General Game Playing

Introduction

Michael Genesereth
Computer Science Department
Stanford University
Game Playing

Human Game Playing
• Intellectual Activity
• Competition

Computer Game Playing
• Testbed for AI
• Limitations
Limitations of Game Playing for AI

Narrowness
- Good at one game, not so good at others
- Cannot do anything else

Not really testing intelligence of machine
- Programmer does all the interesting analysis / design
- Machine simply follows the recipe
General Game Players are systems able to play arbitrary games effectively based solely on formal descriptions supplied at “runtime”.

Translation: They don’t know the rules until the game starts.

Must figure out for themselves:
- legal moves, winning strategy
- in the face of incomplete info and resource bounds
Versatility
Novelty
International GGP Competition
Annual GGP Competition
Held at AAAI or IJCAI conference
Administered by Stanford University
(Stanford folks not eligible to participate)
Winners
2005 - ClunePlayer - Jim Clune (USA)
2006 - FluxPlayer - Schiffel, Thielscher (Germany)
2007 - CadiaPlayer - Bjornsson, Finsson (Iceland)
2008 - CadiaPlayer - Bjornsson, Finsson (Iceland)
2010 - Ary - Mehat (France)
2011 - TurboTurtle - Schreiber (USA)
2012 - CadiaPlayer - Bjornsson, Finsson (Iceland)
2013 - TurboTurtle - Schreiber (USA)
2014 - Sancho - Draper (USA), Rose (UK)
2015 - Galvanise - Emslie
2016 - WoodStock - Piette (France)
GGP-07, GGP-08, GGP-12 Winners
Carbon versus Silicon
Human Race Being Defeated
Game Description
Multiplicity of Games
Finite Synchronous Games

Environment
- Environment with finitely many states
- One initial state and one or more terminal states
- Each state has a unique goal value for each player

Players
- Fixed, finite number of players
- Each with finitely many moves

Dynamics
- Finitely many steps
- Only one player moves on each step
- Environment changes only in response to moves
Common Structure
Good News: Since all of the games that we are considering are finite, it is possible in principle to communicate game information in the form of state graphs.

Problem: Size of description. Even though everything is finite, these sets can be large.

Solution: Exploit regularities / structure in state graphs to produce compact encoding
Structured State Machine
Actions

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>o</td>
<td>b</td>
</tr>
<tr>
<td>x</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>b</td>
<td>x</td>
</tr>
<tr>
<td>o</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Marked cell: \(1,3\)
Game Description Language

\[
\begin{align*}
\text{init} & (\text{cell}(1,1,b)) \\
\text{init} & (\text{cell}(1,2,b)) \\
\text{init} & (\text{cell}(1,3,b)) \\
\text{init} & (\text{cell}(2,1,b)) \\
\text{init} & (\text{cell}(2,2,b)) \\
\text{init} & (\text{cell}(2,3,b)) \\
\text{init} & (\text{cell}(3,1,b)) \\
\text{init} & (\text{cell}(3,2,b)) \\
\text{init} & (\text{cell}(3,3,b)) \\
\text{init} & (\text{control}(x))
\end{align*}
\]

\[
\begin{align*}
\text{next} & (\text{cell}(M,N,P)) : - \\
& \text{next} & (\text{cell}(M,N,Z)) : - \\
& \text{next} & (\text{cell}(M,N,b)) : - \\
\end{align*}
\]

\[
\begin{align*}
\text{legal} (P, \text{mark}(X,Y)) : - \\
& \text{legal} (x, \text{noop}) : - \\
& \text{legal} (o, \text{noop}) : - \\
\end{align*}
\]

\[
\begin{align*}
\text{row} (M,P) : - \\
\text{column} (N,P) : - \\
\text{diagonal} (P) : - \\
\end{align*}
\]

\[
\begin{align*}
\text{goal} (x, 100) : - \\
\text{goal} (x, 50) : - \\
\text{goal} (x, 0) : - \\
\text{goal} (o, 100) : - \\
\text{goal} (o, 50) : - \\
\text{goal} (o, 0) : - \\
\end{align*}
\]

\[
\begin{align*}
\text{terminal} : - \\
\text{open} : - \\
\text{draw} : - \\
\end{align*}
\]
What we see:

\[
\text{next}(\text{cell}(M,N,x)) :-
\]
\[
\text{does(white,mark}(M,N)) \ & \ 
\text{true(cell}(M,N,b))
\]

What the player sees:

\[
\text{next(welcoul}(M,N,himenoing)) :-
\]
\[
\text{does(himenoing,dukepse}(M,N)) \ & \ 
\text{true(welcoul}(M,N,lorenchise))
\]
Game Playing
cell(1,1,b)
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,b)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,b)
control(x)
mark(1,1)
mark(1,2)
mark(1,3)
mark(2,1)
mark(2,2)
mark(2,3)
mark(3,1)
mark(3,2)
mark(3,3)
State Update

cell(1,1,b)    →    cell(1,1,b)
cell(1,2,b)    →    cell(1,2,b)
cell(1,3,b)    →    cell(1,3,x)
cell(2,1,b)    →    cell(2,1,b)
cell(2,2,b)    →    cell(2,2,b)
cell(2,3,b)    →    cell(2,3,b)
cell(3,1,b)    →    cell(3,1,b)
cell(3,2,b)    →    cell(3,2,b)
cell(3,3,b)    →    cell(3,3,b)
control(x)     →    control(o)
Complete Game Graph Search
How do we evaluate non-terminal states?
General Heuristics

