Games

A Microcosm for Intelligence?

What are Games, Anyway?

- A fun diversion for otherwise busy people and machines?
- An intelligence test?
- Mathematical domain?
- Competitive metaphor for reproduction/survival?
- Other Metaphors for Real Life?

Like Intelligence, "Game" has no easy definition

- Every Criteria Has a Counterexample:
  - Mental/Physical
  - Deterministic/Stochastic
  - Knowledge intensive/Knowledge Free
  - Capitalistic/Socialistic

Like Intelligence, "Game" has no easy definition

- Games form an open category, with family resemblances rather than necessary and sufficient features.

Why did AI as a field Focus on Games?

- Human cognitive activity
- "Only Geniuses play chess"
- "Lot of money to be made in Las Vegas"
- Military Fascination with
  - Zero Sum Games
  - Mutually Assured Destruction
  - Differential Games
  - Simulation (Role Playing) Games

Techniques have tighter Focus on specific type of board game

- 2-Player (i.e. not solitaire puzzles)
- Deterministic, extensive form
- Mental/Mathematical
- Board & Piece
  - Chess, Checkers, Othello, Tictactoe, etc.
    - Why?

Why not AI for other kinds of games?

- Work on games with random/hidden factors better done with statistics.
  - Cards, Backgammon, Monopoly
    - spaces are larger and less certain
    - associated with human vice
    - strong methods based on statistics
  - Yes, there is a computer champion of HeadsUp Nolimit Texas Holdem...
- What about games of "Human Knowledge"?
  - Trivial Pursuit, Scrabble, Jeopardy???

Basic Technique

- Move Generator
- Evaluation Function

Move Generator

- Legal Move Generator
  - From current position, find all possible moves
- Plausible Move Generator
  - From current position, find only non-stupid moves
- PMG's help keep branching factor down
  - Fold some domain knowledge into tree expansion
    - Can be exploited by humans taking game to unexplored areas.

Tic Tac Toe LMG (version 1)

- Board is a 9 tuple of 0's 1's and 2's
  - blank(0) or player 1 or 2.
- Initial board is '(0 0 0 0 0 0 0 0 0)

- (defun legal-moves (player board)
- (loop for x in board as i from 1
- when (zerop x) collect i)))

all you really need is a play function and a random strategy

- with a LMG, you provide a player with the current board and request a legal move (if any)
  - pick a random move
- Then switch players, and do it again.
- Until the game is over.
  - what about forced and forfeited moves?

Play Function

```
(defun play (player1-function player2-function
&optional (playernumber 1) (board (init-board)))
 (or (winp board)
     (play player2-function player1-function
                          (opponent playernumber)
          (makemove board
                (funcall player1-function
                     playernumber board)
                playernumber))))
```

WINP

```
(defun winplayer (board player)
  (loop for test in *all-lines* thereis
        (loop for index in test always
              (eql player (nth (- index 1) board)))))

(defun winp (board)
  (cond ((winplayer board 1) 1)
        ((winplayer board 2) 2)
        ((loop for x in board thereis (zerop x)) nil)
        (t 0))) ;;all squares means DRAW
```

how to test for a win?

```
(defvar *all-lines* '((1 2 3)
                      (4 5 6)
                      (7 8 9)
                      (1 4 7)
                      (2 5 8)
                      (3 6 9)
                      (1 5 9)
                      (3 5 7)))
```

Makemove

```
(defun makemove (board position player)
  (let ((newboard (copy-tree board)))
       (setf (nth (- position 1) newboard)
             player)
       newboard))
```

Random Strategy

- of all the legal moves, pick one randomly.
- This provides a machine player for a game!
- Its not very good, however.

Random Strategy for TTT

- (defun pick (seq)
  - "Pick a random element out of a sequence."
  - (and seq (nth (random (length seq)) seq)))

- (defun random-strategy (player board)
  - "Make any legal move."
  - (pick (legal-moves player board)))

Now we want a HUMAN player function

- (defun human (player board)
  - (**printboard** board)
  - (print "your legal moves, no error checking")
  - (print (legal-moves player board))
  - (terpri)
  - (read))

Evaluation function

- Gives a "Score" to a game position
- Usually Symmetric (+ for me and - for you)
- Like in any search, the cost/accuracy tradeoff is still a concern
  - Perfect evaluations for trivial games only, like NIM
- **Because heuristic is inaccurate, we can use search to improve accuracy**

Game Heuristics?

