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

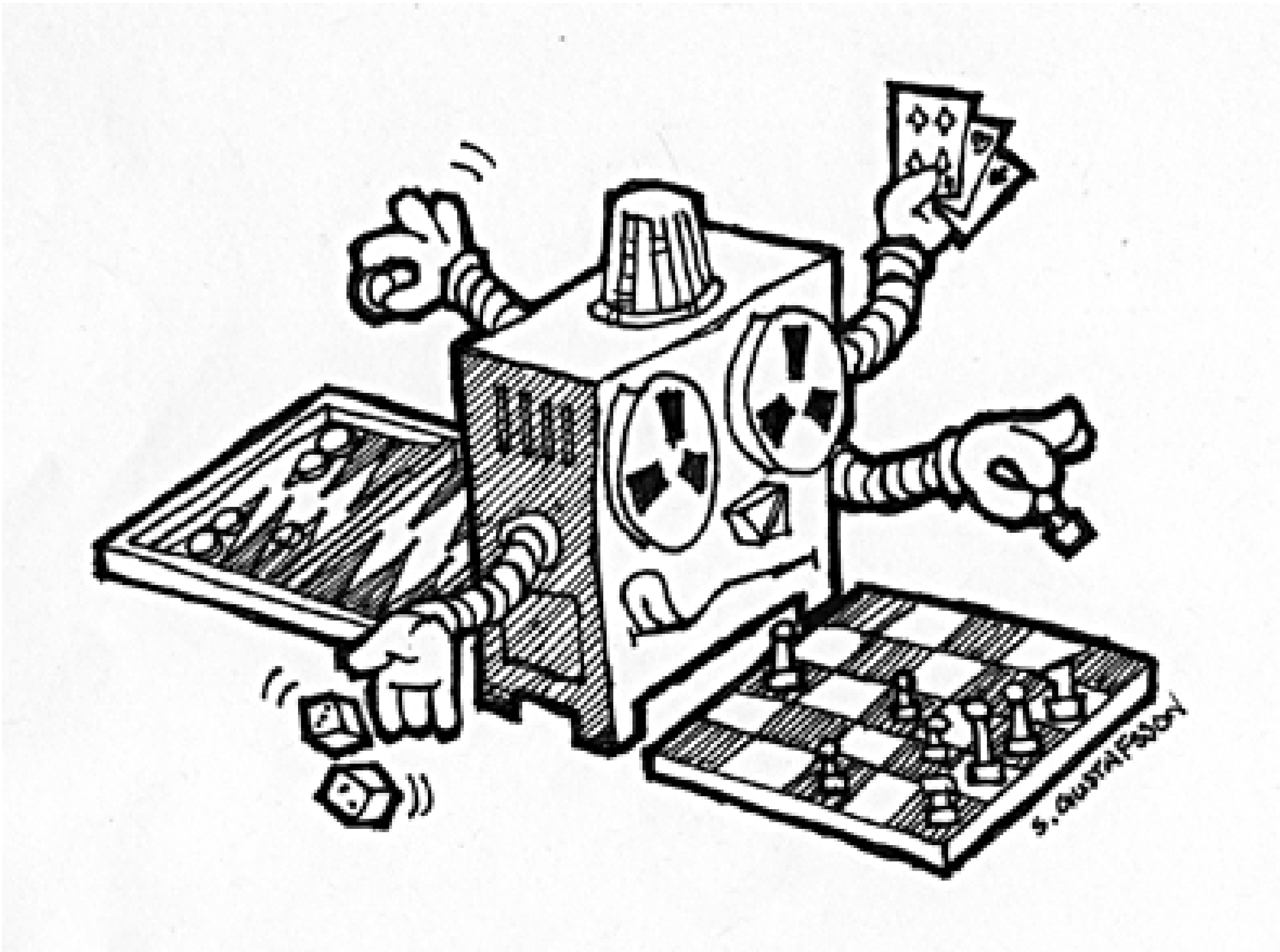
General Game Playing

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.

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

Annual GGP Competition

- Held at AAAI or IJCAI conference
- Administered by Stanford University
- (Stanford folks not eligible to participate)

