More stuff on games

More stuff

- Caching states
- Canonical forms
- Game Theory

Complete Unrolling

- Sometimes games can be completely converted from extensive form to a table of moves and values.

Complete Unrolling of TTT

```
(defun getmoves (board)
  (Unless (OR (WINP1 BOARD)(WINP1 (VECINV BOARD)))
          (loop for i from 0 to 8 when (zerop (nth i board))
                collect (loop for x in board as j from 0
                              collect (if (eql i j) 1 x)))))

;switch -1 and 1
(defun vecinv (x)
  (sublis '((-1 . 1)(1 . -1)) x))

(defun score (board)
  (or (and (winp1 board) 1)
      (and (winp1 (vecinv board)) -1)
      0))

(defun winp1 (board)
  (loop for l in '((0 1 2)(3 4 5)(6 7 8)(0 3 6)(1 4 7)(2 5 8) (0 4 8)(2 4 6))
     thereis (= 3 (loop for i in l sum (nth i board)))))
```

explore full tree, hashing

```
(defun explore (board &aux substuff)
  (setq substuff (vecinv (getmoves board))) ;;;get next moves from minimizing
   (cond ((null substuff) (score board))
              (t (loop for bd in substuff maximize
                          (- (explore bd))))))

(memoize 'explore)
;;;note hash table is empty
;;this explores the entire space of positions and hashes them to 1 0 or -1 as the score
(explore (init-board))
(get 'explore 'memo)


(defun lookup (player board)
  (if (= -1 player) (gethash (list board) (get 'explore 'memo))
    (- (gethash (list (vecinv board)) (get 'explore 'memo)))))
```

Canonical Forms

- many games have symmetries
  - player a or player b
  - Rotation of board
  - Mirror image of board
- Storing board in a canonical form will lessen memory and computational requirements
  - Can the board be converted to an integer?
    - comparison becomes faster "="

the 8 equivalence permutations

```
(defun transpose (x)
  (if (null (car x)) nil (cons (mapcar #'car x)(transpose (mapcar #'cdr x)))))

(defun rotate (x) (reverse (transpose x)))

(defvar *allperms*
  (let ((p '((0 1 2)(3 4 5)(6 7 8))))
    (loop for perm in (list     p
                                (rotate p)
                                (rotate(rotate p))
                                (rotate(rotate(rotate p)))
                                (transpose p)
                                (rotate (transpose p))
                                (rotate(rotate (transpose p)))
                                (rotate(rotate(rotate(transpose p)))))
          collect (apply #'append perm))))
```

Canonical form for TTT

```
(defun canform (board)
  (num2board (loop for perm in *allperms*
                   minimize (board2num (shuffle board perm)))))

(defun shuffle (board perm)
  (loop for x in perm collect (nth x board)))

(defun board2num (board)
  (loop for x in board as i from 0 as j = 1 then (* 3 j) sum (* j (+ x 1))))

(defun num2board (num)
  (loop for i from 0 to 8 as j = num then (floor (/ j 3)) collect (1- (mod j
3))))
```

Caching

- Each time you do a search, you are re-calculating a lot of boards and heuristics
- Find a way to keep some of those around to avoid re-calculation

Memoization (Norvig P 275)

- Caches the results of a function in a hash table.

```
(defun memo (fn name)
  (let ((table (make-hash-table :TEST #'EQUAL)))
    (setf (get name 'memo) table)
    #'(lambda (x)
        (multiple-value-bind (val found-p)
            (gethash x table)
          (if found-p
              val
              (setf (gethash x table)(funcall fn x)))))))

(defun memoize (fn-name)
  (setf (symbol-function fn-name)(memo (symbol-function fn-name) fn-name)))
```

Canonical explore

```
(defun cangetmoves (board)
  (Unless (OR (WINP1 BOARD)(WINP1 (VECINV BOARD)))
          (remove-duplicates
           (loop for i from 0 to 8 when (zerop (nth i board))
                 collect (canform (loop for x in board as j from 0
                                        collect (if (eql i j) 1 x))))
           :test #'equal)))

(defun canexplore (board &aux substuff)
  (setq substuff (vecinv (cangetmoves board)));;get next boards
  (cond ((null substuff) (score board));;no moves, return score
        (t (loop for bd in substuff maximize;;return max of -opponent
                 (- (canexplore bd))))))

(memoize 'canexplore) ;;;use hash table
(canexplore 1 (init-board)) ;;fill up hashtable
```

With Canonical forms and memoization of EXPLORE…

- **Only 765 boards of TTT**

What about Game Theory?

- AI has not focused on "Game Theory"
  - Why? Minsky was made sick of it at Princeton in 1950's!
- Mathematical Model of Decisionmaking
  - under adversary
  - with cooperation
- Zermelo, 1923
- Von Neumann, 1934
- Nash 1950 (Nash Equilibria)
- Field of Mathematical Economics.

