General Game Playing
Human Game Playing

• Intellectual Activity
• Skill Comparison
Computer Game Playing

• Fun
• Testbed for AI
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
Unlike specialized game players (e.g. Deep Blue), they do not use algorithms designed in advance for specific games.

Artificial Intelligence Technologies
- knowledge representation
- deduction, reformulation, induction, …
- rational behavior w/ uncertainty, resource bounds
Variety of Games
Novelty
International GGP Competition
Annual GGP Competition
Held at AAAI or IJCAI conference
Administered by Stanford University
(Stanford folks not eligible to participate)
## History

### Winners

<table>
<thead>
<tr>
<th>Year</th>
<th>Team/Player</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>ClunePlayer - Jim Clune</td>
<td>USA</td>
</tr>
<tr>
<td>2006</td>
<td>FluxPlayer - Schiffel, Thielscher</td>
<td>Germany</td>
</tr>
<tr>
<td>2007</td>
<td>CadiaPlayer - Bjornsson, Finsson</td>
<td>Iceland</td>
</tr>
<tr>
<td>2008</td>
<td>CadiaPlayer - Bjornsson, Finsson</td>
<td>Iceland</td>
</tr>
<tr>
<td>2010</td>
<td>Ary - Mehat</td>
<td>France</td>
</tr>
<tr>
<td>2011</td>
<td>TurboTurtle - Schreiber</td>
<td>USA</td>
</tr>
<tr>
<td>2012</td>
<td>CadiaPlayer - Bjornsson, Finsson</td>
<td>Iceland</td>
</tr>
<tr>
<td>2013</td>
<td>TurboTurtle - Schreiber</td>
<td>USA</td>
</tr>
</tbody>
</table>
GGP-05 Winner Jim Clune
GGP-07, GGP-08, GGP-12 Winners
Other Games, Other Winners
Carbon versus Silicon
Human Race Being Defeated
Game Description
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
- All players move on all steps (some no ops)
- Environment changes only in response to moves
Single Player Game

S1
0

S2
50

S3
50

S4
100

S5
0

S6
25

S7
25

S8
50

a
a
a
a

b
b
b
b

Multiple Player Game

\[
\begin{array}{cccccc}
S1 & \xrightarrow{a/a} & S2 & \xrightarrow{a/a} & S3 & \xrightarrow{a/a} & S4 \\
0/0 & b/a & 50/50 & b/a & 50/50 & b/a & 100/0 \\
S5 & \xrightarrow{b/a} & S6 & \xrightarrow{a/b} & S7 & \xrightarrow{a/b} & S8 \\
0/0 & a/a & 25/25 & a/a & 25/25 & a/a & 0/100 \\
\end{array}
\]
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
Game Description Language (or GDL) is a formal language for encoding the rules of games.

Game rules written as sentences in Symbolic Logic.

GDL is widely used in the research literature and is used in virtually all General Game Playing competitions.

GDL extensions are applicable in real-world applications such as Enterprise Management and Computational Law.
Example

Tic-Tac-Toe

role(x)
role(o)
init(cell(1,1,b))
init(cell(1,2,b))
init(cell(1,3,b))
init(cell(2,1,b))
init(cell(2,2,b))
init(cell(2,3,b))
init(cell(3,1,b))
init(cell(3,2,b))
init(cell(3,3,b))
init(control(x))
control(o)
control(x)

legal(P,mark(X,Y)) :-
true(cell(X,Y,b)) &
true(control(P))

legal(x,noop) :-
true(control(o))

legal(o,noop) :-
true(control(x))

next(cell(M,N,P)) :-
does(P,mark(M,N))

next(cell(M,N,E)) :-
does(P,mark(M,N)) &
true(cell(M,N,Z)) & Z!=b

next(cell(M,N,b)) :-
does(P,mark(J,K)) &
true(cell(M,N,b)) &
distinct(M,J)

next(cell(M,N,b)) :-
does(P,mark(J,K)) &
true(cell(M,N,b)) &
distinct(N,K)

next(control(x)) :-
true(control(o))

next(control(o)) :-
true(control(x))

terminal :- line(P)
terminal :- ~open

goal(x,100) :- line(x) & ~line(o)
goal(x,50) :- ~line(x) & ~line(o)
goal(x,0) :- ~line(x) & line(o)
goal(o,100) :- ~line(x) & line(o)
goal(o,50) :- ~line(x) & ~line(o)
goal(o,0) :- line(x) & ~line(o)

row(M,P) :-
true(cell(M,1,P)) &
true(cell(M,2,P)) &
true(cell(M,3,P))

column(N,P) :-
true(cell(1,N,P)) &
true(cell(2,N,P)) &
true(cell(3,N,P))
diagonal(P) :-
true(cell(1,1,P)) &
true(cell(2,2,P)) &
true(cell(3,3,P))
diagonal(P) :-
true(cell(1,3,P)) &
true(cell(2,2,P)) &
true(cell(3,1,P))

diagonal(P) :-
true(cell(M,N,P))

diagonal(P) :-
true(cell(M,N,P))

diagonal(P) :-
true(cell(M,N,P))

diagonal(P) :-
true(cell(M,N,P))

line(P) :- row(M,P)
line(P) :- column(N,P)
line(P) :- diagonal(P)

open :-
true(cell(M,N,b))
Game Management
Game Management is the process of administering a game in General Game Playing.

Match = instance of a game.

