General Game Playing

Game Playing

Michael Genesereth
Computer Science Department
Stanford University
<table>
<thead>
<tr>
<th>Game</th>
<th>Description</th>
<th>Rulesheet</th>
<th>Stylesheet</th>
<th>Standalone</th>
<th>Human</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>alquerque</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badconnectfour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badhex</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badskirmish</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badtictactoe</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>battleofnumbers</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>bestbuttonsandlights</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>breakthrough</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>buttonsandlights</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>capturetheflag</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>cbnk</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>chinesecheckers3</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>connectfour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>crusade</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>cryptarithmetic</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>donttouch</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>doubleconnectfour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hamilton</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hex</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hex7x7</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hunter</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>jointbuttonsandlights</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>knightstour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>knightthrough</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>kono</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>leafymcfluffface</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>lightboard</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>majorities</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
</tbody>
</table>
Gamemaster

Control: x

Move: [Input Field]
Gamemaster

About    Resources    Software
Games    Players    Metagamers
Matches    Run    Leaderboard
<table>
<thead>
<tr>
<th>Player</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>egghead</td>
<td>Launch</td>
</tr>
<tr>
<td>greedy</td>
<td>Launch</td>
</tr>
<tr>
<td>lady</td>
<td>Launch</td>
</tr>
<tr>
<td>lara</td>
<td>Launch</td>
</tr>
<tr>
<td>legal</td>
<td>Launch</td>
</tr>
<tr>
<td>mcs</td>
<td>Launch</td>
</tr>
<tr>
<td>minimax</td>
<td>Launch</td>
</tr>
<tr>
<td>minimaxdepth</td>
<td>Launch</td>
</tr>
<tr>
<td>minimaxid</td>
<td>Launch</td>
</tr>
<tr>
<td>onestep</td>
<td>Launch</td>
</tr>
<tr>
<td>random</td>
<td>Launch</td>
</tr>
<tr>
<td>slowpoke</td>
<td>Launch</td>
</tr>
<tr>
<td>twostep</td>
<td>Launch</td>
</tr>
</tbody>
</table>
Gamemaster

Protocol: autoplayer
Identifier: legal 🌿
Gamemaster

Protocol: autoplayer
Identifier: egghead 🍳

Clear  Connect  Disconnect
<table>
<thead>
<tr>
<th>Game</th>
<th>Description</th>
<th>Rulesheet</th>
<th>Stylesheet</th>
<th>Standalone</th>
<th>Human</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>alquerque</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badconnectfour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badhex</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badskirmish</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>badtactaco</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>battleofnumbers</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>bestbuttonsandlights</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>breakthrough</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>buttonsandlights</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>capturetheflag</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>cbnk</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>chinesecheckers3</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>connectfour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>crusade</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>cryptarithmetic</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>donttouch</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>doubleconnectfour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hamilton</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hex</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hex7x7</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>hunter</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>jointbuttonsandlights</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>knightstour</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>knightthrough</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>kono</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>leafyfcafallface</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>lightboard</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
<tr>
<td>majorities</td>
<td>Description</td>
<td>Rulesheet</td>
<td>Stylesheet</td>
<td>Standalone</td>
<td>Human</td>
<td>Manager</td>
</tr>
</tbody>
</table>
Game over

Roles | x | o
--- | --- | ---
Players | indy | lara
Score | 100 | 0
Errors | 0 | 0
Programme

Game Management

Game Players

Complete Search Techniques for Single Player Games
Complete Search Techniques for Multiple Player Games
Game Management
Automatic Game Playing

Gamemaster

Manager

Player  ...  Player

Server

Browser
Communication by message passing between manager and players.
Manager writes messages into browser's local storage.
Players receive messages, process, write replies into local storage.
Manager requests:

- ping(n)
- start(n, role, ruleset(r₁, ..., rₖ), startclock, playclock)
- play(n, a)
- stop(n, a)
- abort(n)

Player responses:

- reply(n, value)
A *ping* message is used to determine if a player is running, communicating, and ready to play a match.