- What are qualities of game, besides win/lose, which are predictive of win/lose?
  - piece count
  - piece value
  - position strength
  - mobility (less is usually worse)
  - what else?

Heuristic for TTT

```
(defun heur-value (player board)
 "Is this a win, loss, or draw for player?"
 (cond ((eql player (winp board)) winning-value)
                ((eql (- 3 player)(winp board)) losing-value)
                (t (+ (* 100 (two-marks player board))
                      (* -1000 (two-marks (opponent player)
                                          board))))))
(defun two-marks (player board)
  (loop for test in *all-lines* count
           (and (loop for index in test never
           (= (opponent player) (nth (- index 1) board)))
                (= 2 (loop for index in test count
                   (= player (nth (- index 1) board)))))))
```

Player Function Creator

```
(defun maximizer (eval-fn)
 #'(lambda (player board)
  (let* ((moves (legal-moves player board))
         (scores (loop for m in moves collect
                       (funcall eval-fn player
                                (makemove board m player))))
         (best (apply #'max scores)))
    (nth (index best scores) moves))))
```

- This "creates" a Player function from a heuristic function which can be a funarg to PLAY.

Multi-Ply Search

- A "Ply" is
  - a player's turn in the game
  - a layer in the search tree
- Search involves looking at
- Your response to
- My response to
- Your response to
- My contemplated move
  - At least 2 ply is useful because of ubiquity of "exchanges"

Minimax Search

- Make my maximum score move such that your maximum is minimized!
- Why is your best move Worst?
  - Because my response is best
- Infinite Regression?
  - For small games (TTT), and some end-games, can carry out to end or cache.
  - for real games, under finite time, bottoms out at *approximate heuristic evaluation function*

Knowledge Search Tradeoff

- One of the foundational principles of AI
  - The more you know, the less you have to search.
- In minimax game playing with a static evaluation function, the "knowledge" estimates how good a position is.
- If it was "perfect knowledge" it would be equivalent to full unrolling of the game tree
  - possible only for small games

Knowledge-Search Tradeoff

- Random Testing
- Brute-Force examination
- Hill Climbing
- Organized Search
- Heuristic Search
- Locally informed methods
- "Strong" methods
- Mathematical Solutions
- Insight and Intuition
- Magic

more knowledge
less search

Towards Better AI Players

- You can make the Tree Deeper
  - Takes exponential more CPU time
  - build special purpose hardware
- You can PRUNE better
  - Same CPU gets deeper
- You can improve the Eval Function
  - The first Machine Learning was Arthur Samuels Checker Player 1960ish
- TD Neural Reinforcement 1995
  - One-ply Backgammon Whiz
  - 3-ply Backgammon World Champ

Chess Machine History

- 1750's Baron of Kempelen's TURK
- 1950 Alan Turing does LMG
- 1950 Claude Shannon paper design
- 1957 Herb Simon Predicts 10 Years
- 1968 David Levy bets nothing to beat him in 10 years
- 1978 Levy wins bet
- 1982 Bell Lab's BELLE
- 1988 CMU Deep Thought
  - Levy Loses
- 1997 IBM Deep blue defeats Kasparov
- 2003 Kasparov Ties Blue Junior

Conclusions

- Deep Blue vs Kasparov event was profoundly disturbing to mankind
  - Cover of Time and Newsweek
  - Biggest internet "sport" site
  - commentary on rise of AI
- Was not news in AI field!!!
  - old simple techniques
  - raw machine power
    - Simple techniques plus enormous computer power can creates some kinds of intelligence in formal/mathematical situations

Next time...

- Searching a game tree, ply by ply
- Using the Minimax Principle
- Pruning with Alpha-Beta