Components:
  Game Manager
  Game Playing Protocol
Game Manager

Graphics for Spectators

Game Manager

Game Descriptions
Match Records

Temporary State Data

Tcp/ip

Player
Start

Manager sends Start message to players
Start\((id, \ role, \ description, \ startclock, \ playclock)\)
Start
Manager sends Start message to players
Start(id, role, description, startclock, playclock)

Play
Manager sends Play messages to players
Play(id, actions)
Receives plays in response
Start
Manager sends Start message to players
\[ \text{start}(id, role, description, startclock, playclock) \]

Play
Manager sends Play messages to players
\[ \text{play}(id, actions) \]
Receives plays in response

Stop
Manager sends Stop message to players
\[ \text{stop}(id, actions) \]
Game Playing
Logical reasoning in searching game tree:
  Initial state
  Legal actions for each player in each state
  State resulting from execution of legal action
  Value of each state for each player
  Determination of whether state is terminal

Easy to convert from logic to other representations
  Simplicity of logical formulation
  Simple, widely published algorithms
  3-4 orders or magnitude speedup
  no asymptotic change
Initial State
Legal Moves

White’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)

Black’s moves:

- noop
State Update

mark(1,3) -> noop

X
Game Tree Expansion
Game Tree Search

[Diagram of a game tree search for Tic-Tac-Toe]
Large state spaces
- \( \approx 5000 \) states in Tic-Tac-Toe
- \( >10^{30} \) states in Chess

Limited Resources
- Memory
- Time (start clock, move clock)
Incremental Search

Expand Game Graph incrementally
As much as time allows
Minimax/etc. evaluation on non-terminal states
using an evaluation function of some sort

But how do we evaluate non-terminal states?

In traditional game playing, the rules are known in advance; and the programmer can invent game-specific evaluation functions. Not possible in GGP.
Guaranteed Evaluation Functions

Ideas
* Novelty with reversibility
* Goal-monotonic observables
* Bad states, useless moves
  <insert your good idea here>
Ideas

Goal proximity (everyone)
Maximize mobility (Barney Pell)
Minimize opponent’s mobility (Jim Clune)
<insert your good idea here>
GGP-06 Final - Cylinder Checkers
GGP-06 Final - Cylinder Checkers
Monte Carlo
Metagaming is the process of reasoning about games and, by extension, game players and game playing.

Done *offline*, i.e. during the start clock or between moves or in parallel with regular game play.

*What programmers do in creating specific / general players*
Differences from Game tree search
  more information, e.g. tournament standing
  less information, i.e. more general
  goal - create / optimize player to play games effectively

Techniques:
  Analysis of propositional nets and rule graphs
  Proofs using logic
  Compilation into machine language and/or FPGAs
Hodgepodge = Chess + Othello

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

Trade-off - cost of finding structure vs savings
Sometimes cost proportional to size of description
Sometimes savings proportional to size of the game tree
Axiomatization:

\[ \text{redwin} ::= \text{left}(X) \land \text{redpath}(X,Y) \land \text{right}(Y) \]
\[ \text{redpath}(X,X) ::= \text{true} \left( \text{cell}(X,\text{red}) \right) \]
\[ \text{redpath}(X,Z) ::= \text{true} \left( \text{cell}(X,\text{red}) \right) \land \text{adjacent}(X,Y) \land \text{redpath}(Y,Z) \]

Results:

- Very expensive if path exists
- Can run forever if not
Improved Axiomatization

Axiomatization:

\[
\begin{align*}
\text{redwin} &: \leftarrow \text{left}(X) \& \text{redpath}(X,Y,\text{nil}) \& \text{right}(Y) \\
\text{redpath}(X,X,P) &: \leftarrow \text{true(cell}(X,\text{red})) \\
\text{redpath}(X,Z,P) &: \\
& \quad \text{true(cell}(X,\text{red})) \& \\
& \quad \text{adjacent}(X,Y) \& \\
& \quad \sim \text{member}(Y,P) \& \\
& \quad \text{redpath}(Y,Z,\text{cons}(Y,P))
\end{align*}
\]

Results:

Does not run forever

can take \sim 1 second to compute in bad cases
Result:
takes <1 millisecond to compute in worst cases
\[ p(a,b) \quad q(b,c) \]
\[ \neg p(b,d) \quad \forall x. \forall y. (p(x,y) \Rightarrow q(x,y)) \]
\[ p(c,b) \lor p(c,d) \quad \exists x. p(x,d) \]
Expertise in a Box
Opponent Modeling
Demoralizing the Opponent

\[ \begin{array}{cc}
X & O \\
X & X & O \\
O & & \\
\end{array} \]
Philosophical Remarks
Critique: Game playing is *frivolous*.

Serious Applications:
- Enterprise Management
- Computational Law

Funders:
- Darpa
- SAP
Characteristics of GGP
  game descriptions contain full information
  which determine optimal behavior

Useful for evaluating theories of intelligence
  effects of representation
  incompleteness of information
  resource bounds
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 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.
http://cs227b.stanford.edu
Schedule

April
3 Introduction
10 Complete Search
17 Incomplete Search
24 Monte Carlo Search

May
1 Monte Carlo Tree Search
8 Propositional Nets
15 Factoring
22 Optimization
29 Really General Game Playing

June
5 Final Competition
Teams

Composition

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

Names:

- Pansy Division
- The Pumamen
- Team Camembert
- Mighty Bourgeoisie
- Michael Genesereth
- Greedy Bastards
- Red Hot Chili Peppers

Identifiers:

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

Language
  Java
  Javascript
  Fortran

Operating System
  Mac OS
  Unix
  Linux

Hardware
  Whatever you like … but …
  Accessible via Stanford Wifi
GGP-06 Winners

AAAII-06 Winner

Fluxplayer

Stephan Schiffel and Michael Thielscher

Dresden University
GGP-11, GGP-13 Winner