Lecture 8 of 42

CSP Search Concluded: Arc Consistency (AC-3) Intro to Games and Game Tree Search

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:
Sections 6.4 – 6.8, p. 171 – 185, Russell & Norvig 2nd edition
Outside references:
CSP examples, M. Hauskrecht (U. Pittsburgh) – http://tr.im/zdG6
Notes on CSP, R. Barták (Charles U., Prague) – http://tr.im/zdGE

Lecture Outline

- Reading for Next Class: 6.4 – 6.8 (p. 171 – 185), R&N 2e
- Last Class: Sections 5.1 – 5.3 on Constraint Satisfaction Problems
  - CSPs: definition, examples
  - Heuristics for variable selection, value selection
  - Two algorithms: backtracking search, "one-step" forward checking
- Today: Rest of CSP, 5.4-5.5, p. 151-158; Games Intro, 6.1-6.3, p. 161-174
  - Third algorithm: constraint propagation by arc consistency (AC-3)
  - Scaling up to NP-hard problems
- This Week: CSP and Game Tree Search
  - Rudiments of game theory
  - Zero-sum games *vs.* cooperative games
  - Perfect information *vs.* imperfect information
  - Minimax
  - Alpha-beta (α - β) pruning
  - Randomness and expectiminimax
- Next : From Heuristics to General Knowledge Representation

Acknowledgements

Stuart J. Russell
Professor of Computer Science
Chair, Department of Electrical Engineering and Computer Sciences
Smith-Zadeh Prof. in Engineering

Berkeley
UNIVERSITY OF CALIFORNIA

Russell, S. J.
University of California, Berkeley
http://www.eecs.berkeley.edu/~russell/

Peter Norvig
Director of Research
Google

Norvig, P.
http://norvig.com/

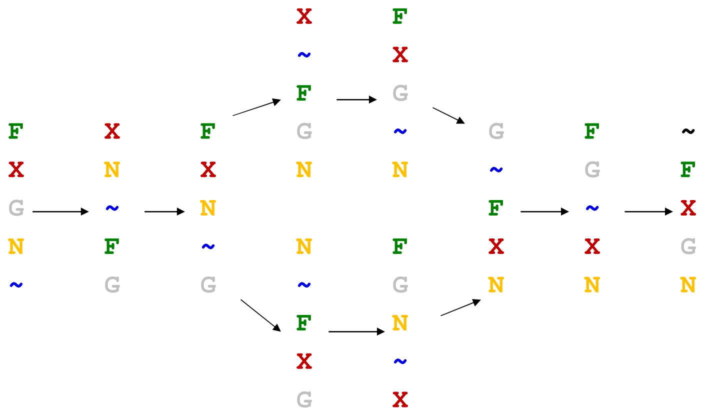
Slides from:
http://aima.eecs.berkeley.edu

Milos Hauskrecht
Associate Professor of Computer Science
University of Pittsburgh

University of Pittsburgh
CS 2710
Foundations of Artificial Intelligence
http://www.cs.pitt.edu/~milos

Farmer, Fox, Goose, & Grain State Space: Review

F = Farmer X = foX G = Goose
N = graiN ~ = River



Adapted from slide © 2008 B. R. Maxim, Univ. of Michigan – Dearborn
CIS 479/579 Artificial Intelligence http://tr.im/zdhV

CSPs: Review

Standard search problem:

state is a "black box"—any old data structure that supports goal test, eval, successor

CSP:

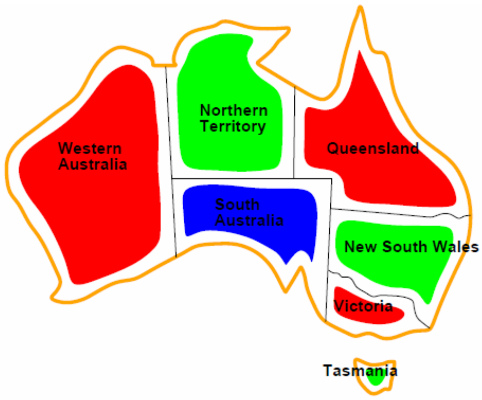
state is defined by variables X_i with values from domain D_i

goal test is a set of constraints specifying allowable combinations of values for subsets of variables