**General Form:**

```
ping(messageid)
```

**Replies:**

- `reply(messageid, ready)` - ready to play
- `reply(messageid, busy)` - not ready

No reply after 5 seconds interpreted as not ready
The start Message

A start message initiates a match.

General Form:
\[
\text{start}(\text{messageid}, \text{role}, \text{ruleset}(r_1, \ldots, r_k), \text{start}, \text{play})
\]

Reply:
\[
\text{reply}(\text{messageid}, \text{ready}) - \text{ready to begin play}
\]

NB: Match begins as soon as all players have replied or when startclock seconds have elapsed whichever comes first.
A play message is an update and a request for an action.

General Form:

\[
\text{play}(\text{msgid}, \text{nil})
\]

\[
\text{play}(\text{msgid}, \text{action})
\]

NB: \text{nil} / [] is argument on first step, \text{action} thereafter, where \text{action} is the action performed on preceding step.

Reply:

\[
\text{reply}(\text{msgid}, \text{action})
\]

NB: No need to reply if not in control.

NB: If player in control returns an illegal action or playclock has elapsed, the Manager substitutes a random legal action.
A stop message is used to inform players that the match has terminated successfully.

General Form:

\[
\text{stop}(\text{messageid}, \text{action})
\]

Reply:

optional
An abort message is used to tell players that a match has terminated abnormally.

General Form:
\texttt{abort(messageid)}

Reply:
\texttt{optional}
Match Management Procedure

Begins when manager receives a request to run a match of a given game with given players and given start clock and play clock.

(1) optionally sends ping messages to players
(2) sends start message with appropriate parameters
(3) sends play messages to all players
(4) sends stop message when game terminates
(5) sends abort for abnormal termination
message(1, manager, indy, ping())
message(1, manager, lara, ping())
message(1, indy, manager, ready)
message(1, lara, manager, ready)
message(2,manager,indy,start(m01,x,rule...,10,10))
message(2,manager,lara,start(m01,o,rule...,10,10))
message(2,indy,manager,ready)
message(2,lara,manager,ready)

Tic-Tac-Toe Example
message(2, manager, indy, start(m01, x, ruleset(...), 10, 10))
message(2, manager, lara, start(m01, x, ruleset(...), 10, 10))
message(2, indy, manager, ready)
message(2, lara, manager, ready)

message(3, manager, indy, play(m01, []))
message(3, manager, lara, play(m01, []))
message(3, indy, manager, mark(1, 1))
message(2, manager, indy, start(m01, x, ruleset(...), 10, 10))
message(2, manager, lara, start(m01, x, ruleset(...), 10, 10))
message(2, indy, manager, ready)
message(2, lara, manager, ready)

message(3, manager, indy, play(m01, []))
message(3, manager, lara, play(m01, []))
message(3, indy, manager, mark(1, 1))

message(4, manager, indy, play(m01, mark(1, 1)))
message(4, manager, lara, play(m01, mark(1, 1)))
message(4, lara, manager, mark(1, 2))
message(2,manager,indy,start(m01,x,ruleSet(...),10,10))
message(2,manager,lara,start(m01,x,ruleSet(...),10,10))
message(2,indy,manager,ready)
message(2,lara,manager,ready)

message(3,manager,indy,play(m01,[]))
message(3,manager,lara,play(m01,[]))
message(3,indy,manager,mark(1,1))

message(4,manager,indy,play(m01,mark(1,1)))
message(4,manager,lara,play(m01,mark(1,1)))
message(4,lara,manager,mark(1,2))

message(5,manager,indy,play(m01,mark(1,2)))
message(5,manager,lara,play(m01,mark(1,2)))
message(5,indy,manager,mark(2,2))
message(2,manager,indy,start(m01,x,ruleset(...),10,10))
message(2,manager,lara,start(m01,x,ruleset(...),10,10))
message(2,indy,manager,ready)
message(2,lara,manager,ready)

message(3,manager,indy,play(m01,[]))
message(3,manager,lara,play(m01,[]))
message(3,indy,manager,mark(1,1))

