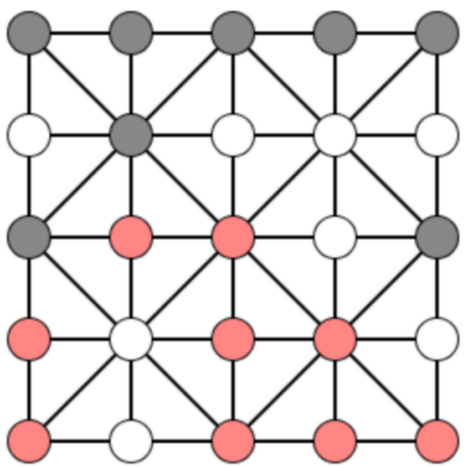
Inductive general game playing

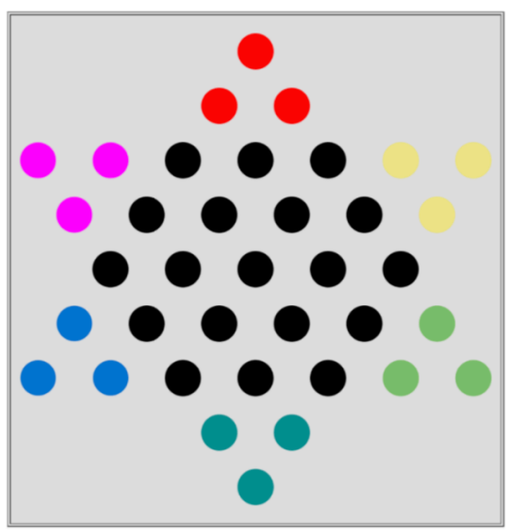
Andrew Cropper, Richard Evans, and Mark Law

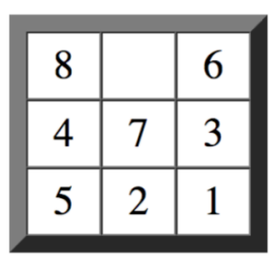
Inductive general game ~~playing~~

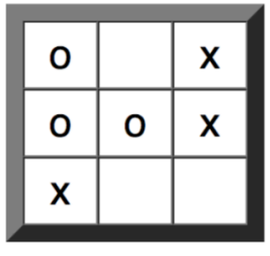
Andrew Cropper, Richard Evans, and Mark Law

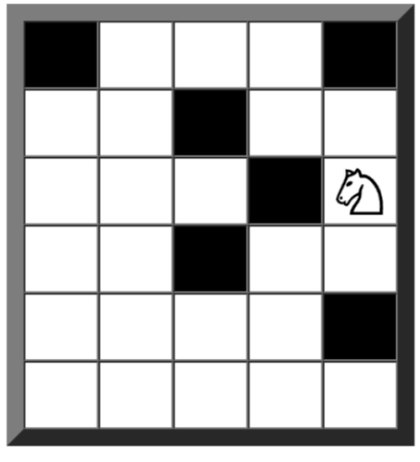
General game playing competition

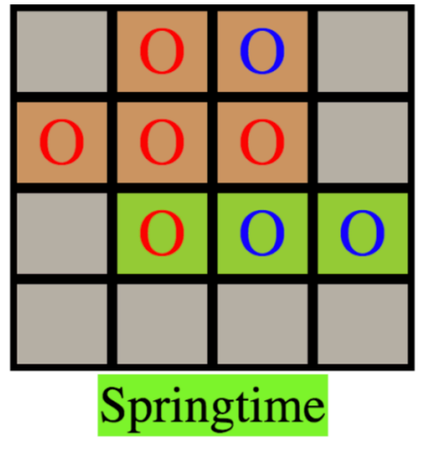












Game description language

- initial game state
- legal moves
- how moves update the game state
- how the game terminates