History

Winners

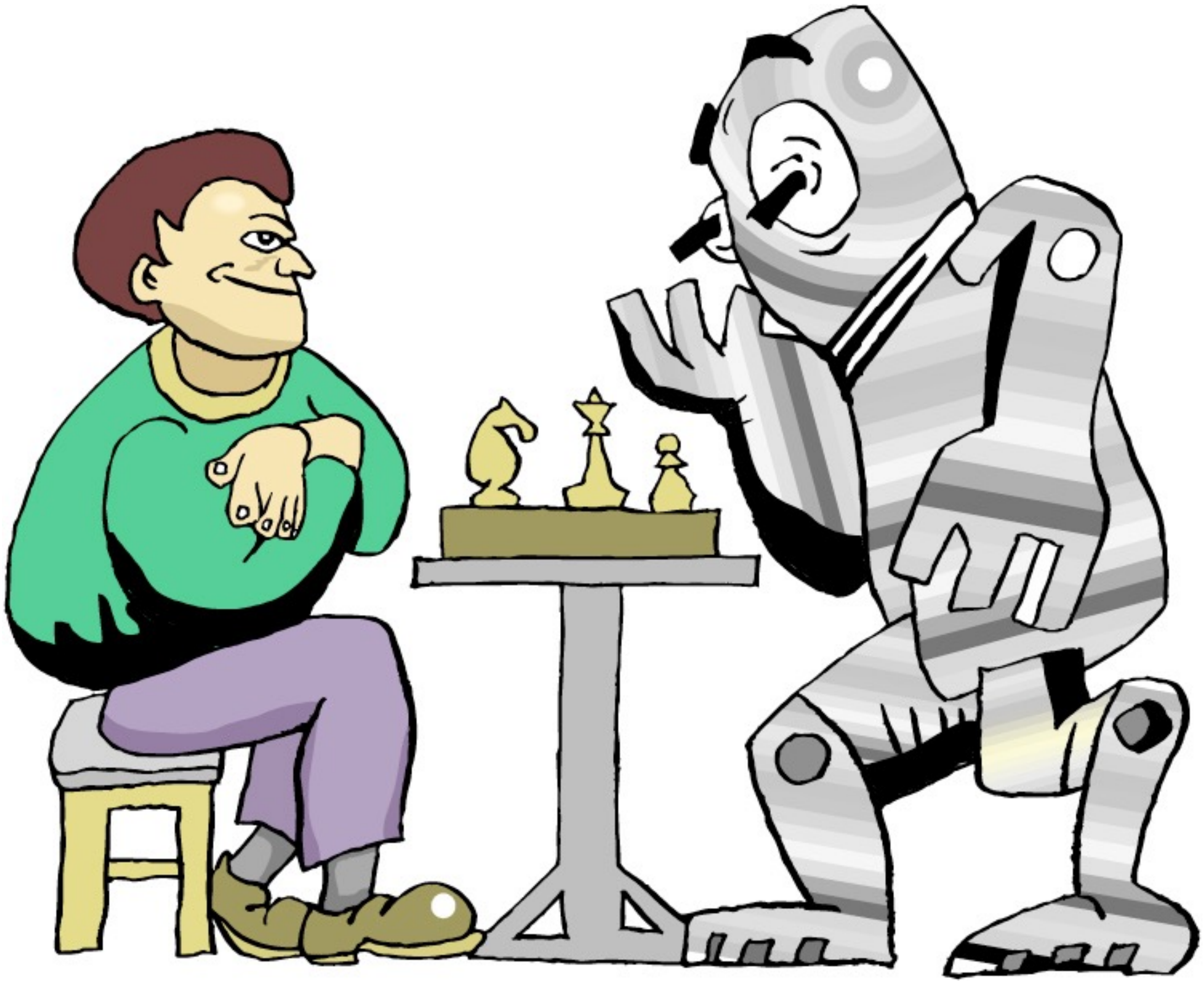
- 2005 - ClunePlayer - Jim Clune (USA)
- 2006 - FluxPlayer - Schiffel, Thielscher (Germany)
- 2007 - CadiaPlayer - Bjornsson, Finsson (Iceland)
- 2008 - CadiaPlayer - Bjornsson, Finsson (Iceland)
- 2010 - Ary - Mehat (France)
- 2011 - TurboTurtle - Schreiber (USA)
- 2012 - CadiaPlayer - Bjornsson, Finsson (Iceland)
- 2013 - TurboTurtle - Schreiber (USA)
- 2014 - Sancho - Draper (USA), Rose (UK)
- 2015 - Galvanise - Emslie
- 2016 - WoodStock - Piette (France)