message(4,manager,indy,play(m01,mark(1,1)))
message(4,manager,lara,play(m01,mark(1,1)))
message(4,lara,manager,mark(1,2))

message(5,manager,indy,play(m01,mark(1,2)))
message(5,manager,lara,play(m01,mark(1,2)))
message(5,indy,manager,mark(2,2))

message(6,manager,indy,play(m01,mark(2,2)))
message(6,manager,lara,play(m01,mark(2,2)))
message(6,lara,manager,mark(1,3))
message(2, manager, indy, start(m01, x, ruleset(...), 10, 10))
message(2, manager, lara, start(m01, x, ruleset(...), 10, 10))
message(2, indy, manager, ready)
message(2, lara, manager, ready)

message(3, manager, indy, play(m01, []))
message(3, manager, lara, play(m01, []))
message(3, indy, manager, mark(1,1))

message(4, manager, indy, play(m01, mark(1,1)))
message(4, manager, lara, play(m01, mark(1,1)))
message(4, lara, manager, mark(1,2))

message(5, manager, indy, play(m01, mark(1,2)))
message(5, manager, lara, play(m01, mark(1,2)))
message(5, indy, manager, mark(2,2))

message(6, manager, indy, play(m01, mark(2,2)))
message(6, manager, lara, play(m01, mark(2,2)))
message(6, lara, manager, mark(1,3))

message(7, manager, indy, play(m01, mark(1,3)))
message(7, manager, lara, play(m01, mark(1,3)))
message(7, indy, manager, mark(3,3))
message(2, manager, indy, start(m01, x, ruleset(...), 10, 10))
message(2, manager, lara, start(m01, x, ruleset(...), 10, 10))
message(2, indy, manager, ready)
message(2, lara, manager, ready)

message(3, manager, indy, play(m01, [[]]))
message(3, manager, lara, play(m01, [[]]))
message(3, indy, manager, mark(1, 1))

message(4, manager, indy, play(m01, mark(1, 1)))
message(4, manager, lara, play(m01, mark(1, 1)))
message(4, lara, manager, mark(1, 2))

message(5, manager, indy, play(m01, mark(1, 2)))
message(5, manager, lara, play(m01, mark(1, 2)))
message(5, indy, manager, mark(2, 2))

message(6, manager, indy, play(m01, mark(2, 2)))
message(6, manager, lara, play(m01, mark(2, 2)))
message(6, lara, manager, mark(1, 3))

message(7, manager, indy, play(m01, mark(1, 3)))
message(7, manager, lara, play(m01, mark(1, 3)))
message(7, indy, manager, mark(3, 3))

message(8, manager, indy, stop(m01, mark(1, 3)))
message(8, manager, lara, stop(m01, mark(1, 3)))
message(9, manager, indy, abort(m01))
message(9, manager, lara, abort(m01))
Game Players
Each player is implemented as an active web page.

Listen Loop
- Receives messages from Game Manager
- Calls appropriate event handler
- Sends result back to Game Manager

Event handlers
- each handles a different type of message
- responds with time bounds

*We provide listen loop, you provide event handlers.*
function ping() - returns ready or busy
   {...code that calls predefined subroutines...}

function start(role, rules, start, play) ready
   {...code that calls predefined subroutines...}

function play(move) - returns action or false
   {...code that calls predefined subroutines...}

function stop(move) - optional value or false
   {...code that calls predefined subroutines...}

function abort() - optional value or false
   {...code that calls predefined subroutines...}
Subroutines defined using epilog.js

`findroles(rules)` - array of roles
`findbases(rules)` - array of factoids
`findactions(rules)` - array of actions
`findinits(rules)` - array of factoids (state)

`findcontrol(state, rules)` - role
`findlegalp(action, state, rules)` - boolean
`findlegalx(state, rules)` - action
`findlegals(state, rules)` - array of actions
`findreward(role, state, rules)` - number
`findterminalp(state, rules)` - boolean

`simulate(action, state, rules)` - array of factoids
Simple behavior
   Maintains current state of the game.
   On each step, selects first legal action it finds.

