Logic Programming

General Game Playing

Michael Genesereth
Computer Science Department
Stanford University
Game Playing

Human Game Playing
• Intellectual Activity
• Skill Comparison

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
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)
GGP-05 Winner Jim Clune
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)
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
Multiple Player Game
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.
Symbols:

- white, black - roles
- 1, 2, 3 - indices of rows and columns
- x, o, b - marks
- noop - no-op action

Constructors:

- cell(index, index, mark) --> proposition
- control(role) --> proposition
- mark(index, index) --> action

Predicates:

- row(index, mark)
- column(index, mark)
- diagonal(mark)
- line(mark)
- open
State Representation

cell(1,1,x)
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,o)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,x)
control(black)
Rules of Tic-Tac-Toe

role(white)
role(black)
base(cell(M,N,Z)) :-
    index(M) &
    index(N) &
    filler(Z)
base(control(W)) :- role(W)

input(W,mark(X,Y)) :-
    role(W) &
    index(X) &
    index(Y)
input(W,noop) :- role(W)

init(cell(X,Y,b)) :-
    index(X) &
    index(Y)
init(control(white))

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

next(cell(M,N,x)) :-
    does(white,mark(M,N))
next(cell(M,N,0)) :-
    does(black,mark(M,N))
next(cell(M,N,Z)) :-
    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(white)) :-
    true(control(black))

next(control(black)) :-
    true(control(white))

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

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

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

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

terminal :- line(P)
terminal :- ~open
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))
role(white)
role(black)
**Propositions**

\[
\text{base}(\text{cell}(X,Y,W)) \leftarrow \\
\quad \text{index}(X) \; \& \\
\quad \text{index}(Y) \; \& \\
\quad \text{filler}(W)
\]

\[
\text{base}(\text{control}(W)) \leftarrow \\
\quad \text{role}(W)
\]

index(1)  
index(2)  
index(3)  

filler(x)  
filler(o)  
filler(b)
\textbf{Actions}

\texttt{input(W,mark(X,Y)) :-}
\begin{itemize}
  \item role(W)
  \item index(X)
  \item index(Y)
\end{itemize}

\texttt{input(W,noop) :-}
\begin{itemize}
  \item role(W)
\end{itemize}
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(white))
legal(W, mark(X,Y)) :-
  true(cell(X,Y,b)) &
  true(control(W))

legal(white,noop) :-
  true(control(black))

legal(black,noop) :-
  true(control(white))
next(cell(M,N,x)) :-
    does(white,mark(M,N))

next(cell(M,N,o)) :-
    does(black,mark(M,N))

next(cell(M,N,b)) :-
    does(W,mark(J,K)) &
    true(cell(M,N,b)) & M!=J

next(cell(M,N,b)) :-
    does(W,mark(J,K)) &
    true(cell(M,N,b)) & N!=K

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

next(control(white)) :-
    true(control(black))

next(control(black)) :-
    true(control(white))
row(M,W) :-
  true(cell(M,1,W)) &
  true(cell(M,2,W)) &
  true(cell(M,3,W))

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

diagonal(W) :-
  true(cell(1,1,W)) &
  true(cell(2,2,W)) &
  true(cell(3,3,W))

diagonal(W) :-
  true(cell(1,3,W)) &
  true(cell(2,2,W)) &
  true(cell(3,1,W))

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

open :- true(cell(M,N,b))
goals and Termination

\[ \text{goal(white,100)} :\text{- line(x)} \]
\[ \text{goal(white,50)} :\text{- line(x)} \& \text{line(o)} \]
\[ \text{goal(white,0)} :\text{- line(o)} \]
\[ \text{goal(black,100)} :\text{- line(o)} \]
\[ \text{goal(black,50)} :\text{- line(x)} \& \text{line(o)} \]
\[ \text{goal(black,0)} :\text{- line(x)} \]

\[ \text{terminal} :\text{- line(W)} \]
\[ \text{terminal} :\text{- open} \]
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
- Temporary State Data
- Game Manager
- Game Descriptions
- Match Records
- Player
- Tcp/ip
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
General Game Playing Protocol

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

Stop
Manager sends Stop message to players
stop(id, actions)
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)
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

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)
cell(1,3)  cell(1,3)
cell(1,3)  cell(1,3)
noop
control(x)  control(o)
Complete Game Graph 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
Monte Carlo Search

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

Monte Carlo Search
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: $a*b$
Fringe of tree at depth $n$ as given: $(a*b)^n$
Fringe of tree at depth $n$ factored: $a^n+b^n$
Reformulation Opportunities

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
```latex
\begin{align*}
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)
\end{align*}
```
Expertise in a Box
## Opponent Modeling

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>