Simple example of a formal representation language

Allows useful general-purpose algorithms with more power than standard search algorithms

Map Coloring Example: Review



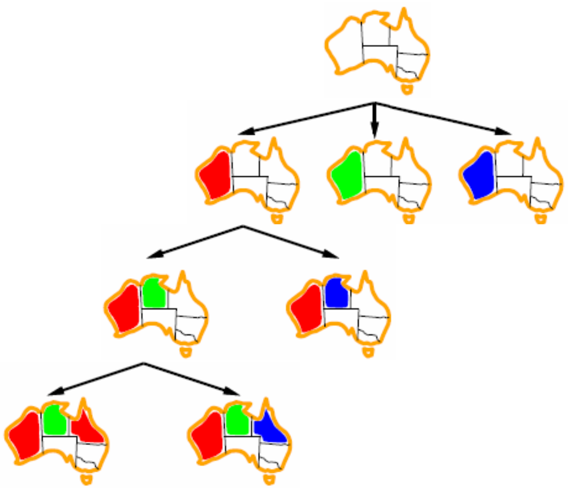
Solutions are assignments satisfying all constraints, e.g.,
$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

Algorithm 1 – Backtracking Search: Review

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
   return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
   if assignment is complete then return assignment
   var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
   for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment given CONSTRAINTS[csp] then
         add {var = value} to assignment
         result ← RECURSIVE-BACKTRACKING(assignment, csp)
         if result ≠ failure then return result
         remove {var = value} from assignment
   return failure
```


Backtracking Example: Review

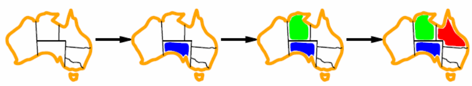


Variable and Value Selection: Review

Minimum remaining values (MRV):
choose the variable with the fewest legal values

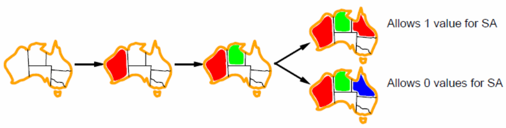
Tie-breaker among MRV variables

Degree heuristic:
choose the variable with the most constraints on remaining variables



MRV
with
degree heuristic:
variable selection

Given a variable, choose the least constraining value:
the one that rules out the fewest values in the remaining variables



LCV
value selection
(for a given variable)

Combining these heuristics makes 1000 queens feasible

Based on slides © 2004 S. Russell & P. Norvig. Reused with permission.

Value Propagation: Constraint Prop Without Lookahead

- **Constraint propagation**

Value propagation. Infers:

– **equations from** the set of **equations** defining the partial assignment, **and a constraint**

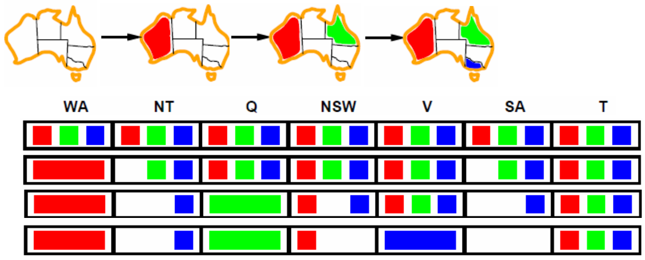
No equations/disequations are inferred

No equations/disequations are inferred

© 2005 M. Hauskrecht, University of Pittsburgh
CS 2710 Foundations of Artificial Intelligence
http://www.cs.pitt.edu/~milos/courses/cs2710/

Algorithm 2 – Forward Checking: Review

Idea: Keep track of remaining legal values for unassigned variables
Terminate search when any variable has no legal values



NT and SA cannot both be blue!

Constraint propagation repeatedly enforces constraints locally

Forward Checking With "One-Step" Constraint Prop

- **Constraint propagation**

Forward checking. Infers:

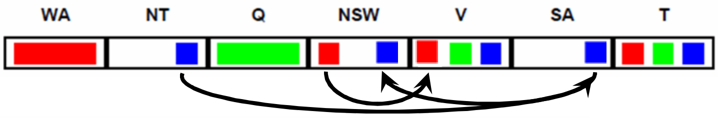
- **disequations from** a set of **equations** defining the partial assignment, and a constraint
- **Equations through the exhaustion of alternatives**

Invalid assignment

Algorithm 3 – Arc Consistency [1]

Simplest form of propagation makes each arc consistent

$X \to Y$ is consistent iff
for **every** value x of X there is **some** allowed y



If X loses a value, neighbors of X need to be rechecked

Arc consistency detects failure earlier than forward checking

Can be run as a preprocessor or after each assignment

Algorithm 3 – Arc Consistency [2]
AC-3 Definition