Game description language

```
(succ 0 1)
(succ 1 2)
(succ 2 3)
(beats scissors paper)
(beats paper stone)
(beats stone scissors)
(<= (next (step ?n)) (true (step ?m)) (succ ?m ?n))
(<= (next (score ?p ?n)) (true (score ?p ?n)) (draws ?p))
(<= (next (score ?p ?n)) (true (score ?p ?n)) (loses ?p))
(<= (next (score ?p ?n)) (true (score ?p ?n2)) (succ ?n2 ?n) (wins ?p))
(<= (draws ?p) (does ?p ?a) (does ?q ?a) (distinct ?p ?q))
(<= (wins ?p) (does ?p ?a1) (does ?q ?a2) (distinct ?p ?q) (beats ?a1 ?a2))
(<= (loses ?p) (does ?p ?a1) (does ?q ?a2) (distinct ?p ?q) (beats ?a2 ?a1))
```

Our problem

Learn rules from observations

- goal
- legal
- next
- terminal

Why?

- Diverse games
- New problems each year
- Independent language
- *Hard problems?*

Capablanca

Rock, paper, scissors

```
% BK
beats(paper,stone).
beats(scissors,paper).
beats(stone,scissors).
player(p1).
player(p2).
succ(0,1).
succ(1,2).
succ(2,3).
does(p1,stone).
does(p2,paper).
true_score(p1,0).
true_score(p2,0).
true_step(0).
```

```
% E+
next_score(p1,0).
next_score(p2,1).
next_step(1).

% E-
next_score(p2,0).
next_score(p1,1).
next_score(p1,2).
next_score(p2,2).
next_score(p1,3).
next_score(p2,3).
next_step(0).
next_step(2).
next_step(3).
```

Rock, paper, scissors

```
next_step(N):-
    true_step(M),
    succ(M,N).
next_score(P,N):-
    true_score(P,N),
    draws(P).
next_score(P,N):-
    true_score(P,N),
    loses(P).
next_score(P,N2):-
    true_score(P,N1),
    succ(N2,N1),
    wins(P).
```

```
draws(P):-
    does(P,A),
    does(Q,A),
    distinct(P,Q).
loses(P):-
    does(P,A1),
    does(Q,A2),
    distinct(P,Q),
    beats(A2,A1).
wins(P):-
    does(P,A1),
    does(Q,A2),
    distinct(P,Q),
    beats(A1,A2).
```

Game	R	L	D	P
Minimal Decay	2	6	0	1
Minimal Even	8	19	0	1
Rainbow	10	48	0	1
Rock Paper Scissors	12	36	0	1
GT Chicken	16	78	0	2
GT Attrition	16	60	0	2
Coins	16	45	0	1
Buttons and Lights	16	44	1	1
Leafy	17	80	2	2
GT Prisoner	17	75	0	2
Eight Puzzle	17	60	2	1
Lightboard	18	69	2	2
Knights Tour	18	46	2	1
Sukoshi	19	49	1	2
Walkabout	22	66	2	2
Horseshoe	22	59	2	2
GT Ultimatum	22	67	0	2
Tron	23	76	2	2
9x Buttons and Lights	24	77	2	1
Hunter	24	69	2	1
GT Centipede	24	69	0	2
Fizz Buzz	25	74	0	1
Untwisty Corridor	27	68	0	1
Don’t Touch	29	84	2	2
Tiger vs Dogs	30	88	2	2

Game	R	L	D	P
Sheep and Wolf	30	89	2	2
Duikoshi	31	76	2	2
TicTacToe	32	92	2	2
HexForThree	35	130	2	3
Connect 4	36	124	2	4
Breakthrough	36	126	2	2
Centipede	37	134	2	1
Forager	40	106	2	1
Sudoku	41	101	2	1
Sokoban	41	172	2	1
9x TicTacToe	42	149	2	2
Switches	44	183	2	1
Battle of Numbers	44	134	2	2
Free For All	46	130	2	2
Alquerque	49	134	2	2
Kono	50	134	2	2
Checkers	52	167	2	2
Pentago	53	188	2	2
Platform Jumpers	62	168	2	2
Pilgrimage	80	240	2	2
Firesheep	85	290	2	2
Farming Quandries	88	451	2	2
TTCC4	94	301	2	2
Frogs and Toads	97	431	2	2
Asylum	101	273	2	2