GGP-05 Winner Jim Clune

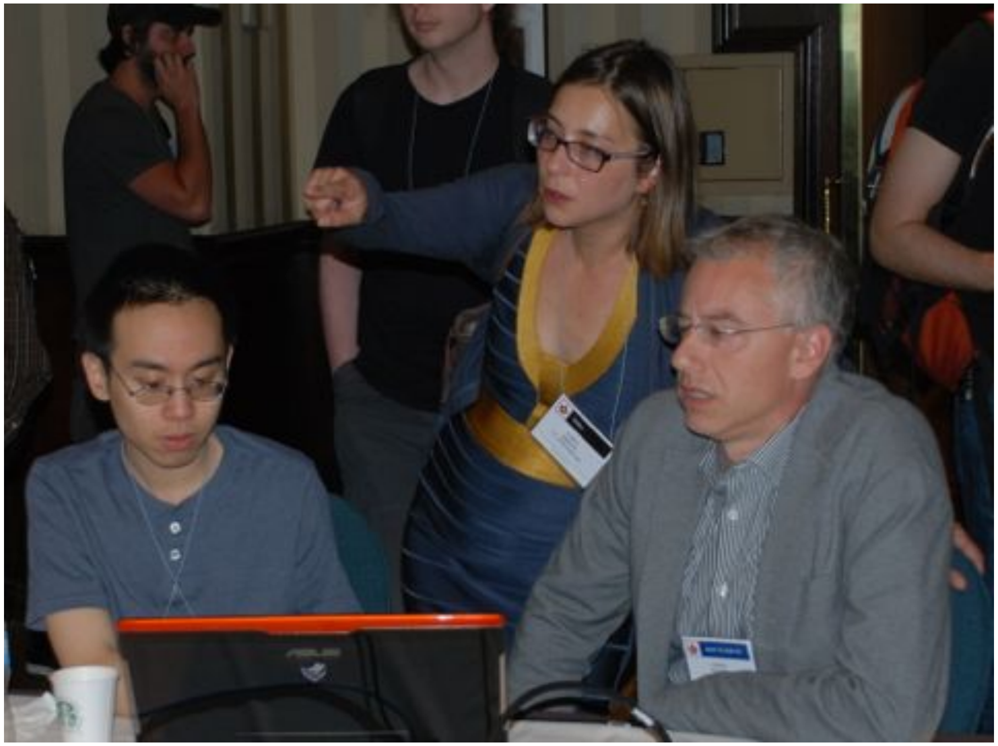
International GGP Competition

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

Carbon versus Silicon

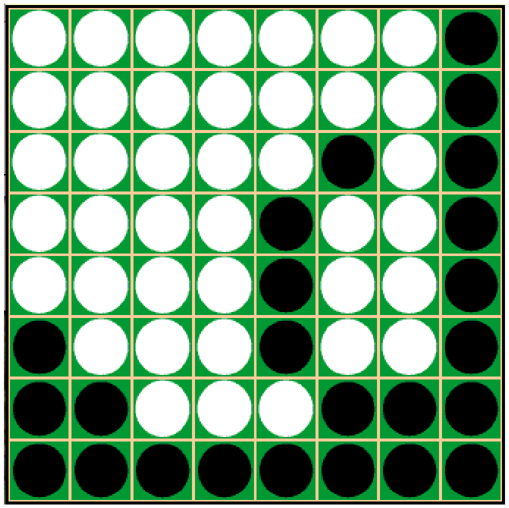


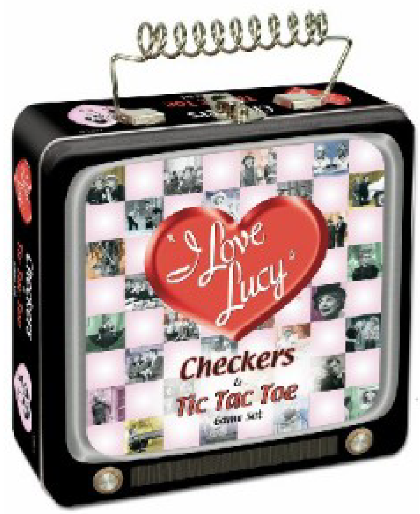
Human Race Being Defeated



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

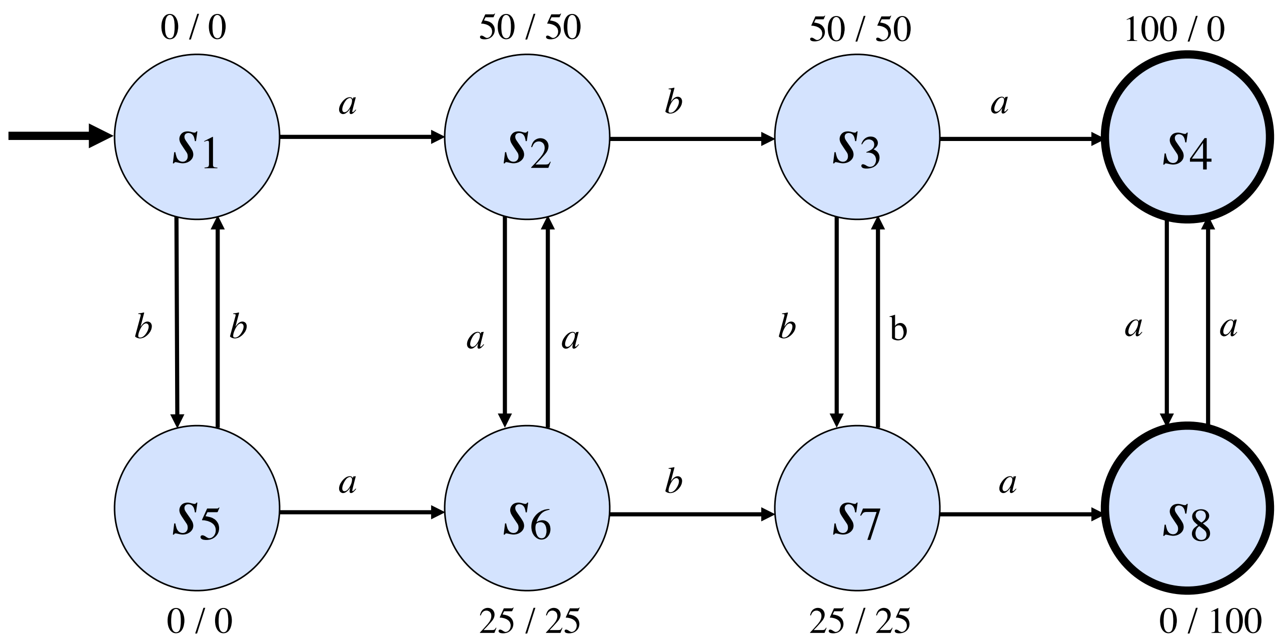
Players

- 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



Direct Description

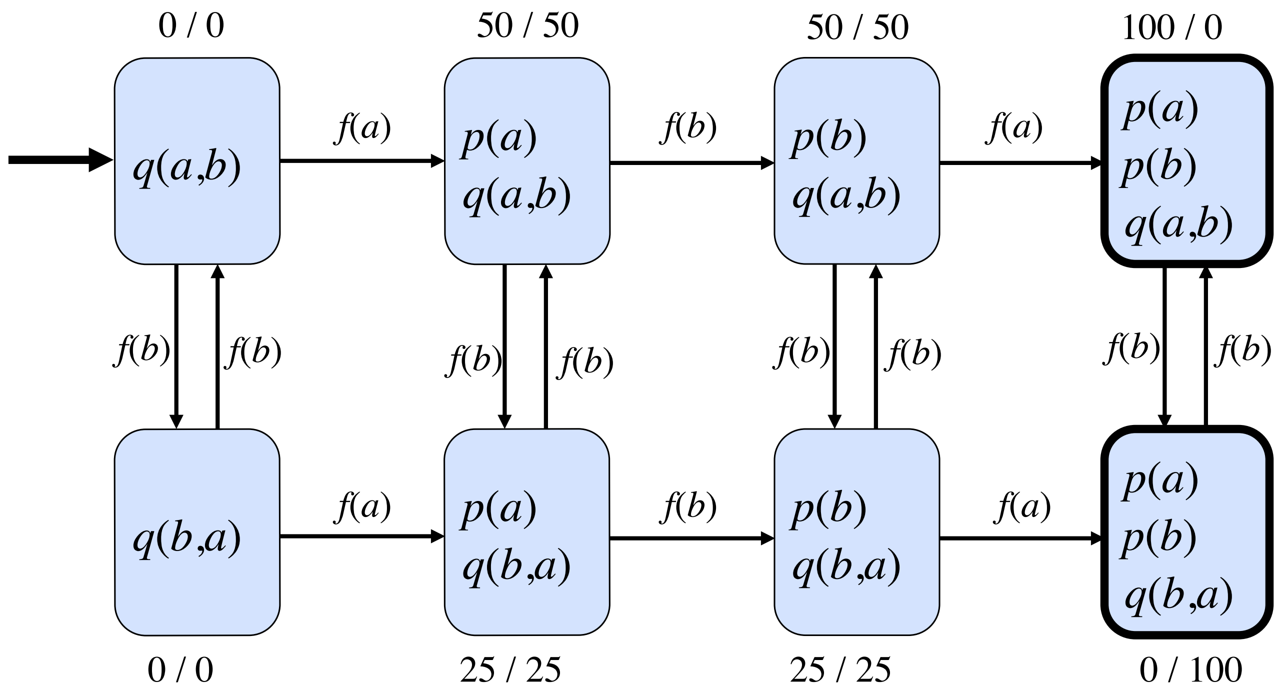