```
function AC-3( csp) returns the CSP, possibly with reduced domains
   inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
   local variables: queue, a queue of arcs, initially all the arcs in csp

   while queue is not empty do
      (Xi, Xj) ← REMOVE-FIRST(queue)
      if REMOVE-INCONSISTENT-VALUES(Xi, Xj) then
         for each Xk in NEIGHBORS[Xi] do
            add (Xk, Xi) to queue
```

```
function REMOVE-INCONSISTENT-VALUES( Xi, Xj) returns true iff succeeds
   removed ← false
   for each x in DOMAIN[Xi] do
      if no value y in DOMAIN[Xj] allows (x,y) to satisfy the constraint Xi ↔ Xj
         then delete x from DOMAIN[Xi]; removed ← true
   return removed
```

$O(n^2 d^3)$, can be reduced to $O(n^2 d^2)$ (but detecting **all** is NP-hard)

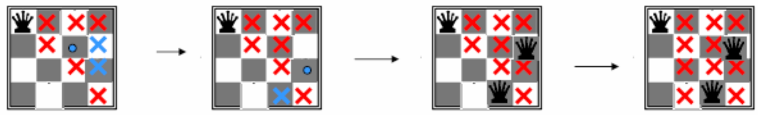
Forward Checking
With Full Arc Consistency

- **Constraint propagation**

Arc consistency. **Infers:**

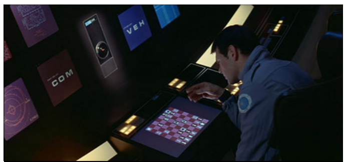
- **disequations from** the set of **equations and disequations** defining the partial assignment, and **a constraint**
- **equations through the exhaustion of alternatives**

After forward checking

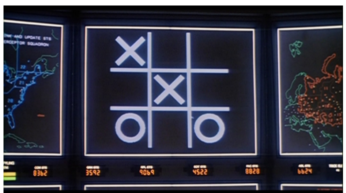


Intro to Games:
Outline

- Games
- Perfect play
  - minimax decisions
  - α–β pruning
- Resource limits and approximate evaluation
- Games of chance
- Games of imperfect information



2001: A Space Odyssey © 1968 MGM.
http://tr.im/zdyp



WarGames © 1983 MGM.
http://en.wikipedia.org/wiki/WarGames

Games *versus* Search

"Unpredictable" opponent ⇒ solution is a strategy
