

# General Game Playing

## *Future of GGP*

Michael Genesereth  
Logic Group  
Stanford University

# Metagaming

*Metagaming* is match-independent game processing, i.e. game processing that is done independent of any particular opponent or any particular state.

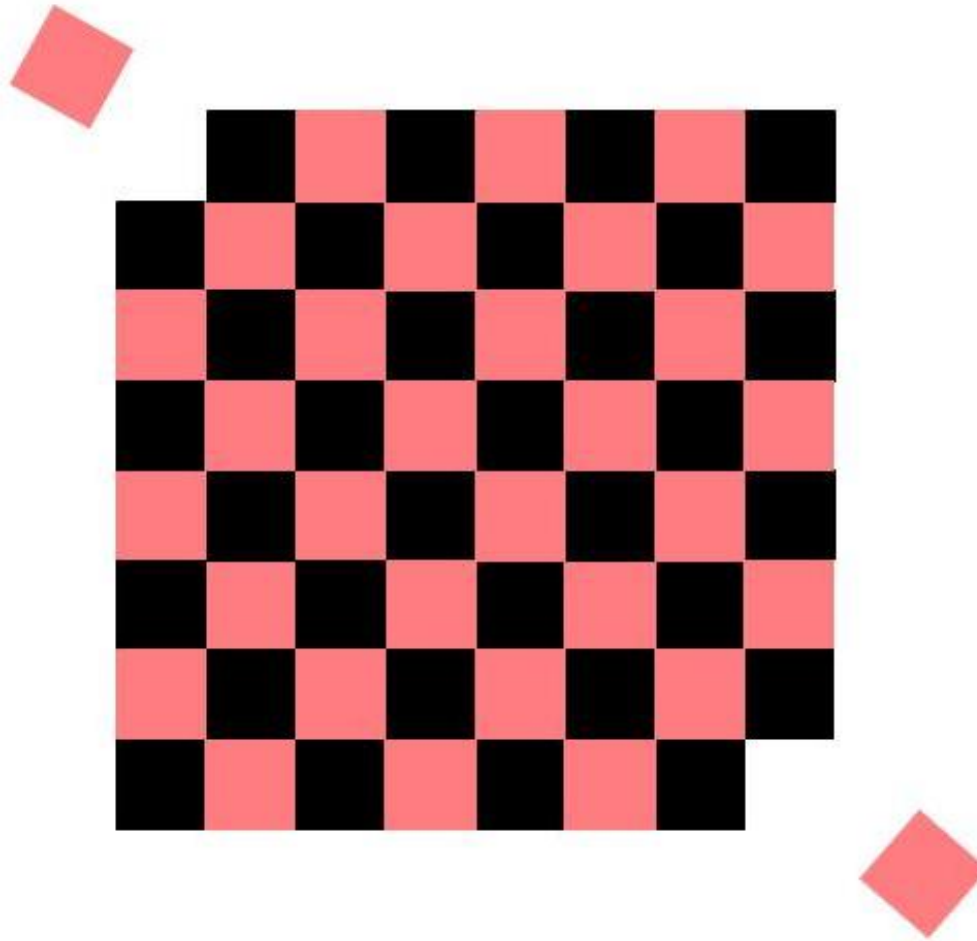
Techniques:

- ✓ Headstart, big switch, etc.
- ✓ Optimization - subgoal and rule ordering, removal
- ✓ Grounding and Symbolizing and Simplifying
- ✓ Game Reformulation

Symbolic Reasoning


Automatic Programming

# Mutilated Checkerboard



# Connect Four Suicide

Not Secure — gamemaster.stanford.edu



## Gamemaster

[Sign In](#)

Protocol: standalone  
Game: connectfour 🗡️

			Red				
		Blue	Red	Blue			
		Red	Blue	Red			

Move:

# 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,HOOP) :-
  true(control(o))

legal(O,HOOP) :-
  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)
```



```
Editeur - [Java syntax text editor.jav]
File Edit Search Macro Tools Window Help
[Icons]

public class CreateObjectDemo {

    public static void main(String[] args) {

        // create a point object and two rectangle objects
        Point origin_one = new Point(23, 94);
        Rectangle rect_one = new Rectangle(origin_one, 100, 200);
        Rectangle rect_two = new Rectangle(50, 100);

        // display rect_one's width, height, and area
        System.out.println("Width of rect_one: " + rect_one.width);
        System.out.println("Height of rect_one: " + rect_one.height);
        System.out.println("Area of rect_one: " + rect_one.area());

        // set rect_two's position
        rect_two.origin = origin_one;

        // display rect_two's position
        System.out.println("X Position of rect_two: " + rect_two.origin.x);
        System.out.println("Y Position of rect_two: " + rect_two.origin.y);