Good News: Since all of the games that we are considering are finite, it is possible in principle to communicate game information in the form of state graphs.

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

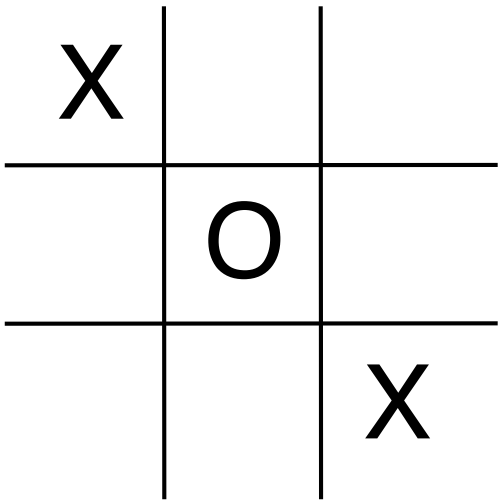
Solution:

Exploit regularities / structure in state graphs to produce compact encoding

Structured State Machine



States




```
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)
```

Actions

```
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)
```

→ mark(1,3) →

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

Game Description Language

```
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 :- ~open

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

goal(o,100) :- line(o)
goal(o,50) :- draw
goal(o,0) :- line(x)
```

```
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))

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

open :- true(cell(M,N,b))

draw :- ~line(x) &
        ~line(o)
```

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))
```

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)
```

Legal 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)
```

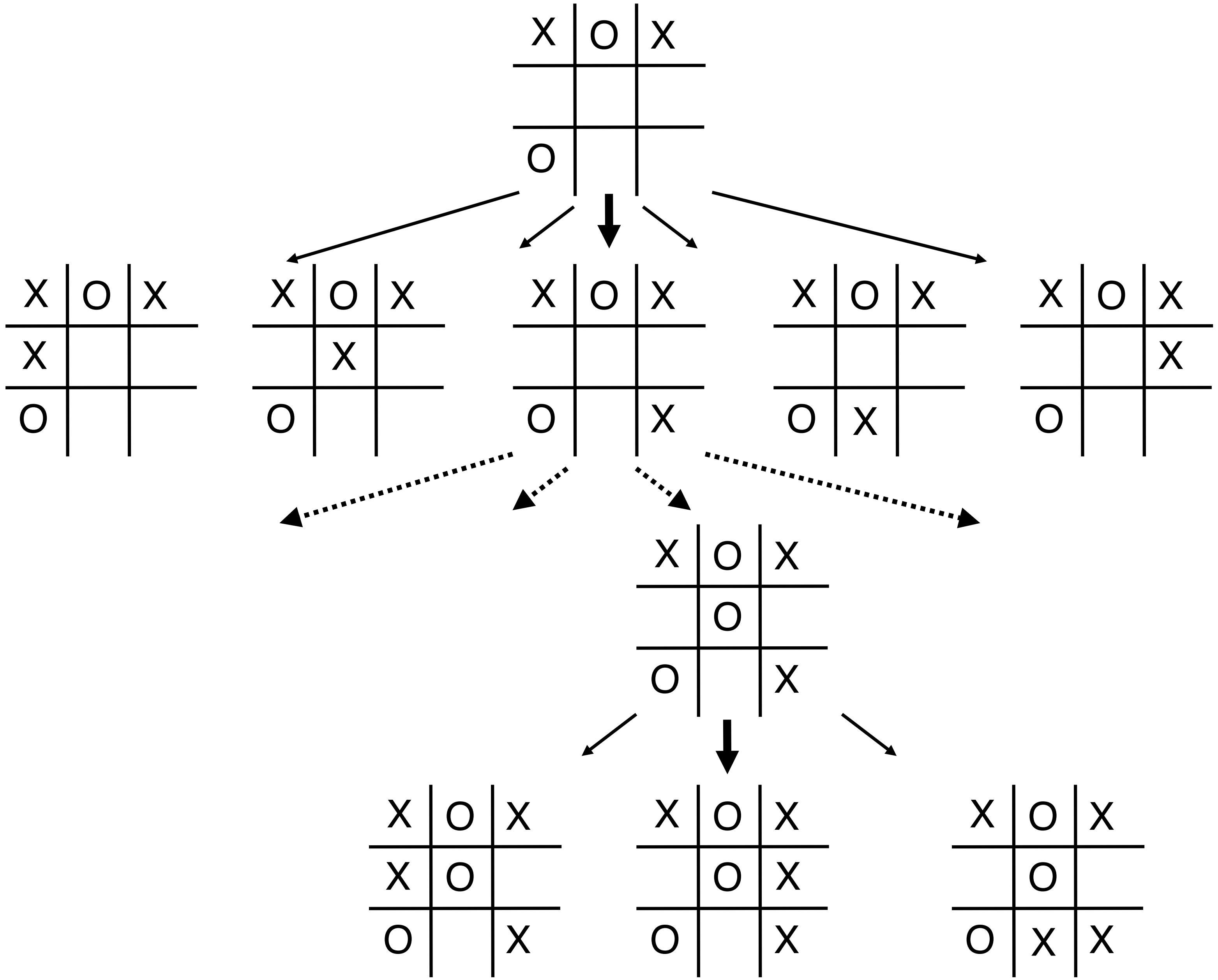
State Update