specifying a move for every possible opponent reply

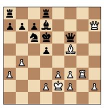
Time limits ⇒ unlikely to find goal, must approximate

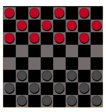
Plan of attack:

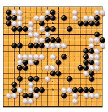
- Computer considers possible lines of play (Babbage, 1846)
- Algorithm for perfect play (Zermelo, 1912; Von Neumann, 1944)
- Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- First chess program (Turing, 1951)
- Machine learning to improve evaluation accuracy (Samuel, 1952–57)
- Pruning to allow deeper search (McCarthy, 1956)

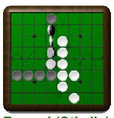
Types of Games

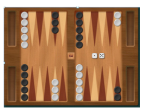
	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleship, blind tictactoe	bridge, poker, scrabble nuclear war


Chess http://tr.im/zdTD


Checkers http://tr.im/zdTW

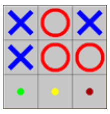

Go http://tr.im/zdVn

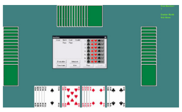

Reversi (Othello) http://tr.im/zdVr


Backgammon http://tr.im/ze1P

Monopoly © Parker Brothers http://tr.im/ze2F


Battleship © Milton Bradley http://tr.im/zdWK

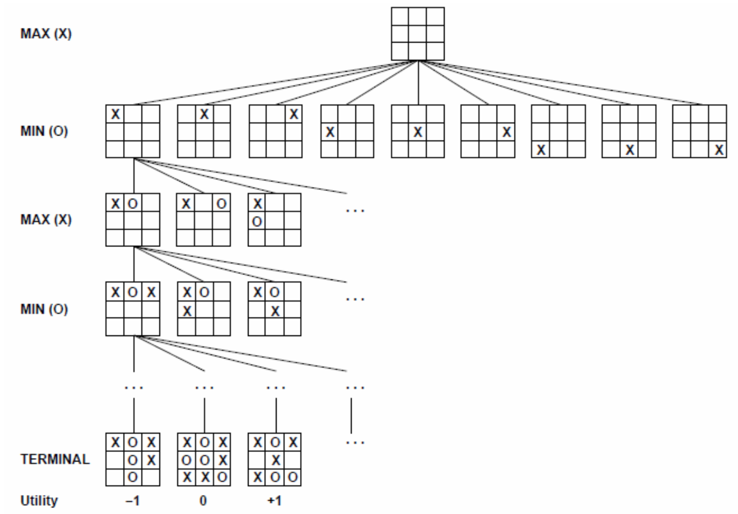

Tic-Tac-Toe http://tr.im/zdXB


Contract Bridge http://tr.im/ze5D


Poker (Texas Hold 'Em) http://tr.im/ze7W


Scrabble © Hasbro http://tr.im/ze90

Game Tree: 2-Player, Deterministic, Turns

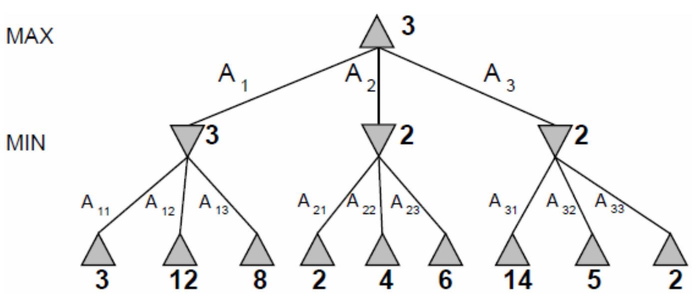


Minimax [1]: Example

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest minimax value
= best achievable payoff against best play

E.g., 2-ply game:



MINIMAX [2]: ALGORITHM