NB: Selects same action every time in same state.
var role, roles, state, library, startclock, playclock;

function ping ()
    {return 'ready'}

function start (r,rs,sc,pc)
    {role = r;
     library = rs;   // definerules([],rs.slice(1));
     roles = findroles(library);
     state = findinits(library);
     startclock = numberize(sc);
     playclock = numberize(pc);
     return 'ready'}

function play (move)
    {if (move!==nil) {state = simulate(move,state,library)};
     if (findcontrol(state,library)!==role) {return false};
     return findlegalx(state,library)}

function stop (move)
    {return false}

function abort ()
    {return false}
Simple behavior
   Maintains current state of the game.
   On each step, selects random legal action.

NB: May take different action each time in a state.
Implementation

```javascript
var role, roles, state, library, startclock, playclock;

function ping ()
    {return 'ready'}

function start (r,rs,sc,pc)
    {role = r;
     library = rs;   // definerules([],rs.slice(1));
     roles = findroles(library);
     state = findinits(library);
     startclock = numberize(sc);
     playclock = numberize(pc);
     return 'ready'}

function play (move)
    {if (move!==nil) {state = simulate(move,state,library)};
     if (findcontrol(state,library)!==role) {return false};
     var actions=findlegals(state,library);
     return actions[randomindex(actions.length)]}

function stop (move)
    {return false}

function abort ()
    {return false}
```
Legal Player:

```javascript
function play (move)
    {if (move!==nil) {state = simulate(move,state,library)};
        if (findcontrol(state,library)!==role) {return false};
        return findlegalx(state,library)}
```

Random Player:

```javascript
function play (move)
    {if (move!==nil) {state = simulate(move,state,library)};
        if (findcontrol(state,library)!==role) {return false};
        var actions=findlegals(state,library);
        return actions[randomindex(actions.length)]}
```
Random Players are no “smarter” than legal players. Appear more interesting because unpredictable. Sometimes avoid traps that befall consistent players like Legal. Often used as a comparison to show that a player or method performs statistically better than chance.

Random players take *slightly* more time than legal Compute all legal actions rather than just one Noticeable only on games with many legal actions
Maximizer
Single Player Games

Example:

Terminology:
- Single Player Games = Puzzles
- Playing Single Player Games = Problem Solving

Easier than multiple player games
- World static (except when single player acts)
- Changes determined entirely by player’s actions
Game Description tells player:
  Initial state
  Legal actions in every state
  Results of performing every action in every state
  Reward for every state
  Whether or not a state is terminal
Resources
  Sufficient time and space
to search the entire game tree

Results
  Players can find optimal actions on each time step
NB: *Sometimes* possible without searching entire tree

This rules out puzzles like Rubik’s Cube, which requires more complicated techniques, some of which are discussed in subsequent lessons.
Implementation

var role, roles, state, library, startclock, playclock;

function ping ()
    {return 'ready'}

function start (r,rs,sc,pc)
    {role = r;
     library = rs;  // definerules([],rs.slice(1));
     roles = findroles(library);
     state = findinits(library);
     startclock = numberize(sc);
     playclock = numberize(pc);
     return 'ready'}

function play (move)
    {if (move!==nil) {state = simulate(move,state,library)};
     if (findcontrol(state,library)!==role) {return false};
     return bestmove(state)}

function stop (move)
    {return false}

function abort ()
    {return false}
Legal Player:

```javascript
function play (move) {
  if (move!==nil) {state = simulate(move,state,library)};
  if (findcontrol(state,library)!==role) {return false};
  return findlegalx(state,library)}
```

Maximizer:

```javascript
function play (id,move) {
  if (move!='nil') {state=simulate(move,state,ruleset)};
  if (findcontrol(state,library)!==role) {return false};
  return bestmove(state)}
```
function bestmove (state)
    {var actions = findlegals(state,library);
        var action = actions[0];
        var score = 0;
        for (var i=0; i<actions.length; i++)
            {var newstate = simulate(actions[i],state,library);
                var result = maxscore(newstate);
                if (result==100) {return actions[i]};
                if (result>score)
                    {score = result; action = actions[i]};
            }
        return action}
function maxscore (state)
    {if (findterminalp(state,ruleset))
      {return findreward(role,state,ruleset)};
    var actions = findlegals(role,state,ruleset);
    var score = 0;
    for (var i=0; i<actions.length; i++)
      {var newstate = simulate(actions[i],state,rules);
        var result = maxscore(newstate);
        if (result==100) {return 100};
        if (result>score) {score = result};
      }
    return score}
Minimax
Example:

