General Game Playing

Introduction

Michael Genesereth
Computer Science Department
Stanford University
Human Game Playing
• Intellectual Activity
• Skill Comparison

Computer Game Playing
• Testbed for AI
• Limitations
Narrowsness
  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
International GGP Competition
Annual GGP Competition
Held at AAAI or IJCAI conference
Administered by Stanford University
(Stanford folks not eligible to participate)
History

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)
International GGP Competition
GGP-07, GGP-08, GGP-12 Winners
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
<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

- $\text{cell}(1,1, x)$
- $\text{cell}(1,2, b)$
- $\text{cell}(1,3, b)$
- $\text{cell}(2,1, b)$
- $\text{cell}(2,2, o)$
- $\text{cell}(2,3, b)$
- $\text{cell}(3,1, b)$
- $\text{cell}(3,2, b)$
- $\text{cell}(3,3, x)$
- $\text{control}(o)$
Actions

\[
\begin{array}{c}
\text{cell}(1,1,x) \\
\text{cell}(1,2,b) \\
\text{cell}(1,3,b) \\
\text{cell}(2,1,b) \\
\text{cell}(2,2,o) \\
\text{cell}(2,3,b) \\
\text{cell}(3,1,b) \\
\text{cell}(3,2,b) \\
\text{cell}(3,3,x) \\
\text{control}(o)
\end{array}
\rightarrow
\begin{array}{c}
\text{cell}(1,1,x) \\
\text{cell}(1,2,b) \\
\text{cell}(1,3,o) \\
\text{cell}(2,1,b) \\
\text{cell}(2,2,o) \\
\text{cell}(2,3,b) \\
\text{cell}(3,1,b) \\
\text{cell}(3,2,b) \\
\text{cell}(3,3,x) \\
\text{control}(x)
\end{array}
\]
init\(\text{cell}(1,1,b))
init\(\text{cell}(1,2,b))
init\(\text{cell}(1,3,b))
init\(\text{cell}(2,1,b))
init\(\text{cell}(2,2,b))
init\(\text{cell}(2,3,b))
init\(\text{cell}(3,1,b))
init\(\text{cell}(3,2,b))
init\(\text{cell}(3,3,b))
init\(\text{control}(x))

legal\(\text{P},mark(X,Y)) \:-\)
true\(\text{cell}(X,Y,b)) \&
true\(\text{control}(P))

legal\(\text{x},noop) \:-\)
true\(\text{control}(o))

legal\(\text{o},noop) \:-\)
true\(\text{control}(x))

next\(\text{cell}(M,N,P)) \:-\)
does\(\text{P},mark(M,N))
next\(\text{cell}(M,N,Z)) \:-\)
does\(\text{P},mark(M,N)) \&
true\(\text{cell}(M,N,Z)) \& Z#b
next\(\text{cell}(M,N,b)) \:-\)
does\(\text{P},mark(J,K)) \&
true\(\text{cell}(M,N,b)) \&
(M#J \mid N#K)
next\(\text{control}(x)) \:-\)
true\(\text{control}(o))
next\(\text{control}(o)) \:-\)
true\(\text{control}(x))

terminal \:- \text{line}(P)
terminal \:- \sim\text{open}

goal\(\text{x},100) \:- \text{line}(x)
goal\(\text{x},50) \:- \text{draw}
goal\(\text{x},0) \:- \text{line}(o)
goal\(\text{o},100) \:- \text{line}(o)
goal\(\text{o},50) \:- \text{draw}
goal\(\text{o},0) \:- \text{line}(x)
goal\(\text{o},100) \:- \text{line}(o)
goal\(\text{o},50) \:- \text{draw}
goal\(\text{o},0) \:- \text{line}(x)

diagonal\(\text{P}) \:-\)
true\(\text{cell}(1,1,P)) \&
true\(\text{cell}(2,2,P)) \&
true\(\text{cell}(3,3,P))

row\(\text{M,P}) \:-\)
true\(\text{cell}(M,1,P)) \&
true\(\text{cell}(M,2,P)) \&
true\(\text{cell}(M,3,P))
column\(\text{N,P}) \:-\)
true\(\text{cell}(1,N,P)) \&
true\(\text{cell}(2,N,P)) \&
true\(\text{cell}(3,N,P))

