



Introduction to Logic

General Game Playing



Game Playing



Human Game Playing

- Intellectual Activity
- Skill Comparison

Computer Game Playing

- Testbed for AI
- Limitations



/42

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

/42

General Game Playing

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.

/42

Technology

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

Artificial Intelligence Technologies

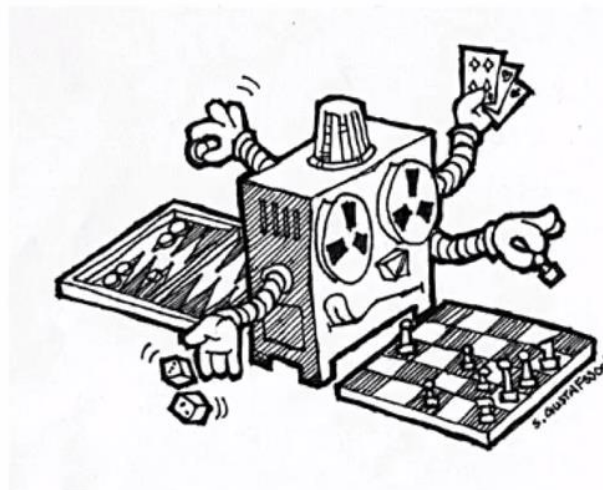
knowledge representation

deduction, reformulation, induction, ...

rational behavior w/ uncertainty, resource bounds

/42

Variety of Games



/42

Programme

- Game Description
- Game Management
- General Game Playing
- Philosophical Remarks
- International General Game Playing Competition

/42

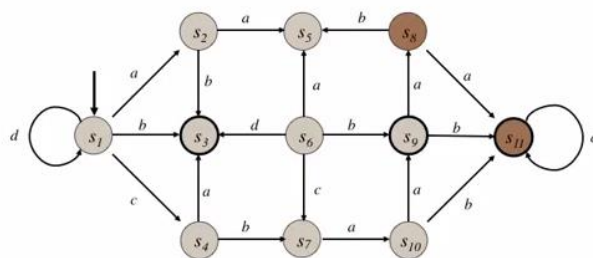
Related Fields

- Game Theory
 - Analysis of game trees
 - No concern for representation and how to use it
 - Little concern for deliberation vs. action
- Planning
 - Inputs: State machine, initial state, goal states
 - Objective: Plan to achieve a goal state
 - No execution environment
 - No concern for deliberation vs. action

/42

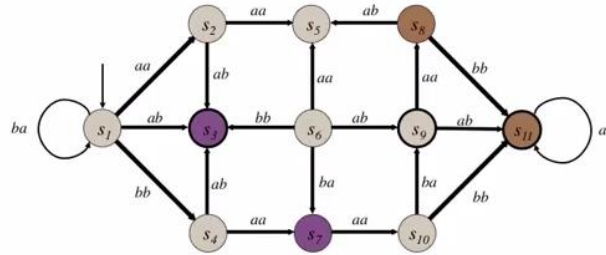
Game Description

Single Player Game as a State Machine



/42

Goal States and Terminal



/42

Direct Description

Since all of the games that we are considering are finite,
it is possible, in principle, to communicate game
information in the form of transition graphs.

Problem: Size of description. Even though everything is
finite, the graphs can be large.

Solution: Compact Encoding

/42

Game Description Language (GDL)

Game Description Language is a formal language for encoding finite, discrete games.

- (1) Conceptualization of game states as databases.
- (2) Herbrand Logic to define legality, update, goals, etc.

