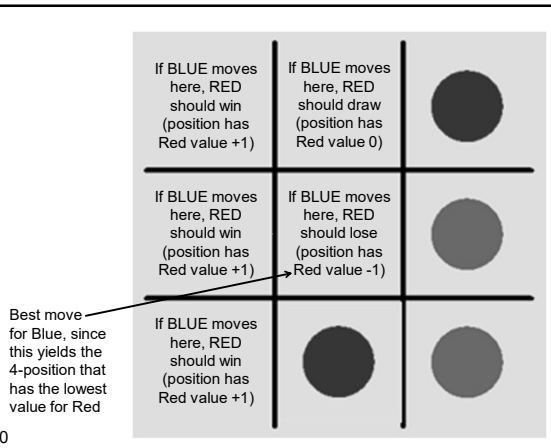
The same kind of problem, a move later …

- Suppose Red moved bottom centre. We would reach this position, with Blue to move:
  - It's a 5-position.
  - After Blue makes his move, his opponent, Red, will be faced with a 4-position in which to make her reply.
  - Suppose there is a method for calculating the value (to Red) of any 4-position: how can Blue use this *now* to work out his best move?



9

9



10

10

The Magic of Recursion

- We have seen how to evaluate a 6-position if we can evaluate a 5-position,
  - … and in the same way we can evaluate a 5-position if we can evaluate a 4-position,
  - … and a 4-position if we can evaluate a 3-position, and a 3-position if we can evaluate a 2-position, and a 2-position if we can evaluate a 1-position, and a 1-position if we can evaluate a 0-position!
- The only other thing we need is to be able to recognise a FINAL position as winning for Red, winning for Blue, or drawn.

11

11

- So to evaluate a position with X to move:
  - Has X already lost (because the opponent has a line of three)? If so, its X-value is -1.
  - Is the position full up (with all nine spaces filled)? If so, its X-value is 0.
  - If neither of these, then construct in turn all of the positions (with Y to move) that can arise from the current position (i.e. try out in turn all of X's possible moves in the current position) …
    - Evaluate each of these new positions *from Y's point of view* (resulting in evaluations Y_1, Y_2, and Y_3, say), and then assign – as X-value of the *current* position – the *inverse* of the *lowest* of Y_1, Y_2, and Y_3.
- Choose the move with the best X-evaluation.

12

12