```
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)
```

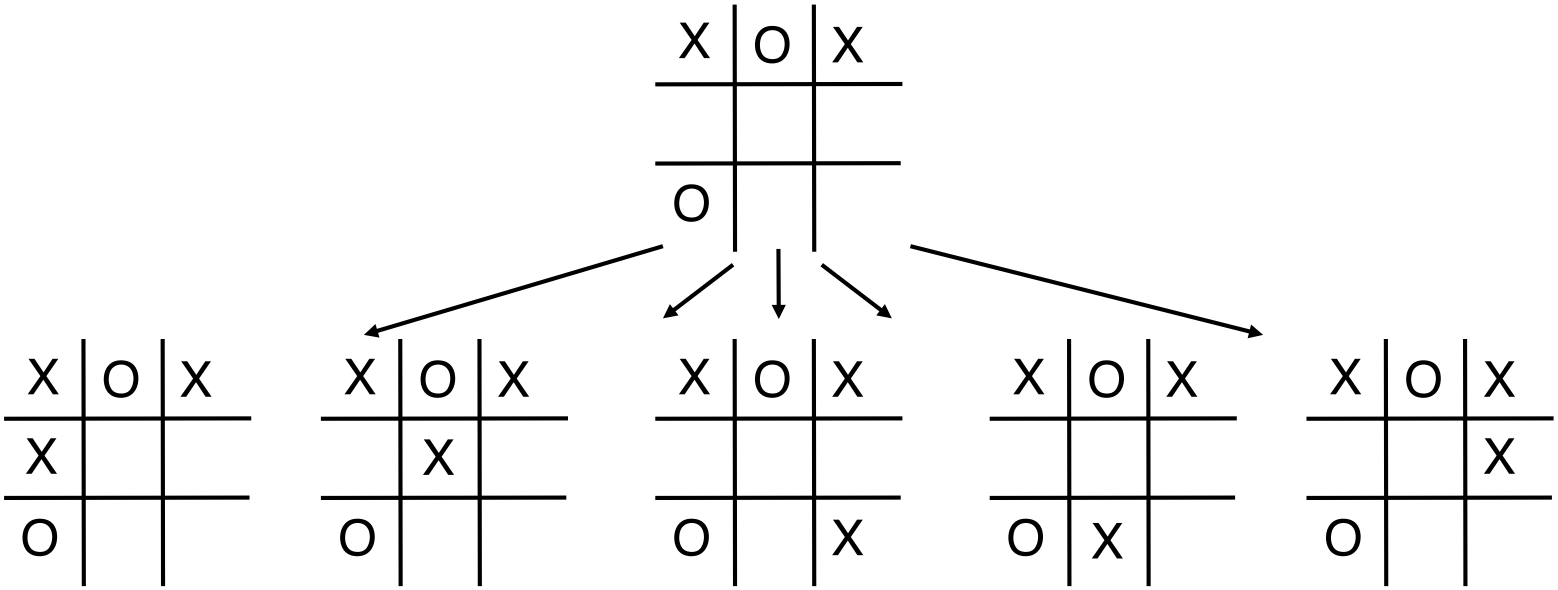
mark(1,3) →

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

Complete Game Graph Search



Incomplete Game Tree Search



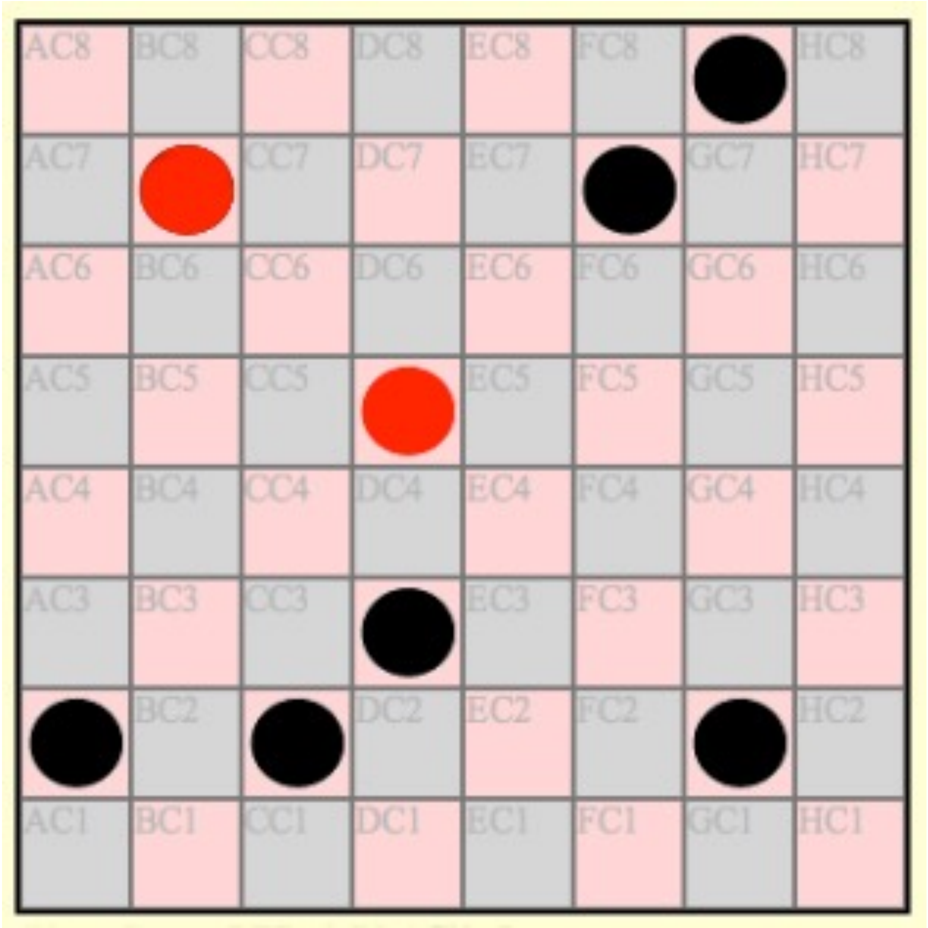
How do we evaluate non-terminal states?

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



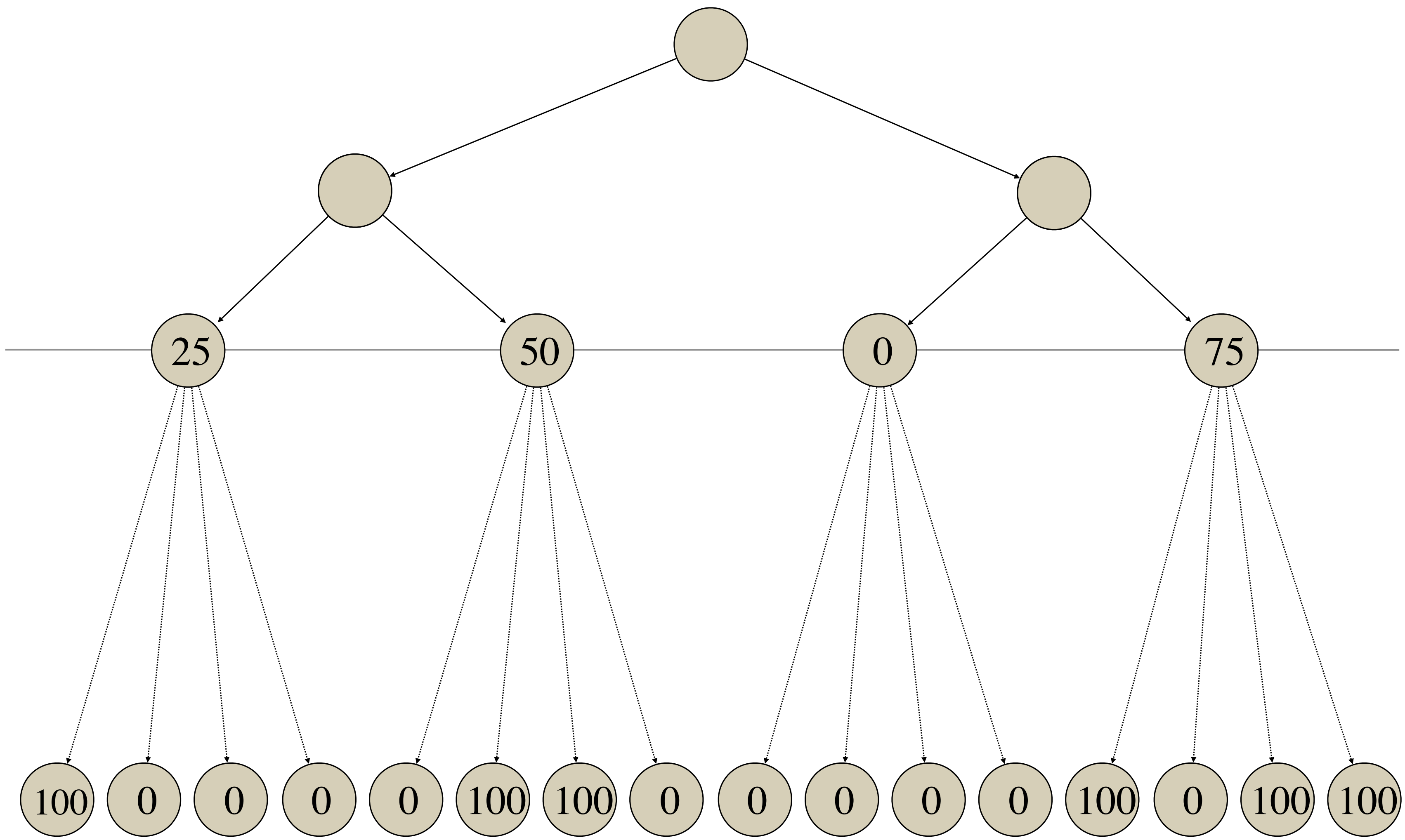
Second Generation GGP (2007 on)

Monte Carlo Search

Monte Carlo Tree Search
  UCT - Uniform Confidence Bounds on Trees

Second Generation GGP

Monte Carlo Search



Third Generation GGP