/42

States as Databases

X		
	O	
		X

```
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(o)
```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= ~EM:EN:true(cell(M,N,b))

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)) |

  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= ~EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |

  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= -EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |
  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= -EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |
  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= -EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |
  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= -EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |
  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= -EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |
  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Logical Encoding of Rules

```

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))

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,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)) &
  (M!=J | N!=K)

next(control(x)) <=
  true(control(o))

next(control(o)) <=
  true(control(x))

terminal <= line(P)
terminal <= -EM:EN:true(cell(M,N,b))

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)

line(P) <=>
  row(M,P) |
  column(N,P) |
  diagonal(P)

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)) |
  true(cell(1,3,P)) &
  true(cell(2,2,P)) &
  true(cell(3,1,P))

```

/42

Obfuscation

What we see:

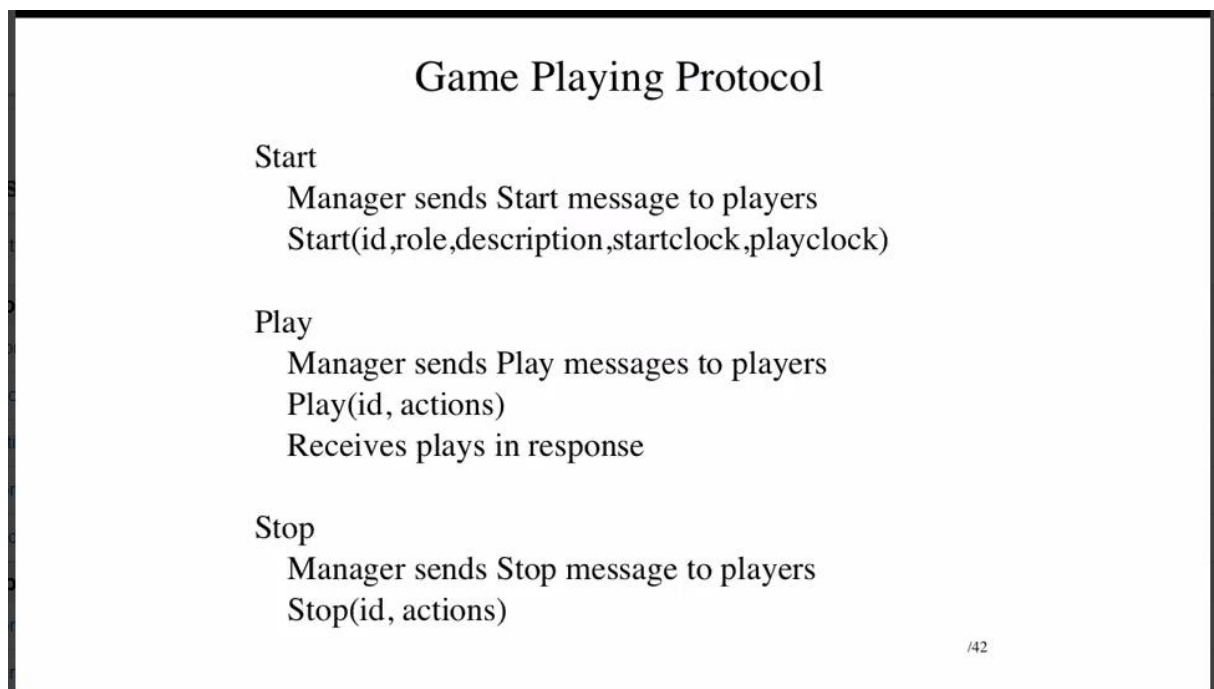
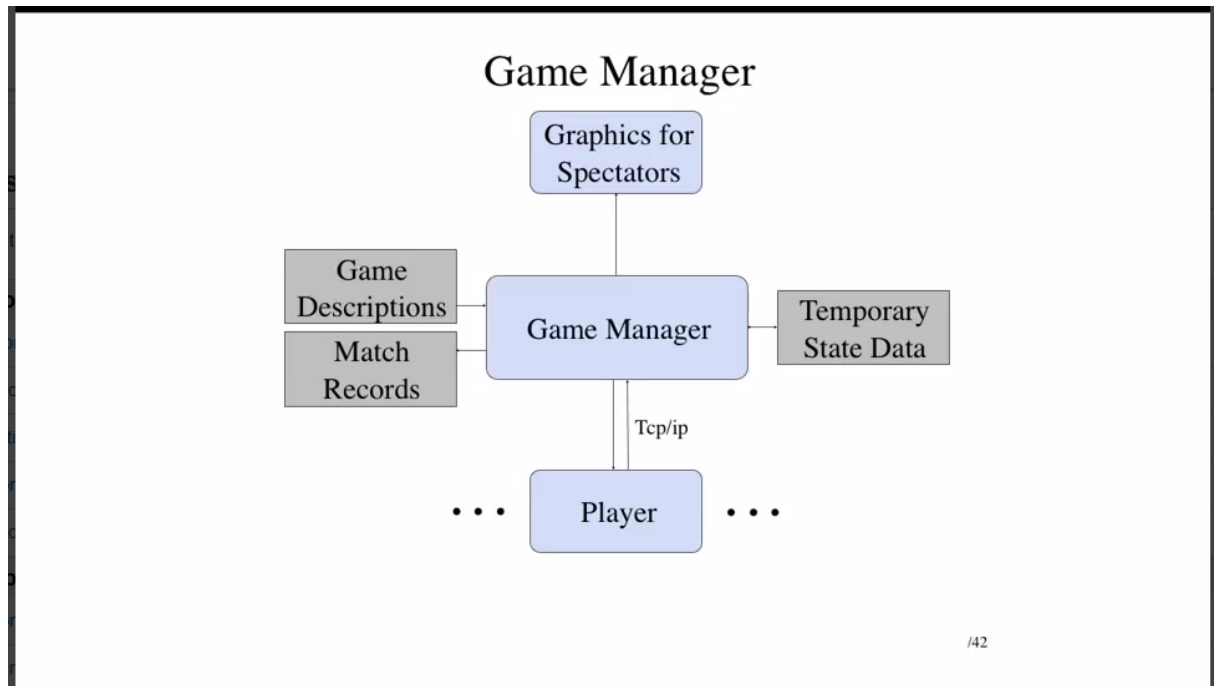
```
next(cell(M,N,x)) <=  
  does(white,mark(M,N)) &  
  true(cell(M,N,b))
```