```
function MINIMAX-DECISION(state) returns an action
   inputs: state, current state in game

   return the a in ACTIONS(state) maximizing MIN-VALUE(RESULT(a, state))
```

```
function MAX-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for a, s in SUCCESSORS(state) do v ← MAX(v, MIN-VALUE(s))
   return v
```

```
function MIN-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← ∞
   for a, s in SUCCESSORS(state) do v ← MIN(v, MAX-VALUE(s))
   return v
```


MINIMAX [3]: PROPERTIES

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
$\Rightarrow$ exact solution completely infeasible

But do we need to explore every path?

ALPHA-BETA (α - β) PRUNING [1]: EXAMPLE

What are α, β values here?

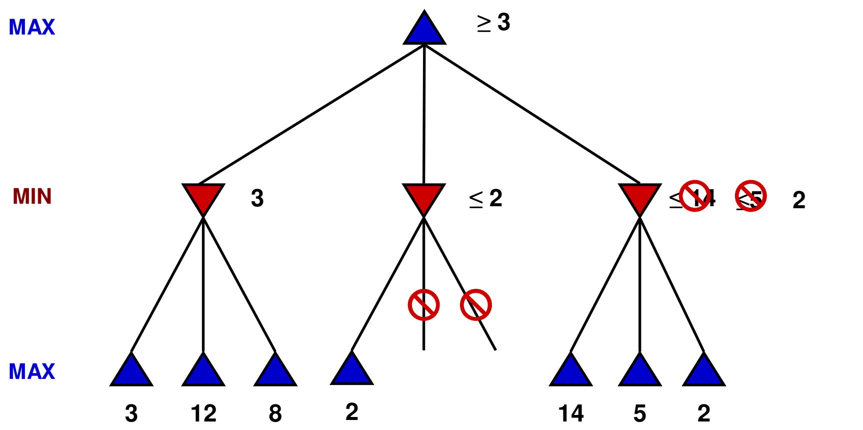


Figure 6.5 p. 168 R&N 2e

ALPHA-BETA (α - β) PRUNING [2]: ALGORITHM

function ALPHA-BETA-DECISION(*state*) **returns** an action
  return the a in ACTIONS(*state*) maximizing MIN-VALUE(RESULT(a, *state*))

---

function MAX-VALUE(*state*, α, β) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE(*state*, α, β) **returns** *a utility value*
  same as MAX-VALUE but with roles of α, β reversed

Alpha-Beta (α - β) Pruning [3]: Properties

Pruning does not affect final result

Good move ordering improves effectiveness of pruning

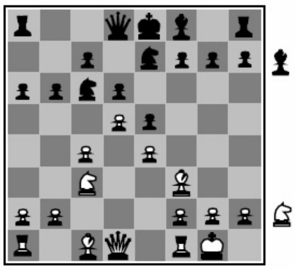
With "perfect ordering," time complexity $= O(b^{m/2})$
$\Rightarrow$ doubles solvable depth

A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

Unfortunately, 35^{50} is still impossible!

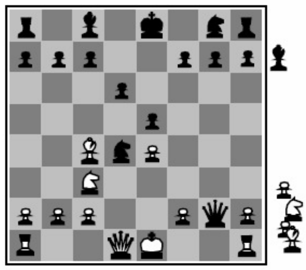
- **Can We Do Better?**
- **Idea: Adapt Resource-Bounded Heuristic Search Techniques**
  - **Depth-limited**
  - **Iterative deepening**
  - **Memory-bounded**

Static Evaluation Functions



Black to move

White slightly better



White to move

Black winning

For chess, typically linear weighted sum of features

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

e.g., $w_1 = 9$ with
$f_1(s) =$ (number of white queens) – (number of black queens), etc.

Terminology

- CSP Techniques
  - Variable selection heuristic: Minimum Remaining Values (MRV)
  - Value selection heuristic: Least Constraining Value (LCV)
  - Constraint satisfaction search algorithms: using variable and value selection
- Detailed CSP Example: 3-Coloring of Planar Graph
- Algorithms
  - Value propagation and backtracking
  - Forward checking: simple constraint propagation, arc consistency (AC-3)
- Games and Game Theory
  - Single-player vs. multi-player vs. two-player
  - Cooperative vs. competitive (esp. zero sum)
  - Uncertainty
    - Imperfect information vs. perfect information
    - Deterministic vs. games with element of chance
- Game Tree Search
  - Minimax, alpha-beta (α - β) pruning
  - Static evaluation functions

Summary Points

- CSP Techniques: Variable Selection, Value Selection, CSP Search
  - Last time: variable and value selection heuristics
  - CSP search algorithms: using heuristics systematically to find solution
- First Algorithm: Backtracking Search with Heuristics (MRV, LCV)
  - MRV for variable selection, LCV for value selection
  - Hard problems (e.g., n-queens) with n = 1000 possible
- Second and Third Algorithms: Forward Checking, Constraint Prop
  - Plain FC: "One-step" lookahead
  - Arc consistency (AC-3): "Multi-step" lookahead
- Detailed CSP Example: 3-Coloring Australian Map
- Intro to Game Theory: Emphasis on Game Tree Search
  - From graph search and CSP search to game tree search
  - Game tree representation
  - Perfect play: Minimax algorithm, speedup with alpha-beta (α - β) pruning
  - Resource-bounded Minimax: static evaluation functions, iterative deepening
  - Emphasis: two-player (with exceptions), zero-sum, perfect info
- Next: Conclusion to Section 2, R&N 2e (Search)