Fizzbuzz BK

```
divisible(12,1).
divisible(12,2).
...
divisible(12,12).
input_say(player,1).
input_say(player,2).
...
input_say(player,30).
input_say(player,fizz).
input_say(player,buzz).
input_say(player,fizzbuzz).
role(player).
int(0).
int(1).
...
int(31).
```

```
less_than(0,1).
less_than(0,2).
...
less_than(30, 31).
minus(1,1,0).
minus(2,1,1).
...
minus(31,31,0).
positive_int(1).
positive_int(2).
...
positive_int(31).
succ(0,1).
succ(0,2).
...
succ(30,31).
```

Fizzbuzz legal

```
% BK
true_count(9).
true_success(6).

% E+
legal_say(player,9)
legal_say(player,buzz)
legal_say(player,fizz)
legal_say(player,fizzbuzz)

% E-
legal_say(player,0).
legal_say(player,1).
...
legal_say(player,8).
legal_say(player,10).
...
legal_say(player,31).
```

Fizzbuzz legal

```
% BK
true_count(9).
true_success(6).

% E+
legal_say(player,9)
legal_say(player,buzz)
legal_say(player,fizz)
legal_say(player,fizzbuzz)

% E-
legal_say(player,0).
legal_say(player,1).
...
legal_say(player,8).
legal_say(player,10).
...
legal_say(player,31).
```

```
% Hypothesis
legal_say(player,N):-
    true_count(N).
legal_say(player,fizz).
legal_say(player,buzz).
legal_say(player,fizzbuzz).
```

Fizzbuzz next count

```
% BK
does_say(player,buzz).
true_count(12).

% E+
next_count(13).

% E-
next_count(0).
next_count(1).
...
next_count(12).
next_count(14).
...
next_count(31).
```

Fizzbuzz next count

```
% BK
does_say(player,buzz).
true_count(12).

% E+
next_count(13).
```

```
% hypothesis
next_count(After):-
    true_count(Before),
    succ(Before,after).
```

```
% E-
next_count(0).
next_count(1).
...
next_count(12).
next_count(14).
...
next_count(31).
```

Fizzbuzz next success

```
% BK
does_say(player,buzz).
true_success(3).

% E+
next_success(3).
```

```
% E-
next_success(0).
next_success(1).
next_success(2).
next_success(4).
...
next_success(31).
```

Fizzbuzz next success

```
next_success(After):-
    correct,
    true_success(Before),
    succ(Before,After).

next_success(A):-
    \+ correct,
    true_success(A).

correct:-
    true_count(N),
    \+ divisible(N,5),
    \+ divisible(N,3),
    does_player_say(N).
```

```
correct:-
    true_count(N),
    divisible(N,15),
    does_player_say(fizzbuzz).
correct:-
    true_count(N),
    divisible(N,3),
    \+ divisible(N,5),
    does_player_say(fizz).
correct:-
    true_count(N),
    divisible(N,5),
    \+ divisible(N,3),
    does_player_say(buzz).
```

Results

Metric	Baseline	Inertia	Mean	KNN_1	KNN_5	Aleph	ASPAL	Metagol	ILASP*
BA (%)	48	56	64	80	80	66	55	69	**86**
PS (%)	4	4	15	16	19	18	10	34	**40**

Results

Metric	Aleph	ASPAL	Metagol	ILASP*
BA (%)	66	55	69	**86**
PS (%)	18	10	34	**40**

Results balanced accuracy

Approach	goal	legal	next	terminal
True	47	56	47	42
Inertia	47	56	80	42
Mean	82	61	62	53
Knn1	**92**	78	86	63
Knn5	**92**	79	86	64
Aleph	83	60	59	60
ASPAL	52	59	50	59
Metagol	74	66	60	77
Ilasp	**92**	**86**	**88**	**80**
Mean	73	67	69	60

Results perfectly solved

Approach	goal	legal	next	terminal
True	0	16	0	0
Inertia	0	16	0	0
Mean	32	16	0	12
Knn_1	34	16	0	12
Knn_5	34	22	0	18
Aleph	32	18	4	16
ASPAL	4	18	0	18
Metagol	**48**	28	6	**52**
ILASP	46	**44**	**18**	**52**
Mean	26	22	3	20

Why?

- Diverse games
- New problems each year
- Independent language
- Hard problems

IGGP competition?