More complicated than single-player games
Changes depend on actions of others
and those actions cannot be controlled
So player must consider all possible actions of others
Fixed Sum Games

Total reward in all states is the same
For one player to get more, the others must get less

Also called zero sum in cases where sum is 0
Note that, in GGP, all rewards are non-negative. However, can be scaled - to 50, for example.

Many common games are zero sum
e.g. Chess - winner and loser

Variable Sum Games

Possible for one player to get more without other players getting less
Some games are even cooperative
Small Games

Resources
  Sufficient time and space
to search the entire game tree

Results
  Players can find optimal strategy
  Not necessarily sequential, as with single player games
  Plans may be conditional on the actions others
  NB: Sometimes possible without searching entire tree

Small size rules out games like Chess and Othello, which require more complicated techniques, some of which are discussed in subsequent lessons.
Opponent Modeling
  May assume opponent will play rationally
  but
  Assumption may be wrong

Also, players do not know identities of other players

Common Alternative - pessimistic / conservative
  Assume the other player will do the worst possible thing
  Not maximizing its score, minimizing your score
Intuition - Select a move that is guaranteed to produce the highest possible return no matter what the opponents do.

In the case of a one move game, a player should choose an action such that the value of the resulting state for any opponent action is greater than or equal to the value of the resulting state for any other action and opponent action.

In the case of a multi-move game, minimax goes to the end of the game and “backs up” values.
Bipartite Game Tree
The value of a max node for player $p$ is either the utility of that state if it is terminal or the maximum of all values for the min nodes that result from its legal actions.

$$\text{value}(p,s) = \begin{cases} 
\text{goal}(p,s) & \text{if terminal}(s) \\
\max(\{\text{value}(p,\text{simulate}(a,s)) \mid \text{legal}(a,s)\}) & \text{otherwise}
\end{cases}$$

The value of a min node is the minimum value that results from any legal opponent action.

$$\text{value}(p,s) = \min(\{\text{value}(p,\text{simulate}(b,s)) \mid \text{legal}(b,s)\})$$
Bipartite Game Tree
Implementation

```javascript
var role, roles, state, library, startclock, playclock;

function ping ()
    {return 'ready'}

function start (r,rs,sc,pc)
    {role = r;
        library = rs;   // definerules([],rs.slice(1));
        roles = findroles(library);
        state = findinits(library);
        startclock = numberize(sc);
        playclock = numberize(pc);
        return 'ready'}

function play (move)
    {if (move!==nil) {state = simulate(move,state,library)};
        if (findcontrol(state,library) !== role) {return false};
        return bestmove(state)}

function stop (move)
    {return false}

function abort ()
    {return false}
```
function bestmove (state)
{var actions = findlegals(role,state,ruleset);
 var action = actions[0];
 var score = 0;
 for (var i=0; i<actions.length; i++)
 {var newstate = simulate(actions[i],state,library);
  var newscore = minimax(newstate);
  if (result>score) {score = result; action = actions[i]};
}
return action}
function minimax (state) {
  if (findterminalp(state,library)) {
    return findreward(role,state,library)*1};
  var active = findcontrol(state,library);
  if (active===role) {return maximize(state)};
  return minimize(state)
function maximize (state)
{var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var score = 0;
  for (var i=0; i<actions.length; i++)
    {var newstate = simulate(actions[i],state,library);
     var newscore = minimax(newstate);
     if (newscore>score) {score = newscore};
    }
  return score}

function minimize (state)
{var actions = findlegals(state,library);
  if (actions.length===0) {return 0};
  var score = 100;
  for (var i=0; i<actions.length; i++)
    {var newstate = simulate(actions[i],state,library);
     var newscore = minimax(newstate);
     if (newscore<score) {score = newscore};
    }
  return score}
If the minvalue for an action is determined to be 100, then there is no need to consider other actions.