What is a two-player game in normal form?

- Each player chooses a move and receives a payoff based on others decisions

zero Sum	Win	Lose
Win	0 0	1 -1
Lose	-1 1	0 0

In a zero sum game, the sum of payoffs to all players, is zero.

What is a two-player game in normal form?

- **Each player simultaneously chooses a move (e.g. row or column) and receives a payoff based on other's choices**

Concepts: pure strategy vs mixed strategy

Concept: Nash Equilbrium

Strategies

- **pure strategy**
  - **Player always chooses the same row or col**
- **Mixed Strategy**
  - **player uses a probability distribution of choices.**

pure vs mixed

- Pure and Mixed Strategy
- A pure strategy is each of the definite courses of action that a player can choose.
- A mixed strategy is a strategy in which the course of action is randomly chosen from one of the pure strategies in the following way:
- Each pure strategy is assigned some probability, indicating the relative frequency with which that pure strategy will be played.
- The specific strategy used in any given play of the game can be selected using some appropriate random device.

- Expected Value E
- If each of the n payoffs, s1, s2, …, sn will occur with the probabilities p1, p2, …, pn, respectively, then the average, or expected value E, is given by:
- $E = p1\ s1 + p2\ s2 + \ldots + pn\ sn$

Nash Equilibrium

- John Nash taught at Brandeis for a few years.
- Won Nobel Prize in Economics in 1980
- Got a movie made "Brilliant Mind"

- Famous Proof:
  - For all zero sum games of n players, there exists an equilibrium of mixed strategies.

Implications of Nash Equilibria?

- Laissez Faire economics does not work but leads to oligarchy or monopoly
- Our understanding of evolution as dog-eat-dog is sadly mistaken
- Markets need central government oversight to prevent monopoly rent-seeking (e.g. microsoft, google, adobe, facebook)

Value of Game Theory

- Extensive Literature
  - Game of Chicken
  - International relations
    - MAD Nuclear Strategy
  - evolution of cooperation
    - positive sum games
    - Prisoner's dilemma's
  - Evolutionary game theory (Maynard Smith)
    - Hawks vs Doves
  - Markets, bidding games
  - Organization and Team theory

Prisoner Dilemma

- Most games we think of are "zero-sum"
- Cooperative and Altruistic Games can be Positive Sum

prisoner dilemma	cooperate	defect
cooperate	3 3	0 5
defect	5 0	1 1

- Iterated prisoners dilemma is basis for many studies and theories in artificial life

Axelrod's Tournaments

- Robert Axelrod of UMich collected small computer programs from experts to play IPD in tournaments.
  - Its a dilemma because the Nash equilbrium is to always defect back propagated from the finite number of iterations
    - Each player gets 1 point per iteration.
    - However if they cooperate, they each get 3 points per iteration

basic strategies

```
(defun tft (histyou histme)
  (if histyou (car histyou) 'c))

(defun alld (histyou histme) 'd)
(defun allc (histyou histme) 'c)
(defun majority (histyou histme)
  (if histyou (if (>= (loop for x in histyou count (eq 'c x))
                      (/ (length histyou) 2)) 'c 'd)
   'c))
```

Tit for Tat

- **submitted by Anatole Rapaport**
  - **First Cooperate**
  - **then play what your opponent played last move.**
- **A very simple simple strategy with minimal memory**
- **Won the tournament**

more strategies

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

(defun mixed (histyou histme) (pick (list 'c 'd)))

(defun pavlov (histyou histme)
  (if histyou (if (> (table (car histme)(car histyou)) 2)
                  (car histme)
                (if (eq (car histme) 'c) 'd 'c)) 'c))
```

play 2 strategies

```
(defun play (p1  p2 &optional (n 100) &aux temp)
  (let ((hist1 nil)(hist2 nil))
    (loop for i from 1 to n do
      (setf temp (funcall p1 hist2 hist1))
      (push (funcall p2 hist1 hist2) hist2)
      (push temp hist1))
    (list (score hist1 hist2)(score hist2 hist1))))
```

scoring the game

- (defun score (hist1 hist2) ;;score for player 1
- (loop for h1 in hist1 as h2 in hist2 sum
- (table h1 h2)))

- (defun table (h1 h2)
- (cond ((and(eq h1 'c)(eq h2 'c)) 3)
- ((and(eq h1 'd)(eq h2 'd)) 1)
- ((and(eq h1 'c)(eq h2 'd)) 0)
- ((and(eq h1 'd)(eq h2 'c)) 5)))

round robin tournament

- (defun tourn (strategies)
- (loop for strat1 in strategies collect
- (loop for strat2 in strategies sum (car (play strat1 strat2)))))

(tourn '(tft allc alld majority pavlov mixed))

- ; (1545 1350 1408 1536 1512 1422)