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

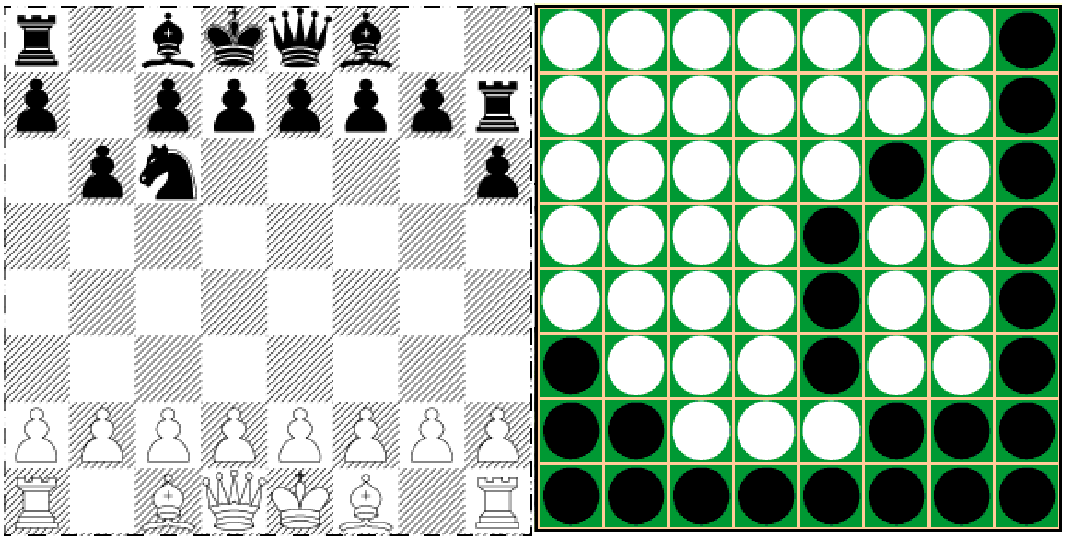
- Simple, widely published algorithms
- several orders or magnitude speedup
- no asymptotic change

Conversion to Field Programmable Gate Arrays (FPGAs)

- several more orders of magnitude improvement

Game Factoring

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

Reformulation Opportunities

Examples

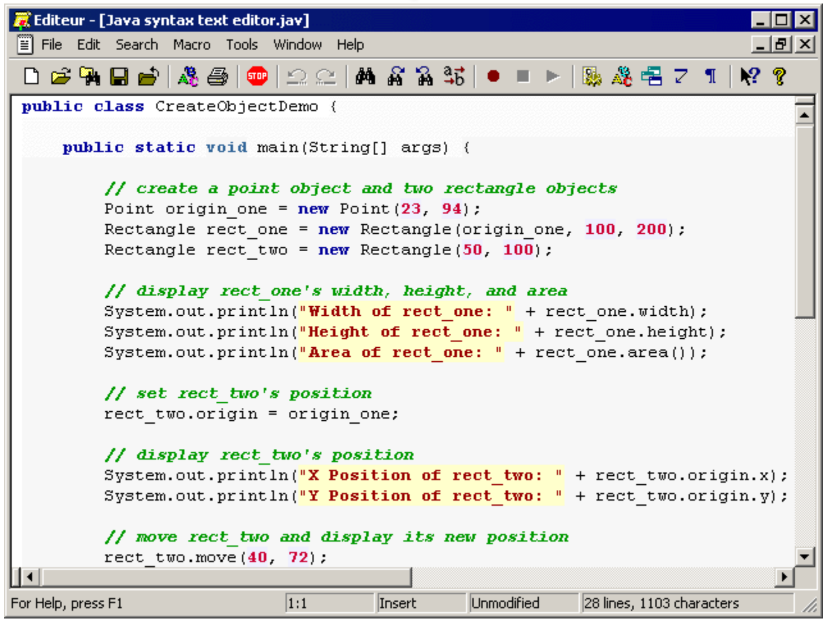
- Factoring, e.g Hodgepodge
- Bottlenecks, e.g. Triathalon
- Symmetry detection, e.g. Tic-Tac-Toe
- Dead State Removal

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