        // move rect_two and display its new position
        rect_two.move(40, 72);
    }
}
```

For Help, press F1 | 1:1 | Insert | Unmodified | 28 lines, 1103 characters

# Algorithmic Expertise



Knuth in a Box; Schaeffer in a Box

# Games with Incomplete Information

# Complete and Incomplete Information

## "Complete" Information Games

Initial State

Legal moves of player in control in all states

Move of each player on each step

State resulting from joint moves of players

Rewards of every player in every state

Termination

## Incomplete Information Games

Incomplete information about initial state

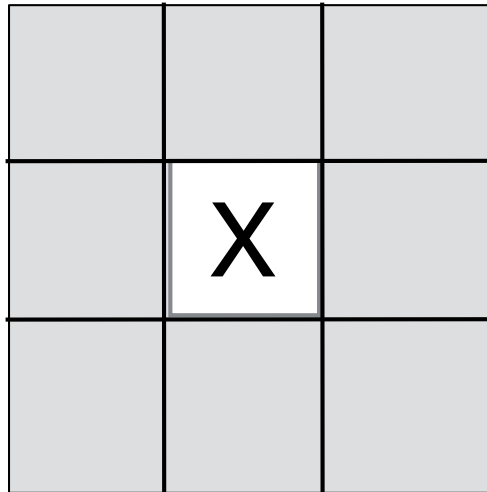
Rules may not be complete (non-determinism)

May not even see the moves of others

# Battleship



# Blind Tic Tac Toe



# State Set for Blind Tic Tac Toe

O		
	X	

	O	
	X	

		O
	X	

O	X	

	X	O

	X	
O		

	X	
	O	

	X	
		O

# GDL-II

“Percepts”

ok

ng

Percepts provided by game manager:

`play(ok)`

# State Set if We See ok

O	X	
	X	

	X	O
	X	

	X	
O	X	

	X	
	X	O

	X	
	X	
O		

	X	
	X	
	O	

	X	
	X	
		O

# State Set if We See ng

	O	
	X	

# Inductive General Game Playing

In traditional GGP, players are given rules:

```
legal(move(X,Y)) :-
    location(robot,X) & left(X,Y)
```

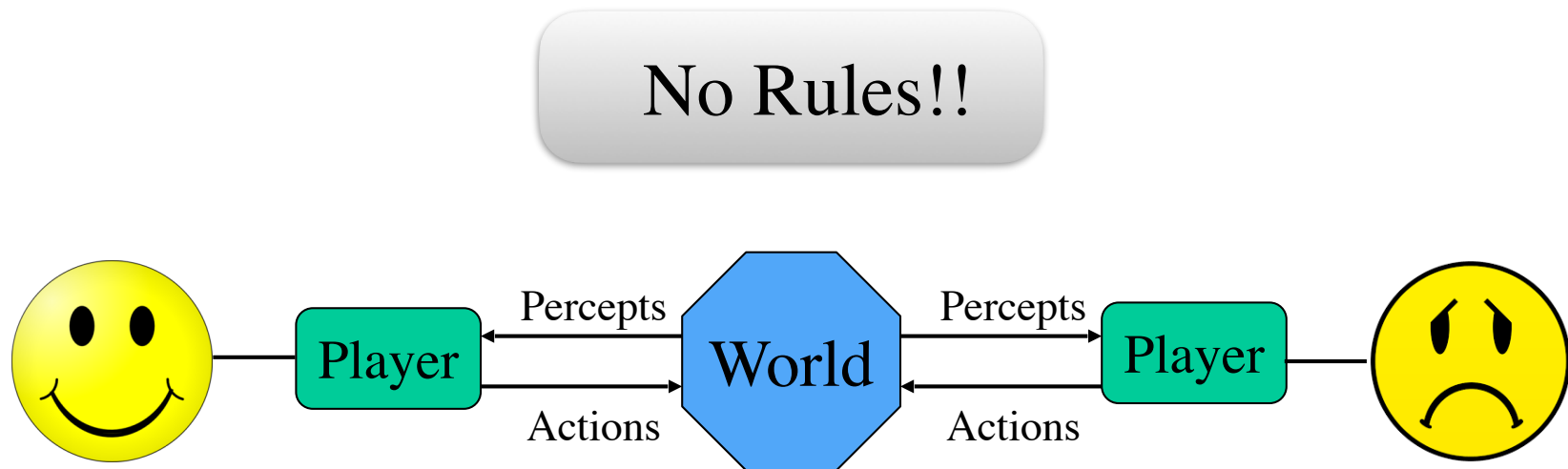
In IGGP, players are given histories:

	white	black			
1	noop	mark(1,1)			
2		white	black		
3	1	noop	mark(1,1)		
4	2		white	black	
5	3	1	noop	mark(1,1)	
6	4	2		white	black
7	5	3	1	noop	mark(1,1)
8	6	4	2	mark(3,3)	noop
9	7	5	3	noop	mark(1,2)
	8	6	4	mark(2,2)	noop
	9	7	5	noop	mark(2,1)
		8	6	mark(1,3)	noop
		9	7	noop	mark(2,3)
			8	mark(3,1)	noop
			9	<b>100</b>	<b>0</b>

Statistical Learning

# Really General Game Playing

Percept-Action and Utility Model for Players



## Discussion

A "bit" more difficult

Inferring hidden state

Deep Learning and Reinforcement Learning

# Competition Strategies

# Most Important Suggestion

**Make sure it works.**

# Second Most Important Suggestion

**Make sure it works.**

# Third Most Important Suggestion

**Make sure it works.**

## Fourth Most Important Suggestion

**Make sure it works.**



**GENERAL  
GAME  
PLAYING**



Celebrating AAI's 25th Anniversary