- Goal proximity (everyone)
- Maximize mobility (Barney Pell)
- Minimize opponent’s mobility (Jim Clune)
GGP-06 Final - Cylinder Checkers
Second Generation GGP (2007 on)

Monte Carlo Search

Monte Carlo Tree Search
  UCT - Uniform Confidence Bounds on Trees
Second Generation GGP

Monte Carlo Search
Offline Processing of Game Descriptions
  Compile to do the search faster
  Reformulate problem to decrease size of search space

What human programmers do in creating game players
Compilation

Conversion of logic to traditional programming language
Simple, widely published algorithms
several orders or magnitude speedup
no asymptotic change

Conversion to Field Programmable Gate Arrays (FPGAs)
several more orders of magnitude improvement
Hodgepodge = Chess + Othello

Analysis of joint game:
Branching factor as given to players: \( a \times b \)
Fringe of tree at depth \( n \) as given: \( (a \times b)^n \)
Fringe of tree at depth \( n \) factored: \( a^n + b^n \)
Examples
- Factoring, e.g. Hodgepodge
- Bottlenecks, e.g. Triathlon
- Symmetry detection, e.g. Tic-Tac-Toe
- Dead State Removal

Trade-off - cost of finding and using structure vs savings
- Sometimes cost proportional to size of description
- Sometimes savings proportional to size of game tree
Automatic Programming

```java
public class CreateObjectDemo {
    public static void main(String[] args) {
        // create a point object and two rectangle objects
        Point origin_one = new Point(23, 94);
        Rectangle rect_one = new Rectangle(origin_one, 100, 200);
        Rectangle rect_two = new Rectangle(50, 100);

        // display rect_one's width, height, and area
        System.out.println("Width of rect_one: " + rect_one.width);
        System.out.println("Height of rect_one: " + rect_one.height);
        System.out.println("Area of rect_one: " + rect_one.area());

        // set rect_two's position
        rect_two.origin = origin_one;

        // display rect_two's position
        System.out.println("X Position of rect_two: " + rect_two.origin.x);
        System.out.println("Y Position of rect_two: " + rect_two.origin.y);

        // move rect_two and display its new position
        rect_two.move(40, 72);
    }
}
```
Algorithmic Expertise

Knuth in a Box
Game Theory

\[\begin{array}{cc}
    a & b \\
    \begin{array}{ccc}
        a & 4 & 3 \\
        4 & 3 & 1 \\
        2 & 1 & 1 \\
    \end{array} & \begin{array}{ccc}
        a & 4 & 1 \\
        4 & 1 & 2 \\
        3 & 2 & 2 \\
    \end{array}
\end{array}\]
Psychology

Demoralizing the Opponent
Fooling the Opponent
Conclusion
General Game Playing is not a game

SOMEONE SAID:

I DID MY MASTER'S THESIS ON DOTS AND BOXES!

NOBODY LAUGHED...
Theory of Intelligence

Dimensions of Intelligence
  Representation of the World
  Correct and efficient reasoning
  Rationality with incomplete info and resource bounds

Generality
  Not just ability to perform well on specific tasks
  But also ability to perform well in general
  Test of intelligence, not just test of knowledge
The main advantage we expect the advice taker to have is that its behavior will be improvable merely by making statements to it, telling it about its … environment and what is wanted from it.

- John McCarthy 1958
The potential use of computers by people to accomplish tasks can be “one-dimensionalized” into a spectrum representing the nature of the instruction that must be given the computer to do its job. Call it the **what-to-how spectrum**.

At one extreme of the spectrum, the user supplies his intelligence to instruct the machine with precision exactly how to do his job step-by-step. ... At the other end of the spectrum is the user with his real problem. ... He aspires to communicate what he wants done ... without having to lay out in detail all necessary subgoals for adequate performance.

- Ed Feigenbaum 1974
The General Problem Solver demonstrates how generality can be achieved by factoring the specific descriptions of individual tasks from the task-independent processes.
A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. 

Specialization is for insects.
Course Details
<table>
<thead>
<tr>
<th>Month</th>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>April</td>
<td>3</td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Game Description</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Game Playing</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>Incomplete Search</td>
</tr>
<tr>
<td>May</td>
<td>1</td>
<td>Statistical Search</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Logical Optimization</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Materialization and Reformulation</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>Game Tree Reformulation, e.g. Factoring</td>
</tr>
<tr>
<td>June</td>
<td>5</td>
<td>Final Competition</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td><em>Really</em> General Game Playing</td>
</tr>
</tbody>
</table>
Teams

Composition

3 people each (2 or 4 okay with *good* reason)

Names:

- Pansy Division
- The Pumamen
- Team Camembert
- Mighty Bourgeoisie
- Greedy Bastards
- Red Hot Chili Peppers
- Michael Genesereth
- /*^*\n- X Æ A-12

Identifiers:

- pansy_division
- punamen
- camembert
- bourgeoisie
- greedybastards
- peppers
- michael_genesereth
- happy
- x_ash_a_12
Technology

Language
Java
***Javascript***
Fortran

Operating System
Mac OS
Unix
Linux

Hardware
Whatever you like … but …
Able to access course website
Required Components
Weekly Assignments
Weekly Competitions
Final Report

Extra Credit Components
Class Participation
Forum Participation
Novel ideas

You do not have to win competitions to get a perfect score, but your players must play correctly and illustrate weekly lessons.

No curve. Grades are based completely on mastery of subject matter as demonstrated via components above.

*Grades in this course are generally quite high (because people tend to work hard).*
http://cs227b.stanford.edu