What the player sees:

```
next(welcoul(M,N,himenoing)) <=  
  does(himenoing,dukepse(M,N)) &  
  true(welcoul(M,N,lorenchise))
```

/42

Game Management



Outlaws

Problems

- Failure to play
- Playing an illegal move

Policy: In such cases, the game manager plays an arbitrary legal move for the player (and reports the move it played in the next Play message).

Reasons:

- Disqualification too severe
- Other players still need to work to win

/42

General Game Playing

Logic

Possible to use logical reasoning for game play

- Computing legality of moves

- Computing consequences of actions

- Computing goal achievement

- Computing termination

Easy to convert from logic to other representations

- many orders or magnitude speedup on simulation

- no asymptotic change

/42

State Representation

X		
	O	
		X

```
true(cell(1,1,x))
true(cell(1,2,b))
true(cell(1,3,b))
true(cell(2,1,b))
true(cell(2,2,o))
true(cell(2,3,b))
true(cell(3,1,b))
true(cell(3,2,b))
true(cell(3,3,x))
true(control(o))
```

/42

Legality Computation

```

legal(W,mark(X,Y)) <=      true(cell(1,1,x))
      true(cell(X,Y,b)) ^   true(cell(1,2,b))
      true(control(W))      true(cell(1,3,b))
                             true(cell(2,1,b))
legal(white,noop) <=      true(cell(2,2,o))
      true(cell(X,Y,b)) ^   true(cell(2,3,b))
      true(control(o))      true(cell(3,1,b))
                             true(cell(3,2,b))
legal(black,noop) <=      true(cell(3,3,x))
      true(cell(X,Y,b)) ^   true(control(o))
      true(control(x))

```

Conclusions:

```

legal(x,noop)              legal(o,mark(1,2))
                             ...
                             legal(o,mark(3,2))