diagonal\(\text{P}) \:-\)
true\(\text{cell}(1,3,P)) \&
true\(\text{cell}(2,2,P)) \&
true\(\text{cell}(3,1,P))

line\(\text{P}) \:-\)
row\(\text{M,P}) \&
true\(\text{cell}(M,1,P)) \&
true\(\text{cell}(M,2,P)) \&
true\(\text{cell}(M,3,P))
column\(\text{N,P}) \:-\)
true\(\text{cell}(1,N,P)) \&
true\(\text{cell}(2,N,P)) \&
true\(\text{cell}(3,N,P))
diagonal\(\text{P}) \:-\)
true\(\text{cell}(1,3,P)) \&
true\(\text{cell}(2,2,P)) \&
true\(\text{cell}(3,1,P))

open \:- \text{true}(\text{cell}(M,N,b))
draw \:- \sim\text{line}(x) \&
\sim\text{line}(o)
What we see:

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

What the player sees:

\[
\text{next(welcoul(M,N,himenoing)) :- does(himenoing,dukepse(M,N)) \& true(welcoul(M,N,lorenchise))}
\]

Obfuscation
Game Playing
Initial State

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)
x's moves:

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

\[
\begin{align*}
\text{cell}(1,1,b) & \quad \text{cell}(1,1,b) \\
\text{cell}(1,2,b) & \quad \text{cell}(1,2,b) \\
\text{cell}(1,3,b) & \quad \text{cell}(1,3,x) \\
\text{cell}(2,1,b) & \quad \text{cell}(2,1,b) \\
\text{cell}(2,2,b) & \quad \text{cell}(2,2,b) \\
\text{cell}(2,3,b) & \quad \text{cell}(2,3,b) \\
\text{cell}(3,1,b) & \quad \text{cell}(3,1,b) \\
\text{cell}(3,2,b) & \quad \text{cell}(3,2,b) \\
\text{cell}(3,3,b) & \quad \text{cell}(3,3,b) \\
\text{control}(x) & \quad \text{control}(o)
\end{align*}
\]
Complete Game Graph Search
Incomplete Game Tree Search

How do we evaluate non-terminal states?
First Generation GGP (2005-2006)

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

Diagram of a tree structure with nodes labeled 25, 50, 0, and 75, and values at each level ranging from 0 to 100.
Offline Processing of Game Descriptions
  Reformulate problem to decrease size of search space
  Compile to do the search faster

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
\[ p(a, b) \quad q(b, c) \]
\[ \neg p(b, d) \quad \forall x. \forall y. (p(x, y) \Rightarrow q(x, y)) \]
\[ p(c, b) \lor p(c, d) \quad \exists x. p(x, d) \]
Algorithmic Expertise

Knuth in a Box
Opponent Modeling
Demoralizing the Opponent

X O

X X O

O O
Conclusion
General Game Playing is not a game

Someone said:
I did my Master's thesis on dots and boxes!

Nobody laughed...
Serious Business

ERP SYSTEM

Financial Management

Customer Relationship Management

Supply Chain Management

Human Resource Management

Manufacturing Resource Planning

SAP

ORACLE

IBM
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. To make these statements will require little, if any, knowledge of the program or the previous knowledge of the advice taker.
The General Problem Solver demonstrates how generality can be achieved by factoring the specific descriptions of individual tasks from the task-independent processes.
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.
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>March</td>
<td>30</td>
<td>Introduction</td>
</tr>
<tr>
<td>April</td>
<td>6</td>
<td>Game Description</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13  Game Playing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20  Incomplete Search</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27  Statistical Search</td>
</tr>
<tr>
<td>May</td>
<td>4</td>
<td>Logical Optimization</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Materialization and Reformulation</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Game Tree Reformulation, e.g. Factoring</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>Really General Game Playing</td>
</tr>
<tr>
<td>June</td>
<td>1</td>
<td>Final Competition</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
/*/^*/
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          Extra Credit Components
Weekly assignments            Class Participation
Weekly Competitions            Arena
Final Report                   Cool 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 work hard).