In computing the minvalue for a state if the value to the first player of an opponent’s action is 0, then there is no need to consider other possibilities.
Bounded Minimax Example
As stated
100 is the limiting case for maxscore
0 is the limiting case for minscore

Other possibilities
Satisficing - fixed minimal score all that is needed
Fixed sum game - 51 is sufficient
Alpha-Beta
Alpha-Beta Search - Same as Bounded Minimax except that bounds are computed dynamically and passed along as parameters.

If partial result of min node less than alpha, can only decrease score and player need not consider.

If partial result of max node greater than beta, can only increase score and opponent will not allow.
Alpha-Beta Example
Alpha-Beta Example
Best case of alpha-beta pruning can reduce search space to square root of the unpruned search space, thereby dramatically increasing depth searchable within given time bound.

For example, it could in some cases reduce a tree with branching factor of 25 to branching factor of 5.
Details
(1) Create your player code.

(2) Test your player code: Open playeropen resource, load player code, use manager to play game.

(3) Download existing player (e.g. Legal) and replace player code with your player code. Then, upload to server and launch from Gamemaster's player page.
Resources

Game Development Utilities

- Rule Checker - allows users to test game descriptions (Sierra configuration)
- Style Checker - allows users to test Javascript stylesheets on sample states.

Game Management Applications

- Standalone - Standalone manager for games on server
- Manager - Regular manager for games on server
- Standaloneopen - Standalone manager for games described in local files
- Manageropen - Regular manager for games described in local files

- Human - Human player page for games on server
- Player - Automated player page for games on server
- Humanopen - Human player page for games described in local files
- Playeropen - Automated player page for games described in local files

Standard Players

- Legal - Legal player
- Random - Random player
- OneStep - One Step player
- TwoStep - Two Step player
- Minimax - Full Minimax player
- MinimaxDepth - Minimax player with fixed depth
- MinimaxID - Minimax player with iterative deepening
- Greedy - Greedy player
- MCS - Monte Carlo Search player
<table>
<thead>
<tr>
<th>Player</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>egghead</td>
<td>Launch</td>
</tr>
<tr>
<td>greedy</td>
<td>Launch</td>
</tr>
<tr>
<td>lady</td>
<td>Launch</td>
</tr>
<tr>
<td>lara</td>
<td>Launch</td>
</tr>
<tr>
<td>legal</td>
<td>Launch</td>
</tr>
<tr>
<td>mcs</td>
<td>Launch</td>
</tr>
<tr>
<td>minimax</td>
<td>Launch</td>
</tr>
<tr>
<td>minimaxdepth</td>
<td>Launch</td>
</tr>
<tr>
<td>minimaxid</td>
<td>Launch</td>
</tr>
<tr>
<td>onestep</td>
<td>Launch</td>
</tr>
<tr>
<td>random</td>
<td>Launch</td>
</tr>
<tr>
<td>slowpoke</td>
<td>Launch</td>
</tr>
<tr>
<td>twostep</td>
<td>Launch</td>
</tr>
</tbody>
</table>
<html>
<head>
<!-- stylesheet links -->
<script type="text/javascript" src="epilepsy/javascript/epilepsy.js"></script>
<script type="text/javascript" src="..\javascript\localstorage.js"></script>
<script type="text/javascript" src="..\javascript\legal.js"></script>
<script type="text/javascript" src="..\javascript\reasoning\general.js"></script>

// Customization

var manager = 'manager';
var player = 'legal';

// End of Customization

</head>

<body bgcolor="#abb9bb" onload="doinitialize()">
<center>
<table width="720" cellspacing="0" cellpadding="40" bgcolor="#ffffff">
<tr><td>
</td></tr>
</table>
</center>

<table width="640" cellspacing="8">
<tr><td align="center" style="font-size:18pt;">
</td>
</tr>
</table>
</body>
</html>