```
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 :- ~open

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

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

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))

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

open :- true(cell(M,N,b))

draw :- ~line(x) &
        ~line(o)
```



Algorithmic Expertise

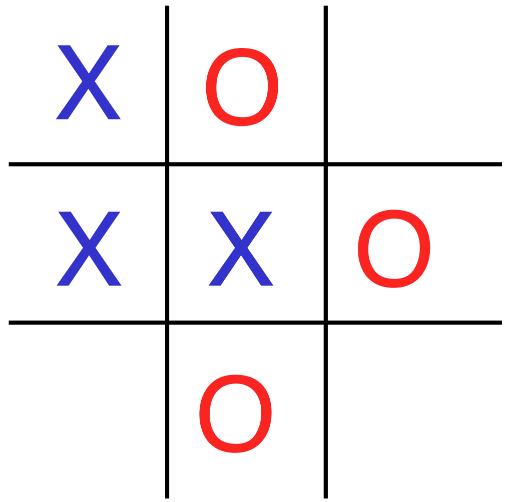
Knuth in a Box

Game Theory

	a	b
a	4, 4	3, 3
b	2, 2	1, 1

	a	b
a	4, 4	1, 1
b	3, 3	2, 2

Psychology



Demoralizing the Opponent
Fooling the Opponent

Conclusion

General Game Playing is not a game

Serious Business

Theory of Intelligence

Dimensions of Intelligence

Representation of the World

Correct and efficient reasoning

Rationality with incomplete info and resource bounds

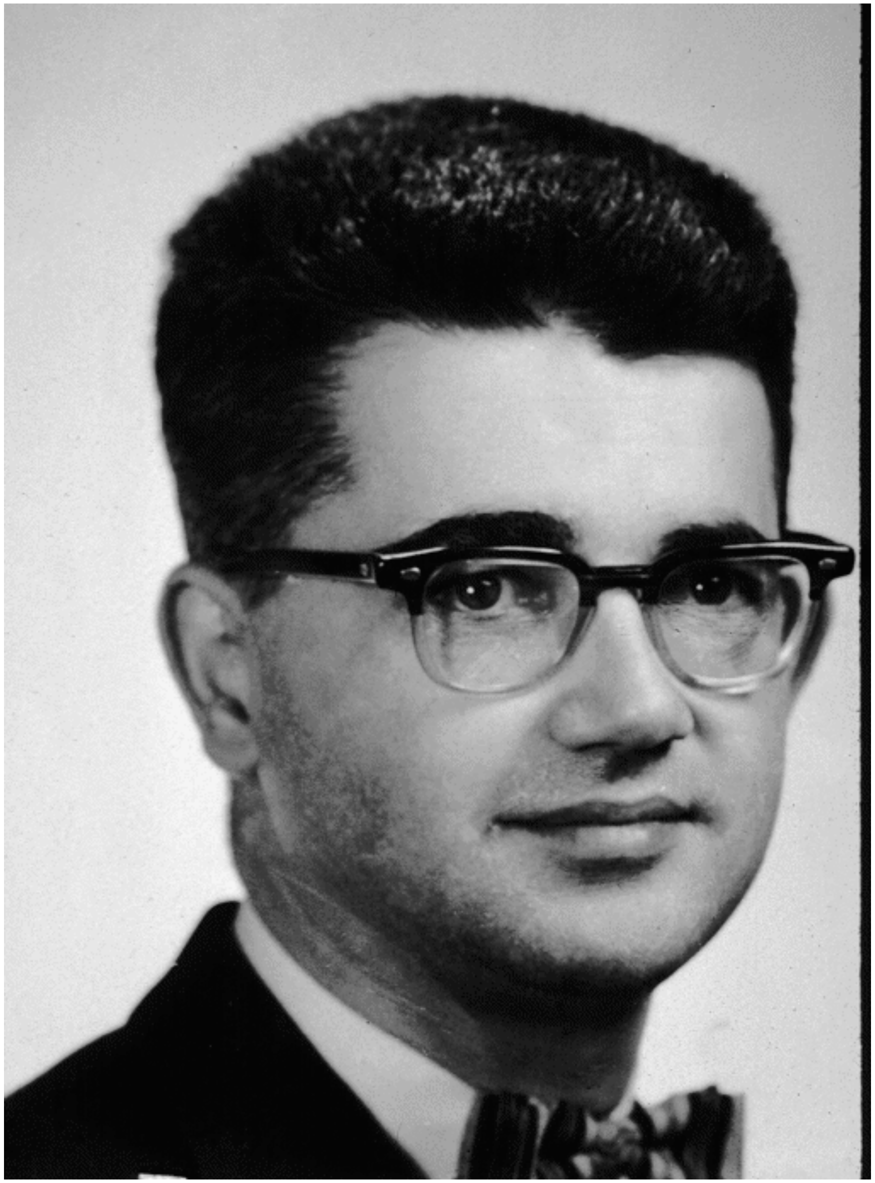
Generality

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*

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.*

- John McCarthy1958

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.*

- Ed Feigenbaum 1974

Newell and Simon

*The **General Problem Solver** demonstrates how generality can be achieved by factoring the specific descriptions of individual tasks from the task-independent processes.*

Robert Heinlein

computer/robot

v

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

Schedule

April	3	Introduction	← You are here.
	10	Game Description	
	17	Game Playing	
	24	Incomplete Search	
May	1	Statistical Search	
	8	Logical Optimization	
	15	Materialization and Reformulation	
	22	Game Tree Reformulation, e.g. Factoring	
	29	*Really* General Game Playing	
June	5	Final Competition	

Teams

Composition

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

Names:	Identifiers:
Pansy Division	pansy_division
The Pumamen	punamen
Team Camembert	camembert
Mighty Bourgeoisie	bourgeosie
Greedy Bastards	greedybastards
Red Hot Chili Peppers	peppers
Michael Genesereth	michael_genesereth
/*^*\	happy
X Æ A-12	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

Grades

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


GENERAL
GAME
PLAYING