```

/42

Update Computation

```

true(cell(1,1,x))           next(cell(1,1,x))
true(cell(1,2,b))           next(cell(1,2,b))
true(cell(1,3,b))           next(cell(1,3,o))
true(cell(2,1,b))  does(x,noop)  next(cell(2,1,b))
true(cell(2,2,o))  —————> next(cell(2,2,o))
true(cell(2,3,b))  does(o,mark(1,3)) next(cell(2,3,b))
true(cell(3,1,b))           next(cell(3,1,b))
true(cell(3,2,b))           next(cell(3,2,b))
true(cell(3,3,x))           next(cell(3,3,x))
true(control(o))            next(control(x))

```

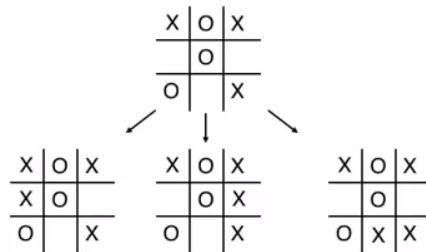
```

next(cell(M,N,o)) :-
    does(o,mark(M,N)) &
    true(cell(M,N,b))

```

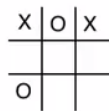
/42

Game Tree Expansion



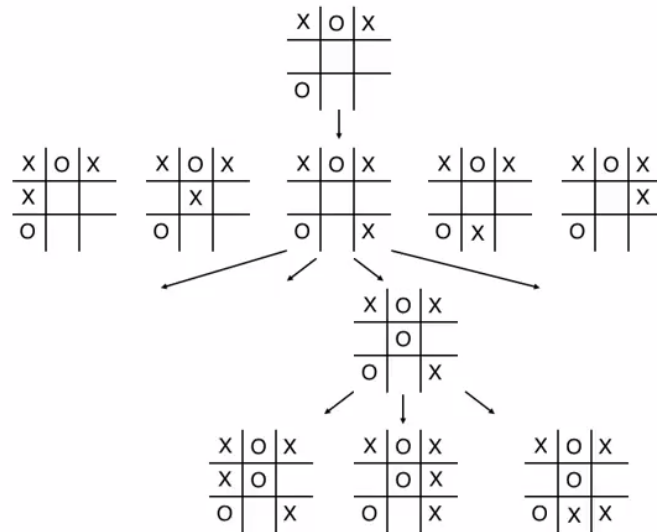
/42

Game Tree Search



/42

Game Tree Search



/42

Resource Limitations

Large state spaces

~5000 states in Tic-Tac-Toe

$>10^{30}$ states in Chess

Limited Resources

Memory

Time (start clock, move clock)

/42

Incremental Search

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.

/42

Non-Guaranteed Evaluation Functions

Ideas

- Goal proximity (everyone)

- Maximize mobility (Barney Pell)

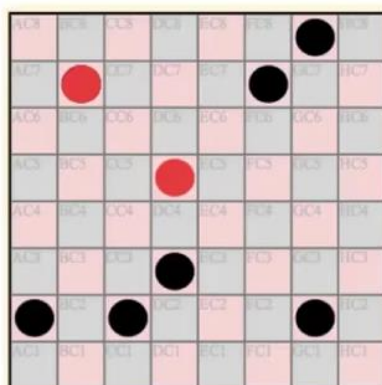
- Minimize opponent's mobility (Jim Clune)

- Depth Charge (aka Monte Carlo)

- <insert your good idea here>

/42

GGP-06 Final - Cylinder Checkers



/42

Guaranteed Evaluation Functions

Ideas

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

/42

Metagaming

Metagaming is the process of analyzing and/or modifying a game in a match-independent fashion so as to improve performance.

Examples of Metagaming Techniques:

Headstart on Game-Graph Search

Game Optimization

Compilation

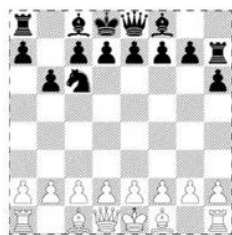
Game Reformulation

Discovery of Game-Specific Evaluation Functions

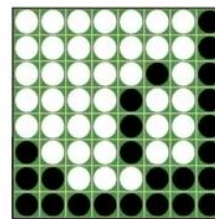
/42

Hodgepodge

Hodgepodge = Chess + Othello



Branching factor: a



Branching factor: b

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: $a^n + b^n$

/42

Game Reformulation

Examples

Factoring, e.g. Hodgepodge

Symmetry, e.g. Tic-Tac-Toe

Bottlenecks, e.g. Triathlon

Techniques:

Graph Analysis (factoring, branching factor, ...)

Proofs (esp those using mathematical induction)

Trade-off - cost of finding structure vs savings

Sometimes cost varies with size of description
rather than size of expanded game graph

/42

Delayed Gratification

X	O	
X	X	O
	O	

/42

Philosophical Remarks

General Game Playing is not a game

Critique: Game playing is *frivolous*.

Applications:

- Enterprise Management
- Computational Law
- Design, Testing, Debugging
- Scientific Theory Formation



/42

Testbed for Theories of Intelligence

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

/42

Tabula Rasa

Critique: Human Intelligence is arguably the product of eons of *evolution*. We are, to some extent at least, “wired” to function well in this world. General Game Players have nothing at their disposal but mathematics.

Reply: A key indicator of intelligence is the ability of each individual to function in radically new environments.

/42

Learning

Critique: *Real* intelligence requires the ability to *figure out* new environments.

Reply: Well, there is no question that that ability is essential. However, real intelligence *also* requires the ability to *use* theories once they are formed.

This is the domain of *general systems* (*universal systems* or, sometimes, *declarative systems*); and interest in this problem dates to the beginning of the field.

/42

John McCarthy

*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.*

/42

Ed Feigenbaum

*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.*

/42

Robert Heinlein

*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.*

/42

Robert Heinlein

computer/robot

*A ~~human-being~~^v 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.*

/42

International GGP Competition

Annual GGP Competition

Annual GGP Competition

Held at AAAI or IJCAI conference

Administered by Stanford University

(Stanford folks not eligible to participate)

/42

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)

/42

GGP-05 Winner Jim Clune



/42

GGP-06 Winners



/42

GGP-07, GGP-08, GGP-12 Winners



/42

Other Games, Other Winners



/42

Format of GGP-2013

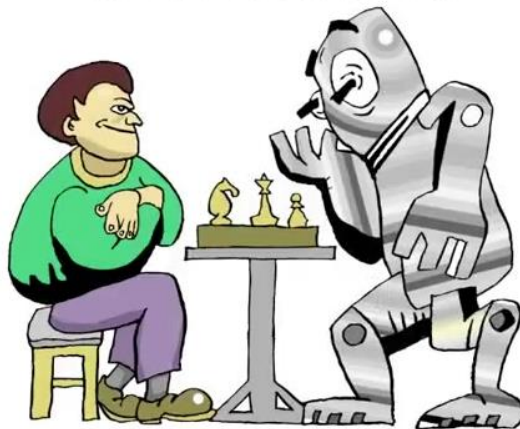
Qualification Round via the web
Tournaments in June

Semi-Final Round at AAAI
Tournament style (one full day)

Final Round at AAAI
Top four competitors
Winner take all (glory + chance to ...)

/42

Carbon versus Silicon



*Come cheer on your favorite.
The battle continues at GGP-2013.*

/42

Human Race Being Defeated



/42

Resources

GGP Websites:

<http://games.stanford.edu>

<http://gamemaster.stanford.edu>

<http://www.ggp.org>

<http://www.general-game-playing.de>

http://en.wikipedia.org/wiki/General_game_playing

Google “general game playing”

Courses:

Coursera course in Spring 2013

<http://class.coursera.org/ggp-001/>

/42



12